Акционерное общество
«Экспериментальный завод научного приборостроения
со Специальным конструкторским бюро
Российской академии наук»

УТВЕРЖДЕН
КУНИ.505200.023-01.01 95-ЛУ

SCADA-СИСТЕМА "СОНАТА"

Руководство пользователя
КУНИ.505200.023-01.01 95

Листов 446

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

2025

Литера О1

АННОТАЦИЯ

Данный документ является руководством по созданию проектов автоматизированных систем управления технологическим процессом (далее АСУ ТП) в SCADA-системе «Соната». В документе описаны общие понятия, порядок работы, интерфейс главного меню программы и встроенных в нее редакторов приложений различного типа. Также описаны способы настройки приложений-драйверов, создания и настройки графических приложений, управляющих программ, отчетов и архивов, составляющих законченный проект АСУ ТП.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ

1.1. Назначение комплекса

Название класса программных систем, к которым относится система "СОНАТА" - SCADA, представляет собой аббревиатуру (Supervisory Control And Data Acquisition), буквально переводимую на русский язык как «Диспетчерское управление и сбор данных» (далее SCADA).

SCADA-система "СОНАТА" состоит из двух частей: средств разработки и среды исполнения.

Средства разработки – набор программ, предоставляющих разработчикам эффективный инструмент для создания проекта автоматизации технологического процесса.

Среда исполнения включает набор программ, выполняющих вспомогательные функции, и набор программ, работающих в реальном времени на контроллерах, АРМах (автоматизированных рабочих местах), архивных станциях и т.д.

Программы, выполняющие вспомогательные функции, могут запускаться в процессе работы системы и, по выполнению своих функций, завершаются. К вспомогательным функциям относятся печать на принтер, считывание конфигурации контроллеров, вывод информации о работающем дистрибутиве SCADA "СОНАТА", поверка входных аналоговых каналов, администрирование пользователей работающей системы и др.

Программы, работающие в реальном времени, выполняют функции сбора данных, исполнение технологических алгоритмов, архивирование и отображение технологической информации.

1.2. Описание комплекса

Таблица 1.1 - Таблица, описывающая принадлежность модулей SCADA системы "СОНАТА" к соответствующей части (средства разработки и средства исполнения в режиме реального времени)

Имя приложения	Средства разработки	Среда исполнения	
		Программы, выполняющие вспомогательные функции	Программы реального времени
AlertArchive			+
Archive			+
ArchiveViewer		+	
Bridge			+
ControlCenter	+	+	
Debugger	+		
Distributer	+	+	
DTS_Editor	+		
EventLogger			+
HTMLEditor	+		
IEC60870_Editor	+		

Имя приложения	Средства разработки	Среда исполнения	
		Программы, выполняющие вспомогательные функции	Программы реального времени
IEC61850			+
IEC61850_Editor	+		
IECConsoleEngine			+
IECEditor	+		
IECWindowEngine			+
ImagePrinter		+	
Katren_LOCALBUS_Editor	+		
Katren_MODBUS			+
Katren_MODBUS_Editor	+		
KM04			+
KM04_Editor	+		
Loader			+
LuaEngine			+
MODBUS			+
MODBUS_Editor	+		
OPCUA			+
OPCUA_Editor	+		
OPCUA_Server			+
OPCUA_Server_Editor	+		
ProjectManager	+		
ReadSwitch		+	
ReportEditor	+		
ReportEngine			+
SignalViewer	+	+	
SNMP			+
SNMP_Editor	+		
SonataVer		+	
Sonet_Failover_Editor	+		
Sonet_LOCALBUS_Editor	+		
Sonet_MODBUS			+
Sonet_MODBUS_Editor	+		
Sound			+
SoundEditor	+		
SourceEditor	+		

Имя приложения	Средства разработки	Среда исполнения	
		Программы, выполняющие вспомогательные функции	Программы реального времени
SPABUS			+
SPABUS_Editor	+		
SQL			+
SQL_Editor	+		
Telekont2_MODBUS			+
Telekont2_MODBUS_Editor	+		
UserListEditor	+	+	
Verification		+	
WebServer			+
WebBrowser		+	

1.3. Возможности комплекса

1.3.1. Возможности средств разработки

Средства разработки реализуют следующие функции:
- создание таблицы сигналов проекта;
- привязка сигналов к физическим каналам ввода-вывода;
- настройка параметров архивирования;
- создание технологических алгоритмов на языках FBD, ST, LUA;
- разработка интерфейса пользователя;
- создание шаблонов отчётов.

1.3.2. Возможности среды исполнения

Средства исполнения системы «Соната» реализуют следующие функции, типичные для систем подобного класса:

- сбор информации с устройств нижнего уровня (датчиков, контроллеров);
- прием и передача команд оператора/диспетчера на контроллеры и исполнительные устройства (дистанционное управление объектами);
- сетевое взаимодействие с информационной системой предприятия (с вышестоящими службами);
- автоматизированное управление объектом автоматизации;
- отображение параметров технологического процесса и состояния оборудования с помощью мнемосхем, таблиц, графиков и т.п. в удобной для восприятия форме;

- оповещение эксплуатационного персонала об аварийных ситуациях и событиях, связанных с контролируемым технологическим процессом и функционированием программно-аппаратных средств автоматизированной системы управления технологическим процессом (далее АСУ ТП) с регистрацией действий персонала в аварийных ситуациях;
- хранение полученной информации в архивах;
- представление текущих и накопленных (архивных) данных в виде графиков (тренды);
- вторичная обработка информации;
- формирование сводок и других отчетных документов по созданным на этапе проектирования шаблонам.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Технические средства, необходимые для выполнения комплекса "Соната"

SCADA-система "Соната" может функционировать на компьютерах и контроллерах, построенных на следующих современных архитектурах процессоров:

- x86, x86_64: AMD, Intel, Vortex;
- ARM: v7, v8, v9, v11;
- MIPS: MIPS32, MIPS64;
- Эльбрус: Эльбрус-2С+, Эльбрус-4С, Эльбрус-8С;

Проект автоматизации на системе "Соната" может функционировать на разных типах процессоров и разных типах операционных систем одновременно. Например, проект может содержать контроллеры, выполненные на процессорах Vortex86, работающие под управлением операционной системы QNX 6.5, и АРМ, работающие на процессорах "Эльбрус-2С+" и операционной системе "Эльбрус". Таким образом, система "Соната" позволяет строить гетерогенные системы управления.

ВНИМАНИЕ! При эксплуатации проекта загрузка процессора на любом из узлов системы не должна превышать 70%. Однако, для контроллеров "Сонет", работающих под управлением операционной системы QNX 6.5.0, допускается загрузка процессора до 100 % при условии отсутствия потери данных.

2.1.1. Технические средства для организации контроллера ввода-вывода

Контроллер ввода-вывода предназначен для опроса аналоговых и дискретных каналов, выполнения примитивных алгоритмов управления и блокировки, передачи полученных данных вычислительному узлу или АРМ.

Минимальные требования:

- процессор: 200 МГц;
- оперативная память: 64 МБ;
- дисковая память: 256 ГБ;
- сетевой интерфейс: 100 МБит/с.

Контроллер в минимальной конфигурации может обеспечить опрос до 128 каналов (физ. каналы и статусы) и обработку порядка 1000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

Рекомендуемые требования:

- процессор: 1 ГГц;
- оперативная память: 512 МБ;
- дисковая память: 1 Гб;
- сетевой интерфейс: 100 МБит/с.

Контроллер в рекомендуемой конфигурации может обеспечить опрос до 512 каналов (физ. каналы и статусы) и обработку порядка 10000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

В вышеуказанной конфигурации загрузка процессора контроллера составляет не более 60%.

2.1.2. Технические средства для организации вычислительного узла

Вычислительный узел представляет собой промышленный компьютер без монитора, с постоянным питанием, на котором выполняются основные технологические алгоритмы проекта. Вычислительный узел используется в том случае, когда не хватает вычислительной мощности контроллеров.

Минимальные требования:

- процессор: 1 ГГц;
- оперативная память: 1 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 100 МБит/с.

Рекомендуемые требования:

- процессор: 2 ГГц, 4 ядра;
- оперативная память: 2 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 1 ГБит/с.

Вычислительный узел в рекомендуемой конфигурации может обрабатывать порядка 100000 элементарных ячеек (логический сигнал, элемент массива или поле структуры).

2.1.3. Технические средства для организации АРМ

АРМ (автоматизированное рабочее место) – компьютер с монитором, предназначенный для визуального отображения состояния проекта и приёма команд управления от оператора. Если в качестве АРМ используется постоянно включенный компьютер, то на него можно возложить функции вычислительного узла проекта. В этом случае нужно объединять минимальные требования вычислительного узла с минимальными требованиями АРМ.

Минимальные требования:

- процессор: 1 ГГц;
- оперативная память: 2 Гб;
- дисковая память: 8 Гб;
- сетевой интерфейс: 100 МБит/с.

Рекомендуемые требования:

- процессор: 3 ГГц, 4 ядра;
- оперативная память: 8 Гб;
- дисковая память: 32 Гб;
- сетевой интерфейс: 1 ГБит/с.

2.1.4. Технические средства для организации архивной станции

Архивная станция представляет собой промышленный компьютер с постоянным питанием, осуществляющий запись на диск архивов значений сигналов, событий и тревог. В качестве архивной станции может выступать вычислительный узел или АРМ (при наличии постоянного питания). Архивная станция, прежде всего, требовательна к надёжности и быстродействию дисков. Рекомендуется использование твердотельных дисков (SSD) таких производителей, как Intel, Kingston, SanDisk.

В настоящее время система "Соната" использует строго периодическую запись на диск. Под этим подразумевается, что даже если сигнал не изменяет своего значения (например, дискретный), то запись данных всё равно производится с указанным интервалом. Данная архитектура используется для обеспечения возможности создания дублированных архивов.

Рекомендуемые требования:

- процессор: 2 ГГц, два или более ядра;
- оперативная память: 4 ГБ;
- дисковая память: SSD 256 ГБ со скоростью записи не менее 250 МБ/с;
- сетевой интерфейс: 1 ГБит/с.

Архивная станция в рекомендуемой конфигурации обеспечивает запись порядка 100000 записей в секунду. Таким образом, если планируется архивировать 1000 сигналов, то 100000/1000 = 100 раз в секунду можно успевать сохранять каждый сигнал.

2.2. Программные средства, необходимые для работы комплекса "Соната"

Для SCADA системы "Соната" поддерживаются следующие платформы:

- Microsoft Windows: 7 32/64 bit, 8 32/64 bit, 10 32/64 bit, server 2012;
- Linux x86, x86_64: Ubuntu 14.4 и старше:
  - необходимо наличие библиотек Qt 4.8.6 или Qt 4.8.7;
- Linux ASTRA - CommonEdition и SpecialEdition (gcc v.4.7.2):
  - необходимо наличие библиотек Qt 4.8.7;
  - проверьте наличие библиотеки libQtSvg нужной версии (она часто отсутствует в общем репозитории Qt) и установите её, в случае отсутствия. Проверить наличие данной библиотеки можно командой: **sudo ldconfig -p | grep libQtSvg**. Если вывод данной команды пустой, то в операционной системе не установлена библиотека libQtSvg. Для установки из репозиториев с помощью стандартных утилит apt-get или Synaptic необходим пакет libqt4-svg.
- Linux для ARM v.7 (gcc v.4.4.0):
  - необходимо наличие библиотек Qt 4.8.6;
- Linux Elbrus: e2k-2c+-linux, gcc v.4.4.0:
  - необходимо наличие библиотек Qt 4.8.6;
- QNX: x86 v. 6.5.0 (ЗОСРВ "Нейтрино"), gcc v.4.4.2:
  - необходимо наличие библиотек Qt 4.8.7;

Для остальных платформ система "Соната" может быть скомпилирована по требованию заказчика. Необходимым условием является наличие компилятора языка C++, совместимого с GCC версии 4.4.0 и старше, а также графической библиотеки Qt версии 4.8.4, 4.8.5, 4.8.6 или 4.8.7.

3. ВЫПОЛНЕНИЕ КОМПЛЕКСА ПРОГРАММ

3.1. Структура проекта

Проект автоматизации, созданный при помощи SCADA-системы "Соната", представляет собой совокупность приложений, исполняемых на узлах, обменивающихся между собой информацией (сигналами) в реальном времени. С точки зрения обмена информацией все приложения (драйверы, алгоритмы, мнемосхемы и т.п.) являются равноправными. Проект требует для своего исполнения аппаратных средств, которые могут быть компьютерами или специализированными контроллерами. В проекте может использоваться множество контроллеров и компьютеров, обьединенных в компьютерную сеть. Для проекта каждый такой компьютер или контроллер является узлом компьютерной сети. Если на узле нет ни одного приложения SCADA-системы "Соната", то он не является частью проекта.

Контроллер - устройство управления в электронике и вычислительной технике.

Приложение – представляет собой программу (драйвер, алгоритм, мнемосхема т.п.), исполняемую на узле системы. На этапе проектирования разработчик создает или настраивает различные типы приложений, после чего размещает экземпляры приложений нужного типа на узлах системы.

Узел – компьютер или контроллер, на котором могут быть запущены приложения среды исполнения SCADA-системы «Соната». Основными характеристиками узла являются имя, набор IP-адресов сетевых интерфейсов, через которые осуществляется обмен данными между приложениями и перечень приложений. Если количество IP- адресов больше одного, то данные отправляются через все каналы передачи данных одновременно, организуя тем самым дублирование сетевых интерфейсов.

Минимальный по составу рабочий проект, созданный в SCADA-системе «Соната», должен состоять хотя бы из одного узла и одного приложения (то есть, хотя бы из одного аппаратного и одного программного средства). Такой проект может, например, выполняться на контроллере, на котором запущено приложение, обрабатывающее физические сигналы (приложение-драйвер).

Физические сигналы – это сигналы с аналоговых (напряжения, тока, сопротивления и т.п.) и дискретных датчиков, а также выходные аналоговые (напряжения и тока) и дискретные (например, сигнальные лампочки или реле) сигналы.

Проект, состоящий из одного приложения и выполняемый на одном узле, может служить только для целей отладки и не имеет практической ценности. Минимальный проект, имеющий практическую ценность, должен, помимо приложения-драйвера, включать в себя хотя бы еще одно приложение, например, управляющую программу. Если на узле запущены два приложения с различной функциональностью, то необходимо решить задачу обмена информацией между этими приложениями в рамках единого проекта. Носителем информации в проекте является сигнал. Приложения могут обмениваться значениями сигналов. Полный список сигналов приложения называется интерфейсом приложения.

Сигнал – носитель информации в проекте.

Интерфейс приложения – полный список сигналов приложения.

Физические сигналы преобразуются в сигналы проекта с помощью приложения-драйвера.

В ряде случаев необходимо преобразование значения физического сигнала в другие физические единицы (например, преобразование сигнала от термопары, который измеряется в вольтах, в градусы Цельсия, или Кельвина, которым соответствуют измеряемые в вольтах значения). Для этой цели в SCADA-системе «Соната» вводится специальный элемент – преобразователь значения сигнала.

Преобразователь значения сигнала – описание монотонно возрастающей или монотонно убывающей функции, позволяющей однозначно осуществлять преобразование аргумента в значение и обратно.

Преобразователь значения сигнала позволяет приложению использовать значения сигналов, соответствующие значениям физических величин, а не только тем значениям, которые непосредственно от датчиков получает приложение-драйвер.

В SCADA-системе «Соната» сигналы могут быть локальными и глобальными:

- локальный сигнал доступен только внутри работающего приложения;

- глобальный сигнал доступен для любого приложения, работающего в составе данного проекта.

В SCADA-системе «Соната» приложения могут обмениваться значениями локальных сигналов путем создания связей приложений.

Связь приложений – прямое указание на обмен значениями сигнала одного работающего приложения и сигнала другого работающего приложения.

Другой способ передать значение сигнала приложениям – объявить его глобальным.

Приложение-драйвер, управляющая программа, графическое приложение и т.п. выполняют разные функции. Для выполнения похожих задач в проекте могут существовать приложения со схожей функциональностью. Для быстрого создания приложений со схожей функциональностью в SCADA-системе «Соната» существуют виды приложений.

Вид приложения – набор определённых свойств и функциональностей приложения, заданных SCADA-системой «Соната». Все приложения, относящиеся к данному виду, обладают заданными SCADA-системой «Соната» свойствами и функциональностью.

Примерами видов приложений могут быть приложение-драйвер, среда исполнения программы на технологическом языке, приложение-архив и т.п. Некоторые виды приложений обладают только теми свойствами и функциональностью, которые заданы SCADA-системой «Соната», другие, помимо заданных SCADA-системой «Соната» свойств, могут быть дополнены новыми качествами при создании и редактировании проекта.

Одним из важных видов приложения является графическое приложение, т.к. оно позволяет создавать человеко-машинный интерфейс.

При добавлении графического приложения к минимальному рабочему проекту на узле появляется человеко-машинный интерфейс. Для получения доступа к функциям, предоставляемым графическим приложением, необходима авторизация пользователя.

Пользователь – оператор, зарегистрированный в проекте. Каждый пользователь имеет индивидуальный логин и пароль.

Пользователи в проекте могут быть объединены в группы с различными правами доступа.

Список пользователей – полный перечень всех пользователей в проекте.

Минимальный рабочий проект, пусть он даже и включает в себя человеко-машинный интерфейс, (т.е. проект, состоящий из приложения-драйвера, управляющей программы и графического приложения) не требует, как правило, использования столь мощного средства программирования, как SCADA-система, и может быть выполнен на простом алгоритмическом языке программирования. Использование SCADA-систем оправдано при большом количестве узлов, имеющих близкую функциональность и типовое оборудование, т.е. в том случае, когда требуется обработка большого объема информации, поступающего от типовых стандартных объектов. Для такого случая характерно использование большого количества однотипных приложений-драйверов, управляющих программ и графических приложений. Для облегчения работы по созданию однотипных приложений в SCADA-системе «Соната» введено понятие типа приложения.

Тип приложения (шаблон) – это унифицированное описание интерфейса приложения и его конфигурации/схемы/программы. Каждый тип приложения относится к определённому виду приложения.

Экземпляр приложения – приложение, выполняемое на одном из узлов проекта и относящееся к одному (порождённое от одного) из типов приложения; все экземпляры приложений обладают одинаковым интерфейсом приложения и поведением.

Создание приложения в проекте начинается с создания типа приложения (см. Руководство оператора). Каждое приложение, выполняемое на каком-либо узле, является экземпляром приложения, порожденным определенным типом приложения. Соответственно, с учетом того, что каждое работающее приложение является экземпляром определенного типа приложения, нужно уточнить понятие связи приложений.

Связь приложений – прямое указание на обмен значениями сигнала одного экземпляра приложения и сигнала другого экземпляра приложения; типы приложений для этих экземпляров приложений могут не совпадать.

Проект, созданный с использованием всех возможностей SCADA-системы «Соната», содержит следующие элементы:

- типы приложений (шаблоны) различных видов приложений;
- узлы;
- экземпляры приложений различных типов приложений, выполняемые на узлах;
- преобразователи значения сигнала;
- связи между экземплярами приложений;
- глобальные сигналы;
- пользователи.

В процессе создания проекта разработчик создаёт сначала типы приложений (шаблоны). Затем указывает перечень узлов, образующих АСУ ТП. Затем на каждом узле размещает экземпляры приложений, порождённые от тех или иных типов приложений. В конце осуществляет связывание экземпляров приложений между собой связями или глобальными сигналами и создает пользователей.

3.2. Описание программы ProjectManager (Менеджер проекта)

Работа в SCADA-системе "Соната" начинается с создания нового проекта. Для создания проекта разработчик должен запустить программу **ProjectManager**, в которой в дальнейшем будет происходить редактирование проекта (см. Руководство системного программиста). При необходимости программа **ProjectManager** будет самостоятельно вызывать другие программы для выполнения задач по созданию и редактированию различных элементов проекта.

3.2.1. Интерфейс программы ProjectManager

3.2.1.1. Описание главного окна

При запуске **ProjectManager** открывается главное окно этой программы (см. рис. 3.1).

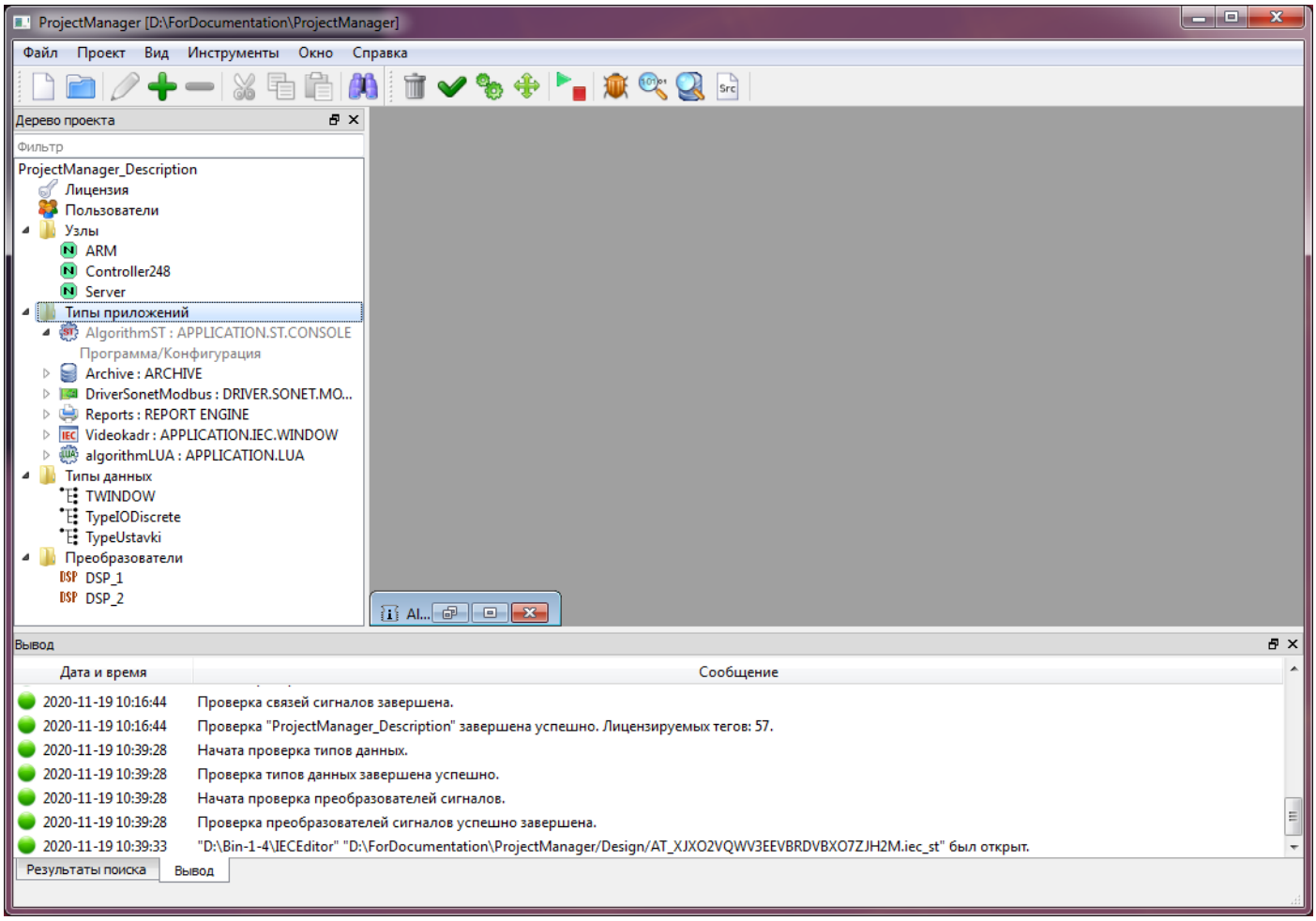


Рисунок 3.1 - Главное окно программы ProjectManager

Главное окно программы содержит следующие элементы:

- главное меню (см. раздел 3.2.1.2);
- панель инструментов (см. раздел 3.2.1.2);
- окно дерева проекта (левая часть главного окна);
- окно редактора интерфейса вида приложения (правая часть главного окна);
- окно списка событий (вкладка Вывод внизу слева);
- окно результатов поиска и замены (вкладка Результаты поиска и замены внизу слева).

В левой части окна расположено дерево проекта, отображающее всё его содержимое:

- строка фильтр - помогает отфильтровать элементы в дереве проекта (при фильтрации можно использовать стандартные символы подстановки, такие как *, ? и др.);
- таблицу глобальных сигналов, настроек проекта и связей приложений (см. раздел 3.2.2.8);
- пункт для работы с лицензией (см. раздел 3.2.2.9);
- список пользователей проекта (см. раздел 3.2.2.10);
- список узлов (см. раздел 3.2.2.5);
- список типов приложений (см. раздел 3.2.2.4);
- типы данных (пользовательские типы данных, см. раздел 3.2.2.7);
- список преобразователей сигналов (см. раздел 3.2.2.6).

Внизу главного окна **ProjectManager** расположены сообщения о работе системы (вкладка Вывод) и сообщения результатов поиска и замены (вкладка Результаты поиска и замены). На рис. 3.1 активно окно списка Вывод (событий), а окно результатов поиска и замены «спрятано».

Над одним и тем же проектом одновременно могут работать несколько операторов, для этого каждый элемент дерева проекта можно редактировать независимо от остальных элементов.

В момент редактирования элемент блокируется в дереве проекта (для исключения повторного открытия) и окрашивается в серый цвет.

3.2.1.2. Главное меню

В верхней части окна располагается главное меню, в котором собраны все доступные команды для редактирования проекта и настройки **ProjectManager**.

Главное меню делится на следующие разделы:

Файл – группа команд работы с файлами проекта (см. раздел 3.2.1.2.1);

Проект – группа команд редактирования и отладки проекта (см. раздел 3.2.1.2.2);

Вид – группа команд управления видом главного окна и сообщениями программы **ProjectManager** (см. раздел 3.2.1.2.3);

Инструменты – группа дополнительных команд (см. раздел 3.2.1.2.4);

Окно – группа команд, управляющих расположением открытых окон редактора интерфейса вида приложения (см. раздел 3.2.1.2.5);

Справка – группа команд доступа к справочной информации (см. раздел 3.2.1.2.6).

3.2.1.2.1. Файл – группа команд работы с файлами проекта

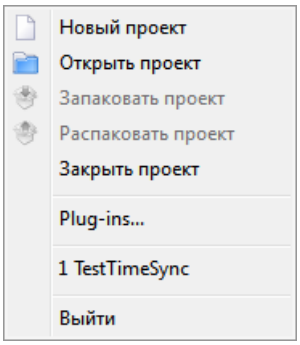


Рисунок 3.2 - Раскрытое меню Файл

Раздел Файл (группа команд работы с файлами проекта) содержит следующие команды:

Новый проект – создать новый проект (см. раздел 3.2.2.1);

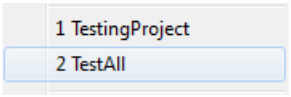
Открыть проект – открыть существующий проект (см. раздел 3.2.2.2);

Запаковать проект – запаковать все файлы проекта в один архивный файл;

Распаковать проект – распаковать архивный файл для работы с проектом;

Закрыть проект – закончить работу с проектом, не закрывая программу **ProjectManager** (см. раздел 3.2.2.3);

Plug-ins... - это механизм расширения функциональности ProjectManager, при помощи которого пользователи смогут писать свои дополнения для автоматизации рутинных операций, например, создания нескольких узлов или типов приложений и т.п. (подробно см. раздел 3.2.1.2.1.1);



- в этом разделе указываются проекты, с которыми недавно проводились работы;

– закончить работу с программой **ProjectManager** .

3.2.1.2.1.1. Описание механизма расширения функциональности Менеджера проектов (Plug-ins)

Менеджер проектов (ProjectManager) при старте сканирует свою папку на предмет наличия в этой папке файлов с расширением *.pm.lua. Если есть такие файлы, то он загружает их, как программы на Lua. Далее функционал, написанный в данных файлах можно выполнить с помощью меню **Файл - Plug-ins...**

В каждом файле с расширением *.pm.lua должна быть определенна таблица, которая должна возвращаться при запуске данного файла.

```
local M = {
name = "TestName",
run = function()
...
end
}
return M
```

В таблице должно быть обязательным поле name, которое служит для указания имени plug-in, которое будет отображаться в меню ProjetcManager.

Поле run - при выборе в меню plug-ins в Менеджере проектов будет запускаться функция run( ).

Для работы в данном механизме доступны все стандартные функции LUA (см. SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33).

Так же для данного механизма доступны дополнительные функции:

Имя функции	Описание
Dialog.critical([msg], [title])	Выводит окно с сообщением msg и заголовком окна title с критической иконкой
Dialog.warning([msg], [title])	Выводит окно с сообщением msg и заголовком окна title с предупреждающей иконкой
Dialog.information([msg], [title])	Выводит окно с сообщением msg и заголовком окна title с информационной иконкой
local b = Dialog.yesNo([msg], [title])	Выводит окно с сообщением msg и заголовком окна title и двумя кнопками Yes и No. Возвращает bool - результат нажатия на кнопки
local s(тип string|nil) = Dialog.getString([label], [title], [defaulValue])	Выводит окно с подписью label и заголовком окна title, и полем ввода строки. В defaulValue можно указать значение по умолчанию. Возвращает строку, если пользователь ввёл её или nil, если пользователь отказался от ввода

local i(тип integer|nil) = Dialog.getInteger([label], [title], [defaulValue=0], [min=-2147483647], [max=2147483647])	Выводит окно с подписью label и заголовком окна title и полем ввода целого числа. В defaulValue можно указать значение по умолчанию. Возвращает целое число, если пользователь ввёл его или nil, если пользователь отказался от ввода;
local n(number|nil) = Dialog.getNumber([label], [title], [defaulValue=0], [min=-2147483647], [max=2147483647])	Выводит окно с подписью label и заголовком окна title и полем ввода вещественного числа. В defaulValue можно указать значение по умолчанию. Возвращает вещественное число, если пользователь ввёл его или nil, если пользователь отказался от ввода
local i = Dialog.selectItem([label], [title], {1="Item1", 2="Item2"..})	Выводит окно с подписью label и заголовком окна title и выпадающим списком. Возвращает целое число, соответствующее выбранной позиции или nil, если пользователь отказался от ввода
local {1,3,5}|nil = Dialog.selectItems([label], [title], {1="Item1", 2="Item2"..})	Выводит окно с подписью label и заголовком окна title и списком, в котором можно выбирать множество позиций. Возвращает таблицу с номерами выбранных позиций
local s = Project.getDesignPath()	Возвращает путь к Design папке проекта;
local s = Project.getUUID([base32])	Возвращает UUID в виде HEX строки или base32, если данный аргумент = true
Project.createAppType(name, [comment], appKind)	Добавляет в проект новый тип приложения с именем name, комментарием comment и видом приложения appKind. Возвращает имя файла, в который произошло сохранение нового типа приложений и возвращает UUID типа приложения, по которому к этому типу можно обращаться в других функциях
Пример: local appTypeFileName, appTypeUuid = Project.createAppType("TControlApp", "|", "APPLICATION.LUA");	
Project.createNode(nodeName, nodeComment, apps)	Добавляет в проект новый узел с именем nodeName, комментарием nodeComment и списком приложений узла apps. apps - это массив(таблица), начинающийся с нулевой ячейки. Нулевая ячейка - описатель приложения для Loader. Следующие ячейки - описатели приложений узла. В каждой ячейке массива apps должна лежать таблица с двумя полями: { name = "AppName", appType=appTypeUuid}. Возвращает имя файла, в который была сохранена информация об узле и его UUID
Пример:	

```
local appTypeFileName0, appTypeUuid0 = Project.createAppType("TControlApp", "ControlApp", "APPLICATION.LUA");
local appTypeFileName1, appTypeUuid1 = Project.createAppType("TAppType", "TAppTypeComment", "DRIVER.MODBUS");
local apps =
{
[0]={name="Loader", appType=appTypeUuid0},
[1]={name="App", appType=appTypeUuid1 }
};
local nodeFilePath, uuid = Project.createNode("NodeName", "NodeComment", apps);
```

local filePath, uuid = Project.createAppType(name, comment, appKind)	Добавляет в проект новый тип приложения с именем name, комментарием comment и видом приложения appKind. Возвращает имя файла (с полным путём), который содержит конфигурацию нового типа приложения и возвращает uuid - уникальный идентификатор типа приложения
local filePath, uuid = Project.createNode(name, [comment], apps)	Добавляет в проект новый узел с именем name, комментарием comment, и списком приложений apps, где apps - массив структур { name="name", appType="AppTypeUuid"}, [0] - ячейка для Loader. Возвращает имя файла (с полным путём), который содержит конфигурацию нового узла и возвращает uuid - уникальный идентификатор созданного узла
local {name, uuid, filePath} = Project.getNodeList()	Возвращает список узлов проекта. Возвращает массив структур с полями name, uuid, filePath
local {name, uuid, appType, appKind} = Project.getNodeAppList(name|uuid)	Возвращает в виде массива список приложений узла проекта. Каждая ячейка массива содержит поля: name - имя экземпляра приложения, uuid - уникальный идентификатор экземпляра, appType - тип приложения (uuid), appKind - вид приложения (DRIVER.MODBUS, APPLICATION.LUA и т.п.) Внимание! Если в конфигурации узла есть ошибки, то есть не до конца настроен перечень приложений, например не указан тип, то поля структуры appType или appKind могут быть nil
local b = Project.pairNode(name1|uuid1, name2|uuid2)	Связывает два узла для резервирования, где name1 и name2 - имена узлов, uuid1 и uuid2 - уникальные идентификаторы узлов. Если не указать второй аргумент, то снимает пару для указанного первого узла

3.2.1.2.2. Проект – группа команд редактирования и подготовки проекта

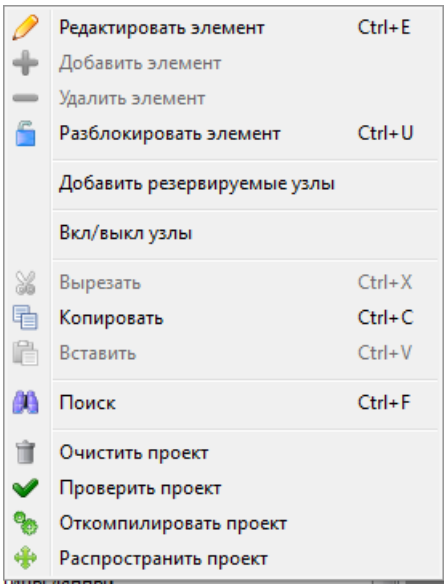
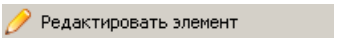


Рисунок 3.3 - Раскрытое меню Проект

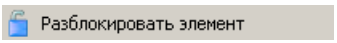
Раздел Проект (группа команд редактирования и отладки проекта) содержит следующие команды:



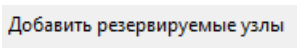
– редактировать выбранный элемент проекта;

– добавить новый элемент проекта;

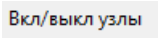
– удалить выбранный элемент проекта;



– разблокировать элемент проекта;



- добавляет в дерево проекта необходимые для настройки резервирования элементы (два узла: Node и NodeR, имеющие настройки для резервирования по умолчанию, и управляющее приложение TControlApp:Application.LUA);

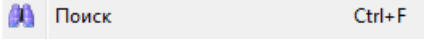


- включает/выключает выбранные узлы в проекте;

– вырезать выделенный элемент проекта и сохранить его в буфере памяти;

– копировать выделенный элемент проекта в буфер памяти;

– вставить вырезанный или скопированный элемент проекта из буфера памяти;



- меню поиска по проекту. По нажатию на данный пункт меню откроется окно диалога поиска (см. рис. 3.4).

