ГЛАВА 10

Глава 10 Программа "Arm-Builder"

10.1 Введение

ПО "SyTrack-TOOL" ARMBuilder (программа "**Arm-Builder**") предназначено для разработки автоматизированного рабочего места (АРМа). Включает в себя средства конфигурирования, визуализации мнемосхем, а также клиентский и серверный модули АРМа.

Программа "**Arm-Builder**" может работать в двух режимах:

1. Конфигурационный режим.
2. Рабочий режим.

В конфигурационном режиме программа выполняет роль визуальной среды разработки АРМа.

В рабочем режиме программа выполняет роль самого АРМа.

В **ПО "SyTrack-TOOL" ARMBuilder** предусмотрены следующие варианты лицензирования:

По пользователю, который будет использовать разработанное автоматизированное рабочее место (в зависимости от уровня доступа пользователя меняются права использования программы):

- TM (телемеханик)
- RZA (инженер-РЗА)
- Operator (диспетчер)

По количеству точек ввода/вывода:

- до 100
- до 200
- до 300
- до 400
- от 400

По количеству элементов и серверов:

- 1 клиентское место

10.2 Конфигурационный режим

В конфигурационном режиме программа Arm-Builder представляет собой визуальную среду разработки АРМа.

Запуск
Работа с проектом
Конструирование АРМа
Стандартные компоненты
Редактор статических свойств
Редактор динамических свойств
Тестирование АРМа

10.2.1 Запуск

Запуск программы в конфигурационном режиме производится путём запуска ярлыка Arm-Builder.lnk, лежащего в папке установки, или запуском Arm-Builder.exe из командной строки с использованием параметра /cfg:

Arm-Builder.exe /cfg

Внешний вид главного окна программы показан на рисунке:

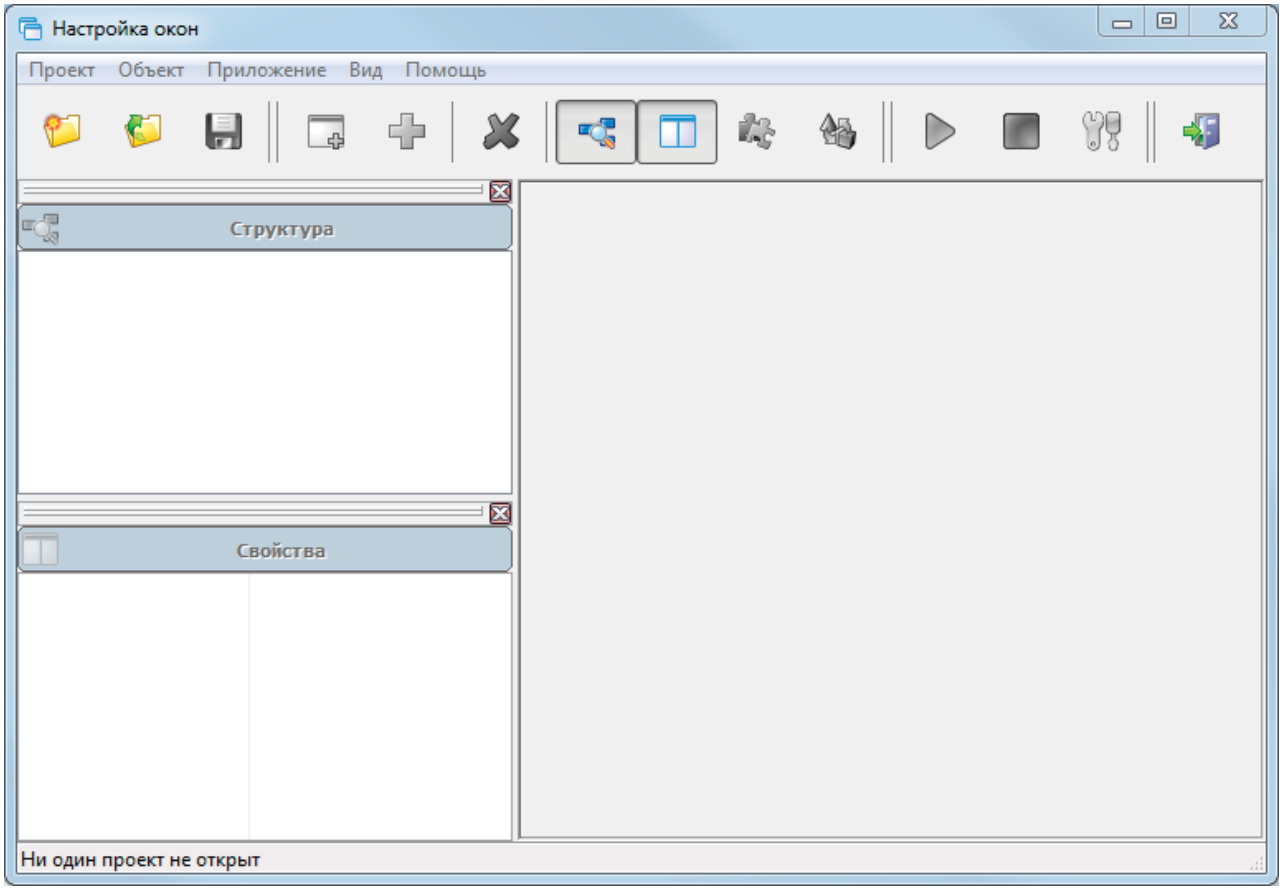


Рис. 1

10.2.2 Работа с проектом

Для построения рабочего приложения используется понятие проекта. Проект состоит из набора окон, компонентов, привязок или других данных, необходимых для работы АРМа.

Основные команды для работы с проектом находятся в меню *Проект*:

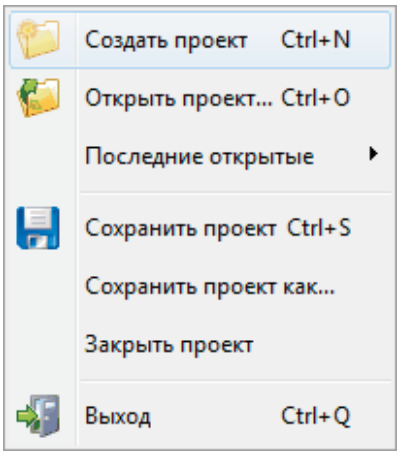


Рис. 2

1. *Создать проект* - будет создан новый проект, содержащий одно пустое окно.
2. *Открыть проект* - появится диалог, в котором нужно выбрать файл ранее сохраненного проекта. Файл проекта имеет расширение **.armx**.
3. *Сохранить проект* - изменения, произведенные в проекте, будут сохранены. Если проект новый и еще ни разу не сохранялся, будет выдан диалог *Сохранить проект как*.
4. *Сохранить проект как* - появится диалог, в котором нужно выбрать директорию для сохранения проекта и ввести имя файла проекта. При сохранении нового проекта рекомендуется всегда создавать новую папку и сохранять файл проекта туда.
5. *Закрыть проект* - текущий проект будет закрыт. Для продолжения работы необходимо создать новый проект или открыть какой-либо уже существующий.

Для удобства три основные операции с проектом -  *Новый проект*,  *Открыть проект*,  *Сохранить проект* - вынесены в виде кнопок в панель инструментов, расположенную под главным меню программы.

Вид главного окна программы после создания нового проекта показан на рисунке, функциональные области выделены разными цветами:

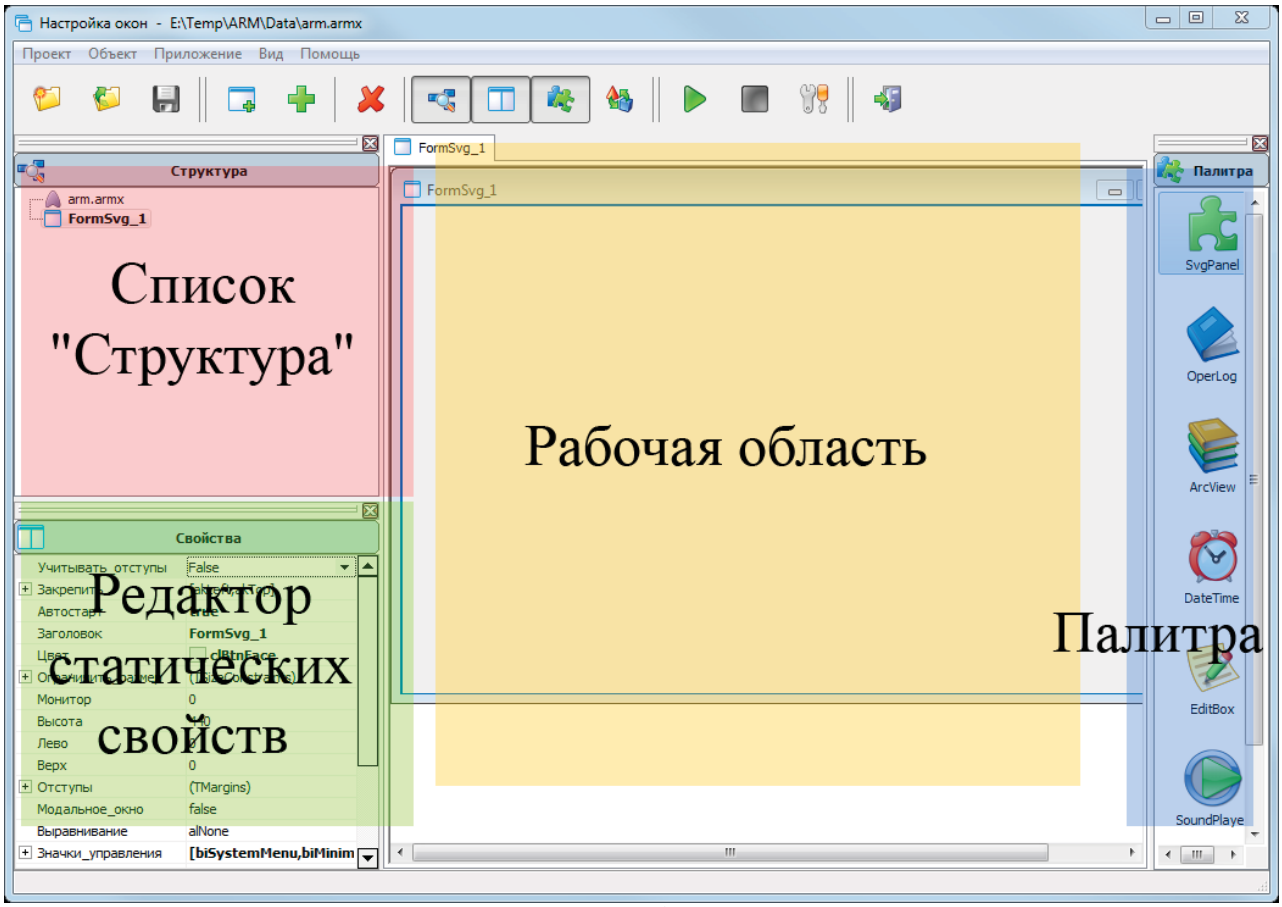


Рис. 3

10.2.3 Конструирование АРМа

Построение рабочего приложения - АРМа - производится путем добавления в проект новых окон и компонентов. Основные команды для конструирования АРМа находятся в меню "Объект":

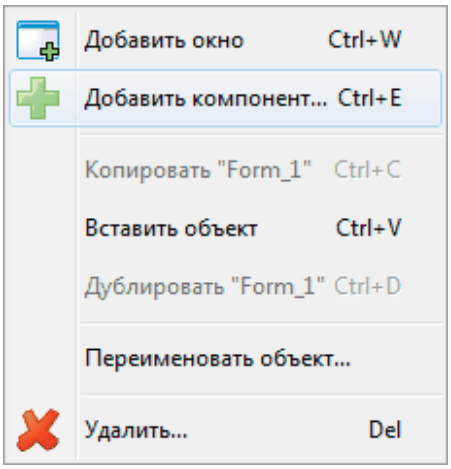


Рис. 4

1. *Добавить окно* - в проекте будет создано новое окно. При этом соответствующий элемент добавится в список "Структура", а само окно будет показано в новой вкладке рабочей области.

Таким образом можно добавлять в проект сколько угодно окон, но главное окно будет всегда только одно. В списке "Структура" главное окно отмечено полужирным шрифтом. Главное окно будет показано при старте приложения, а его закрытие приведет к завершению приложения. Когда новый проект только что создан, в нем одно окно, и оно является главным. При добавлении других окон в проект можно сменить главное окно - для этого щелкните правой кнопкой мыши по нужной записи в списке "Структура" и выберите пункт меню *Сделать окно главным:*

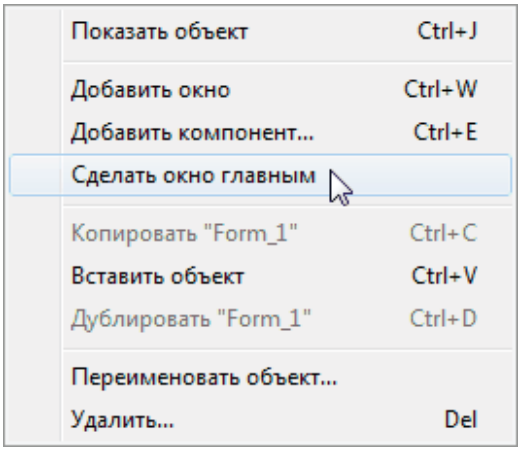


Рис. 5

Добавить окно можно также с помощью кнопки  на панели инструментов под главным меню.

2. *Добавить компонент* - появится диалог, позволяющий выбрать тип добавляемого компонента:

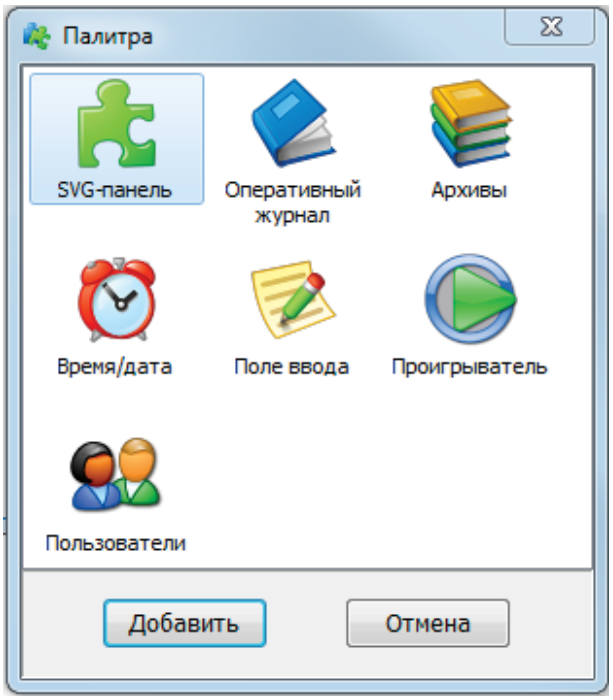


Рис. 6

Этот диалог также можно вызвать с помощью кнопки  на панели инструментов под главным меню или щелкнув правой кнопкой мыши в списке "Структура" и выбрав пункт меню *Добавить компонент*.

Для того чтобы добавить компонент, нужно выбрать его в списке и нажать кнопку "Добавить". Также можно добавить компонент двойным щелчком на соответствующей пиктограмме в списке.

Важно понимать, что компоненты добавляются в одно из окон, созданных в проекте. Перед тем как добавлять компонент, убедитесь, что в списке "Структура" выбрано именно то окно, на которое планируется поместить данный компонент. Если ни одно из окон не выбрано, добавить новый компонент невозможно. Если в списке "Структура" выбран компонент, а не окно, то новый компонент будет добавлен на то окно, на котором находится выбранный компонент.

Предусмотрен еще один метод добавления компонентов в проект. Он заключается в перетаскивании пиктограммы компонента из палитры на рабочую область окна. В стандартной конфигурации программы палитра компонентов расположена справа от рабочей области. Если палитра скрыта, ее можно показать, выбрав пункт меню *Вид -> Палитра*, или воспользовавшись кнопкой  на панели инструментов под главным меню.

После того как компонент помещен на одно из окон проекта, его можно настраивать с помощью следующих операций:

- компонент можно "перетаскивать" мышкой по рабочей области - при этом меняется его местоположение;
- можно изменять размеры компонента, "перетаскивая" мышкой синюю рамку, нарисованную вокруг компонента;
- изменять свойства компонента (такие как Отступы, Выравнивание и т.п.) в панели редактора статических свойств;
- "перекладывать" компонент на другие окна или компоненты - для этого необходимо "перетащить" компонент **правой** кнопкой мыши внутри списка "Структура".

Если одно окно содержит несколько компонентов, то изображения этих компонентов могут перекрываться. Для того чтобы вывести один из компонентов на передний план, надо щелкнуть правой кнопкой мыши по названию компонента в списке "Структура" и выбрать пункт *Показать элемент*:

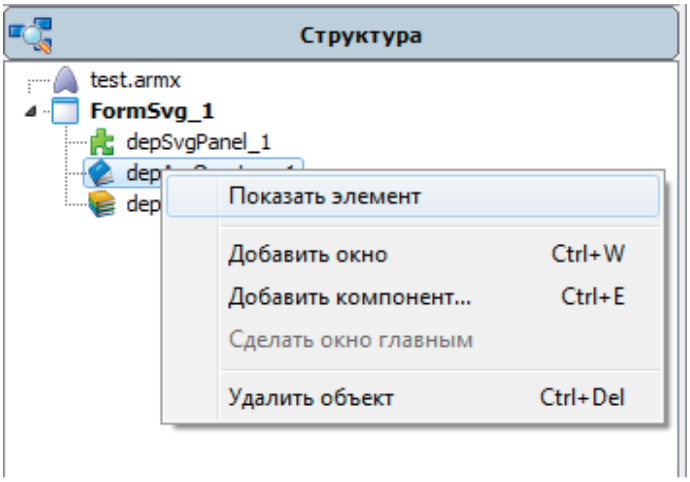


Рис. 7

3. *Копировать - Вставить* - стандартные операции, позволяющие копировать компонент в буфер обмена и вставлять содержимое буфера обмена как в пределах одного окна, так и между различными окнами, даже если они находятся в разных приложениях. *Дублировать* - операция быстрого создания копии компонента в том же окне, где находится выделенный компонент.

4. *Переименовать объект* - появится диалог, в котором можно ввести новое имя для компонента. Если имя содержит недопустимые символы или повторяет уже существующее имя в проекте, то будет выдано сообщение о невозможности переименования.

5. *Удалить* - объект, выбранный в списке "Структура" (это может быть как окно, так и компонент), будет удален из проекта. При этом программа запросит подтверждение на удаление.

Также удалить объект можно с помощью кнопки  на панели инструментов под главным меню или нажав клавишу Del на клавиатуре.

10.2.4 Стандартные компоненты

Для построения рабочего приложения используются компоненты следующих типов:

 SVG-панель. Компонент, позволяющий отображать SVG-файл.

 Оперативный журнал. Компонент, отображающий содержимое оперативного журнала из OPC-модели.

 Архивы. Компонент, отображающий архивные данные.

 Время/дата. Компонент, отображающий текущие время и дату.

 Поле ввода. Компонент, позволяющий вводить данные с помощью клавиатуры и мыши.

 Проигрыватель. Компонент, позволяющий проигрывать звуковые сигналы.

 Пользователи. Компонент, обеспечивающий операции с пользователями при работе АРМа.

SVG-панель.

SVG-панель - компонент, отображающий SVG-файл. SVG - формат векторной графики. Файл SVG (расширение .svg) можно создать, например, в графическом редакторе Inkscape. Основные свойства SVG-панели:

- **Имя_файла** - задается путем выбора SVG-файла при нажатии на кнопку ... в редакторе статических свойств. При задании этого свойства указанный файл будет автоматически загружен на панель.
- **Растягивать** - при установке значения **true**, если размеры рисунка в файле не соответствуют размерам SVG-панели, рисунок будет растянут или сжат до размеров панели.
- **Сохранять_пропорции** - при установке значения **true**, если свойство **Растягивать** также установлено в **true**, то рисунок будет растягиваться и сжиматься таким образом, чтобы отношение размеров сторон сохранялось.
- **Масштабирование** - при установке значения **true** изображение, загруженное на панель, можно масштабировать. Плавное увеличение и уменьшение масштаба производится с помощью колесика мыши, перемещение по изображению осуществляется путем "перетаскивания" левой кнопкой мыши. Увеличить определенную область изображения можно, выделив ее правой кнопкой мыши. Быстрое увеличение изображения производится путем двойного нажатия левой кнопки мыши. Возврат к исходному масштабу производится с помощью одиночного нажатия правой кнопки мыши в любой точке

изображения.

Оперативный журнал.

Оперативный журнал - компонент, автоматически отображающий содержимое оперативного журнала, сформированного в OPC-модели. Поскольку данные оперативного журнала, как и все архивные данные, находятся в хранилище, указанном в настройках модели в программе WinDeCont, то для правильной работы компонента "Оперативный журнал" в проекте АРМа необходимо указать, к какому хранилищу должно производиться подключение. Обычно в OPC-модели создают специальное текстовое поле, содержащее псевдоним хранилища для записи архивных данных. В проекте АРМа достаточно указать имя этого поля - для этого надо выбрать в списке "Структура" элемент с символом  и названием проекта и в редакторе статических свойств ввести необходимое OPC-имя в поле **Хранилище--OPC-имя** (например, ArchName).

Если имя OPC-элемента, содержащего псевдоним хранилища, задано правильно, то при старте АРМ подключится к хранилищу и будет автоматически вычитывать данные оперативного журнала. Свойства компонента в основном предназначены для настройки отображения данных. Основные свойства компонента:

- **Число_записей** - число строк в оперативном журнале. Например, если число записей равно 10, то будут показываться 10 последних записей оперативного журнала.
- **Столбец_"<...>"** - составные свойства, определяющее характеристики столбцов оперативного журнала. Для редактирования свойства надо нажать на значок  слева от названия свойства. В раскрывающемся списке можно настроить заголовок столбца, его номер в журнале, ширину столбца и видимость (показывать данный столбец или скрыть).

Если раскрыть свойство **Столбец_"Важность"**, то можно видеть вложенное свойство **Интервалы**. Используя **Интервалы**, можно настроить вид записи оперативного журнала в зависимости от значения важности. Нажав на кнопку  , можно вызвать редактор интервалов. Кнопка  добавляет интервал, параметры которого в свою очередь можно настроить в редакторе статических свойств.

Архивы.

Архивы - компонент, отображающий архивные данные из хранилища. Для работы компонента необходимо указать, к какому хранилищу подсоединяться. Это делается по схеме, описанной выше (компонент "Оперативный журнал"). Основные свойства архивного компонента:

- **Включить** - при установке значения **true** компонент начинает работу. По умолчанию значение **false**, т.е. компонент находится в неактивном состоянии.
- **Начальная_дата, Конечная_дата** - интервал времени, по которому показываются архивы. Дата и время записываются в формате **дд.мм.гггг чч:мм:сс**.
- **Страница** - номер страницы, которая будет показана по умолчанию (0 - Аналоги, 1 - Счетчики, 2 - Дискреты, 3 - Аналоги по изменению, 4 - Счетчики по изменению, 5 - Оперативный журнал, 6 - Графики).
- Также можно настроить, какие страницы показывать, и на каких страницах обновлять информацию с помощью свойств **Обновлять_счетчики** и т.п., **Показывать_аналоги** и т.п.

Время/дата.

Дата/время - компонент, отображающий текущие время и дату. В качестве текущих берутся значения системных времени и даты. Значение, отображаемое компонентом, автоматически обновляется. Основные свойства компонента:

- **Формат_даты_времени** - задает формат, в котором будут отображаться данные. Пример: **dd.mm.yyyy hh:mm:ss**. Такой формат соответствует следующей записи - 25.09.2010 12:55:34.

Поле ввода.

Поле ввода - компонент, позволяющий вводить данные с клавиатуры.

Свойство **Тип** определяет, для ввода каких данных предназначен компонент:

1. **etEdit** - ввод текста. Основные свойства:

Подстраивать_размер - при установке значения **true** поле ввода будет изменять размер в зависимости от размера содержащегося в нем текста.

Выравнивание_текста - определяет положение текста внутри поля ввода: слева (**taLeftJustify**), справа (**taRightJustify**), в центре (**taCenter**).

Только_чтение - при установке значения **true** редактирование содержимого поля ввода невозможно.

Текст - содержимое поля ввода.

2. **etSpinEdit** - ввод числовых данных (целых чисел и чисел с плавающей точкой). Основные свойства:

Тип_величины - определяет, какие числа будут вводиться в поле: целые (**vkInteger**), с плавающей точкой (**vkFloat**), в шестнадцатеричном формате (**vkHex**).

Значение - содержимое поля ввода (число).

Проверять_на_минимум, **Проверять_на_максимум** - при установке значения **true** будет производиться проверка содержимого поля ввода на минимум или на максимум.

Минимум, **Максимум** - значения, определяющие минимальное и максимальное значения содержимого поля ввода.

Подсвечивать_некорректные_значения - при установке значения **true** при вводе числа, выходящего за пределы интервала (**Минимум** ... **Максимум**), поле ввода будет обведено красной рамкой.

Также для данного типа компонента предусмотрено свойство **cmd_ValidateMessage**, при установке значения которого в true (в OPC-привязке) будет явно произведена проверка содержимого на минимум и максимум и в случае нарушения диапазона будет выдано сообщение.

3. **etDateEdit** - ввод даты. Основные свойства:

Дата - содержимое поля ввода. Может содержать дату (**dd.mm.yyyy**) или дату и время (**dd.mm.yyyy hh:mm:ss**).

Свойства - составное свойство. Содержит следующие записи:

Выравнивание_текста - определяет положение содержимого внутри поля ввода:

Горизонтальное - слева (**taLeftJustify**), справа (**taRightJustify**), в центре (**taCenter**).

Вертикальное - сверху (**taTopJustify**), снизу (**taBottomJustify**), в центре (**taVCenter**).

Тип_данных - определяет, будет ли в поле ввода содержаться только дата (**ckDate**), или же дата и время (**ckDateTime**).

Только_чтение - при установке значения **true** редактирование содержимого поля ввода невозможно.

Также для данного типа компонента предусмотрены свойства **cmd_SetToday** (при установке значения **true** содержимое автоматически заполняется текущей датой) и **cmd_ShiftDay** (при установке ненулевого значения содержимое автоматически сдвигается на указанное число дней. Например: если содержимое поля ввода было 12.03.2010, то при установке значения **cmd_ShiftDay** в 4 содержимое станет равно 16.03.2010). Данные свойства используются при задании OPC-привязок в редакторе динамических свойств.

4. **etTimeEdit** - ввод времени.

Время - содержимое поля ввода.

Свойства - составное свойство. Содержит следующие записи:

Выравнивание_текста - определяет положение содержимого внутри поля ввода:

Горизонтальное - слева (**taLeftJustify**), справа (**taRightJustify**), в центре (**taCenter**).

Вертикальное - сверху (**taTopJustify**), снизу (**taBottomJustify**), в центре (**taVCenter**).

Корректировка - при установке значения **true** время, вводимое в поле ввода, будет автоматически преобразовано к 24-часовому формату.

Шаг - определяет, на сколько будет изменяться содержимое поля ввода при нажатии на кнопки увеличения и уменьшения.

Кнопки - составное свойство, определяет вид кнопок увеличения и уменьшения.

Формат_времени - определяет формат, в котором будет отображаться время: только часы (**tfHour**), часы и минуты (**tfHourMin**), часы, минуты и секунды (**tfHourMinSec**).

Только_чтение - при установке значения **true** редактирование содержимого поля ввода невозможно.

Проигрыватель.

Проигрыватель - компонент, позволяющий воспроизводить звуковые сигналы в приложении. Основные свойства:

Имя_звукового_файла - определяет путь к файлу, который будет использоваться в качестве звукового. Диалог выбора файла вызывается нажатием кнопки ... в редакторе статических свойств. Файл должен иметь формат WAV. Если имя файла не указано, то будет использоваться встроенный сигнал.

Также, для данного компонента предусмотрено свойство **cmd_Play** - при установке его значения в **true** начинается воспроизведение звука, при установке значения **false** воспроизведение останавливается. Данное свойство используется при задании OPC-привязок в редакторе динамических свойств.

Пользователи.

Пользователи - компонент, обеспечивающий операции с пользователями: вход, выход, добавление новых пользователей, изменение параметров пользователя. Основные свойства:

OPC-имя - имя элемента типа tUsers в OPC-модели.

Также, для данного компонента предусмотрено свойство **cmd_Login**, которое используется при задании OPC-привязок в редакторе динамических свойств. При задании значения этого свойства **true** вызывается стандартный диалог работы с пользователями.

10.2.5 Редактор статических свойств

Для настройки таких свойств, как внешний вид компонента, его расположение, размеры и стандартное поведение, используется **редактор статических свойств** (панель "Свойства"). Редактор можно показать, выбрав пункт меню *Вид -> Статические свойства* или с помощью кнопки  на панели инструментов под главным меню.

Внешний вид панели редактора статических свойств показан на рисунке:

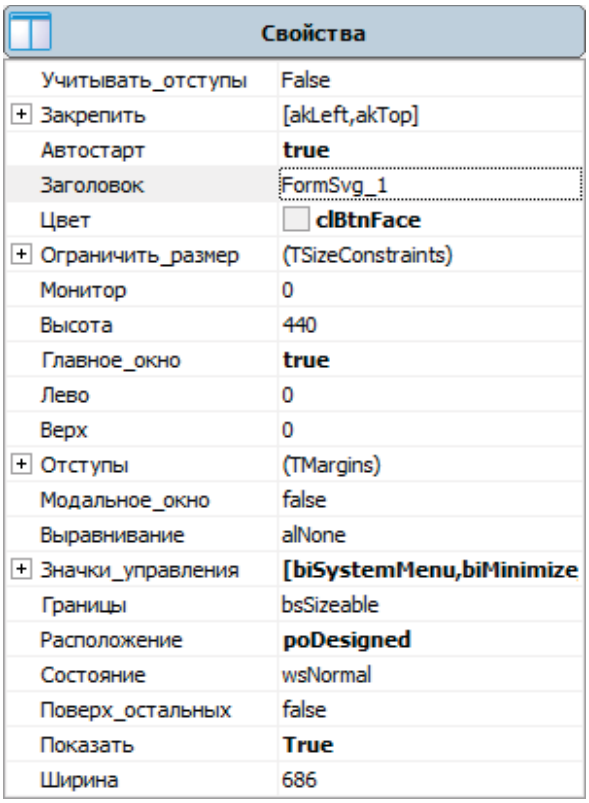


Рис. 8

Редактор статических свойств показывает свойства того объекта, который выбран в списке "Структура". Левая колонка редактора отображает названия свойств, а правая - их значения в данный момент. Свойства могут простыми и составными. Значения простых свойств редактируются одним из следующих способов (в зависимости от типа свойства):

- ввод текста в поле;
- изменение значения с помощью кнопок со стрелками "вверх" и "вниз";
- выбор значения из раскрывающегося списка;
- вызов специального редактора для свойства (например, диалога выбора файла).

Составные свойства отмечены в редакторе значком "+" слева от названия свойства. При нажатии на этот значок

раскрывается список, показывающий содержимое составного свойства, например:

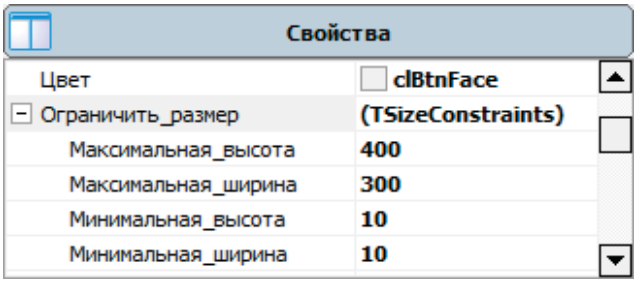


Рис. 9

Существуют стандартные статические свойства, которые есть практически у всех компонентов. К таким свойствам относятся:

Выравнивание - задает положение компонента относительно окна (или относительно другого компонента), на котором лежит данный компонент. Может принимать значения **alNone** (нет выравнивания), **alLeft** (выравнивание по левому краю), alRight (выравнивание по правому краю), alTop (выравнивание по верхнему краю), alBottom (выравнивание по нижнему краю) и alClient (компонент занимает все предоставленное пространство).

Учитывать_отступы - определяет, будут ли учитываться поля при выравнивании.

Отступы - определяет значения полей при выравнивании компонента (имеет смысл, только если свойство **Учитывать_отступы** имеет значение **true**): левого, правого, нижнего и верхнего.

Лево, **Верх** - определяет координаты левого верхнего угла компонента.

Ширина, **Высота** - размеры компонента.

Показать - определяет видимость компонента. При установке значения **true** компонент будет виден, при установке значения **false** - скрыт.

10.2.6 Редактор динамических свойств

Для задания поведения компонентов при работе АРМа используется **редактор динамических свойств**. Этот редактор позволяет создавать, изменять и удалять так называемые "OPC-привязки", т.е. настраивать взаимодействие АРМа с OPC-моделью. Окно редактора вызывается одним из трех способов:

- пункт меню *Вид -> Динамические свойства*;
- кнопка  на панели инструментов под главным меню;
- комбинация клавиш Shift+F3.

10.2.6.1 Редактор динамизации компонентов

Для вызова редактора динамизации компонентов на любом из расположенных на форме компонентов кликните правой клавишей мыши и выберите пункт меню OPC Design Form... или нажмите комбинацию клавиш <Shift+F3>.

Дерево компонентов построено по принципу: "кто Parent компонента, тот и его владелец", для SVG объектов под Parent`ом подразумевается SVG группа.

Здесь и далее условимся VCL компонент или SVG объект называть просто объектом для простоты изложения.

Выбрав в дереве объект, можно для него создать привязки к элементам OPC сервера. Для этого нужно нажать кнопку с зеленым плюсом. В результате этого появится окно выбора свойства или действия, которое должно совершиться при выполнении заданного условия. После выбора свойства или действия будет создана привязка к данному объекту, для которой потребуется возможно задать состояния сравнения и привязки к свойствам объекта. При привязки к событиям потребуется задать OPC элемент приемник для записи и OPC элемент источник или константу.

Для визуальных объектов есть возможность управлять их видимостью в зависимости от наличия заданного одного или нескольких OPC элементов в сервере:

Как у Parent - если флажок установлен, то настройки видимости будут браться из Parent`а данного объекта, в противном случае будет анализироваться флажок Виден всегда.

Виден всегда - если флажок установлен, то объект будет виден независимо от наличия OPC элемента на сервере, в противном случае объект будет скрыт, если хотя бы один OPC элемент отсутствует на сервере.

OPC имя привязки(компонента) можно ввести вручную в соответствующем поле или, нажав кнопку  для вызова диалога выбора OPC имени привязки.

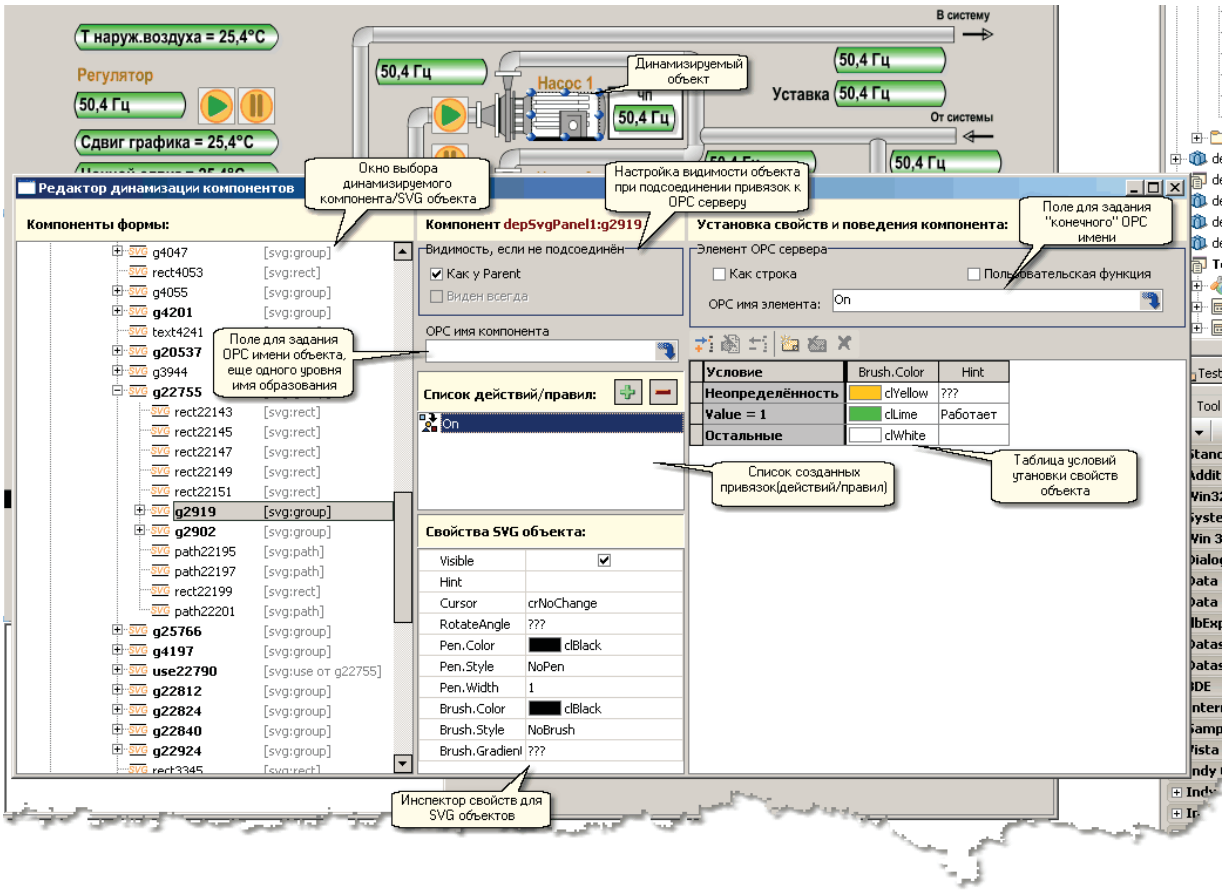


Рис. 10

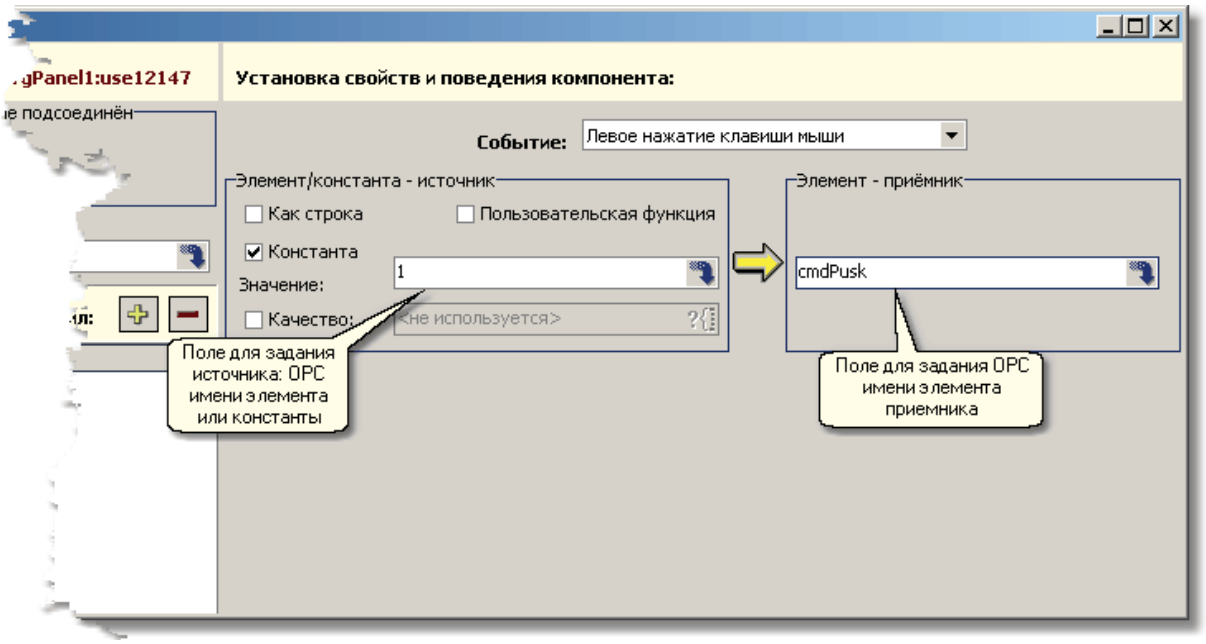


Рис. 11

10.2.6.1.1 Привязка к свойствам объекта

Для любого VCL компонента и SVG объекта могут быть динамизированы:

- любые его публичные свойства (свойства, которые показывает Object Inspector в среде программирования CodeGear RAD Studio), для SVG объекта набор основных свойств(цвет и стиль заливки, обводки, хинт, курсор, видимость, для текста - шрифт, содержание);
- события нажатия клавиш мыши (левое нажатие, правое нажатие, двойное нажатие левой клавиши мыши, двойное нажатие правой клавиши мыши) для записи значений в элементы OPC сервера;
- специальные свойства (мигание).

10.2.6.1.1.1 Привязка к публичным свойствам

Добавление и удаление динамизируемых свойств для объекта происходит с помощью соответствующих кнопок, описанных на рисунке. При этом создается столбец с заголовком выбранного свойства. В ячейках данного столбца для каждого состояния сравнения можно задать свое значение свойства.Каждое значение свойства редактируется в соответствии с его базовым типом.

Для свойств с базовым типом UString(LString), Float и Integer предусмотрено форматирование значений. Форматирование значений аналогично работе функции Format или FormatDateTime (для типа TDateTime). Например, если динамизируется свойство Caption у компонента TLabel, то в строке столбца "Caption" можно записать "ПУ %d". В результате если будет установлено данное состояние при сравнении, то вместо %d ,будет выведено значение (Value) элемента модели. В отличии от функции Format использование маски "%s" позволяет вывести любое значение вне зависимости от типа Value элемента модели.

Для получения элемента OPC сервера в виде строки установите галочку "Как строка". В данном случае если со значением элемента в OPC сервере сопоставлена строка, то Value примет строковое значение.

Существует возможность вычислять значения Value и Quality, которые передаются для сравнения на основе реальных значений из OPC сервера, в пользовательской функции.

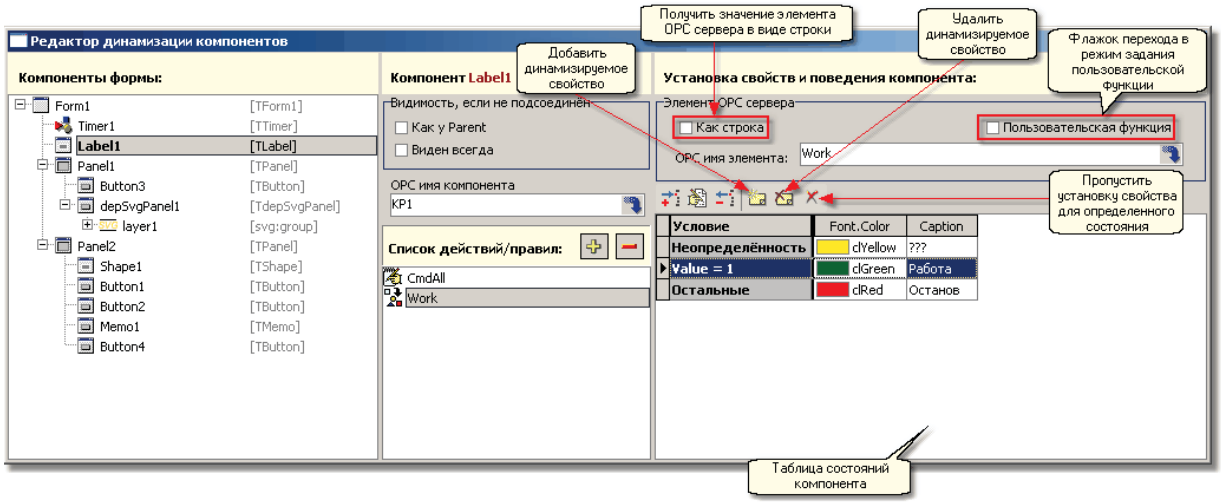


Рис. 12

При добавлении новой привязки в редакторе состояний компонентов или при редактировании привязки можно выбрать свойства компонента для динамизации. Возможно выбрать сразу нескольких свойств, для этого при выделении каждого свойства удерживайте клавишу Ctrl.

Описание свойств VCL компонентов можно прочитать в справке к CodeGear RAD Studio.

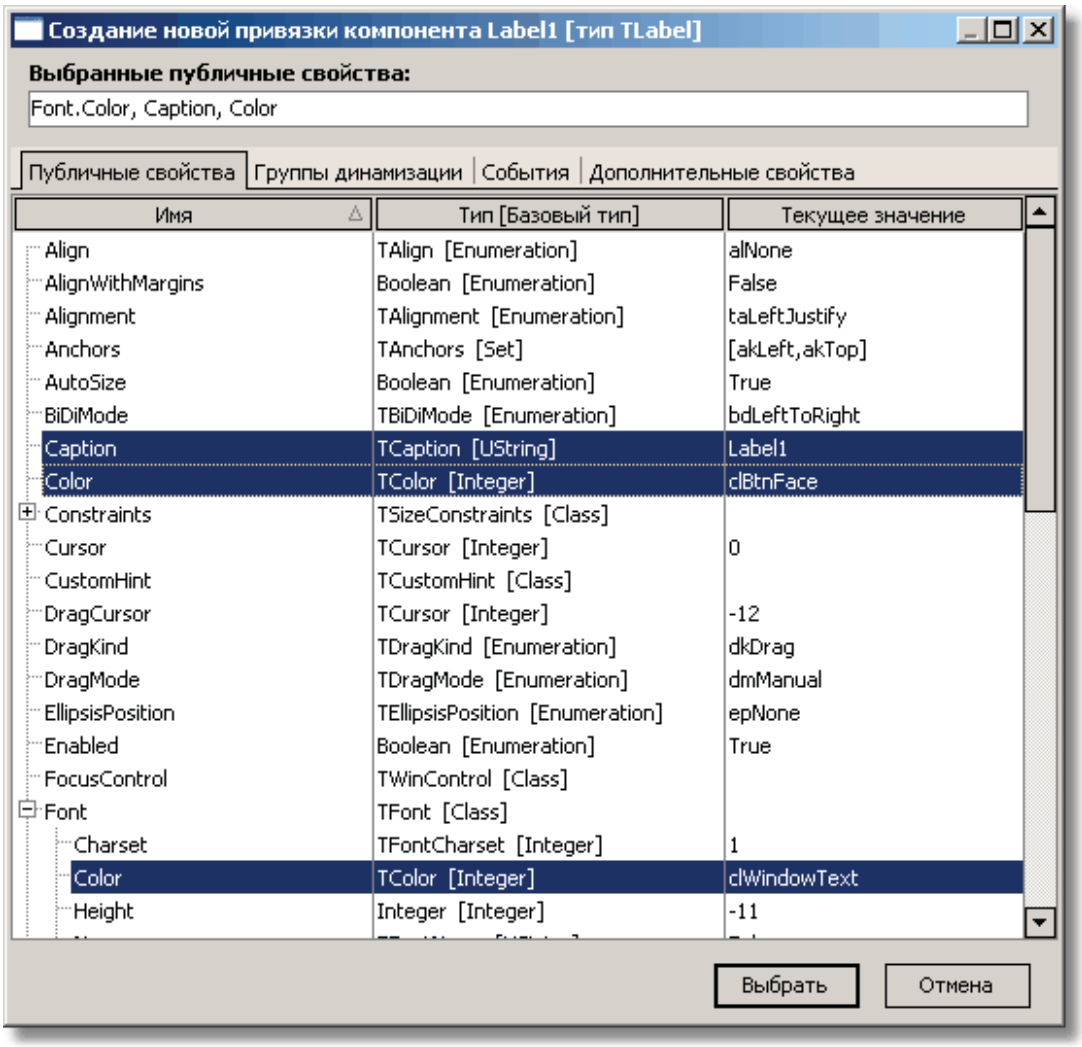


Рис. 13

10.2.6.1.1.2 Добавление состояний сравнения

Для задания поведения компонента в зависимости от значения элемента модели требуется создать состояния сравнения (условия). При каждом изменении значения элемента модели (Value или Quality) происходит поиск сверху вниз истинного состояния. При его нахождении происходит установка соответствующих динамизируемых свойств в выбранной строке.

Для добавления, изменения и удаления состояний сравнения можно воспользоваться соответствующими кнопками, обозначенными на рисунке.

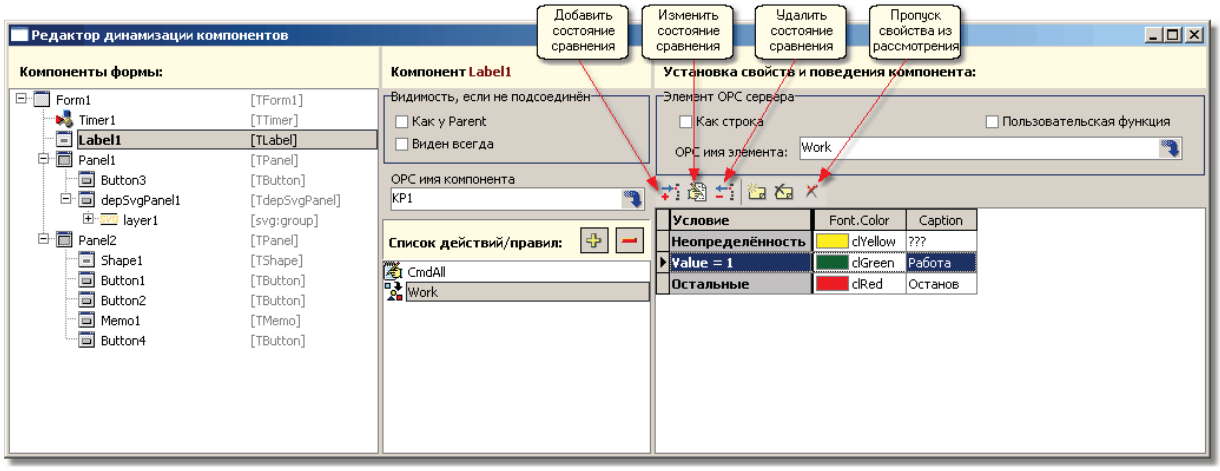


Рис. 14

Добавление или изменении состояния производится в окне "Добавление нового условия"("Изменение условия"). В этом окне требуется задать условие сравнения с значением Value (возможно использование знаков "=" (равно), "!=" (не равно),"<' (меньше), ">" (больше)) или с каким либо битом Quality (выбор какого-либо бита Quality означает проверку установки этого бита в Quality изменившегося элемента OPC сервера).

При изменении значения привязанного элемента модели (изменилось его Value или Quality) в процессе выполнения программы происходит поиск сверху вниз истинного выражения. Обязательное состояние "Остальные" (всегда последнее), которое нельзя удалить из рассмотрения. При найденном истинном утверждении происходит установка всех свойств описанных в данной строке.

Существует возможность пропустить свойство из рассмотрения в определённой строке. При этом если в найденном истинном утверждении какое-либо свойство пропущено из рассмотрения, поиск по условиям продолжится для установки пропущенных свойств.

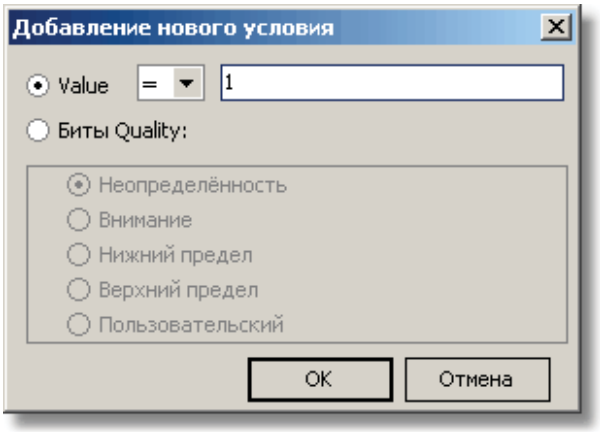


Рис. 15

10.2.6.1.1.3 Привязка к событиям для записи

Для визуальных компонентов возможна привязка к действиям пользователя. В данной версии доступны четыре возможные события:

- левое нажатие клавиши мыши;
- правое нажатие клавиши мыши;

- двойное нажатие левой клавиши мыши;
- двойное нажатие правой клавиши мыши;

По данным событиям возможна запись значения в элемент модели, а также вызов пользовательской функции перед записью. Если компонент невидим, то данные события не могут произойти.