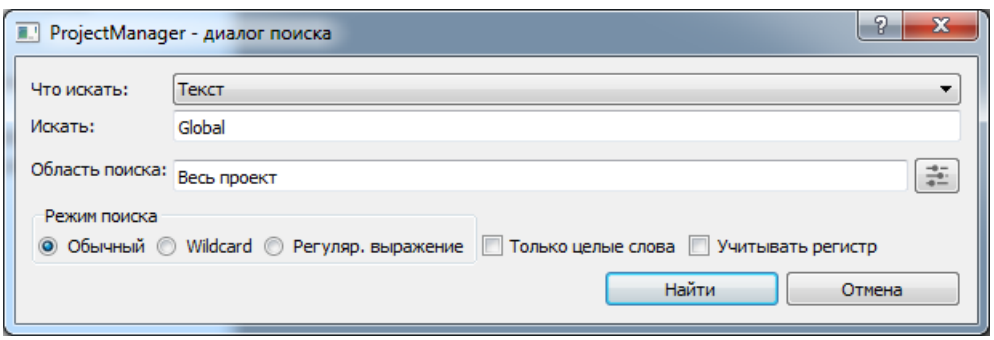


Рисунок 3.4 - Окно диалога поиска

Описание окна диалога поиска:

- **Что искать:** - в данном поле можно выбрать два варианта: Текст или Сигнал. При выборе варианта "Текст", будет происходить поиск текста по всему содержимому проекта. Соответственно при выборе варианта "Сигнал" будет происходить поиск сигнала. В конфигурациях приложений, которые не имеют структуры (Например: код на языке Lua), будет искаться текст даже при выборе варианта Сигнал.

- **Искать:** - в данном поле вводится строка, которая является происто текстом или именем сигналаа, в зависимости от выбора значения в поле "Что искать". При различных режимах поиска, которые описаны ниже, здесь могут применяться специальные символы и регулярные выражения;

- **Область поиска:** - в данном поле выполняется настройка области, в которой будет производиться поиск. Справа от данного поля находится кнопка  (Редактировать область поиска), по нажатию на которую раскрывается меню для редактирования области поиска (см. рис. 3.5).

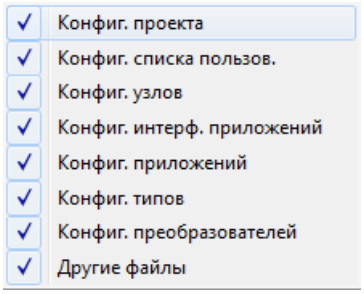


Рисунок 3.5 - Меню редактирования области поиска

- **Регуляр. выражение** - в настройках поиска по умолчанию установлен шаблон поиска Wildcard (Расширенный поиск с использованием специальных символов: * - любая последовательность символов, ? - любой символ в одном экземпляре, [a,b] или [ab] - любой символ из перечисленных в скобках). Если поставить данную галочку, то данные будут искаться по регулярному выражению, которое пользователь укажет в поле "Искать".

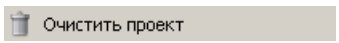
- **Только целые слова** - если поставить данную галочку, то будет поиск только по словам, которые отделены пробелом или символом "не слова" по краям, т.е. поиск по отдельным словам или фразам идентичные искомой строке. Иначе поиск будет также и по частям других слов;

- **Учитывать регистр** - если поставить данную галочку, то поиск будет регистрозависимым. Иначе, поиск будет регистронезависимым;

- **кнопка Найти** - по нажатию на данную кнопку будет выполнен поиск. Результаты поиска будут отображены в нижнем окне Менеджера проекта во вкладке Результаты поиска;

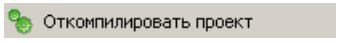
- **кнопка Отмена** - по нажатию на данную кнопку будет закрыто окно диалога поиска и поиск не будет выполнен.

Описание окна диалога поиска:

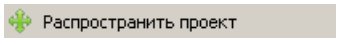


– удалить откомпилированный проект и все временные файлы;

– провести быструю проверку проекта на согласованность интерфейсов приложений ;



– откомпилировать проект;



– запустить программу **Distributer**, которая осуществит копирование проекта на узлы (см. раздел 3.2.2.13).

3.2.1.2.3. Вид – группа команд управления видом главного окна и сообщениями программы ProjectManager

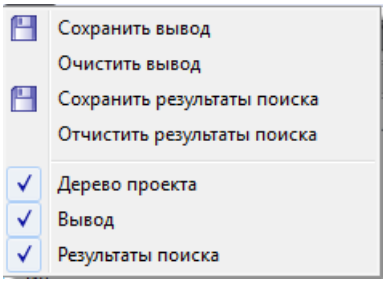


Рисунок 3.6 - Раскрытое меню Вид

Раздел Вид (группа команд управления видом главного окна и сообщениями программы **ProjectManager**) содержит следующие команды:

Сохранить вывод – открыть диалоговое окно для сохранения лога сообщений в отдельном файле;

Очистить вывод – удалить лог сообщений в окне Вывод;

Сохранить результаты поиска – открыть диалоговое окно для сохранения результатов поиска в отдельном файле;

Отчистить результаты поиска – удалить лог сообщений в окне Результаты поиска;

Дерево проекта – спрятать/показать окно дерева проекта (по умолчанию это окно открыто);

Вывод – спрятать/показать лог сообщений о работе (по умолчанию эти логи активны);

Результаты поиска - спрятать/показать лог результатов поиска (по умолчанию эти логи активны).

3.2.1.2.4. Инструменты – группа команд, помогающих выполнять дополнительные функции

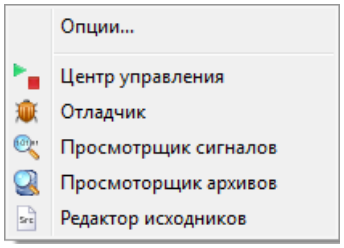
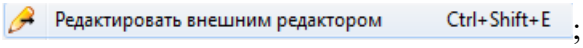


Рисунок 3.7 - Раскрытое меню Инструменты

Раздел Инструменты содержит следующие команды (см. рис. 3.7):

Опции... – по нажатию на данный пункт меню откроется окно для настройки внешних редакторов (см. рис. 3.8). В данном окне можно настроить каким внешним редактором будут открываться форматы файлов .dsp (файлы преобразователей сигналов), .int (файлы интерфейсов

приложений), .nod (файлы узлов проекта), .prj (файл проекта с глобальной таблицей сигналов и др. настройкми) и др. форматы. В дальнейшем, к примеру, для открытия внешним редактором файла преобразователей сигналов можно будет нажать на преобразователе правой клавишей мыши и выбрать из контекстного меню пункт 

;

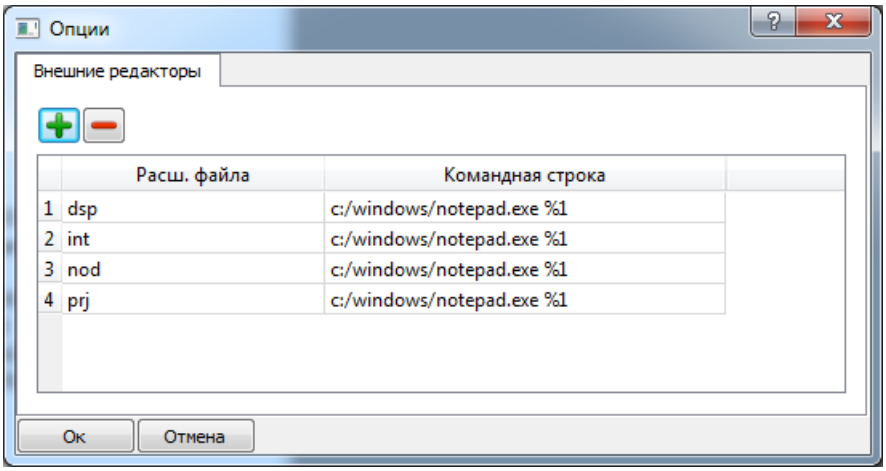
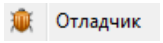
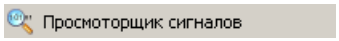


Рисунок 3.8 - Окно настроек внешних редакторов

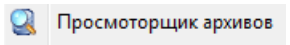
- по нажатию на данный пункт меню откроется окно Центр управления (см. раздел 3.4.4);



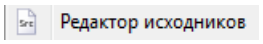
- по нажатию на данный пункт меню откроется окно отладчика кода (см. раздел 3.4.2);



- по нажатию на данный пункт меню откроется окно Просмотра сигналов (см. раздел 3.4.5);



- по нажатию на данный пункт меню откроется окно Просмотра архивов сигналов и событий (см. раздел 3.4.6);



- по нажатию на данный пункт меню откроется окно Редактора исходников (см.).

3.2.1.2.5. Окно – группа команд, управляющих расположением открытых окон редактора интерфейса вида приложения

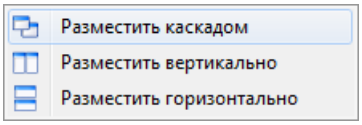
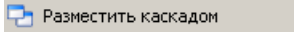


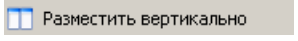
Рисунок 3.9 - Раскрытое меню Окно

Раздел 

(группа команд, управляющих расположением открытых окон редактора интерфейса вида приложения) содержит следующие команды:



– располагает окна редактора каскадом;



– располагает окна редактора вертикально;

– располагает окна редактора горизонтально.

3.2.1.2.6. Справка – группа команд доступа к справочной информации

Раздел Справка (группа команд доступа к справочной информации) содержит одну команду:

О программе – открывает информационное окно, содержащее сведения о программе **ProjectManager**.

3.2.1.3. Панель инструментов

Панель инструментов расположена под главным меню – на этой панели расположены значки (кнопки), которые вызывают наиболее часто используемые команды.

С помощью нажатия манипулятором "мышь" на значки (кнопки), вынесенные на панель инструментов, выполняются следующие команды:

 («Новый проект») – создать новый проект (из раздела **Файл**, см. раздел 3.2.2.1);

 («Открыть проект») – открыть существующий проект (из раздела **Файл**, см. раздел 3.2.2.2);

 («Редактировать элемент») – редактировать выбранный элемент проекта (из раздела **Проект**);

 («Добавить элемент») – добавить новый элемент проекта (из раздела **Проект**);

 («Удалить элемент») – удалить выбранный элемент проекта (из раздела **Проект**);

 («Вырезать») – вырезать выделенный элемент проекта и сохранить его в буфере памяти (из раздела **Проект**);

 («Копировать») – копировать выделенный элемент проекта в буфер памяти (из раздела **Проект**);

 («Вставить») – вставить вырезанный или скопированный элемент проекта из буфера памяти (из раздела **Проект**);

 ("Поиск") - по нажатию на данную иконку откроется окно поиска (см. рис. 3.4);

 («Очистить проект») – удалить откомпилированный проект и все временные файлы (из раздела **Проект**);

 («Проверить проект») – провести быструю проверку проекта на согласованность интерфейсов приложений (из раздела **Проект**);

 («Откомпилировать проект») – откомпилировать проект (из раздела **Проект**);

 («Распространить») – запустить программу **Distributer**, которая осуществит копирование проекта на **узлы** (из раздела **Проект**, описание смотри в раздел 3.2.2.13);

 («Центр управления») – запустить программу управление проектом – **ControlCenter** (из раздела **Проект**, описание смотри в раздел 3.4.4);

 («Debugger») – запустить программу **Debugger** для отладки приложений проекта (из раздела **Проект**, смотри описание в раздел 3.4.2);

 («IECDebugger») – запустить программу **Debugger** для отладки приложений проекта (из раздела **Проект**, описание смотри в раздел 3.4.3);

 («Просмотрщик сигналов») – запустить программу **SignalViewer** для просмотра сигналов приложений (из раздела **Проект**, описание смотри в раздел 3.4.5);

 ("Просмотрщик архивов") - запустить программу **ArchiveViewer** для просмотра архивов сигналов и событий, описание смотри в раздел 3.4.6;

 ("Редактор исходников") - запустить программу **SourceEditor** для редактирования исходных кодов различных программ, описание смотри в .

3.2.2. Работа в программе ProjectManager

3.2.2.1. Создание нового проекта

Для создания нового проекта нужно в главном окне программы **ProjectManager** нажать кнопку  («Новый проект») на главной панели инструментов или в разделе  главного меню выбрать команду  . Сначала будет открыт диалог для ввода имени проекта (см. рис. 3.10), затем будет открыт диалог для выбора расположения проекта на диске (см. рис. 3.11).

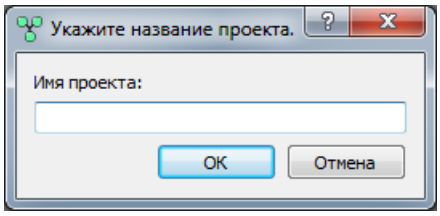


Рисунок 3.10 - Диалог ввода имени проекта

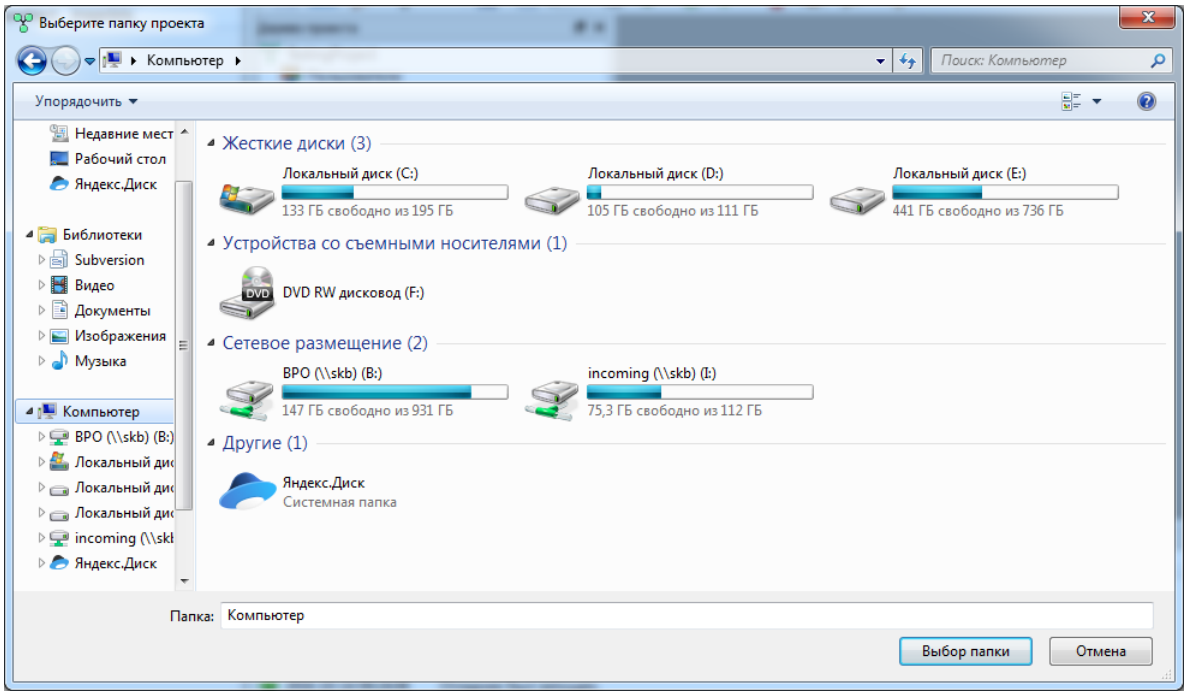


Рисунок 3.11 - Диалог выбора расположения проекта на диске

Необходимо задать имя проекта и выбрать расположение проекта на диске. Рекомендуется создать новую папку, нажав правую кнопку мыши и выбрав в появившемся меню **Создать-Папку** (стандартное всплывающее меню Windows), и дать папке имя, которое можно легко ассоциировать с названием проекта. Для подтверждения совершенных действий следует нажать кнопку Выбор папки. **ProjectManager** в указанной папке создаст вложенные папки **Design** и **Runtime**. В папке **Design** будут размещаться файлы проекта, необходимые на этапе разработки. В папку **Runtime** будут помещены файлы, необходимые для выполнения проекта на узлах (данные файлы создаются автоматически в процессе компиляции проекта).

После создания заготовки проекта на диске он будет автоматически открыт программой **ProjectManager**.

3.2.2.2. Открытие существующего проекта

Для открытия уже существующего проекта нужно нажать кнопку  («Открыть проект») на главной панели инструментов или в разделе Файл главного меню выбрать команду Открыть проект. Будет открыт диалог для выбора папки с проектом (см. рис. 3.12).

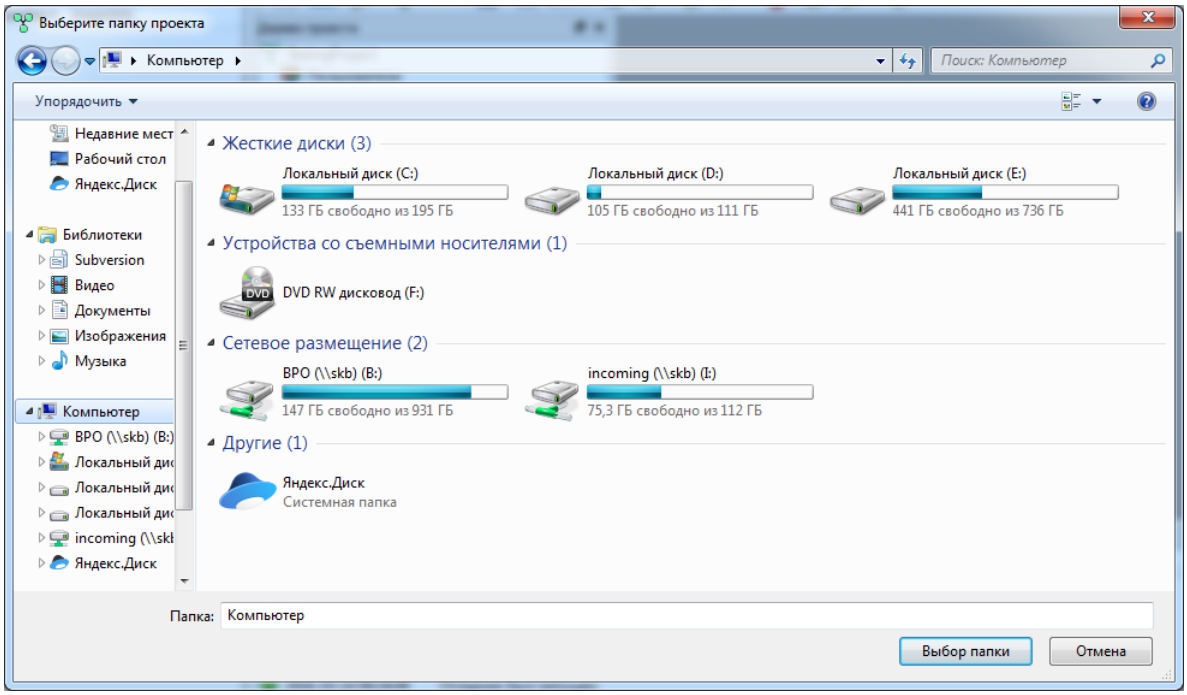


Рисунок 3.12 - Диалог выбора расположения папки проекта

В открывшемся диалоге выберите папку проекта, после чего нажмите кнопку Выбор папки.

Если проект открывался для редактирования недавно, то он сохранился в списке меню Файл. В таком случае, достаточно «кликнуть» «мышью» на соответствующее название проекта в списке (см. рис. 3.13).

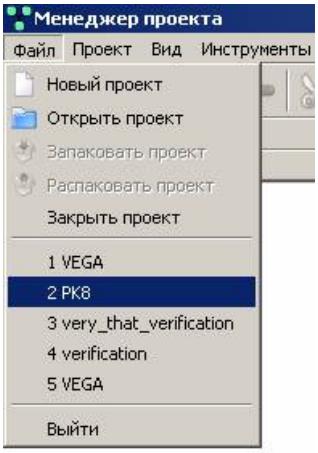


Рисунок 3.13 - Выбор проекта для открытия через список меню Файл

3.2.2.3. Закрытие проекта

Для закрытия проекта, например, с целью перехода к редактированию нового проекта, нужно выбрать строку Закрыть проект в меню Файл (см. рис. 3.14).

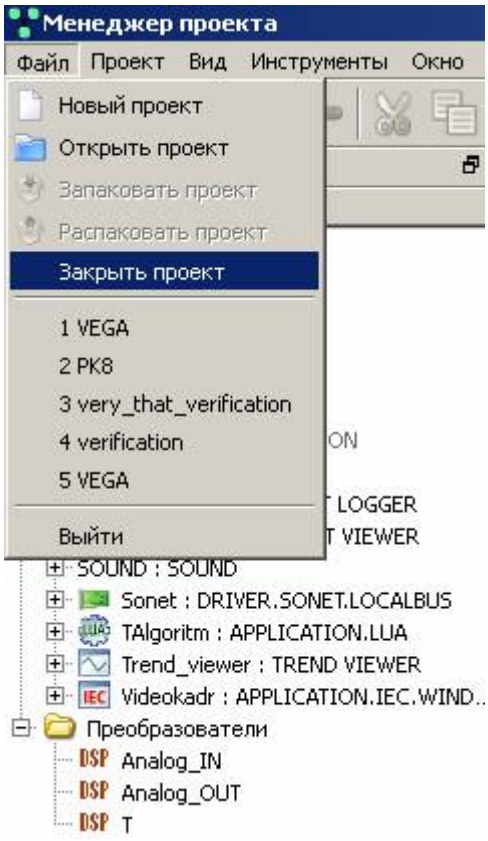


Рисунок 3.14 - Закрытие проекта с помощью меню Файл

3.2.2.4. Редактирование типов приложений

Как правило, создание проекта начинается с создания типов приложений (шаблонов), которые будут использованы в данной АСУ ТП. Типы приложений располагаются в дереве проекта в папке **Типы приложений**.

Любое приложение состоит из двух частей:

- интерфейса приложения;
- самой программы или конфигурационного файла.

Интерфейс всех видов приложений редактируется единообразно, а для редактирования программы приложения или конфигурационного файла приложения открывается специализированный редактор. В дереве проекта тип приложения отображается в виде двух позиций: верхняя позиция – интерфейс приложения, вложенная позиция – программа приложения или его конфигурационный файл. Добавляются и удаляются эти две позиции одновременно, но редактируются раздельно.

3.2.2.4.1. Создание нового типа приложения

Для создания нового типа приложения необходимо в дереве проекта главного окна программы **ProjectManager** выделить папку Типы приложений (см. рис. 3.15), а затем либо нажать кнопку  («Добавить элемент») на главной панели инструментов (см. рис. 3.16), либо вызвать контекстное меню с помощью правой кнопки «мыши» и выбрать в нем команду Добавить элемент (см.

рис. 3.17), либо выбрать в главном меню в разделе Проект команду Добавить элемент (см. рис. 3.18) – в перечень типов приложений в дереве проекта будет добавлена новая позиция.

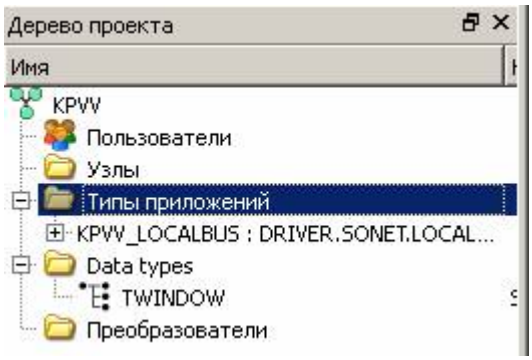


Рисунок 3.15 - Папка Типы приложений в дереве проекта

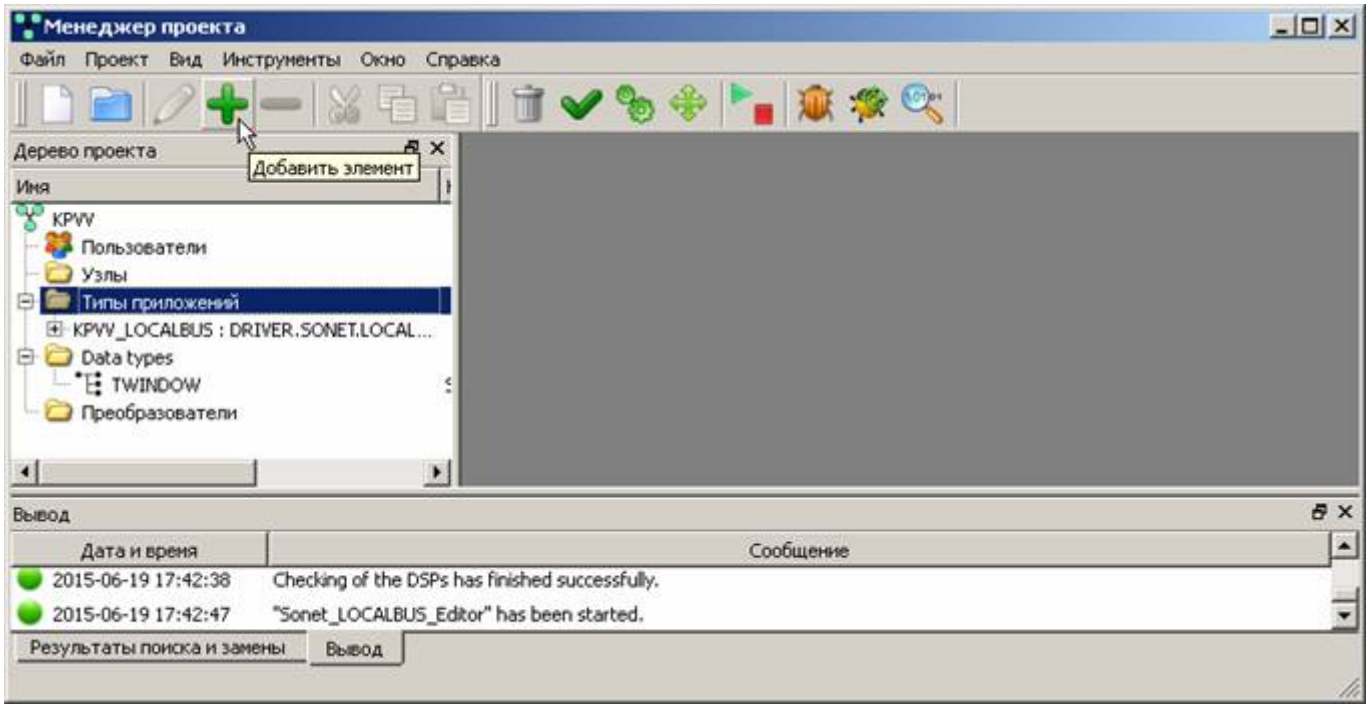


Рисунок 3.16 - Добавление нового типа приложения с помощью кнопки «Добавить элемент»

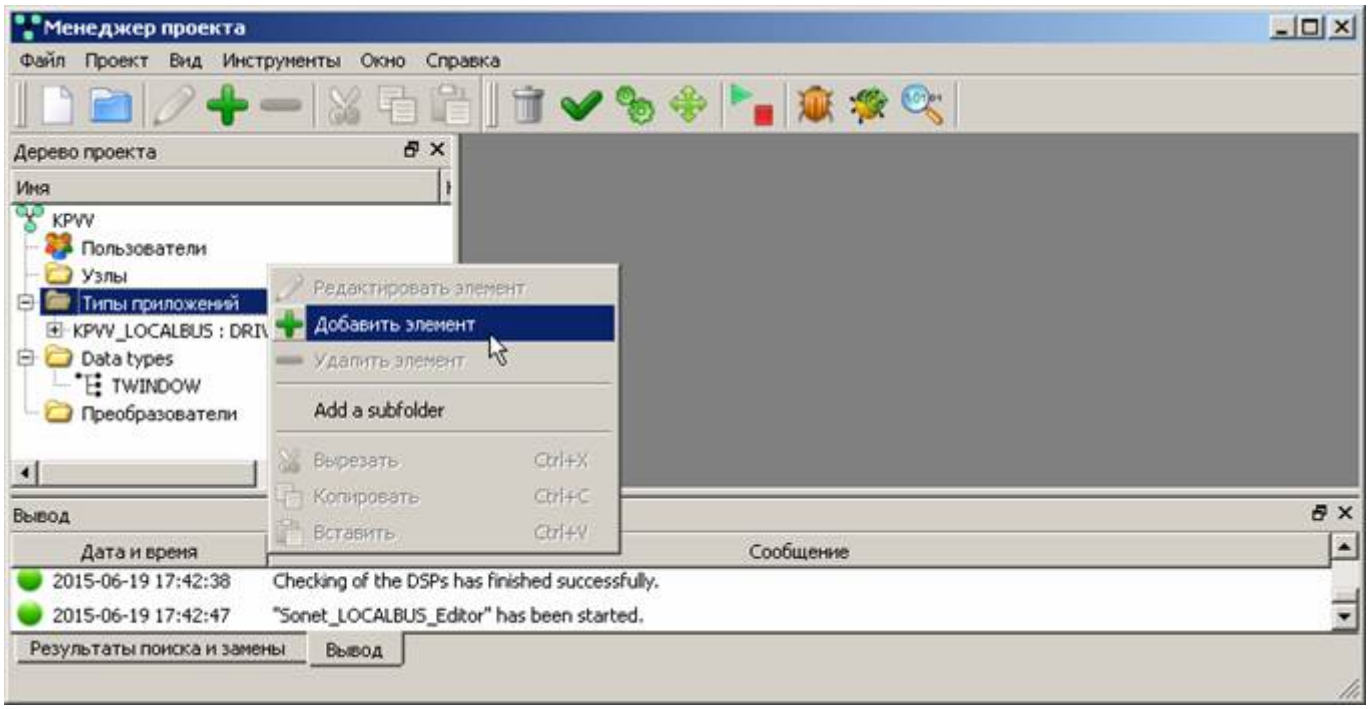


Рисунок 3.17 - Добавление нового типа приложения с помощью контекстного меню

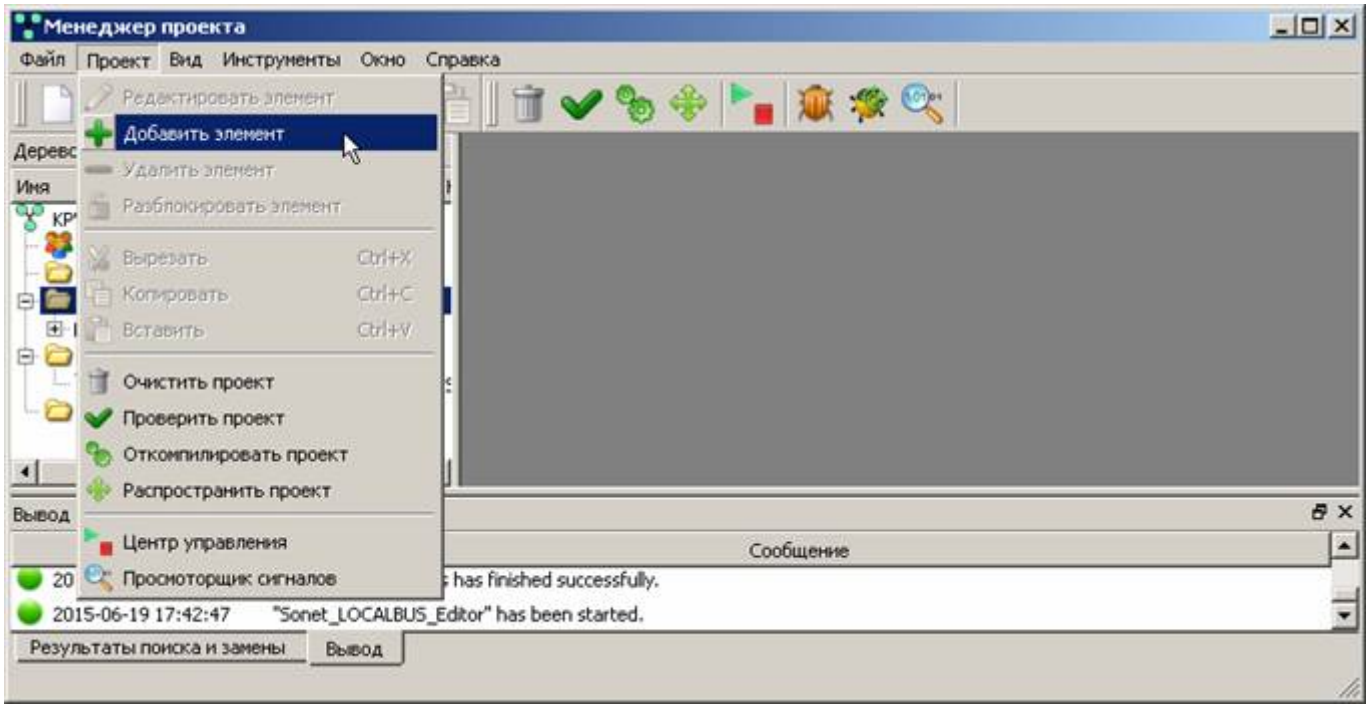


Рисунок 3.18 - Добавление нового типа приложения с помощью команды главного меню

Имена новым типам приложений присваиваются автоматически по шаблону **AppType_N**, где **N** – уникальное число. Также новому типу приложения задается вид **APPLICATION** (см. рис. 3.19). В дальнейшем имя типа и вид типа можно легко заменить на более подходящие (см. раздел 3.2.2.4.3).