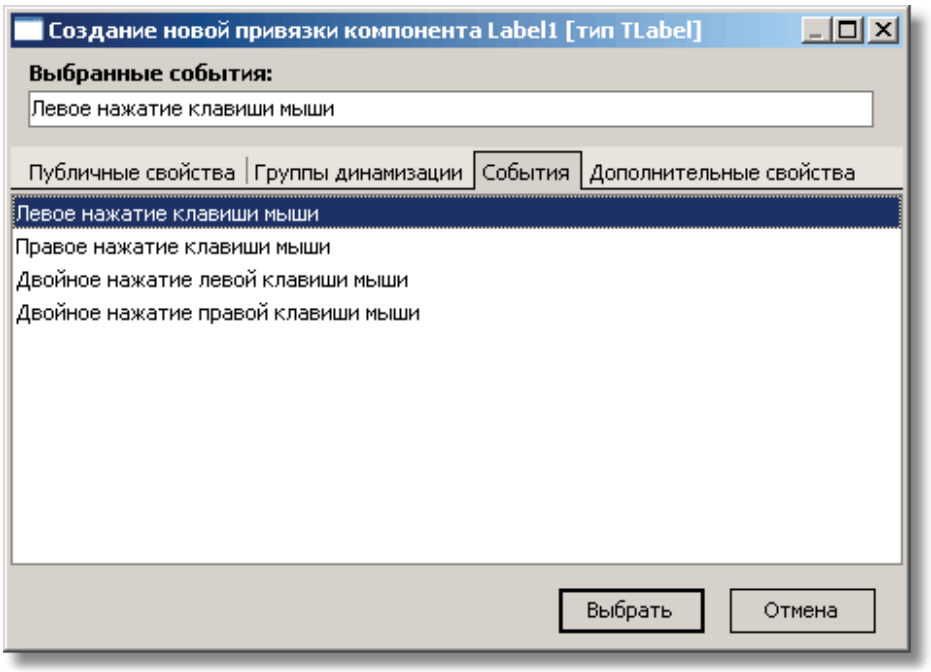


Рис. 16

При редактировании привязки к событиям для записи требуется указать OPC элемент-приемник и OPC элемент-источник или константу. Возможен вызов пользовательской функции для источника, в которой можно отменить запись, изменить записываемое значение.

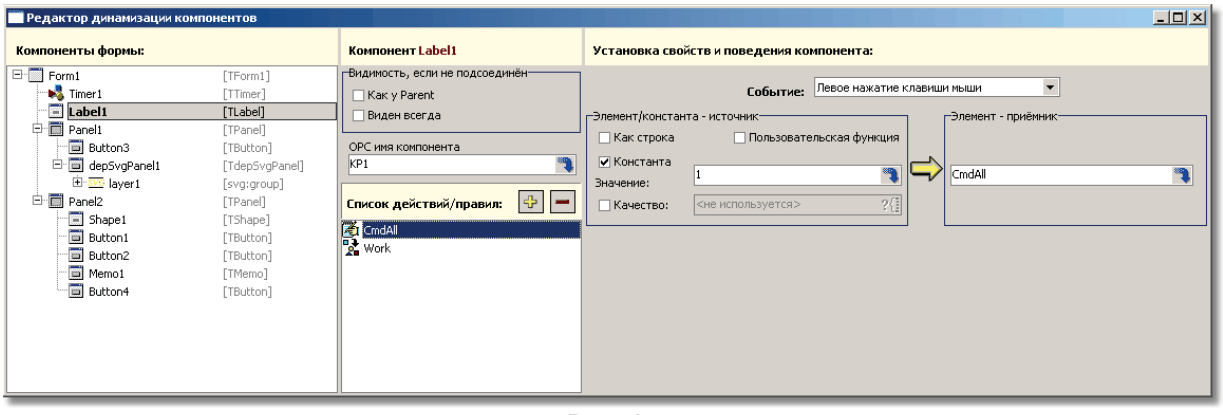


Рис. 17

10.2.6.1.1.4 Привязка к специальным свойствам

В данной версии возможна привязка только к свойству "Мигание" (только для визуальных компонентов). Под миганием компонента понимается периодичное изменение набора его свойств.

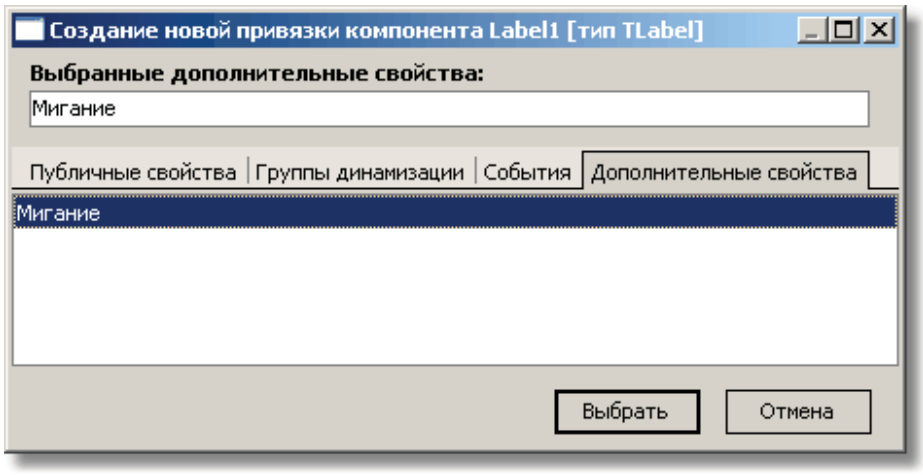


Рис. 18

Специальное свойство - мигание позволяет установить поочередное изменение набора публичных свойств объекта во времени. Свойство мигание может быть добавлено в любой момент создания привязки к публичным свойствам. Столбец "Мигание" будет всегда установлен первым в наборе столбцов свойств. Возможны четыре значения свойства "Мигание":

· "Нет" - не мигать;

· "Медленное" - медленная скорость мигания;

· "Среднее" - средняя скорость мигания;

· "Быстрое" - высокая скорость мигания.

При установлении состояния "мигать" (не в состояние "Нет") компонент переходит в режим мигания. В этом режиме происходит поочередная смена свойств: 1/3 цикла компонент в состоянии "Мигание" - устанавливаются все значения свойств для установившегося состояния; 2/3 цикла компонент - в основном состоянии. В основном состоянии, проходя по всем привязкам, устанавливаются лишь те свойства, которые удовлетворяют условиям и в которых свойство "Мигание" установлено в "Нет".

Например, для Label1, если элемент OPC сервера станет неопределенным, то компонент перейдет в состояние "мигания". При этом мигать он будет в желтый цвет из зеленого или красного в зависимости от того какое значение принимало Value до неопределенности. Caption метки при мигании не будет меняться, т.к. это свойство пропущено в состоянии "мигать". Состояние "Value = 2" никак не участвует в мигании, т.к. для мигания оно является пропущенным.

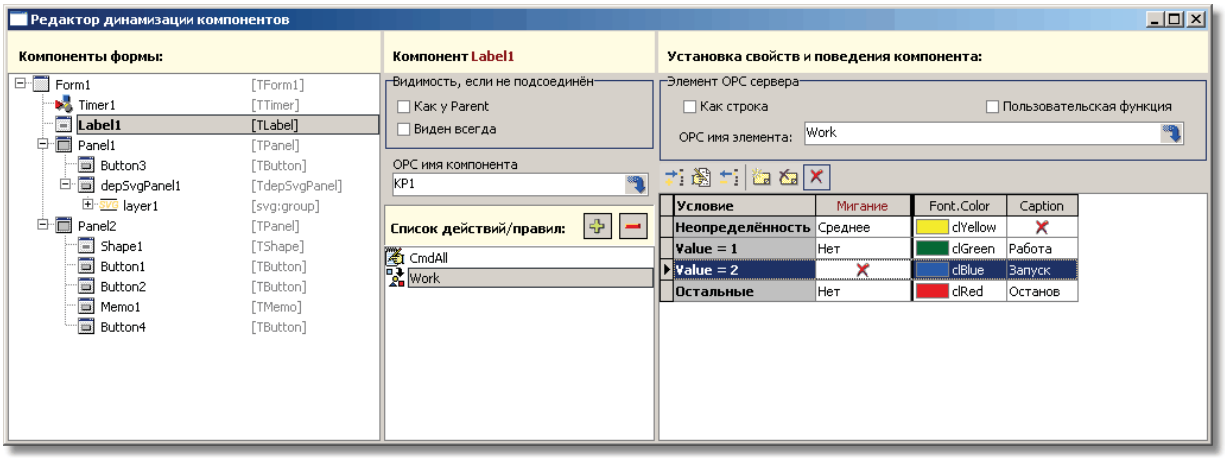


Рис. 19

10.2.6.1.2 Выбор OPC имени привязки

Данный диалог предназначен для выбора OPC имени привязки. Для каждого OPC псевдонима в проекте создается своя закладка с иерархическим представлением OPC элементов. Рисунок в виде кружка на каждой закладке показывает состояние соединения с OPC сервером:

 - соединение с OPC сервером установлено;

 - соединение с OPC сервером по каким-либо причинам не удается установить, нужно проверить настройки OPC псевдонима и доступность OPC сервера.

Соединение с OPC сервером устанавливается автоматически, если в OPC псевдониме указано автоматически устанавливать соединение, иначе соединение потребуется устанавливать вручную, нажав кнопку  . Для отсоединения потребуется нажать кнопку  .

Выбор элемента OPC сервера можно производить в трех режимах, переключаясь соответствующими кнопками:

.  Авто имя относительно Parent. Из имени будет урезана начальная часть, составляющая OPC-имя Parent`а элемента.

.  Имя относительно заданного элемента. Выбрав этот режим требуется также указать, нажав кнопку  , OPC элемент, относительно которого будет производиться имя образование.

.  Абсолютное (полное) имя элемента.

Все перечисленные действия можно произвести также выбрав соответствующий пункт меню или по "горячим" клавишам.

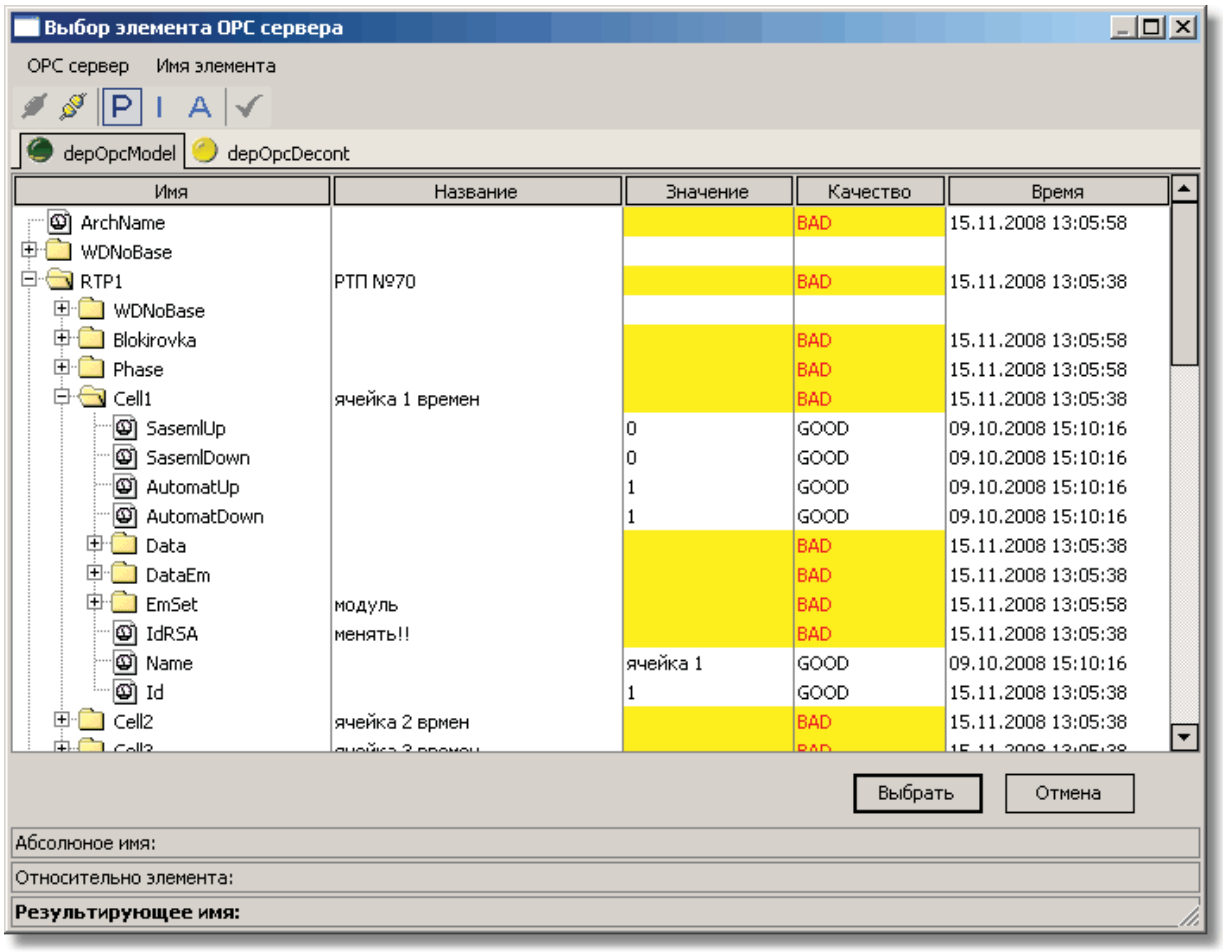


Рис. 20

10.2.6.1.3 Пользовательские функции

Пользовательские функции позволяют проанализировать значения OPC тэгов до выполнения состояний сравнения или до выполнения записи в OPC тэг. Пользовательскую функцию можно задать как для привязки к публичным свойствам компонента, так и в привязке к событиям для записи. При написании кода пользовательской функции, требуется подключить заголовочный файл depOpc.hpp для C++Builder, для Delphi в секцию uses добавить подключение модуля depOpc.

При установке флажка пользовательской функции кнопка  в поле выбора элемента OPC сервера заменяется на кнопку  редактирования вызова редактора пользовательской функции.

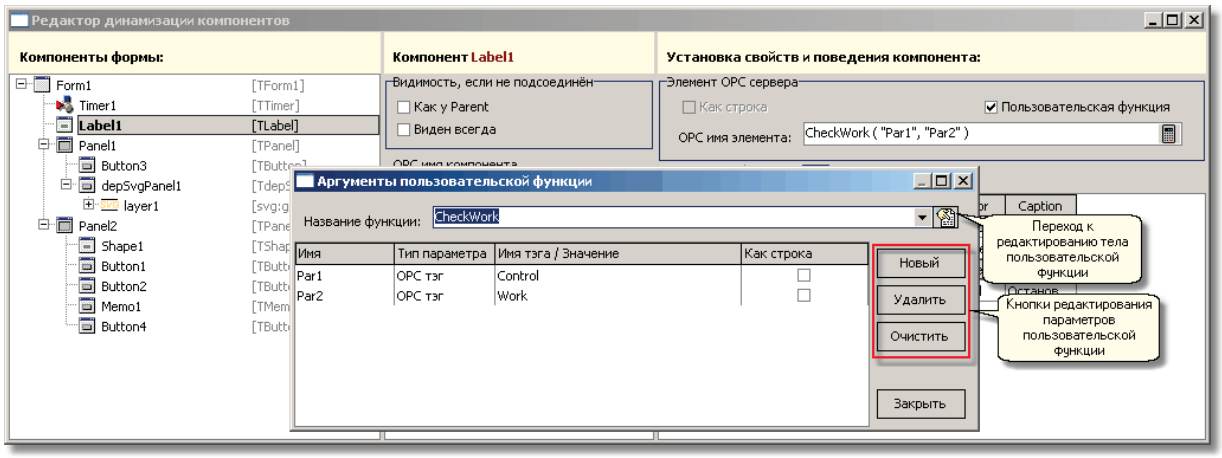


Рис. 21

В редакторе аргументов пользовательской функции возможно добавление, удаление и очистка всех аргументов. Функция будет вызываться каждый раз, когда происходит инициализация или изменение(меняется Value или Quality OPC элемента) хотя бы одного параметра с типом OPC тэг. Для каждого аргумента указывается:

- **Имя** - название параметра, по которому можно к данному аргументу обращаться в коде пользовательской функции;
- **Тип параметра** может принимать два значения: OPC тэг(в следующем поле указывается относительное или абсолютное название элемента OPC сервера) и Константа(в следующем поле указывается строковая константа);
- **Имя тэга / Значение** - название элемента OPC сервера или константа в зависимости от поля Тип параметра;
- **Как строка** - получать значение OPC элемента в виде строки(только для типа параметра - OPC тэг)

Возможно создание двух типов пользовательских функций:

- TdepOpcUserSetFuncNotify - создается в привязках к публичным свойствам;
- TdepOpcUserWriteFuncNotify - создается в привязках к событиям для записи.

```
TdepOpcUserSetFuncNotify = procedure (const aFullRelativeOpcName: string; aListParUserFunc: TdepOpcListParUserFunc;
        aIndexParChange: Integer; aAllConnected: Boolean; aAllChanged: Boolean;
        var aValue: Variant; var aQuality: TdepOpcQuality; var aSetProperties: Boolean) of object;

TdepOpcUserWriteFuncNotify = procedure (const aFullRelativeOpcName: string; aListParUserFunc: TdepOpcListParUserFunc;
        aIndexParChange: Integer; aAllConnected: Boolean; aAllChanged: Boolean;
        var aWriteValue: Variant; var aQualitySpecified: Boolean; var aWriteQuality: TdepOpcQuality;
        var aWriting: Boolean) of object;
```

Описание аргументов:

aFullRelativeOpcName - полное имя привязки, полученное в результате имя образования;

aListParUserFunc - список объектов класса TdepOpcParUserFunc, в котором находится описание параметров, заданных в редакторе аргументов пользовательской функции, и их текущие значения. Список представляет собой типизированный список класса TList;

aIndexParChange - индекс параметра в списке aListParUserFunc, для которого произошло изменение значения(Value или Quality).

aAllConnected - флаг подсоединения(наличия) всех параметров функции с типом OPC тэг в OPC сервере.

aAllChanged - флаг установления значения для всех параметров функции с типом OPC тэг в OPC сервере.

только для TdepOpcUserSetFuncNotify:

aValue - значение Value, выходной параметр, который будет анализироваться в привязке к публичным свойствам в соответствии с условиями состояний;

aQuality - значение Quality, выходной параметр, который будет анализироваться в привязке к публичным свойствам в соответствии с условиями состояний;

aSetProperties - выходной параметр, указывающий требуется(True) или нет(False) после вызова функции анализировать значения Value и Quality в привязке к публичным свойствам в соответствии с условиями состояний;

только для TdepOpcUserWriteFuncNotify:

aWriteValue - значение Value для записи в элемент OPC сервера;

aQualitySpecified - флаг, указывающий требуется(True) или нет(False) записывать значение Quality в элемент OPC сервера;

aWriteQuality - значение Quality для записи в элемент OPC сервера;

aWriting - флаг, указывающий требуется(True) или нет(False) производить запись Value и Quality;

Описание класса, используемого для элементов списка TdepOpcListParUserFunc:

```
TdepOpcParUserFunc = class
public
    property Name: string read fName; //название параметра, заданное в редакторе аргументов пользовательской функции
    property ConstOrOpcName: string read fConstOrOpcName; //константа(TypeSource = tsConst) в виде строки или имя элемента OPC сервера(TypeSource = tsOpcTag)
    property TypeSource: TdepOpcTypeSourceUserFunc read fTypeSource; //указывает, что хранится в поле ConstOrOpcName, если равно tsConst - константа, tsOpcTag - имя элемента OPC сервера
    property IsAsString: Boolean read fIsAsString; //брать значение элемента OPC сервера в виде строки(только для TypeSource = tsOpcTag)
    property Value: Variant read fValue; //значение элемента OPC сервера(только для TypeSource = tsOpcTag)
    property Quality: TdepOpcQuality read fQuality; //качество элемента OPC сервера(только для TypeSource = tsOpcTag)
    property Connected: Boolean read fConnected; //флаг подсоединения элемента OPC сервера(только для TypeSource = tsOpcTag)
end;
```

10.2.7 Тестирование АРМа

После того как создан проект, добавлены нужные окна и компоненты, настроены их статические и динамические свойства, получившийся АРМ можно запустить на выполнение из визуальной среды.

Проще всего сделать это с помощью кнопки  на панели инструментов под главным меню или нажав клавишу F9. Запускается всегда последняя сохраненная версия. Если последние изменения не были сохранены, то будет выдан диалог с предложением сохранить проект. Если отказаться от сохранения, то будет запущена версия АРМа без последних изменений. Если проект был недавно создан и еще не сохранялся, то его запустить на выполнение невозможно - сначала сохраните проект в определенную директорию на жестком диске!

Если приложение не запускается или если вместо главного окна на экране появляется предупреждение, то какие-то настройки сделаны неправильно. В первую очередь необходимо проверить, есть ли главное окно в проекте и установлено ли его свойство **Автостарт** (т.е. показывать при старте приложения) в **true**.

Запущенное приложение можно принудительно завершить с помощью кнопки  на панели инструментов под главным меню или нажав клавишу F10.

Предусмотрена возможность запуска приложения с параметрами командной строки. Для этого надо вызвать диалог выбора параметров, воспользовавшись кнопкой  на панели инструментов под главным меню.

В данной версии программы Arm-Builder существует возможность запуска АРМа с подключением к OPC-серверу, запущенному на удаленном компьютере. Для этого надо в качестве параметров командной строки ввести */server:[host_ip]* или */server:[host_name]*, где вместо *[host_ip]* надо указать ip-адрес удаленного компьютера, или вместо *[host_name]* указать сетевой псевдоним компьютера. Пример приведен на рисунке:

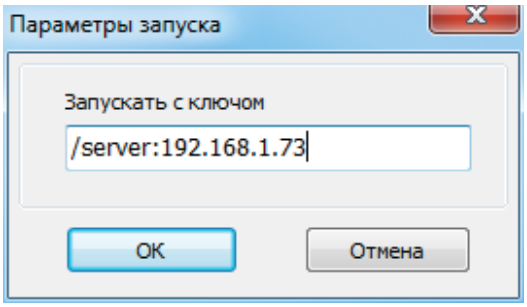


Рис. 22

Команды по запуску АРМа также находятся в пункте меню *Приложение*:

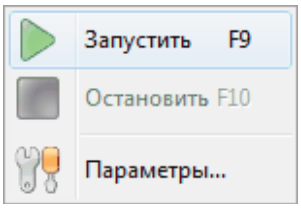


Рис. 23

10.3 Рабочий режим

В рабочем режиме программа Arm-Builder представляет собой АРМ.

Запуск

10.3.1 Запуск

Запуск АРМа осуществляется путем запуска файла *Arm-Builder.exe*.

Существует возможность запуска АРМа таким образом, чтобы в качестве OPC-сервера использовался WinDeCont, запущенный на удаленном компьютере. Для этого нужно запустить программу из командной строки со следующими параметрами:

Arm-Builder.exe /server:[host_ip]

где *[host_ip]* - адрес удаленного компьютера. При работе с удаленным компьютером рекомендуется создать ярлык для файла *Arm-Builder.exe*, в свойствах которого прописать имя файла и параметры командной строки.

10.4 Пример построения АРМа

В данном разделе будет показано, как создать простой АРМ пошагово. Процесс создания законченной системы отображения можно условно разделить на 4 стадии:

1. Определение структуры объекта и требований к АРМ.
2. Создание OPC-модели в программе "Конструктор OPC-модели".
3. Создание технологических экранов в программе "Inkscape".
4. Создание приложения в программе "Arm-Builder" и настройка взаимодействия АРМ с OPM-моделью.

10.4.1 Структура объекта

В качестве примера, рассмотрим упрощенную структуру центрального теплового пункта (ЦТП).

Структура объекта

Пусть ЦТП содержит две группы насосов:

1. Группа насосов горячей воды (2 насоса).
2. Группа насосов отопления (2 насоса).

Далее предположим, что для каждой группы имеются следующие характеристики:

1. Входное и выходное давления.
2. Каждый из насосов в группе обладает:
2.1 Состоянием включен\выключен.
2.2 Импульсными командами на включение и выключение.
2.3 Аварийным сигналом.

Пусть также, ЦТП производит мониторинг параметров:

1. Наличие пожарной сигнализации.
2. Контроль напряжения.

Исходя из предложенной структуры, мы имеем следующий набор сигналов.

Набор сигналов

Телесигнализация (ТС)

- Состояние насоса горячей воды 1.
- Состояние насоса горячей воды 2.
- Состояние насоса отопления 1.
- Состояние насоса отопления 2.
- Пожарная сигнализация.
- Напряжение.

Телеуправление (ТУ)

- Включить насос горячей воды 1.
- Отключить насос горячей воды 1.
- Включить насос горячей воды 2.
- Отключить насос горячей воды 2.
- Включить насос отопления 1.
- Отключить насос отопления 1.
- Включить насос отопления 2.
- Отключить насос отопления 2.

Телеизмерение (ТИ)

- Давление горячей воды на входе.
- Давление горячей воды на выходе.
- Давление отопления на входе.
- Давление отопления на выходе.

Следующий шаг: Список требований.

10.4.2 Список требований

Пусть мы имеем следующий список требований к АРМ.

1. Работать непрерывно в режиме реального времени под управлением операционной системы Windows 2000 или Windows XP.
2. Отображать технологическую схему и текущие технологические параметры ЦТП.
3. Отображать индикацией работающее и неработающее оборудование, а также оборудование в неопределенном состоянии.
4. Производить управление оборудованием ЦТП.

Описание технологического экрана (отображения)

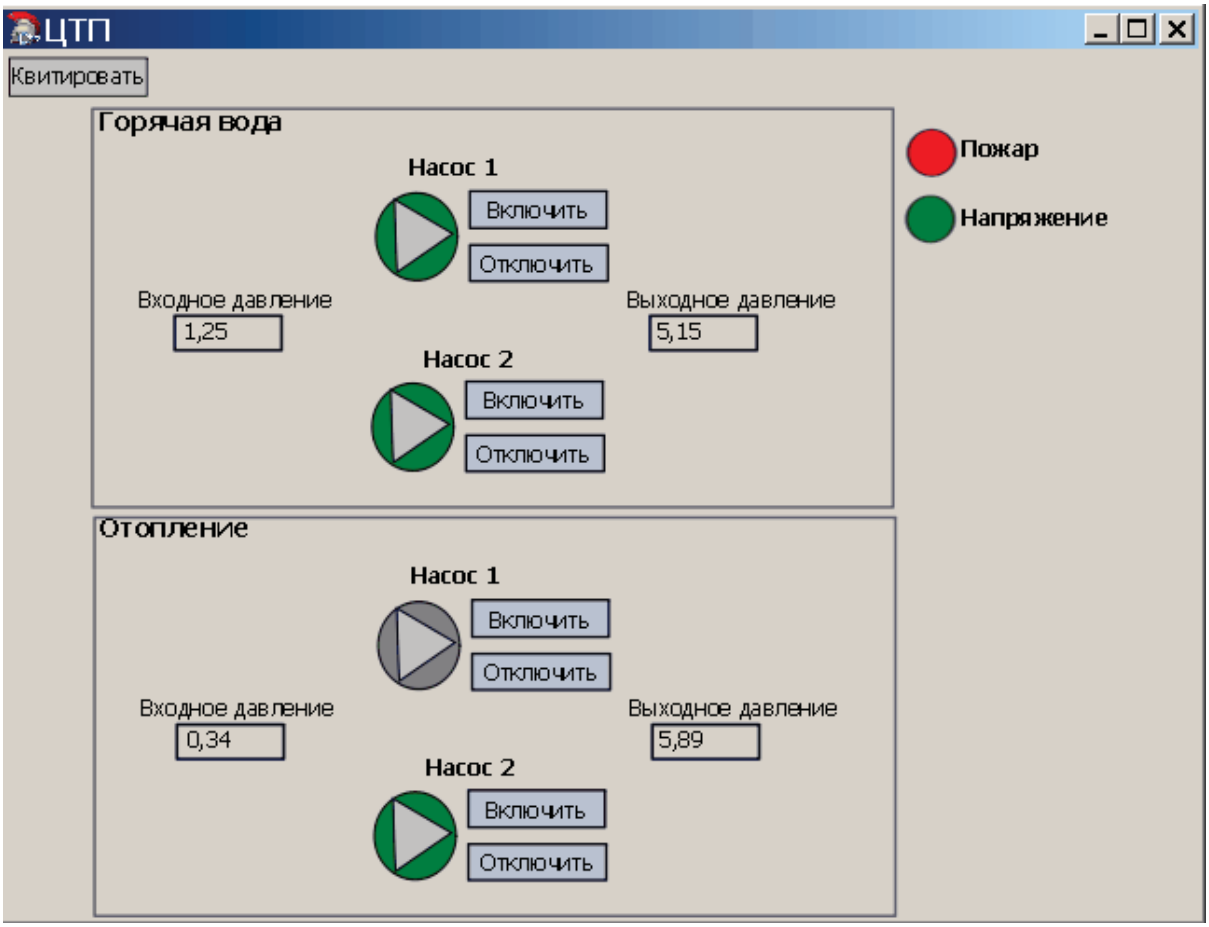


Рис. 24. Вид технологического экрана.

Насосы

Работающий насос имеет зеленый цвет фона круга, неработающий - белый. Если состояние насоса неопределенно - фон круга имеет желтый цвет.

Насос, находящийся в аварийном состоянии имеет цвет фона треугольника - мелькающий красный цвет, не в аварийном - серый.

Кнопка "Включить" посылает импульсную команду на включение насоса.

Кнопка "Отключить" посылает импульсную команду на отключение насоса.

Давления

В соответствующих полях выводятся значения входного и выходного давлений.

Пожар

При срабатывании датчика пожарной сигнализации круг становиться красного цвета, в нормальном состоянии - зеленым. Если

значение сигнала от датчика неопределенно круг имеет желтый цвет.

Напряжение

При пропадании напряжения на ЦТП круг становится красного цвета, в нормальном состоянии - при наличии сигнала - зеленым. Если значение сигнала от датчика неопределенно круг имеет желтый цвет.

Следующий шаг: **Создание модели**.

10.4.3 Создание модели

В данном разделе будет показано, как создать OPC-модель для нашего проекта с использованием программы "Конструктор OPC-модели" **(ПО "SyTrack-TOOL" OPCModelBuilder**).

Начнем с создания нового проекта.

10.4.3.1 Создание нового проекта

Начнем построение модели с создания нового проекта.

Запустите "Конструктор OPC-модели".

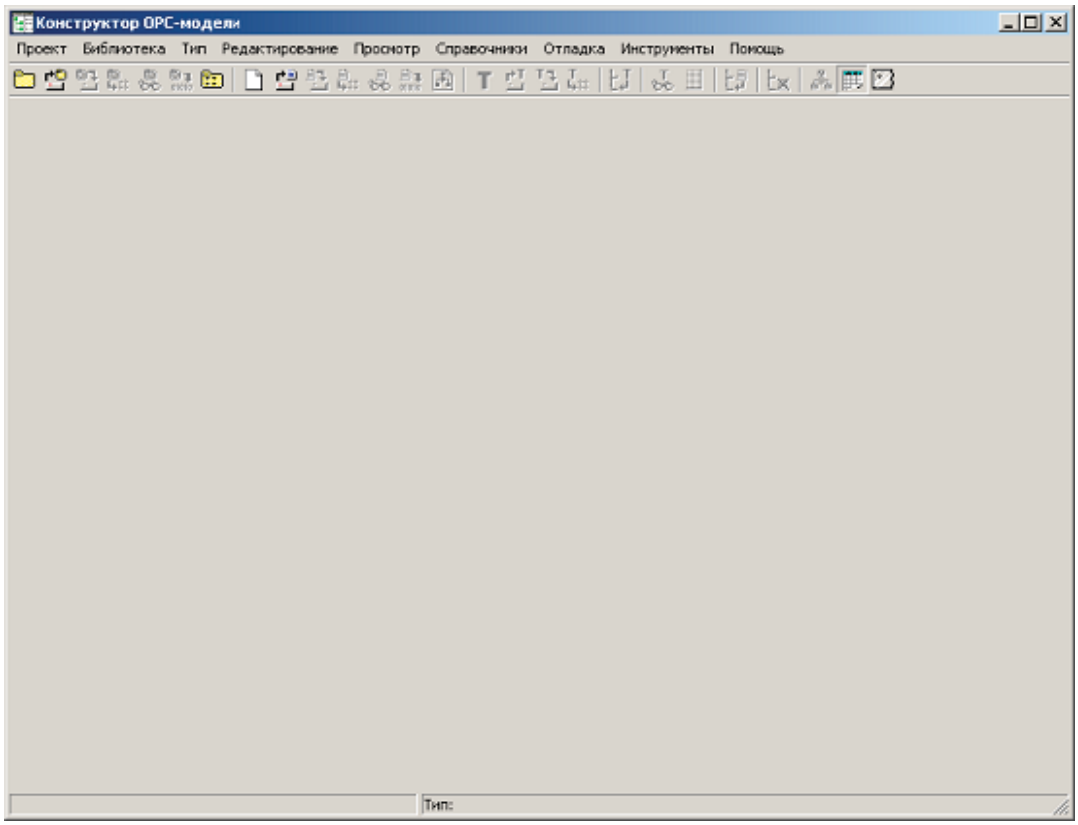


Рис. 25. Конструктор OPC-модели.

Создайте новый проект ( ) и сохраните его ( ). Библиотеку назовите CTP, а проект ARM.

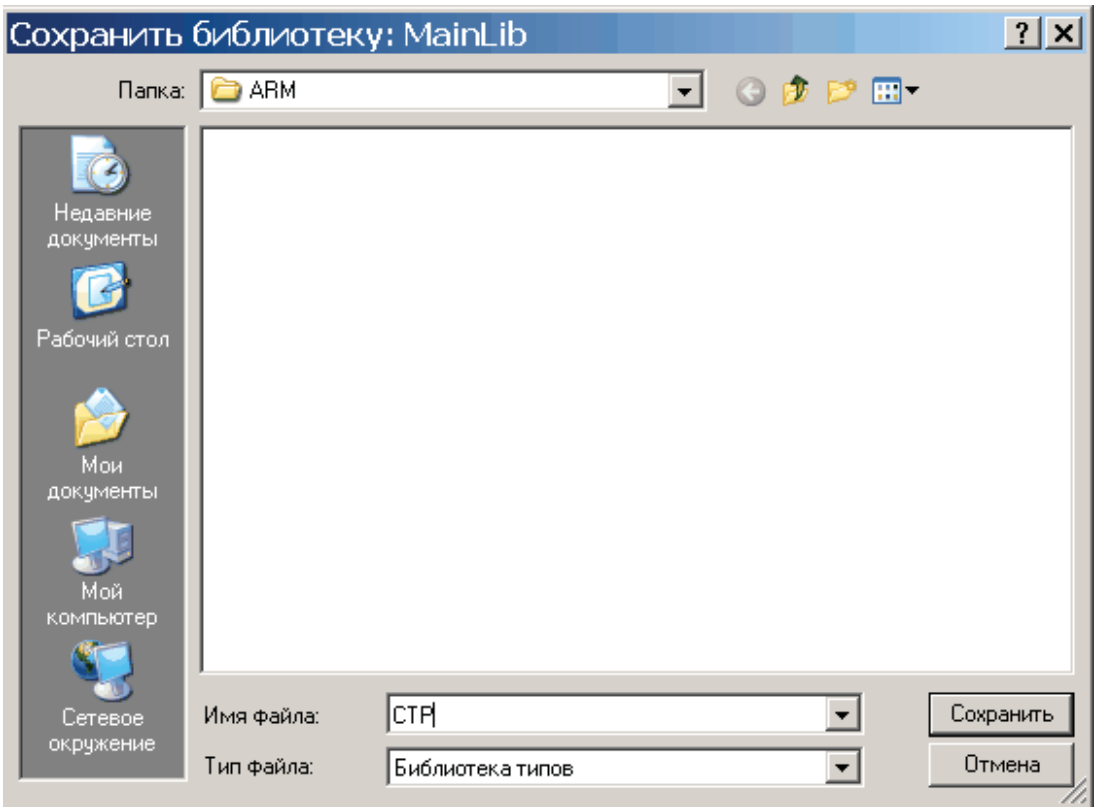


Рис. 26. Сохранение библиотеки.

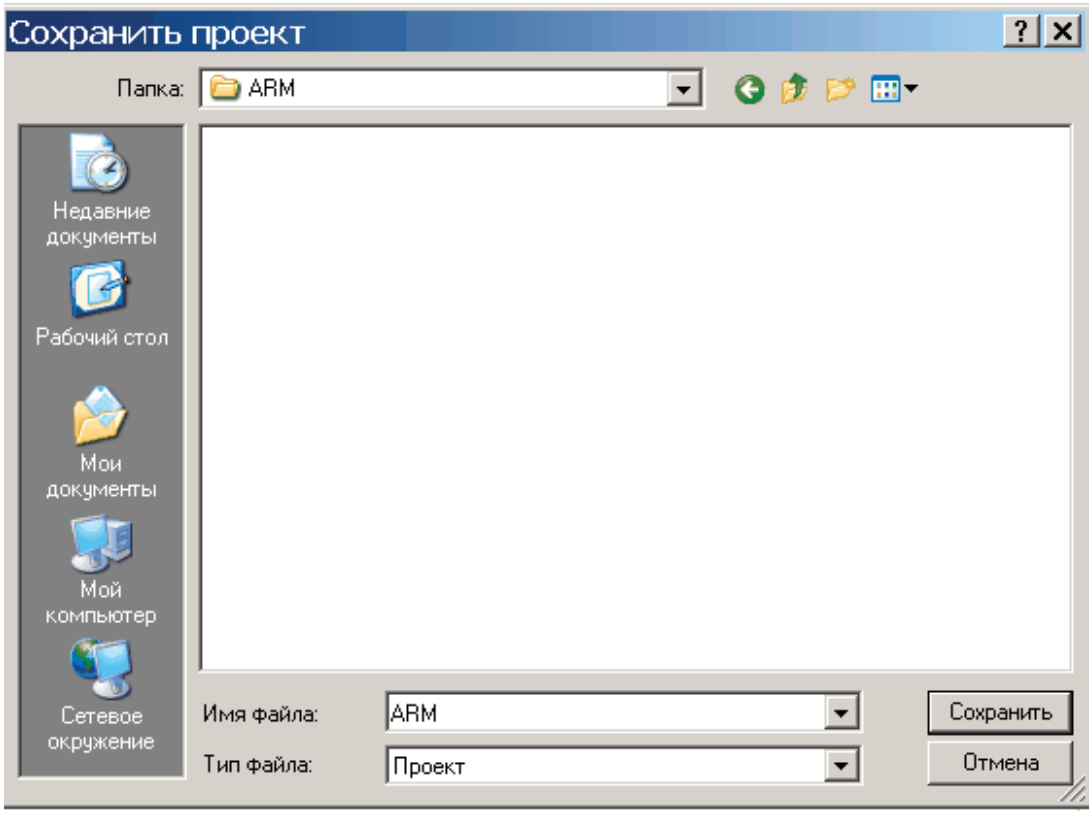


Рис. 27. Сохранение проекта.

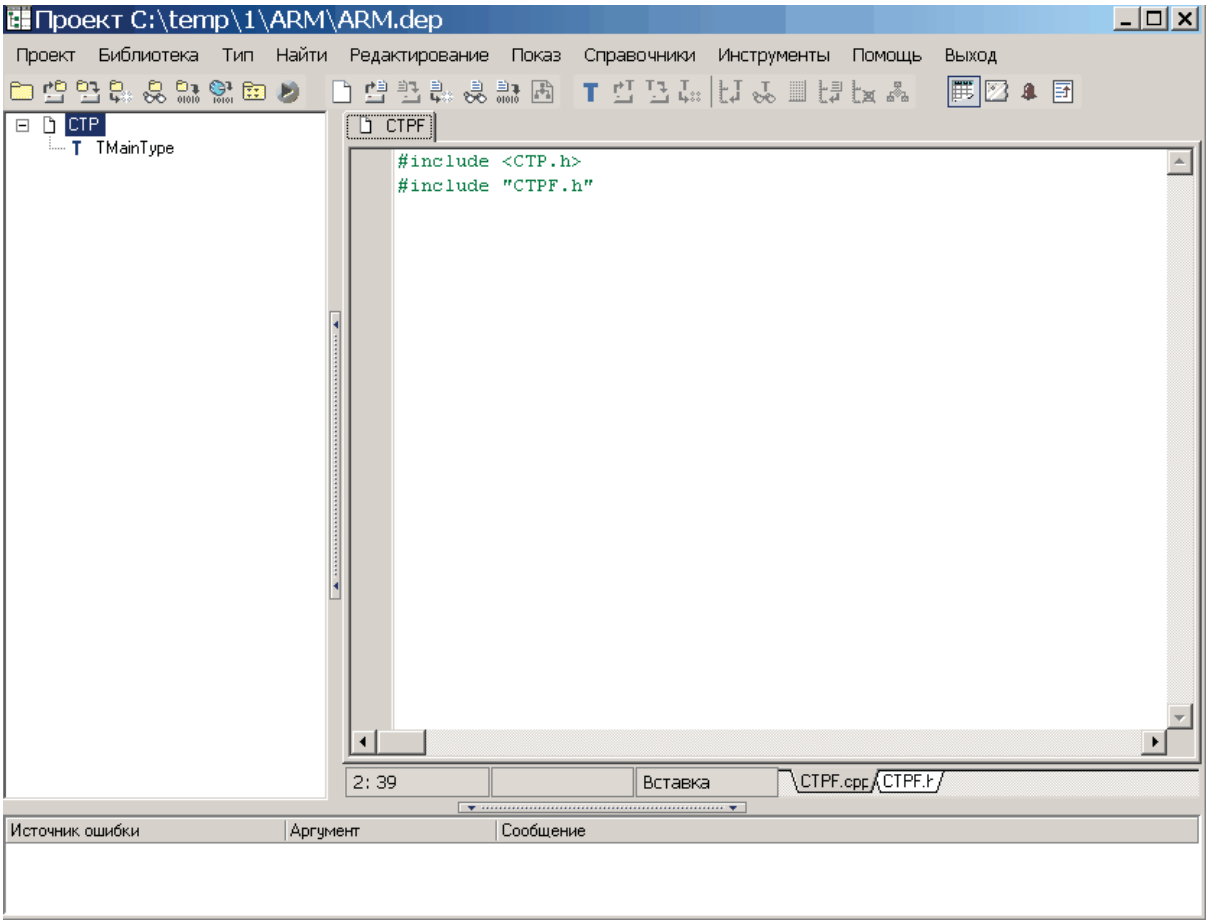


Рис. 28. Новый проект.

Давайте теперь подумаем, какие нам потребуются типы и как нам надо будет их структурировать.

Следующий шаг: Структура модели.

10.4.3.2 Структура модели

Исходя из структуры объекта (Структура объекта) нам потребуются следующие типы:

Тип - насос (Pump)

Насос имеет состояние (включен/выключен), аварийный сигнал и управление (включить\выключить). Состояние насоса - это дискретный входной сигнал, управление - это дискретный импульсный выходной сигнал, аварийный сигнал - это сигнал, показывающий есть ли неопределенное состояние у насоса. Исходя из этого, построим структуру типа насос (в названиях типов и элементов используются только латинские символы).

Pump (насос)

- State (состояние) - входной дискрет.

- CmdOn (включить) - выходной импульсный дискрет.
- CmdOff (выключить) - выходной импульсный дискрет.
- SignalAlarm (сигнал аварии) - сигнал, показывающий есть ли неопределенное состояние у насоса.

Насос не имеет сценариев работы.

Тип - группа насосов (GroupPump)

Группа насосов имеет два насоса, входное и выходное давления. Входное и выходное давления - это входные аналоговые сигналы.

GroupPump (группа насосов)

- Pump1 (насос 1) - тип Pump.
- Pump2 (насос 2) - тип Pump.
- InPressure (входное давление) - входной аналог.
- OutPressure (выходное давление) - входной аналог.

Группа насосов не имеет сценариев работы.

Главный тип - ЦТП (TCTP)

ЦТП имеет две группу насосов, а также ведет контроль пожарной сигнализации и напряжения. Пожарная сигнализация и напряжение - это входные дискретные сигналы.

TCTP (ЦТП)

- GroupOtopl (группа насосов отопления) - тип GroupPump.
- GroupHot (группа насосов горячей воды) - тип GroupPump.
- Fire (пожарная сигнализация) - входной дискрет.
- Voltage (напряжение) - входной дискрет.

ЦТП не имеет сценариев работы.

Теперь нужно создать все эти типы в "Конструкторе OPC-модели".

Следующий шаг: Создание типа - насос.

10.4.3.3 Создание типа - насос

Создайте новый тип (). Назовите его Pump, выберите iInt в качестве базового типа.

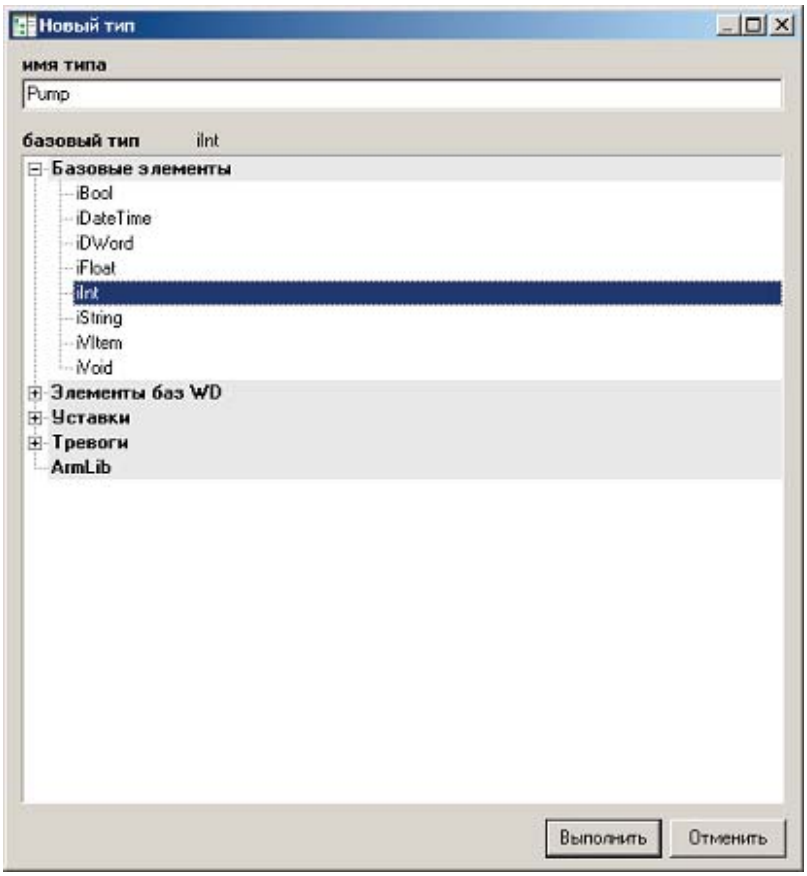


Рис. 29. Создание нового типа.

Добавьте типу новый элемент ( ). Назовите его State (Состояние). Как вы помните, состояние - это входной дискрет, поэтому, в качестве базового типа выбираем wdDIn из библиотеки "Элементы баз WD".

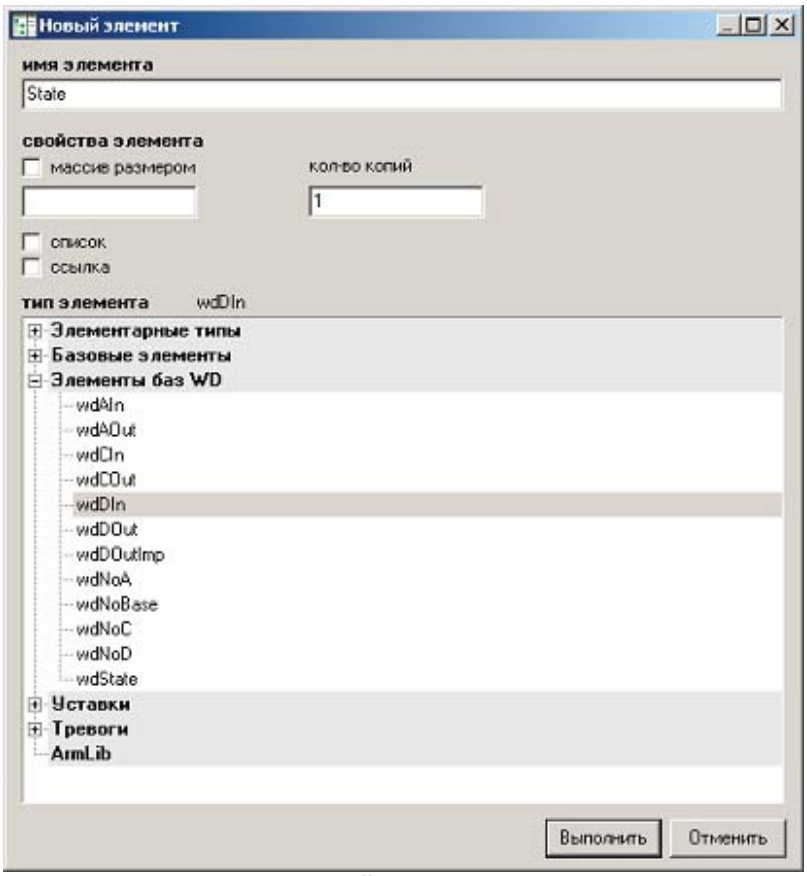


Рис. 30. Добавление свойства State (состояние насоса).

Добавьте свойства управления (включение и выключение). Так как включение и выключение это выходные импульсные дискреты в качестве базового типа выбираем wdOutImp из библиотеки "Элементы баз WD".

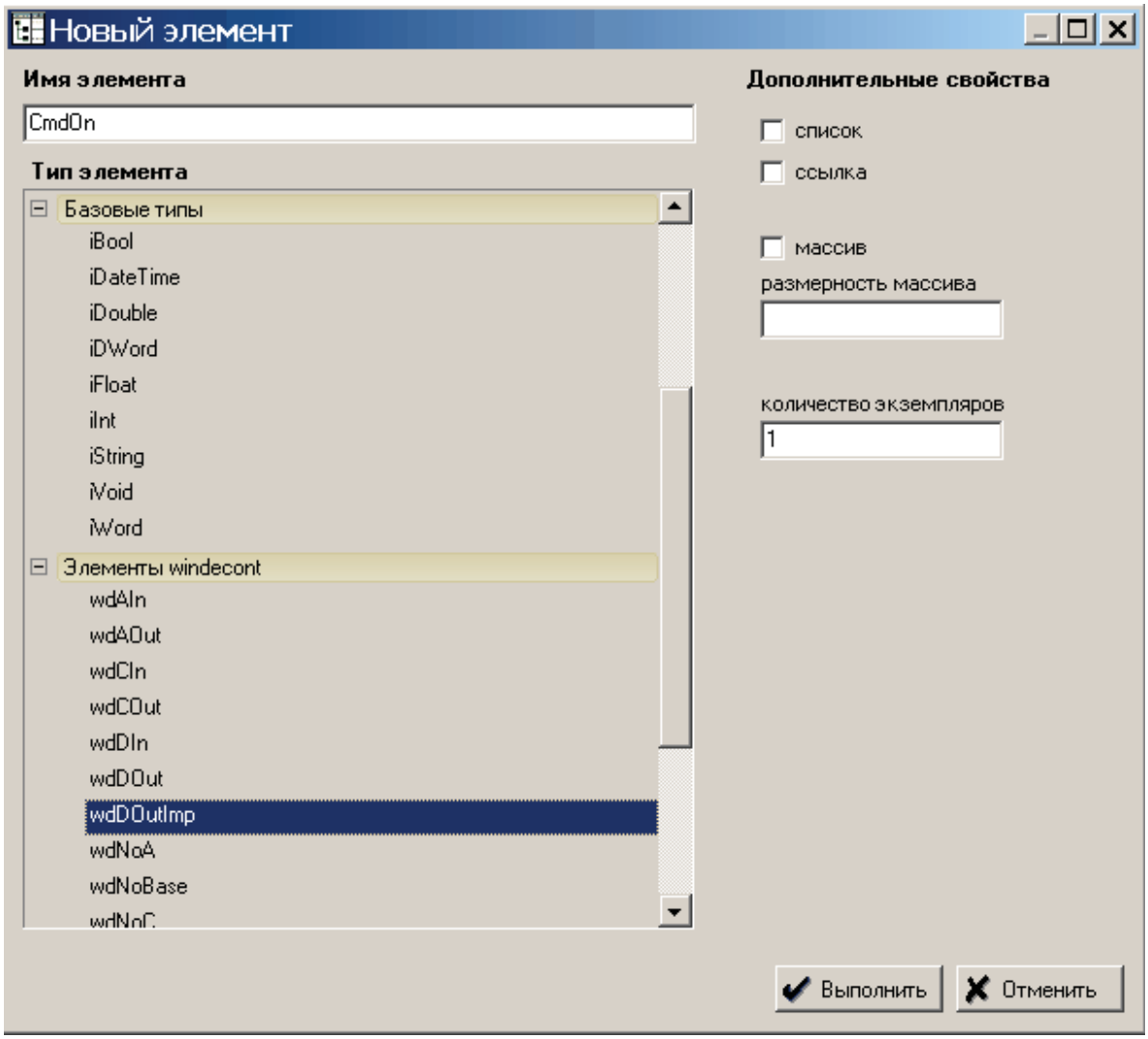


Рис. 31. Добавление свойства CmdOn (включение насоса).

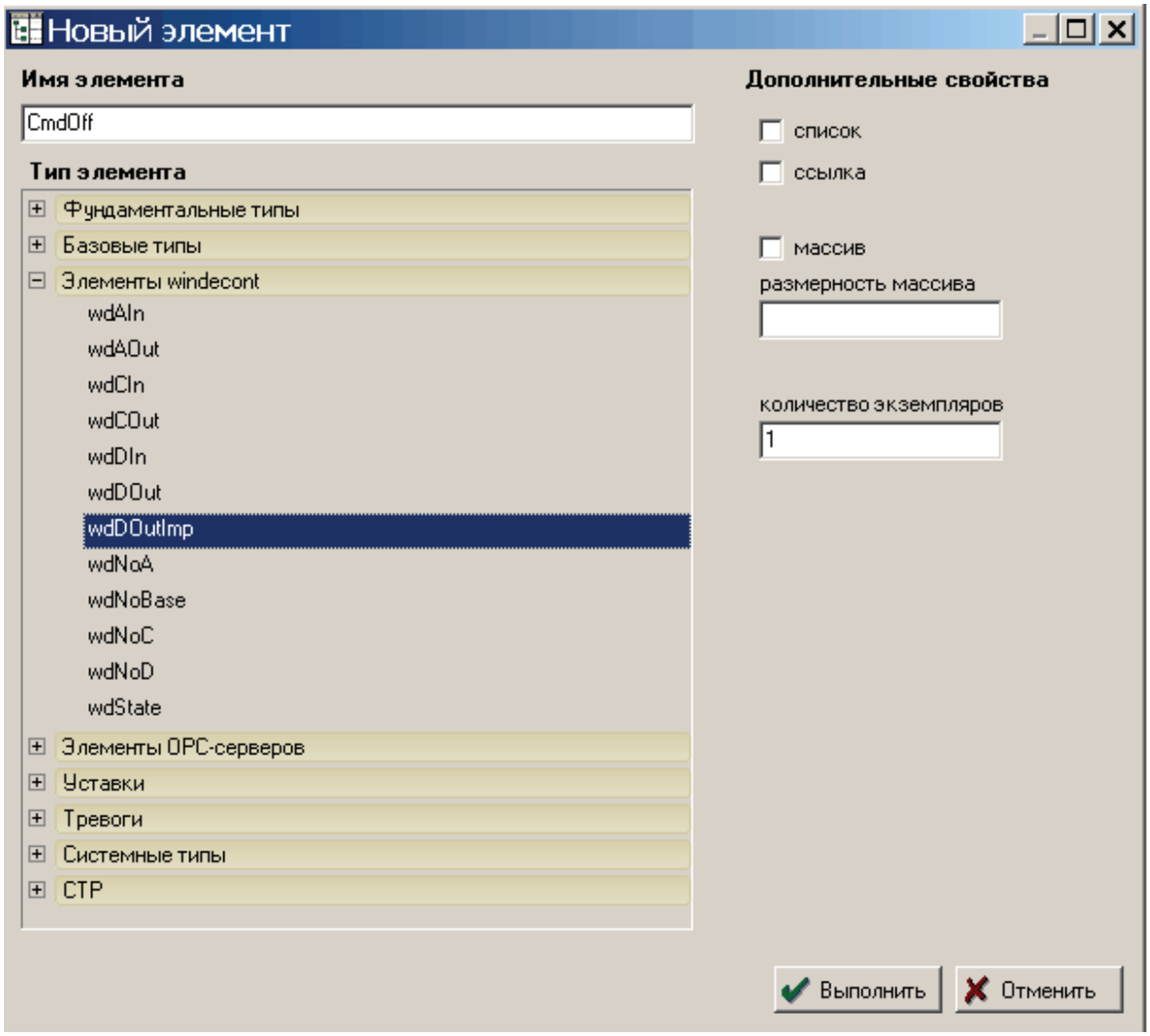


Рис. 32. Добавление свойства CmdOff (выключение насоса).

После всех операций окно редактирования типа примет вид.

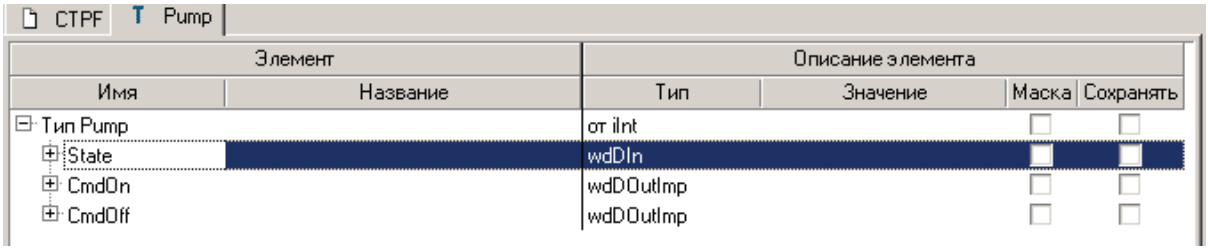


Рис. 33. Тип Pump.

Дадим название всем свойствам. Для этого щелкните мышкой по ячейке в строке State и столбце "Название". Вокруг ячейки при этом появится пунктирная рамочка. Напишите "состояние".

Рис. 34. Редактирование названия для свойства State.

Аналогичным образом впишите названия для CmdOn - "Включить" и CmdOff - "Выключить".

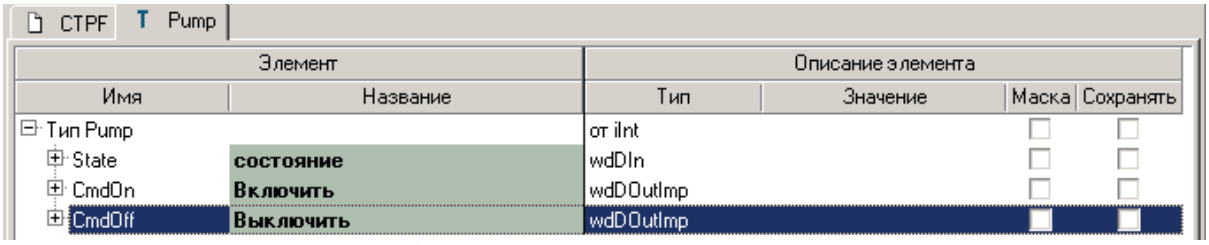


Рис. 35. Тип Pump с названиями элементов.

Сохраните тип Pump ( ). Тип Pump появится в нашей библиотеке.

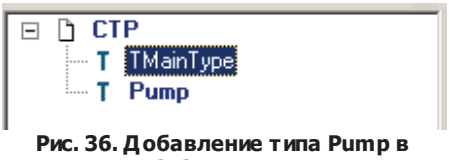


Рис. 36. Добавление типа Pump в библиотеку.

Следующий шаг: **Создание типа - группа насосов**.

10.4.3.4 Создание типа - группа насосов

Создайте новый тип, назовите его GroupPump. В качестве базового типа выберите iInt.

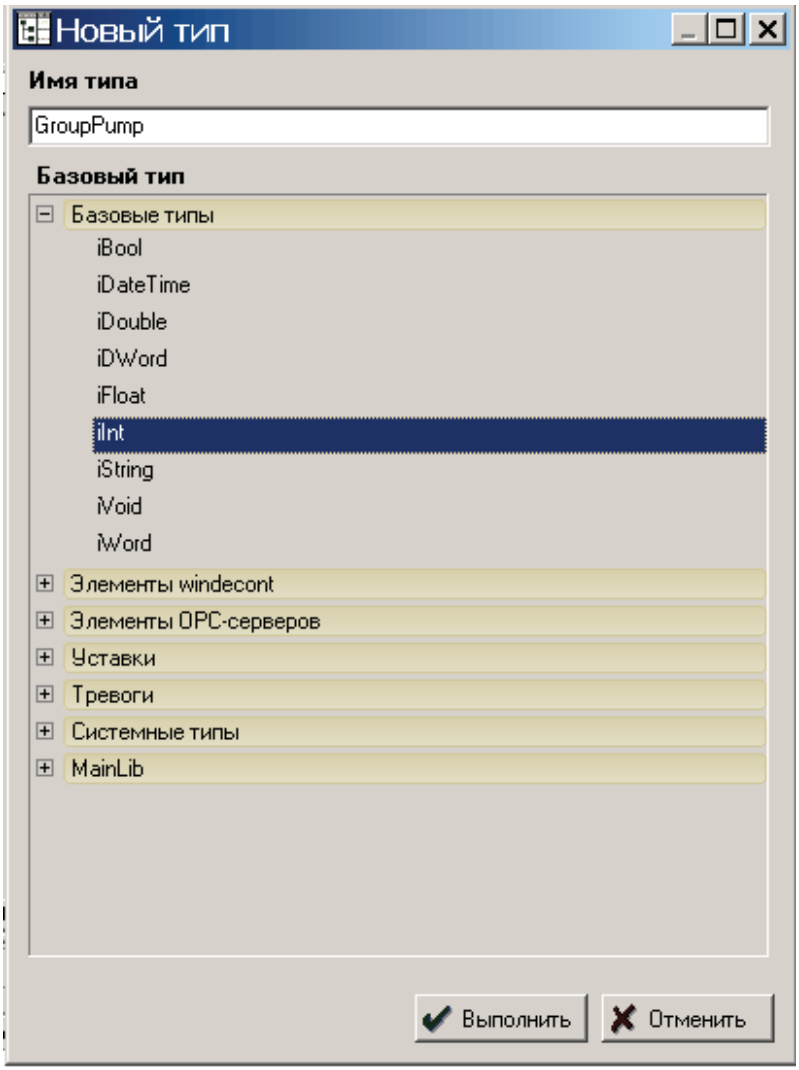


Рис. 37. Создание типа PumpGroup (группа насосов).

Добавьте типу элементы: Pump1 (насос 1) - тип Pump (библиотека "СТР"), Pump2 (насос 2) - тип Pump (библиотека "СТР"), InPressure (входное давление) - тип wdAIn (библиотека "Элементы баз WD"), OutPressure (выходное давление) - тип wdAIn (библиотека "Элементы баз WD").

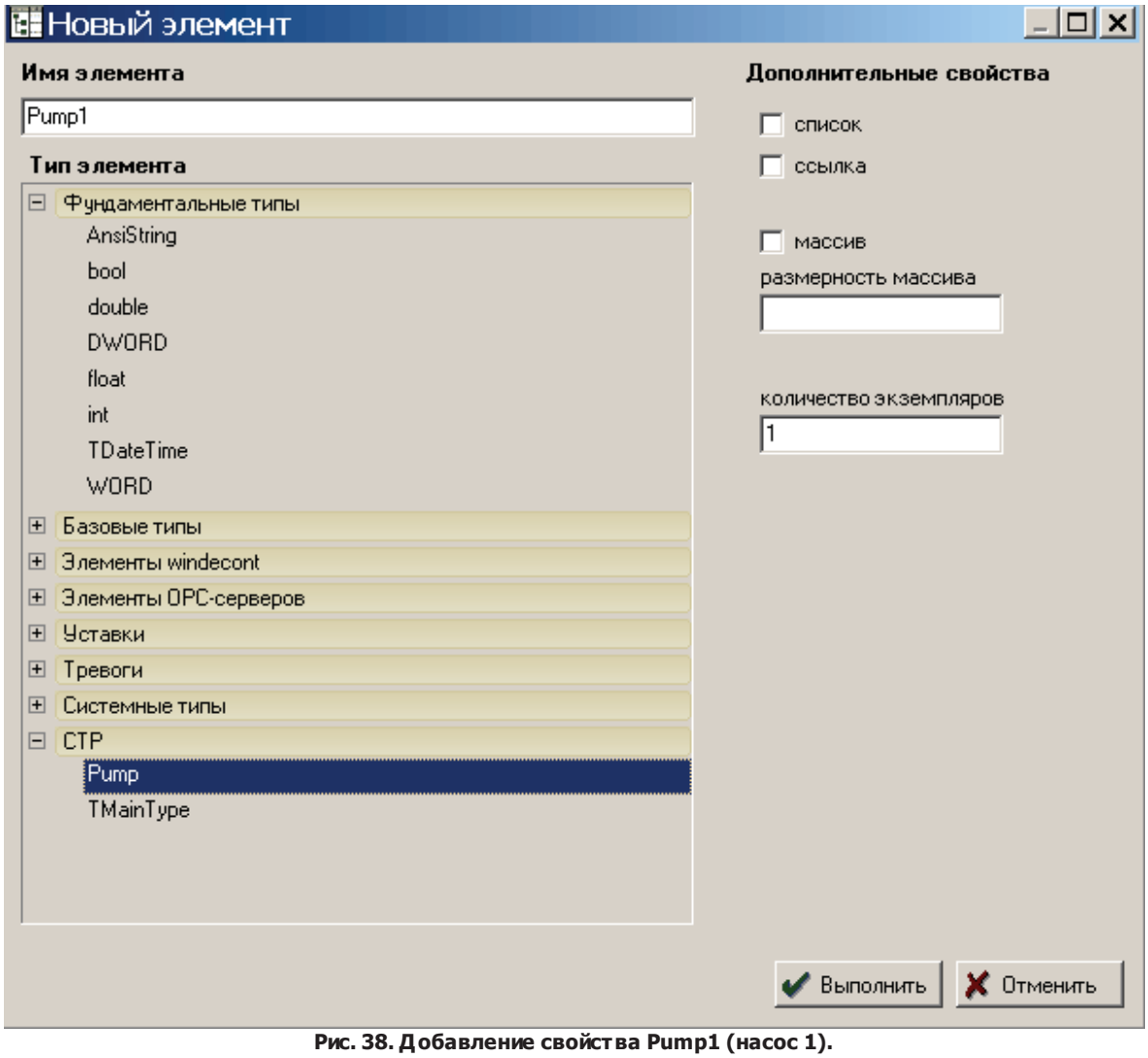


Рис. 38. Добавление свойства Pump1 (насос 1).

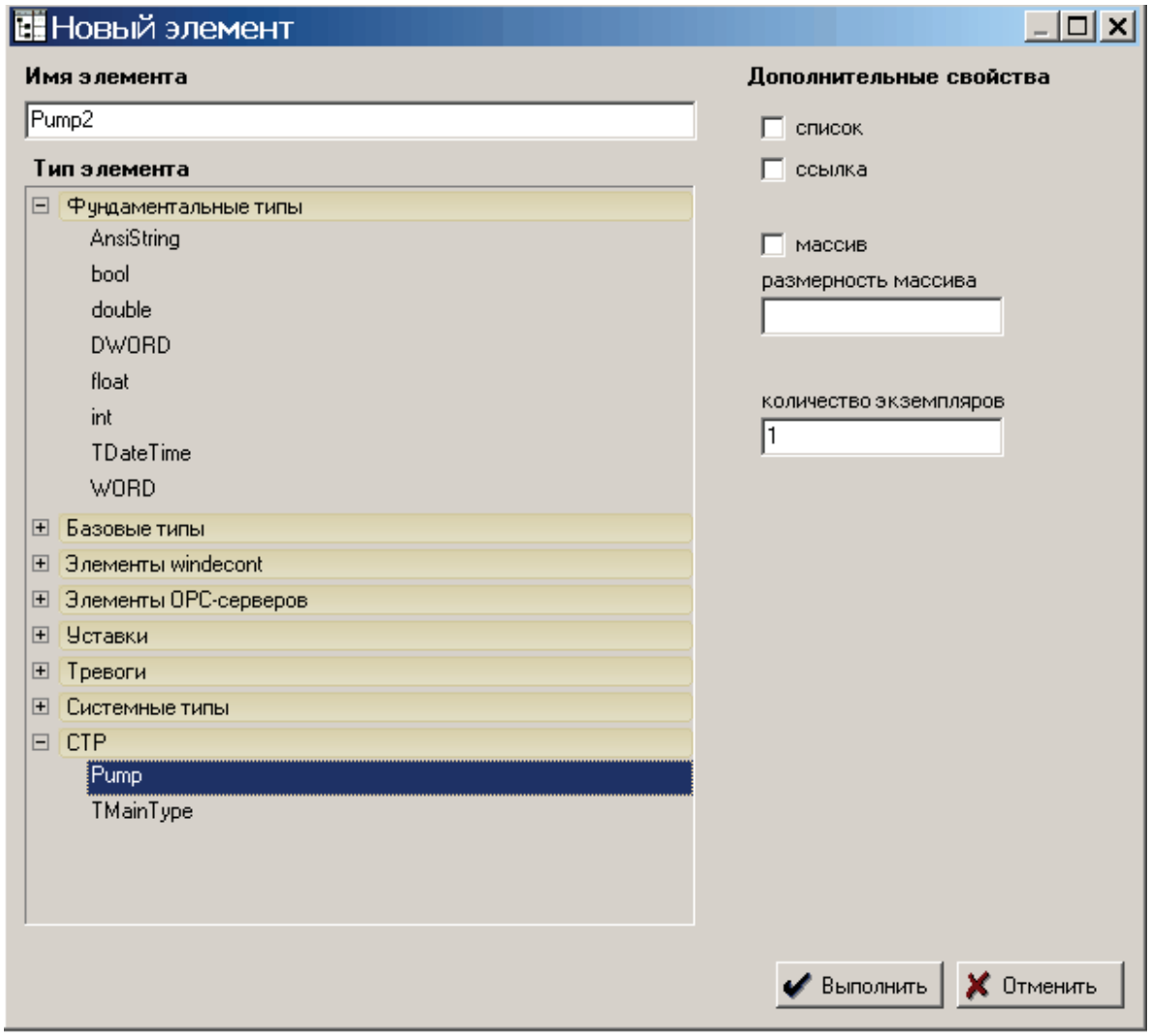


Рис. 39. Добавление свойства Pump2 (насос 2).

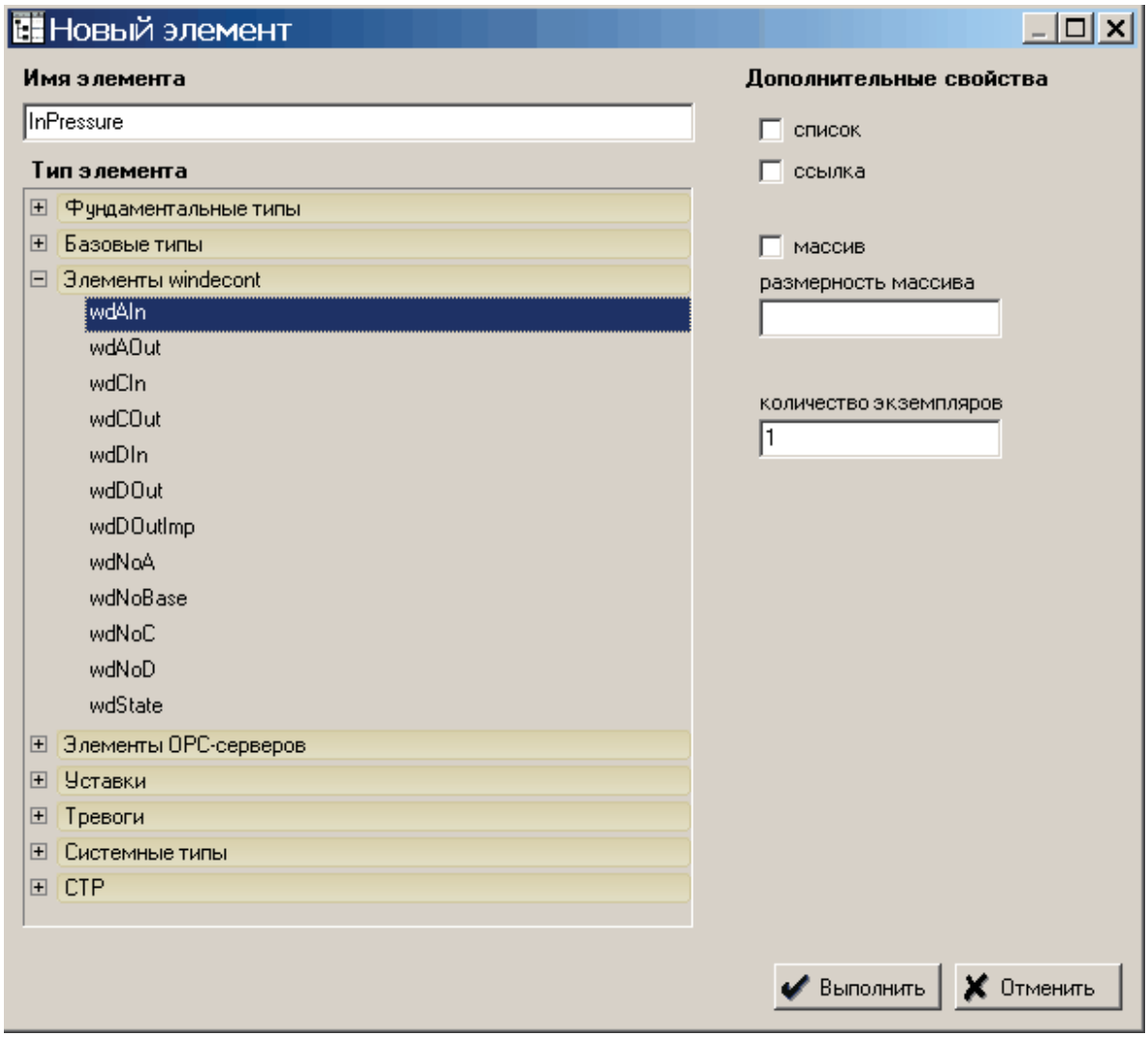


Рис. 40. Добавление свойства InPressure (входное давление).

Рис. 41. Добавление свойства OutPressure (выходное давление).

Окно редактирования типа примет вид.

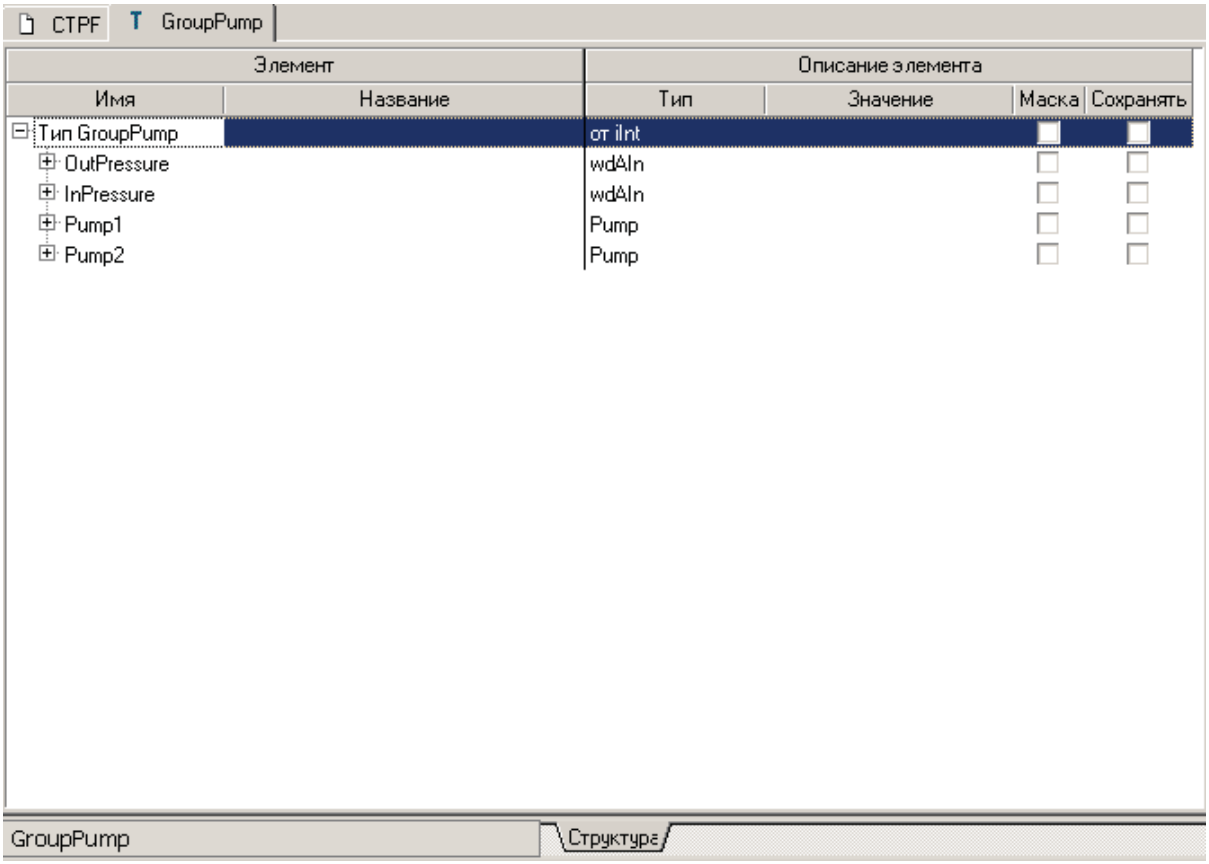


Рис. 42. Тип PumpGroup (группа насосов).

Дадим название всем свойствам.

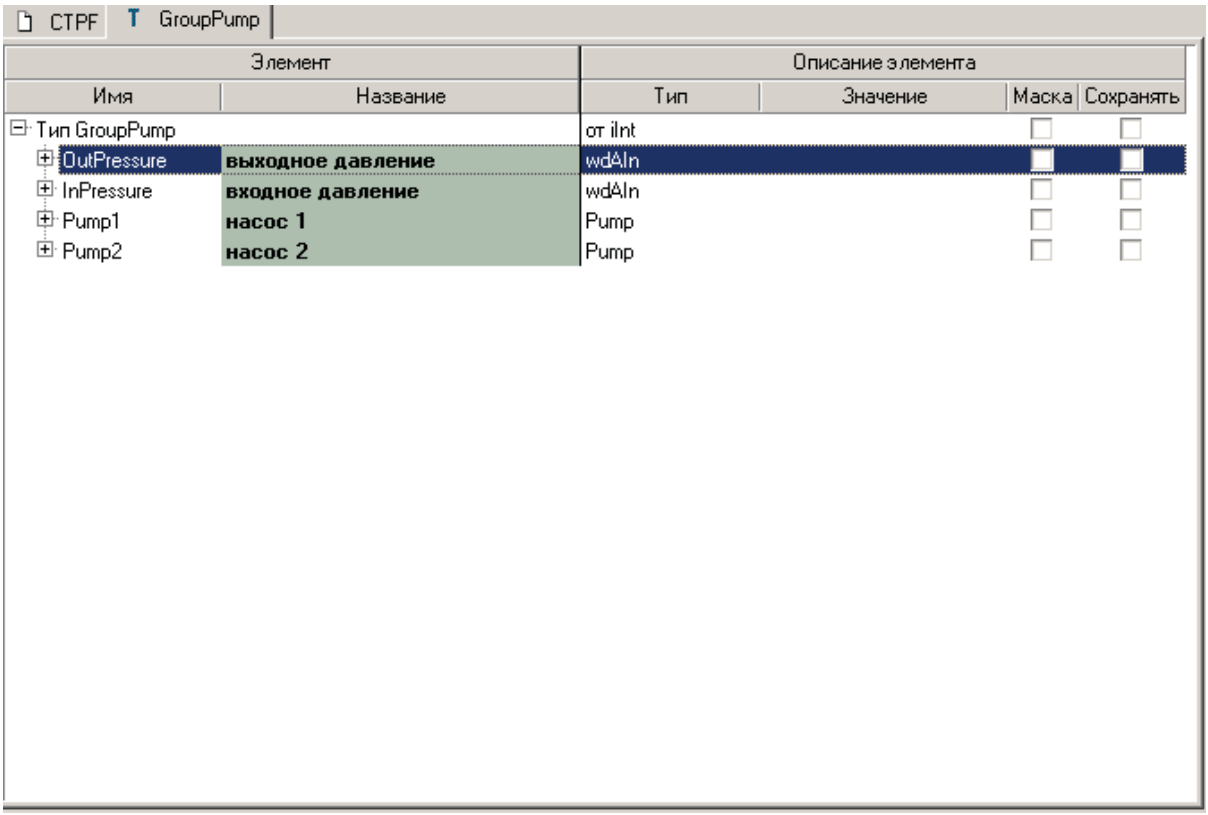


Рис. 43. Тип GroupPump с названиями элементов.

Сохраните тип GroupPump.

Рис. 44. Добавление типа GroupPump в библиотеку.

Следующий шаг: [Создание главного типа (ЦТП)](#).

10.4.3.5 Создание главного типа (ЦТП)

Создайте новый тип. Назовите его TCTP, в качестве базового типа выберите iInt.

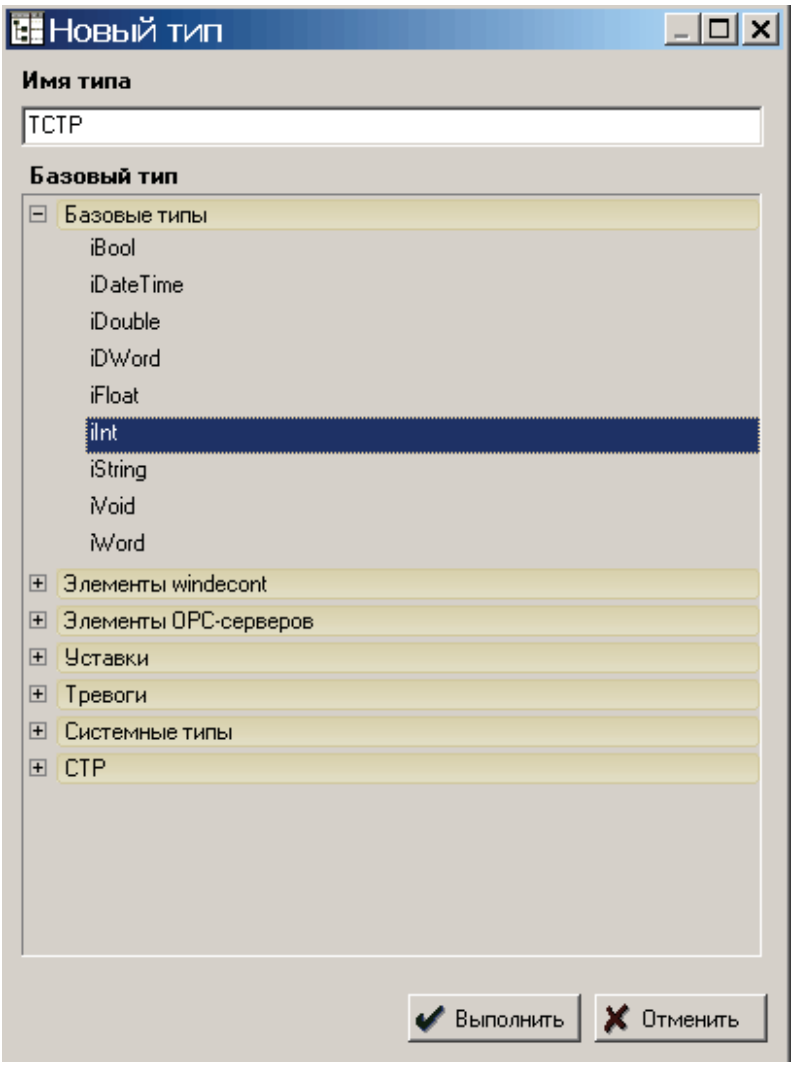


Рис. 45. Создание типа СТР (ЦТП).

Добавьте типу элементы: GroupHot (группа насосов горячей воды) - тип GroupPump (библиотека "СТР"), GroupOtopl (группа насосов отопления) - тип GroupPump (библиотека "СТР"), Fire (пожарная сигнализация) - тип wdDIn (библиотека "Элементы баз WD"), Voltage (напряжение) - тип wdDIn (библиотека "Элементы баз WD").

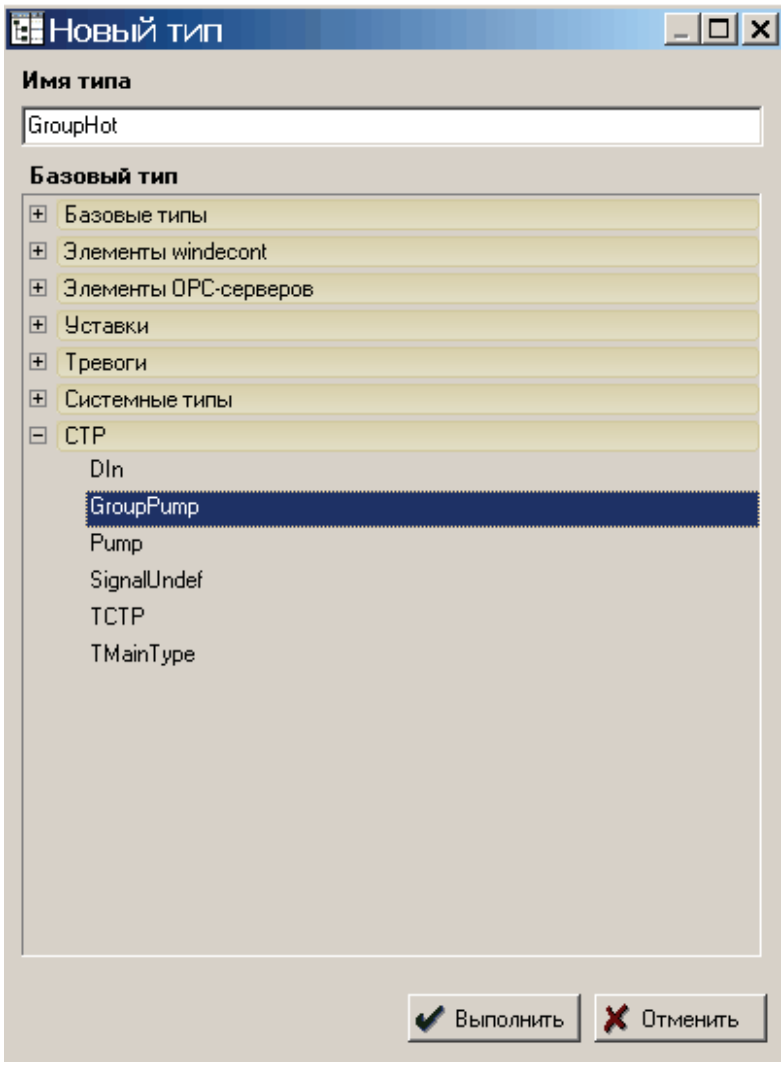


Рис. 46. Добавление свойства GroupHot (группа насосов горячей воды).

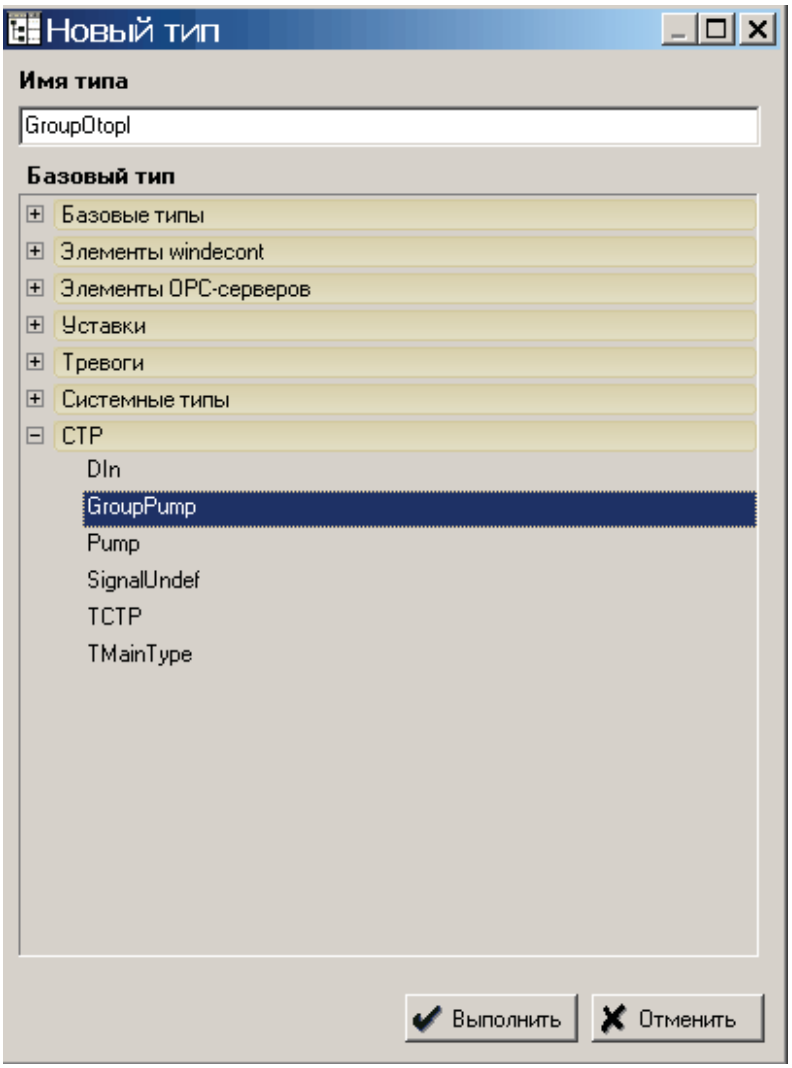


Рис. 47. Добавление свойства GroupOtopl (группа насосов отопления).

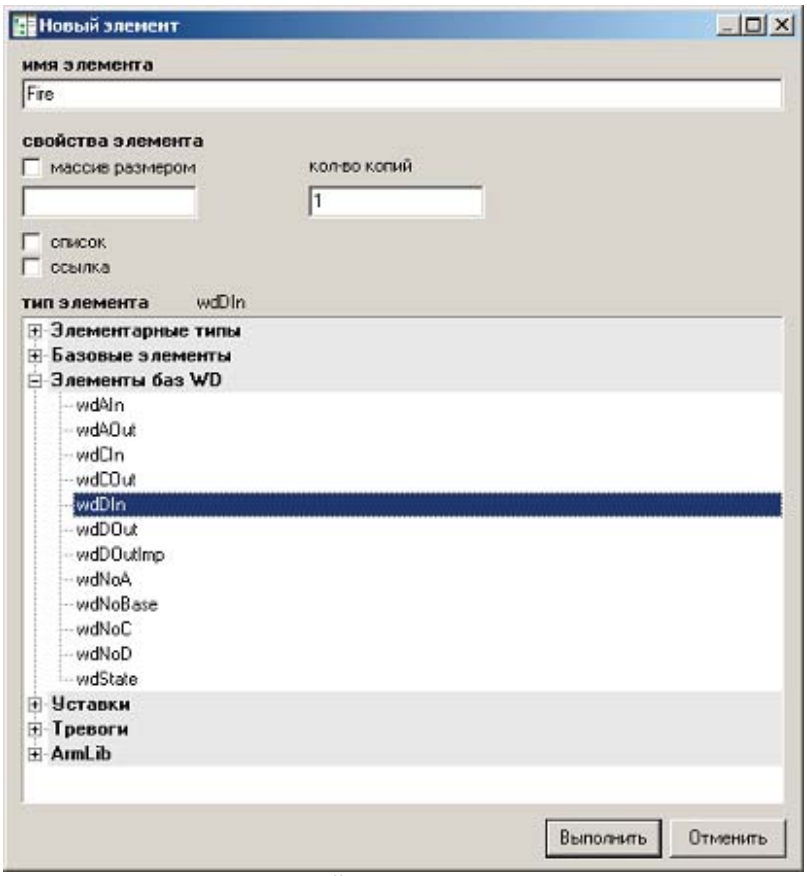


Рис. 48. Добавление свойства Fire (пожарная сигнализация).

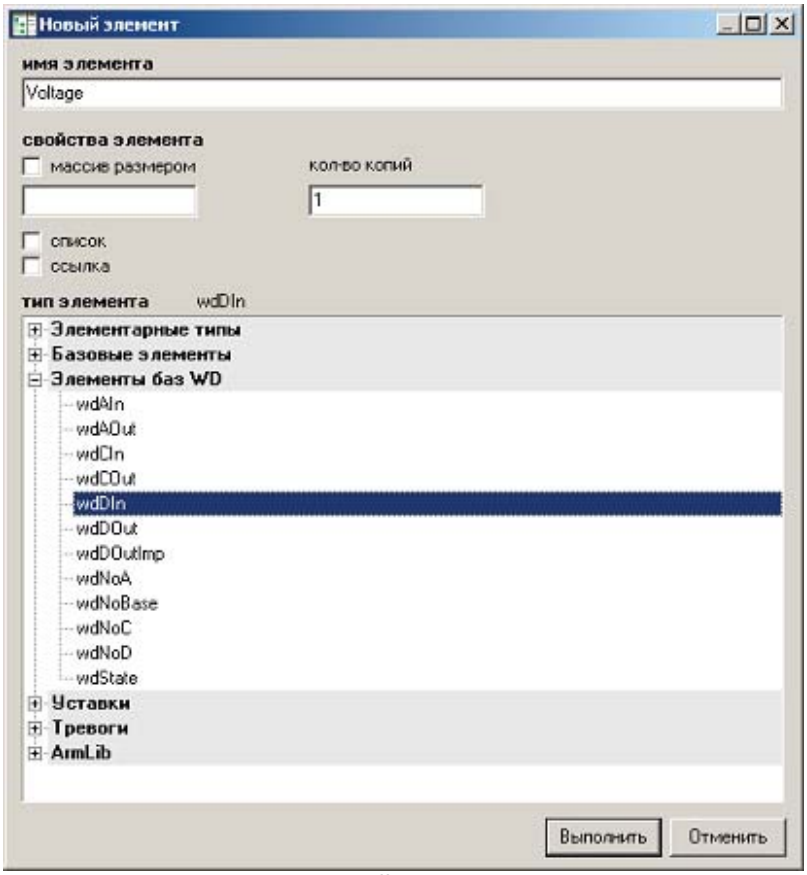


Рис. 49. Добавление свойства Voltage (напряжение).

Получаем:

CTPF | T TCTP

Элемент		Описание элемента			
Имя	Название	Тип	Значение	Маска	Сохранять
Тип TCTP		от iInt			
AllUndef	**авария по всем насосам**	alAlarm			
GroupOtopl	**группа насосов отопления**	GroupPump			
GroupHot	**группа насосов горяч. воды**	GroupPump			
Voltage	**напряжение**	wdDIn			
Fire	**пожар**	wdDIn			

Рис. 50. Тип TCTP (ЦТП).

Не забудьте сохранить тип TCTP.

Мы закончили создавать типы нашей модели. Сохраните все сделанные изменения ( ).

Следующий шаг: **Создание аварийного сигнала насоса**.

10.4.3.6 Создание аварийного сигнала насоса

Теперь приступим к формированию аварийного сигнала насоса. Этот сигнал будет показывать есть ли неопределенность у данного насоса.

Для начала создадим новый тип DIn. Этот тип будут иметь все сигналы, по которым будет выставляться неопределенность у насоса. Это необходимо, так как имя типа элементов, по которым будет формироваться авария, должно отличаться от имен типов всех остальных элементов.

То есть, если будет хотя бы один сигнал, имеющий данный тип и неопределенность, у соответствующего насоса будет аварийное состояние.

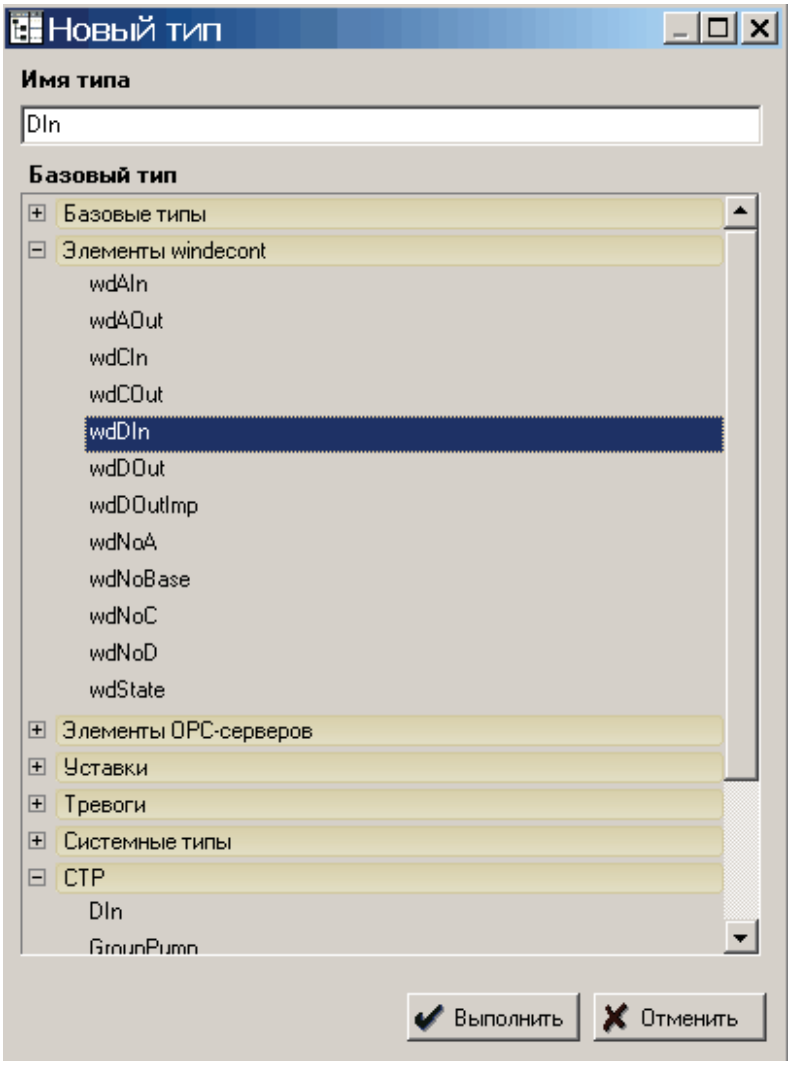


Рис. 51. Создание типа DIn.

Теперь поставим этот тип у всех интересуемых сигналов. В нашем случае - это единственный сигнал - состояние насоса.

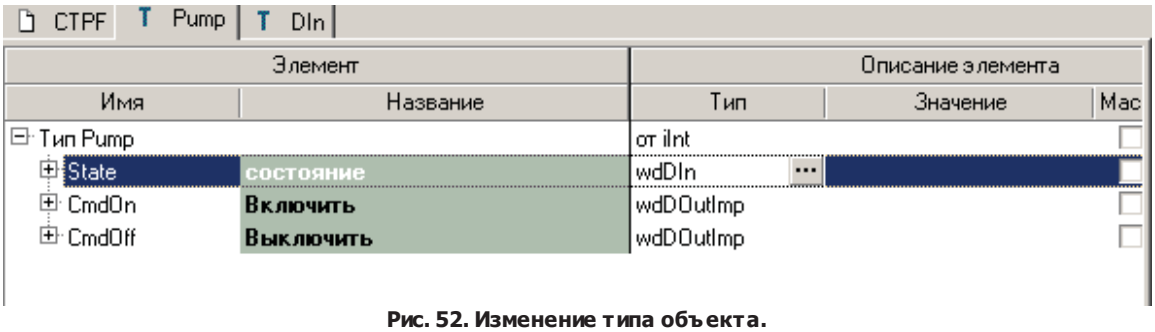


Рис. 52. Изменение типа объекта.

В результате получаем:

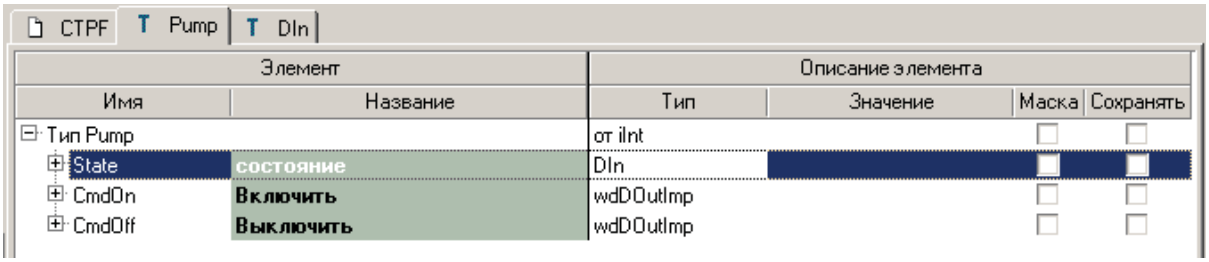


Рис. 53. Тип Pump.

Добавим аварийный сигнал в тип Pump, назовем его SignalAlarm. Этот сигнал будет показывать есть ли неопределенность у данного насоса. В качестве типа элемента выберем "alToBad" из библиотеки "Тревоги". Более подробно описание типов тревог можно посмотреть в документации по работе с моделью в разделе «Элементы и функции модели/Тревоги».

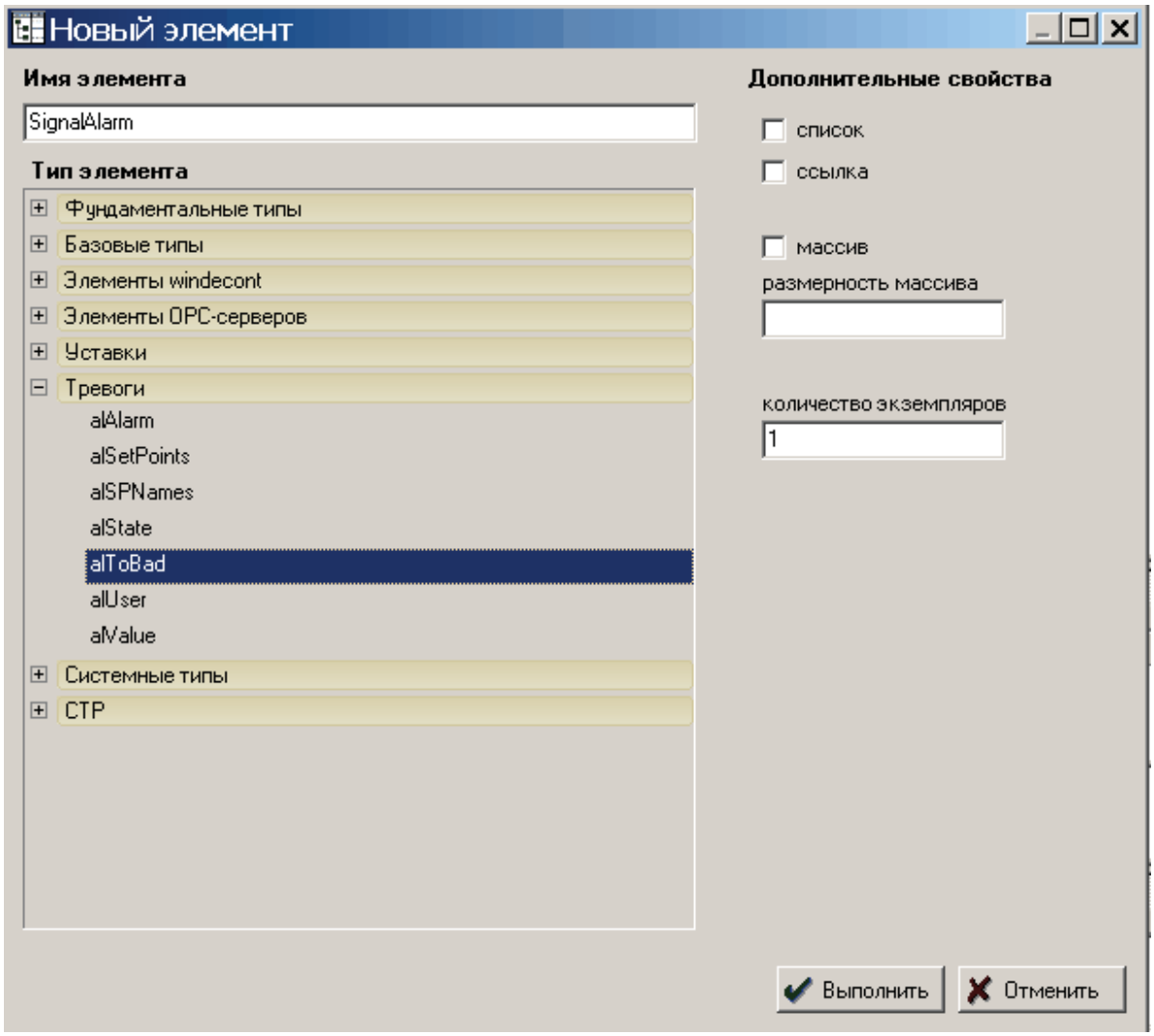


Рис. 54. Создание типа SignalAlarm.