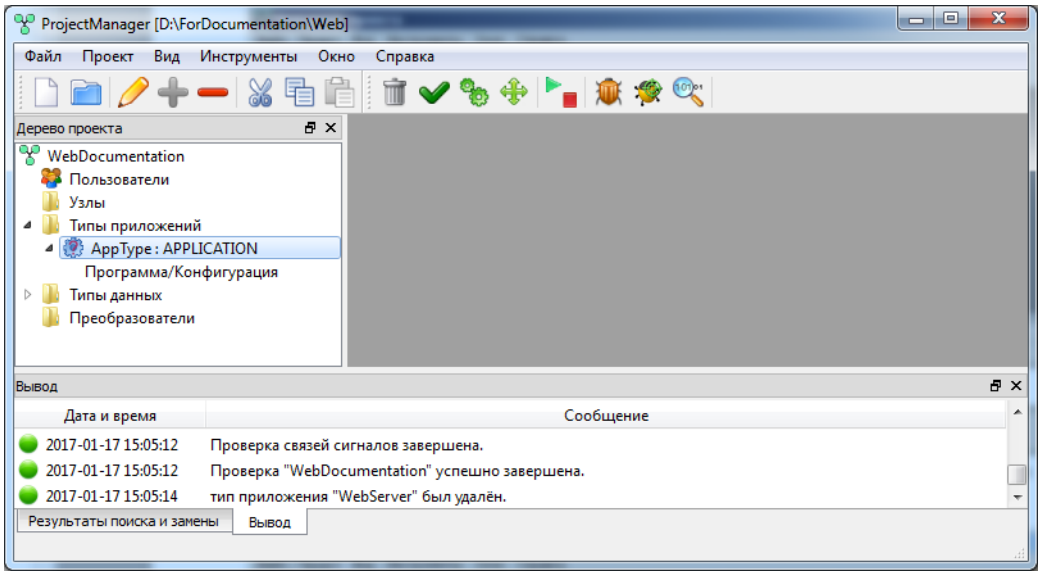


Рисунок 3.19 - Как выглядит по умолчанию новый тип приложения

3.2.2.4.2. Удаление типа приложения (шаблона)

Для удаления типа приложения (шаблона) нужно в дереве проекта главного окна **ProjectManager** выделить удаляемую позицию, а затем либо нажать кнопку  («Удалить элемент») на главной панели инструментов (см. рис. 3.20), либо вызвать контекстное меню с помощью правой кнопки «мыши», а затем выбрать команду 

(см. рис. 3.21), либо выбрать в главном меню в разделе 

команду 

(см. рис. 3.22) – будет выведен диалог подтверждения удаления типа приложения (см. рис. 3.23). При положительном ответе тип приложения будет удалён. Перед удалением рекомендуется убедиться, что данный тип приложения нигде в проекте не используется.

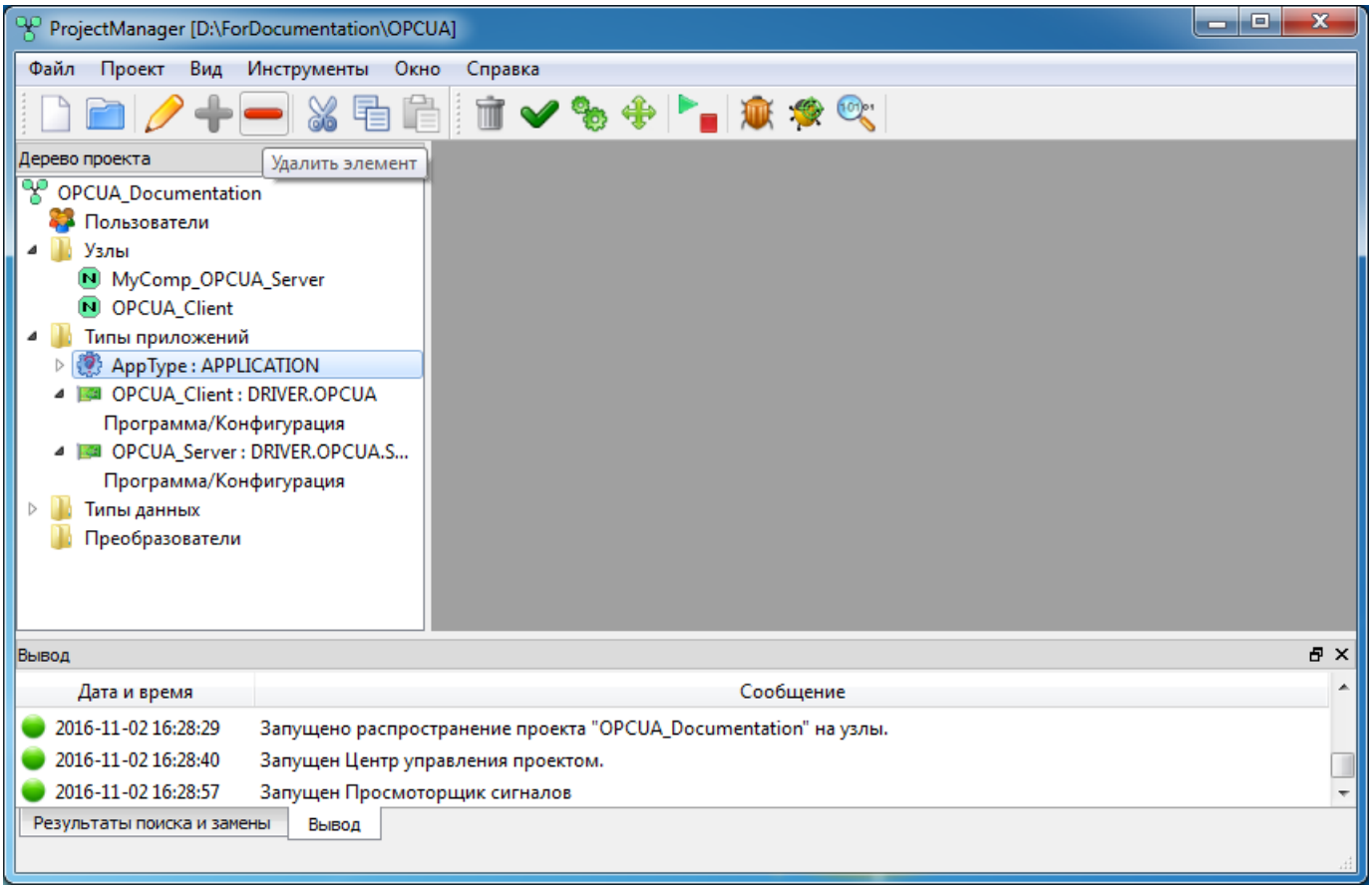


Рисунок 3.20 - Удаление типа приложения с помощью кнопки "Удалить элемент" на панели инструментов

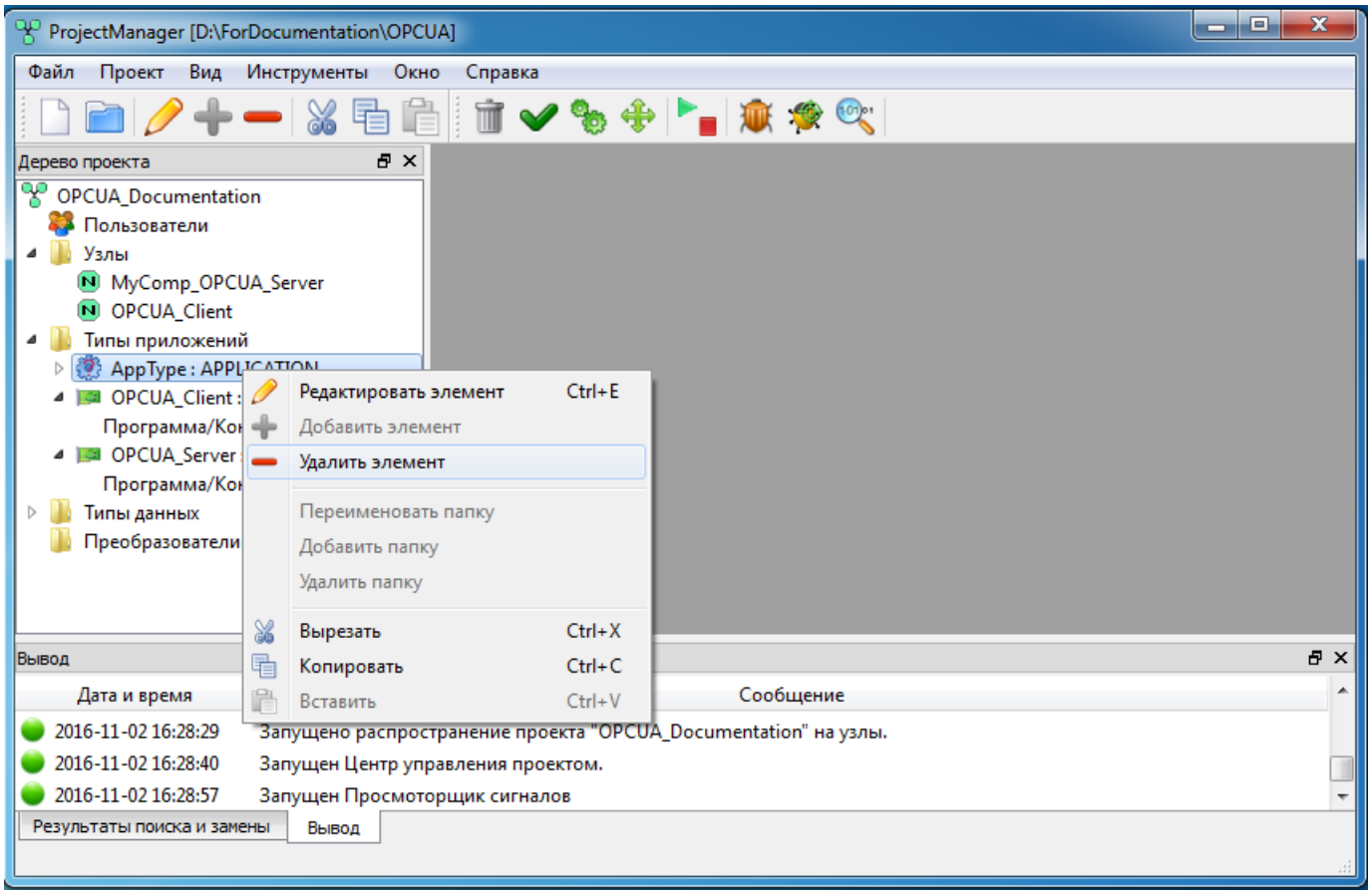


Рисунок 3.21 - Удаление типа приложения с помощью контекстного меню

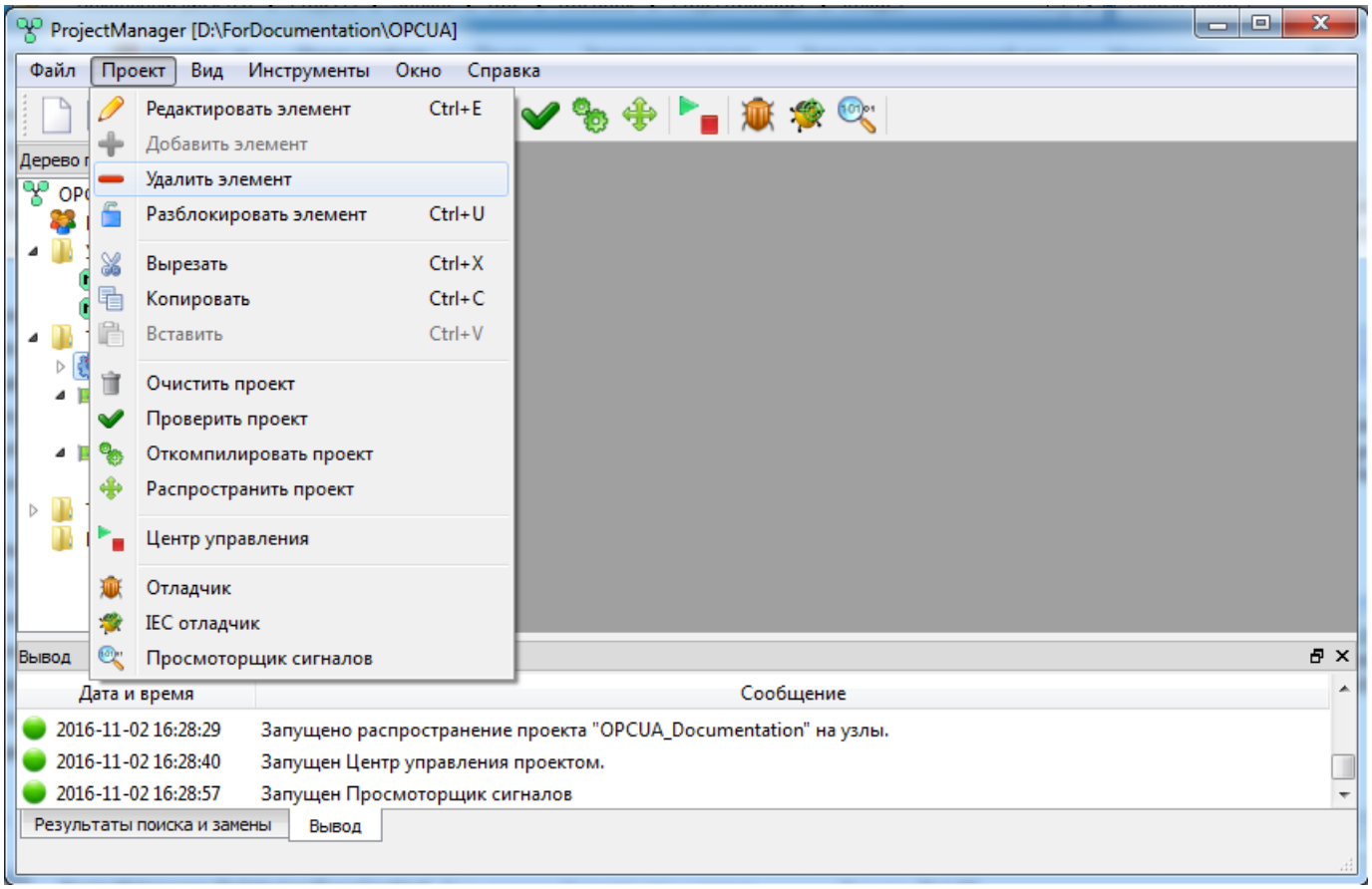


Рисунок 3.22 - Удаление типа приложения с помощью команды главного меню "Удалить элемент"

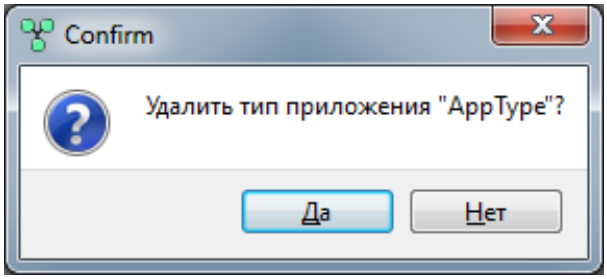


Рисунок 3.23 - Диалог подтверждения удаления типа приложения

3.2.2.4.3. Редактирование интерфейса типа приложения (шаблона)

Для редактирования интерфейса типа приложения(шаблона) нужно в дереве проекта главного окна **ProjectManager** выделить позицию, соответствующую интерфейсу типа приложения (т.е., содержащую имя типа и вид приложения, например, строку AppType : APPLICATION) , а затем либо нажать кнопку («Редактировать элемент») на главной панели инструментов (см. рис. 3.24), либо вызвать контекстное меню с помощью правой кнопки «мыши», а затем выбрать команду Редактировать элемент (см. рис. 3.25), либо выбрать в главном меню в разделе Проект команду Редактировать элемент (см. рис. 3.26) – откроется диалог редактирования интерфейса типа приложения(см. рис. 3.27).

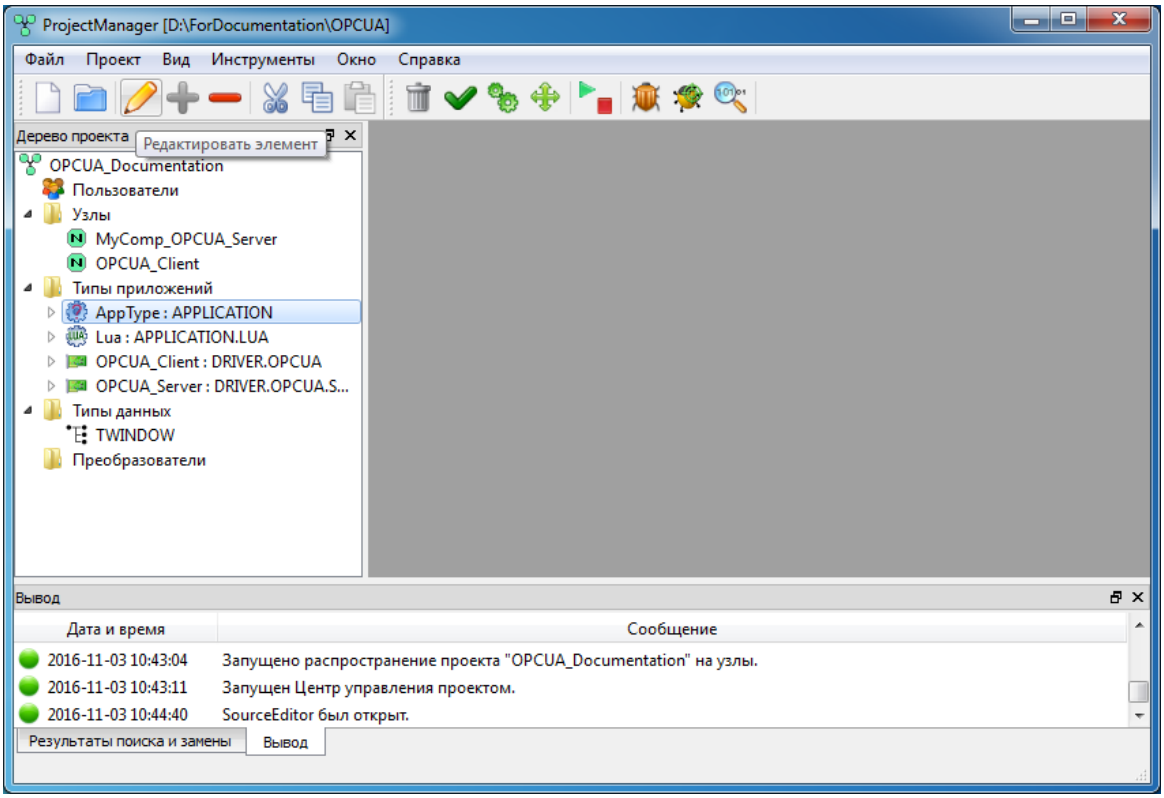


Рисунок 3.24 - Редактирование интерфейса приложения с помощью кнопки "Редактировать элемент" на панели инструментов

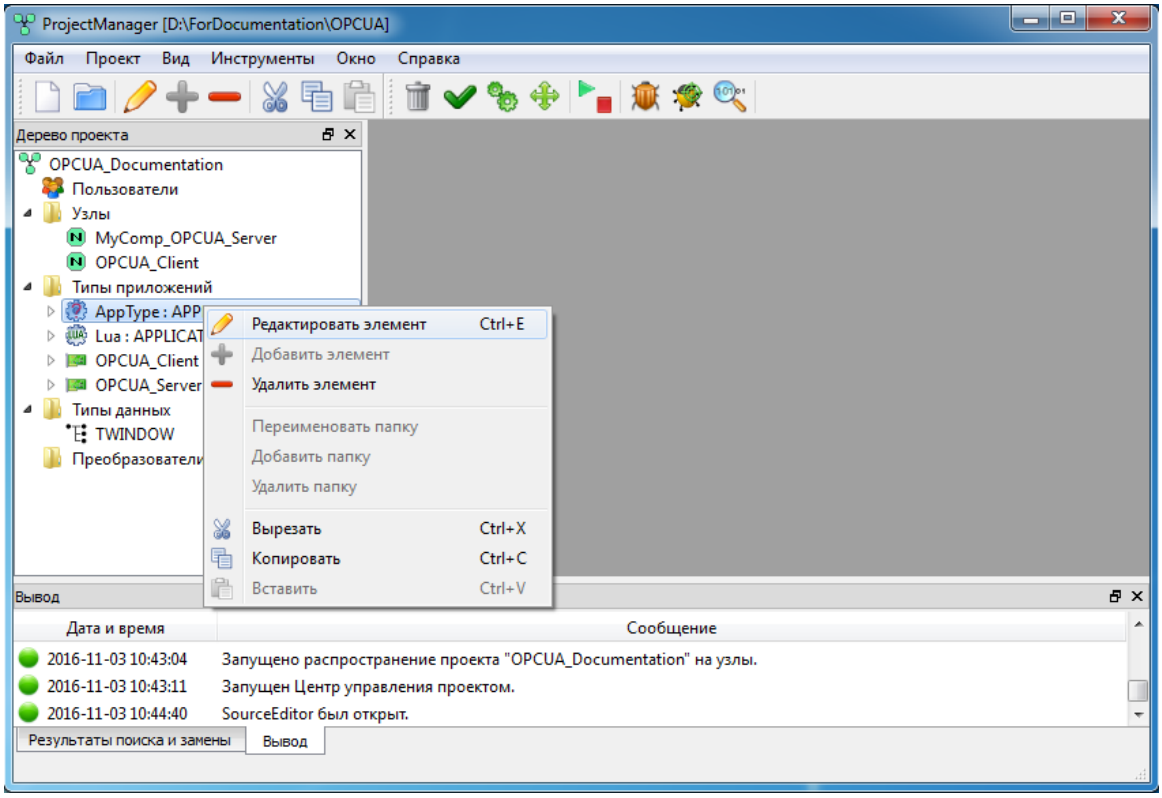


Рисунок 3.25 - Редактирование интерфейса приложения с помощью кнопки "Редактировать элемент" контекстного меню

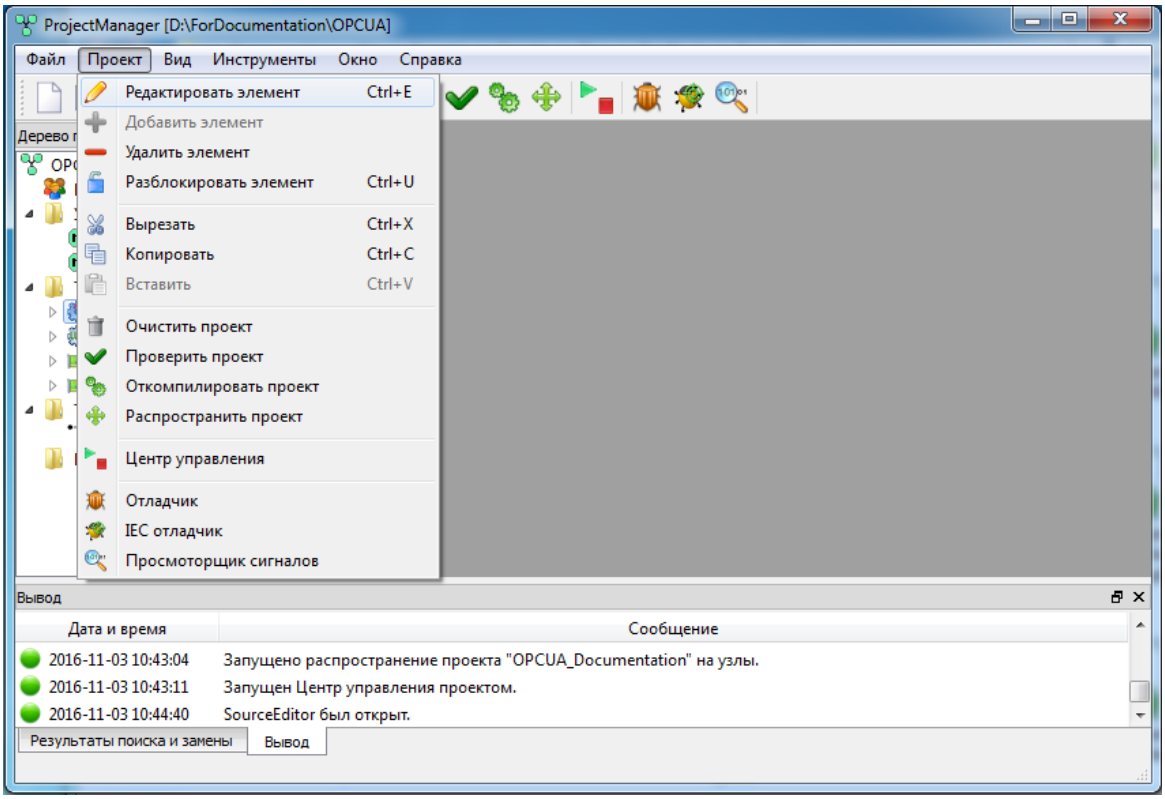


Рисунок 3.26 - Редактирование интерфейса приложения с помощью команды "Редактировать элемент" главного меню

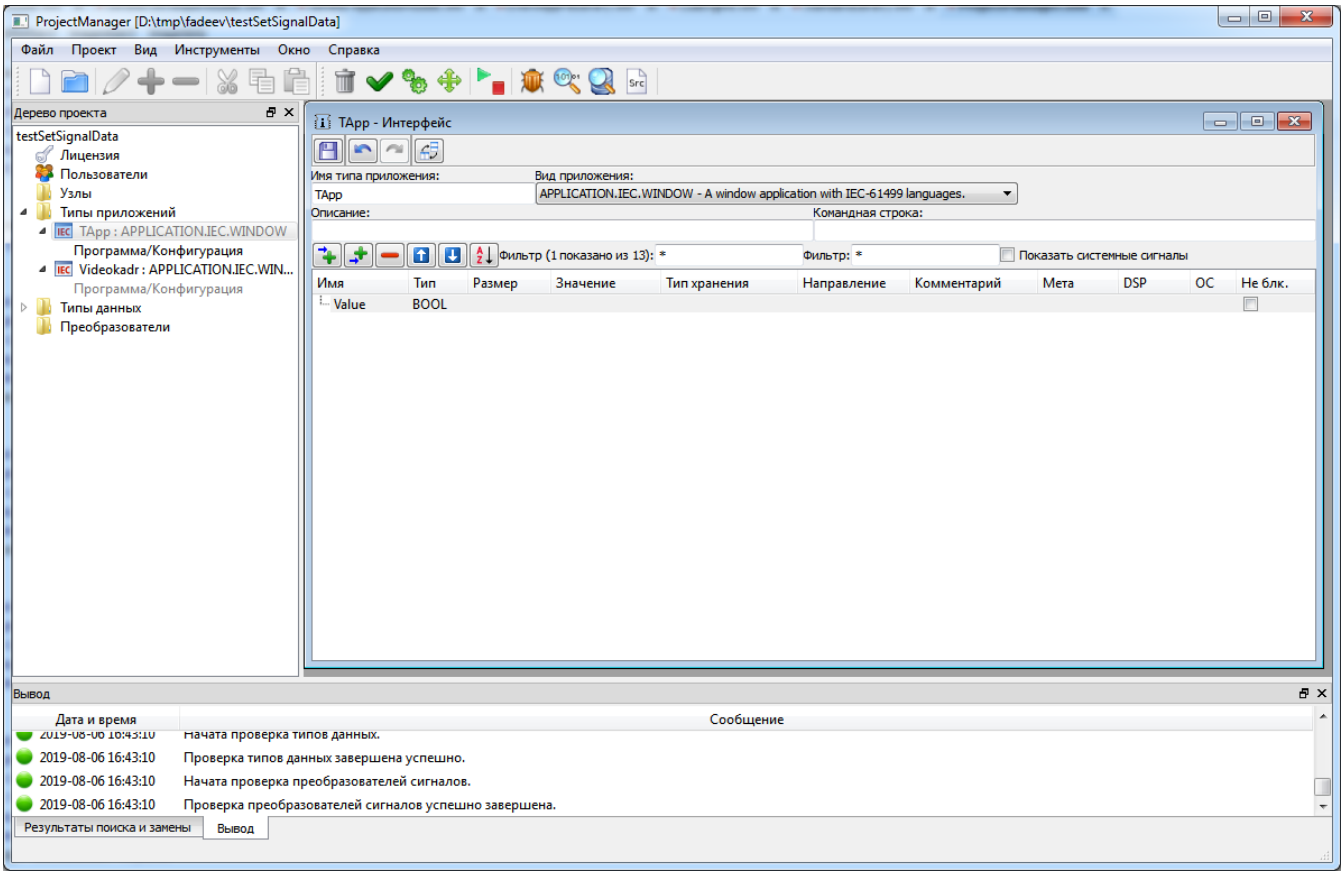


Рисунок 3.27 - Диалог для редактирования интерфейса типа приложения

Слева вверху в окне диалога редактирования расположены следующие кнопки:

 («Сохранить») – сохранить внесённые в интерфейс типа приложения изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

 («Изменить ориентацию») – изменить расположение окон, полей ввода и др. элементов.