В результате получили

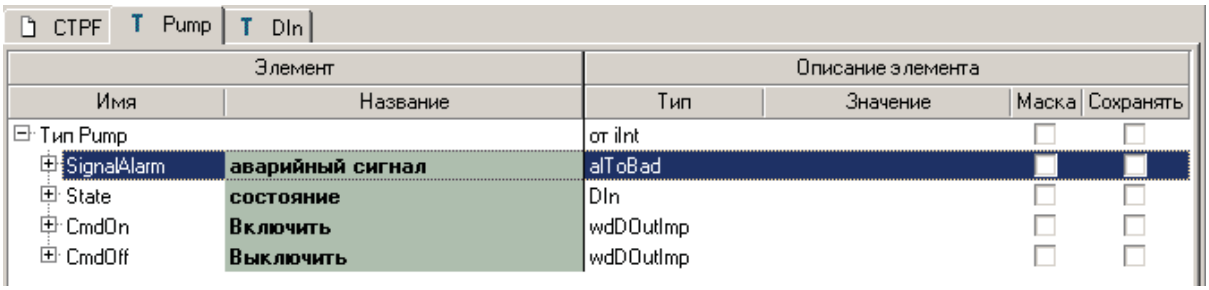


Рис. 55. Добавление аварийного сигнала в тип Pump.

В качестве одного из параметров только что созданного элемента "SignalAlarm" необходимо указать имя типа контролируемых элементов (TypeName). Такими элементами в нашем случае являются элементы, имеющие тип DIn. Значением параметра "Root" будет тип "Pump" (в этом узле будут найдены элементы типа "DIn"). Остальные параметры необходимо указать как показано на рисунке:

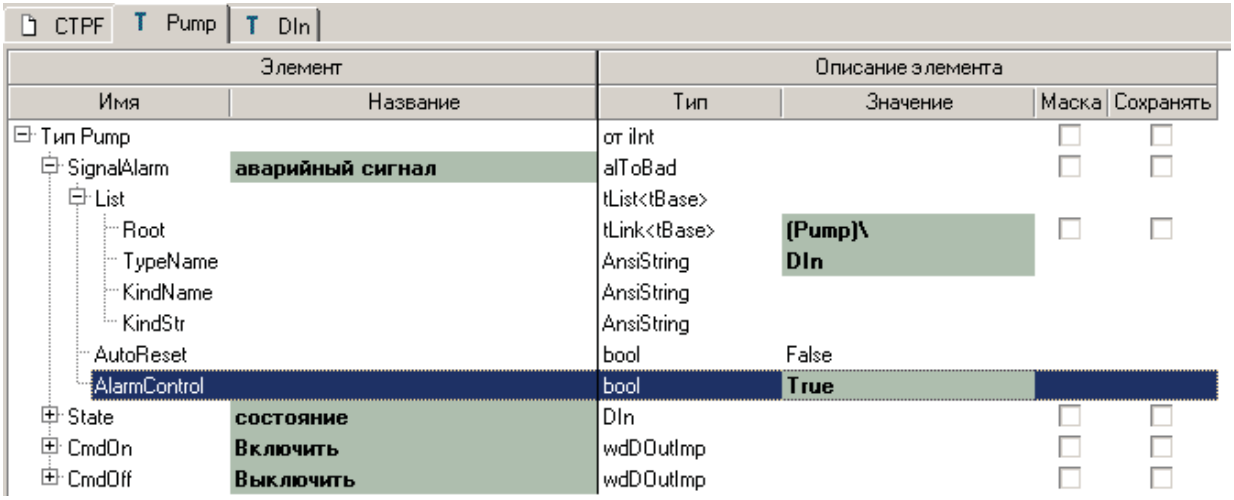


Рис. 56. Параметры аварийного сигнала.

"AutoReset" – (False) для автоматического сброса значения элемента тревоги в «0», если контролируемые элементы не в аварийном состоянии.

"AlarmControl" – (True) для установки признака «Внимание» у контролируемых элементов.

Следующий шаг: **Создание кнопки Квитировать**.

10.4.3.7 Создание кнопки Квитировать

Теперь приступим к формированию кнопки "Квитировать". Эта кнопка будет мигать красным цветом, если хотя бы у одного насоса есть аварийный сигнал. При нажатиии - мигание прекратится.

Для этого нам надо, чтобы в типе TCTP был общий сигнал по всем аварийным сигналам насоса.

Назовем этот элемент AllUndef и дадим тип alAlarm, означающий что он будет выставлять тревогу, если есть тревога хотя бы у одного контроллируемого элемента.

Как уже описывалось в Создание аварийного сигнала насоса необходимо создать тип, который будут иметь все элементы, по которым будет формироваться авария. Назовем этот тип SignalUndef. Он будет иметь тип alToBad.

И изменим на этот тип аварийный сигнал Pump.

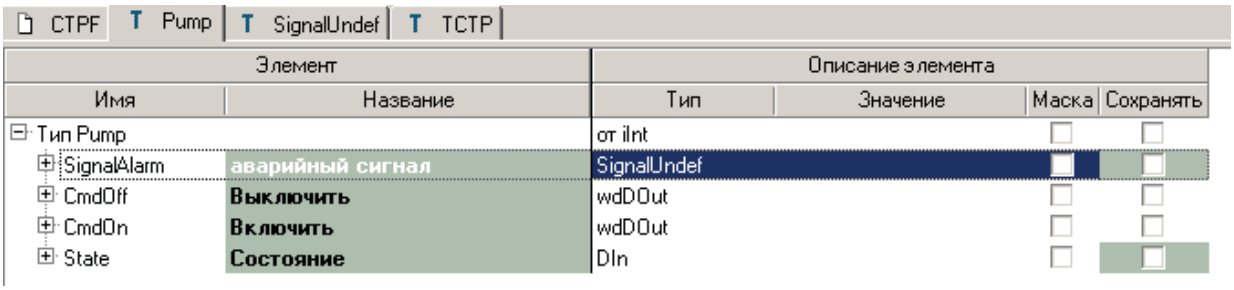


Рис. 57. Тип Pump.

Получается, что аварийные сигналы всех насосов имеют тип SignalUndef. Осталось определить, чтобы аварийный сигнал типа TCTP собирал все аварийные сигналы насосов.

Для этого выставляем сигналу AllUndef типа TCTP следующие параметры:

- поле Root - тип CTP
- TypeName - SignalUndef
- AutoReset - False
- AlarmControl - True

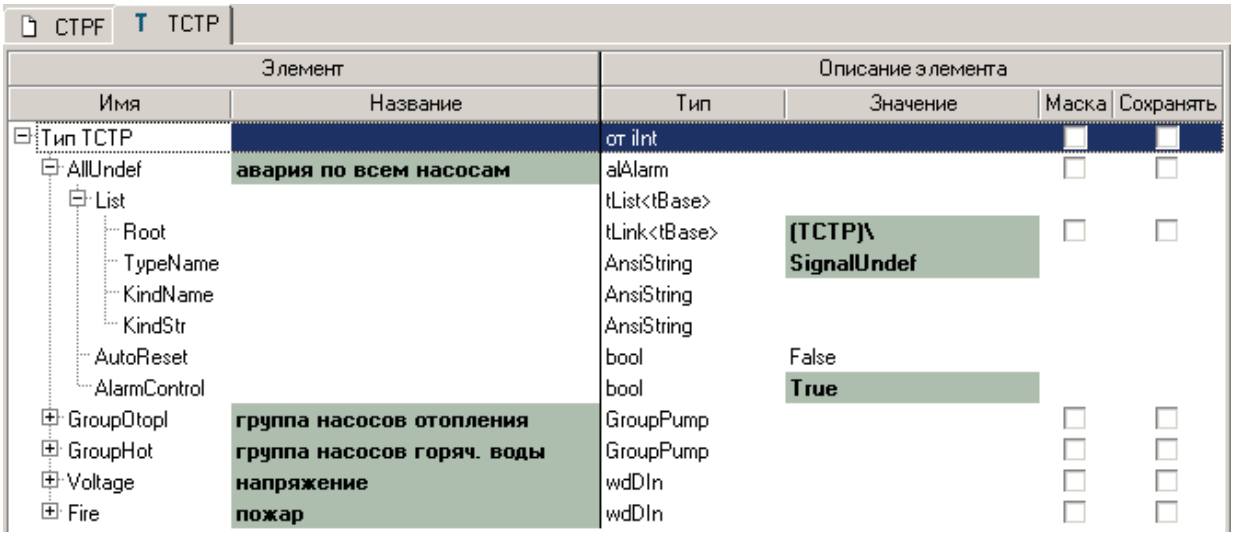


Рис. 58. Тип TCTP.

Следующий шаг: Установка связи модели с базами WinDecont.

10.4.3.8 Установка связи модели с базами WinDecont

После того как все типы созданы и структурированы, необходимо установить связь элементов модели с базами параметров WinDecont. Связь задается для элементов из библиотеки "Элементы баз WD": wdDIn, wdAIn, wdOutImpl, и т.д. с помощью параметра No (номер).

Пусть сигналы описаны в WinDecont следующим образом.

Входные дискреты

2. Состояние насоса горячей воды 1.
3. Состояние насоса горячей воды 2.
5. Состояние насоса отопления 1.
6. Состояние насоса отопления 2.
7. Пожарная сигнализация.

8. Напряжение.

Выходные дискреты

101. Включить насос горячей воды 1.
102. Отключить насос горячей воды 1.
103. Включить насос горячей воды 2.
104. Отключить насос горячей воды 2.
105. Включить насос отопления 1.
106. Отключить насос отопления 1.
107. Включить насос отопления 2.
108. Отключить насос отопления 2.

Входные аналоги

1. Давление горячей воды на входе.
2. Давление горячей воды на выходе.
3. Давление отопления на входе.
4. Давление отопления на выходе.

В окне редактирования типов откройте закладку с типом TCTP (ЦТП).

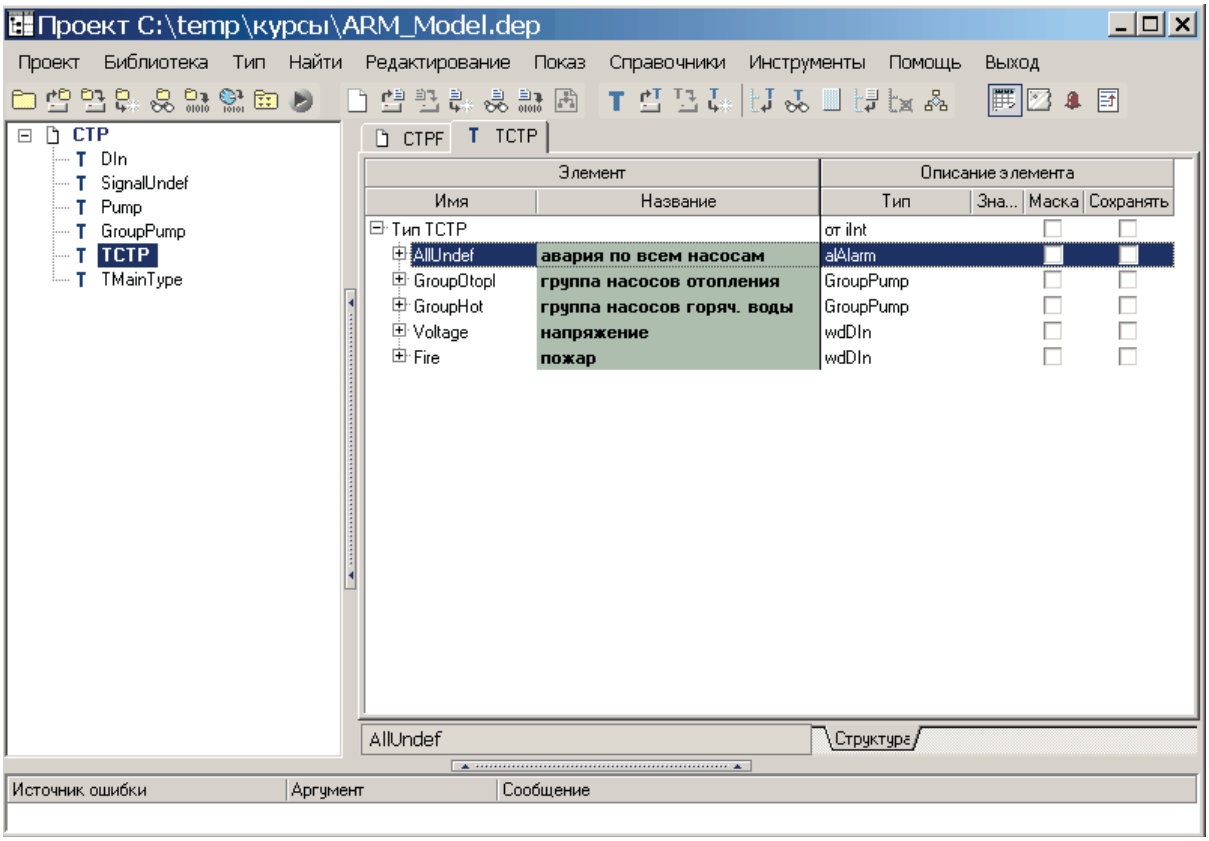


Рис. 59. В окне редактирования типов открыт тип TCTP (ЦТП).

Для быстрого доступа к номерам элементов баз WD можно использовать фильтр строк. Перейдите в меню "Просмотр | Фильтр строк".

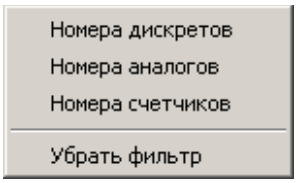


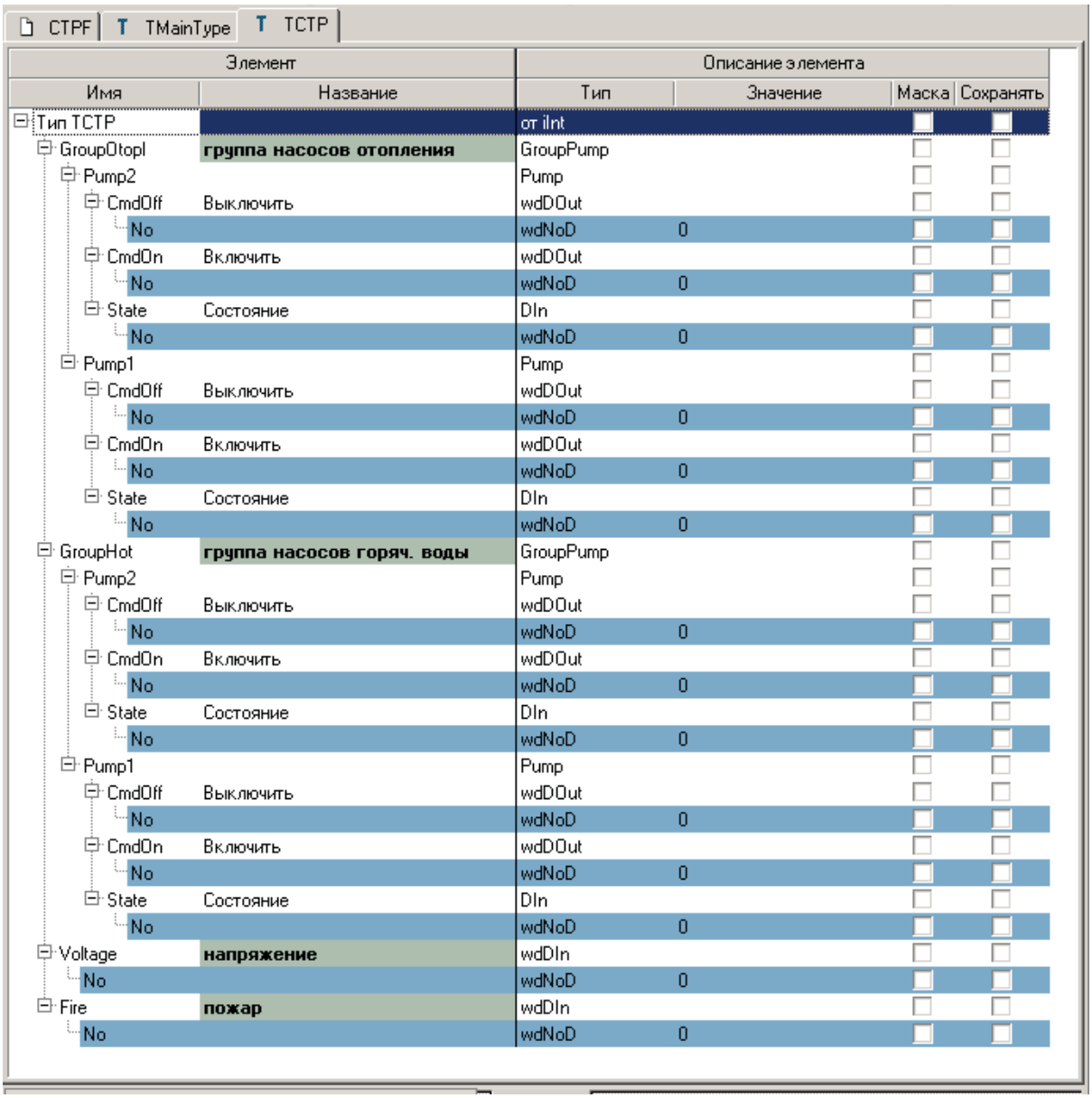
Рис. 60. Меню "Фильтр строк".

Вы можете использовать следующие фильтры:

- **номера дискретов** - показать все поля No (номер) для элементов пронаследованных от типа дискрета (wdDin, wdOut, wdOutImpl).
- **номера аналогов** - показать все поля No (номер) для элементов пронаследованных от типа аналога (wdAIn, wdAOut).
- **номера счетчиков** - показать все поля No (номер) для элементов пронаследованных от типа счетчика (wdCIn, wdCOut).
- **убрать фильтр** - убрать ранее наложенный фильтр.

Если использовать фильтр, то в окне редактирования типа показываются только элементы привязки к базам WinDecont (номер), что заметно сокращает время редактирования.

Используем фильтр "номера дискретов". Выберите меню "Просмотр | Фильтр строк | Номера дискретов". При этом окно редактирования типов для типа ТСТР (ЦТП) примет вид.



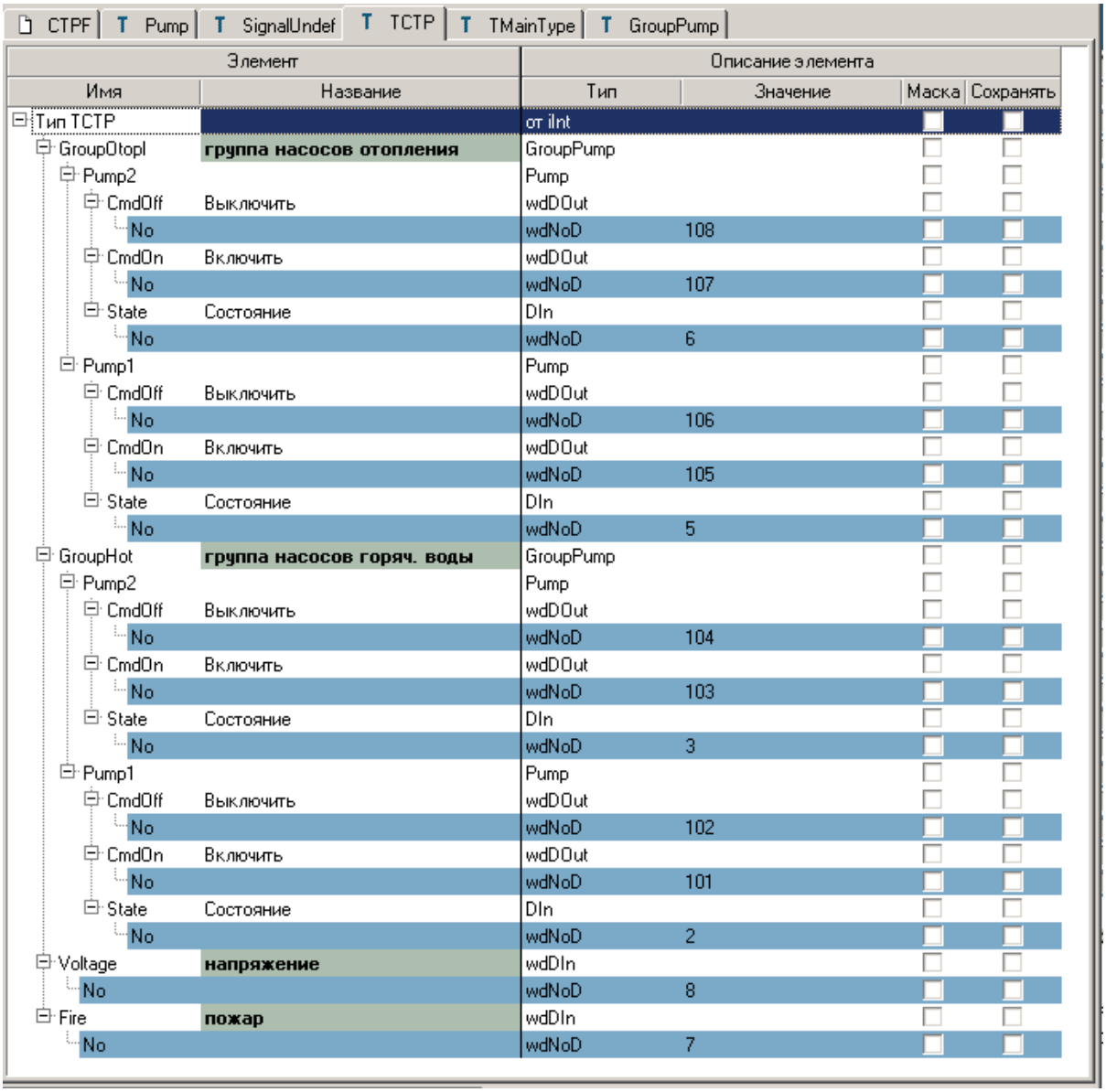
CTPF | TMainType | TCTP

Элемент		Описание элемента			
Имя	Название	Тип	Значение	Маска	Сохранять
Тип TCTP		от iInt			
GroupOtopl	**группа насосов отопления**	GroupPump			
Pump2		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	0		
CmdOn	Включить	wdDOut			
No		wdNoD	0		
State	Состояние	DIn			
No		wdNoD	0		
Pump1		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	0		
CmdOn	Включить	wdDOut			
No		wdNoD	0		
State	Состояние	DIn			
No		wdNoD	0		
GroupHot	**группа насосов горяч. воды**	GroupPump			
Pump2		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	0		
CmdOn	Включить	wdDOut			
No		wdNoD	0		
State	Состояние	DIn			
No		wdNoD	0		
Pump1		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	0		
CmdOn	Включить	wdDOut			
No		wdNoD	0		
State	Состояние	DIn			
No		wdNoD	0		
Voltage	**напряжение**	wdDIn			
No		wdNoD	0		
Fire	**пожар**	wdDIn			
No		wdNoD	0		

Рис. 61. Использование фильтра номеров дискретов для типа TCTP (ЦТП).

Теперь вам надо задать все номера в соответствии с таблицей сигналов WinDecont, приведенной выше. Для этого надо щелкнуть мышкой по полю "Значение" (строка No). Ввести номер (по умолчанию 0) и нажать Enter. Проделаем это на примере состояния первого насоса из группы насосов горячей воды.

В результате получаем:



CTPF | Pump | SignalUndef | TCTP | TMainType | GroupPump

Элемент		Описание элемента			
Имя	Название	Тип	Значение	Маска	Сохранять
Тип TCTP		от iInt			
GroupOtopl	**группа насосов отопления**	GroupPump			
Pump2		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	108		
CmdOn	Включить	wdDOut			
No		wdNoD	107		
State	Состояние	DIn			
No		wdNoD	6		
Pump1		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	106		
CmdOn	Включить	wdDOut			
No		wdNoD	105		
State	Состояние	DIn			
No		wdNoD	5		
GroupHot	**группа насосов горяч. воды**	GroupPump			
Pump2		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	104		
CmdOn	Включить	wdDOut			
No		wdNoD	103		
State	Состояние	DIn			
No		wdNoD	3		
Pump1		Pump			
CmdOff	Выключить	wdDOut			
No		wdNoD	102		
CmdOn	Включить	wdDOut			
No		wdNoD	101		
State	Состояние	DIn			
No		wdNoD	2		
Voltage	**напряжение**	wdDIn			
No		wdNoD	8		
Fire	**пожар**	wdDIn			
No		wdNoD	7		

Рис. 62. Использование фильтра номеров дискретов для типа TCTP (ЦТП).

Щелкните мышкой по корню дерева (Тип TCTP) и уберите фильтр строк "Просмотр | Фильтр строк | Убрать фильтр".

Аналогично проделайте и с аналогами.

Следующий шаг: [Установки проекта](#).

10.4.3.9 Установки проекта

После того, как созданы все типы модели, необходимо произвести установки проекта.

1. Указать главный тип модели.
2. Архивное хранилище (*).
3. Такт модели.

* - необязательно.

Главный тип нашей модели это СТР. Такт модели характеризует, с какой частотой запускается тактовая функция главного типа (частота обновления данных в модели). Зададим такт равный 500 мс.

Щелкните мышкой по значку  (установки проекта). Укажите в диалоге главный тип и такт модели.

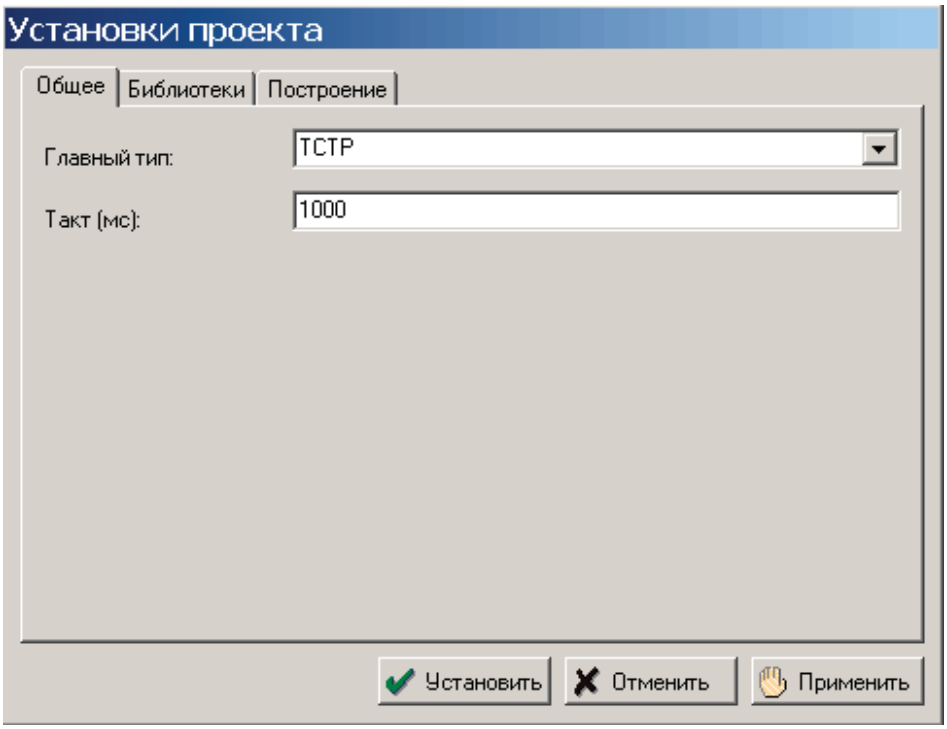


Рис. 63. Установки проекта.

Нажмите "Установить".

Следующий шаг: **Завершение создания модели**.

10.4.3.10 Завершение создания модели

Нам осталось только построить и запустить нашу модель.

Постройте модель  (предварительно сохранив все изменения ).

!!!Не забывайте, что в библиотеке модели типы, которые используются в других типах, должны идти раньше последних !!!

Если все было сделано правильно, в окне сообщений будет написано, что проект собран и ошибок не найдено.

```
18:09:54: Генерация кода C:\alex\works\Manuals\ModelBuilder\Models\ARM\RESULT\ArmLib.h
18:09:54: Генерация кода C:\alex\works\Manuals\ModelBuilder\Models\ARM\RESULT\ArmLib.cpp
18:09:54: Генерация кода C:\alex\works\Manuals\ModelBuilder\Models\ARM\RESULT\Main.h
18:09:54: Генерация кода C:\alex\works\Manuals\ModelBuilder\Models\ARM\RESULT\Main.cpp
18:09:54: Построение MAKE-файла C:\alex\works\Manuals\MODELB~1\Models\ARM\RESULT\tempcmp.mak
18:09:55: Выполнение MAKE-файла ...
MAKE Version 5.2  Copyright (c) 1987, 2000 Borland
        D:\PROGRA~1\Borland\CBUILD~2\BIN\bcc32.exe -WD -O2 -b -k- -vi -H=dml.csm -Vx -Ve -X- -a8 -tWD -tWM -c -D_DEBUG;NO_STRICT -ID:\PROGRA~1\
Borland C++ 5.6 for Win32 Copyright (c) 1993, 2002 Borland
C:\alex\works\Manuals\MODELB~1\Models\ARM\RESULT\ArmLib.cpp:
C:\alex\works\Manuals\MODELB~1\Models\ARM\RESULT\Main.cpp:
Loaded pre-compiled headers.

18:10:02: Выполнение MAKE-файла завершено
18:10:02: Построение LINK-файла
18:10:02: Сборка проекта ...
Turbo Incremental Link 5.60 Copyright (c) 1997-2002 Borland

18:10:04: Сборка проекта завершена
```

Добавьте модель в список моделей WinDecont. Для этого откройте программу WinDecont.

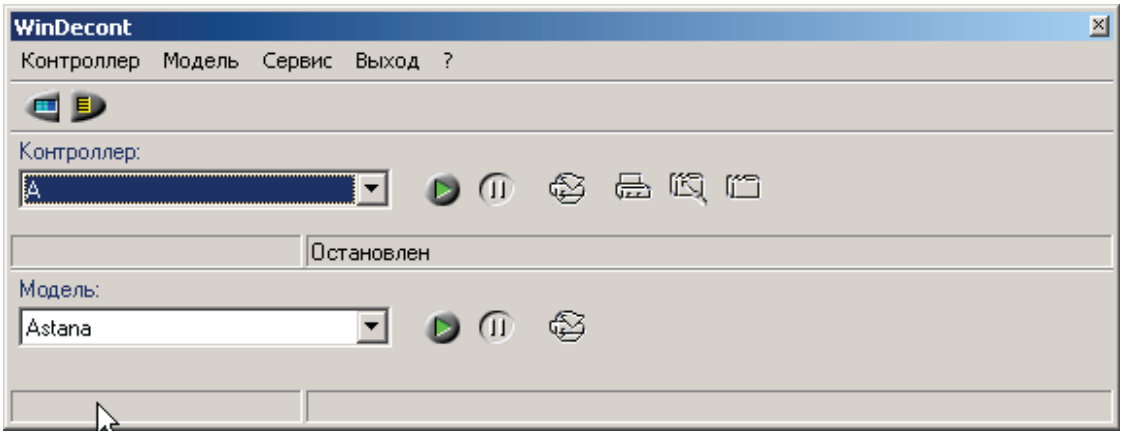


Рис. 64. WinDecont.

Откройте окно параметров WinDecont (меню Сервис | Параметры).

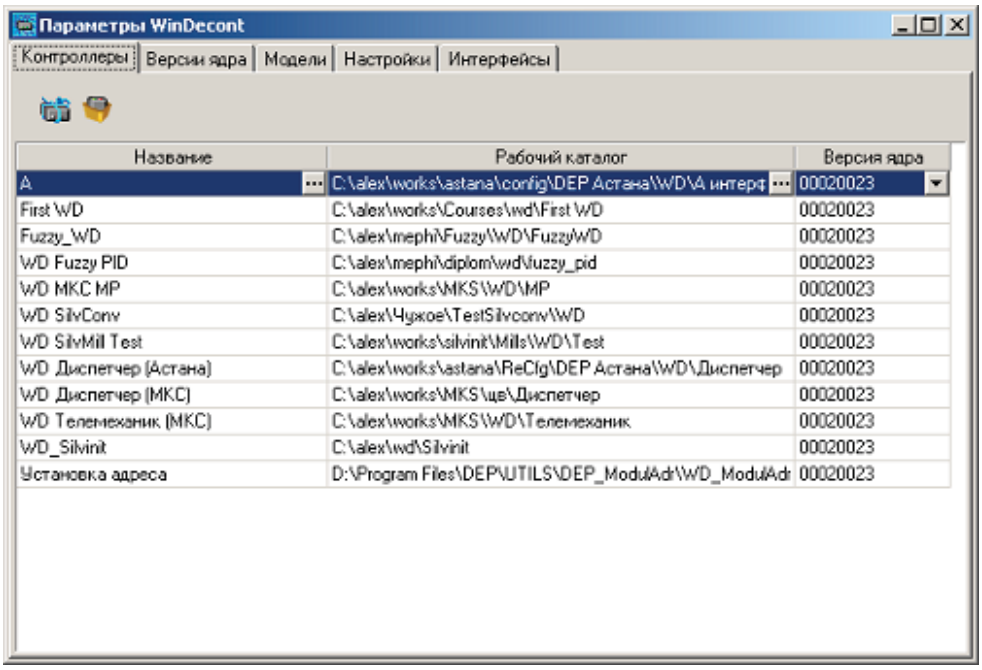


Рис. 65. Параметры WinDecont.

Откройте закладку "Модели", щелкните мышкой по значку  (установить новую модель). Напишите название модели (АРМ) и укажите путь к файлу Model.dll (...| ваша модель | RESULT).

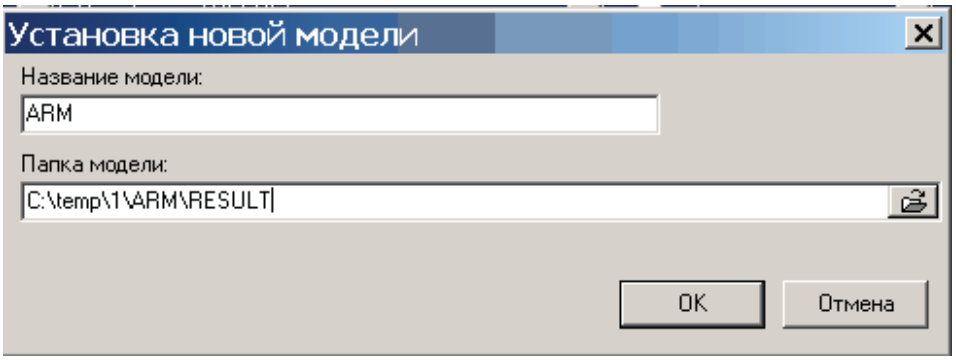


Рис. 66. Установка новой модели.

Закройте окно параметров WinDecont. Выберите модель АРМ из списка и нажмите  (запустить модель).

Все! Модель создана и запущена. Пора заняться отображением.

Следующий шаг: **Создание отображения**.

10.4.4 Создание отображения

В данном разделе будет показано, как нарисовать изображение технологического экрана в программе "Inkscape".
Начнем со знакомства с программой "Inkscape".

10.4.4.1 Знакомство с Inkscape

Для создания технологического экрана необходимо сначала создать изображение этого экрана.

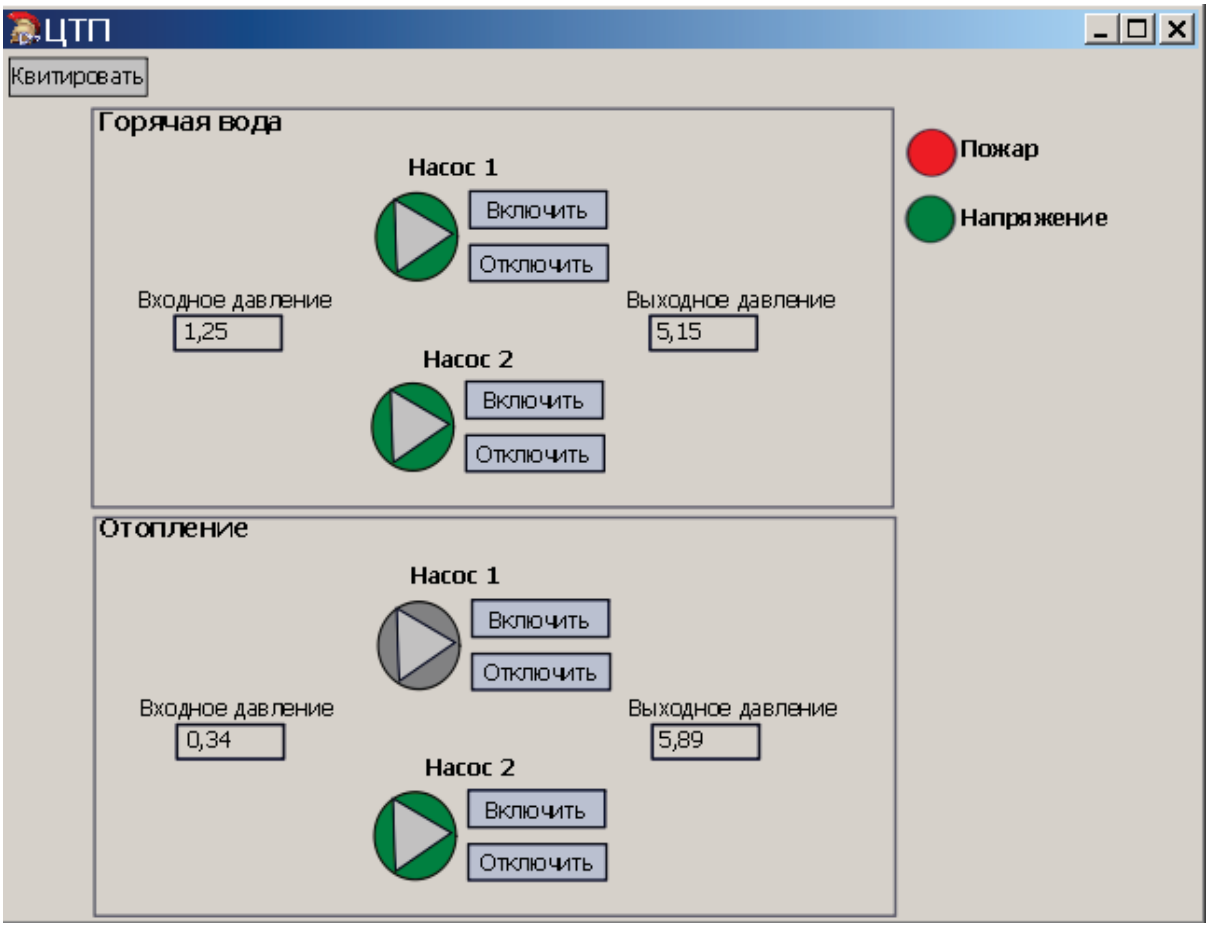


Рис. 67. Вид технологического экрана.

Для построения отображения будем использовать программу **Inkscape**. Запустите ее.
Перед Вами будет следующее приложение

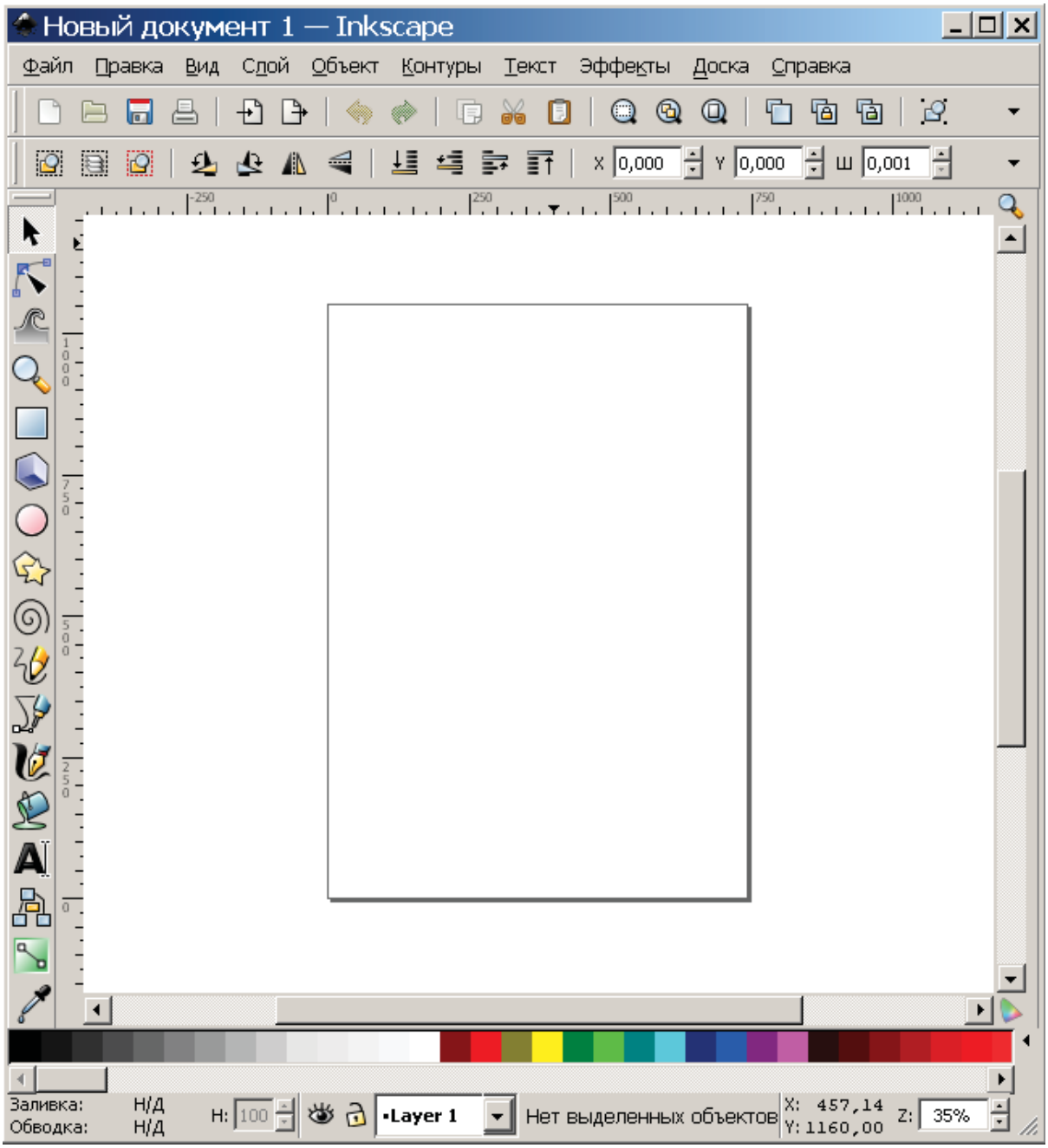


Рис. 68. Программа Inkscape.

Сохраните текущий документ командой File->Save as..

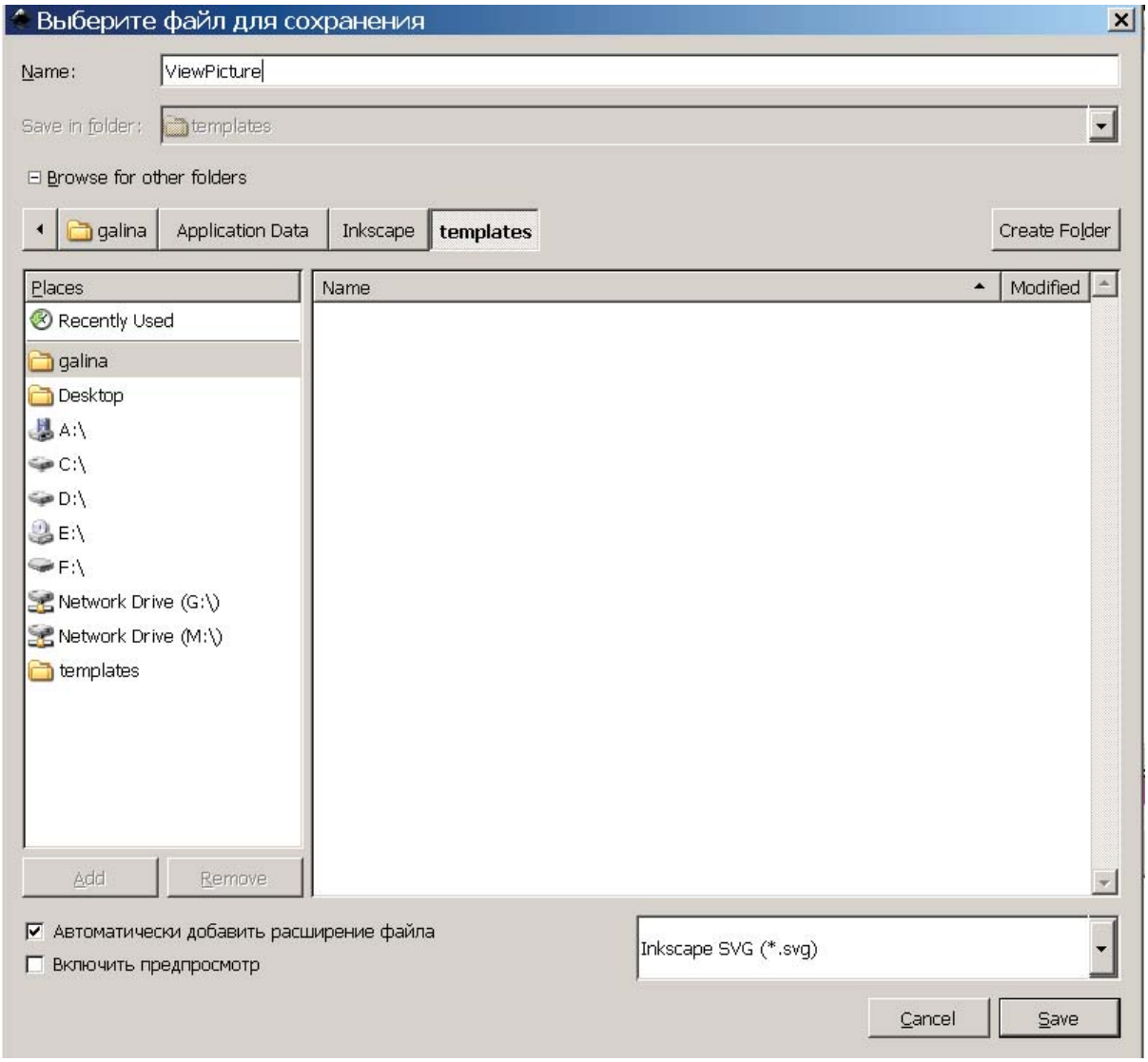


Рис. 69. Сохранение документа Inkscape.

Следующий шаг: [Работа с Inkscape](#).

10.4.4.2 Работа с Inkscape

Построение изображения в программе **Inkscape** происходит аналогично рисованию картинки в графическом редакторе.

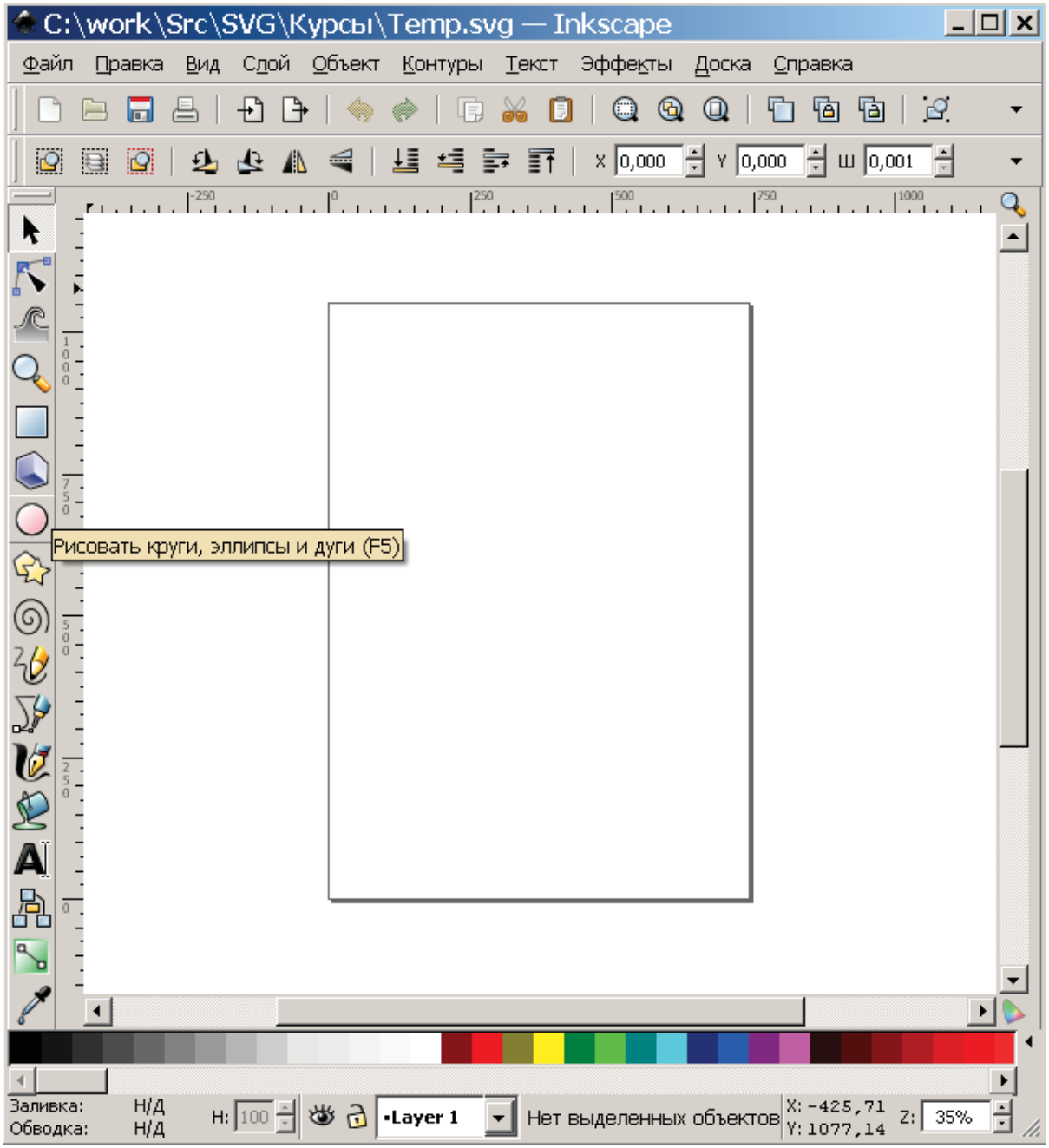


Рис. 70. Программа Inkscape.

Элементы для рисования представлены справа на панели. Если пройтись по каждому элементу и навести мышкой, то всплывут подсказки. По этим подсказкам можно понять, что делает данный элемент.

Рисовать необходимо в области, обведенной в прямоугольник. Размер данной области можно поменять через меню Файл->Свойства документа..

В данной работе от нас требуется нарисовать технологический экран (рис.1 из Знакомство с Inkscape).

Начнем рисовать элемент 

. Для этого нам потребуется:

- для текста -  . Редактировать шрифт можно из меню Текст->Текст и шрифт..

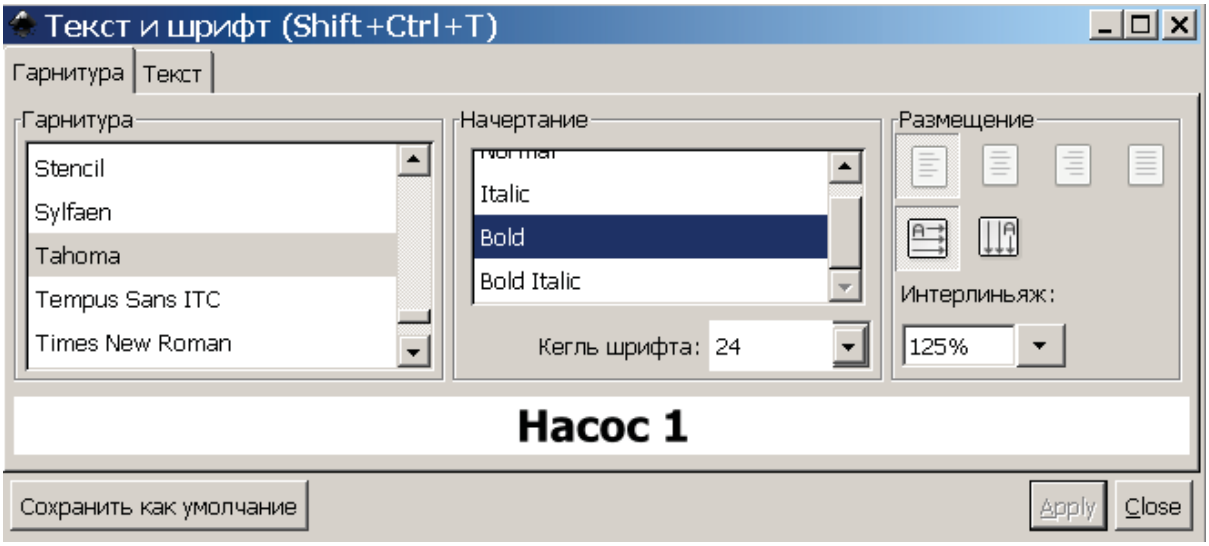


Рис. 71. Изменение шрифта текста.

- для прямоугольника -  .

Чтобы задать заливку, контур и др. надо щелкнуть мышкой на панели заливка-обводка.

.

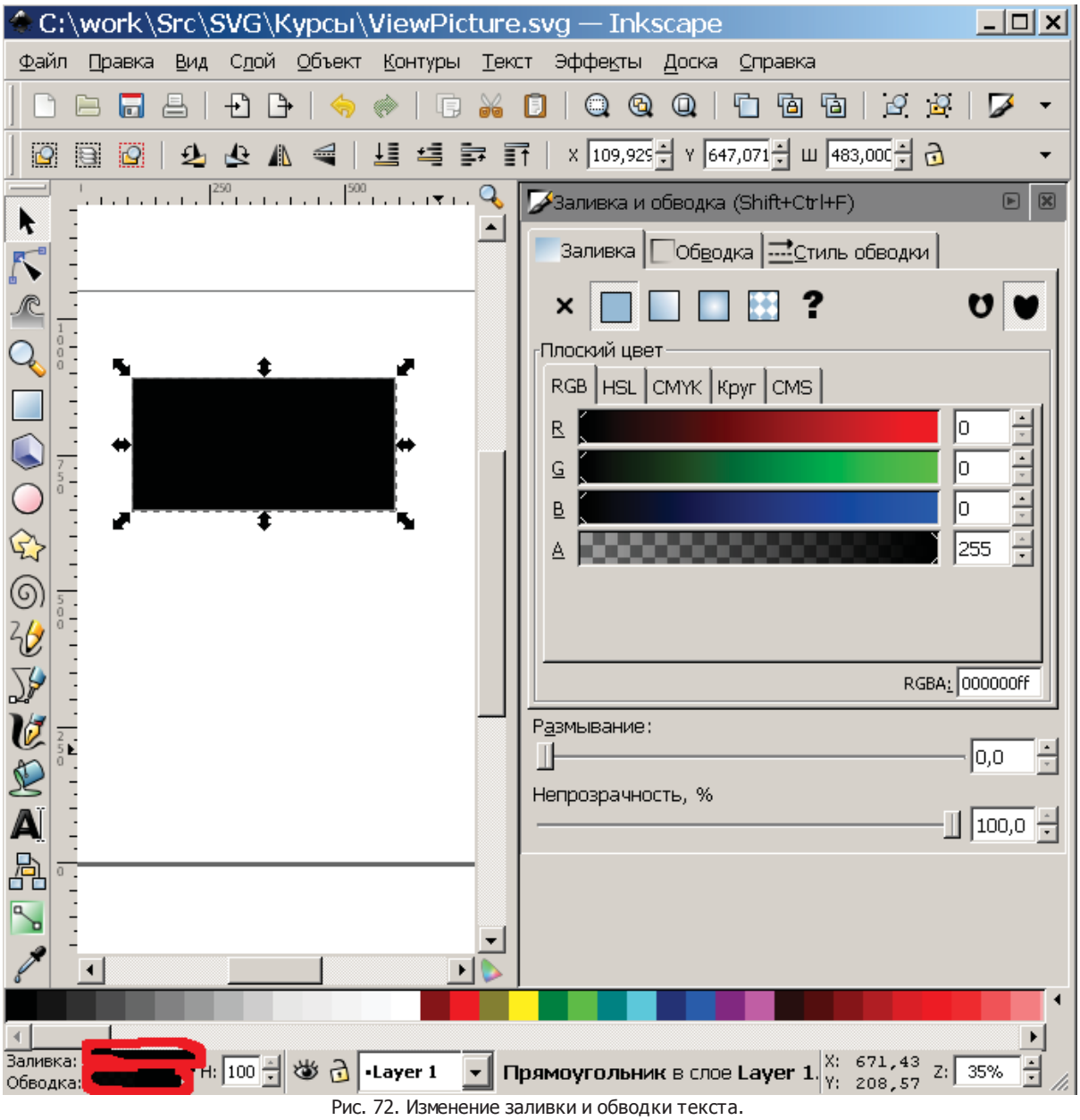


Рис. 72. Изменение заливки и обводки текста.

Аналогично заливку и обводку можно поменять через меню Объект->Заливка и обводка..

- для круга -  . Заливка и обводка меняется аналогично, как и при рисовании прямоугольника.

- для треугольника -  . Заливка и обводка меняется аналогично, как и при рисовании прямоугольника.

В результате получаем следующее изображение:

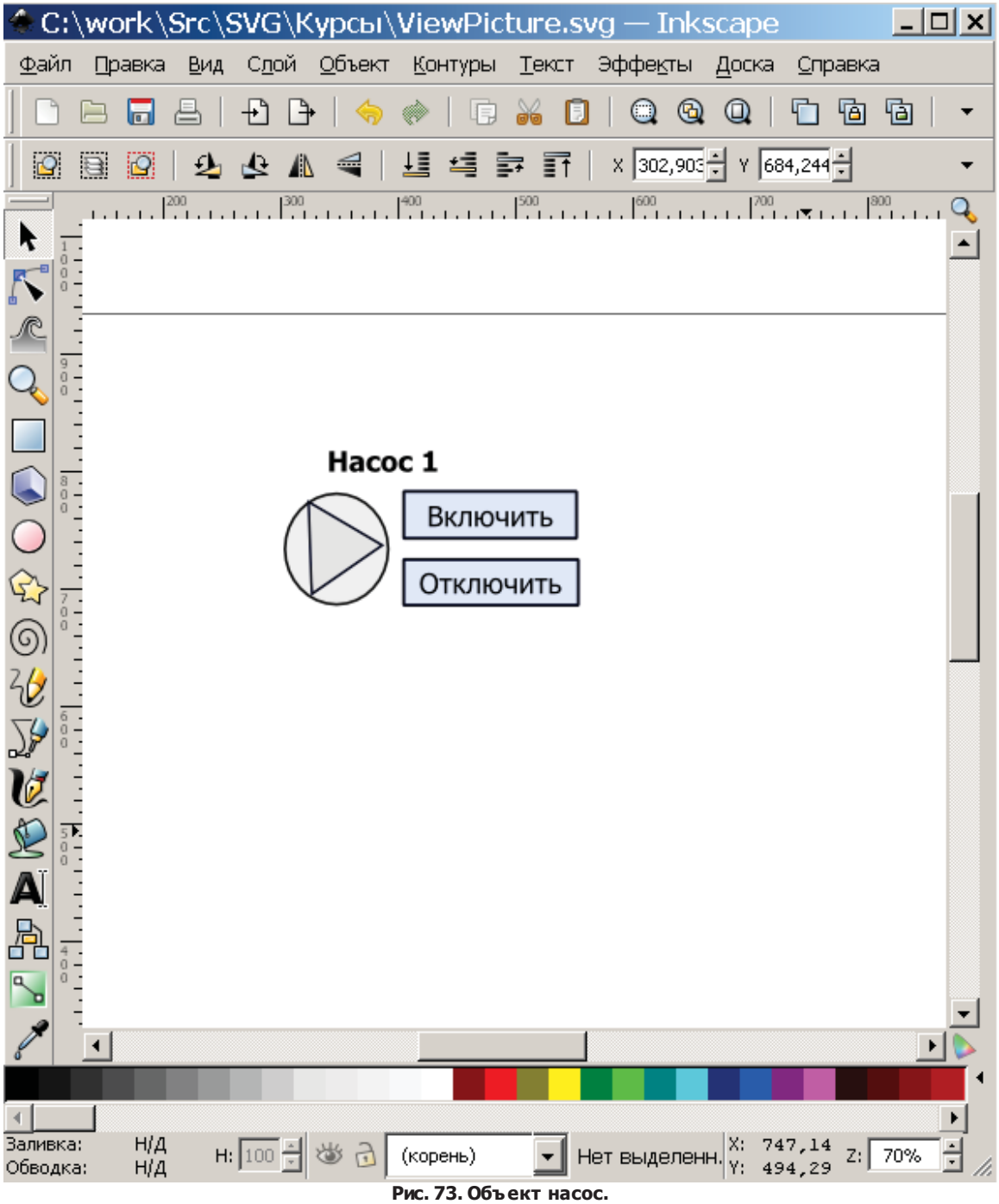


Рис. 73. Объект насос.

В Inkscape существуют два очень важных понятия:

- Группировка

Несколько объектов могут быть объединены в группу. При перемещении и трансформации группа ведет себя как обычный объект.

Для создания группы нужно выбрать один или более объектов и нажать Ctrl+G. Разгруппировать их можно, нажав Ctrl+U, предварительно выбрав группу. Сами по себе группы могут быть сгруппированы и как одиночные объекты. Подобная поэтапная группировка может быть сколь угодно сложной. При этом следует помнить, что Ctrl+U разгруппирует только

последнюю группировку. Нужно нажать Ctrl+U несколько раз, если вы хотите полностью разгруппировать сложно сгруппированные объекты.

Очень удобно, что вам не нужно разбивать группу для редактирования отдельных объектов. Выполнив Ctrl+щелчок по объекту, вы его выберете и сможете редактировать. Таким же образом работает комбинация Shift+Ctrl+щелчок, позволяющая редактировать несколько объектов независимо от группы.

- Клон.

Любой объект может иметь клона. Клон обладает всеми свойствами оригинала.

При изменении оригинала - меняется и клон. Свойства клона, кроме положения и трансформации, менять нельзя.

Для создания клона выделите объект и нажмите Alt+D.

Если необходимо отсоединить - нажмите - Shift + Alt + D. При этом клон становится просто копией соответствующего объекта.

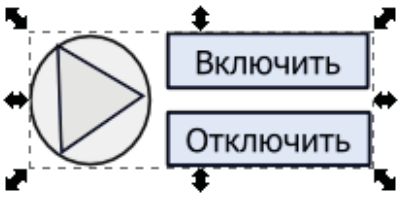
В следующей главе мы познакомимся с этими понятиями более подробно.

Следующий шаг: Создание отображения.

10.4.4.3 Создание отображения

В разделе Работа с Inkscape мы нарисовали простейшую часть АРМа - один из насосов.

Выделим часть этого отображения (без надписи) и сгруппируем через меню Объект->Сгруппировать (или нажатием комбинации Ctrl + G).



В результате сгруппированные объекты ведут себя как единый объект. Для того, чтобы выделить объект, входящий в группу, необходимо несколько раз щелкнуть на него мышкой по очереди выделяя все вложения или по комбинации Ctrl+щелчок по объекту.

Далее создадим клон этого объекта. Выделим объект и через меню Правка->Клоны->Создать клон (или нажатием комбинации Alt + D) создадим клона, то есть копию данного объета. Теперь, если что-либо поменяется в оригинале, то изменения отразятся и на клоне.

Отдельно создадим надпись - Насос2.

Надписи создавались отдельно, чтобы их названия можно было менять, так как в клонах нельзя менять объект независимо от оригинала!!

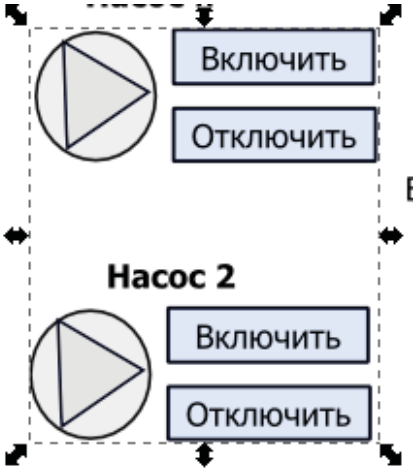
В результате получаем

В нашем ЦТП - это группа насосов горячей воды.

Как уже упоминалось, все наши нарисованные объекты хранятся в XML - формате, в котором каждый объект имеет имя. Например, для групп "Насос 1" и "Насос 2" можно задать новые имена. Это делается щелчком правой кнопки мыши и выбором "Свойства объекта" - ID или через XML дерево проекта. Более подробно это описано на следующем шаге: Задание имен объектов в Inkscape.

Необходимо помнить, что изменение имени объекта, при наличии у него клонов, ведет к потере связи клонов с родительским объектом и пропаданию самих клонов. Следовательно, если требуется поменять имя объекта, делать это нужно только до создания клонов этого объекта.

Чтобы объекты вели себя, как единое целое, выделим их и сгруппируем, как описано ранее.



Добавим справа - входное давление и слева - выходное давление. **Поле давления помимо прямоугольника имеет также текстовое поле.**

Нарисуем вокруг полученной группы прямоугольник и сделаем надпись - Горячая вода.

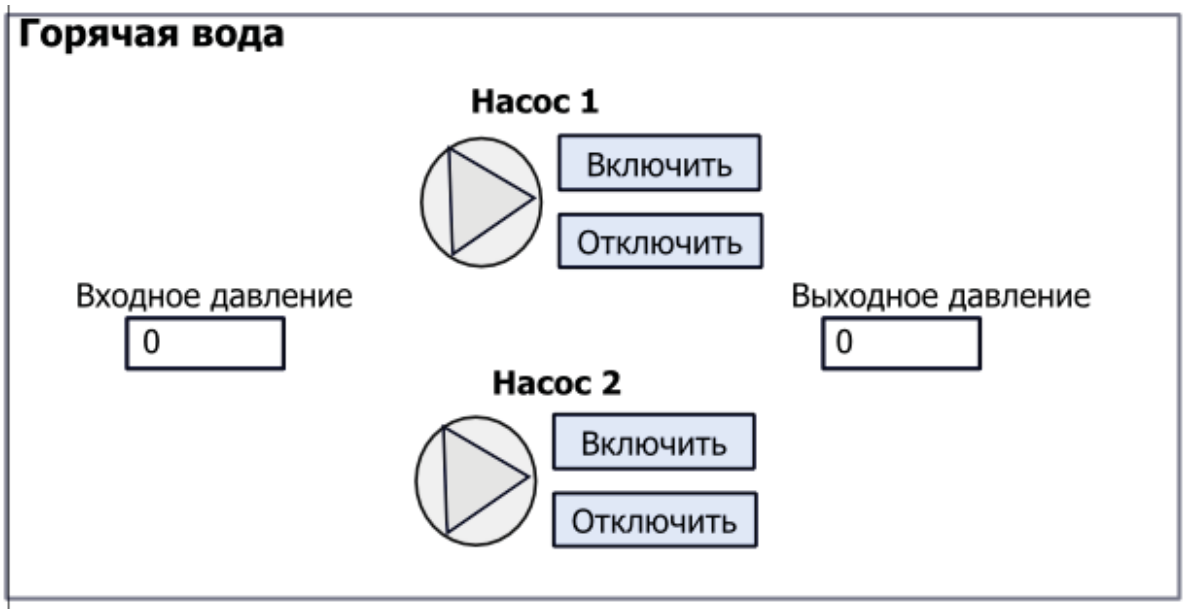


Рис. 74. Группа насосов горячей воды.

Аналогично создадим клона - группу отопления.

В результате получаем

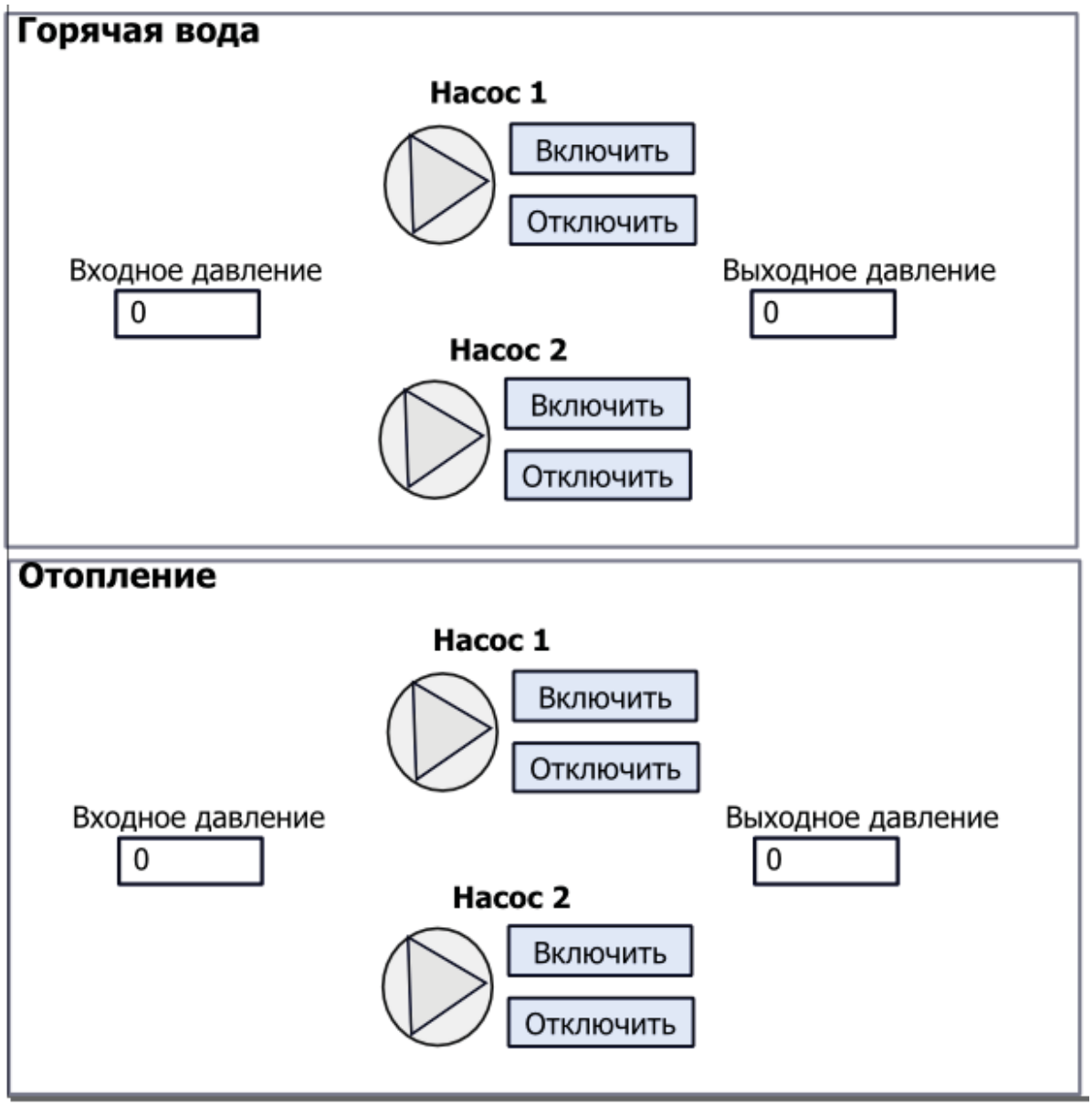


Рис. 75. Группа насосов горячей воды и отопления.

Нарисуем кнопку "Квитировать" и индикацию пожара и напряжения.

В результате получим:

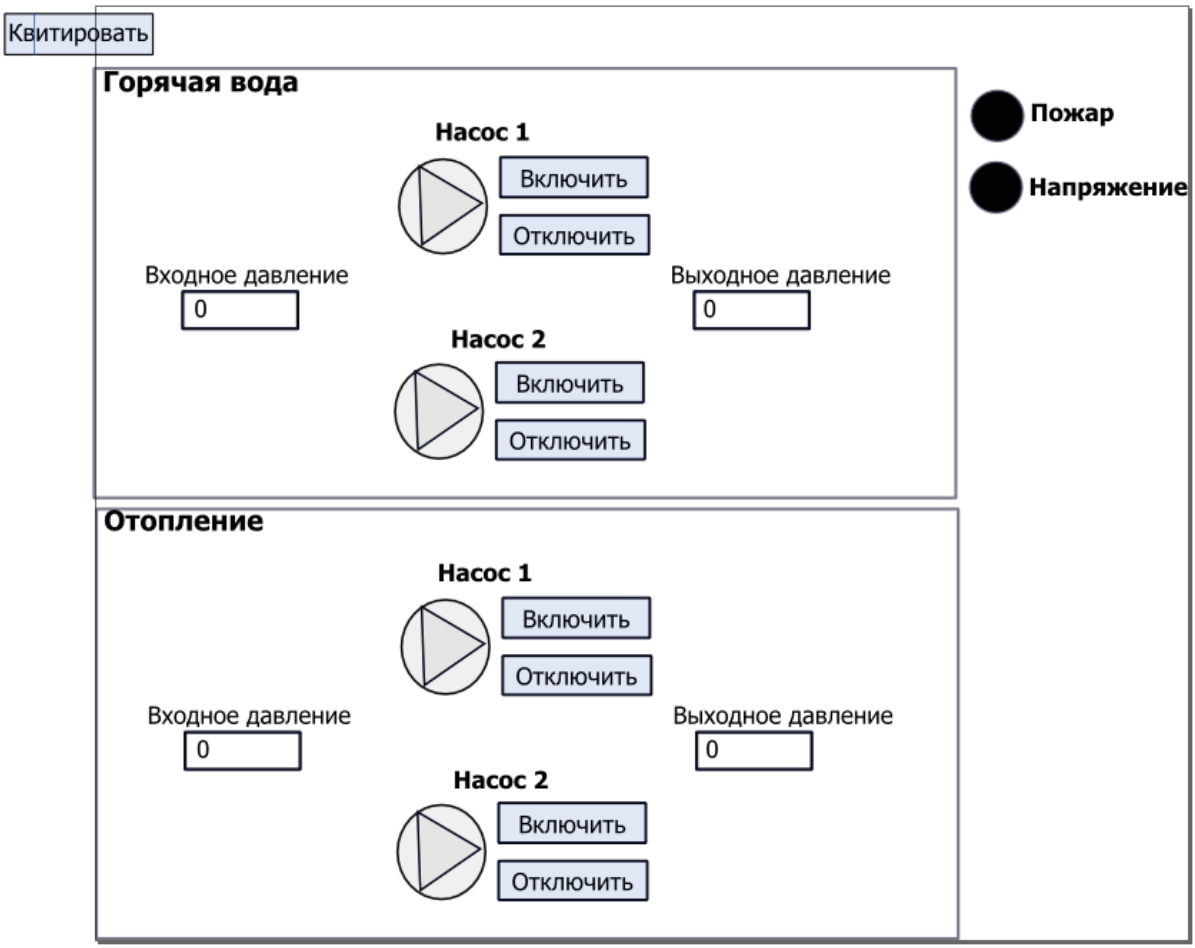


Рис. 76. АРМ - диспетчера.

В этом разделе мы закончили создавать отображение нашего технологического экрана.

Следующий шаг: **Задание имен объектов в Inkscape**.

10.4.4.4 Задание имен объектов в Inkscape

Теперь немного познакомимся с внутренней структурой объектов в Inkscape.

Объекты, которые мы рисуем в программе Inkscape имеют гибкую структуру, которая хранится в XML - файлах.

Посмотреть ее можно нажав  (или нажатием комбинации Shift+Ctrl+X, или через меню - Правка->Редактор XML..).

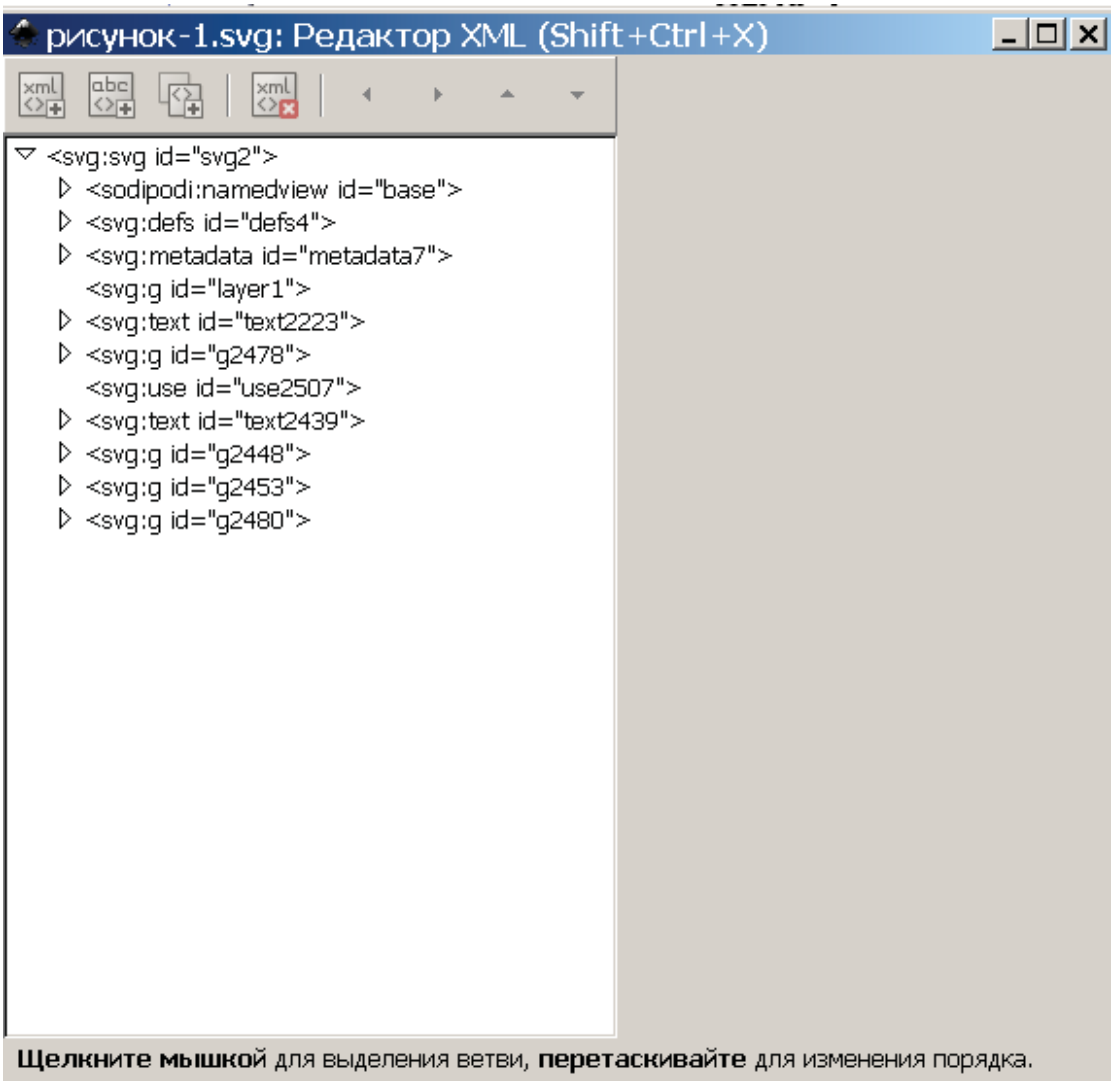


Рис. 77. Редактор XML.

Если пройтись по структуре слева данного редактора, то можно заметить, что мы проходимся по элементам того, что мы нарисовали в программе Inkscape. Текущие объекты выделяются.

Давайте посмотрим, как формируются имена объектов:

- Например, <svg:text id="text2223"> означает текстовое поле с номером 2223

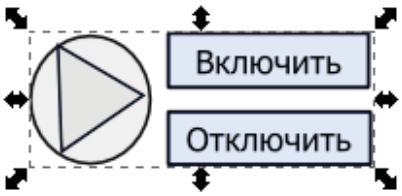
- Например, ▷ <svg:g id="g2478"> означает группу с номером 2478. Чтобы посмотреть элементы, которые входят в эту группа необходимо нажать ▷.

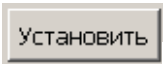
- Например, <svg:rect id="rect6874"> означает прямоугольную область с номером 6874

- Например, <svg:use id="use2507"> означает клона оригинала с номером 2507

и т.д.

Для того, чтобы нам было более удобно в дальнейшем идентифицировать нужные объекты можно менять их имена.

Найдем в XML - редакторе группу: 

и зададим название Nasos1. В поле id отображается текущее название. Внизу в поле можно ввести новое название и кнопкой 

записать значение. Аналогично можно сделать и для насоса 2.

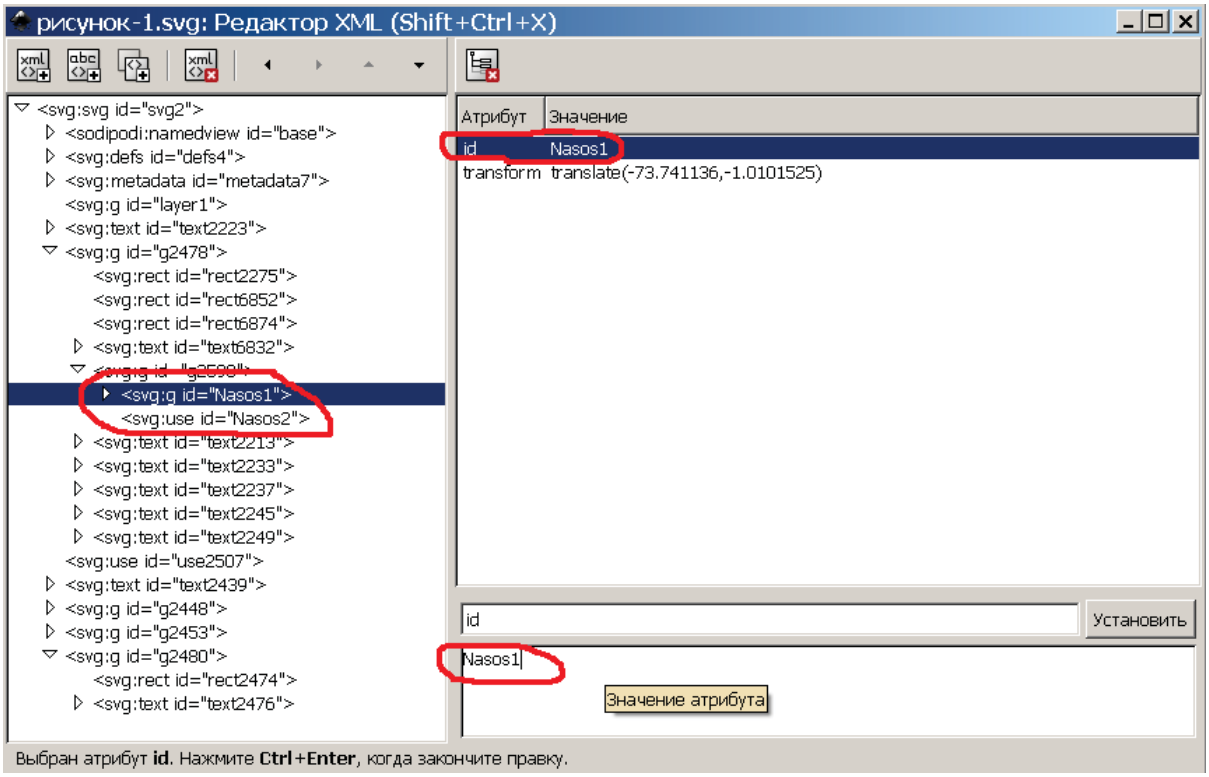


Рис. 78. Задание имени объекту.

Это мы сделали для насосов горячей воды. Для насосов отопления мы не можем поменять имена отдельных объектов, так как эта группа является клоном. Можно только изменить имя самого клона.

Аналогично можно задать имена для остальных элементов (для входного давления, пожара и т.д.).

Следующий шаг: **Создание проекта АРМ**.

10.4.5 Создание приложения АРМ

В данном разделе будет показано, как создать приложение АРМ, загрузить в него изображение технологического экрана,

созданного в программе "Inkscape", и связать его с OPC-моделью.

Начнем с создания проекта АРМ.

10.4.5.1 Создание проекта АРМ

Создадим специальную директорию для АРМа. Например, выберем путь *E:\Temp* и создадим папку ARM. Теперь скопируем файл *Arm-Builder.exe* в папку *E:\Temp\ARM*. Создадим ярлык для файла *Arm-Builder.exe* - щелкаем правой кнопкой мыши на *Arm-Builder.exe -> Создать ярлык*. Переименуем созданный ярлык в *Cfg.lnk*. Изменим свойства ярлыка - щелкаем правой кнопкой мыши на *Cfg.lnk -> Свойства*. В поле *Объект* добавляем символы **/cfg**, нажимаем *ОК*:

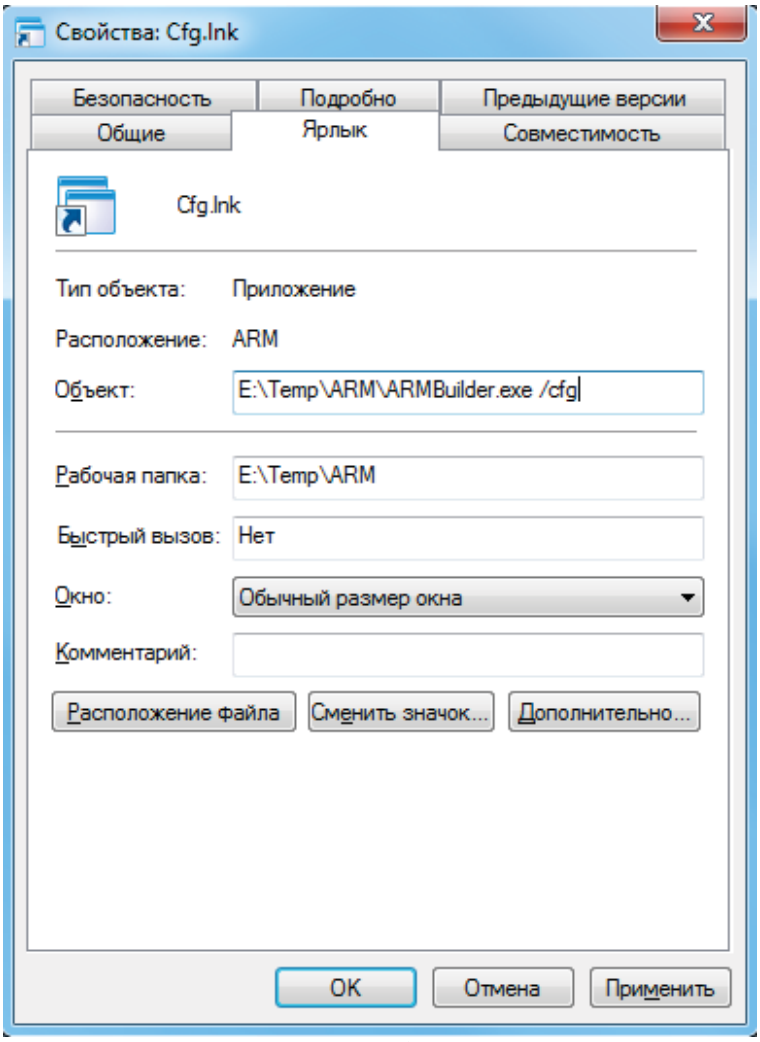


Рис. 79

Далее запустим *Cfg.lnk*. Выберем пункт меню *Проект -> Создать проект*. Сохраним проект с помощью меню *Проект -> Сохранить проект*. В диалоге сохранения создадим новую папку, например, *E:\Temp\ARM\Data*. Выберем эту папку, введем название проекта, например, *arm*, и нажмем *Сохранить*. На рисунке показано главное окно программы, как оно должно выглядеть после описанных действий:

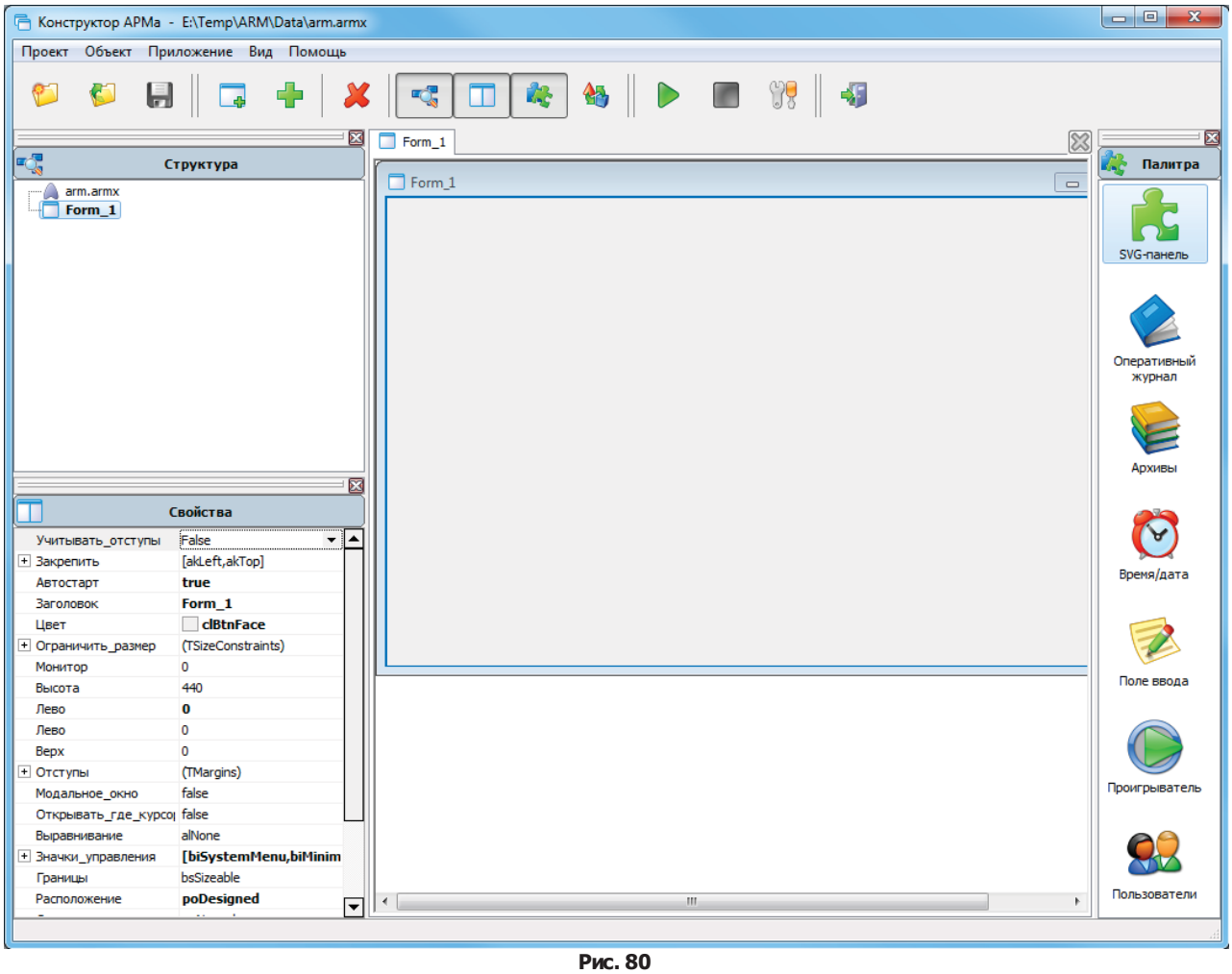


Рис. 80

После того как мы создали и сохранили проект, в директории *E:\Temp\ARM* должны быть следующие файлы и папки:

- папка *Data* - в ней содержится файл проекта *arm.armx* и служебные папки проекта *Static* и *Dynamic*;
- файл *Arm-Builder.exe* - исполняемый файл АРМа;
- ярлык *Cfg.lnk* - файл для запуска программы в конфигурационном режиме;
- файл *ProjectPath.txt* - в нем записан путь к проекту, который считается рабочим (проект АРМа).

Следующий шаг: **Загрузка изображений**.

10.4.5.2 Загрузка изображений

Чтобы использовать файл с изображением технологического экрана, созданный в разделе Создание отображения, создадим внутри папки *Data* отдельную папку *Svg*, и скопируем туда файл с изображением (в нашем случае это *ViewPicture.svg*). Можно оставить файл и там, где он есть, но рекомендуется сохранять все файлы, необходимые для работы АРМ, в одном месте (в

нашем случае это папка *Data*), чтобы потом проект можно было легко перенести на другой компьютер.

В проекте, который мы только что создали в программе ARMBuilder, уже есть одно окно (под названием *Form_1*). Это окно - единственное окно в нашем проекте, и оно является главным. Его мы и будем использовать для размещения изображения технологического экрана.

Добавим новый компонент SVG-панель из палитры компонентов (ему будет автоматически присвоено имя *SvgPanel_1*). В редакторе статических свойств найдем свойство **Выравнивание**, и установим его значение в *alClient*. Таким образом, панель займет все предоставленное ей в окне пространство.

Выберем наш файл с изображением технологического экрана для отображения в SVG-панели. Для этого найдем у панели свойство **Имя_файла** и, нажав на кнопку ..., выберем в открывшемся диалоге наш файл с изображением (в данном случае это *E:\Temp\ARM\Data\Svg\ViewPicture.svg*).

Перейдем к редактированию свойств окна (*Form_1*), выбрав его в списке "Структура". Установим нужные размеры (свойства **Ширина** и **Высота**), а также присвоим значение *ЦТП* свойству **Заголовок**. Свойству **Расположение** установим значение *poScreenCenter*, чтобы наше окно было показано в центре экрана.

Сохраним изменения, сделанные в проекте, нажав на кнопку  , или использовав комбинацию клавиш Ctrl+S. Теперь окно программы будет выглядеть примерно так:

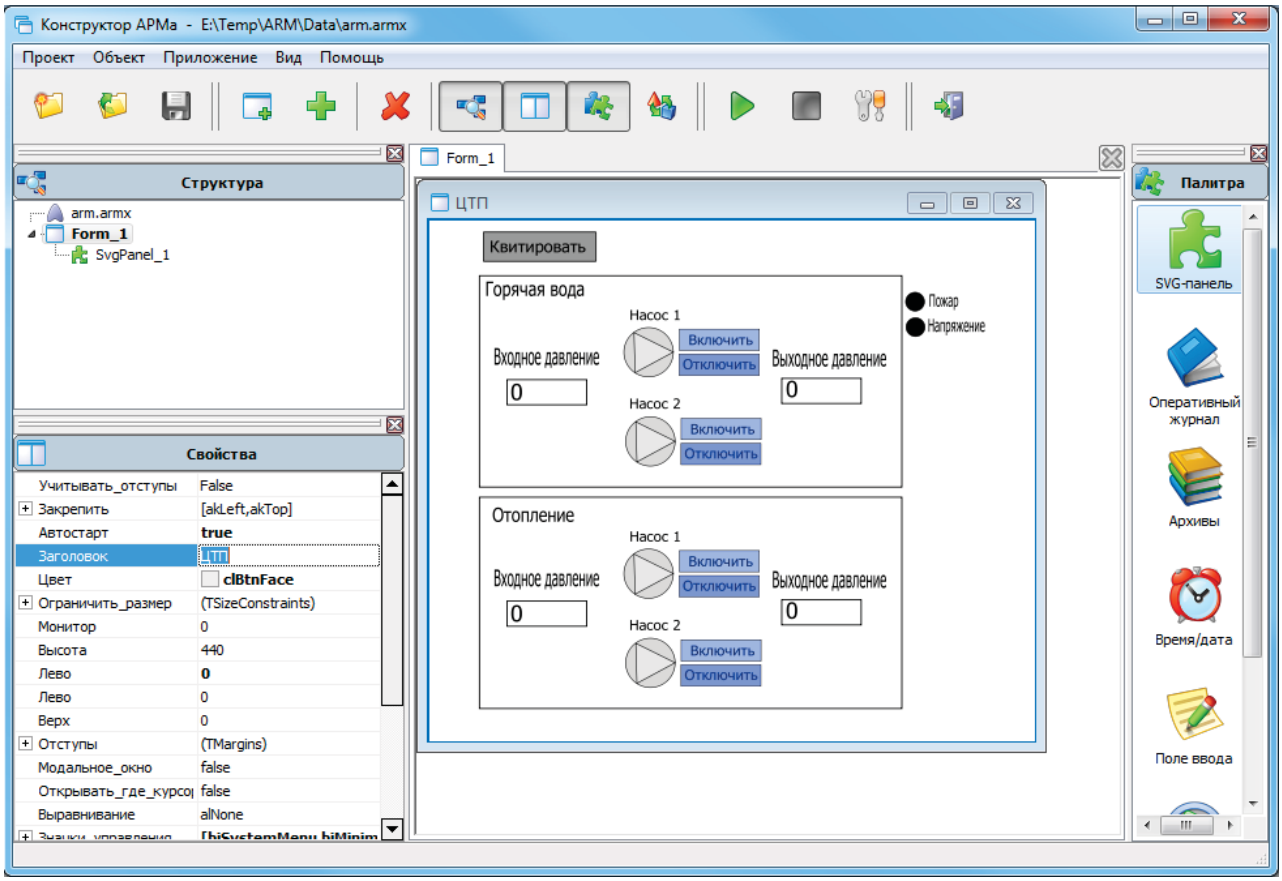


Рис. 81

Можно запустить АРМ на выполнение, нажав кнопку  или клавишу F9. На экране появится окно с изображением технологического экрана. Пока никакой информации этот технологический экран не отображает. Закройте окно; поскольку это окно является главным в проекте, его закрытие приведет к закрытию всего приложения АРМ.

Наступило время связать наш АРМ с работой OPC-модели.

Следующий шаг: Использование редактора динамизации.

10.4.5.3 Использование редактора динамизации

Для того чтобы связать АРМ с OPC-моделью, мы должны создать и заполнить "OPC-привязки". Привязка элемента АРМ к элементу OPC-модели позволяет либо менять свойства элемента АРМ в зависимости от состояния элемента модели, либо наоборот, записывать из АРМ в элементы модели те или иные значения.

Привязки создаются и редактируются в программе ARMBuilder с помощью редактора динамических свойств.

Выберем элемент SvgPanel_1 и откроем редактор динамических свойств с помощью кнопки  или комбинации клавиш Shift+F3. Нажимая левой кнопкой мыши на различные элементы изображения технологического экрана, можно наблюдать, как эти элементы выделяются в редакторе:

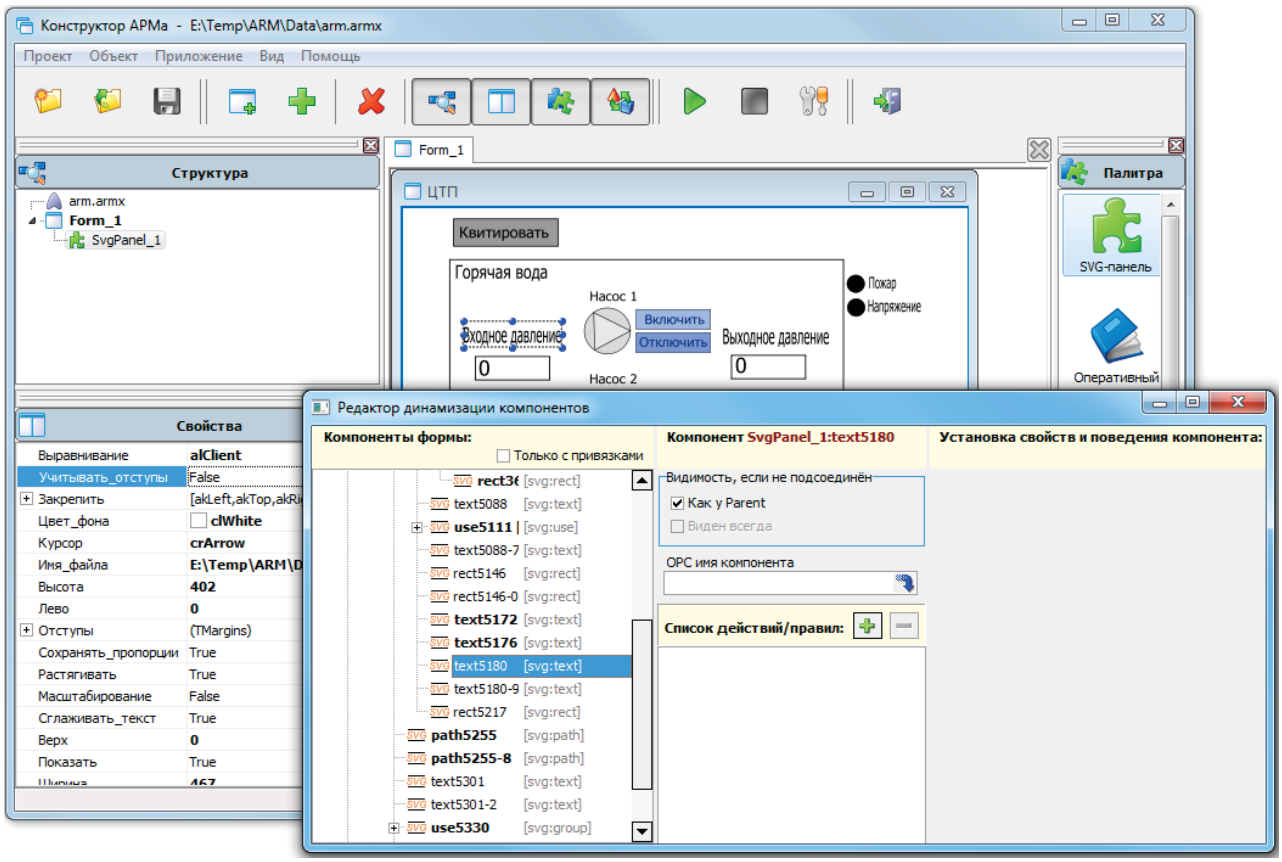
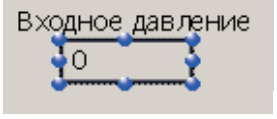


Рис. 82

Далее мы создадим привязки к необходимым элементам технологического экрана.

Следующий шаг: Привязка давлений.