Ниже расположены следующие поля ввода:

- **Имя типа** – имя типа приложения;
- **Вид приложения** – вид типа приложения (см. раздел 3.3.1);
- **Комментарий** – опциональное описание типа приложения;
- **Командная строка** – опциональная командная строка запуска экземпляра приложения данного типа.

Под полями ввода расположены кнопки редактирования таблицы интерфейса типа приложения:

 («Вставить сверху») – добавить новый элемент интерфейса сверху от выбранной в таблице позиции (см. раздел 3.2.2.4.3.2);

 («Вставить снизу») – добавить новый элемент интерфейса снизу от выбранной в таблице позиции (см. раздел 3.2.2.4.3.2);

 («Удалить») – удалить выбранный в таблице элемент интерфейса;

 («Поднять») – переместить выбранный в таблице элемент интерфейса вверх;

 («Опустить») – переместить выбранный в таблице элемент интерфейса вниз;

 («Принудительная сортировка») – перегруппировать элементы интерфейса по алфавиту.

Правее кнопок расположены управляющие компоненты для настройки фильтрации отображаемых в таблице элементов интерфейса типа приложения.

Доступны следующие виды фильтров:

- **Wilcard** – фильтрация по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

За компонентами для настройки фильтрации расположен флажок **Показывать системные сигналы**. Данный флажок управляет отображением системных сигналов в таблице элементов интерфейса (см. Приложение D).

Под кнопками расположена таблица элементов интерфейса.

У каждого элемента интерфейса есть следующие настраиваемые свойства:

- **Имя** – уникальное имя сигнала (допустимыми символами в имени являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя сигнала не может начинаться с цифры);
- **Тип** – тип данных сигнала (один из 19 видов сигнала, используемых в SCADA-системе "СОНАТА", см. Приложение А);
- **Размер** – размер данных сигнала (0 – скалярный сигнал, >0 – векторный сигнал);
- **Значения** – начальное значение сигнала;
- **Тип хранения** – тип хранения значения сигнала (доступные опции: **NORMAL** – при перезагрузке системы последнее значение сигнала не сохраняется, сигнал инициализируется начальным значением; **RETAIN** – при перезагрузке системы сохраняется последнее значение сигнала, "Холодный старт" сбрасывает значения **RETAIN** сигналов на начальное значение; **PERSISTENT** – сигналы всегда сохраняют своё последнее значение);
- **Направление** – направление сигнала в приложениях, служит для контроля и оптимизации передачи данных сигналов, а так же для работы признака недостоверности (НОПР, см. подробно ниже). Доступные направления сигналов:
  - **input** - сигнал считается как входной, принимающий значение от другого приложения;
  - **output** - сигнал считается как выходной, передающий значение другому приложению;

- **inout** - сигнал является одновременно выходным и входным;

- ___ - (пусто, значение по умолчанию). Для сигнала не применяются какие-либо правила контроля и ограничения.

Признак недостоверности НОПР формируется автоматически для сигнала в приложениях потребителях в случае потери связи этих приложений с приложениями источниками. Приложениями источниками являются те приложения, у которых сигнал помечен как output или inout. Все остальные приложения являются потребителями.

- **Комментарий** – опциональное справочное описание сигнала;

- **Мета** - это строка, в которую можно внести дополнительную информацию и в дальнейшем использовать её в Lua алгоритмах. Данную информацию никто не будет видеть. В Lua алгоритмах данную информацию извлекает команда Core.getSignalMeta;

- **DSP** – имя используемого преобразователя значения сигнала (о DSP см. раздел 3.2.2.6);

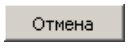
- **ОС** – сигнал, для которого компилятор, при отсутствии связи с другим сигналом, не будет выдавать предупреждения в ходе компиляции проекта (**Optional Connected**).

- **Не блк.** – данное поле используется при настройках резервирования узлов. Если поставить данную галочку, то сигнал не будет блокироваться (т.е. ведет себя как в рабочем приложении), когда приложение находится в состоянии резерва. В состоянии резерва приложение получает все изменения из вне, но ничего не выдает наружу. Данная галочка позволяет сигналу выдавать свои изменения наружу.

При создании нового типа приложения нужно обязательно, в соответствующих полях ввода, указать: имя типа приложения и вид приложения. Затем нужно заполнить таблицу элементов интерфейса перечнем сигналов.

Для добавления, удаления и перемещения элементов интерфейса используйте соответствующие кнопки, расположенные над таблицей. Для редактирования ячеек в таблице используйте двойной щелчок «мышью».

Для сохранения внесенных изменений служит кнопка  («Сохранить»).

По завершении редактирования интерфейса типа приложения диалог нужно закрыть крестиком  в верхнем правом углу окна редактирования. Если внесенные изменения не были сохранены ранее с помощью кнопки  («Сохранить»), то на экране появится диалог (см. рис. 3.28), в котором нужно выбрать либо сохранение изменений и выход из редактора (кнопка ) , либо отказ от внесенных изменений и выход из редактора (кнопка ) , либо отказаться от выхода из процедуры редактирования (кнопка ).

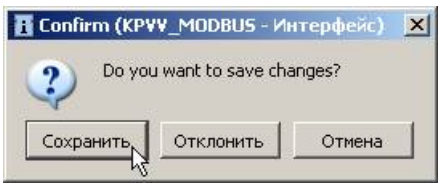


Рисунок 3.28 - Диалог подтверждения выхода из редактора интерфейса приложения

ВНИМАНИЕ! Если редактируемый тип приложения уже используется где-либо в проекте, то для переименования самого типа приложения или для переименования какого-либо элемента интерфейса необходимо воспользоваться диалогом поиска и замены. Данный диалог осуществляет изменение имени во всех элементах проекта, а не только в интерфейсе типа приложения.

3.2.2.4.3.1. Изменение имени типа и вида приложения

При создании нового приложения в SCADA-системе "СОНАТА" ему задаются имя типа **AppType_N**, где **N** - число от 1 и далее и вид приложения **APPLICATION** - внешнее неуправляемое приложение (см. рис. 3.27). Чтобы поменять имя типа приложения, необходимо в поле ввода **Имя типа** ввести удобное название для дальнейшего использования в проекте. Допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры (см. рис. 3.29).

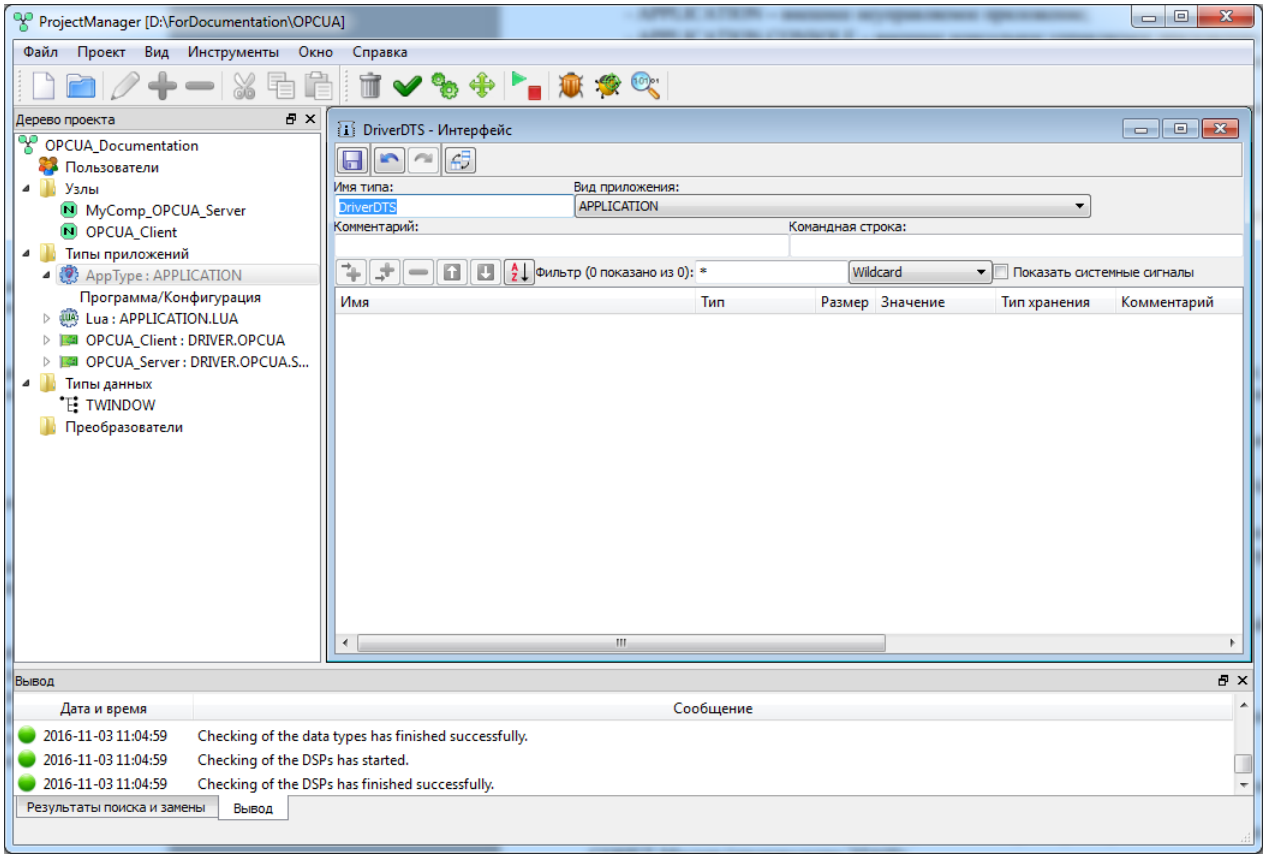


Рисунок 3.29 - Изменение имени типа приложения

Далее в поле выбора **Вид приложения** можно выбрать нужный вид приложения (см. раздел 3.3.1). На рис. 3.30 показано, как выбирается вид приложения.

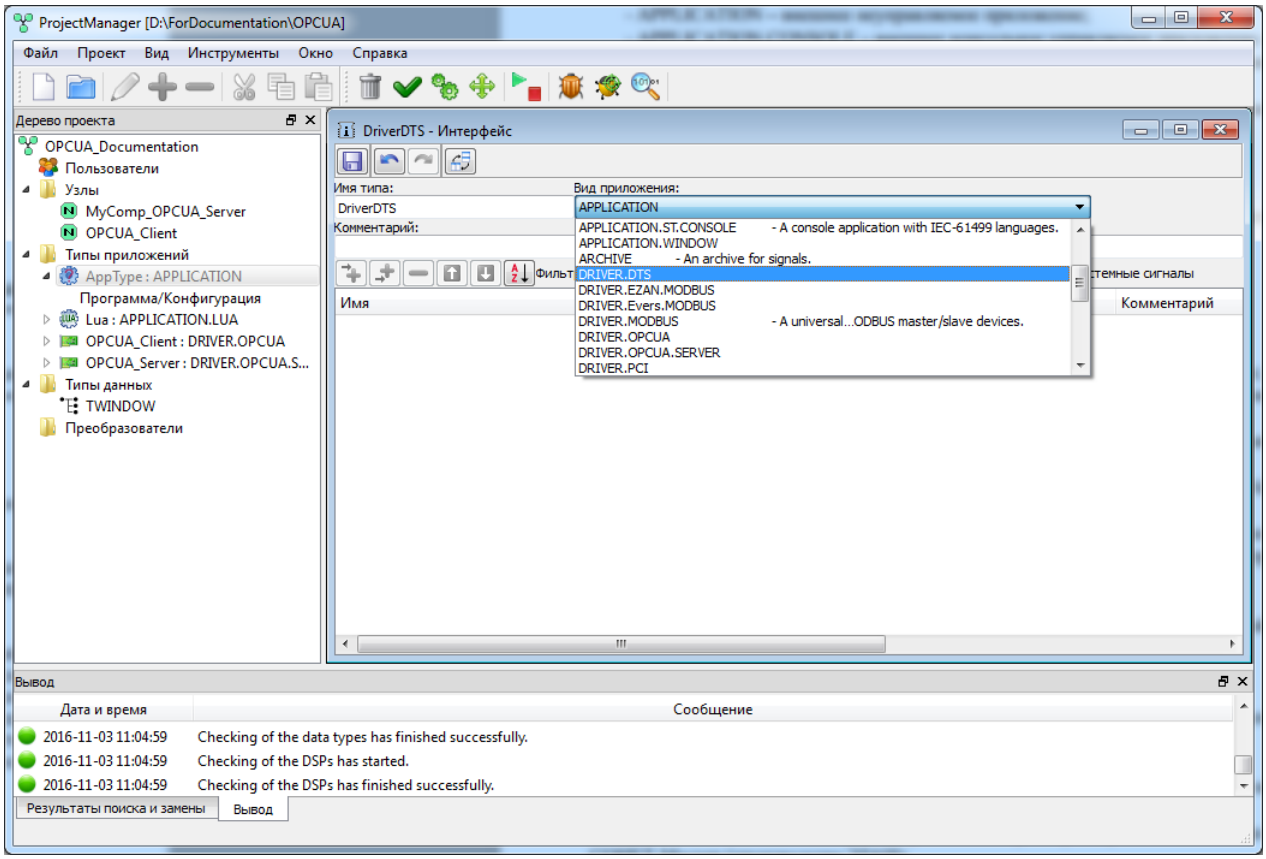


Рисунок 3.30 - Изменение вида приложения

3.2.2.4.3.2. Добавление локальных и глобальных сигналов в интерфейс приложения

Добавление локальных и глобальных (создание глобальных сигналов см. раздел 3.2.2.8) сигналов SCADA-системы "СОНАТА" в интерфейс приложения осуществляется кнопками  ("Вставить сверху") и  ("Вставить снизу"). По нажатию на данные кнопки откроется диалог выбора (см. рис. 3.31).

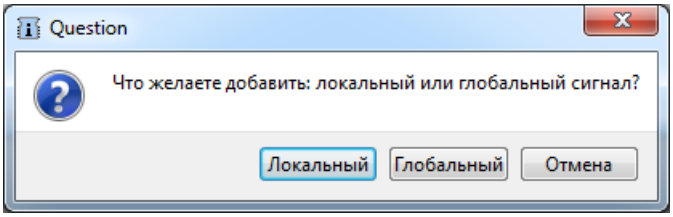


Рисунок 3.31 - Диалог - создать локальный сигнал или выбрать глобальный сигнал из таблицы глобальных сигналов SCADA-системы "СОНАТА"

Если нажать кнопку Локальный, то будет создан локальный сигнал приложения (см. рис. 3.32). Если нажать кнопку Глобальный, то откроется диалог выбора сигналов из таблицы глобальных сигналов SCADA-системы "СОНАТА" (см. рис. 3.33). Если нажать на кнопку Отмена, то диалог закроется и никакие сигналы не будут добавлены.

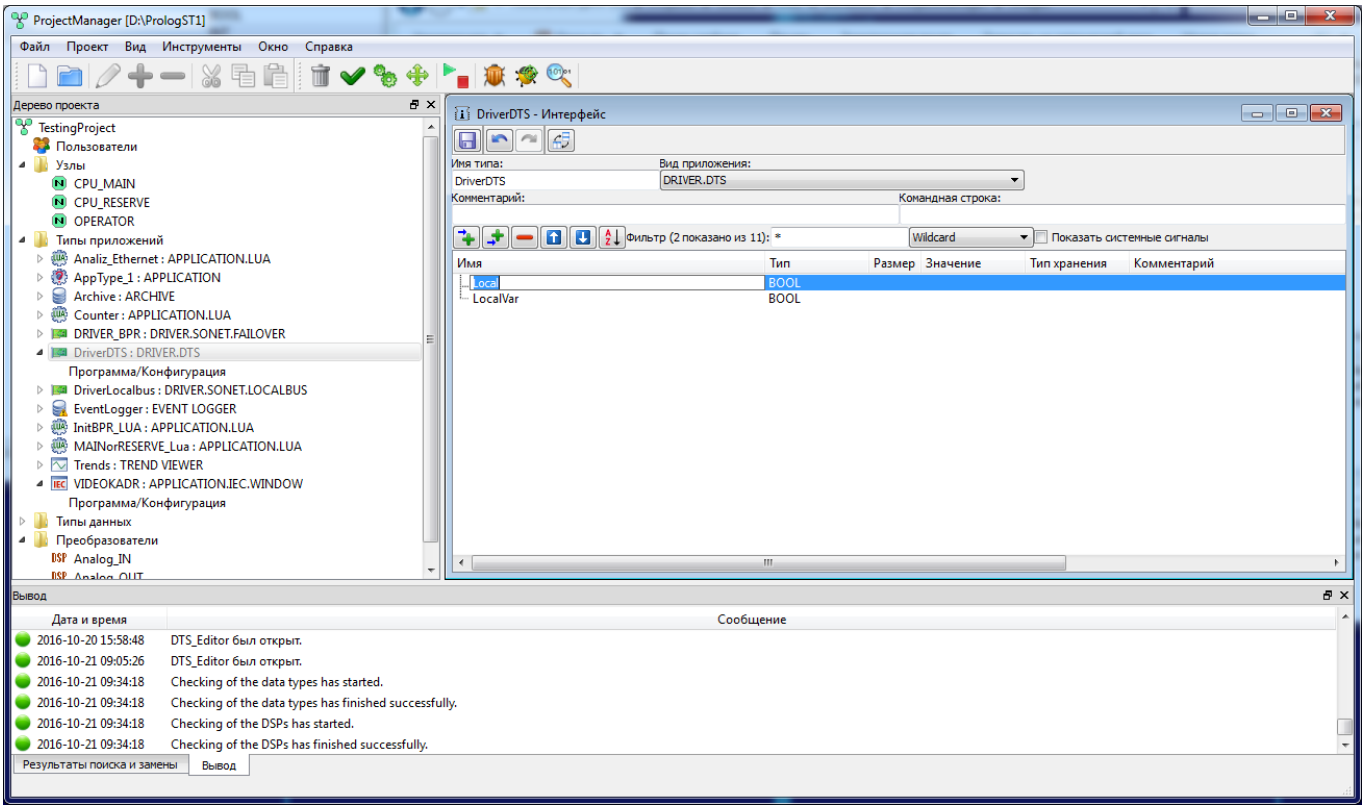


Рисунок 3.32 - Локальный сигнал в интерфейсе приложения

Как видно на рис. 3.32, в интерфейс приложения добавлен локальный сигнал с именем **Local**. Локальные сигналы добавляются с маской **Local_N**, где **N** - число от 1 и далее. Имя локального сигнала и его характеристики можно поменять в соответствующих ячейках, используя двойной щелчок «мыши». Описание полей столбцов, характеризующих сигнал, смотри в раздел 3.2.2.4.3.

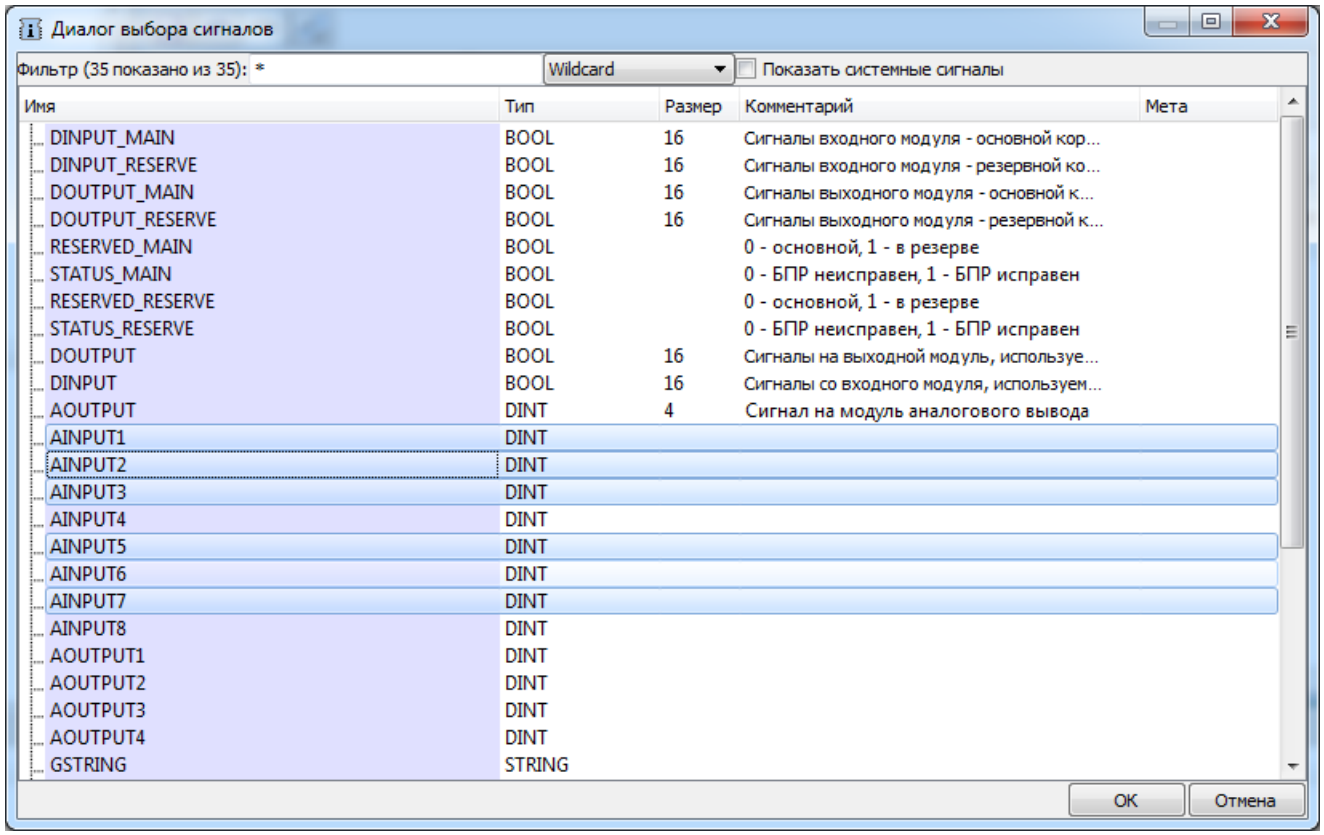


Рисунок 3.33 - Диалог выбора сигналов из таблицы глобальных сигналов SCADA-системы "СОНАТА"

Окно на рис. 3.33 состоит из строки фильтров (находится в верхней части под названием окна), таблицы глобальных сигналов (основная часть окна) и нижней строки подтверждения или отмены совершённых действий.

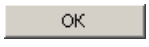
В строке фильтров отображается общее количество глобальных сигналов SCADA-системы "СОНАТА" и сколько отображено на данный момент, в соответствии с выборкой по шаблону. На рис. 3.33 отображено 35 сигналов из 35, что соответствует выборке по шаблону * (см. далее).

Поле **Wildcard** – выборка по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

Галочка **Показать системные сигналы** - добавляет или убирает системные сигналы SCADA-системы "СОНАТА" в список отображения. Описание системных сигналов SCADA-системы "СОНАТА" смотри в Приложение D.

Описание столбцов таблицы сигналов аналогично описанию в таблице глобальных сигналов SCADA-системы "Соната" (см. раздел 3.2.2.8.1).

С помощью левой кнопки «мыши» производится выбор одного или нескольких сигналов (не отпуская левую клавишу «мыши», выбираются несколько сигналов подряд), если нужны несколько сигналов, которые не располагаются рядом, то используйте клавишу **CTRL** на клавиатуре (зажмите и не отпускайте, в это время левой клавишей «мыши» выбирайте нужные сигналы). На рис. 3.33 выбраны сигналы INPUT1, INPUT2, INPUT3, INPUT5 и INPUT7, что видно по светло-синей раскраске ячеек данных сигналов.

По нажатию на кнопку 

все выбранные сигналы будут добавлены в интерфейс приложения (см. рис. 3.34).

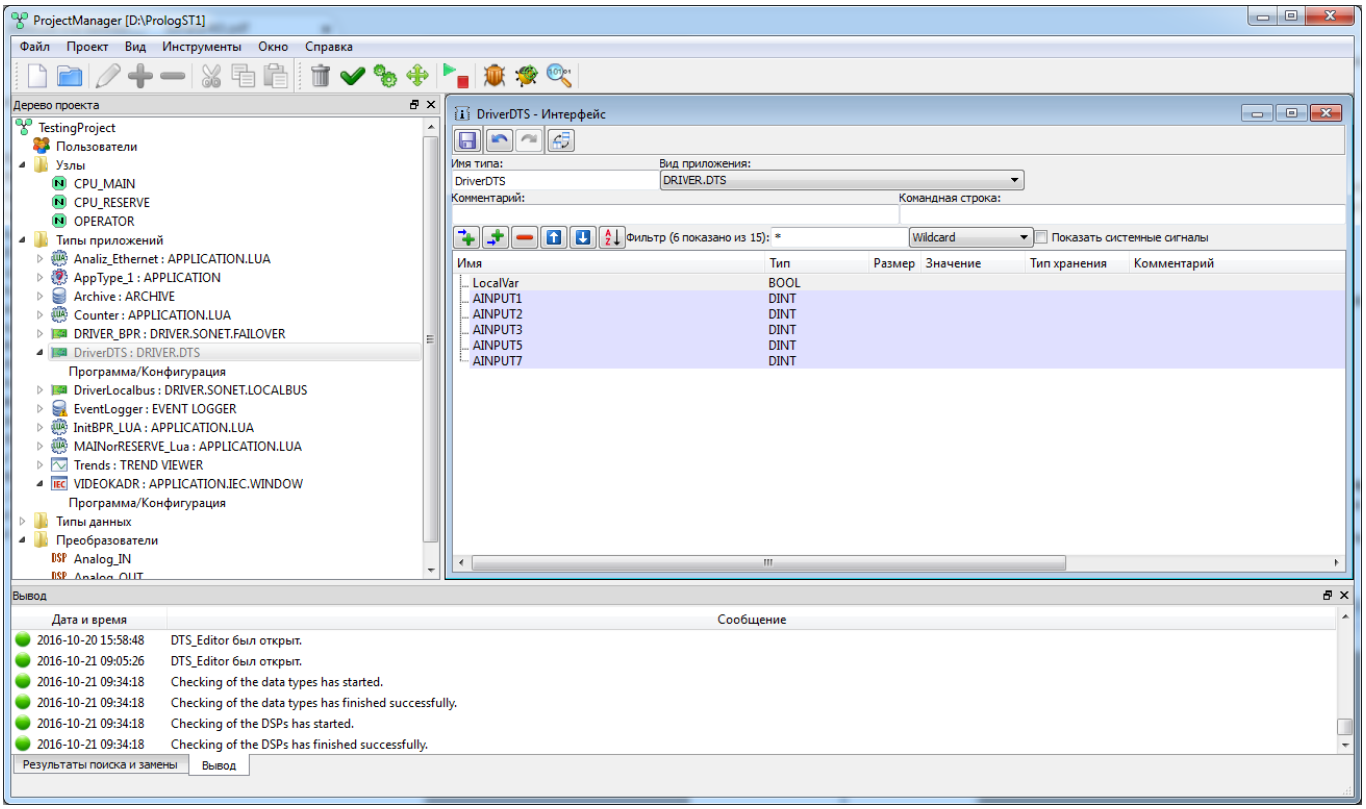
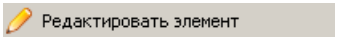


Рисунок 3.34 - Интерфейс приложения и его сигналы

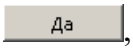
По нажатию на кнопку 

окно диалога выбора глобальных сигналов закроется, а сигналы не будут добавлены в интерфейс приложения.

3.2.2.4.4. Редактирование программы или конфигурации типа приложения

Для вызова специализированного редактора приложения сначала необходимо раскрыть папку **типа приложения** в дереве проекта, а затем установить курсор на строку Программа/Конфигурация и выделить ее, далее либо нажать кнопку  («Редактировать элемент») на главной панели инструментов (см. рис. 3.35), либо вызвать контекстное меню с помощью правой кнопки мыши, а затем выбрать команду 

(см. рис. 3.36), либо выбрать в главном меню в разделе Проект команду 

(см. рис. 3.37). Перед открытием появится диалог создания нового конфигурационного файла приложения (см. рис. 3.38). Если выбрать 

, то будет создан конфигурационный файл приложения и откроется специализированный редактор. В зависимости от вида приложения открываются разные редакторы (описание редакторов идет вместе с описанием типов приложений), есть приложения, у которых данная функция отсутствует.