10.4.5.4 Привязка давлений

Начнем привязку с текста, отвечающего за отображение значения входного давления для насосов горячей воды. Щелкните на элемент 

и посмотрите на "Редактор динамизации"

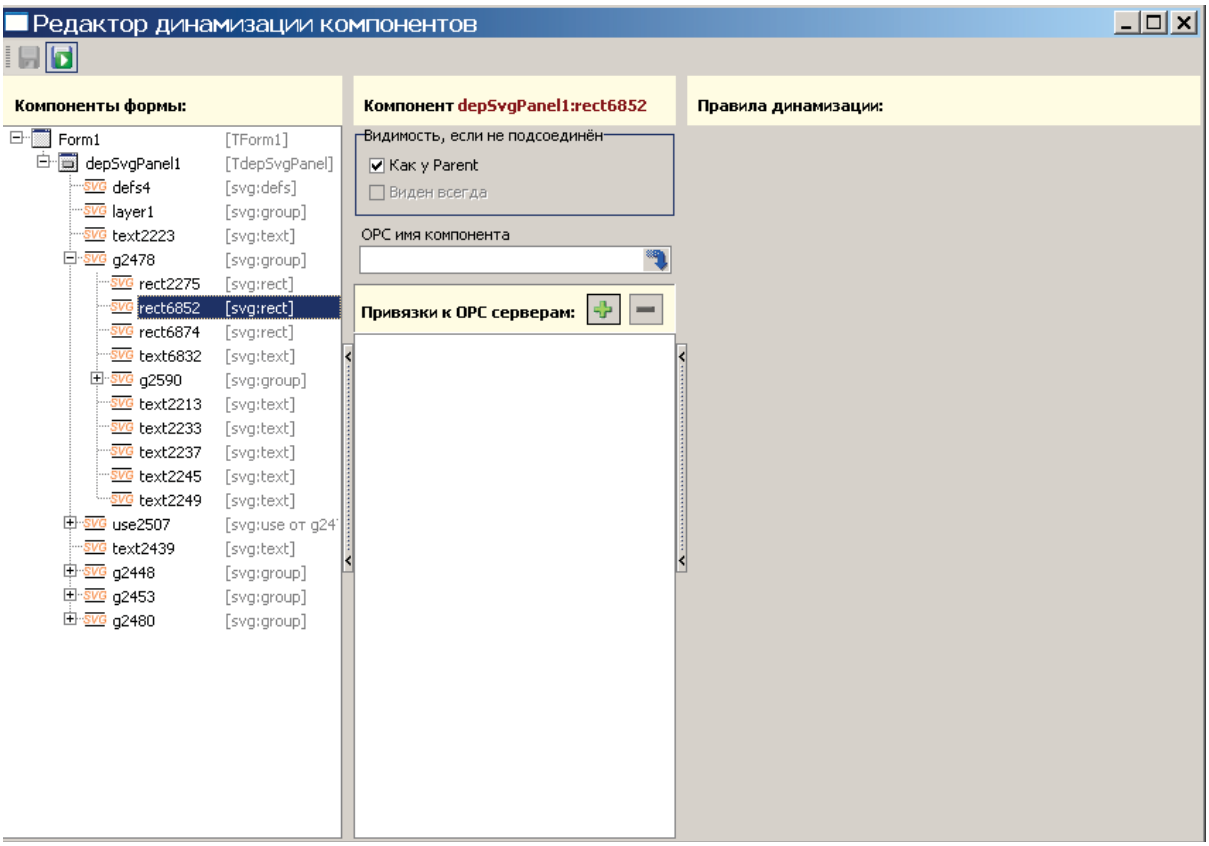


Рис. 83. Редактор динамизации компонентов.

Синим цветом, в дереве объектов формы, выделен наш объект.

Мы хотим, чтобы в этой области отображалось текущее входное давление. Выделенный элемент - это просто прямоугольник. Нам необходимо выделить текстовое поле, в котором выводится значение 0.

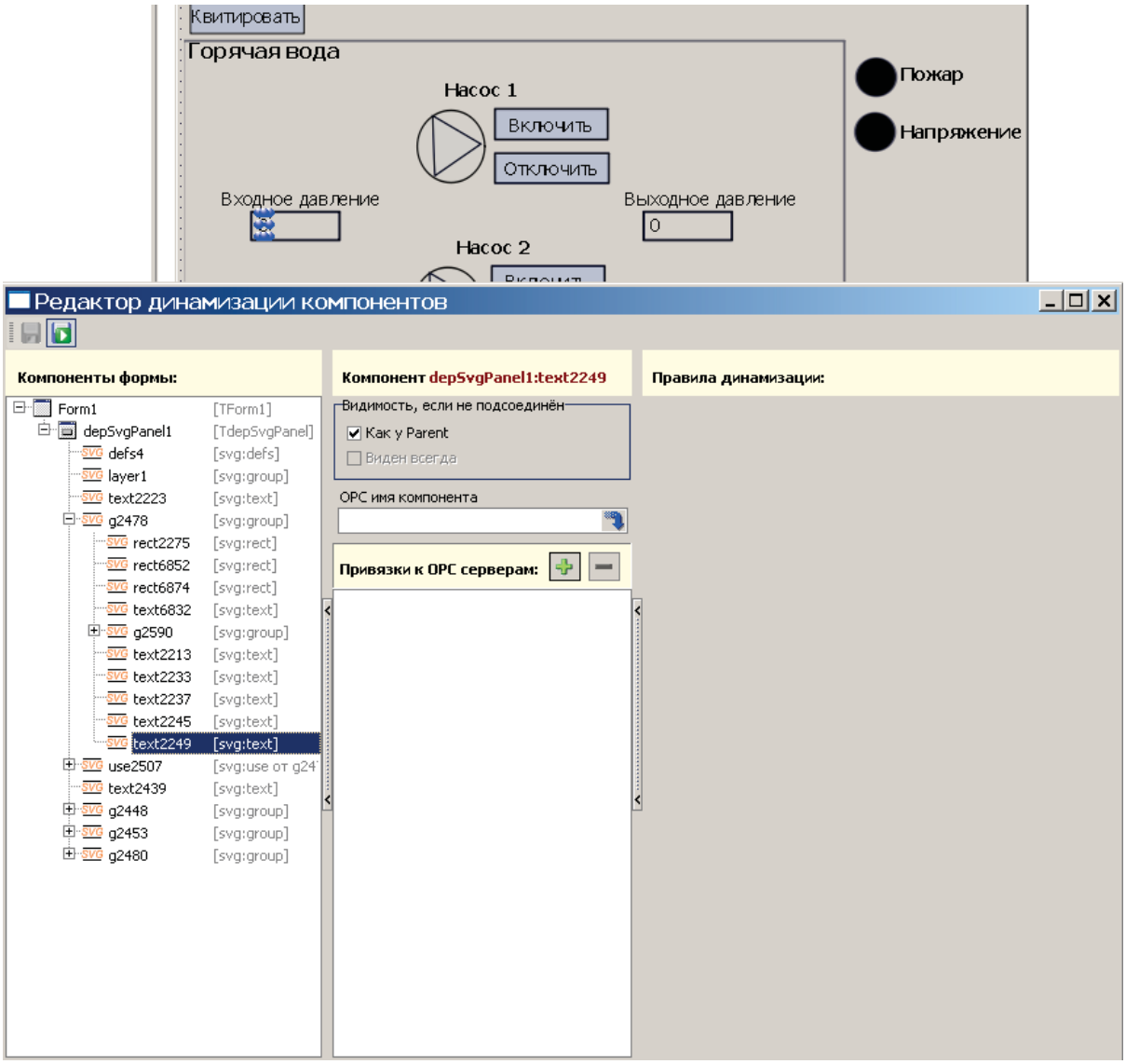


Рис. 84. Редактор динамизации компонентов.

Нажмите "Добавить привязку"  . Выберите свойство метки, к которому хотите сделать привязку. В нашем случае это Text.Caption - текст метки. Нажмите "Выбрать".

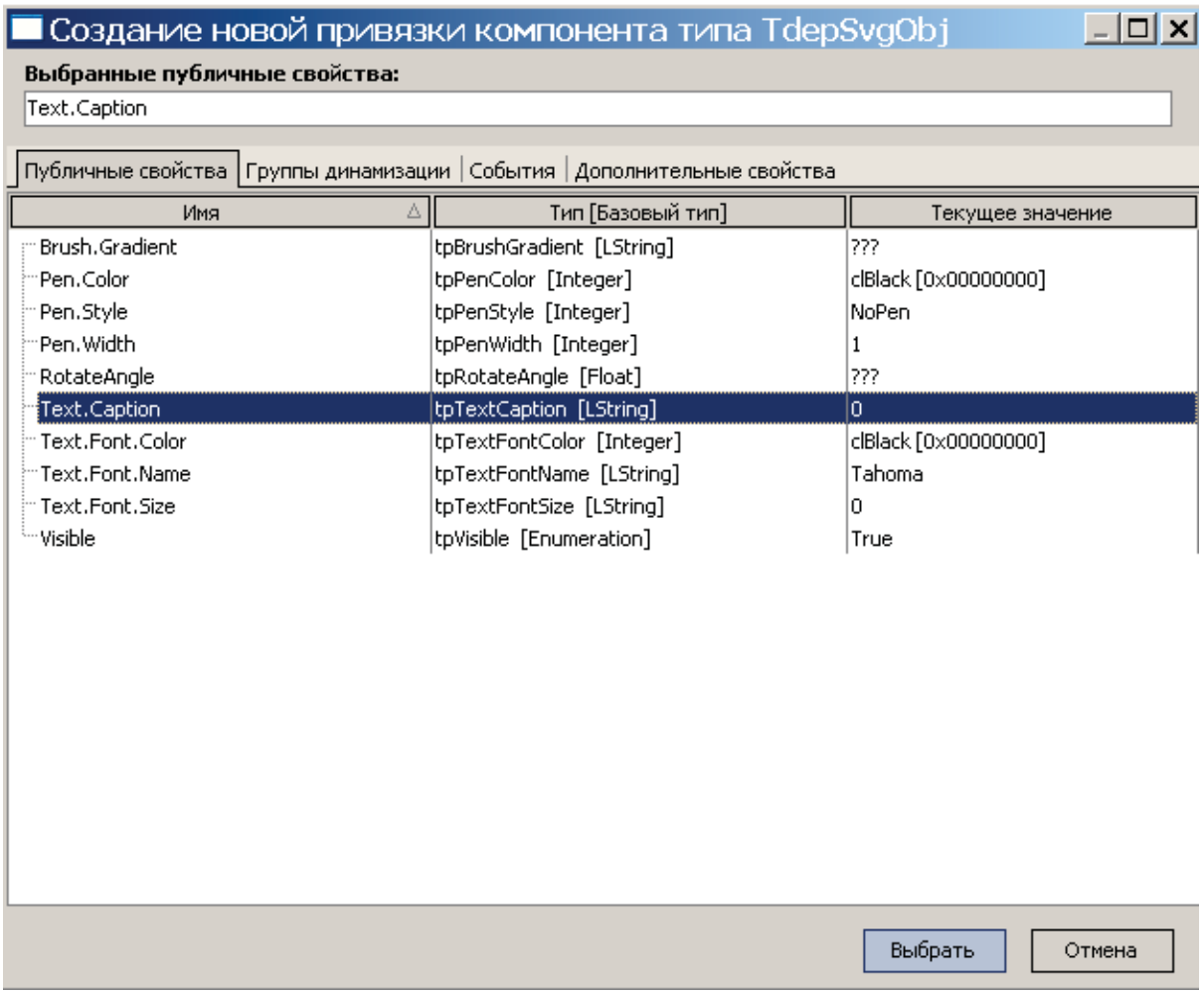


Рис. 85. Создание новой привязки.

В полученном окне выберите элемент модели для привязки с помощью  . В нашем случае мы хотим привязать текст метки к входному давлению для группы насосов горячей воды, поэтому мы должны привязаться к элементу "InPressure". Выберите элемент и нажмите "Выбрать".

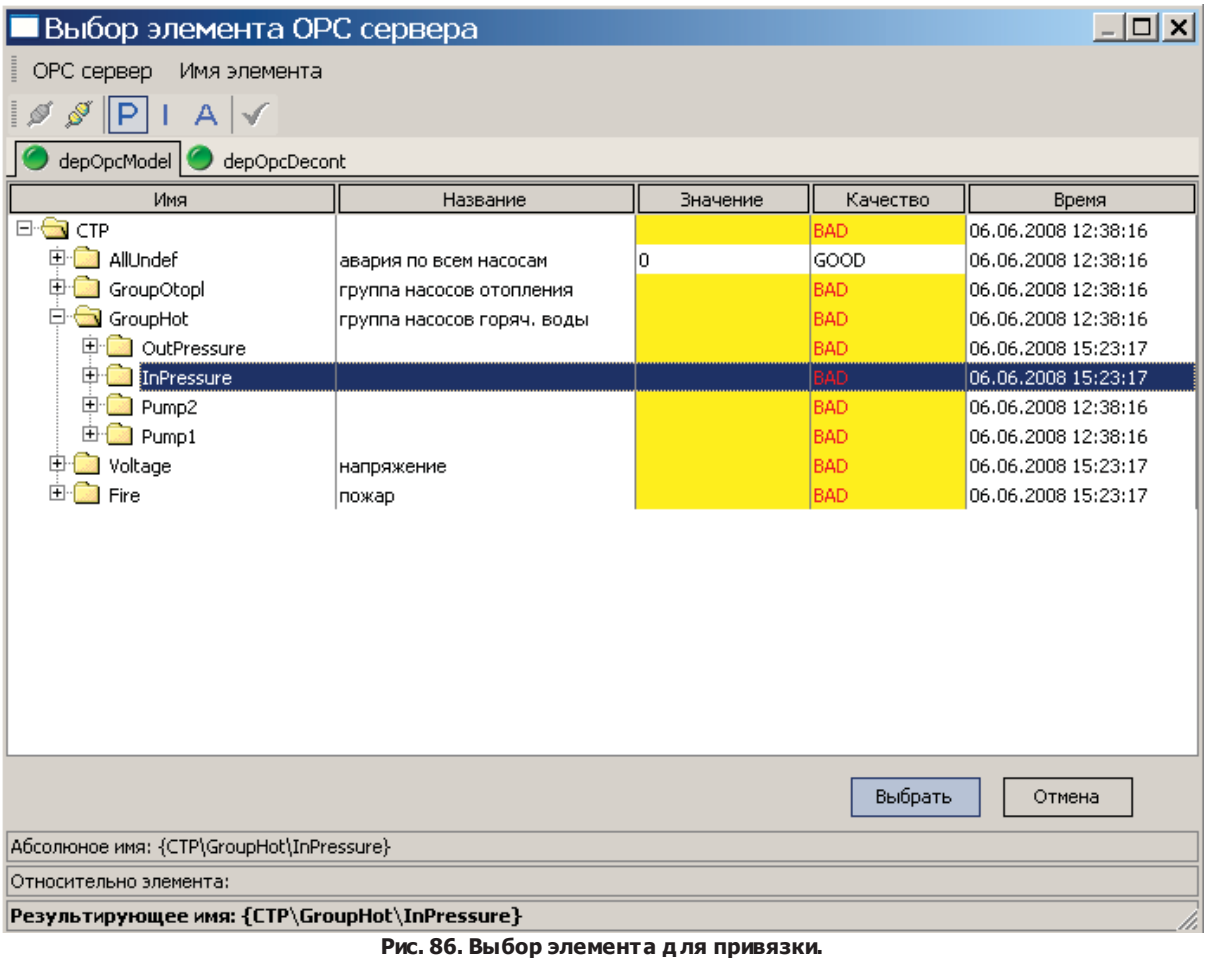


Рис. 86. Выбор элемента для привязки.

Замечание. CTP\GroupHot необходимо удалить! Оставить только InPressure!

В поле "Остальные" в окне редактирования состояний напишите %f.

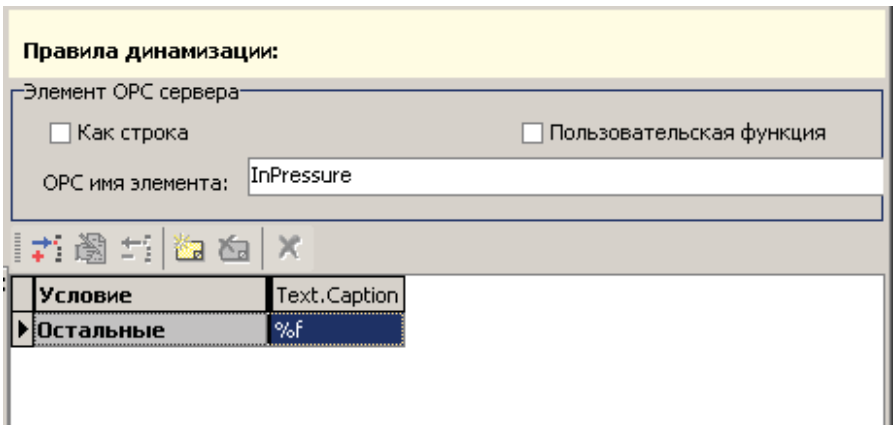


Рис. 87. Редактирование поля Caption для текстового поля входного давления насосов горячей воды.

Также сделаем, чтобы при неопределенном значении показывался 0. Для этого нажимаем  и выбираем "Неопределенность"

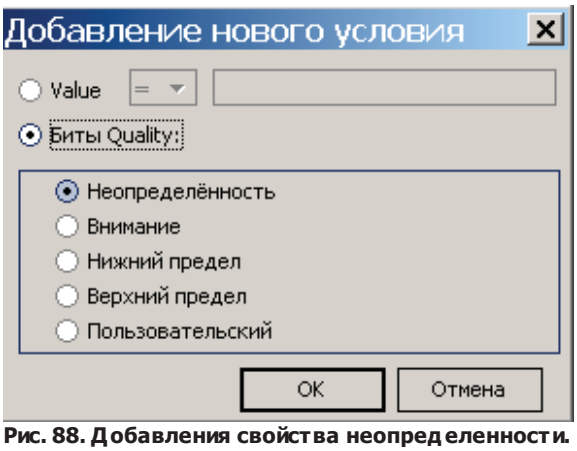


Рис. 88. Добавления свойства неопределенности.

В результате получаем:

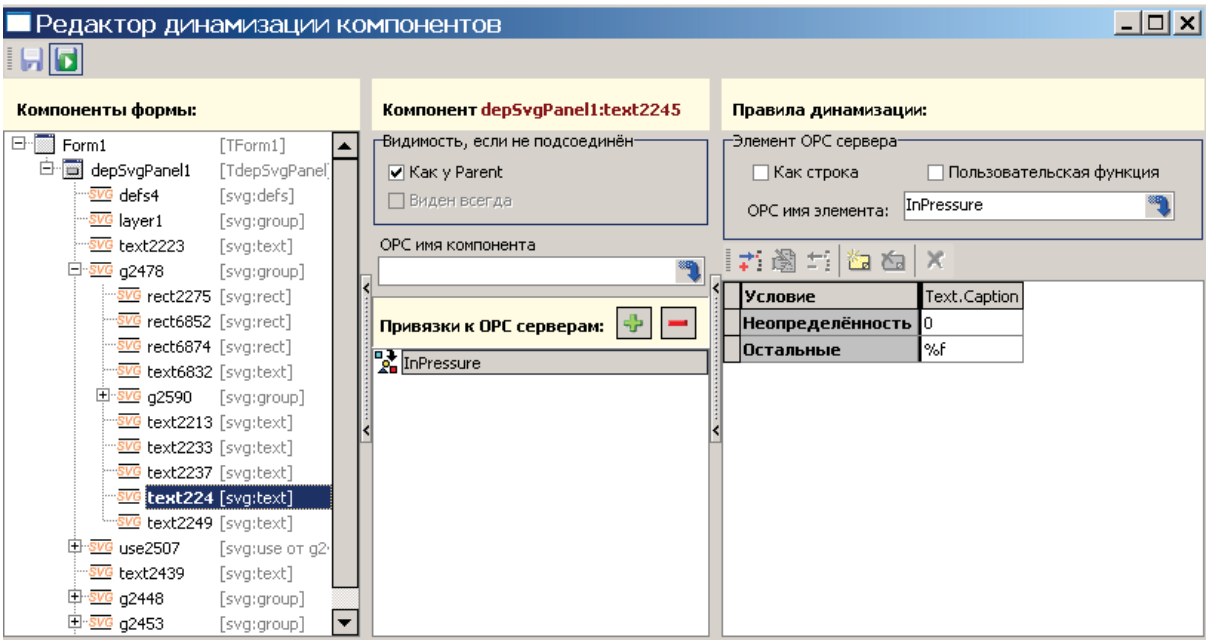


Рис. 89. Редактор динамизации компонентов.

Здесь следует дать пояснение. При выводе численной (строковой) информации мы должны взять значение OPC-элемента и записать его в текстовое поле (свойство Caption). При этом используются, так называемые, маски. С помощью маски задается тип значения, который мы хотим выводить в текстовом виде.

Маски

- **%f** - числа с плавающей запятой.
- **%d** - целые числа.
- **%s** - строки.

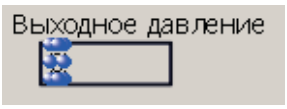
Так как входное давление - это аналог, то тип данных у нас - float (числа с плавающей запятой), поэтому мы используем маску %f.

Условия для состояния OPC-элемента просматриваются сверху вниз, и последним всегда идет "Остальные", которое используется, если не подошло ни одно из предыдущих условий.

Исходя из выше сказанного, привязка работает следующим образом: берется значение элемента модели, оно рассматривается как число с плавающей запятой, и записывается в текстовое поле.

На этом привязка метки закончилась. Теперь наша метка будет отображать значение входного давления для группы насосов горячей воды. Закройте редактор OPC состояний.

Привяжем метку выходного давления для группы насосов горячей воды.

Выделите 

и нажмите Shift+F3. Нажмите добавить привязку  и выберите свойство Caption. Выберите элемент модели для привязки  . В нашем случае мы хотим привязать текст метки к выходному давлению для группы насосов горячей воды, поэтому мы должны привязаться к элементу "OutPressure". Выберите элемент и нажмите "Выбрать".

В поле Caption в окне редактирования состояний напишите %f.

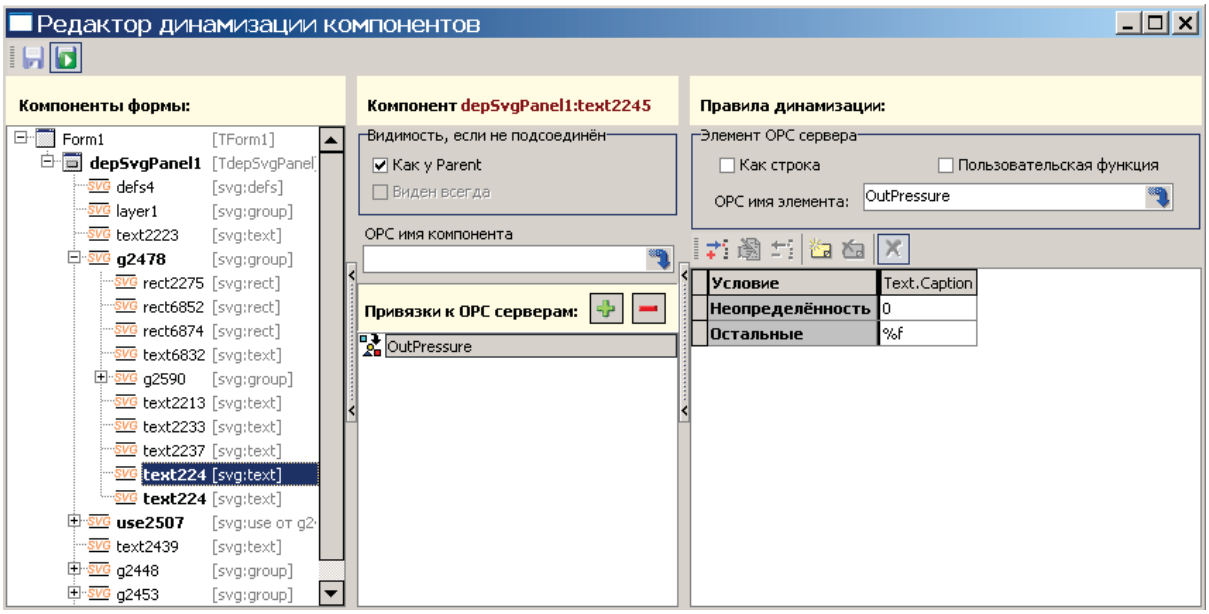


Рис. 90. Задание привязки к выходному давлению.

После проделанных действий заметим, что мы создавали привязки только к конечным элементам динамизации. Полный путь к ним мы удалили. Это сделано для того, чтобы с легкостью можно было раширять наше приложение.

Поэтому далее необходимо сделать так, чтобы полный путь все-таки сформировался.

Каждому компоненту можно задать OPC имя компонента. Путь нижнего элемента формируется из суммы путей всех верхних элементов.

Поэтому зададим сейчас верхние пути.

Для начала выделим группу, в которую входят входное и выходное давление:

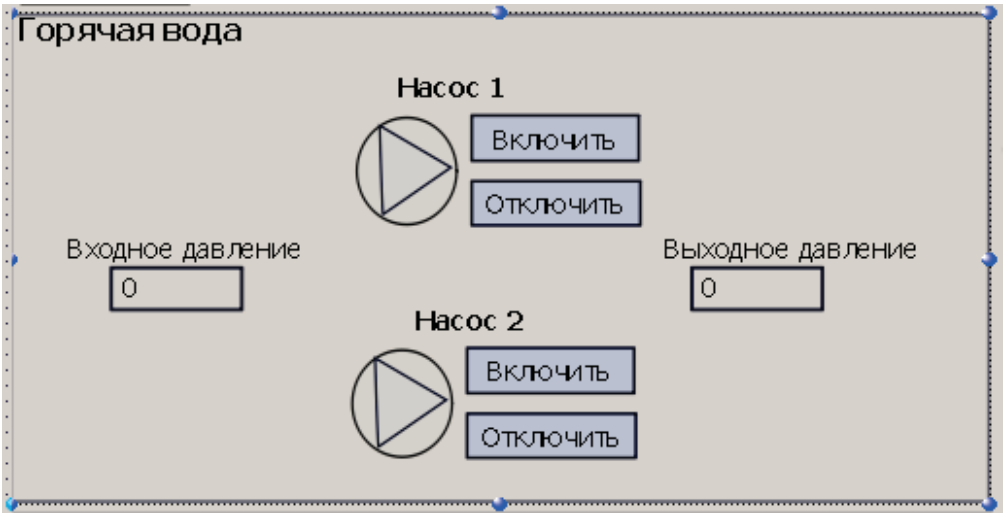


Рис. 91. Выделение группы.

В поле "OPC имя компонента" укажем OPC имя компонента:

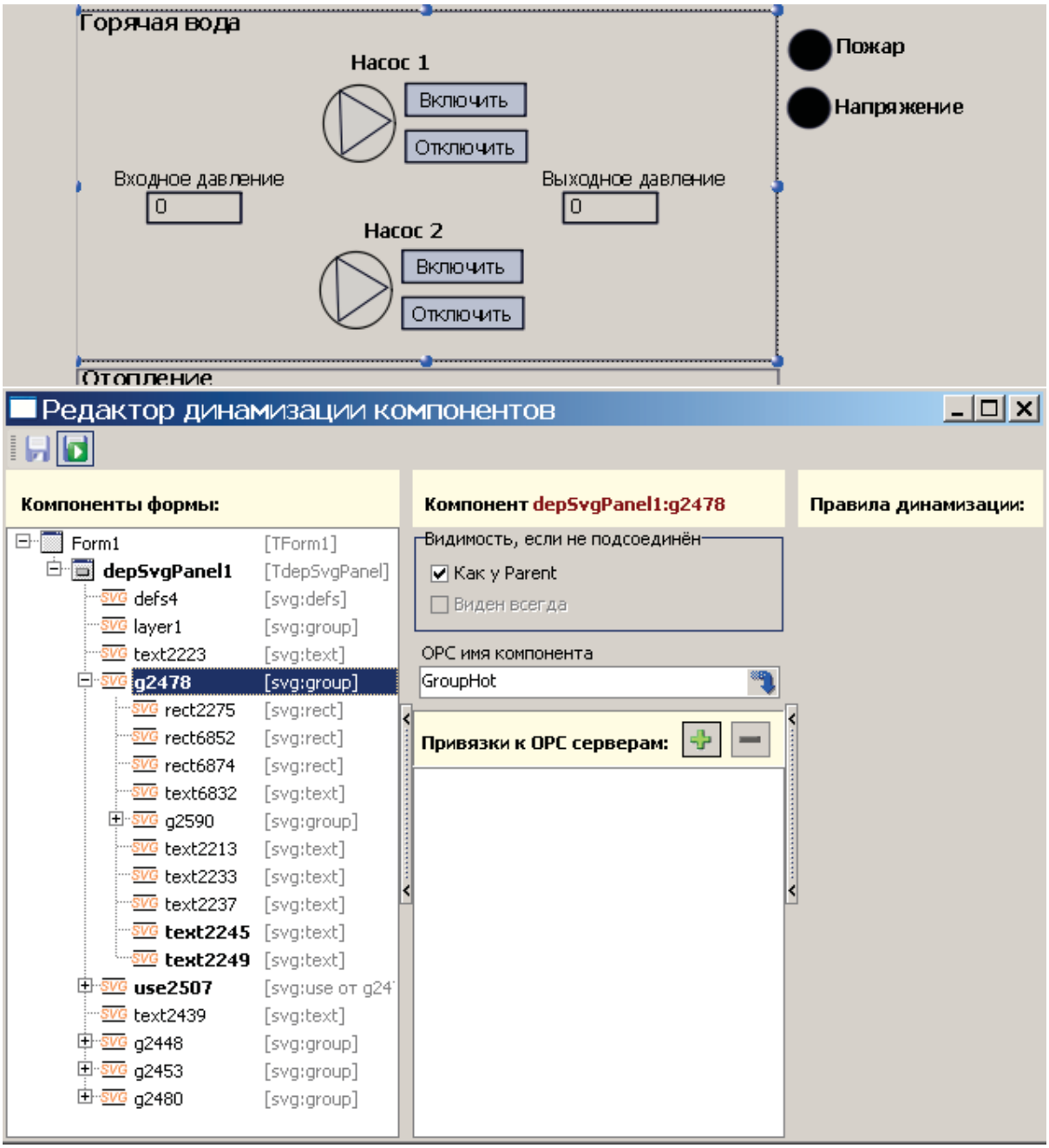


Рис. 92. Задание OPC имени для оригинала.

В результате имя к входному давлению будет: GroupHot\InPressure. Аналогично и к выходному.

До сих пор мы делали привязки к группе насосов горячей воды, для его клона-группы насосов отопления привязки наследуются. Они будут такими же, как и в оригинале.

Единственное, что можно поменять у клона - это "OPC имя компонента".

Поэтому в разделе группы насосов отопления нужно задать этот параметр:

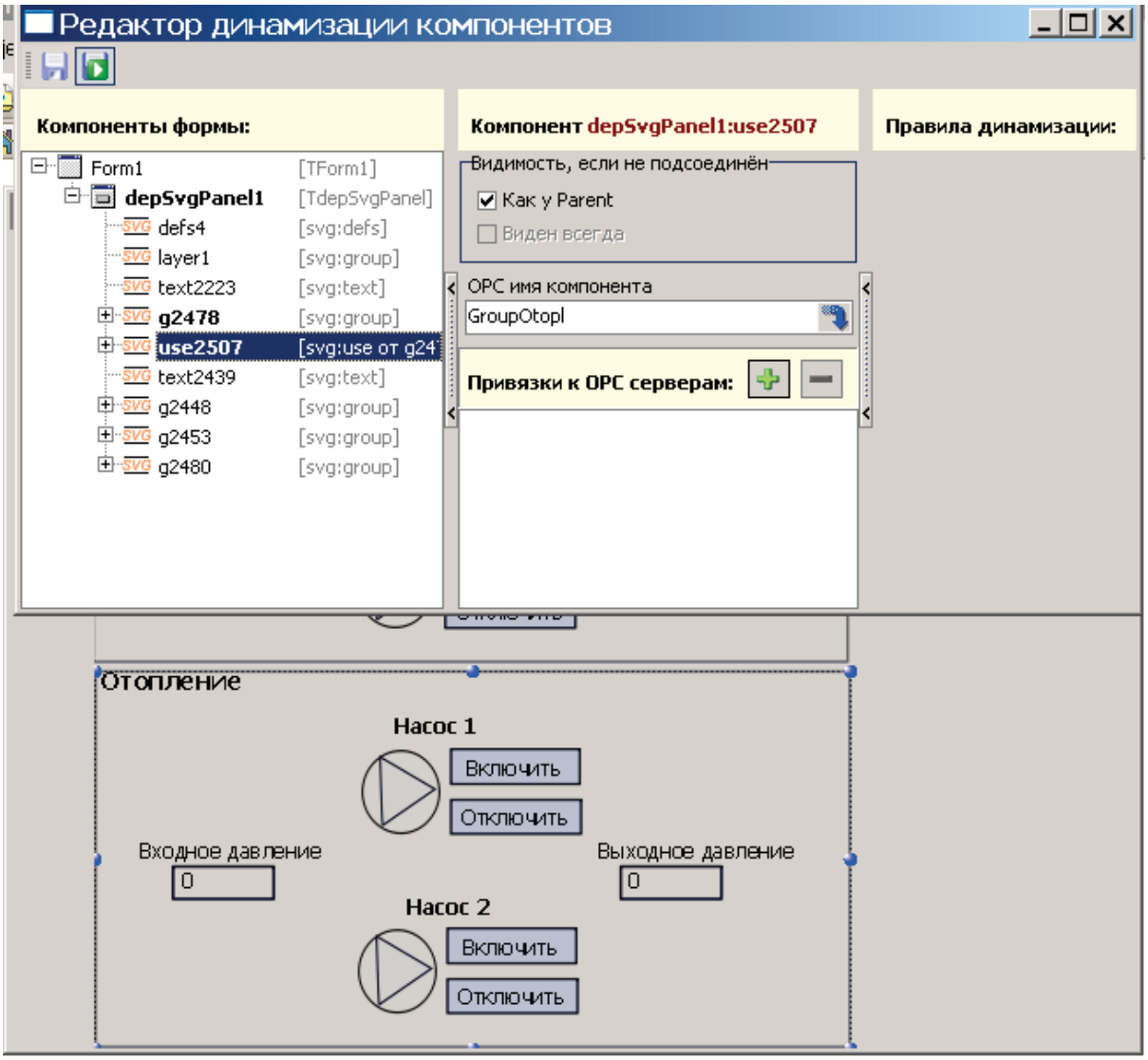


Рис. 93. Задание OPC имени для клона.

В результате имя к входному давлению будет: GroupOtopl\InPressur. Аналогично и к выходному.

Создав привязки, проект можно сохранить с помощью кнопки  , или нажав комбинацию клавиш Ctrl+S.

Следующий шаг: [Привязка изображений насосов](#).

10.4.5.5 Привязка изображений насосов

Как вы помните, в соответствии с заданием мы должны менять цвет фона изображений насосов в зависимости от их состояния. Значит, мы должны привязать свойство, отвечающее за цвет фона к элементу модели, отвечающему за состояние

насоса.

Щелкните левой кнопкой мышки по окружности насоса 1 из группы насосов горячей воды.

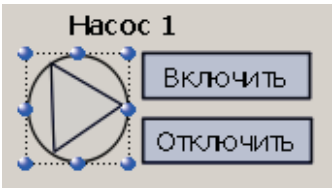


Рис. 94. Выделение области.

В редакторе динамизации добавьте привязку  . Выберите свойство Brush.Color (цвет фона). Нажмите "Выбрать".

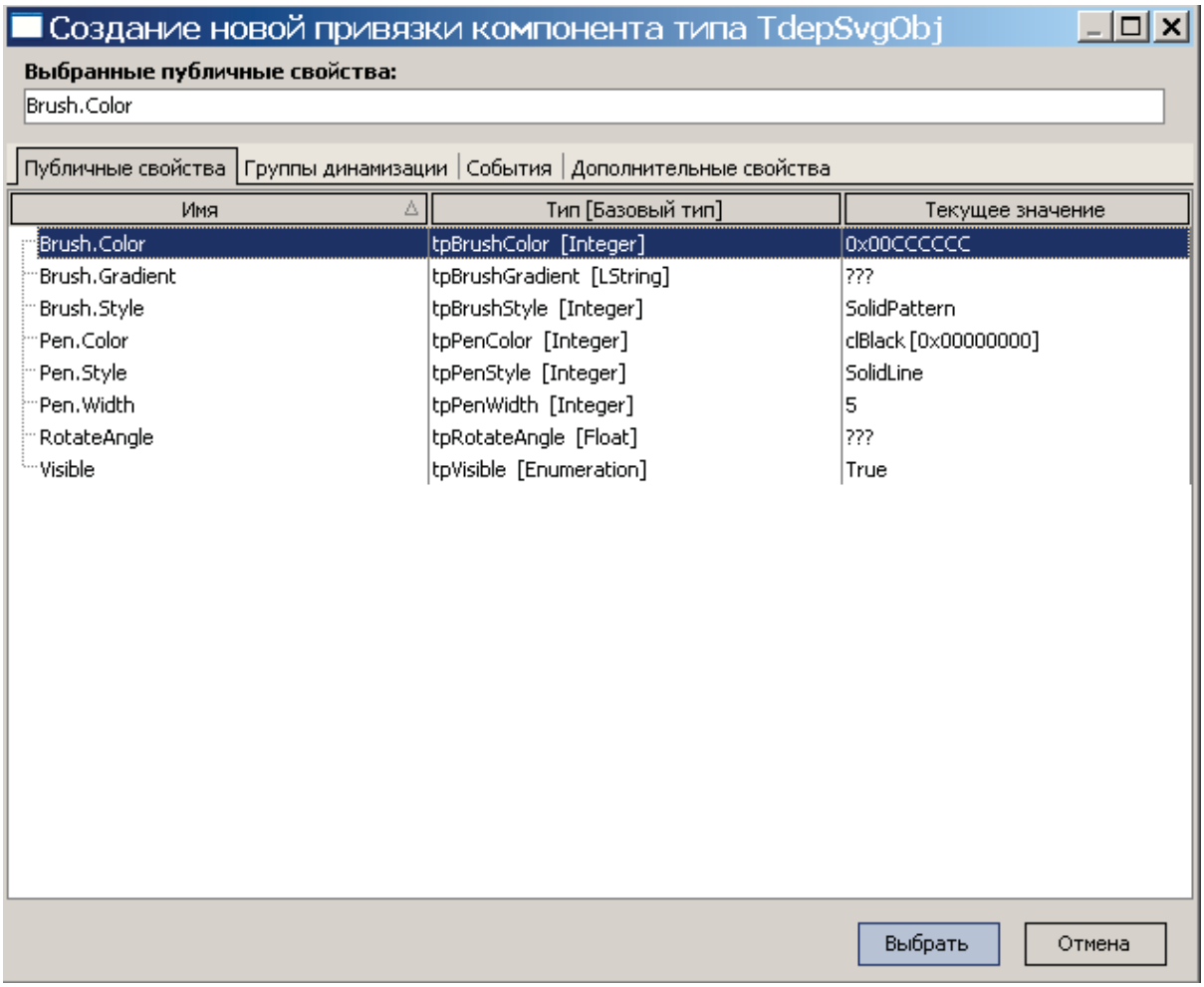


Рис. 95. Выбор свойства Brush.Color.

Нажмите "Выбрать элемент модели"  . Выберите элемент State (состояние первого насоса из группы насосов горячей воды) . Нажмите "Выбрать".

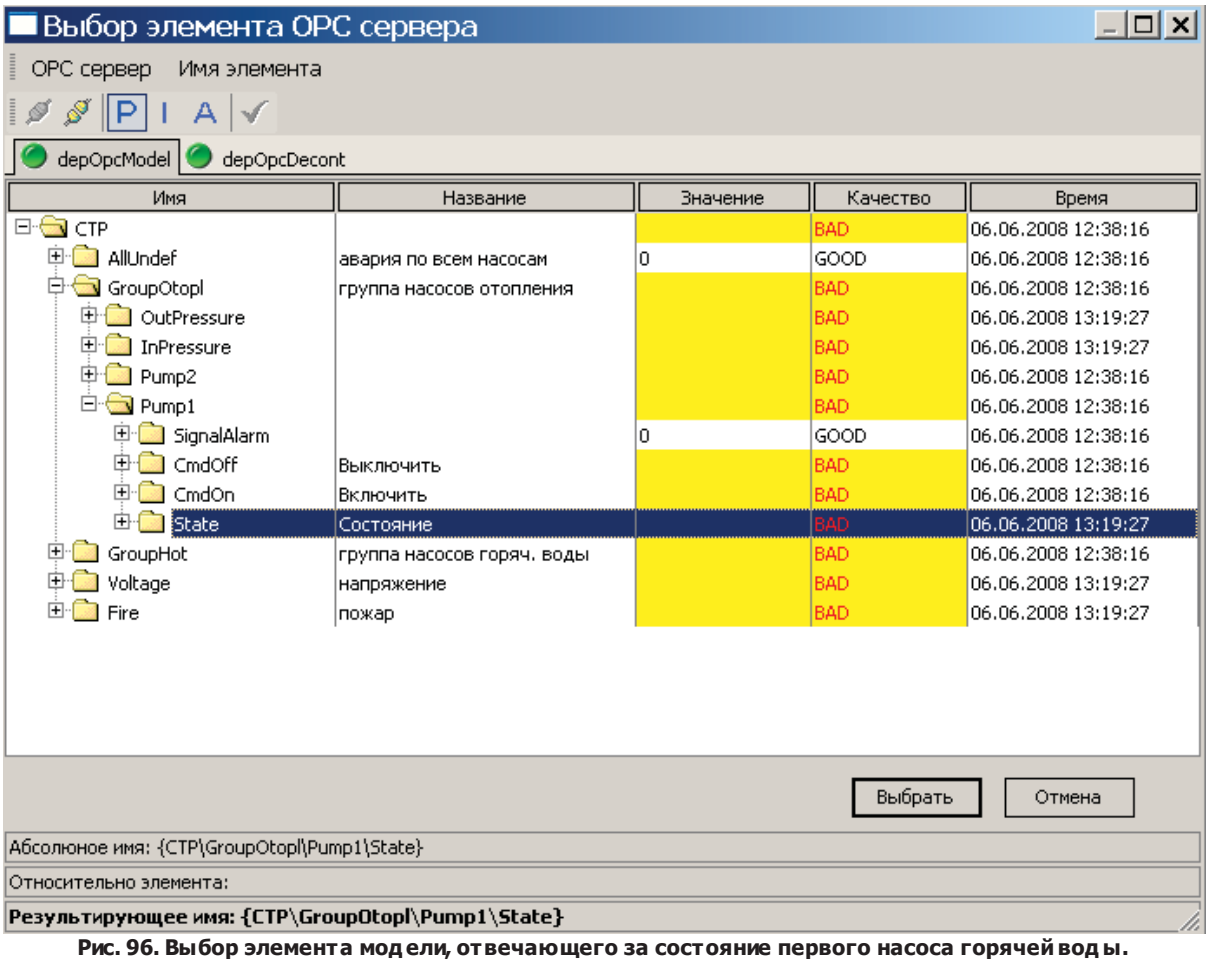
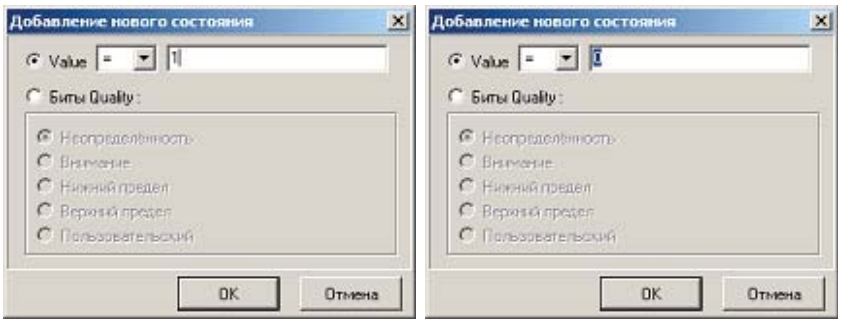


Рис. 96. Выбор элемента модели, отвечающего за состояние первого насоса горячей воды.

Добавьте включенное, выключенное и неопределенное состояние. Для этого щелкните мышкой по значку  (добавить состояние). Определимся, чем характеризуется каждое состояние. Насос включен - value (значение дискрета) = 1, насос выключен - value (значение дискрета) = 0, состояние неопределенно - Quality. Неопределенность (неопределенное значение дискрета). Добавьте в соответствии с этим состояния для насоса. Для добавления каждого состояния щелкаете мышкой по значку "Добавить состояние", выбираете нужное состояние, после чего нажимаете "OK".

Необходимо помнить, что верхним в списке состояний всегда должно быть: "Неопределенность", это требуется для корректного отображения всех последующих состояний.



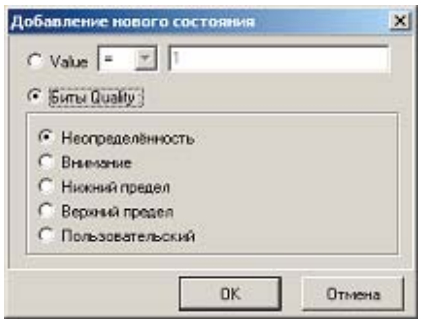


Рис. 97. Добавление включенного, выключенного и неопределенного состояний насоса.

Выберите для каждого состояния цвет в соответствии с заданием на АРМ. Для этого щелкните мышкой по полю Brush.Color напротив каждого состояния и выберите цвет из списка.

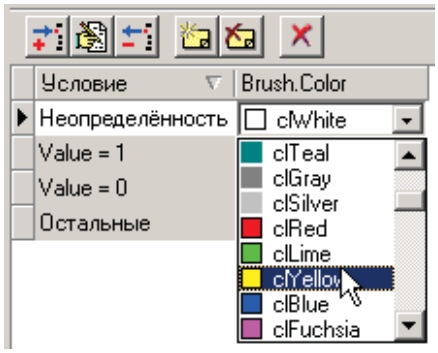


Рис. 98. Выбор цвета для неопределенного состояния.

После выбора цвета для каждого состояния окно редактирования состояний примет вид.

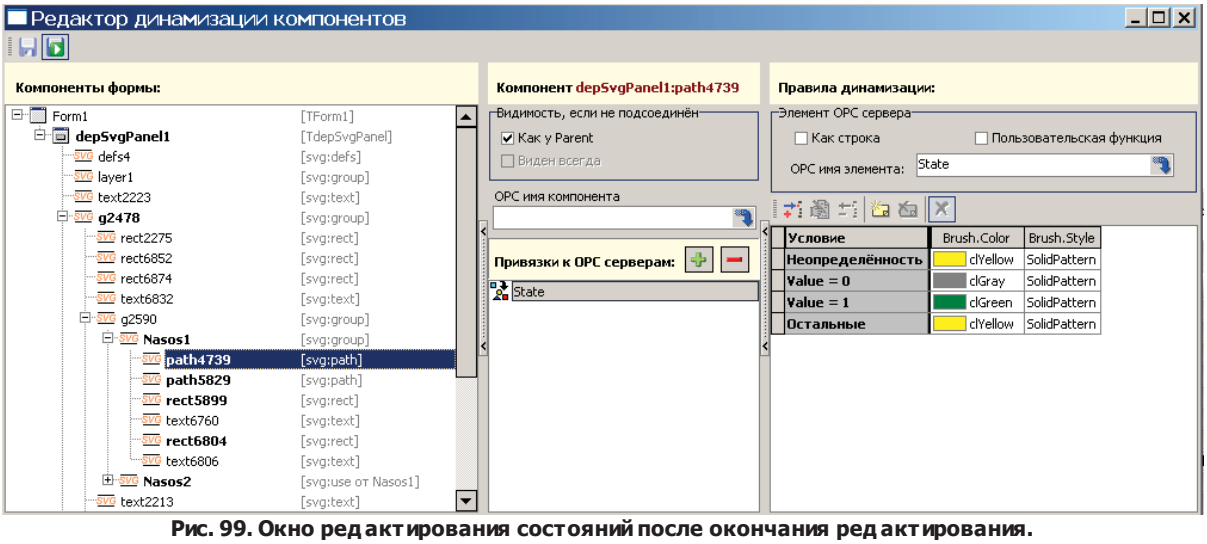
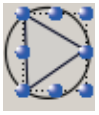


Рис. 99. Окно редактирования состояний после окончания редактирования.

Поскольку Насос 2 в группе насосов горячей воды является клоном Насоса 1, то только что сделанные привязки автоматически добавятся и к элементам Насоса 2.

Теперь приступим к изменению цвета фона треугольника насоса.

Выделим элемент  и добавим привязку к Brush.Color. Зададим следующие параметры:

Условие	Brush.Color
Внимание	clRed
Остальные	clSilver

Рис. 100. Редактирования свойства Brush.Color.

Добавим с помощью кнопки  новое свойство - мигание, с помощью  зададим OPC - имя компонента (SignalAlarm). В результате получаем:

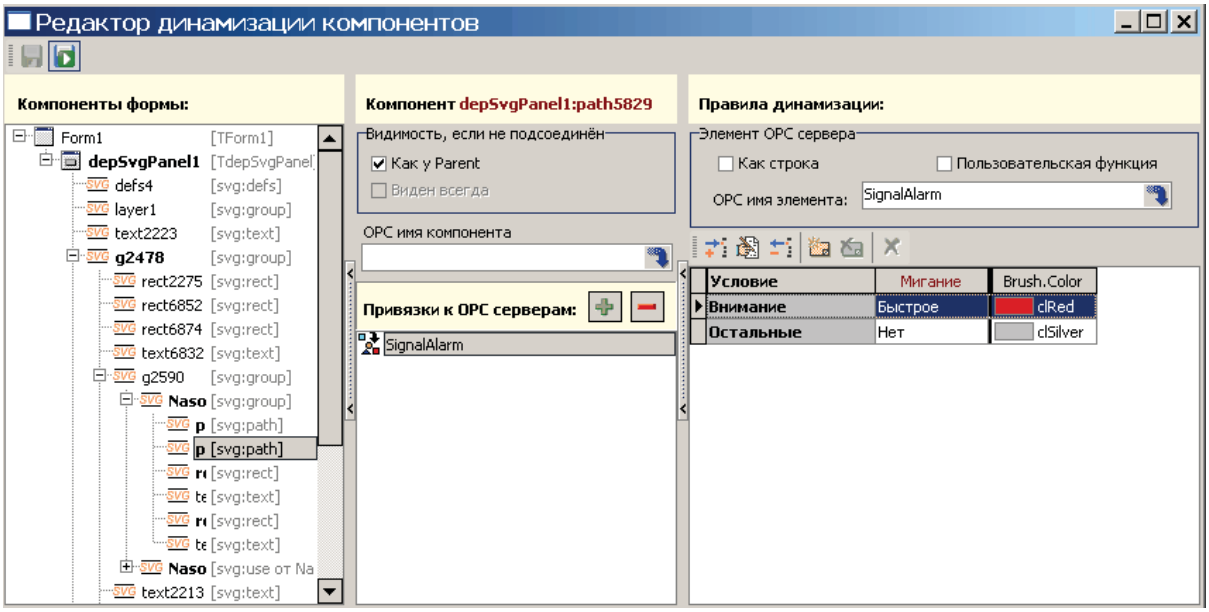


Рис. 101. Редактирование привязки

.

В результате, треугольник насоса будет мигать с красного на серый цвет, если будет тревога у соответствующего насоса, и будет просто серым - если нет тревоги.

Теперь, как описано в Привязка давлений зададим полный путь к элементам динамизации.