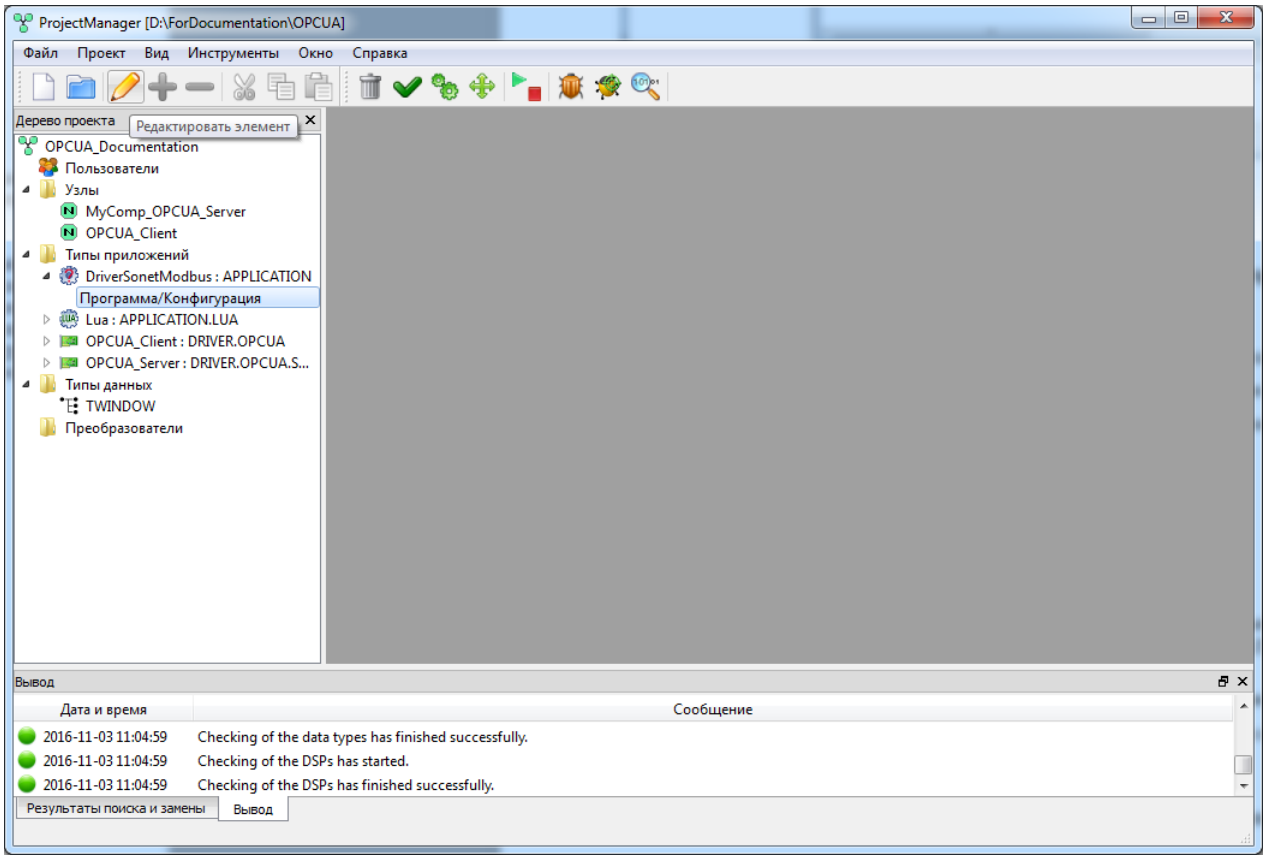


Рисунок 3.35 - Запуск специализированного редактора приложения с помощью кнопки главной панели инструментов "Редактировать элемент"

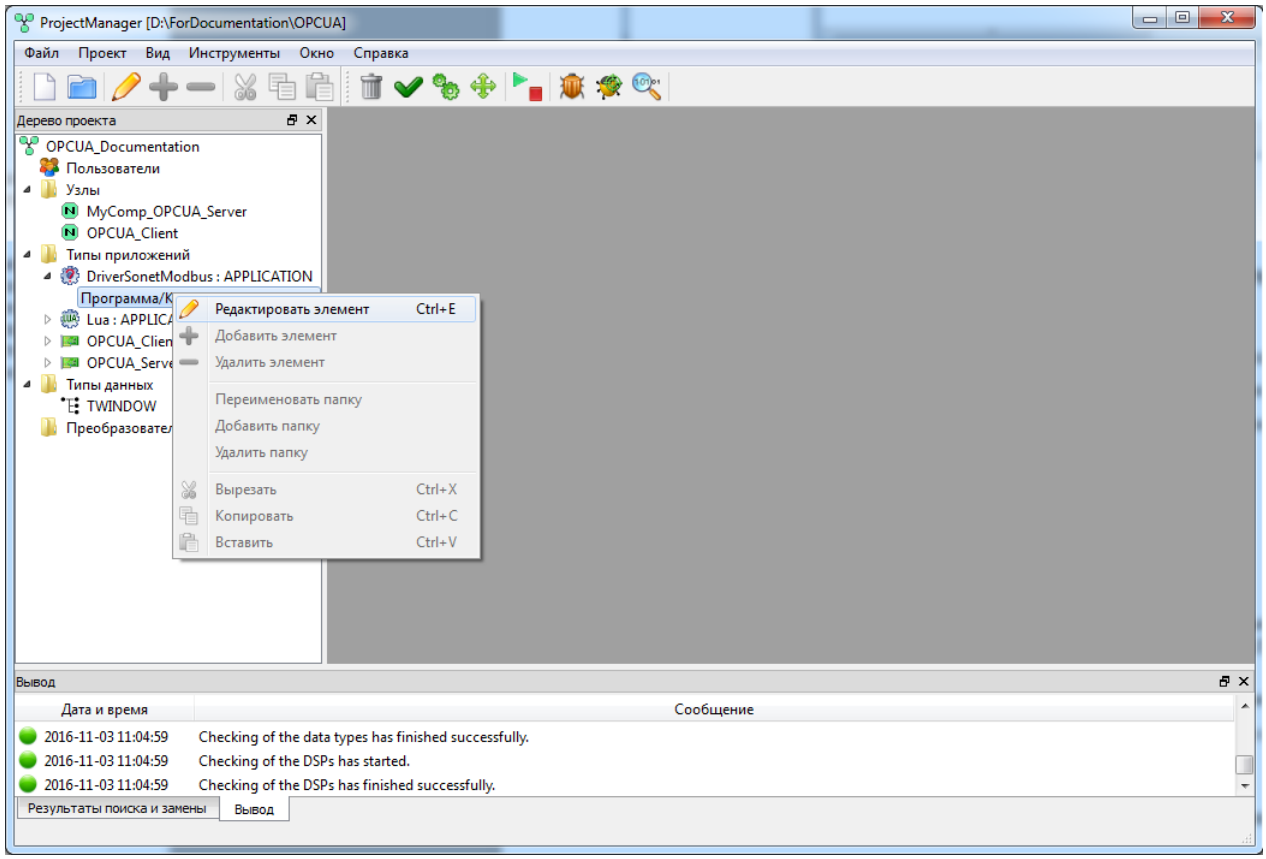


Рисунок 3.36 - Запуск специализированного редактора приложения с помощью команды контекстного меню "Редактировать элемент

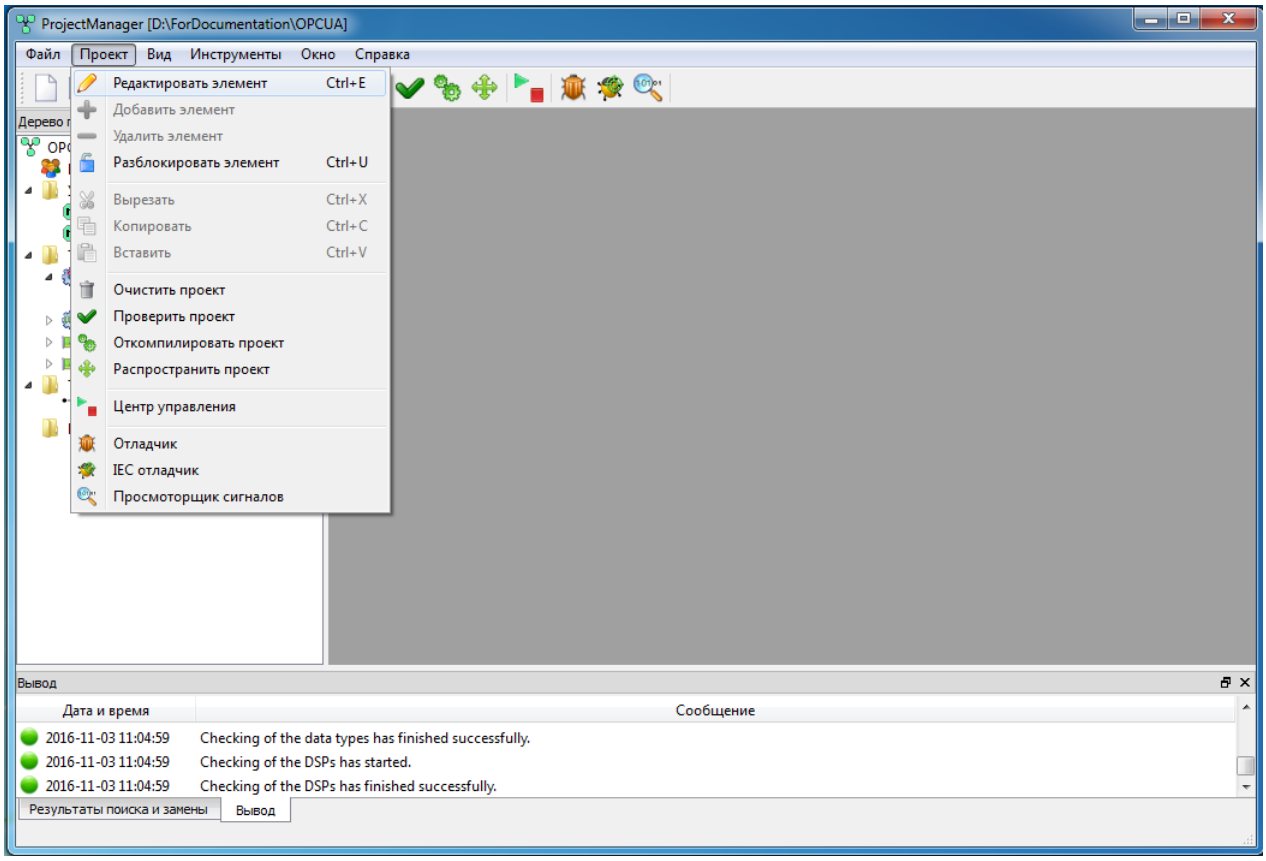


Рисунок 3.37 - Запуск специализированного редактора приложения
с помощью команды главного меню Редактировать элемент

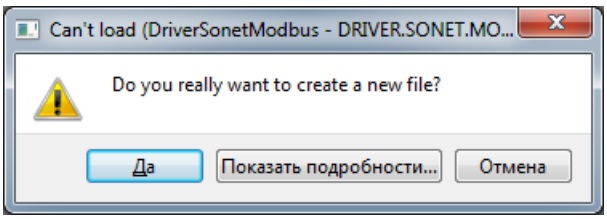


Рисунок 3.38 - Диалог создания нового конфигурационного файла приложения

Внимание! При редактировании каких-либо свойств в любом из окон ProjectManager, эти свойства не будут автоматически подхвачены зависимыми окнами. Например, при редактировании интерфейса типа приложения (добавление/удаление сигналов), эти изменения не будут автоматически подхвачены редактором архивируемых сигналов. Чтобы внесённые изменения были учтены зависимыми окнами (редакторами), данные окна требуется переоткрыть.

3.2.2.5. Редактирование конфигурации узлов

Проект АСУ ТП состоит из совокупности взаимодействующих экземпляров приложений, запущенных на узлах системы. В процессе редактирования конфигурации узлов создается перечень узлов системы с указанием списка экземпляров приложений, выполняемых на том или ином узле.

3.2.2.5.1. Создание нового узла

Для создания нового узла нужно выделить в дереве проекта главного окна программы **ProjectManager** папку Узлы и нажать кнопку  («Добавить элемент») на главной панели инструментов (см. рис. 3.39). В перечень узлов в дереве проекта будет добавлена новая позиция. Имена новым узлам присваиваются автоматически по шаблону **Node_N**, где **N** – уникальное число. В дальнейшем имя узла можно заменить на более подходящее.

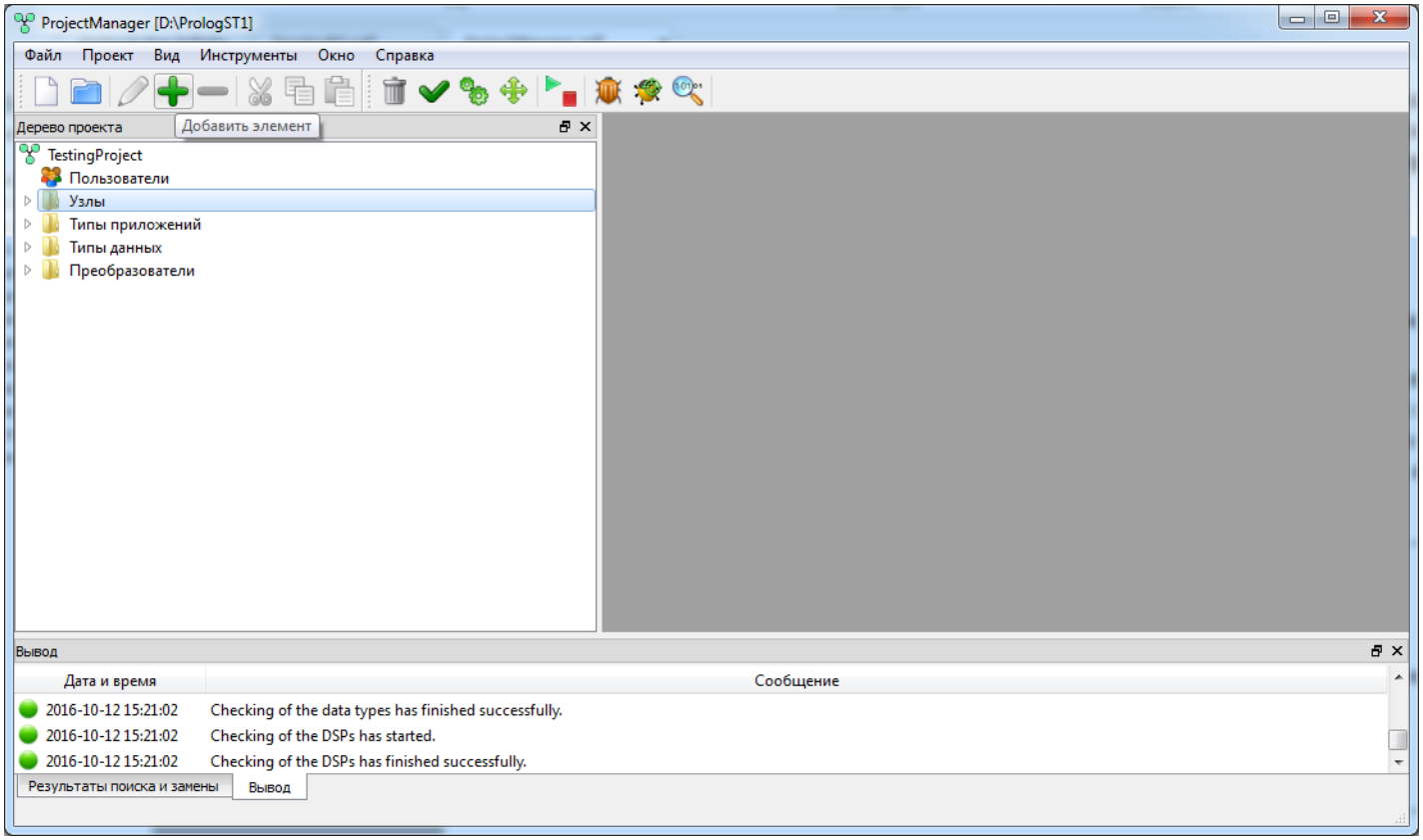


Рисунок 3.39 - Создание нового узла

3.2.2.5.2. Удаление конфигурации узла

Для удаления узла нужно в дереве проекта главного окна **ProjectManager** выделить позицию удаляемого узла и нажать кнопку  («Удалить элемент») на главной панели инструментов. Будет выведен диалог подтверждения удаления. При положительном ответе узел будет удалён.

3.2.2.5.3. Редактирование конфигурации узла

Для редактирования конфигурации узла нужно в дереве проекта главного окна **ProjectManager** выделить позицию соответствующего узла и нажать кнопку  («Редактировать элемент») на главной панели инструментов. Откроется диалог для редактирования конфигурации узла (см. рис. 3.40).

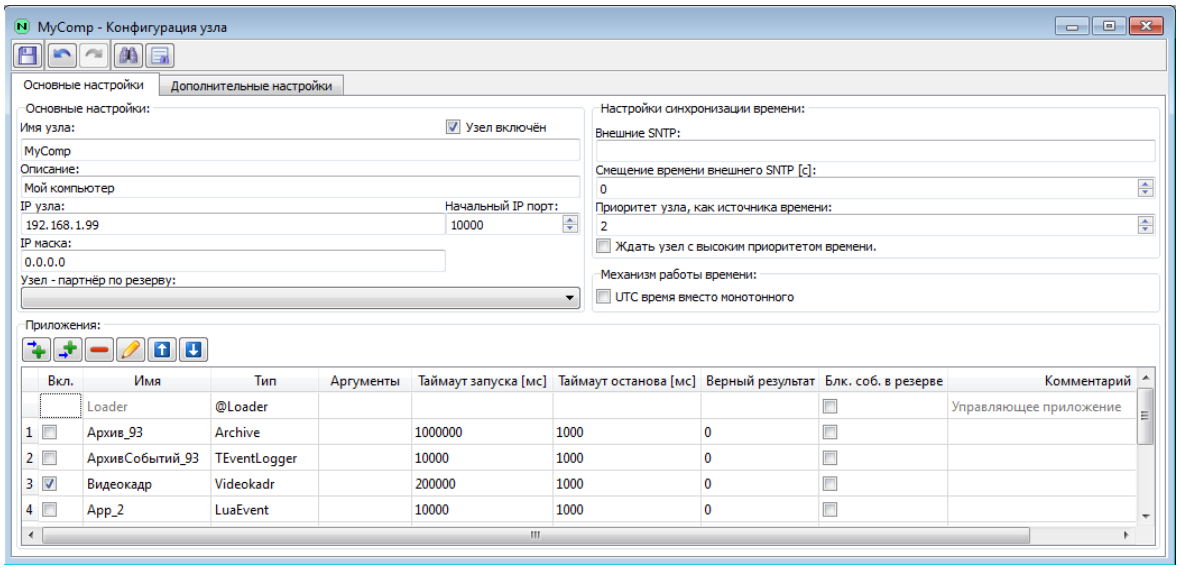


Рисунок 3.40 - Диалог для редактирования конфигурации узла

Вверху окна диалога расположены следующие кнопки:

 («Сохранить») – сохранить внесённые в конфигурацию узла изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

 ("Поиск") - откроется окно диалога поиска;

 ("Показать/скрыть результаты поиска") - показать или скрыть окно с результатами поиска.

Ниже расположены две вкладки с настройками узла:

- **Основные настройки** - основные настройки, которые нужны для работы узла;

- **Дополнительные настройки** - дополнительные настройки узла позволяют настроить еще некоторое количество параметров (можно оставить со значениями по умолчанию).

Вкладка **Основные настройки** состоит из следующих параметров (см. рис. 3.40):

- **Основные настройки:**

- **Имя узла** – имя узла, которое будет отображаться в информационных сообщениях, имя узла должно подчиняться требованиям стандарта IEC-61131 на именование переменных (имя может содержать только буквы и цифры, а также символ подчёркивания);

- **Узел включён** – флаг, используемый для временного отключения компиляции данного узла, если галочка стоит, то узел включён в проект, иначе исключен из проекта и компилироваться не будет;

- **Описание** – подробное описание узла;

- **IP узла** – IP-адрес узла, если узел имеет несколько сетевых карт и используется дублирование сети, то в этом поле указываются адреса сетей через запятую (к примеру 192.168.1.93,192.168.2.93);

- **Начальный IP порт** – порт, на котором предварительно запущен Loader данного узла. Остальные приложения данного узла будут запускаться на следующих портах в порядке увеличения на 1. Данный функционал позволяет запускать на одном АРМ несколько узлов проекта (см. подробно раздел 3.5);

- **IP маска** - маска подсети. Используется для того, чтобы описать или ограничить (оптимизировать) количество возможных связей данного узла с другими узлами. Для узлов проекта, которые не подходят введёной IP маске, при компиляции не будут создаваться связи между сигналами. К примеру узел с ip-адресом 192.168.1.* и маской 255.255.255.**0** будет иметь связи с узлом 192.168.1.* маска 255.255.255.**0**. Но не будет иметь связей с узлом 192.168.**2**.* маска

255.255.**255**.0 (обратите внимание на выделенные значения в масках и адресах). По умолчанию на узле выставляется маска 0.0.0.0, что означает все возможные варианты ip-адресов. В данном случае вся маршрутизация сетевых пакетов лежит на совести операционной системы. Если в поле **IP узла** введено несколько ip-адресов через запятую, то для каждого ip-адреса нужно через запятую указать свою маску.

- **Узел - партнер по резерву** – в данном выпадающем списке можно выбрать узел, который выполняет функцию резервного для данного узла;

- **Настройки синхронизации времени:**

- **Внешние SNTP** - в данном поле могут быть указаны внешние (относительно проекта) сервера или устройства, отдающие точное время по протоколу SNTP/NTP;

- **Смещение времени внешнего SNTP[s]** - это значение в секундах, которое прибавляется к значению времени, взятому с внешнего SNTP (данный параметр может быть отрицательным);

- **Приоритет узла, как источника времени** - необходимо указать число, которое будет обозначать приоритет данного узла по отношению к другим узлам проекта, как источника времени. Чем меньше число, тем выше приоритет. Все узлы будут пытаться получить точное время от узла с самым высоким приоритетом (меньшим числом), если он не доступен, то от узлов с меньшим приоритетом. И т. д. Все узлы образуют иерархию источников точного времени и поэтому в проекте не может быть двух узлов с одинаковым приоритетом;

ВАЖНО! При старте проекта происходит ударная синхронизация времени, а при работе проекта происходит безударная синхронизация времени с подтяжкой 3-5 сек в час.

Описание процесса синхронизации времени смотрите в SCADA-система "Соната" Описание применения КУНИ.505200.023-01.01 31.

- **Ждать узел с высоким приоритетом времени** - если поставить данную галочку, то узел не будет запускаться пока не получит время от узла с более высоким приоритетом времени;

- **Механизм работы времени:**

- **UTC время вместо монотонного** - при выставлении данного флага сигналам проставляется отметка времени об их изменении, используя локальное время на данном компьютере. В данном режиме система чувствительна к ударному переводу часов на компьютере. При снятой отметке на данном флаге используется монотонное время, не привязанное к календарному времени на данном компьютере;

Ниже располагается таблица **Приложения** со списком запускаемых на узле приложений. Приложения запускаются на узле в порядке, указанном в таблице.

В таблице **Приложения** есть следующие поля:

- **Вкл.** - если данная галочка включена, то приложение будет запускаться на узле, иначе не будет запускаться;

- **Имя** – имя экземпляра приложения, которое будет отображаться в информационных сообщениях;

- **Тип** – тип экземпляра (шаблона) приложения;

- **Аргументы** – аргументы командной строки для запуска экземпляра приложения (при запуске будут использованы аргументы типа приложения плюс аргументы экземпляра приложения). Описание аргументов смотрите в Приложение J;

- **Таймаут запуска[ms]** – время запуска приложения, т.е. время от момента запуска исполняемого файла загрузчиком узла, до момента, когда ядро приложения становится видимым в системе и переходит в состояние приёма команд управления;

- **Таймаут останова[ms]** – время для корректного завершения работы приложения, по истечении которого отдается команда на принудительное завершение процесса;

- **Верный результат** – время для корректного завершения работы приложения, по истечении которого отдается команда на принудительное завершение процесса;

- **Блк. соб. в резерве** – если приложение находится в состоянии резерва, то при выставленной данной галочке, события от приложения не будут поступать;

- **Комментарий** – опциональное поле для описания приложения.

Важно! В первой строке таблицы приложений в колонке **Тип** по умолчанию указан Loader. При использовании в проекте резервирования узлов, в данной колонке необходимо указать управляющее приложение, т.е. приложение, которое управляет переключением между приложениями партнерами. На данный момент управляющим приложением является встроенное приложение **TControlApp.APPLICATION.LUA**. В него можно вносить изменения для решения своих задач резервирования. Данное приложение будет автоматически создано в списке типов приложений, если создавать новый проект через меню Файл - Новый проект с резервированием.

Над таблицей со списком запускаемых на узле экземпляров приложений расположены кнопки для её модификации:

 («Вставить сверху») – добавить новую программу сверху от выбранной в таблице позиции;

 («Вставить снизу») – добавить новую программу снизу от выбранной в таблице позиции;

 («Удалить») – удалить выбранную в таблице программу;

 («Редактировать») – редактировать выбранную в таблице программу;

 («Поднять») – переместить выбранную в таблице программу вверх;

 («Опустить») – переместить выбранную в таблице программу вниз.

При редактировании конфигурации узла необходимо задать имя узла и его сетевой адрес, а затем составить список исполняемых на узле приложений (имена приложений должны подчиняться требованиям стандарта IEC-61131 на именование переменных: имя может содержать только буквы и цифры, а также символ подчёркивания).

Первая строка в писке приложений применяется в качестве настройки управляющего приложения. Данное управляющее приложение в основном используется для описания алгоритмов переключения при резервировании. Для алгоритмов резервирования используется тип приложения APPLICATION.LUA.

Существует возможность изменять (перегружать) начальные значения сигналов экземпляров приложений, определённых в их типах. Для этого нужно выделить в таблице приложений нужную позицию и нажать кнопку  («Редактировать») – появляется форма редактирования сигналов конкретного экземпляра приложения (см. рис. 3.41)

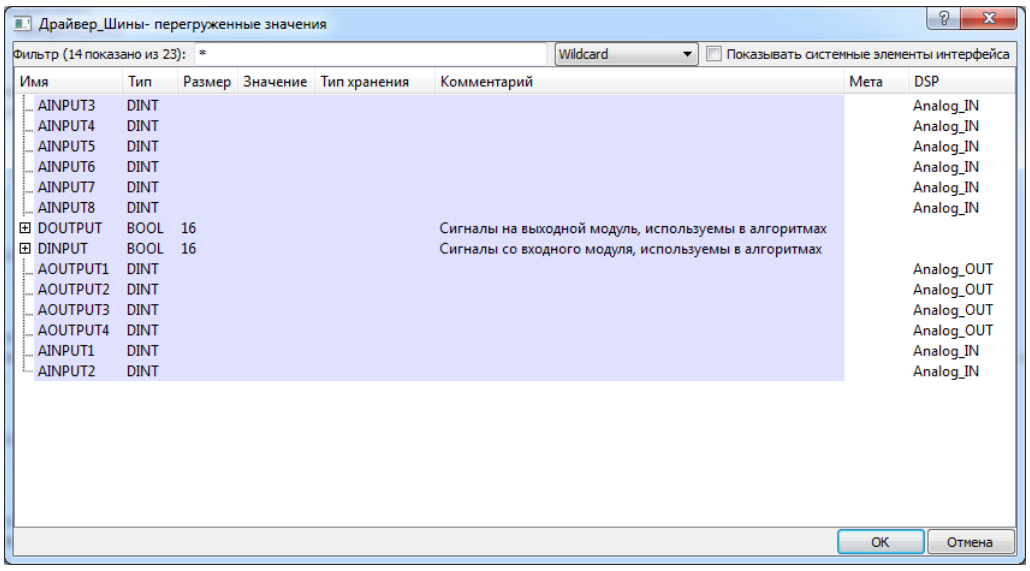


Рисунок 3.41 - Диалог для редактирования сигналов экземпляра приложения

Во вкладке Дополнительные настройки находятся следующие параметры (см. рис. 3.42):

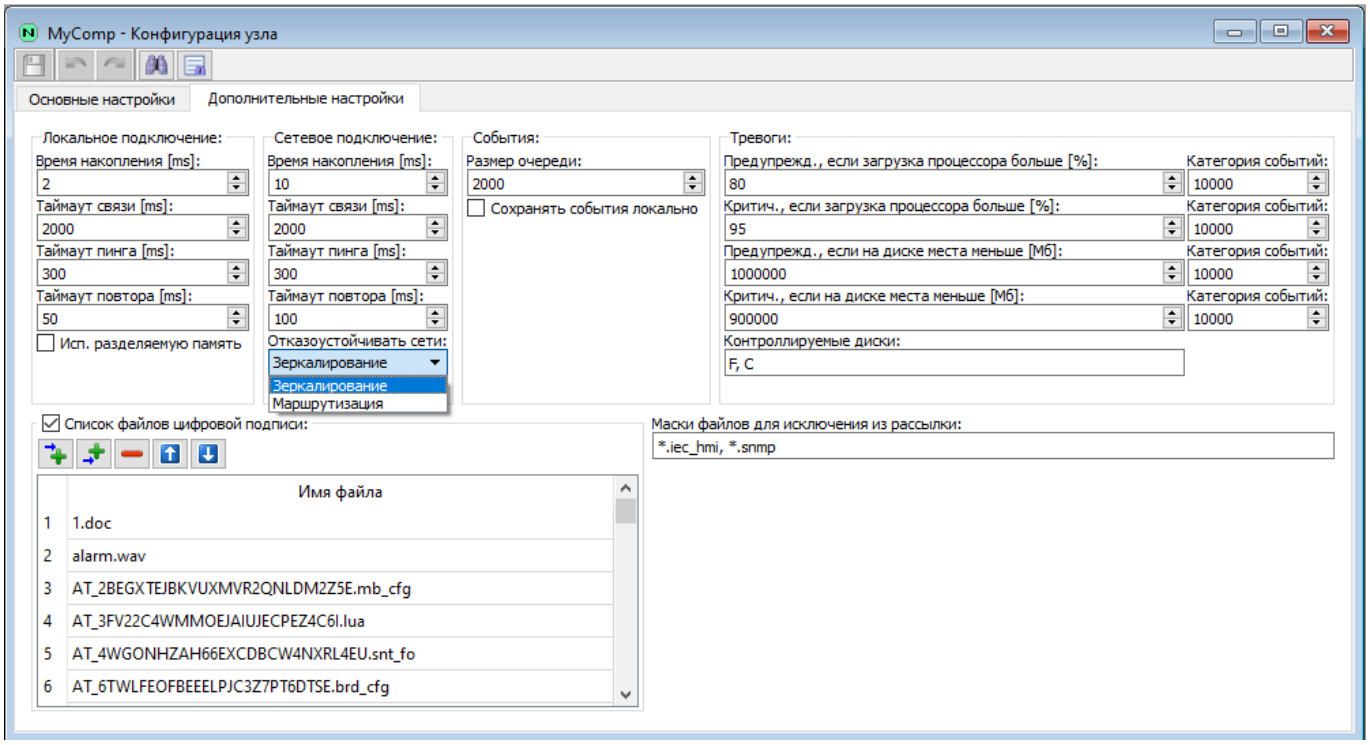


Рисунок 3.42 - Вкладка Дополнительные настройки

- Локальное подключение (настройки взаимодействия между приложениями в пределах одного узла):

- Время накопления[ms] - максимальное время, которое затрачивается на формирование пакета информации, отправляемого по сети. Если пакет сформируется, т.е. наполнится информацией быстрее, то он будет отправлен сразу по заполнению. Объем одного пакета вмещает информацию примерно о 40 сигналах;

- Таймаут связи[ms] - максимально допустимое время (в миллисекундах) на подтверждение приёма сетевого пакета с уведомлением;

- Таймаут пинга[ms] - время (в миллисекундах) ожидания прихода ответа на запрос;

- Таймаут повторов[ms] - время (в миллисекундах) перед началом повторного запроса.

- Сетевое подключение (настройки взаимодействия между приложениями на разных узлах):

- Время накопления[ms] - максимальное время, которое затрачивается на формирование пакета информации, отправляемого между приложениями одного узла. Если пакет сформируется, т.е. наполнится информацией быстрее, то он будет отправлен сразу по заполнению. Объем одного пакета вмещает информацию примерно о 40 сигналах;

- Таймаут связи[ms] - максимально допустимое время (в миллисекундах) на подтверждение приёма сетевого пакета с уведомлением;

- Таймаут пинга[ms] - время (в миллисекундах) ожидания прихода ответа на запрос;

- Таймаут повторов[ms] - время (в миллисекундах) перед началом повторного запроса;

- Отказоустойчивость сети - возможны два ваимоисключающих варианта (на всех узлах системы должен быть выбран одинаковый вариант)(описание и схемы подключения см. в документе SCADA-система "СОНАТА" Описание применения КУНИ.505200.023-01.01.31 п. Функция разграничения передачи информации по IP-адресам и маскам подсети между узлами системы):

- Зеркалирование - используются все указанные IP адреса узлов, по которым выполняется синхронная рассылка сетевых пакетов. Маршрутизацией занимается операционная система;

- Маршрутизация - в сети должен присутствовать хотя бы один узел с двумя сетевыми картами и двумя непересекающимися подсетями. На данном узле Соната выполняет функцию передачи сетевых пакетов из одной подсети в другую. ВНИМАНИЕ! Количество маршрутов

в сети равно 2*N, где N - количество узлов с двумя IP адресами (сетевыми картами). С увеличением количества маршрутов будет увеличиваться и нагрузка на узлах.

- **События** (каждое приложение имеет индивидуальный кэш событий, данные настройки влияют на все приложения узла):

- **Размер очереди** - количество событий, накапливаемых приложением, в случае отсутствия связи с регистратором событий;

- **Сохранять события локально**-при установленном флаге, события дополнительно сохраняются на диск. Это обеспечивает сохранность событий при временном отсутствии связи с регистратором событий и перезагрузки узла.