Выделяем Nasos1 и задаем "OPC имя компонента"

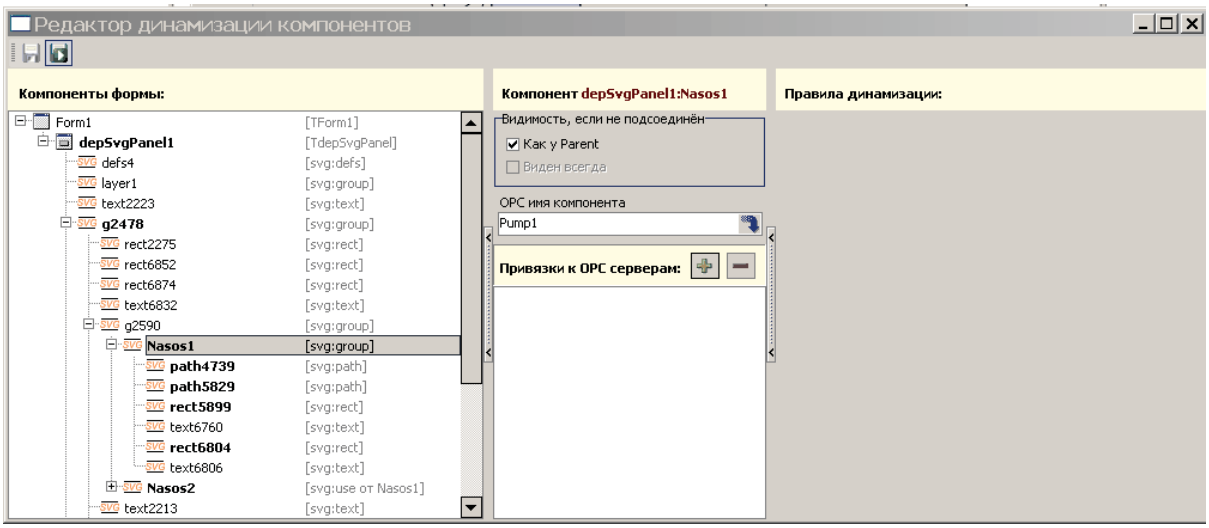


Рис. 102. Редактор динамизации компонентов.

Аналогично, зададим и для Nasos2:

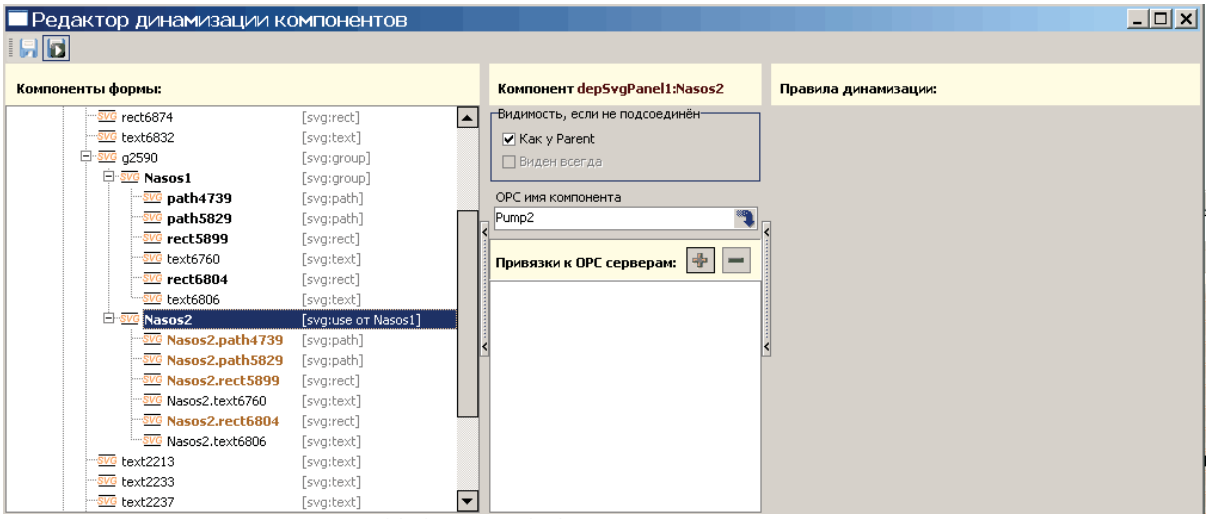


Рис. 103. Задание OPC имя компонентов у насосов

В результате, путь к состоянию насоса 1 горячей воды будет следующим: GroupHot\Pump1\State.

Теперь зададим OPC - имя компонент насосов в клоне - группе насосов отопления.

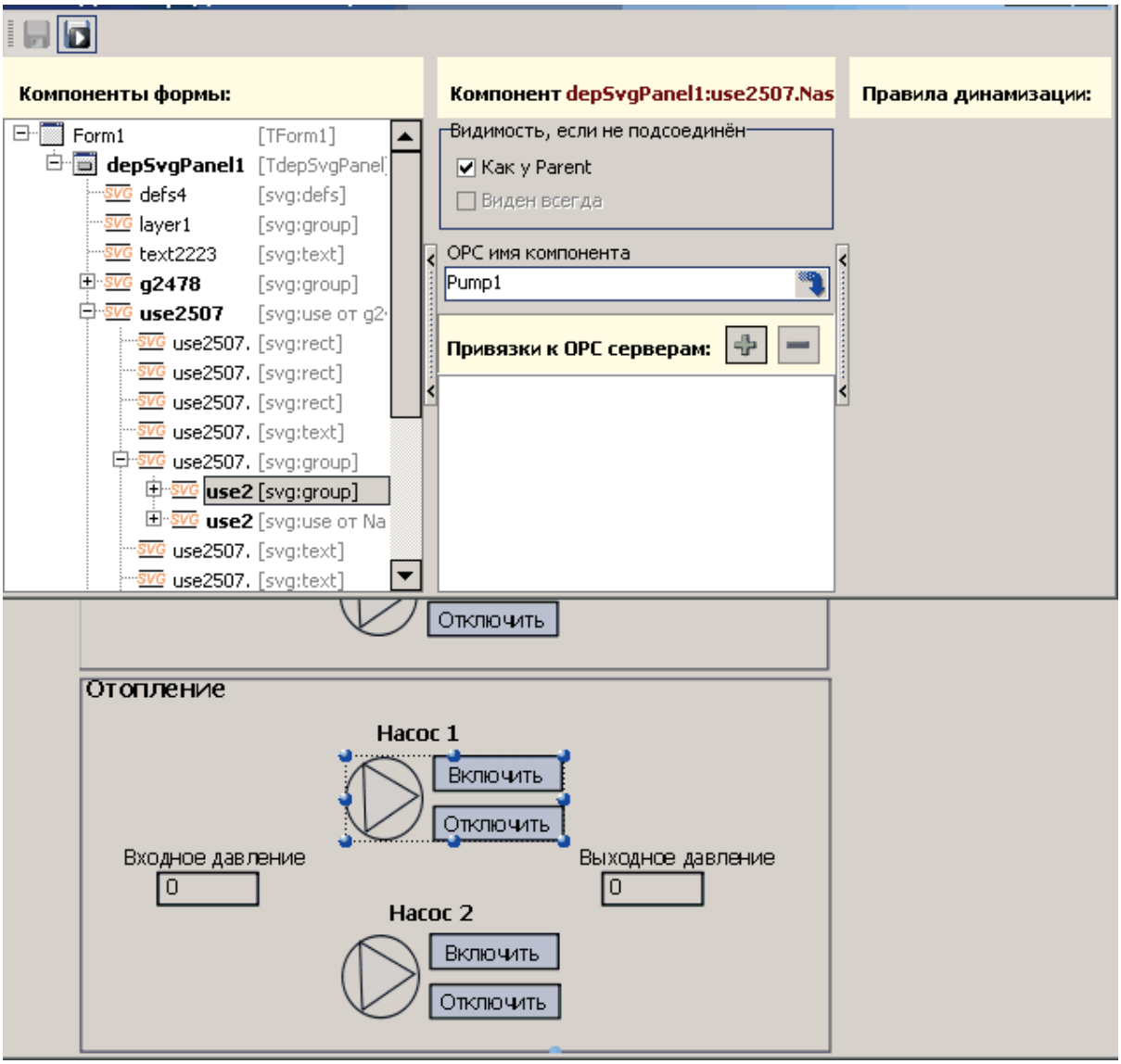


Рис. 104. Задание OPC имя компонентов у насосов.

Не забывайте периодически сохранять проект с помощью кнопки  или комбинации клавиш Ctrl+S.

Теперь привяжем кнопки, отвечающие за управление насосами.

Следующий шаг: Привязка кнопок.

10.4.5.6 Привязка кнопок

Выделите кнопку 

у первого насоса горячей воды. В окне редактора динамизации нажмите добавить привязку  . Откройте закладку "События".

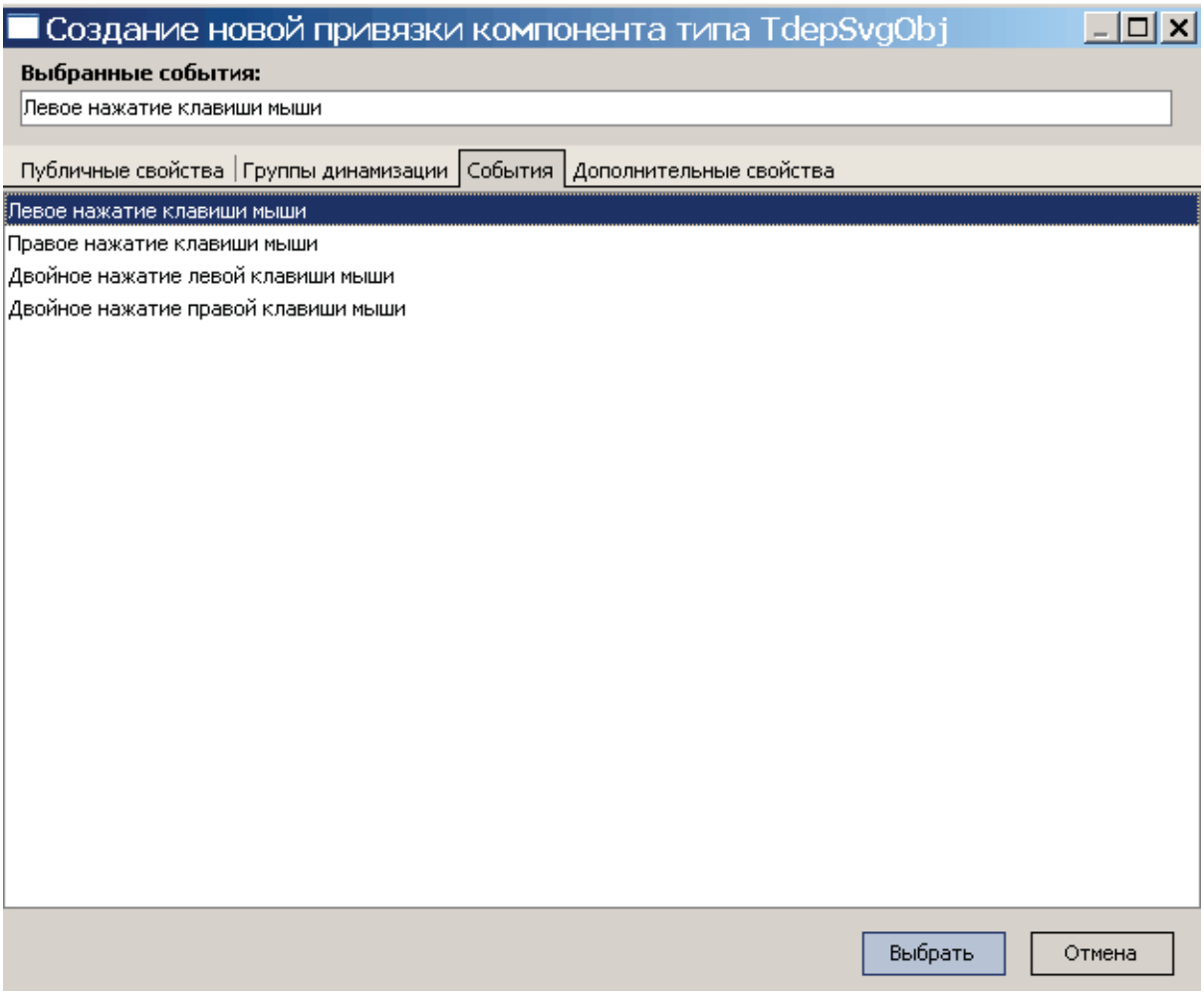


Рис. 105. Закладка "События" окна выбора свойств.

Выберите необходимое событие, в нашем случае это "Левое нажатие клавиши мыши", и нажмите "ОК". Окно редактирования привязок для кнопки примет вид:

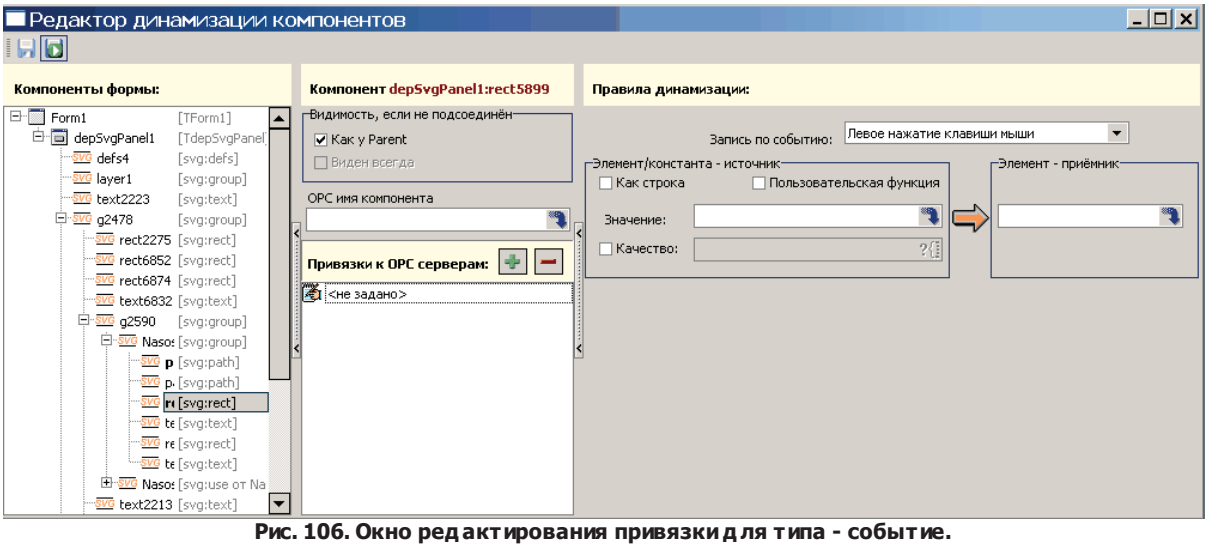


Рис. 106. Окно редактирования привязки для типа - событие.

В левом поле надо написать константу для записи, в нашем случае 1, а в правом поле выбрать элемент модели, куда будет производиться запись, в нашем случае CmdOn.

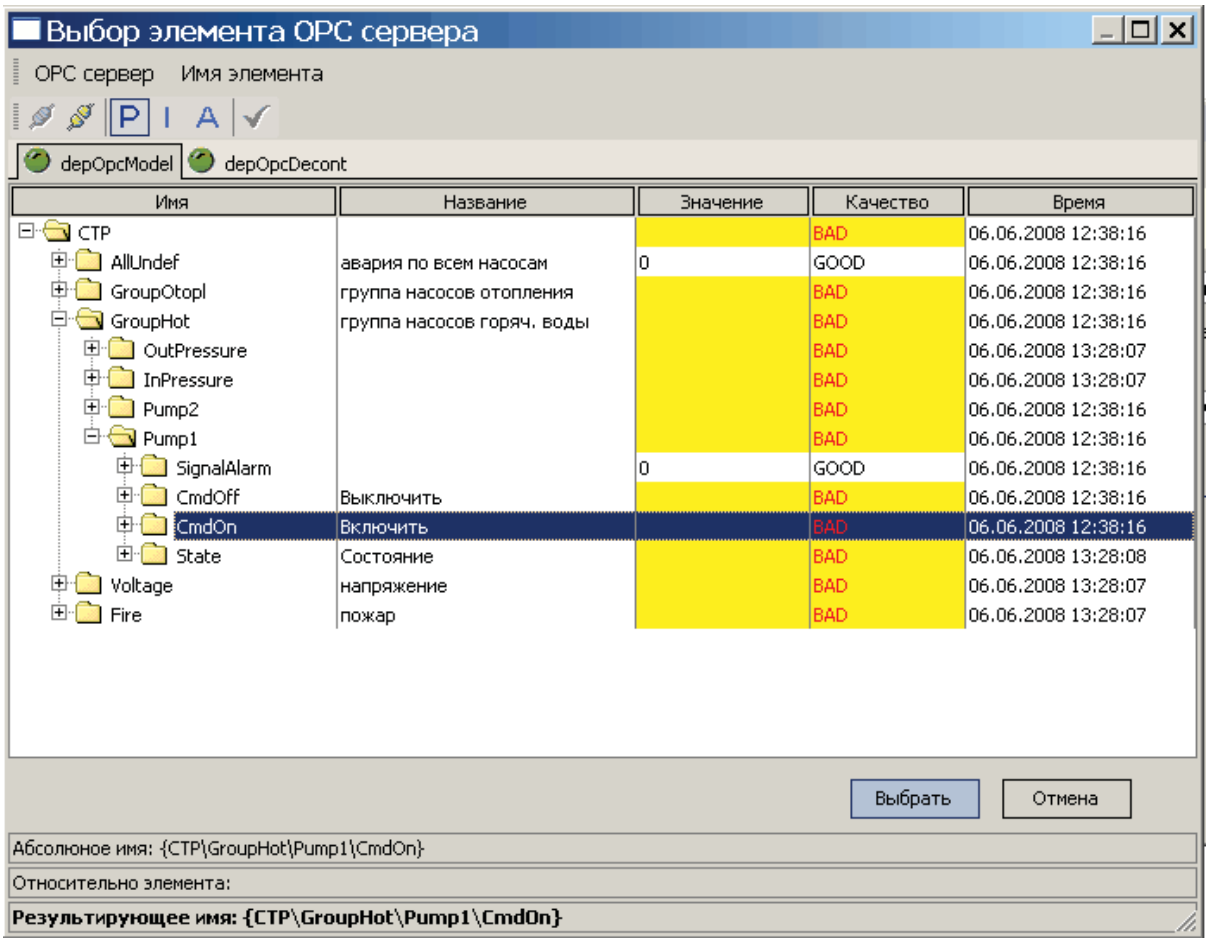


Рис. 107. Выбор элемента модели, отвечающего за включение первого насоса горячей воды.

После этого окно редактирования привязок примет вид.

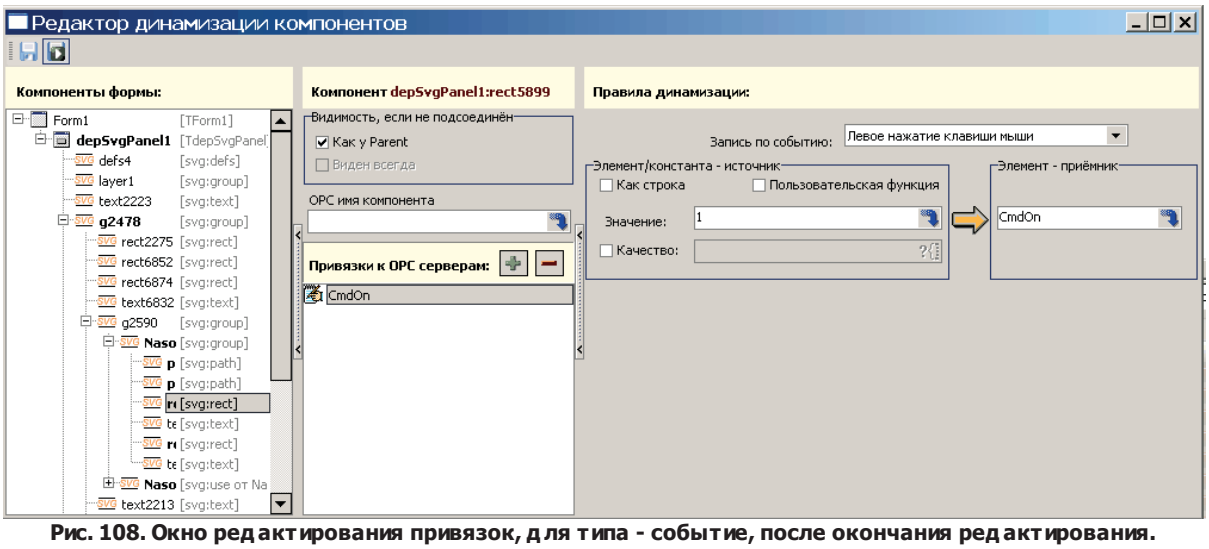


Рис. 108. Окно редактирования привязок, для типа - событие, после окончания редактирования.

Привяжите аналогичным способом остальные кнопки.

Сохраните проект.

И последнее, привяжем индикаторы, отвечающие за пожарную сигнализацию и напряжение.

Следующий шаг: **Привязка индикаторов пожара и напряжения**.

10.4.5.7 Привязка индикаторов пожара и напряжения

Привязка индикаторов напряжения и пожара производится аналогично привязки изображений насосов.

Щелкните левой кнопкой мышки по индикатору пожара. В окне редактора динамизации нажмите добавить привязку  .

В окне свойств выберите свойство Brush.Color, нажмите "ОК". Нажмите "Выбрать элемент модели"  . Выберите элемент Fire, нажмите "Выбрать".

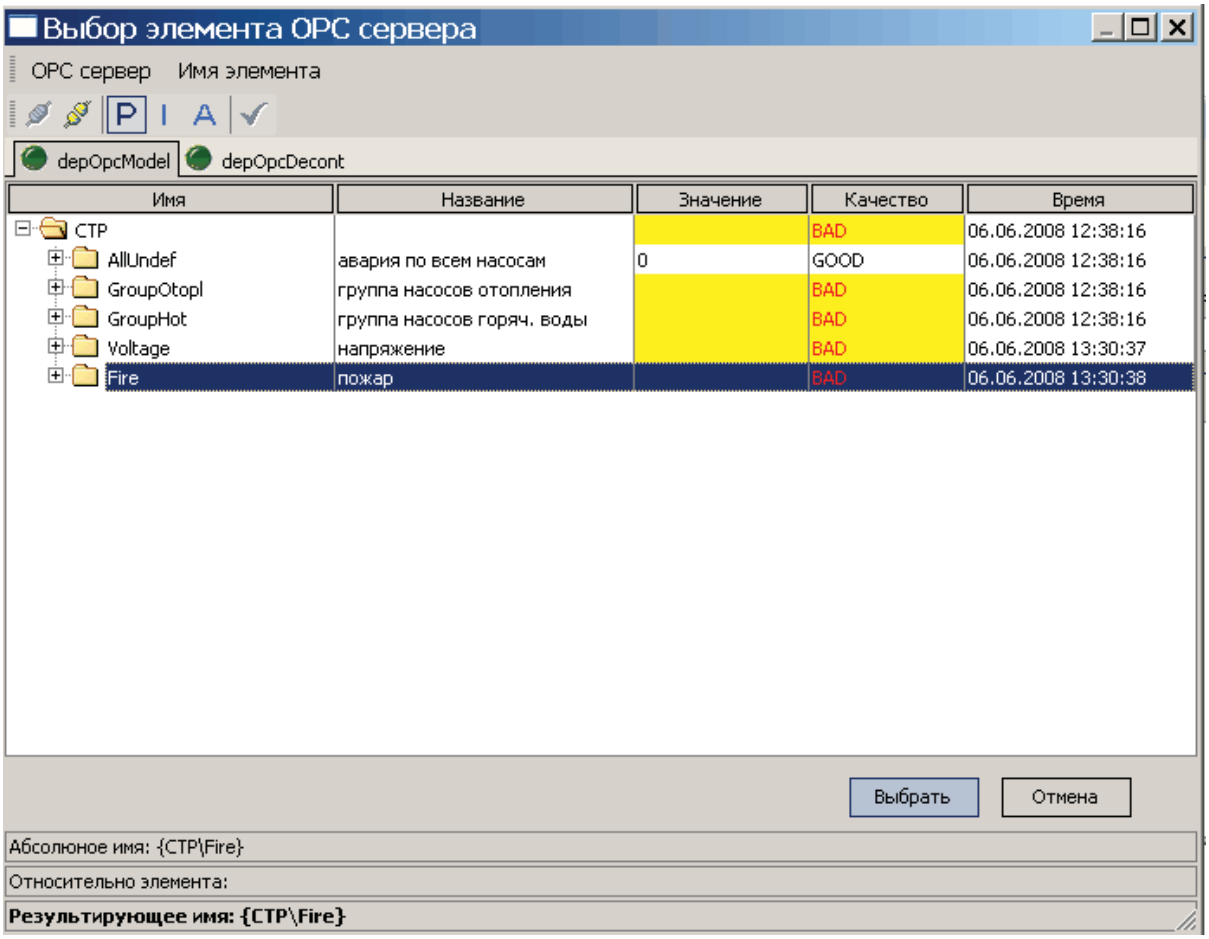
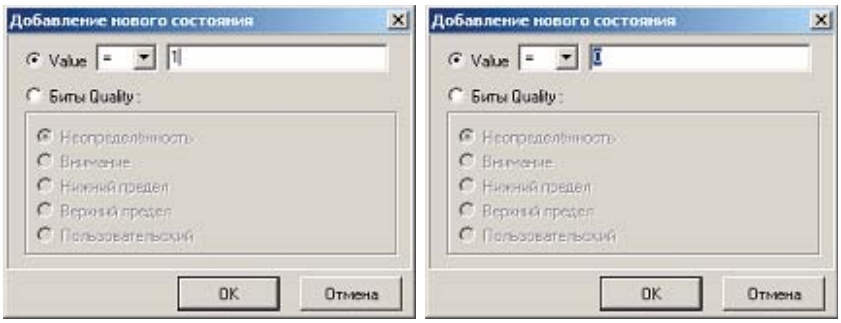


Рис. 109. Выбор элемента модели, отвечающего за пожарную сигнализацию.

Добавьте состояния 1 - сработала пожарная сигнализация, 0 - пожара нет, неопределенно - значение датчика неопределенно.



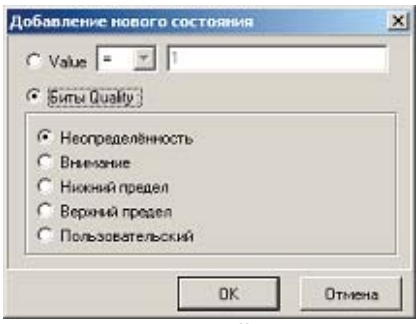


Рис. 110. Добавление состояний для индикатора пожара.

Выберите цвет для каждого состояния в соответствии с заданием на АРМ.

Условие	Brush.Color
Неопределённость	clYellow
Value = 1	clRed
Value = 0	clGreen
Остальные	clWhite

Рис. 111. Выбор цветов для состояний индикатора.

Аналогичным образом необходимо привязать индикатор напряжения к элементу Voltage.

Замечание. Не забудьте, что 0 - красный цвет (напряжения нет), 1 - зеленый (напряжение есть), неопределенность - желтый.
Сохраните проект.

Следующий шаг: **Привязка кнопки Квитировать**.

10.4.5.8 Привязка кнопки Квитировать

Привязка кнопки Квитировать аналогична привязки кнопок у насоса.

Щелкните левой кнопкой мыши на кнопку 

.
Вспомним, что кнопка Квитировать мигает, если есть тревога хотя бы у одного насоса и при нажатии необходимо сбрасывать тревоги у всех насосов.

В окне редактора динамизации нажмите добавить привязку  . В окне свойств выберите свойство Brush.Color, нажмите "ОК". Нажмите "Выбрать элемент модели"  . Выберите элемент AllUndef, нажмите "Выбрать".

Добавьте состояния мигание:

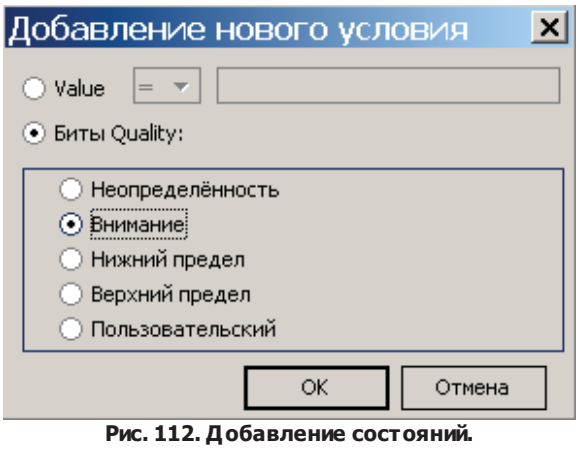


Рис. 112. Добавление состояний.

С помощью  добавьте свойство мигания на вкладке "Дополнительные свойства":

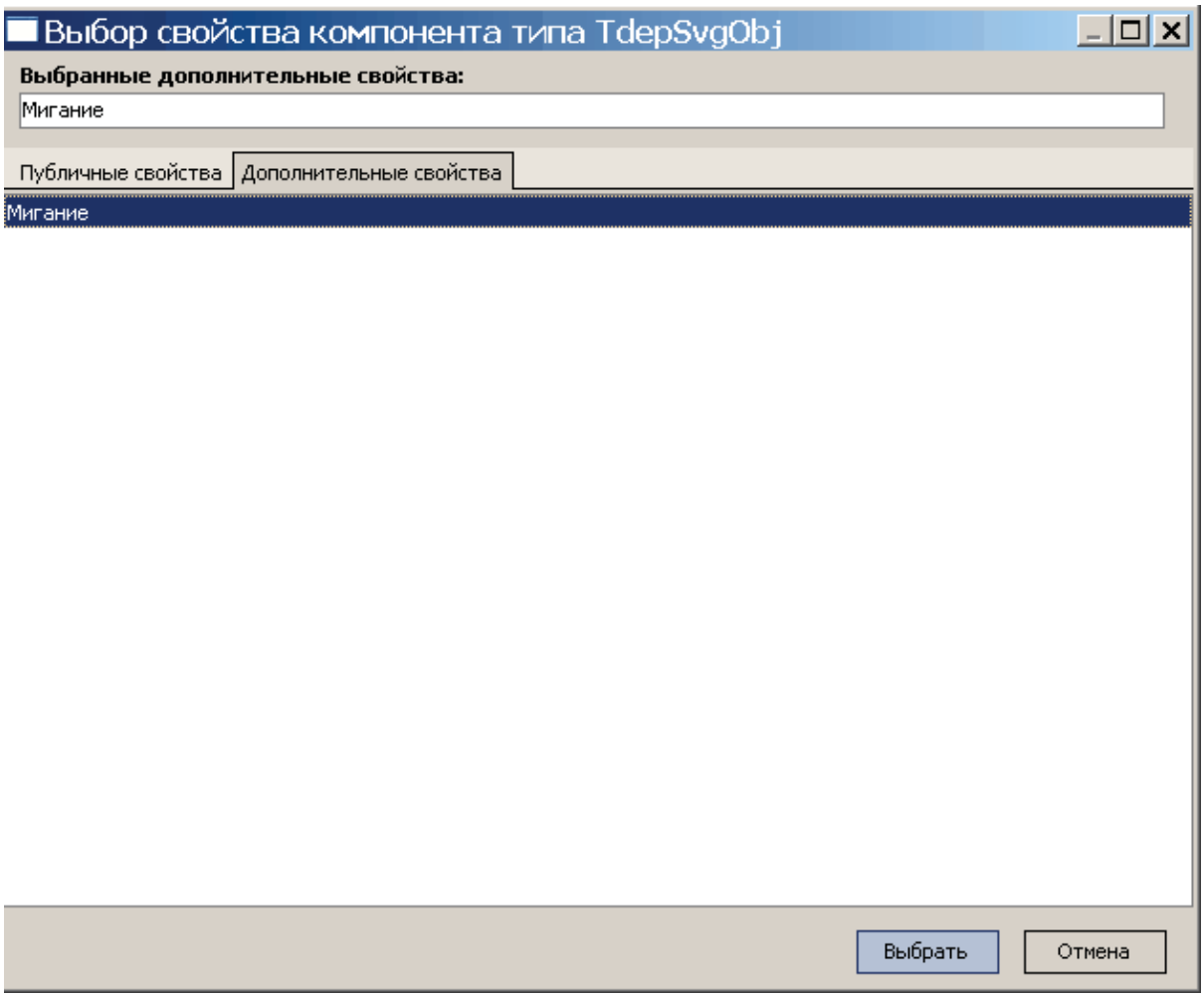


Рис. 113. Выбор свойства компонента.

Теперь зададим цвета состояний в соответствии с заданием на АРМ.

Условие	Мигание	Brush.Color
Внимание	Быстрое	clRed
Остальные	Нет	clSilver

Рис. 114. Выбор цветов для состояний индикатора.

Теперь сделаем, чтобы при нажатии кнопки тревоги сбрасывались. Нажмите снова добавить привязку  . И выберем привязку на событие "Левое нажатие клавиши мыши":

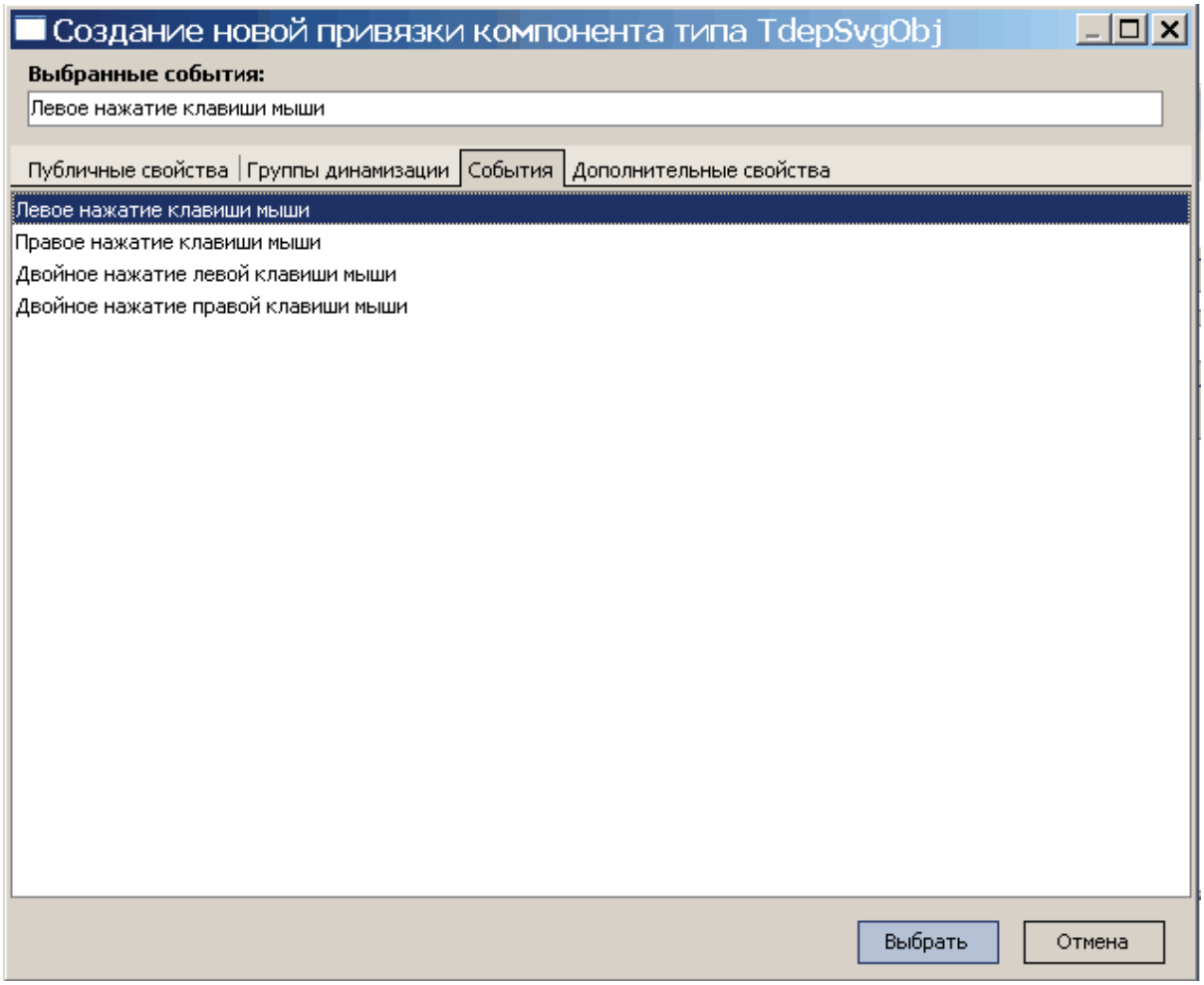


Рис. 115. Создание новой привязки.

И будем писать 6 в AllUndef\Control

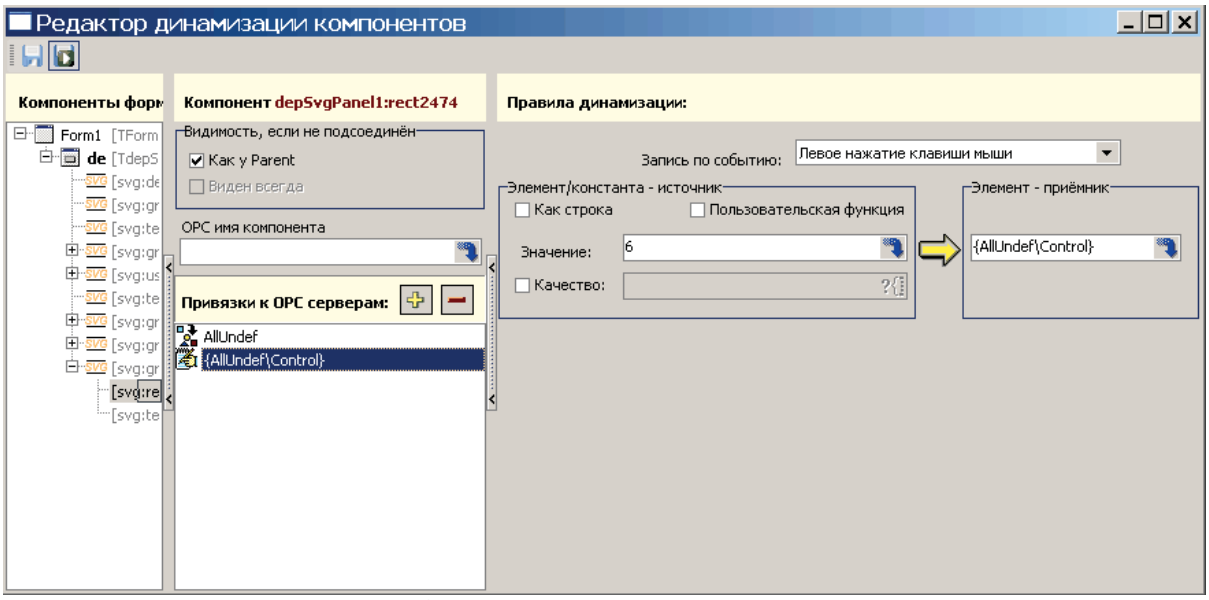


Рис. 116. Редактор динамизации компонентов.

Пишем 6, так как 6 = 2 + 4, где

* 2 - сбросить признак "Внимание" элемента тревоги;

* 4 - сбросить признак "Внимание" у всех контролируемых элементов.

Более подробно описание тревог можно посмотреть в документации по работе с моделью в разделе «Элементы и функции модели/Тревоги».

На этом привязки окончены. Закройте редактор динамических свойств и сохраните проект.

Следующий шаг: Тестирование приложения.

10.4.5.9 Тестирование приложения

Запустите модель и отображение. При этом на картинке у вас все картинки будут желтые (сигналы не определены).

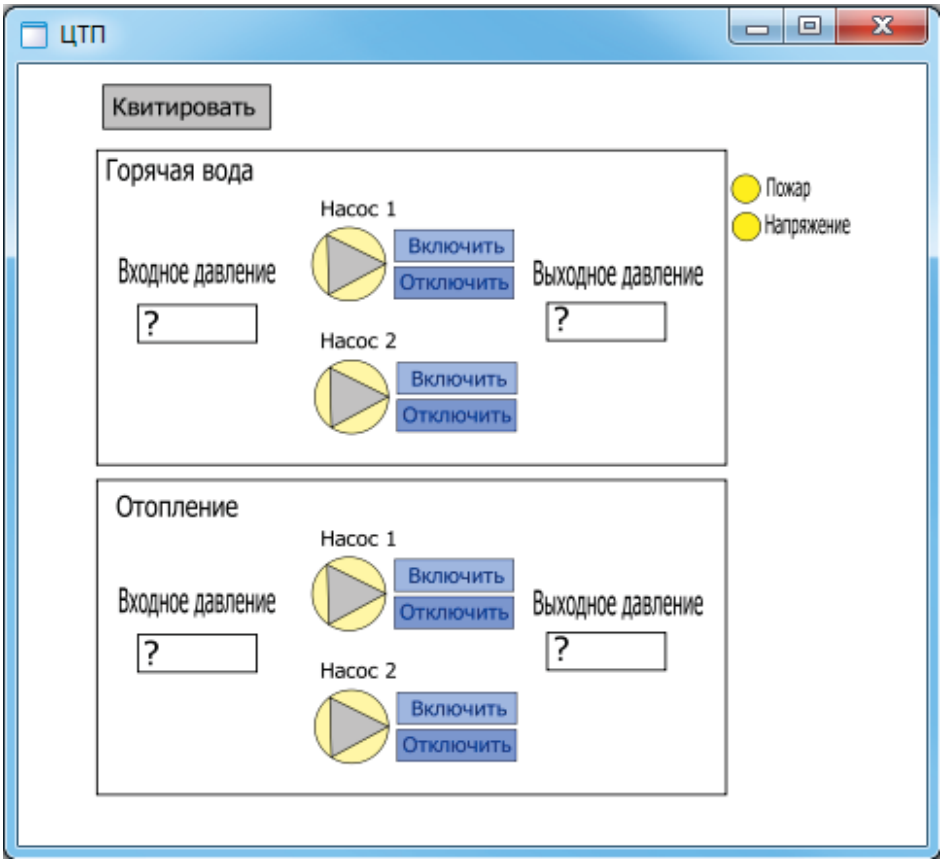


Рис. 117. Неопределенные состояния насосов и индикаторов.

Попробуйте установить в WinDecont, в соответствии с нашей таблицей сигналов (таблица сигналов), единицы в дискреты состояний насосов. При этом отображение изменится.

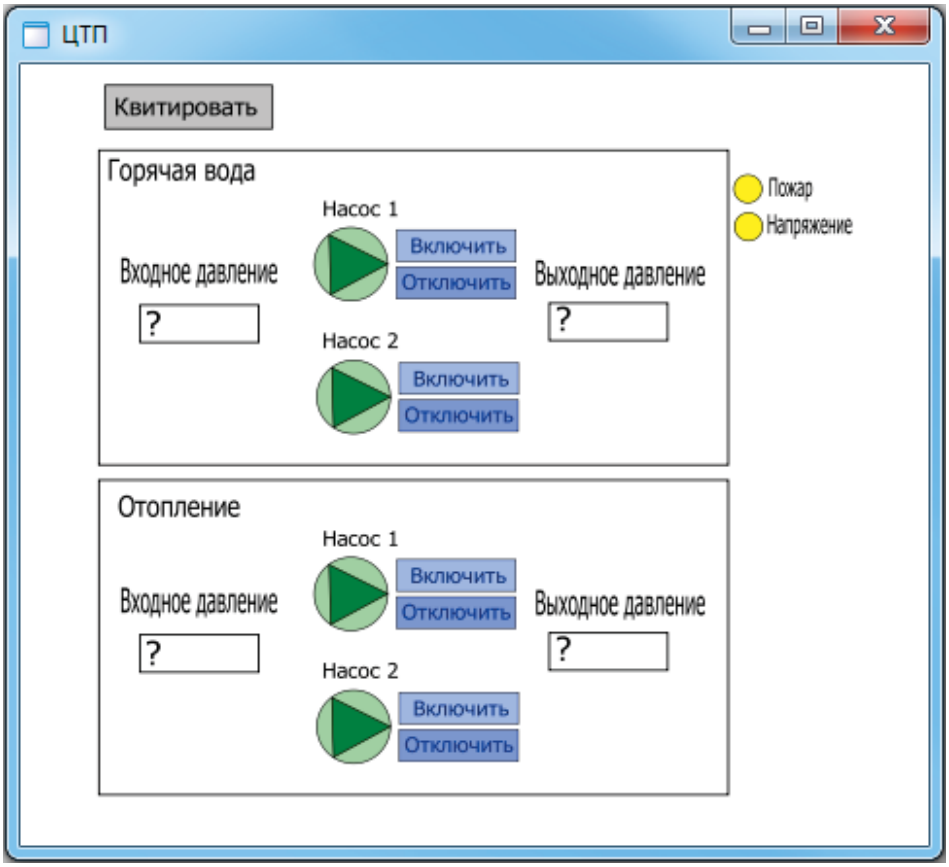


Рис. 118. Насосы включены.

Отключите первые насосы обеих групп, записав 0.

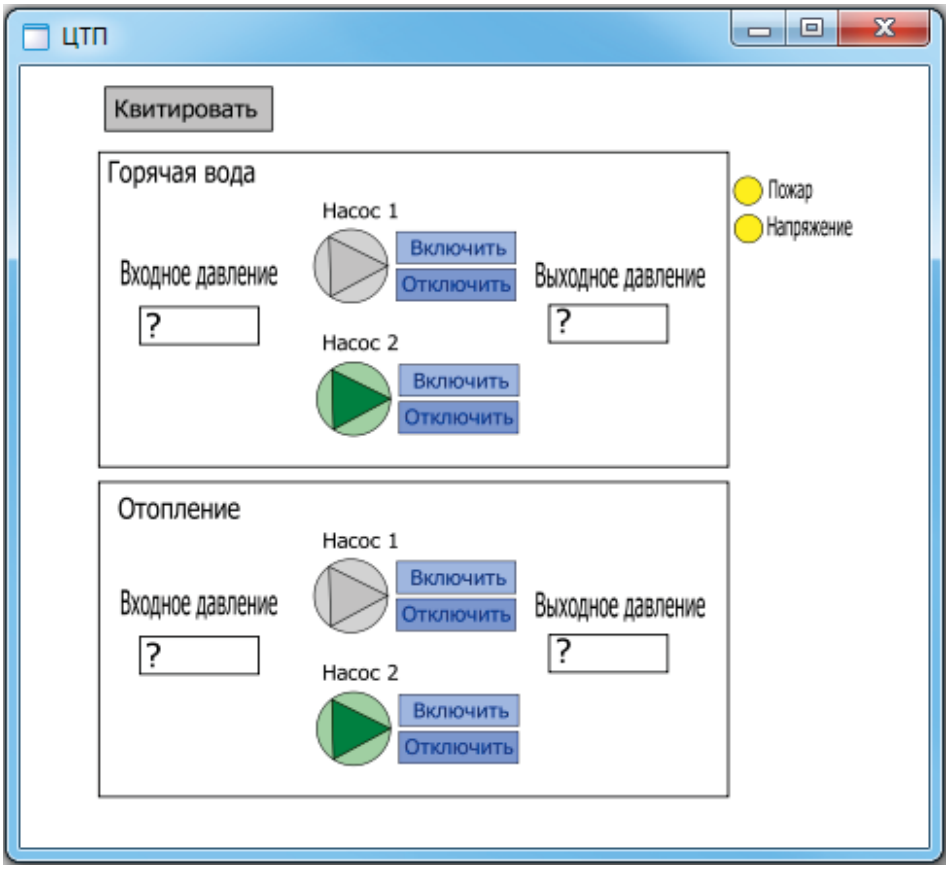


Рис. 119. Первые насосы отключены, вторые включены.

Состояние насосов отображается корректно. Давайте проверим работу индикаторов пожара и напряжения. Запишите 0 (нормальное состояние) в дискрет пожара и 1 в дискрет наличия напряжения.

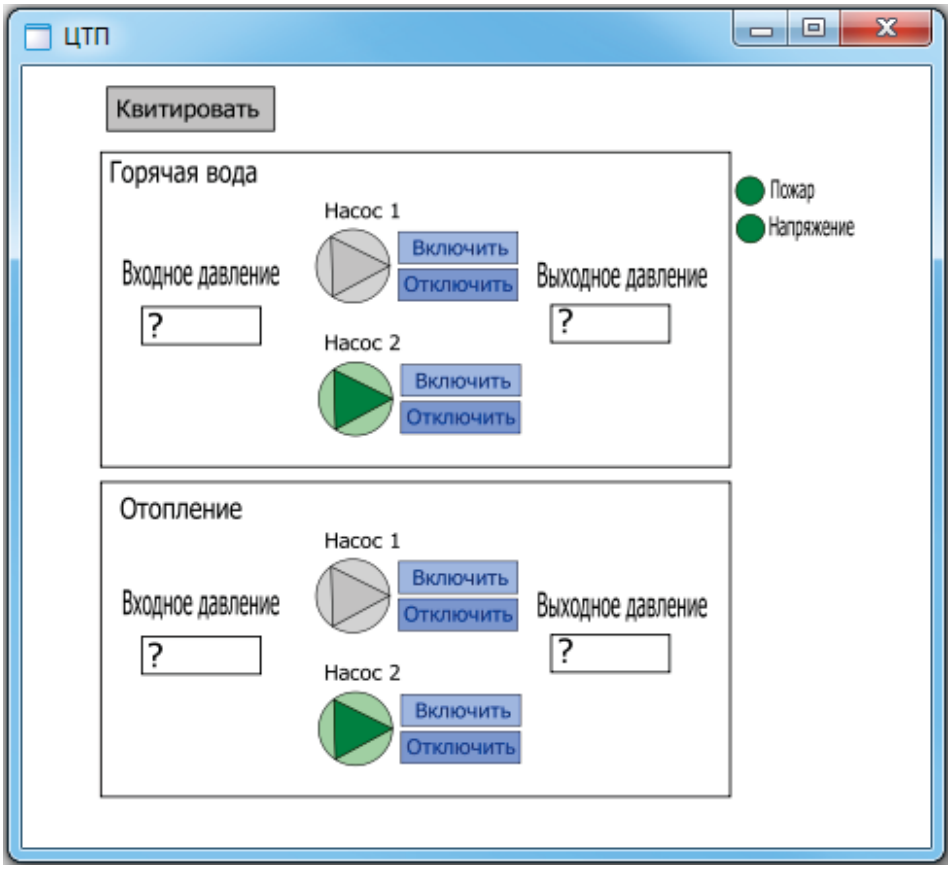


Рис. 120. Индикаторы пожара и напряжения в нормальном состоянии.

Индикаторы отработали как надо, теперь запишите 1 (аварийное состояние) в дискрет пожара и 0 в дискрет наличия напряжения.

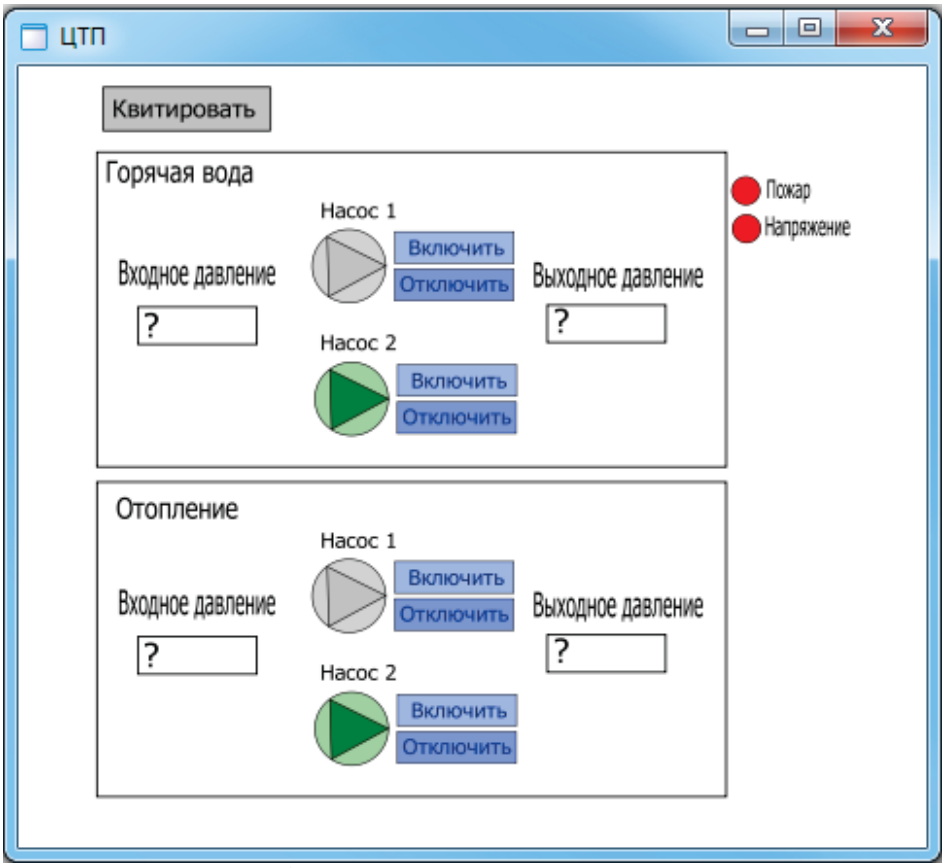


Рис. 121. Индикаторы показывают аварию.