- **Тревоги:**

- **Предупрежд., если загрузка процессора больше [%]** - в данном поле настраивается процент загрузки процессора узла, при котором будет генерироваться предупредительная тревога. Если значение поставить равное 0, то тревога генерироваться не будет. Правее находится поле **Категория событий**, в котором настраивается категория к которой будет относиться данная тревога;

- **Критич., если загрузка процессора больше [%]** - в данном поле настраивается процент загрузки процессора узла, при котором будет генерироваться критическая (аварийная) тревога. Если значение поставить равное 0, то тревога генерироваться не будет. Правее находится поле **Категория событий**, в котором настраивается категория к которой будет относиться данная тревога;

- **Предупрежд., если на диске места меньше [Мб]** - в данном поле настраивается объем диска узла в Мб, при котором будет генерироваться предупредительная тревога. Если значение поставить равное 0, то тревога генерироваться не будет. Правее находится поле **Категория событий**, в котором настраивается категория к которой будет относиться данная тревога;

- **Критич., если на диске места меньше [Мб]** - в данном поле настраивается объем диска узла в Мб, при котором будет генерироваться критическая (аварийная) тревога. Если значение поставить равное 0, то тревога генерироваться не будет. Правее находится поле **Категория событий**, в котором настраивается категория к которой будет относиться данная тревога;

- **Контроллируемые диски:** - в данном поле можно указать имена дисков в системе, по которым будет осуществляться контроль загрузки процессора и свободного места. Если данное поле пустое, то контроль осуществляется по всем дискам. Для настройки нескольких дисков, их нужно перечислить через запятую (C, D, E).

Рассмотрим пример тревоги по диску. При тех настройках, что на рис. 3.42. При уменьшении свободного объема диска меньше 1024 Мб (1 Гб) сформируется тревога предупредительная, с указанной в правом столбце категорией. При последующем уменьшении свободного диска до 100 Мб, тревога предупредительная исчезнет и возникнет тревога аварийная, о том, что место на диске совсем заканчивается. При увеличении объёма свободно пространства на диске > 1024 Мб обе тревоги исчезнут. Контроль происходит при каждом старте проекта и далее каждые 10 минут. Дополнительно данная информация пишется в логи узла.

Слева в нижней части окна дополнительных настроек располагается список файлов цифровой подписи (см. рис. 3.43).

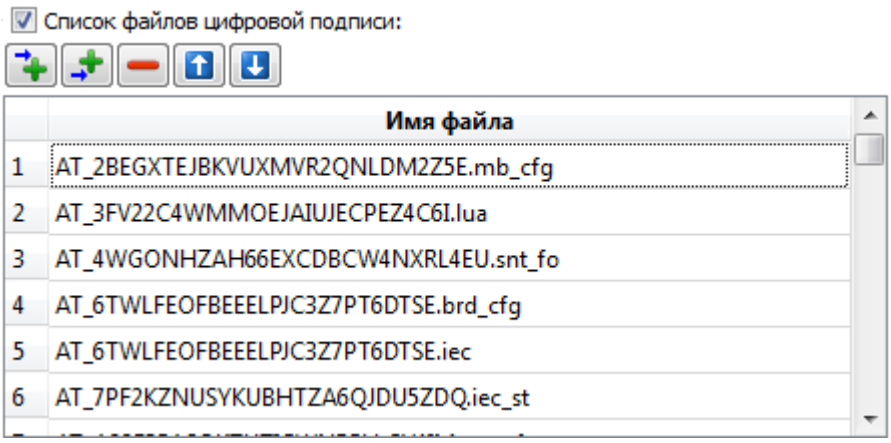


Рисунок 3.43 - Окно настроек списка файлов цифровой подписи

В данном окне выполняется настройка списка файлов, которые будут подписаны цифровой подписью при использовании данной функции. Функционал цифровой подписи описан в п. Программа управления "ControlCenter" или Центр управления (см. рис. 3.44).

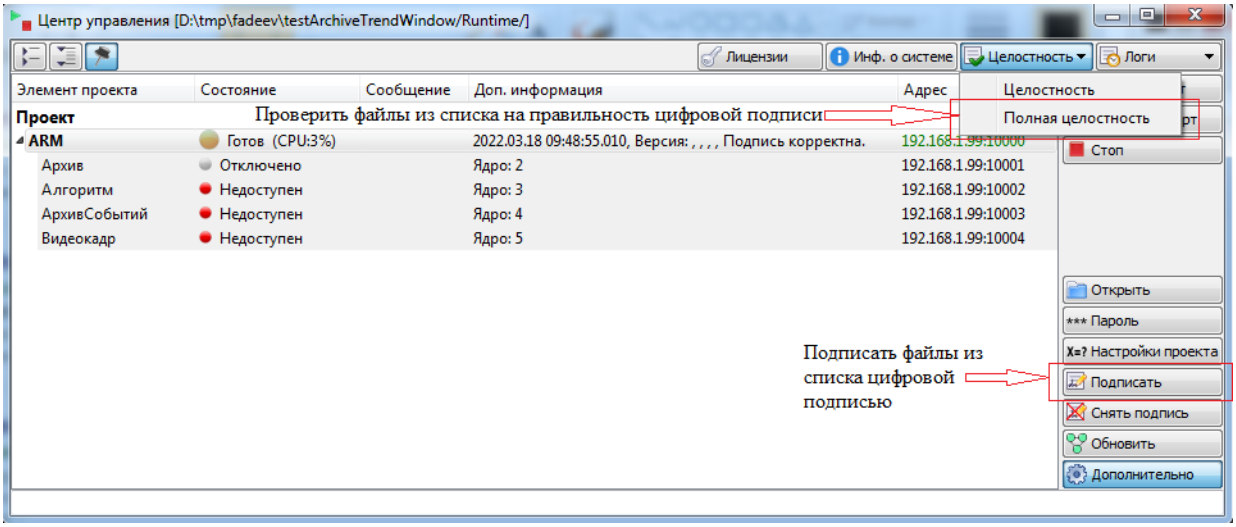


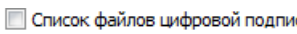
Рисунок 3.44 - Функционал цифровой подписи в приложении Центр управления

Для включения функционала настройки списка файлов цифровой подписи нужно активировать флаговую кнопку Список файлов цифровой подписи:. При активации данной флаговой кнопки в список будут добавлены все файлы проекта. Далее можно с помощью кнопок настроить измененный список файлов.

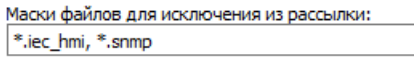
Внимание! В список файлов цифровой подписи так же всегда входят все файлы дистрибутива SCADA "Соната". Это файлы приложений, библиотеки и др. Данные файлы не отображаются в списке файлов цифровой подписи, т.к. их нельзя исключать из данного списка.

Описание кнопок:

- - добавить дополнительную строку выше выделенной;
- - добавить дополнительную строку ниже выделенной;
- - переместить выделенную строку на одну позицию вверх;
- - переместить выделенную строку на одну позицию вниз;
- - удалить выделенные строки.

Если функция настройки списка файлов цифровой подписи не активна 

, то в данном случае при использовании цифровой подписи будут подписаны все файлы проекта.

Справа в нижней части окна дополнительных настроек располагается строка 

- данная настройка позволяет убрать из рассылки на узел файлы, которые соответствуют настроенным маскам.

3.2.2.6. Редактирование преобразователей значения сигнала

Преобразователь значения сигнала осуществляет преобразование значения сигнала из внутреннего представления программы во внешнее представление (которое пересылается другим программам) и осуществляет обратную операцию: преобразование внешнего представления значения сигнала во внутреннее. Для преобразования используется монотонная кусочно-линейная функция.

Преобразователи значения сигнала в основном используются приложениями-драйверами, для пересчёта кодов аналогово - цифрового преобразователя (АЦП) в физическую величину, например, температуру или давление.

В текущей версии SCADA-системы "Соната" преобразователь сигнала может осуществлять две операции:

- преобразование значения при помощи монотонно возрастающей или монотонно убывающей кусочно-линейной функции с неограниченным количеством точек;

- квантование (загрубение) значения до указанной точности (квантование используется для сокращения нагрузки на сеть и вычислительные алгоритмы).

Для разных сигналов может быть использован один и тот же преобразователь значения сигнала.

3.2.2.6.1. Создание нового преобразователя значения сигнала

Для создания нового преобразователя нужно выделить в дереве проекта главного окна **ProjectManager** папку **Преобразователи** и нажать кнопку  («Добавить элемент») на главной панели инструментов (см. рис. 3.45). В перечень преобразователей значения сигнала в дереве проекта будет добавлена новая позиция. Имена новым преобразователям присваиваются автоматически по шаблону **DSP_N**, где **N** –уникальное число. В дальнейшем имя преобразователя можно легко заменить на более подходящее.

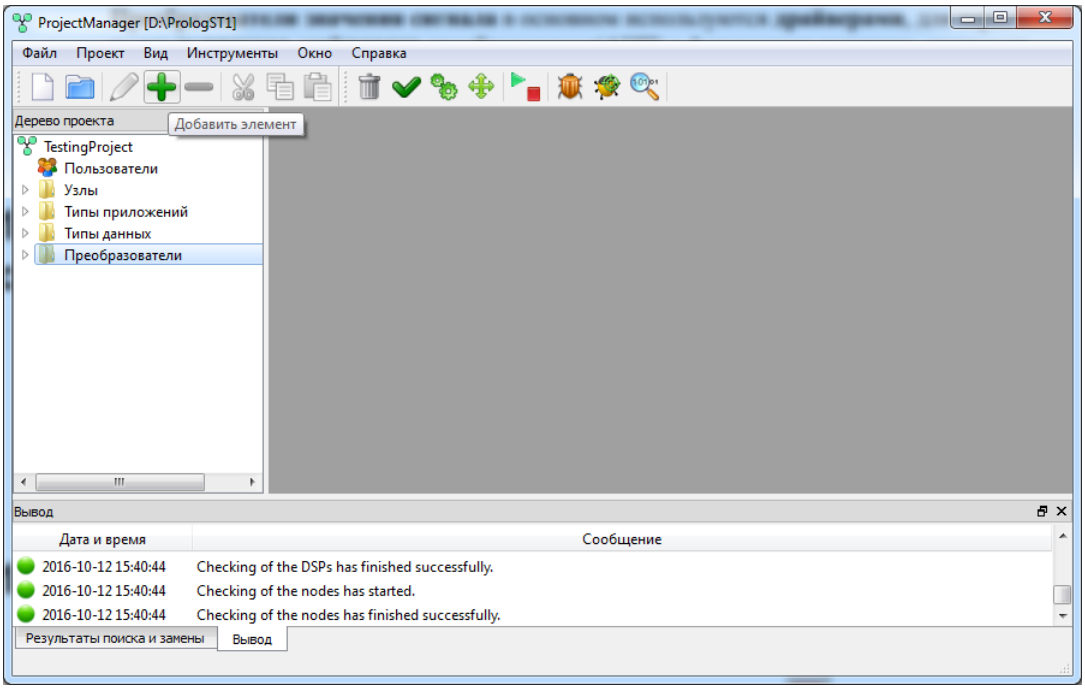


Рисунок 3.45 - Добавление нового преобразователя значения сигнала

3.2.2.6.2. Удаление преобразователя значений сигнала

Для удаления преобразователя нужно в дереве проекта главного окна **ProjectManager** выделить позицию удаляемого преобразователя и нажать кнопку  («Удалить элемент») на главной панели инструментов. Будет выведен диалог подтверждения удаления. При положительном ответе преобразователь будет удалён. Перед удалением рекомендуется убедиться, что данный преобразователь нигде в проекте не используется.

3.2.2.6.3. Редактирование преобразователя значений сигнала

Для редактирования преобразователя нужно в дереве проекта главного окна **ProjectManager** выделить соответствующую позицию и нажать кнопку  («Редактировать элемент») на главной панели инструментов. Откроется диалог редактирования преобразователя (см. рис. 3.46).

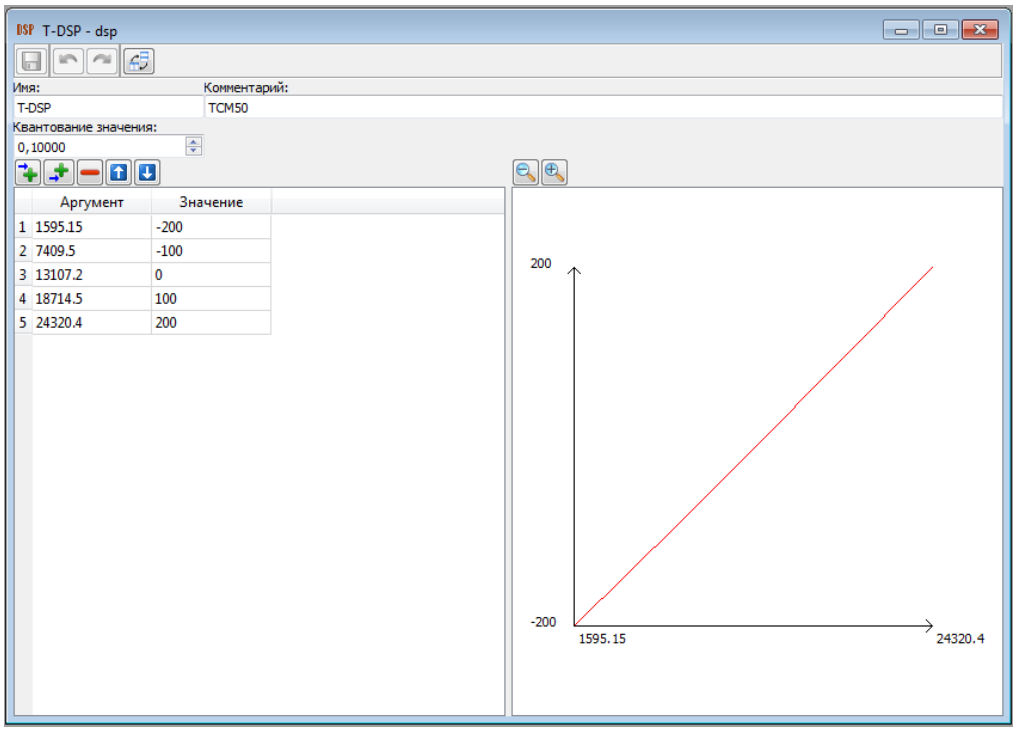


Рисунок 3.46 - Диалог для редактирования преобразователя значений сигнала

Слева вверху в окне диалога редактирования расположены следующие кнопки:

(«Сохранить») – сохранить внесённые изменения;

(«Отменить») – отменить предыдущее действие по редактированию;

(«Повторить») – повторить ранее отменённое действие;

(«Изменить ориентацию») – изменить ориентацию окон.

Ниже располагаются три поля:

- **Имя** – название преобразователя;
- **Комментарий** – опциональное поле для описания преобразователя;
- **Квантование значения** - изменения в данных пределах не будут регистрироваться системой.

Окно диалога разделено на две части. В левой части находится редактируемая таблица, в правой – графическое представление функции преобразования.

Над таблицей расположены кнопки управления:

(«Вставить сверху») – добавить новое значение сверху от выбранной в таблице позиции;

(«Вставить снизу») – добавить новое значение снизу от выбранной в таблице позиции;

(«Удалить») – удалить выбранное в таблице значение;

(«Поднять») – переместить выбранное в таблице значение вверх;

(«Опустить») – переместить выбранное в таблице значение вниз.

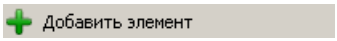
В процессе редактирования нужно задать в табличном представлении монотонную кусочно-линейную функцию. В качестве «аргумента» выступают значения внутри приложения, в качестве «значения» – значения вне приложения.

Над окном графического представления располагаются следующие кнопки:

- уменьшить отображение графического представления;

- увеличить отображение графического представления.

3.2.2.7. Создание и редактирование пользовательских типов данных сигналов

Для создания новых типов данных сигналов (далее в разделе просто тип данных), необходимо выбрать папку Типы данных в дереве проекта программы **ProjectManager** и нажать кнопку  ("Добавить элемент") на панели инструментов или вызвать контекстное меню, нажав правой кнопкой «мыши» на папке Типы данных и далее выбрать в меню 

(см. рис. 3.47).

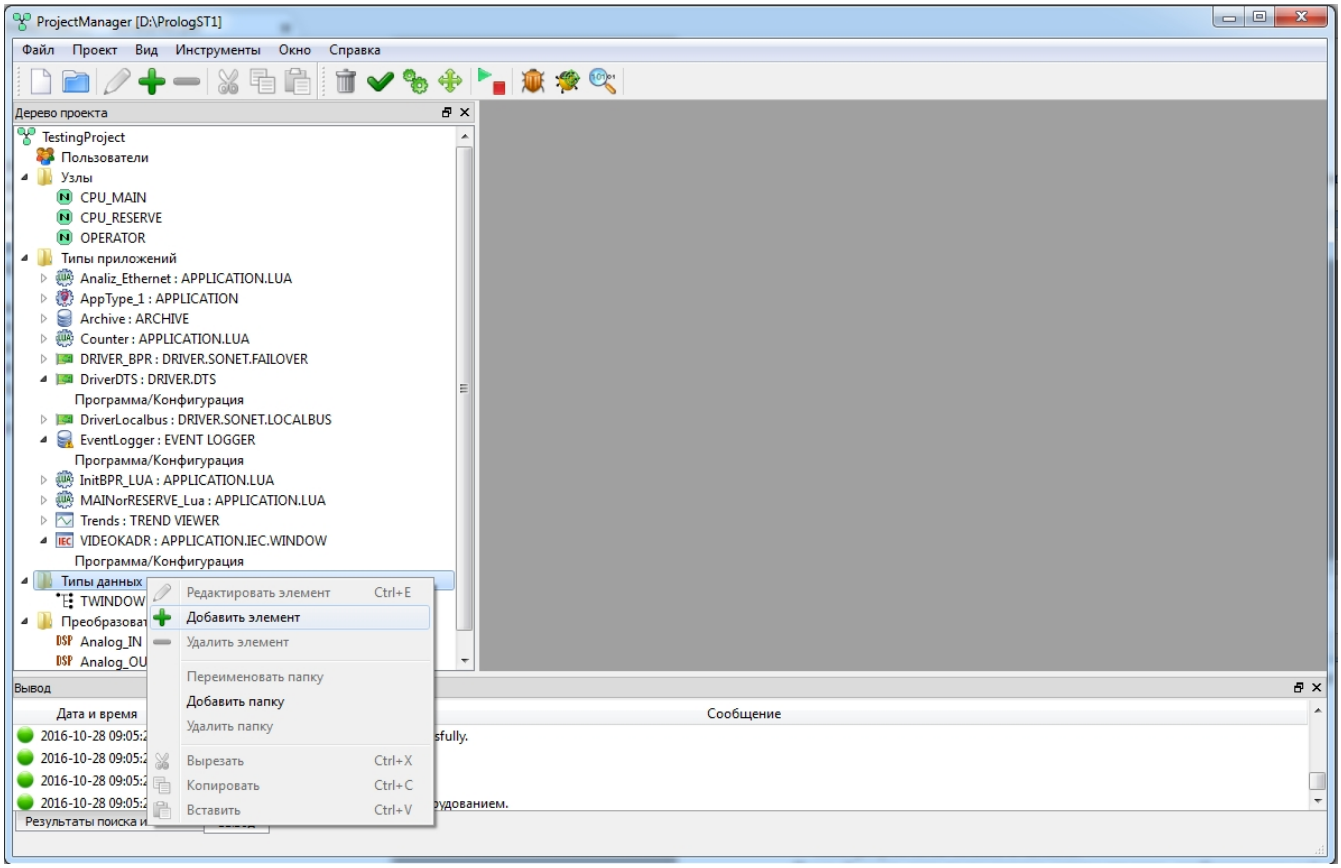
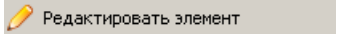


Рисунок 3.47 - Добавление нового типа данных из контекстного меню

Новый тип данных будет иметь имя **Type_N**, где **N** - число от 1 и далее, и будет располагаться внутри папки Типы данных. Для редактирования необходимо выделить строку, с ранее созданным типом данных, и нажать кнопку  ("Редактировать элемент") на панели инструментов или вызвать контекстное меню, нажав правой кнопкой «мыши» на выделенной строке, и выбрать пункт 

(см. рис. 3.48).

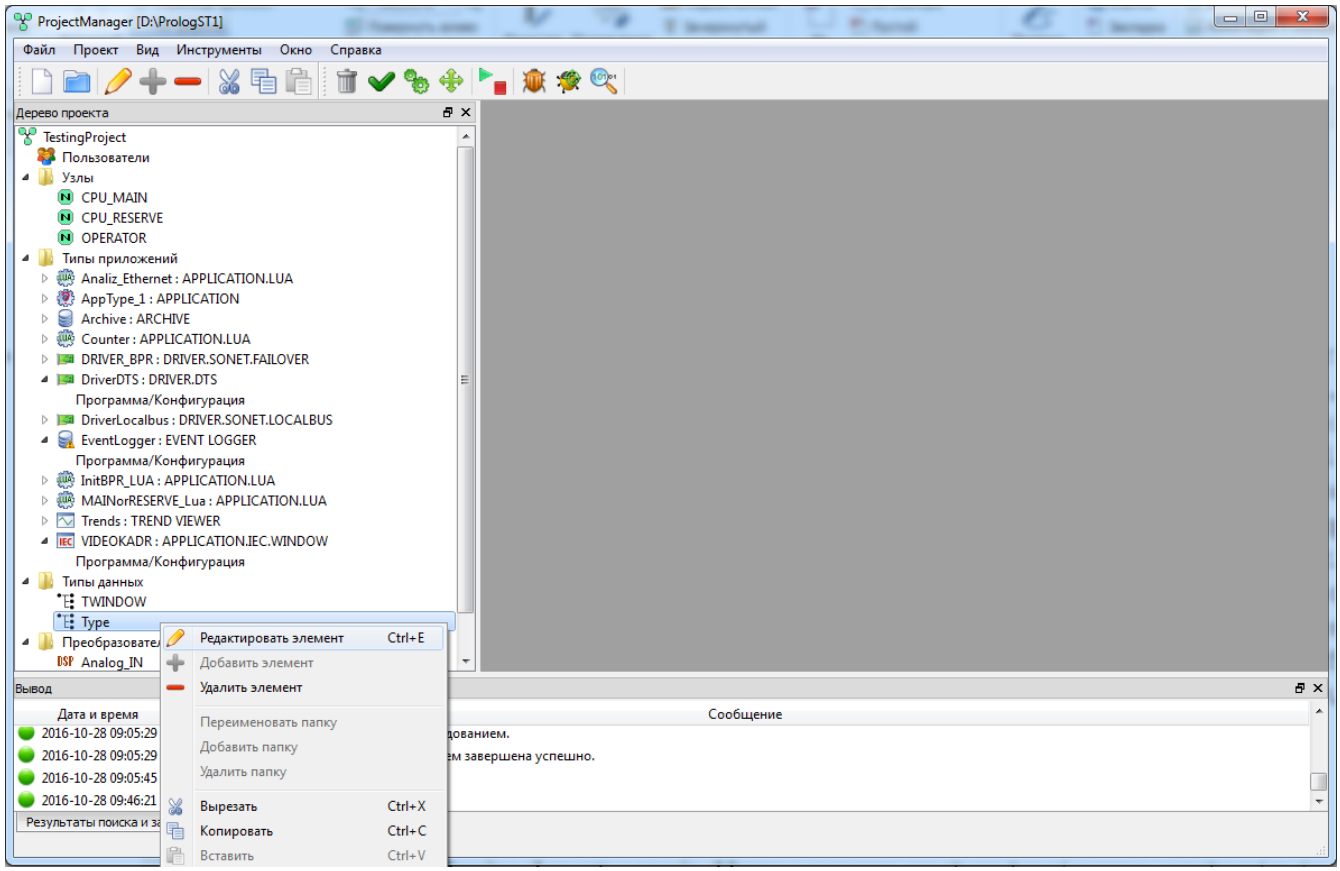


Рисунок 3.48 - Редактирование типа данных из контекстного меню

Для редактирования типа данных откроется специализированный редактор (см. рис. 3.49).

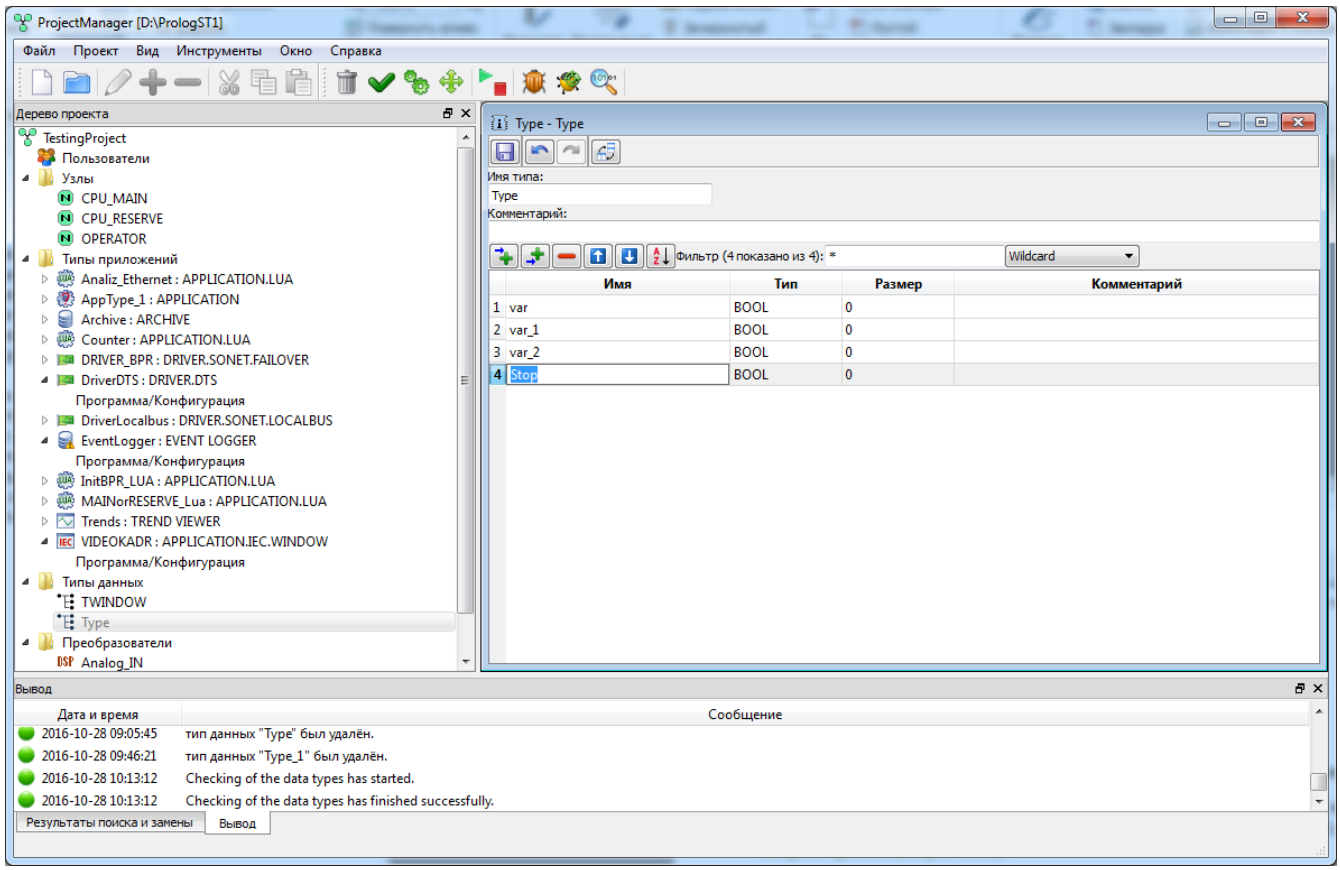


Рисунок 3.49 - Специализированный редактор типов данных

Окно специализированного редактора называется **Type**. Слева вверху данного окна редактора расположены следующие кнопки:

 («Сохранить») – сохранить внесённые изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

 («Изменить ориентацию») – изменить ориентацию окон.

Ниже расположены поля ввода:

- **Имя типа** - имя, которое будет использоваться в дальнейшем, как название данного типа данных (в названии используются только английские буквы);

- **Комментарий** - описание данного типа данных.

Под полями ввода расположены кнопки редактирования таблицы полей типа данных:

 («Вставить сверху») – добавить новое поле сверху от выбранной в таблице позиции;

 («Вставить снизу») – добавить новое поле снизу от выбранной в таблице позиции;

 («Удалить») – удалить выбранное в таблице поле;

 («Поднять») – переместить выбранное в таблице поле вверх;

 («Опустить») – переместить выбранное в таблице поле вниз;

 («Принудительная сортировка») – перегруппировать поля по алфавиту.

Правее кнопок расположены управляющие компоненты для настройки фильтрации отображаемых в таблице полей.

Доступны следующие виды фильтров:

- **Wilcard** – фильтрация по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

Таблица полей типа данных состоит из следующих столбцов:

- **Имя** - имя поля в типе данных (по умолчанию имеют вид var_N, где N - число от 1 и далее)

- **Тип** - тип данных из имеющихся в SCADA-системе "СОНАТА"(см. Приложение А) и ранее созданных пользовательских типов данных;

- **Размер** - размер данных сигнала (0 – скалярный сигнал, >0 – векторный сигнал);

- **Комментарий** - описание данного поля типа данных.

Внесённые изменения в тип данных сохраняются кнопкой  и окно закрывается кнопкой  . Новый тип данных может применятся так же, как и типы данных SCADA-системы "СОНАТА".

3.2.2.8. Редактирование настроек проекта и связей приложений

Для редактирования настроек проекта и связей приложений нужно в дереве проекта главного окна **ProjectManager** выделить корневую позицию с названием проекта и нажать кнопку  («Редактировать элемент») на главной панели инструментов (см. рис. 3.50). Откроется диалог конфигурации проекта (см. рис. 3.51).