Последнее, что мы должны проверить, это отображение значений входных и выходных давлений. Запишите в соответствующие аналоги любые числа, например 1,25 и др.

Рис. 122. Отображение значений входных и выходных давлений.

Поздравляю! Наше отображение работает в соответствии с заданием на АРМ и мы можем сдавать его заказчику.

10.4.5.10 Дополнительные возможности

Выше мы рассмотрели пример создания простейшего АРМ. В данном разделе описано, как можно расширить функциональность приложения, пользуясь возможностями программы Arm-Builder.

К рассматриваемым возможностям относятся:

1. Создание дополнительных окон.
2. Организация просмотра архивов.
3. Отображение оперативного журнала.
4. Организация работы с пользователями.

10.4.5.10.1 Дополнительные окна

До сих пор в нашем проекте ЦТП было одно рабочее окно - главное окно программы, содержащее изображение

технологического экрана. Рассмотрим, как добавить в проект другое окно.

Окно добавляется с помощью кнопки  , по умолчанию ему будет присвоено имя *Form_2*. Если окно должно быть показано при старте приложения, то все просто - надо найти его свойство **Автостарт**, и установить значение в *true*. Но как правило, окно должно быть показано в ответ на какое-то действие, поэтому мы пока оставим значение свойства **Автостарт** по умолчанию (*false*). У окна изменим размеры, свойство **Заголовок** ("Архив") и свойство **Расположение** (poScreenCenter).

Предположим, что нам нужно вызывать дополнительное окно при нажатии на какую-либо кнопку. Дорисуем в программе "Inkscape" еще одну кнопку "Архив" рядом с кнопкой "Квитировать". Чтобы открыть файл *ViewPicture.svg* для редактирования в программе "Inkscape", вызовите контекстное меню, щелкнув правой кнопкой мыши на SVG-панели, и выберите пункт *Редактировать Svg-файл*:

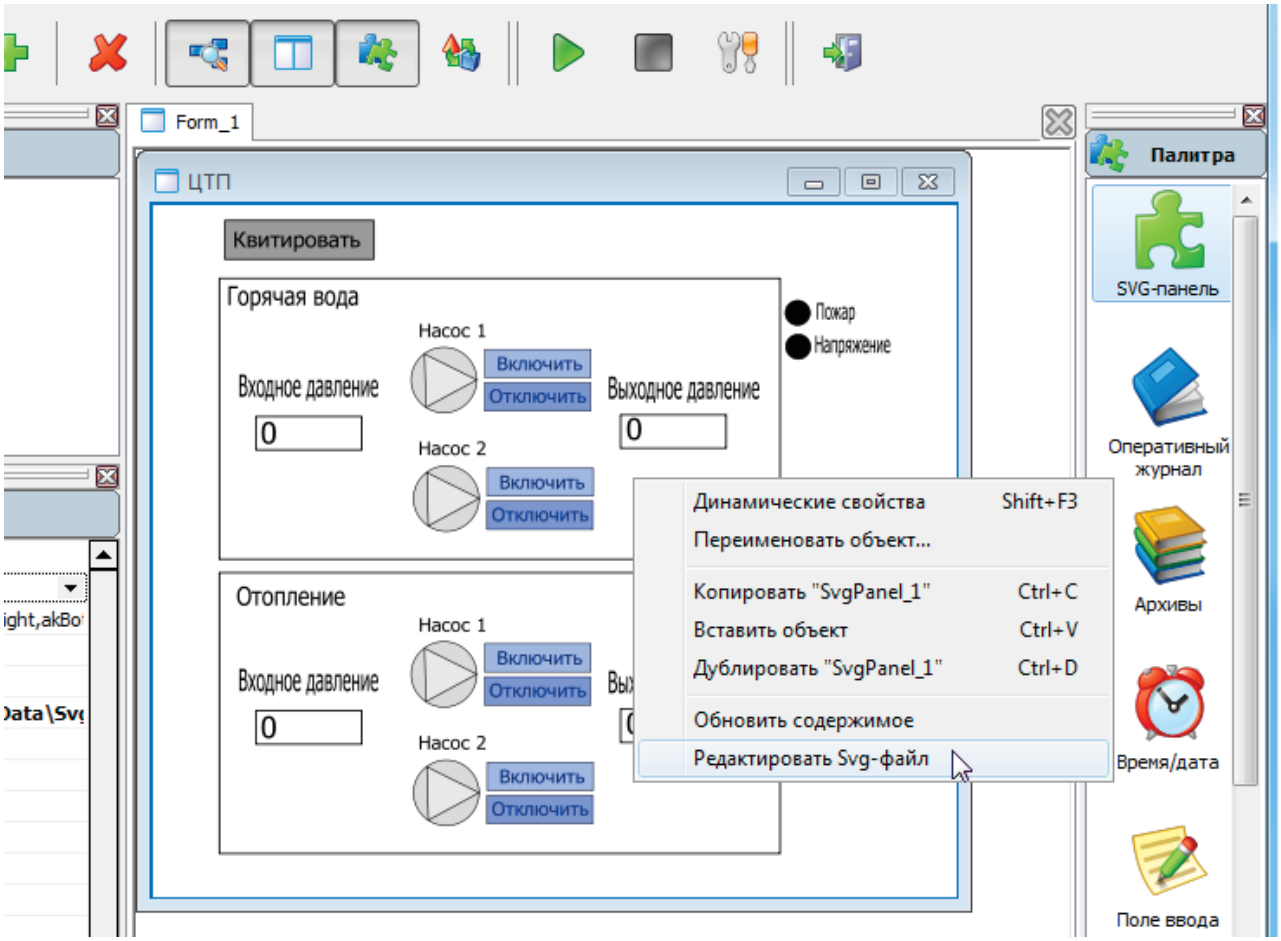


Рис. 123

После того как кнопка нарисована и изменения в Svg-файле сохранены, можно воспользоваться пунктом того же контекстного меню "Обновить содержимое". Добавим привязку на нажатие новой кнопки (вызываем редактор динамизации  , нажимаем "Добавить привязку"  , выбираем вкладку "События" - "Левое нажатие клавиши мыши", жмем *Выбрать*):

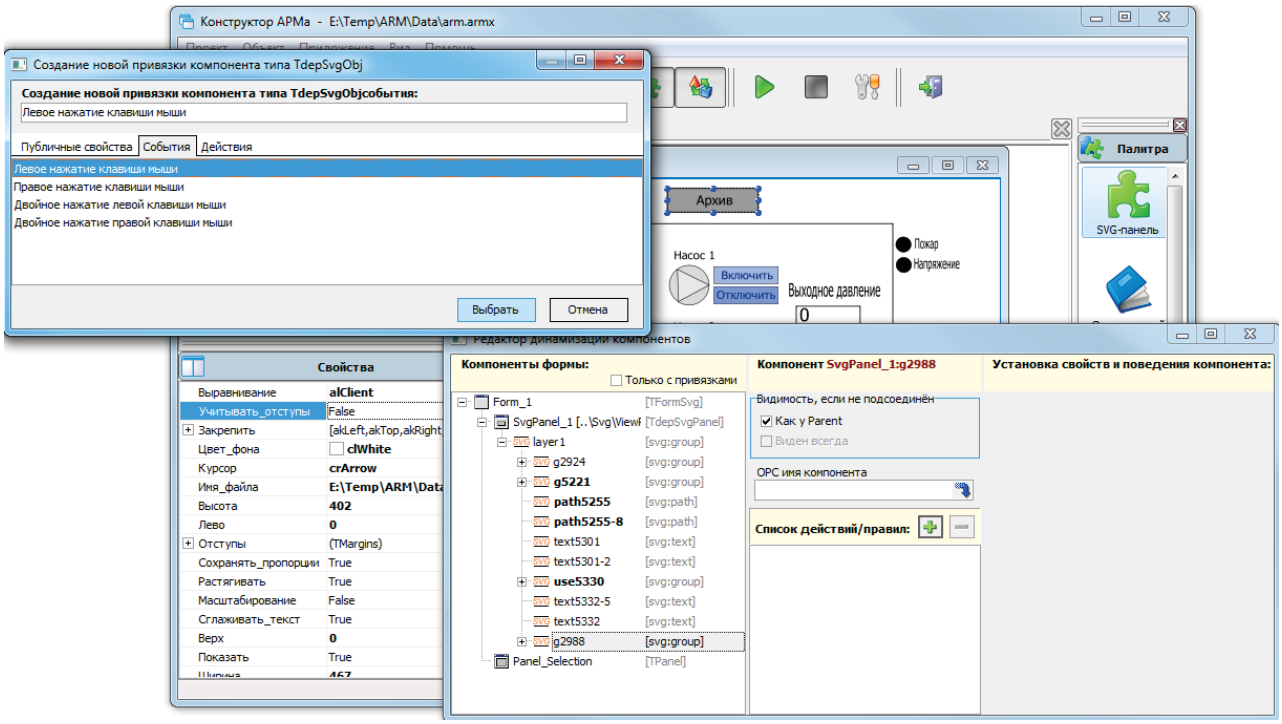


Рис. 124

У появившейся привязки выбираем:

1) элемент-источник - "Константа", зачение true,

2) элемент-приемник - "Свойство компонента",

3) нажимаем на кнопку  и выбираем компонент *Form_2* и свойство **cmd_Open** (внизу списка):

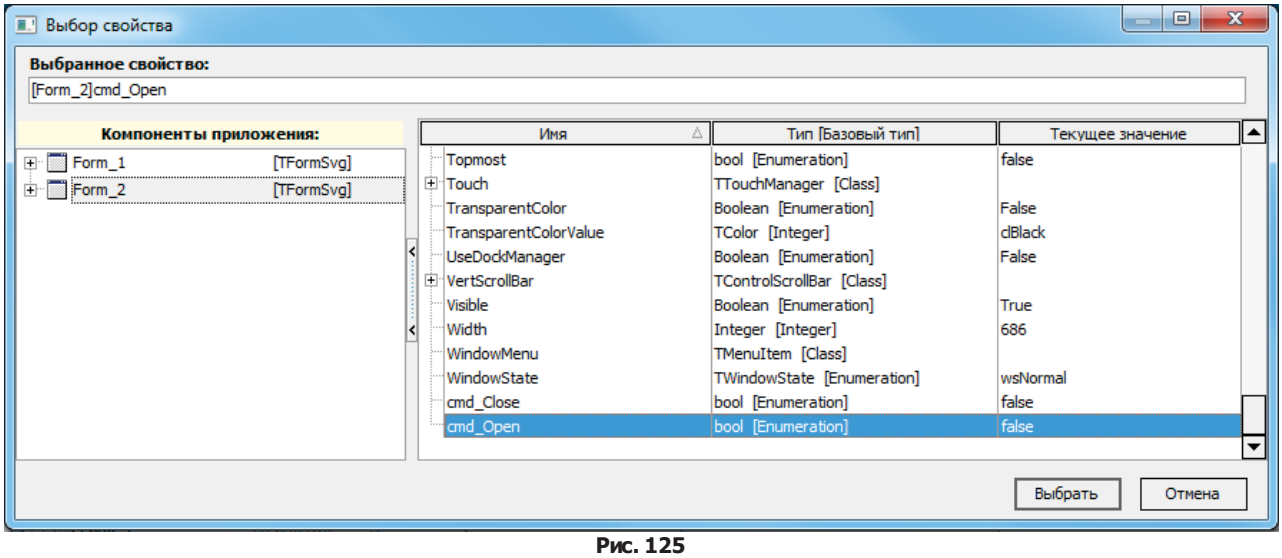


Рис. 125

Получившаяся привязка должна выглядеть примерно так:

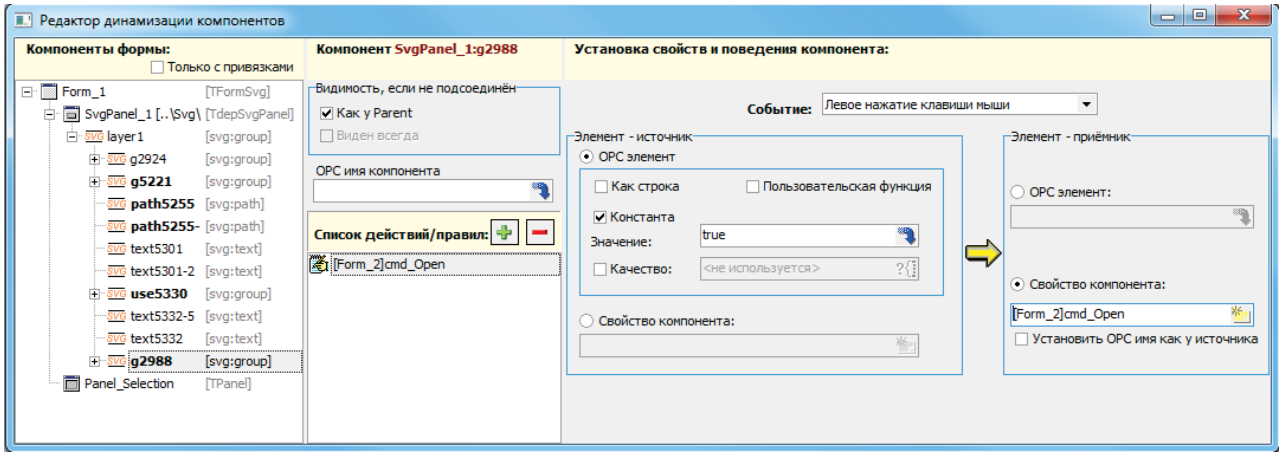


Рис. 126

Закройте редактор динамизации и сохраните проект.

Запустите приложение на выполнение с помощью кнопки  или клавиши F9. Как можно видеть, при нажатии на кнопку "Архив" на экране появляется наше второе окно, которое пока ничего не содержит:

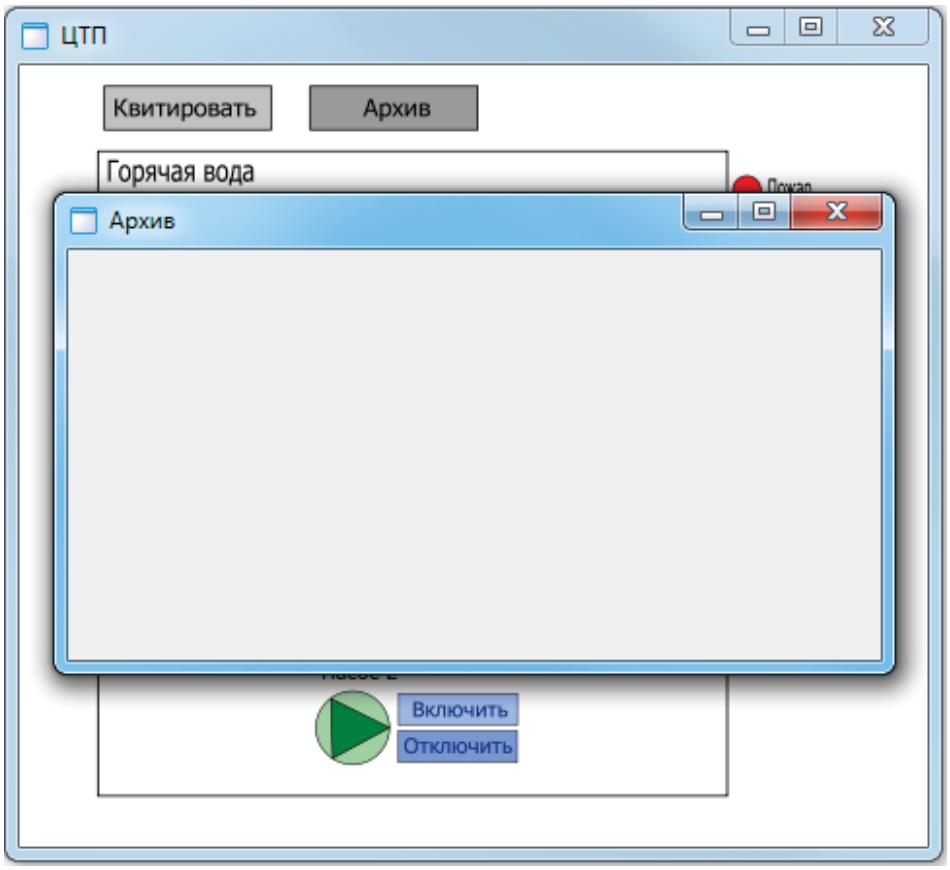


Рис. 127

Теперь можно наполнить второе окно содержимым - поместим туда компонент для просмотра архивов.

10.4.5.10.2 Просмотр архивов

Как мы знаем, OPC-модель при работе может архивировать значения элементов. Для этого нужно в параметрах модели в программе WinDeCont (*WinDeCont -> Сервис -> Параметры -> Модели*) установить флаг "Архивировать" и выбрать хранилище.

При этом в OPC-модели существует глобальная переменная *gArchName*, которая будет содержать имя используемого хранилища.

В модель необходимо добавить код для считывания значения это глобальной переменной. Для этого, например, в главном типе модели добавим элемент *archName* типа *iString* и переопределим функцию *tact()* следующим образом:

```
//--!-- virtual void TMainType::tact() ;public
void TMainType::tact()
{
 if (gArchName != "")
 {
    archName.assign(gArchName);
 }
 else
```

```
    {
       archName.makeBad();
    }

    inherited::tact();
  }
```

После внесения изменений в модель ее необходимо сохранить и пересобрать, после чего перезапустить модель в программе WinDeCont.

Далее в программе ARMBuilder в нашем проекте выделим в списке структура элемент "arm.armx" и его единственному свойству **Хранилище--OPC-имя** присвоим значение *archName*. Это можно сделать напрямую с клавиатуры, или нажав кнопку ... и выбрав имя элемента в модели.

После проделанных операций АРМ "знает", к какому хранилищу подключаться. Поэтому мы можем смело добавить компонент "Архивы" из палитры в окно *Form_2*. Установите свойство компонента **Выравнивание** в *alClient*, **Включить** в *true*, и при показе архивный компонент будет отображать архивные данные. Для простоты можно установить все свойства **Показывать_...** в *false*, кроме свойства **Показывать_дискреты**, тогда компонент будет отображать только дискреты.

Сохраните проект и запустите приложение. Если в модели правильно настроено архивирование, то при нажатии на кнопку "Архив" на экране появится окно с отображением архивных данных:

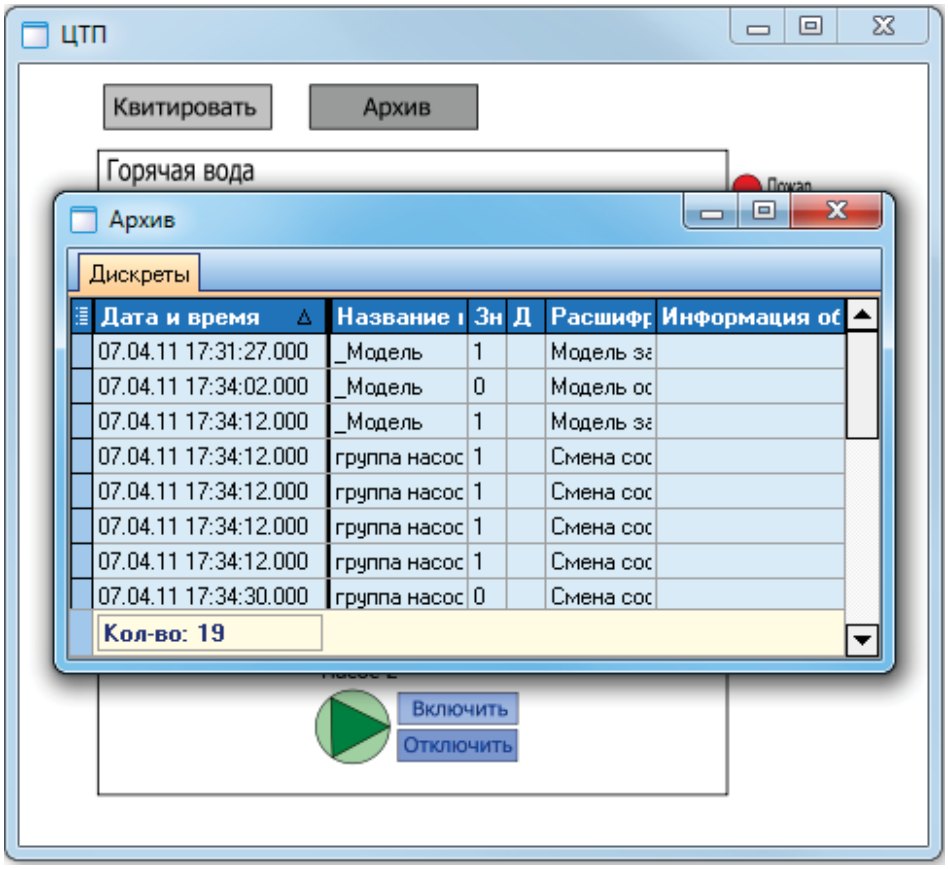


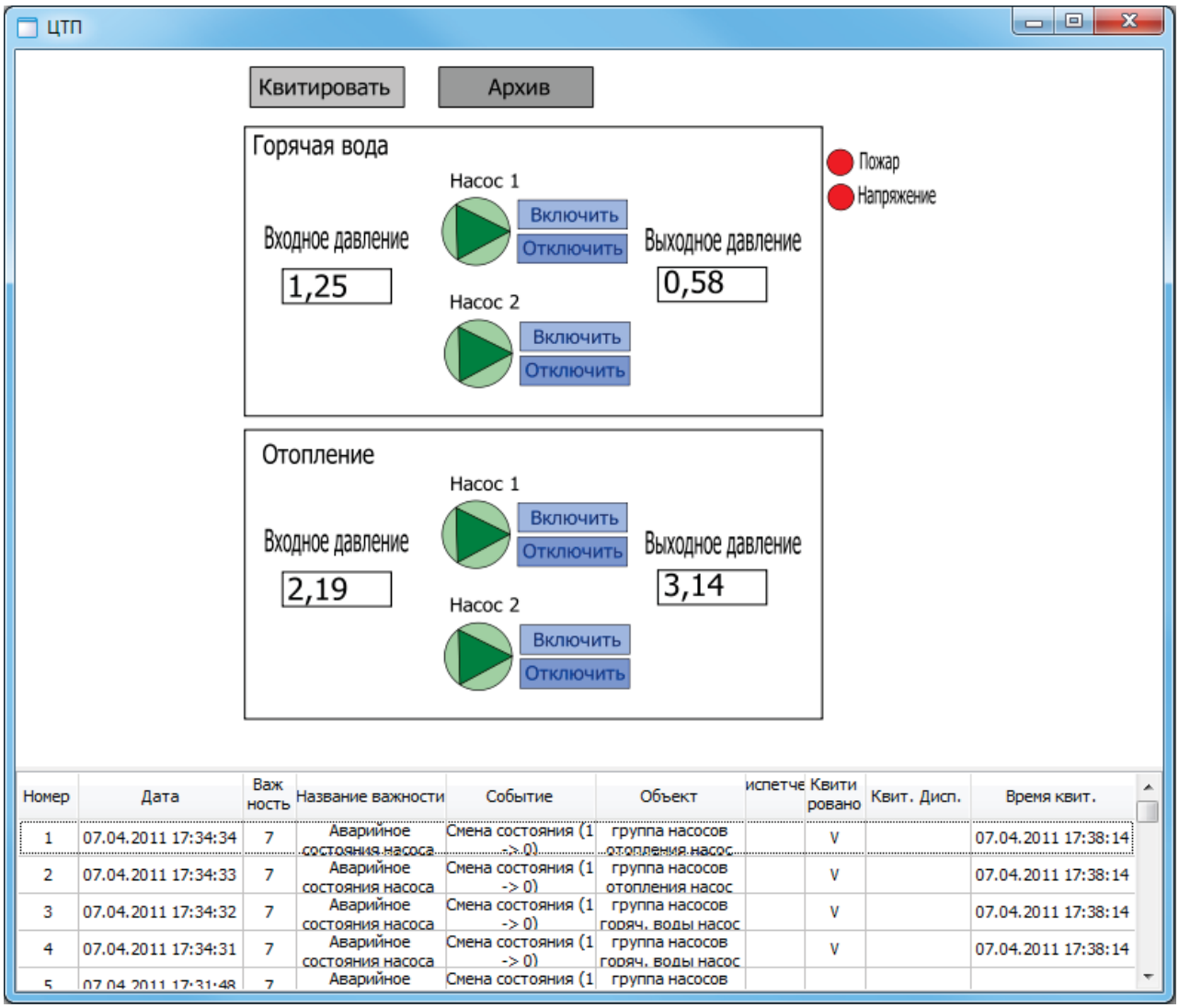
Рис. 128

В следующем разделе мы добавим в АРМ оперативный журнал.

10.4.5.10.3 Оперативный журнал

Если все настроено так, как показано в разделе Просмотр архивов, то добавить оперативный журнал в АРМ очень просто.

Добавьте компонент "Оперативный журнал" из палитры в окно *Form_1*. Установите свойство компонента **Выравнивание** в *alBottom*. Настройте размеры окна *Form_1* и высоту оперативного журнала. Сохраните проект и запустите приложение. Если в модели правильно настроена запись событий в оперативный журнал, то АРМ будет отображать содержимое оперативного журнала внизу главного окна:



Номер	Дата	Важность	Название важности	Событие	Объект	испетче	Квитировано	Квит. Дисп.	Время квит.
1	07.04.2011 17:34:34	7	Аварийное состояния насоса	Смена состояния (1 -> 0)	группа насосов отопления насос		V		07.04.2011 17:38:14
2	07.04.2011 17:34:33	7	Аварийное состояния насоса	Смена состояния (1 -> 0)	группа насосов отопления насос		V		07.04.2011 17:38:14
3	07.04.2011 17:34:32	7	Аварийное состояния насоса	Смена состояния (1 -> 0)	группа насосов гоpяч. воды насос		V		07.04.2011 17:38:14
4	07.04.2011 17:34:31	7	Аварийное состояния насоса	Смена состояния (1 -> 0)	группа насосов гоpяч. воды насос		V		07.04.2011 17:38:14
5	07.04.2011 17:31:48	7	Аварийное	Смена состояния (1	группа насосов				

Рис. 129

В следующем разделе будет рассмотрено, как организовать работу с пользователями.

10.4.5.10.4 Пользователи

При использовании АРМ важно разграничить права пользователей, имеющих доступ к приложению. Как правило, есть 4 категории пользователей - Гость, Администратор, Диспетчер, Телемеханик.

Чтобы воспользоваться возможностью разграничения прав доступа, необходимо указать наличие пользователей в OPC-модели. Для этого в программе "Конструктор OPC-модели", например, в главный тип добавим элемент ctpUsers типа tUsers. Укажем две категории пользователей - 0 (Администратор), 1 (Диспетчер):

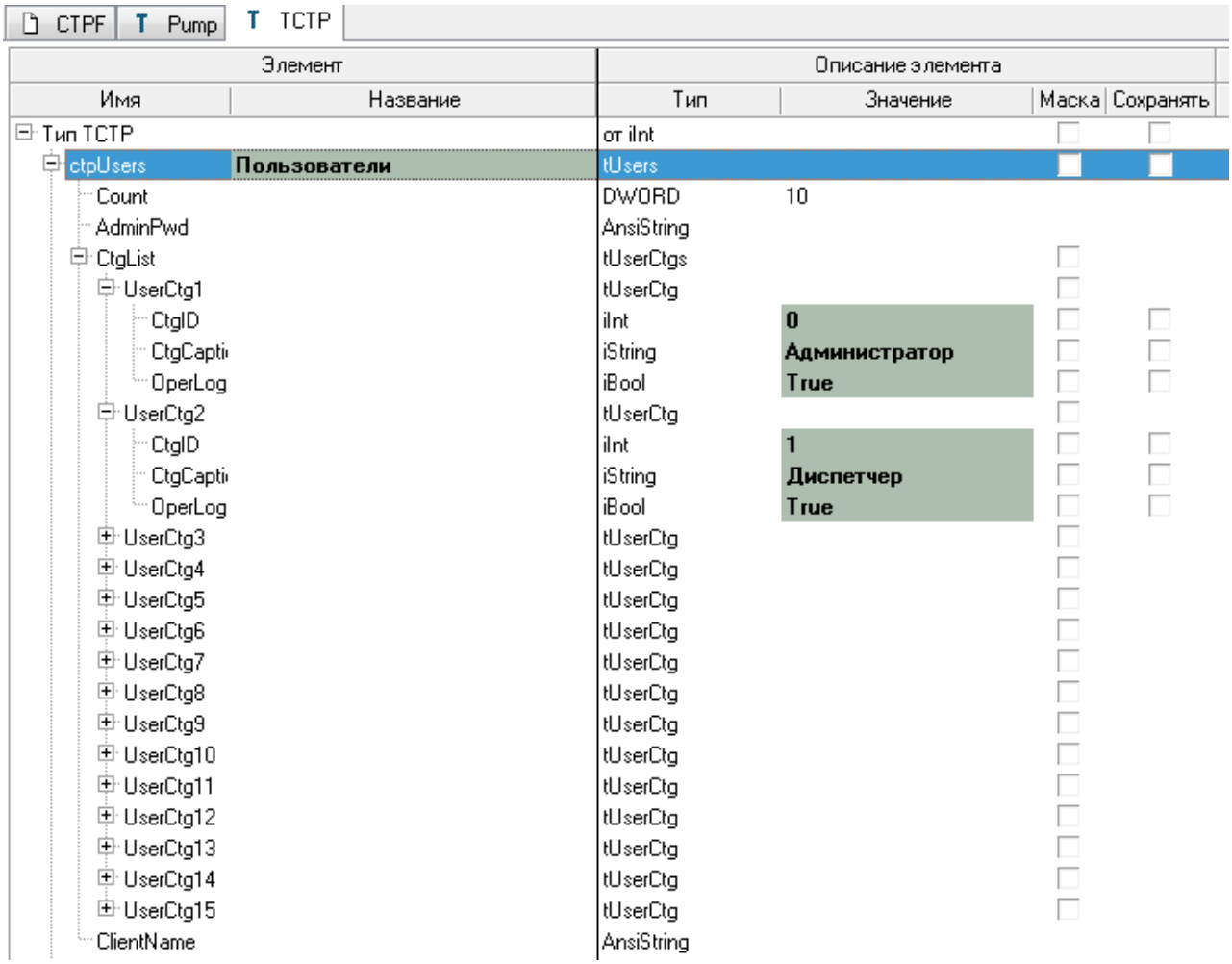


Рис. 130

Сохраните модель, пересоберите ее и перезапустите ее в программе WinDeCont. Теперь, если проимитировать аварийную ситуацию (чтобы замигала кнопка "Квитировать"), то просто так сквитировать тревогу уже нельзя. Для этого нужно войти в систему как зарегистрированный пользователь.

Рассмотрим, как организовать регистрацию, вход, выход и изменение параметров пользователя в АРМ.

Для этого добавим компонент "Пользователи" из палитры в окно *Form_1*. По умолчанию новый компонент будет назван *Users_1*. Свойство **OPC-имя** этого компонента установим в *ctpUsers*.

Дорисуем в программе "Inkscape" дополнительную кнопку к изображению технологического экрана. Пусть эта кнопка будет

называться "Пользователи" и располагаться рядом с кнопкой "Архив". Сохраните Svg-файл, обновите содержимое SVG-панели в проекте и сохраните проект. Рабочая область программы Arm-Builder должна после этого выглядеть примерно так:

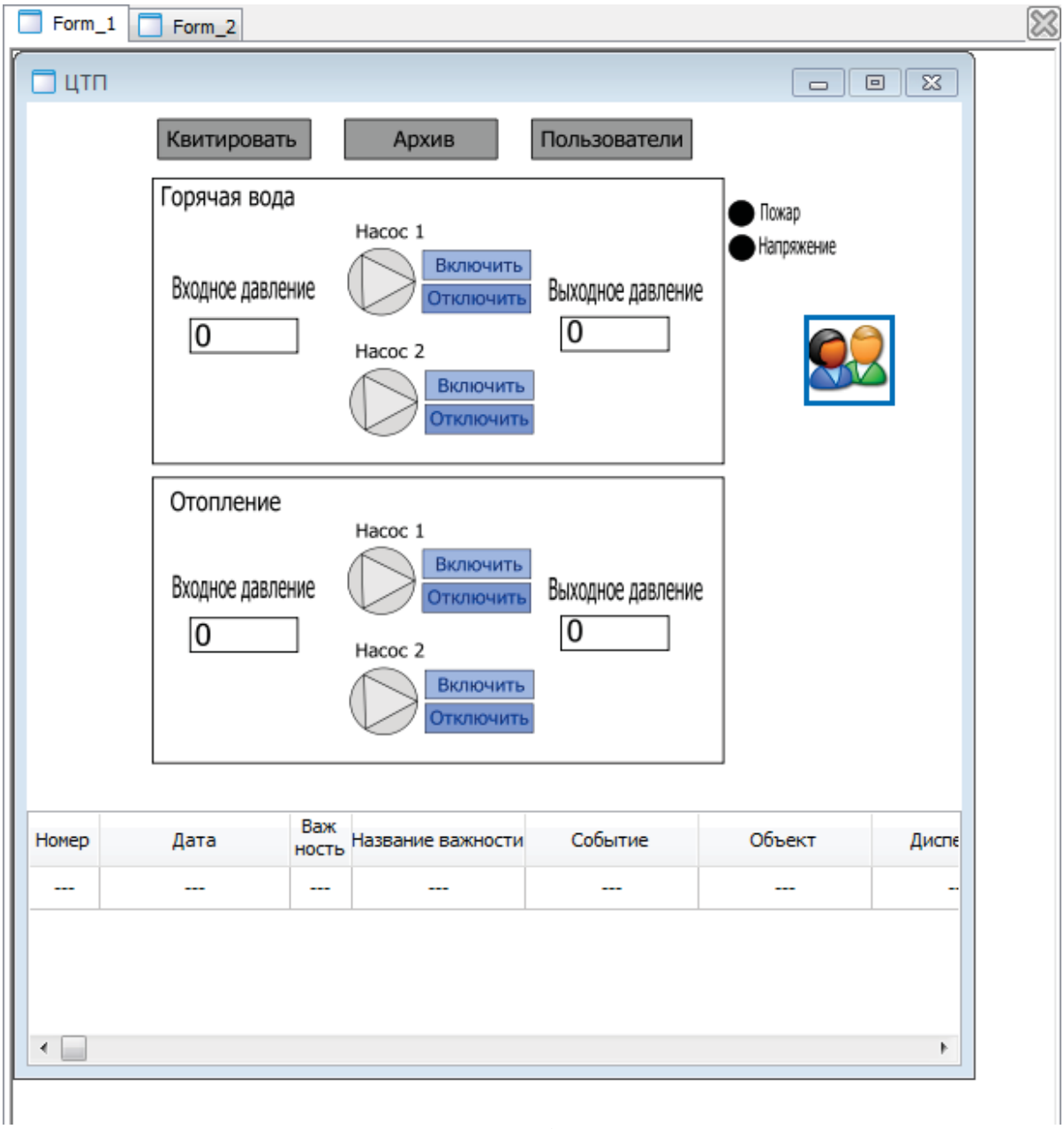


Рис. 131

Мы хотим, чтобы при нажатии на кнопку "Пользователи" появлялся диалог для работы с учетными записями пользователей. Поэтому добавим OPC-привязку на кнопку "Пользователи". Для этого откройте редактор динамизации (Shift+F3), щелкните левой кнопкой мыши на кнопке "Пользователи", в окне редактора динамизации нажмите "Добавить привязку"  ,

выберите вкладку "События", выберите "Левое нажатие клавиши мыши", нажмите *Выбрать:*

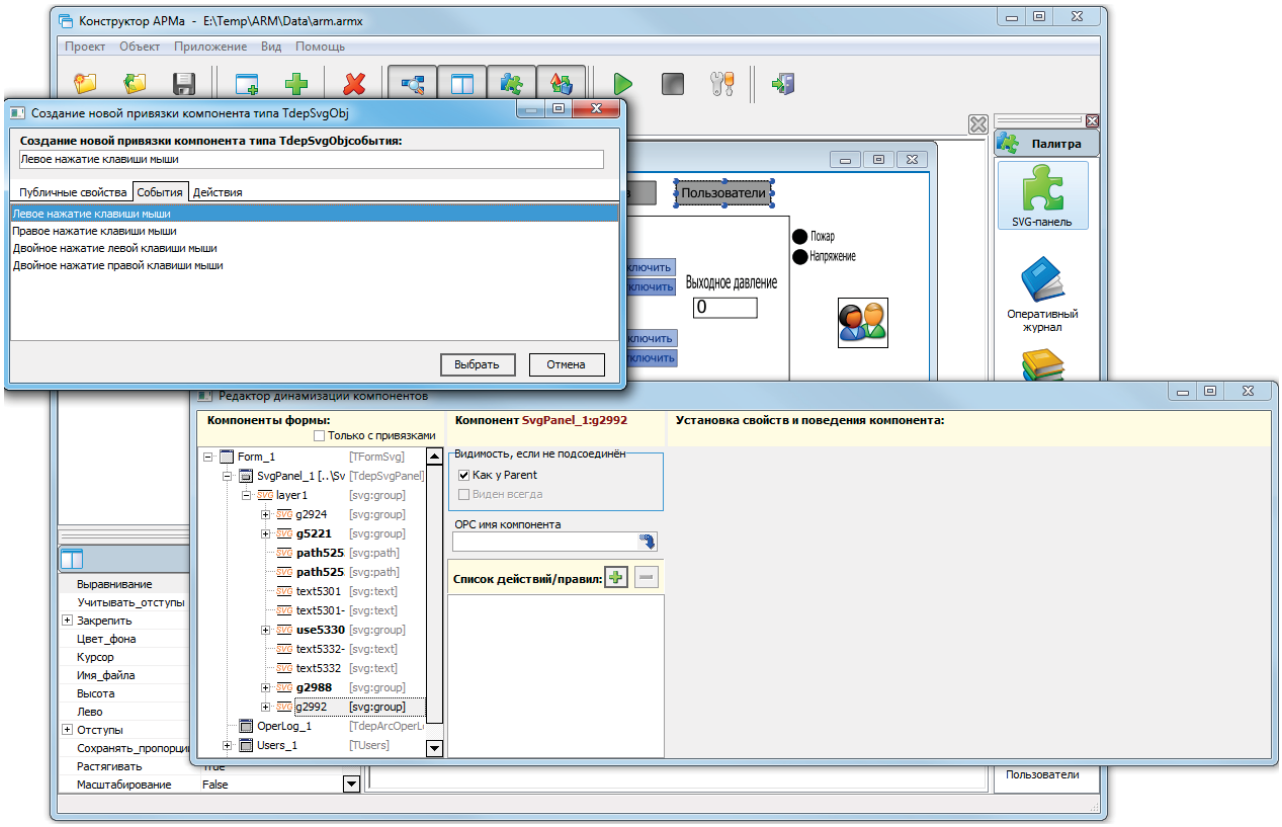


Рис. 132

У появившейся привязки выбираем:

1) элемент-источник - "Константа", зачение true,

2) элемент-приемник - "Свойство компонента",

3) нажимаем на кнопку  и выбираем компонент *Form_1 -> Users_1* и свойство **cmd_Login** (внизу списка):

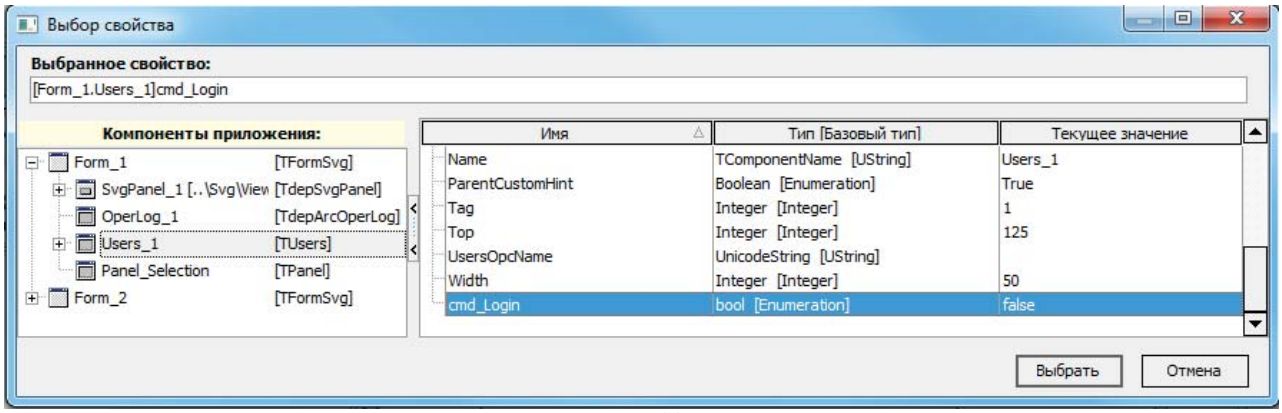


Рис. 133

Привязка на кнопку "Пользователи" готова. Теперь при нажатии на эту кнопку будет появляться диалог "Пользователи". Проверим, как это работает.

Сохраните проект (Ctrl+S) и запустите приложение (F9).

В заголовке главного окна приложения появилась запись "Пользователь не зарегистрирован". Нажмите на кнопку "Пользователи":

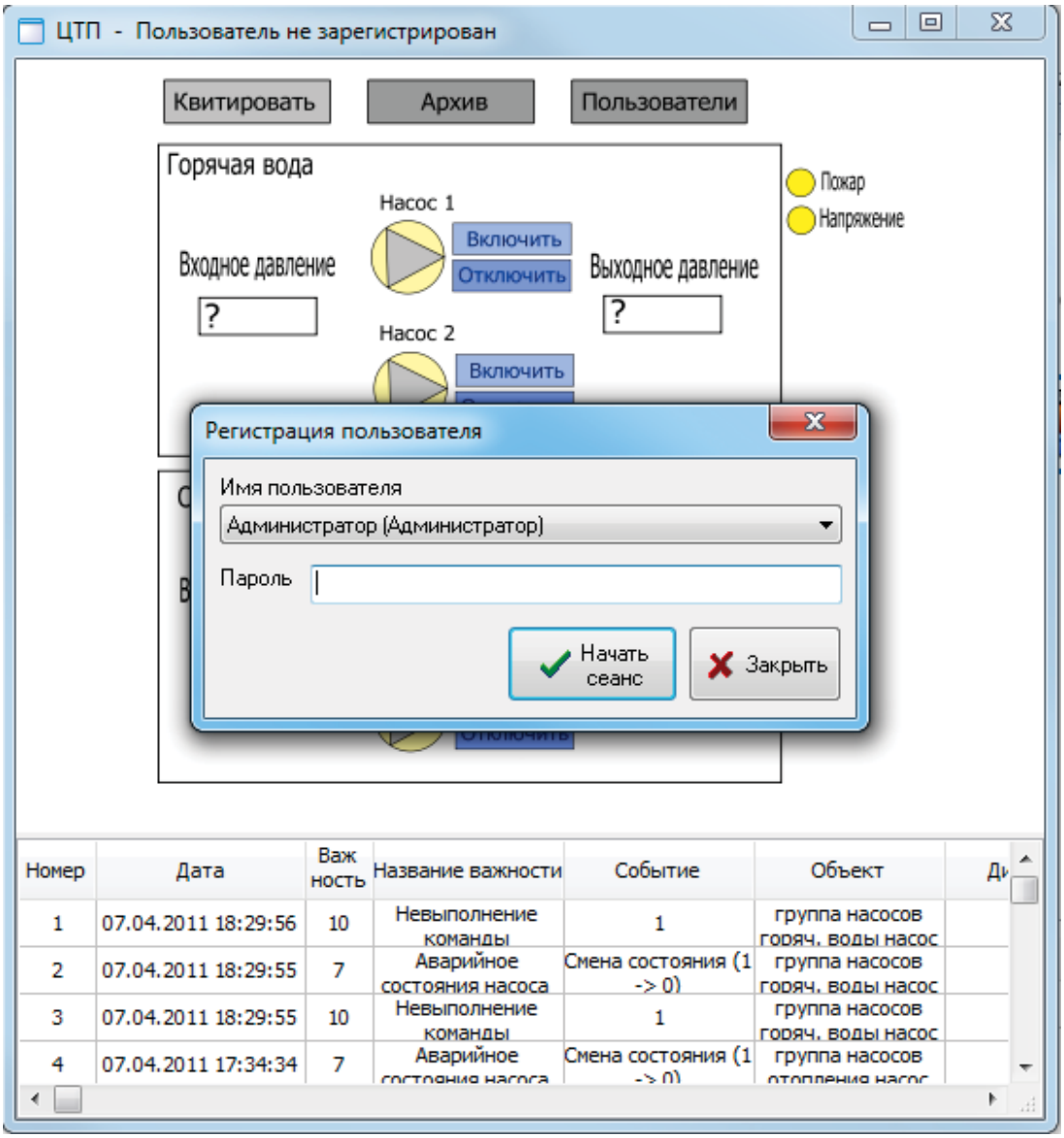


Рис. 134

Нажмите кнопку "Начать сеанс". В заголовке главного окна приложения должна появиться надпись "Пользователь Администратор". Вы успешно вошли в систему в качестве администратора. Если теперь нажать на кнопку "Пользователи", то появится диалог, позволяющий изменять параметры пользователей и добавлять новые учетные записи:

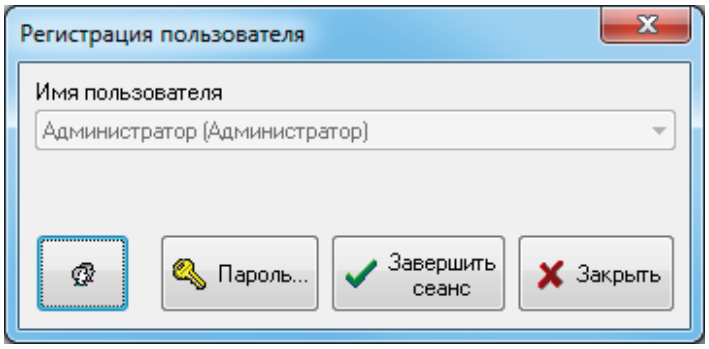


Рис. 135

Нажмите на кнопку, расположенную слева, и в появившемся диалоге добавьте новую учетную запись типа "Диспетчер". Завершите сеанс Администратора и войдите как Диспетчер. Теперь у Вас есть возможность записывать значения в элементы модели, т.е. при нажатии на кнопку "Квитировать" тревога будет сквитирована.

10.5 Клиент-серверная архитектура

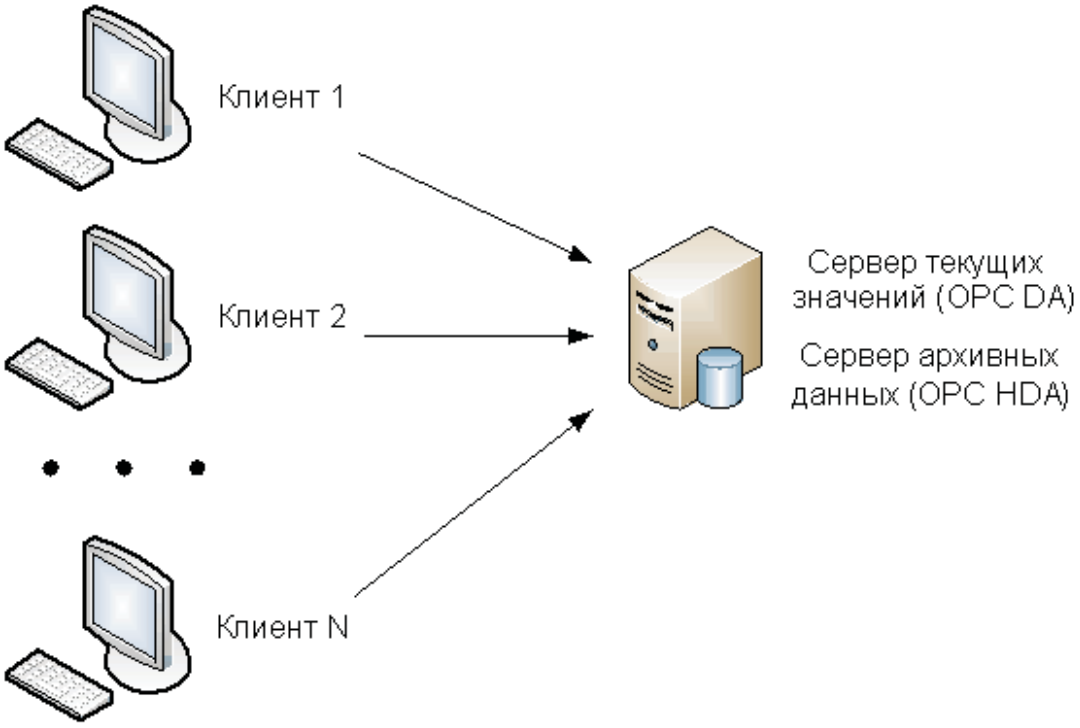


Рис. 1 Структура клиент-серверной архитектуры.

ПО "SyTrack-WRT" ARM-SERVER (**среда исполнения сервера**) и **ПО "SyTrack-WRT" ARM-CLIENT** (**среда исполнения клиента визуализации: просмотр мнемосхем, архивов, выдача команд**) необходимы, если система имеет двухуровневую клиент-серверную архитектуру, приведённую на рисунке 1. В качестве серверной части для отображения текущих значений выступает виртуальный контроллер «WinDecont» и OPC Модель, являющаяся надстройкой

для виртуального контроллера. Для архивных данных - база данных, управляемая сервером InterBase(6.0 или выше) или сервером Firebird (1.5 или выше). Система организована по технологии OPC DA для передачи текущих значений и OPC HDA для передачи архивных данных. Клиентская часть представлена так называемым "толстым" клиентом, то есть АРМ-ом диспетчера, на котором сконцентрированы основные правила работы системы и расположен пользовательский интерфейс программы. АРМ диспетчера представлен ПО "SyTrack-Tool" ARMBuilder.

ПО "SyTrack-WRT" ARM-Lib (**библиотека образцов для SCADA-диспетчеризации систем энергетики, энергоснабжений, вентиляции и др.**) предназначено для построения графического отображения АРМ диспетчера в ПО "SyTrack-Tool" ARMBuilder. **ПО "SyTrack-WRT" ARM-Lib** представляет из себя набор стандартных библиотек-шаблонов графических файлов, в которых поставляются различные изображения элементов с заданным поведением, в зависимости от области применения SCADA-системы.

Данные библиотеки могут идти в комплекте с ПО "SyTrack-Tool" ARMBuilder. При создании нового проекта файлы графических библиотек помещаются в папку SVG файлов проекта.

Для использования файлов библиотек необходимо, чтобы рядом с SVG файлом редактируемой мнемосхемы находилась папка «Pattern» с файлами шаблонов.

Для добавления образца на мнемосхему необходимо в редакторе Inkscape вызвать панель образцов(Ctrl + B), выбрать необходимый образец и добавить его на мнемосхему. После чего в редакторе динамизации привязать созданный образец к элементу модели.

В **ПО "SyTrack-WRT" ARM-SERVER** предусмотрены следующие варианты лицензирования:

По пользователю, который будет использовать разработанное автоматизированное рабочее место (в зависимости от уровня доступа пользователя меняются права использования программы):

- TM (телемеханик)
- RZA (инженер-РЗА)
- Operator (диспетчер)

По количеству точек ввода/вывода:

- до 100
- до 200
- до 300
- до 400
- от 400

По количеству клиентских мест:

- до 3
- от 3 до 5
- от 5 до 10
- от 10

В **ПО "SyTrack-WRT" ARM-Lib** предусмотрено лицензирование по области применения SCADA-системы:

- электроэнергетика
- горношахтная автоматика
- и др.