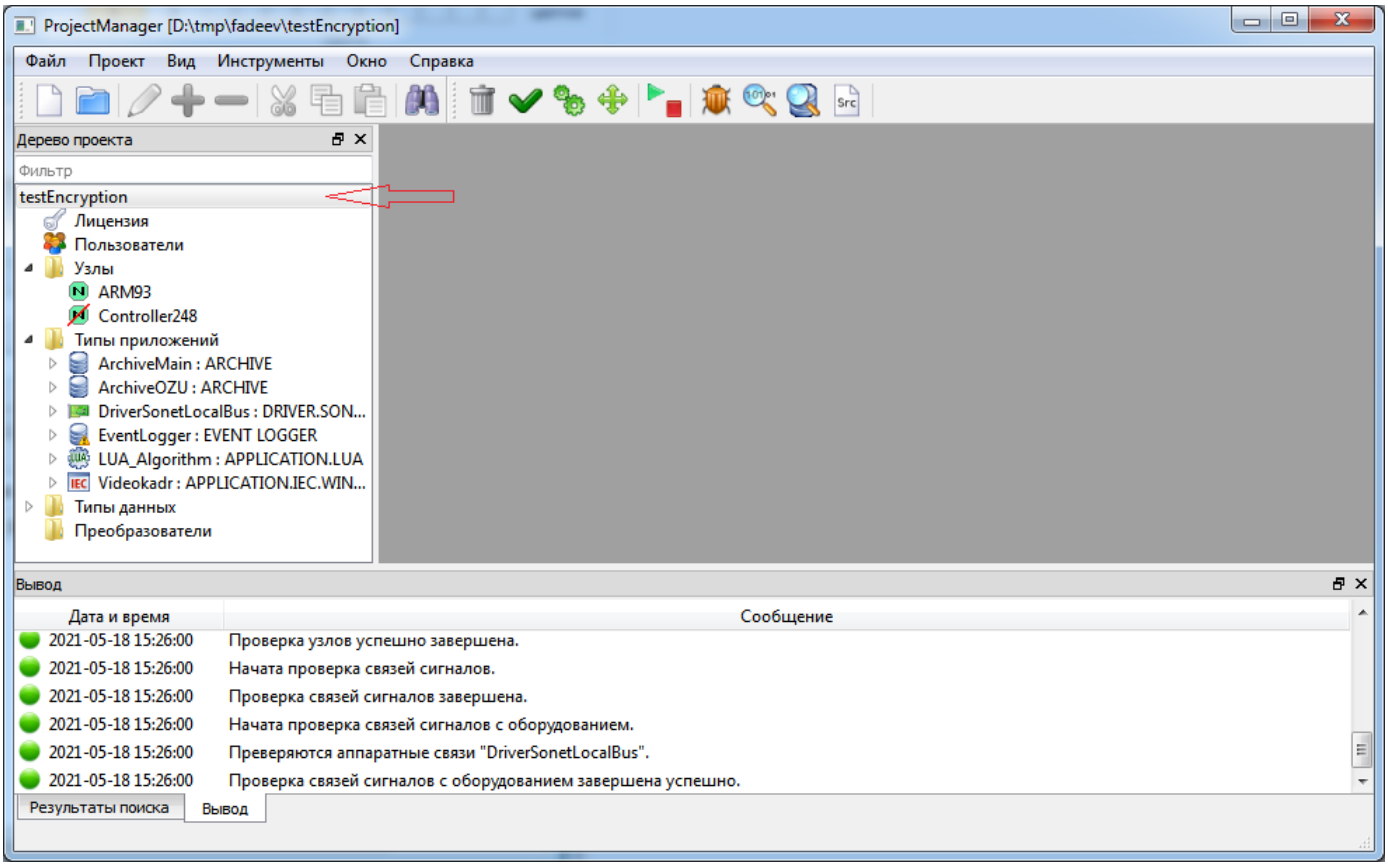


Рисунок 3.50 - Редактирование настроек проекта

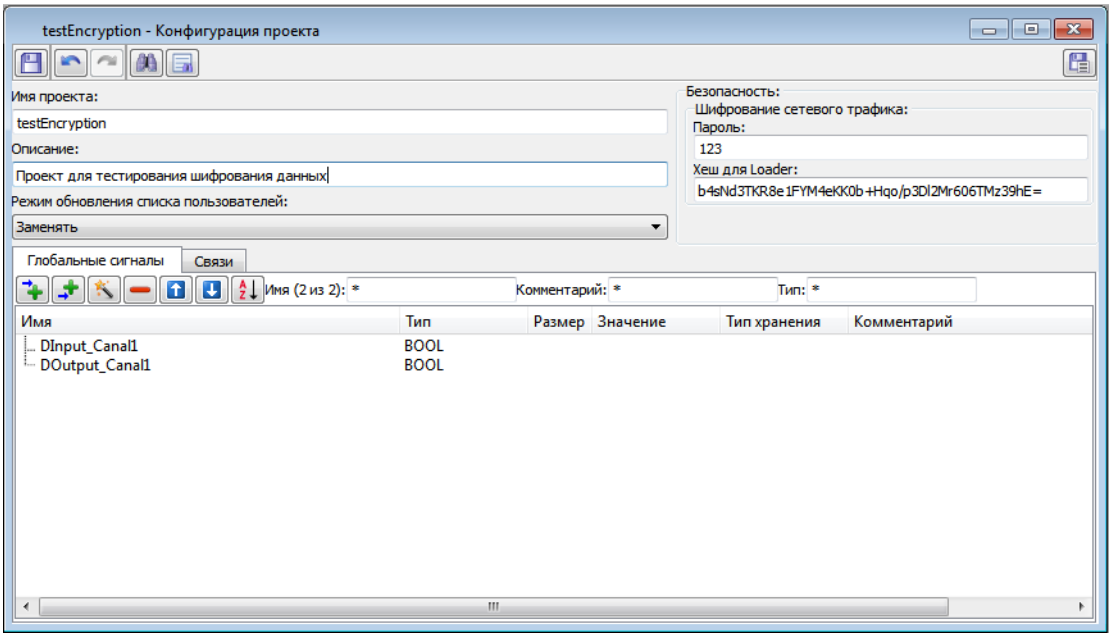


Рисунок 3.51 - Диалог для редактирования конфигурации проекта и связей приложений – вкладка Глобальные сигналы

Вверху окна диалога расположены следующие кнопки:

 («Сохранить») – сохранить внесённые в конфигурацию узла изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

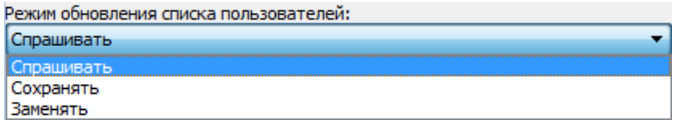
 («Повторить») – повторить ранее отменённое действие;

 ("Поиск") - выполнить поиск;

 ("Показать/скрыть результаты поиска") - позволяет отобразить или скрыть окно результатов поиска;

 ("Сохранить отчёт") - сохранить конфигурацию проекта в файл.

Далее расположены следующие поля ввода:

- **Имя проекта** – название проекта;
- **Описание** – опциональное описание проекта;
- 

- при загрузке проекта на узлы могут быть выбраны 3 варианта работы со списком пользователей:
  - "**Спрашивать**" - выдается сообщение на подтверждение обновлять список пользователей на узлах или не обновлять;
  - "**Сохранять**" - список пользователей на узлах не изменяется, т.е. сохраняется неизменным;
  - "**Заменять**" - список пользователей заменяется на список из загружаемого проекта;

Справа расположены настройки Безопасности:

- **Шифрование сетевого трафика** - если сделать данные настройки, то сетевые пакеты проекта SCADA "СОНАТА" будут передаваться в зашифрованном виде. Для этого выполняются следующие настройки:
  - **Пароль** - пароль проекта, используемый для шифрования данных. Данный пароль автоматически применяется для шифрования данных в проекте, но так же может понадобиться для подключения приложений ControlCenter, SignalViewer и ArchiveViewer к проекту, использующему шифрование данных;
  - **Хеш для Loader** - хеш от пароля проекта (нужен для применения в аргументах приложения Loader).

Ниже расположены две вкладки:

- **Глобальные сигналы** – настройки глобальных сигналов для всего проекта;
- **Связи** – настройка всех связей приложений для всего проекта (на этой вкладке можно увидеть все сигналы (глобальные и локальные) проекта и их связи).

Под названиями вкладок расположены кнопки редактирования.

Далее идут управляющие компоненты для настройки фильтрации отображаемых в таблице элементов интерфейса.

3.2.2.8.1. Вкладка Глобальные сигналы

Во вкладке **Глобальные сигналы** (см. рис. 3.51) расположены следующие кнопки:

 («Вставить сверху») – добавить глобальный сигнал сверху от выбранной в таблице позиции;

 («Вставить снизу») – добавить глобальный сигнал снизу от выбранной в таблице позиции;

 ("Вставить снизу") - добавить группу глобальных сигналов снизу от выбранной в таблице позиции (по нажатию на данную кнопку откроется диалог группового добавления глобальных сигналов (см. рис. 3.52));

 («Удалить») – удалить выбранный в таблице глобальный сигнал;

 («Поднять») – переместить выбранный в таблице глобальный сигнал вверх;

(«Опустить») – переместить выбранный в таблице глобальный сигнал вниз;

(«Принудительная сортировка») – перегруппировать глобальные сигналы по алфавиту.

Доступны следующие виды фильтров:

Wilcard – фильтрация по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

Под вкладками расположена таблица глобальных сигналов проекта.

У каждого глобального сигнала есть следующие настраиваемые свойства:

Имя – уникальное имя глобального сигнала (допустимыми символами в имени являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя глобального сигнала не может начинаться с цифры);

Тип – тип данных глобального сигнала;

Размер – размер данных глобального сигнала (0 – скалярный сигнал, >0 – векторный сигнал);

Значения – начальное значение глобального сигнала;

Тип хранения – тип хранения значения глобального сигнала (доступные опции: **NORMAL** – при перезагрузке системы последнее значение глобального сигнала не сохраняется, глобальный сигнал инициализируется начальным значением; **RETAIN** – при перезагрузке системы сохраняется последнее значение глобального сигнала, "Холодный старт" сбрасывает значения **RETAIN** глобальных сигналов на начальное значение; **PERSISTENT** – глобальные сигналы всегда сохраняют своё последнее значение);

Комментарий – опциональное справочное описание сигнала;

Мета - это строка, в которую можно внести дополнительную информацию и в дальнейшем использовать её в Lua алгоритмах. Данную информацию никто не будет видеть. В Lua алгоритмах данную информацию извлекает команда Core.getSignalMeta;

Для добавления, удаления и перемещения глобальных сигналов используются соответствующие кнопки, расположенные над таблицей. Для редактирования ячеек в таблице используется двойной щелчок «мышью».

Для сохранения внесенных изменений служит кнопка («Сохранить») – при нажатии кнопки редактор сохранит внесённые изменения.

По завершении редактирования таблицы глобальных сигналов диалог нужно закрыть крестиком в верхнем правом углу окна редактирования – программа предложит сохранить внесённые изменения. При положительном ответе данные будут сохранены.

По нажатию на кнопку откроется окно диалога группового добавления сигналов (см. рис. 3.52).

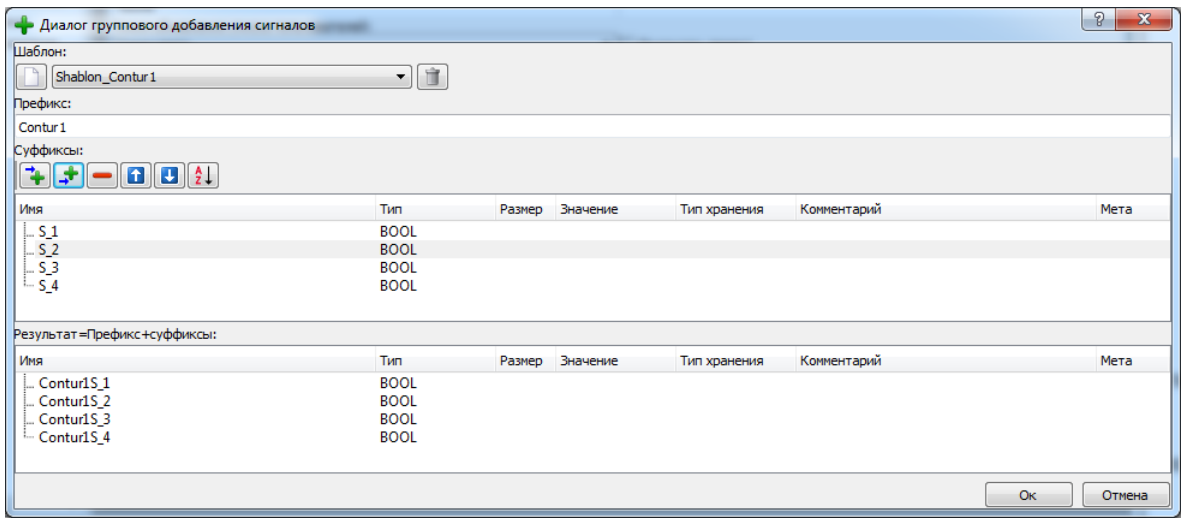


Рисунок 3.52 - Окно диалога группового добавления сигналов

Описание диалога группового добавления сигналов:

 ("Новый шаблон") - создается новый шаблон;

Префикс - префикс для всех создаваемых в шаблоне глобальных сигналов;

Суффиксы - добавляем суффиксы - вторые части имен глобальных сигналов (функции кнопок управления аналогичны описанным выше в данном разделе);

Результат=Префикс+суффиксы - отображаются результирующие имена глобальных сигналов, которые будут добавлены.

По нажатию на кнопку  будет произведено добавление группы глобальных сигналов, по нажатию на кнопку  будет закрыт диалог, а глобальные сигналы добавлены не будут.

3.2.2.8.2. Вкладка Связи

Во вкладке **Связи**(см. рис. 3.53) расположены следующие кнопки:

 («Добавить связь») – добавить новую связь приложений;

 («Удалить связь») – удалить выбранную связь приложений.

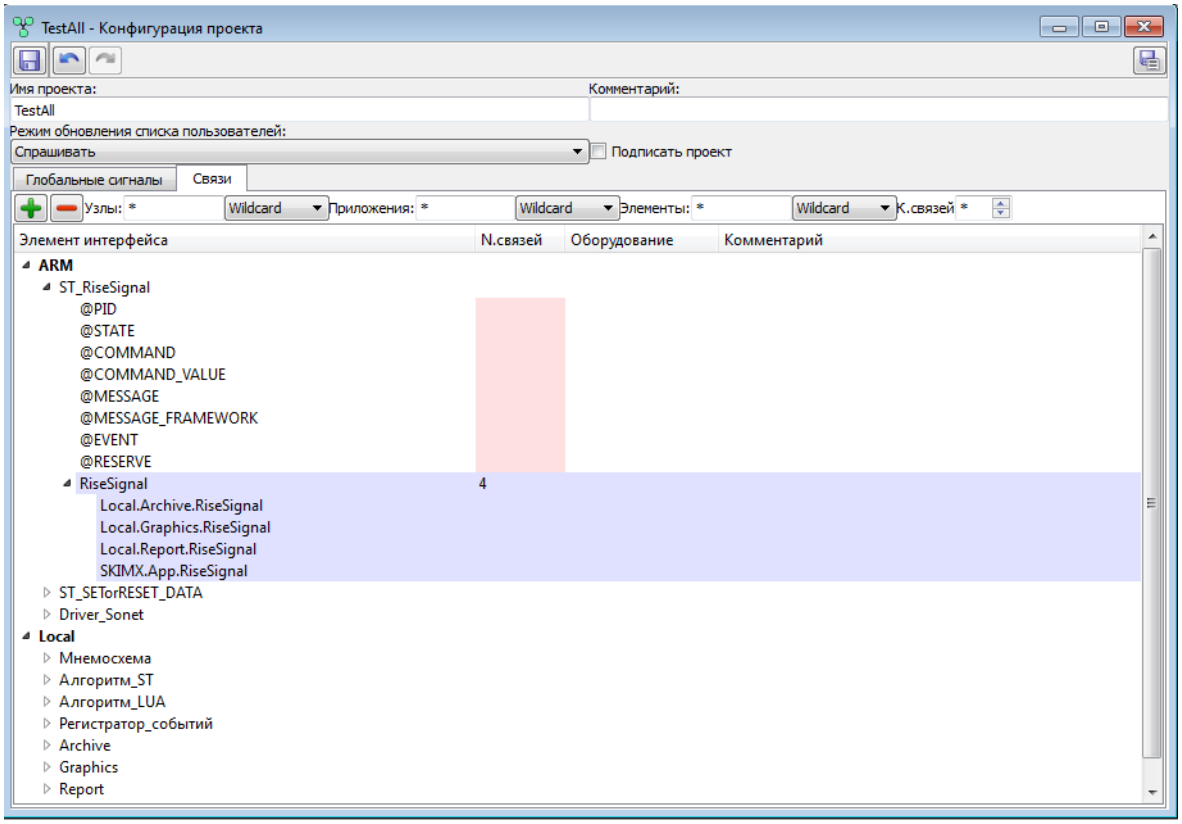


Рисунок 3.53 - Диалог для редактирования настроек проекта и связей приложений – вкладка Связи

Доступные поля для фильтрации:

- **Узлы** - фильтр по именам узлов;
- **Приложения** - фильтр по именам приложений;
- **Элементы** – фильтр по именам сигналов с привязкой к узлу и приложению;
- **К. связей** – фильтр по количеству связей, использующих один сигнал приложений.

Для полей фильтрации **Узлы**, **Приложения,** **Элементы** доступны следующие виды фильтров:

- **Wilcard** – фильтрация по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

Для поля фильтрации **К. связей** доступны следующие виды фильтров:

- * – любое число связей;

- **n** – какое - либо целое число (0, 1, 2 и т.д.).

Под вкладками расположена таблица связей проекта.

Для каждой связи в таблице отображаются следующие данные:

- **Элемент интерфейса** – список сигналов, представленный в виде «дерева»: **узел => приложение => сигнал => использующие сигнал приложения**;

- **N. связей** – количество связей для каждого сигнала;

- **Оборудование** – по какому адресу доступен конкретный сигнал с привязкой к оборудованию (номер контроллера, адрес модуля, номер клеммы);

- **Комментарий** – опциональное описание сигнала.

Для создания новой связи требуется нажать на кнопку  («Добавить связь»). При этом откроется окно **Добавить связь** (см. рис. 3.54). Данное окно состоит из двух частей: **Начало** и **Конец**. Эти части описывают два сигнала, которые будут связаны друг с другом.

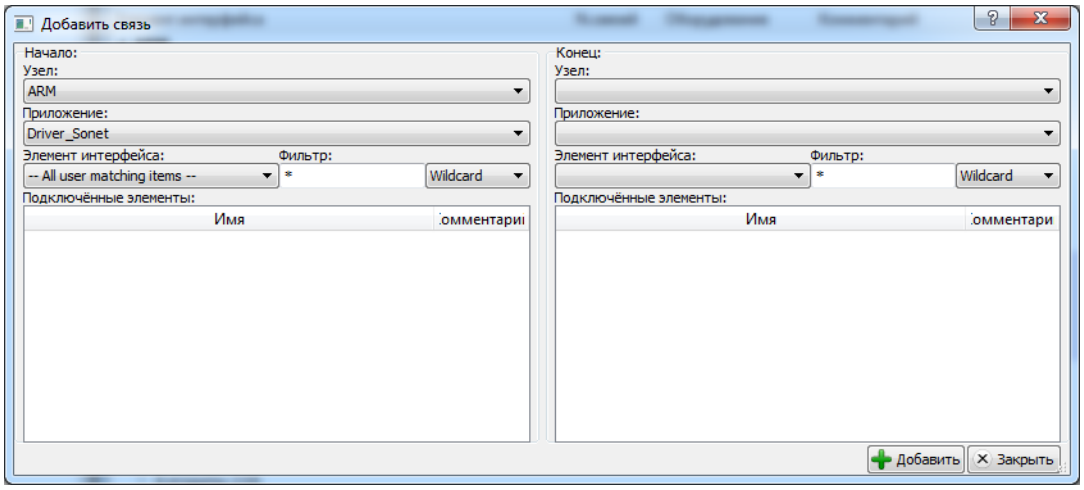


Рисунок 3.54 - Диалог для создания новой связи приложений

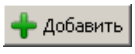
Для добавления связи необходимо настроить следующие параметры (для раздела **Начало** и **Конец** параметры имеют аналогичное предназначение):

- **Узел** - узел, на котором работает приложение с данным сигналом;

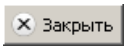
- **Приложение** - приложение, в интерфейсе которого находится данный сигнал;

- **Элемент интерфейса** - сигнал, для которого настраивается связь;

- **Фильтр** - строка фильтрации по шаблону для поля **Элемент интерфейса**, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора.

После указания всех параметров следует нажать на кнопку 

в правом углу окна редактирования.

Для отказа от редактирования следует нажать на кнопку 

в правом углу окна редактирования.

Для удаления связи необходимо выбрать ее в таблице на вкладке **Связи** и нажать кнопку  («Удалить связь»).

3.2.2.9. Работа с лицензией

В SCADA-систему "Соната", начиная с версии 1.4, включена лицензионная защита с привязкой к оборудованию узла. Под привязкой подразумевается, что в файле лицензионного ключа указаны конфигурации оборудования узлов лицензируемого проекта.

Процесс получения лицензионного ключа состоит из следующих этапов:

- формирование файла конфигурации проекта;

- отправка файла конфигурации в отдел разработки SCADA системы по адресу sonata-sale@ezan.ac.ru;

- получение лицензионного ключа и его добавление в проект.

3.2.2.9.1. Формирование файла конфигурации проекта

Для того чтобы сформировать файл конфигурации проекта, необходимо запустить программу ProjectManager, открыть проект и в дереве проекта (см. рис. 3.55) выбрать пункт Лицензия .

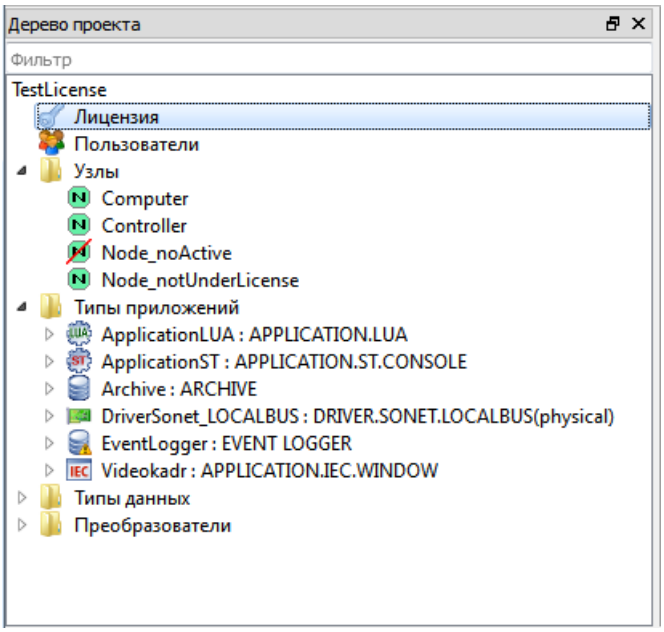


Рисунок 3.55 - Дерево проекта

Далее откроется диалоговое окно лицензии (см. рис. 3.56).

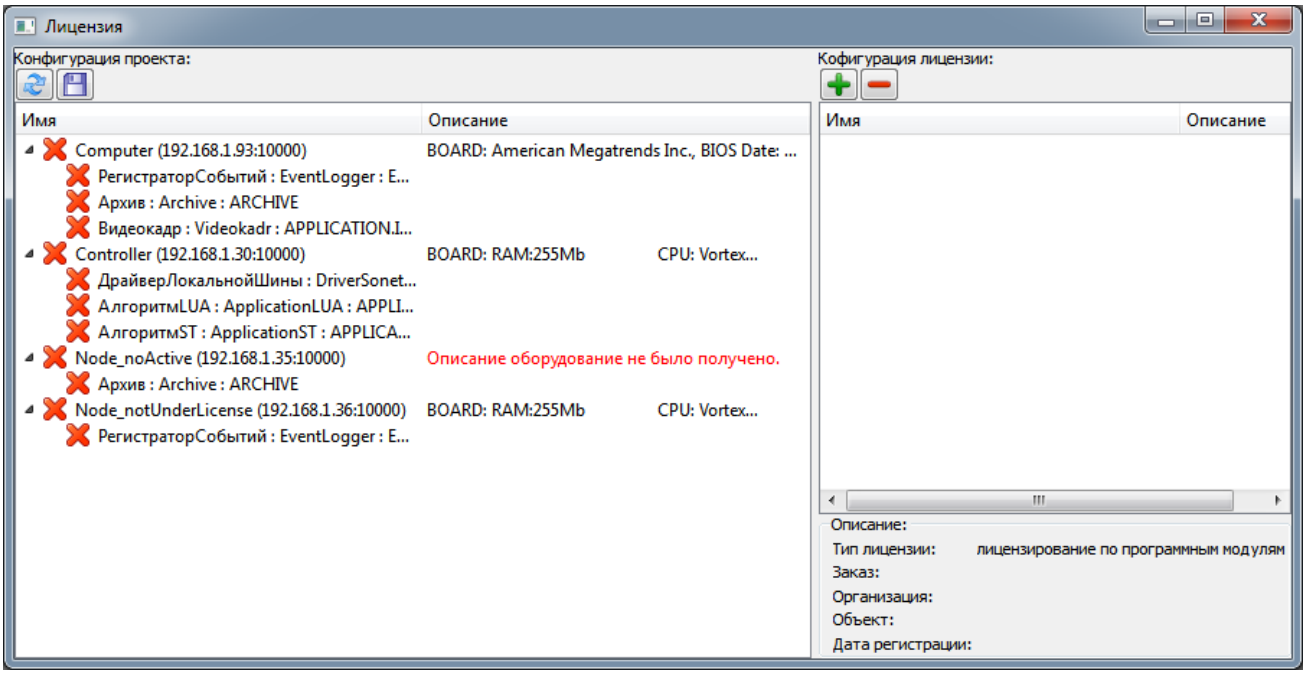


Рисунок 3.56 - Диалоговое окно лицензии

В левой части окна располагается дерево конфигурации проекта, а в правой части - дерево конфигурации лицензии.

В дереве конфигурации проекта отображена иерархия узлов и приложений проекта. Напротив приложений-драйверов и некоторых других приложений находится описание количества лицензируемых связей. В случае приложений-драйверов связями являются соединения сигналов системы с физическими каналами устройств.

Напротив каждого узла должна находиться конфигурция его оборудования. Если конфигурация оборудования не получена, то в поле Описание будет сообщение, что описание оборудования не было получено. Отсутствие конфигурации оборудования одного из узлов свидетельствует об отсутствии связи с этим узлом или о том, что на данном узле не запущено приложение Loader, необходимое для работы узла в составе проекта и для данной операции. Для получения описания оборудования необходимо восстановить связь с узлом, затем перезапустить диалоговое окно лицензии или нажать на кнопку  (Получить конфигурацию оборудования).

Для того чтобы сохранить конфигурацию проекта, необходимо кликнуть по кнопке  (Сохранить конфигурацию проекта). Файл конфигурации проекта сохраняется в формате XML.

3.2.2.9.2. Подключение лицензионного ключа.

Лицензионный ключ представляет собой файл с описанием узлов и продуктов. Каждый продукт предназначен для лицензирования одного или нескольких приложений.

Продукты имеют следующие параметры:

- Всего приложений - количество приложений, которые может лицензировать данный продукт.
- Используется - количество приложений, которые данный продукт лицензирует в данный момент.
- Макс. Связей - максимальное количество связей приложения для возможности его лицензирования данным продуктом.

Приложение может быть лицензировано продуктом, если продукт поддерживает вид этого приложения и количество лицензируемых связей этого приложения не больше максимального количества связей продукта.

После того как лицензионный ключ получен, необходимо привязать его к проекту. Для этого надо запустить программу ProjectManager, открыть проект и в дереве проекта выбрарь пункт Лицензия . Далее в диалоге лицензии необходмо кликнуть на кнопку (Добавить лицензию). Данная кнопка привязывает лицензионный ключ к проекту. Если на момент привязки лицензионного ключа у проекта уже есть ключ, то старый ключ будет отвязан. Кнопка (Удалить лицензию) отвязывает лицензионный ключ от проекта.

После подключения лицензии в дереве лицензии отобразится конфигурация лицензии (см. рис. 3.57), которая будет сопоставлена с конфигурацией проекта.

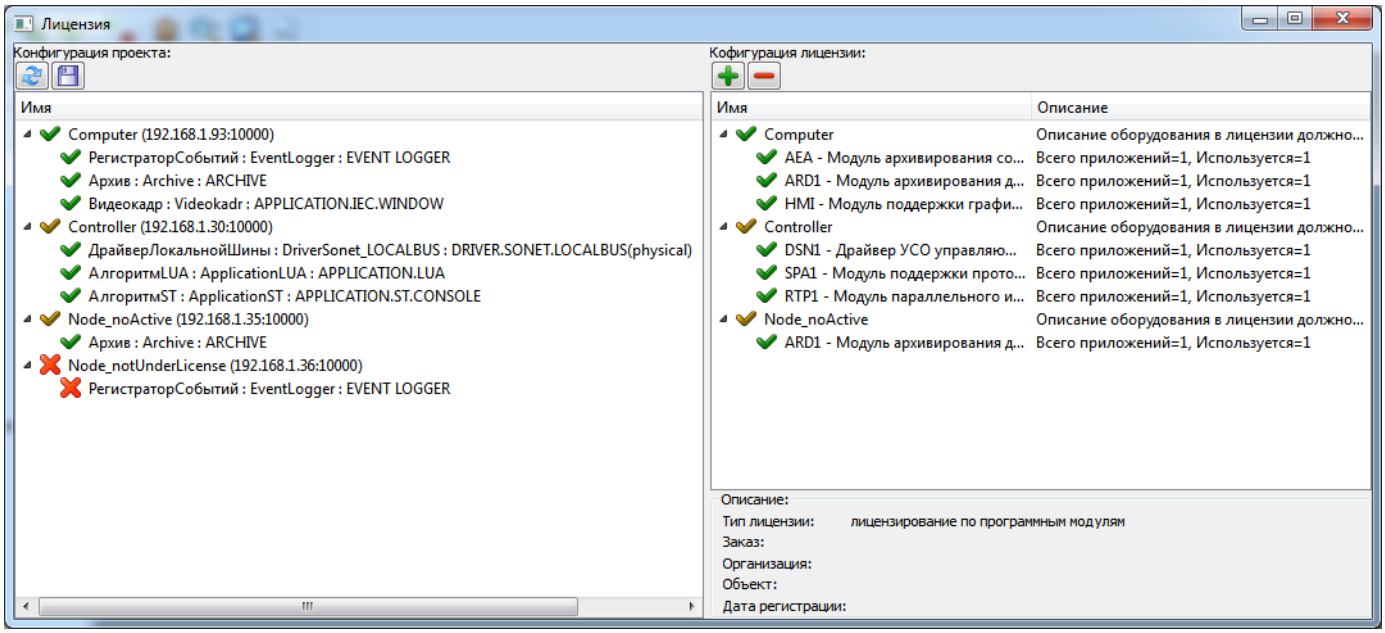


Рисунок 3.57 - Конфигурация лицензии

В результате сопоставления конфигурации проекта и конфигурации лицензии перед описаниями узлов и приложений появятся специальные маркеры:

- ✔ - описание данного узла или приложения в конфигурации проекта полностью соответствует описанию узла или продукта в лицензии;

- ✔ - описание данного узла в конфигурации проекта не полностью соответствует описанию узла лицензии. В данном случае узел будет считаться нелицензированным. О несоответсвиях сообщит всплывающая подсказка;

- ✘ - данный узел или приложение отсутствует в лицензии;

- ? - данный продукт в лицензии не соответсвует ни одному из приложений в конфигурации узла.

В проекте на рис. 3.57 лицензированным считается только узел Computer. У узлов Controller и Node_noActive не совпадает конфигурация оборудования в проекте и в лицензии. Узел Node_notUnderLicense в лицензии не найден.

3.2.2.9.3. Особенности работы с лицензионным ключом.

Лицензия распространяется только на средства исполнения. На средства разработки лицензия не распространяется.

В случае запуска проекта без лицензии, в лог-файлах приложений выводится сообщения об отсутствии лицензии. Если в проекте есть приложение вида APPLICATION.IEC.WINDOW, то его окно регистрации будет иметь заголовок "Запущено без лицензии", а само окно будет окнашено в красный цвет. (см. рис. 3.58). При этом данное приложение с интервалом в 30 минут будет совершать завершение сессии с текущим пользователем и требовать повторной регистрации.

Так же, если в проекте используется окно отображения списка событий системы (EventViewer), то при отсутствии корректной лицензии в данном окне каждые 30 минут будет выводиться событие с сообщением об отсутствии лицензии. Данное событие не смогут убрать фильтры, наложенные на данное окно.

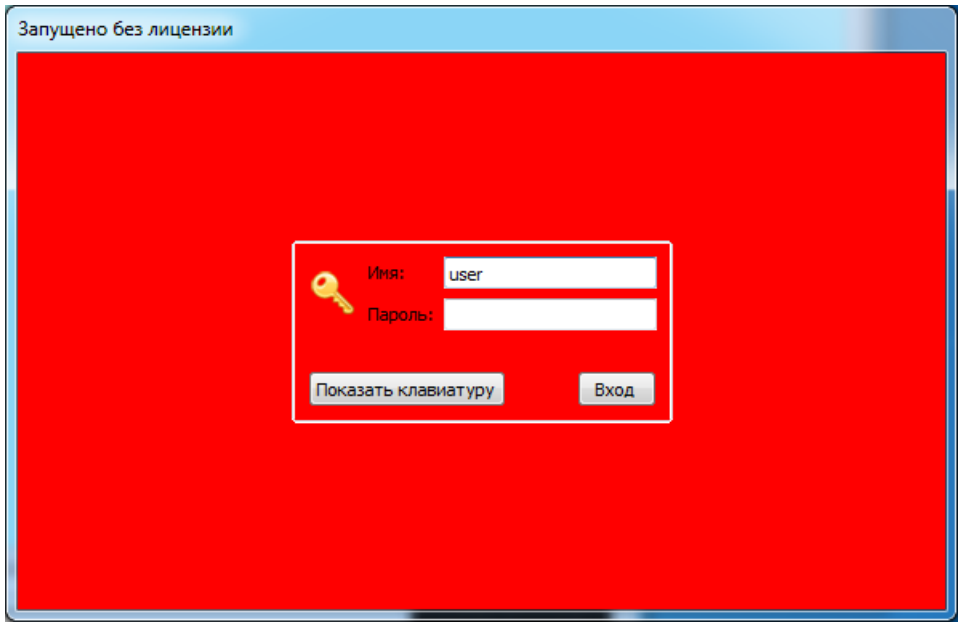


Рисунок 3.58 - Окно регистрации пользователя при запуске приложения-видеокадра без лицензии

При запуске проекта с лицензией в лог-файлах приложений выводиться сообщение об успешной загрузке лицензии.

В случае изменения конфигурации проекта (замена оборудования, добавление или удаление узлов, приложений) при уже имеющемся лицензионном ключе, необходимо послать письмо с описанием возникшей ситуации в отдел разработки по адресу sonata-info@ezan.ac.ru. К письму необходимо прикрепить файл новой конфигурации проекта.

3.2.2.9.4. Дополнительные пояснения к процессу получения лицензии

При начальной разработке проекта далеко не всегда есть всё оборудование для узлов, которые будут в нём участвовать. Т.е. получить файл конфигурации с корректной лицензией нет возможности.

В данном случае процесс получения лицензии выглядит следующим образом:

- **Декларация о намерениях**. Вы высылаете файл конфигурации проекта без описания оборудования (см. раздел 3.2.2.9.1). В этом файле должна присутствовать информация о всех узлах

проекта и какие приложения на них будут работать. Мы регистрируем у себя перечень узлов и их приложений. Отправляем вам лицензию без описания оборудования. Такая лицензия работать не будет, но она покажет, что все необходимые позиции (модули Сонаты) заказаны. На основе этой информации может быть изготовлен бумажный вариант лицензии для бухгалтерии и прочей отчётности.

- **Получение рабочей лицензии**. Уже на вводимом в эксплуатацию объекте вы повторно запускаете диалог лицензирования в ProjectManager. Он собирает информацию об оборудовании. Вы повторно высылаете файл конфигурации проекта нам. Мы выпускаем лицензию с этим описанием оборудования, которая в данном случае будет у вас корректно работать.

3.2.2.10. Редактирование списка пользователей

Внимание! Принципы работы со списком пользователей описаны в документе SCADA-система "СОНАТА" Описание применения КУНИ.505200.023-01.01 31 в п. Функция создания, редактирования и контроля учетных записей пользователей.

Для создания и редактирования списка пользователей АСУ ТП используется программа **UserListEditor**. Данная программа доступна как на этапе разработки проекта системы (описано в данном разделе), так и на этапе его эксплуатации (смотрите SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33 функциональный блок USERS_EDITOR).

На этапе разработки проекта программа **UserListEditor** запускается автоматически программой **ProjectManager** при нажатии кнопки **"Edit / Редактировать"** на позиции **"Пользователи"** в дереве проекта (см. рис. 3.59). После нажатия откроется диалог редактирования списка пользователей и групп пользователей (см. рис. 3.60)

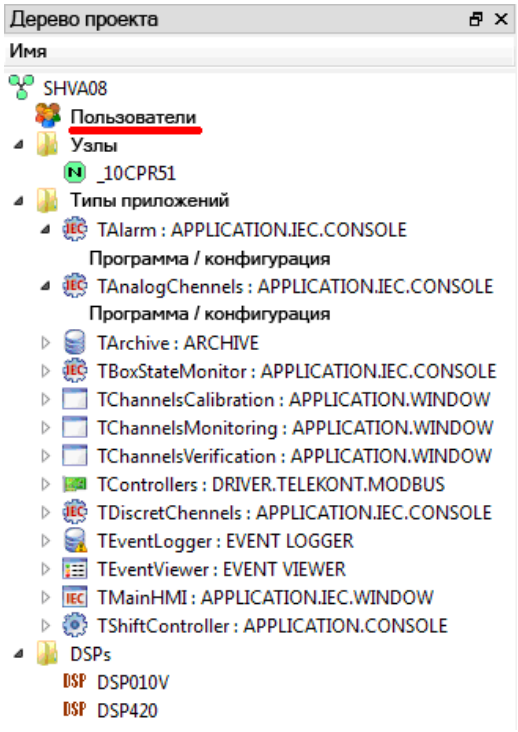


Рисунок 3.59 - Дерево проекта в программе ProjectManager

На рис. 3.60 представлено основное окно программы **UserListEditor**.

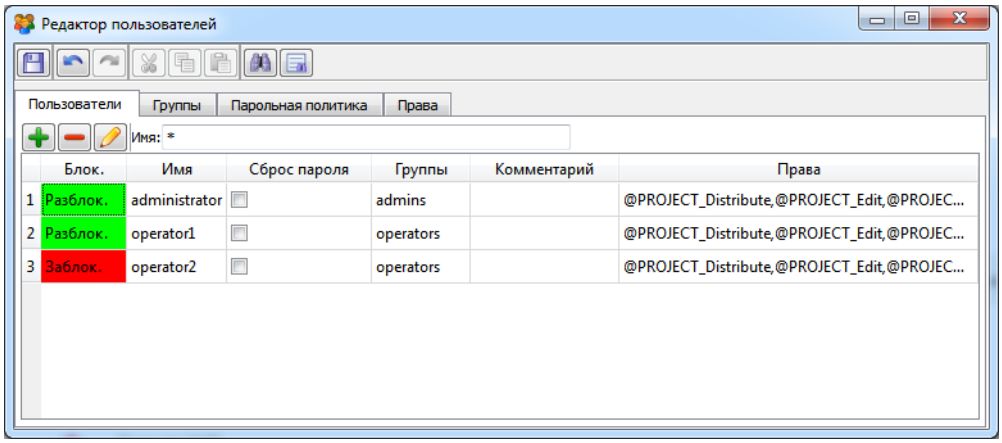


Рисунок 3.60 - Диалог для редактирования списка пользователей и групп

Вверху окна диалога расположены следующие кнопки управления:

(**"Сохранить"**) - сохранить внесённые в права пользователя изменения;

(**"Отменить"**) - отменить предыдущее действие по редактированию;

(**"Повторить"**) - повторить ранее отменённое действие;

(**"Вырезать"**) - вырезать данные и сохранить в буфер;

(**"Копировать"**) - копировать данные в буфер;

(**"Вставить"**) - вставить данные из буфера;

(**"Поиск"**) - поиск вхождений текста в редаторе;

(**"Показать/скрыть результаты поиска"**) - показывает или скрывает окно результатов поиска.

Ниже расположены 4 вкладки:

- **Пользователи** - вкладка с настройками для пользователей системы;
- **Группы** - вкладка с настройками для групп пользователей;
- **Парольная политика** - вкладка с настройками для парольной политики;
- **Права** - вкладка, где можно создать пользовательские права на систему.

3.2.2.10.1. Вкладка Пользователи

Во вкладке Пользователи можно создавать, удалять и редактировать настройки пользователей системы (см. рис. 3.60).

В верхней части окна пользователей расположены кнопки управления списком пользователей:

(**"Добавить"**) - добавить пользователя;

(**"Удалить"**) - удалить выбранного пользователя;

(**"Редактировать"**) - выполнить настройки прав и др. свойств у пользователя (настройка временных интервалов работы пользователя и т.д.).

Ниже располагаются следующие информационные данные:

- **"Блок."** - в данной колонке отображается состояние блокировки пользователя. Есть два состояния: Разблок. (разблокирован) - окрашивается зеленым фоном, Заблок. (заблокирован) окрашивается красным фоном;
- **"Имя"** – имя пользователя;

- **"Сброс пароля"** - отображает информацию, что для данного пользователя выставлено требование поменять пароль при следующем входе;
- **"Группы"** - группа, в которой состоит пользователь;
- **"Комментарии"** - опциональное описание пользователя;
- **"Права"** - права, которыми обладает данный пользователь (отображаются только права пользователя без учета прав группы, в которую он входит).

По нажатию на кнопку (**"Редактировать"**) откроется окно настроек пользователя системы (см. рис. 3.61).

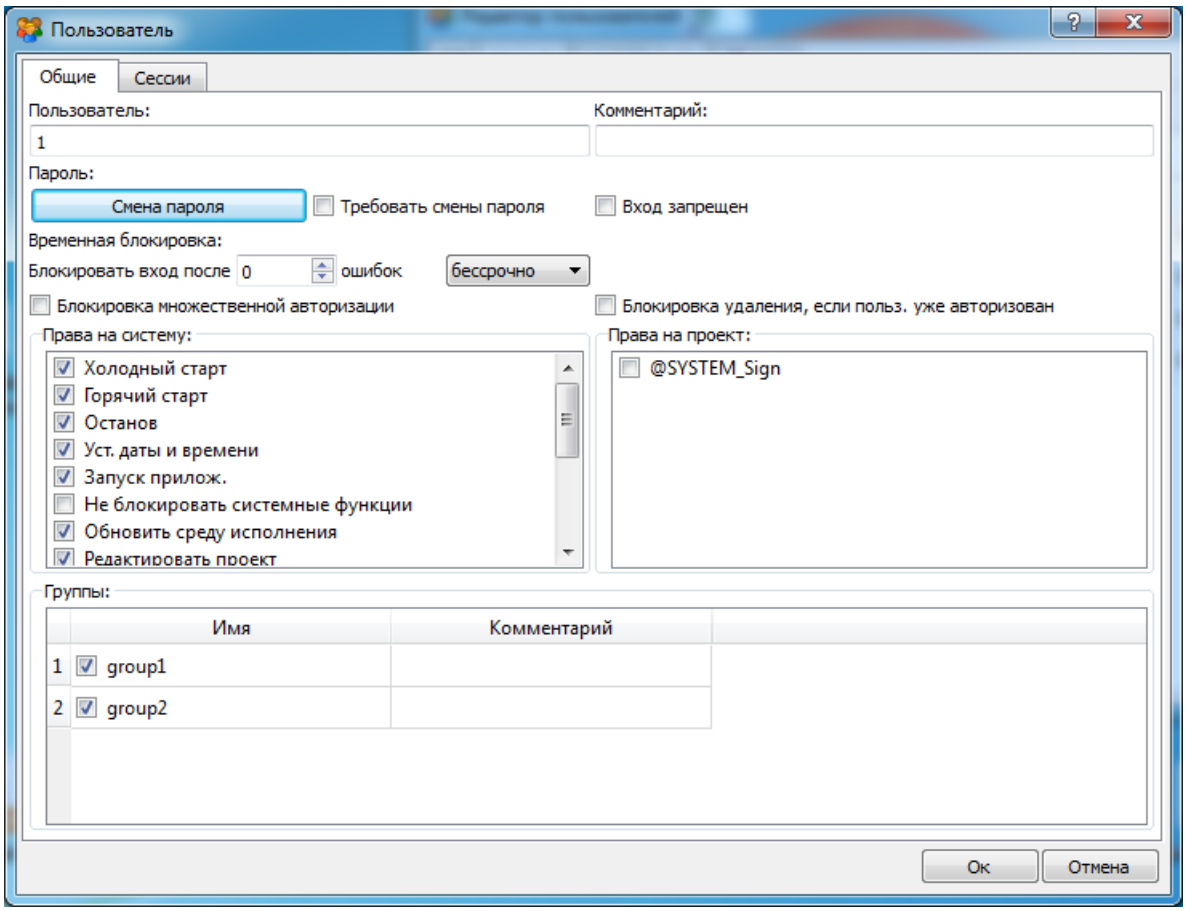


Рисунок 3.61 - Окно настроек пользователя системы

Окно настроек пользователя системы состоит из двух вкладок:
- вкладка **Общие** - общие настройки для пользователя;
- вкладка **Сессии** - настройка рабочих сессий пользователя.

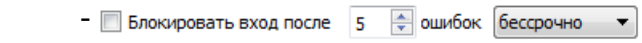
3.2.2.10.1.1. Вкладка Общие

Во вкладке Общие располагаются настройки имени пользователя, пароля и парольной политики, прав пользователя на работу с системой и принадлежность к группе пользователей.

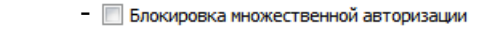
В верхней части располагаются следующие настройки:
- поле ввода **Пользователь** - поле для ввода имени пользователя;
- поле ввода **Комментарий** - поле для ввода комментария о пользователе;
- кнопка **Смена пароля** - откроется окно для смены текущего пароля пользователя (первоначальный пароль пустой);

- флаговая кнопка **Требовать смены пароля** - активирует функцию смены пароля пользователя. У пользователя после ввода его текущего пароля появится окно для смены пароля пользователя;

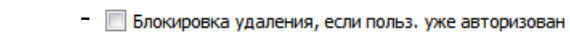
- флаговая кнопка **Вход запрещен** - в активном состоянии запрещает пользователю вход в систему;

- 

- в активном состоянии блокирует вход пользователя в систему после указанного количества неверных попыток ввода пароля на указанный интервал времени (при правильном вводе пароля количество неверных попыток сбросится);

- 

- в активном состоянии не позволяет под данным пользователем входить на нескольких рабочих местах;

- 

- не даёт удалить пользователя из списка пользователей, если в данный момент пользователь активен.

В средней части располагаются настройки прав пользователя. Они разделяются на:

- **Права на систему**:

- **Холодный старт** - пользователь может выполнять холодный старт проекта (запуск проекта со сбросом значений сигналов с признаком **RETAIN**);

- **Горячий старт** - пользователь может выполнять горячий старт проекта (запуск проекта без сброса значений сигналов с признаком **RETAIN**);

- **Останов** - пользователь может выполнить остановку проекта;

- **Уст. даты и времени** - установка системной даты и времени;

- **Запуск прилож.** - запуск внешних приложений;

- **Не блокировать системные функции** (относится к ОС Windows) - в активном состоянии не блокирует различные сочетания клавиш, которые сворачивают окно, вызывают диспетчер задач и т.д. В неактивном состоянии блокирует сочетания клавиш, которые могут позволить пользователю получить доступ к файловой системе или свернуть рабочее окно, если этого нельзя делать;

- **Обновить среду исполнения** - пользователь сможет обновлять среду исполнения с помощью кнопки **Обновить**, которая находится в приложении Центр управления;

- **Редактировать проект** - право на редактирование проекта (данный функционал еще не реализован);

- **Разослать проект** - право разослать проект на узлы (данный функционал еще не реализован);

- **Редактировать пользователей** - сможет редактировать пользователей с помощью кнопки **Редактор пользов.**, которая находится в приложении Центр управления;

- **Блокировка узла** - право на блокировку узла из Центра управления. Заблокированный узел не будет запускаться (данный функционал еще не реализован);

- **Проверить подпись Runtime** - пользователь сможет пользоваться функционалом проверки целостности системы. Данный функционал состоит из кнопок **Целостность** и **Полная целостность**, которые располагаются в приложении Центр управления;

- **Подписать Runtime** - пользователь сможет пользоваться функционалом добавить/удалить цифровые подписи файлов проекта. Данный функционал состоит из кнопок **Подписать** и **Снять подпись**, которые располагаются в приложении Центр управления;

- **Просмотр архива ИБ** - для данного пользователя во всех приложениях, где можно просматривать события системы, будут дополнительно отображаться события ИБ (информационной безопасности) системы.

- **Права на проект** - в данном поле будут отображаться различные права на проект, которые создаются вручную во вкладке **Права** (см. далее). Функционал данных прав необходимо реализовывать самостоятельно (см. документ SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01.23 функциональный блок **CHECK_RIGHTS**).

В нижней части окна общих настроек пользователя располагается список групп пользователей, в которые он может быть включен. Группы создаются во вкладке Группы (см. далее).

ВНИМАНИЕ! Итоговые права пользователя, которые располагаются во вкладке Общие, получаются сложением по принципу логического ИЛИ прав пользователя и прав группы, к которой пользователь относится.

ВНИМАНИЕ! При первоначальном создании пользователя администратор может не назначать пользователю пароль по кнопке **Смена пароля**. В данном случае пользователю будет назначен пустой пароль и при первом входе пустым паролем пользователю будет выведено окно смены пароля. Если администратор при создании пользователя назначит ему пароль, то при первом входе данным паролем окно на смену пароля выводиться не будет. Если нужно чтобы администратор назначил первоначальный пароль, а потом пользователь его сменил, то после назначения пароля администратором активируйте флаговую кнопку **Требовать смены пароля**.

3.2.2.10.1.2. Вкладка Сессии

Во вкладке Сессии располагаются настройки рабочих сессий для пользователя (см. рис. 3.62).

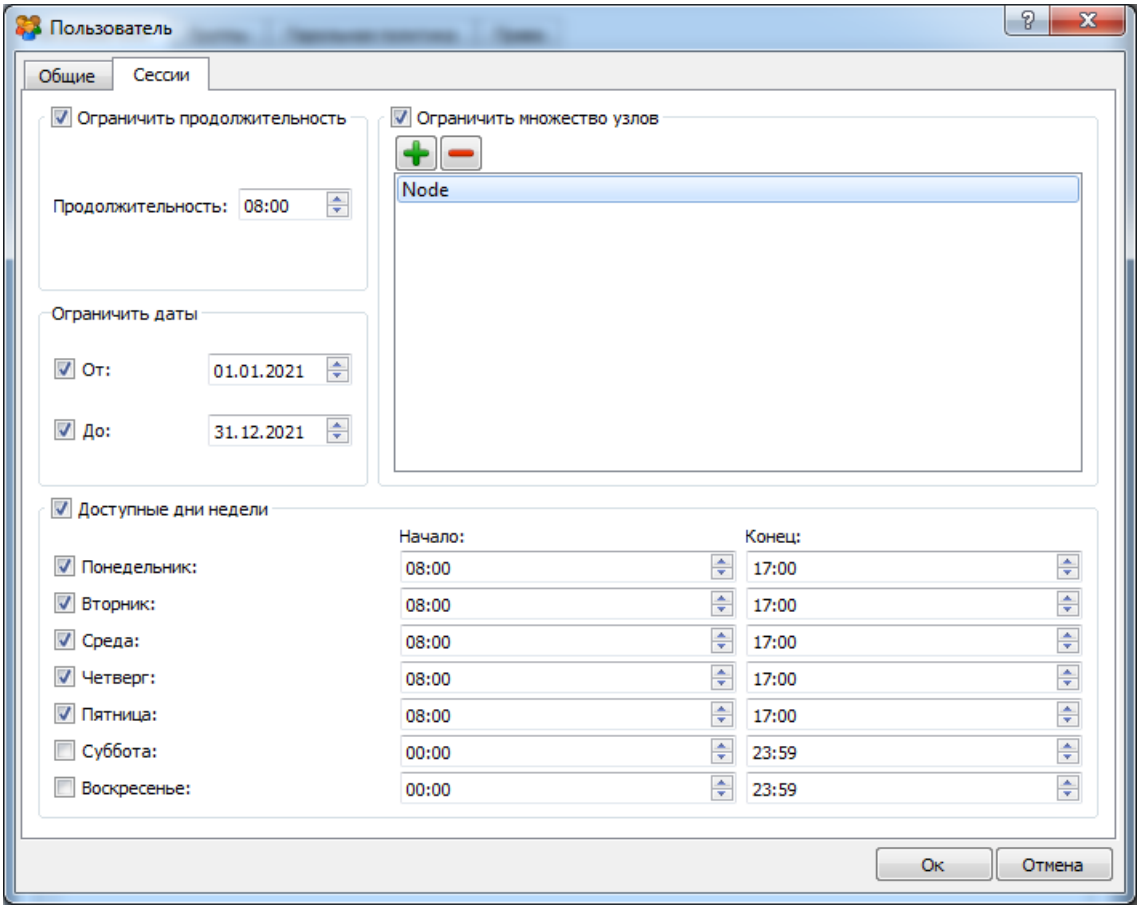


Рисунок 3.62 - Окно вкладки Сессии

Окно вкладки Сессии состоит из следующих элементов:

- чекбокс Ограничить продолжительность в активном состоянии ограничивает время работы пользователя в текущей сессии на указанное количество времени. Продолжительность работы указывается в поле Продолжительность: 08:00 ;

- поле Ограничить даты позволяет настроить даты начала и конца периода, в течении которого пользователь будет иметь право работать с системой;

- чекбокс Ограничить множество узлов в активном состоянии позволяет работать пользователю только с узлами, которые указаны в списке ниже. Узлы в список добавляются и удаляются с помощью кнопок ("Добавить") и ("Удалить");

- чекбокс Доступные дни недели в активном состоянии позволяет настроить доступные для работы пользователя дни недели и время, в течении которого пользователь сможет работать в этот день.

ВНИМАНИЕ! Настройки во вкладке Сессии работают по принципу логического ИЛИ, т.е., если продолжительность работы пользователя настроена на 8 часов, а начало - конец работы настроены с 8-00 до 17-00, то сколько бы пользователь по времени не отработал, но в 17-00 доступ ему будет закрыт и т.д.

Особенности наследования значений параметров сессионных ограничений.

Если в настройках пользователя активирован определенный параметр ограничения по времени (продолжительность, диапазон на каждый день, диапазон по дням недели), то это значение является результирующим (приоритетным), вне зависимости от значений аналогичных параметров в группах, в которые входит пользователь.

Если же у пользователя определенный параметр не активирован, то его значение наследуется от аналогичных параметров групп, в которые входит пользователь.

Наследование параметров от групп происходит следующим образом:

Значения параметров групп агрегируются, чтобы в результате получить наибольшую длительность и наибольшие диапазоны времени.

Примеры:

- Польз.Продолж.=откл., Группа1.Продолж.=1день, Группа2.Продолж.=5дней => Результир.Продолж.=5дней.

- Польз.ДиапазнНаКаждДень=откл, Группа1.ДиапазнНаКаждДень=12:00-15:00, Группа2.Диапазон=7:00-13:00 => Результир.ДиапазнНаКаждДень=7:00-15:00;

- Польз.ДиапазнНаКаждДень=10:00-13:00, Группа1.ДиапазнНаКаждДень=8:00-17:00 => Результир.ДиапазнНаКаждДень=10:00-13:00.

Временные диапазоны по дням будут агрегироваться следующим образом:

Если у пользователя не активирован параметр временного диапазона по определенному дню недели, а у группы данный параметр активирован, то значение параметра из группы будет браться за результирующее. Если имеется несколько групп, у которых есть активированный параметр по аналогичному дню, то значения параметров этих групп будут агрегироваться по аналогии с параметром диапазона на каждый день.

Например:

Польз.ДиапазонПоДням{ПН=10:00-12:00,ВТ=12:00-15:00},
Группа1.ДиапазонПоДням{ПН=8:00-15:00,СР=12:00-15:00},
Группа2.ДиапазонПоДням{ВТ=8:00-13:00,СР=7:00-13:00,ЧТ=8:00-17:00} =>
Результир.ДиапазонПоДням{ПН=10:00-12:00,ВТ=12:00-15:00,СР=7:00-15:00,ЧТ=8:00-17:00}

3.2.2.10.2. Вкладка Группы

Во вкладке Группы можно создать группы для пользователей, обладающих одинаковыми правами (см. рис. 3.63).

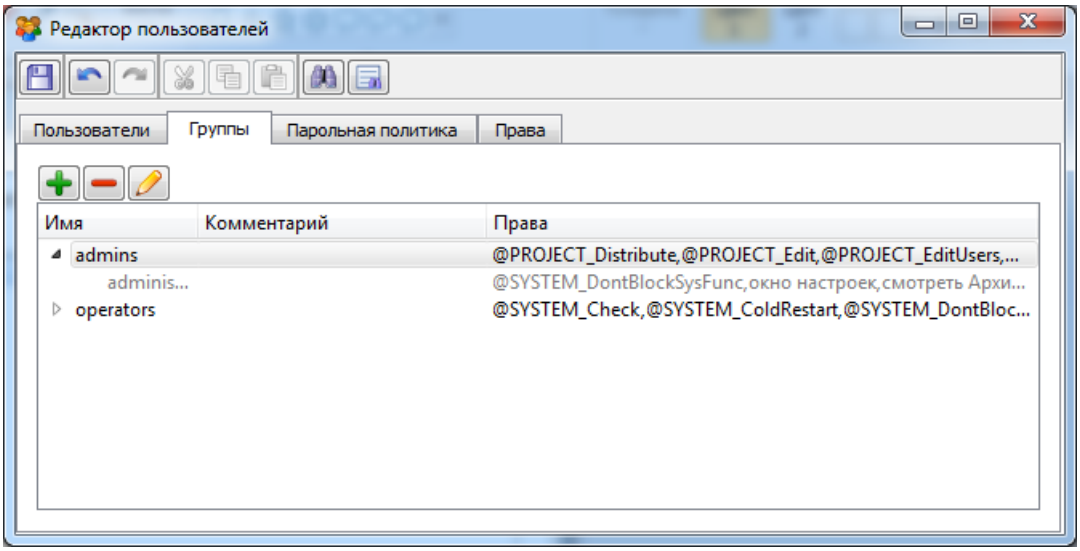


Рисунок 3.63 - Окно вкладки Группы

В окне вкладки Группы располагаются следующие кнопки:

- ("**Добавить**") - по нажатию на данную кнопку откроется окно с настройками прав для группы (см. рис. 3.64). Данные настройки такие же, как настройки при создании пользователя (см. раздел 3.2.2.10.1);

- ("**Удалить**") - удаляет выбранную группу из списка;

- ("**Редактировать**") - открывает на редактирование окно с настройками прав для группы (см. рис. 3.64). Данные настройки такие же, как настройки при создании пользователя (см. раздел 3.2.2.10.1).

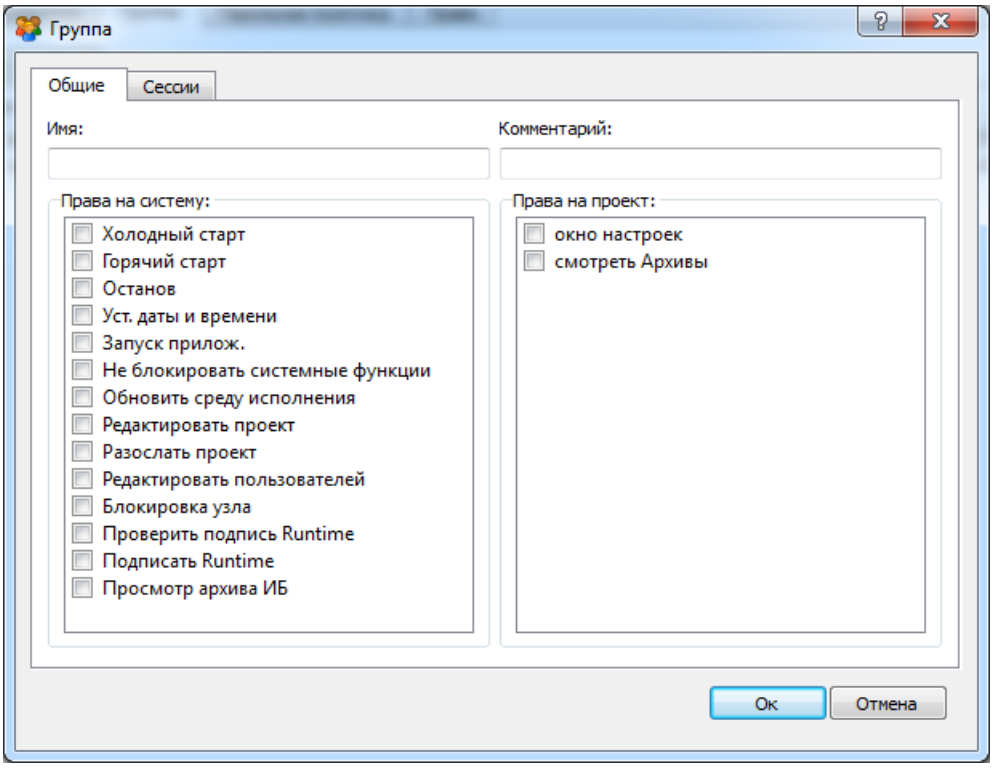


Рисунок 3.64 - Окно настроек для группы

3.2.2.10.3. Вкладка Парольная политика

Во вкладке Парольная политика можно настроить политики для паролей пользователей (см. рис. 3.65).

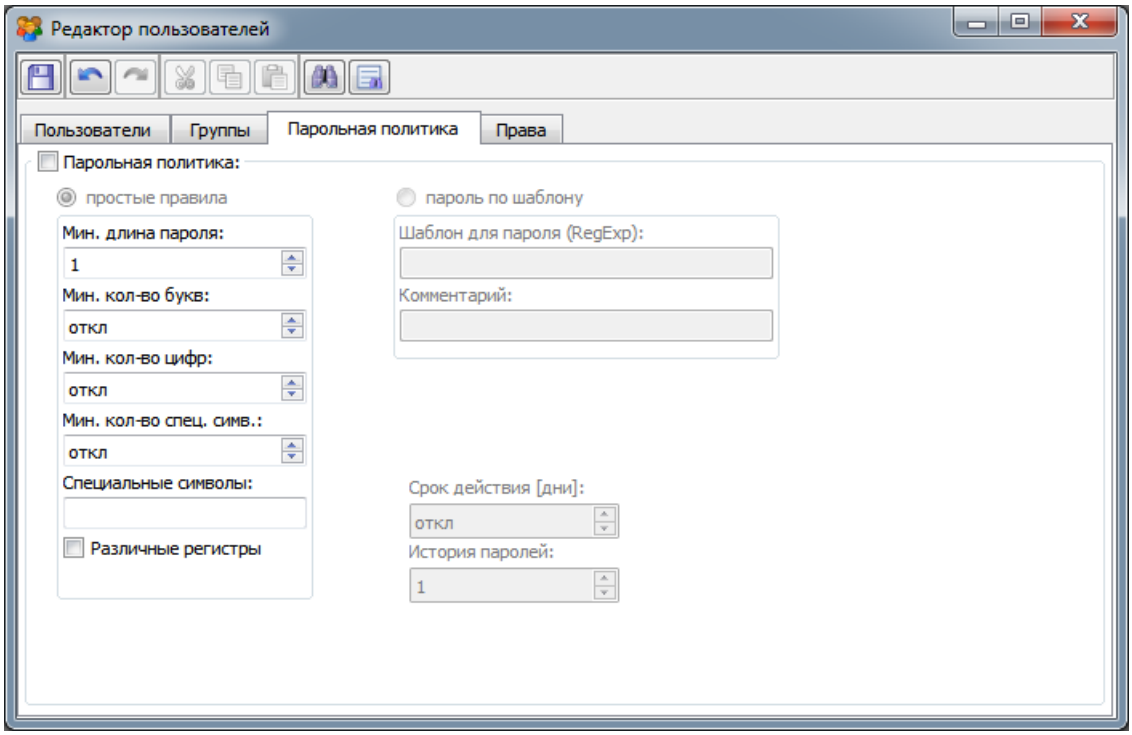
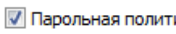
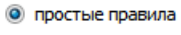
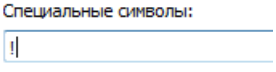
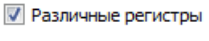
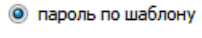


Рисунок 3.65 - Окно настроек для парольной политики

Окно настроек Парольная политика состоит из полей, описанных далее.

Чекбокс  - в активном состоянии включает возможность задать правила для паролей в системе.

Парольная политика может быть настроена с использованием двух вариантов правил:

-  - в активном состоянии включает следующие настройки:
  - Мин. длина пароля:
     - минимальное количество символов в пароле;
  - Мин. кол-во букв:
     - задаёт минимальное количество букв в пароле;
  - Мин. кол-во цифр:
     задаёт минимальное количество цифр в пароле;
  - Мин. кол-во спец. симв.:
     - задаёт минимальное количество специальных символов в пароле;
  - Специальные символы:
     - специальные символы, которые обязательно должны присутствовать в пароле;
  -  - в активном состоянии требует чтобы в пароле присутствовали символы в различных регистрах.
-  - делает активными настройки для задания шаблона используемого пароля:
  - Шаблон для пароля (RegExp):
     - с помощью регулярных выражений можно задать шаблон для используемого пароля;

- Комментарий:

 - комментарии к шаблону.

- Общие настройки:
  - Срок действия [дни]:

 - указывается срок действия пароля в днях. После истечения срока действия будет выдан запрос на замену пароля;

  - История паролей:

 - количество (глубина хранения) старых паролей пользователя, которые хранит система и не позволяет повторять для задания, как новый пароль.

3.2.2.10.4. Вкладка Права

Во вкладке Права создаются различные права на проект (см. рис. 3.66). Функционал данных прав необходимо реализовывать самостоятельно (см. документ SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01.23 функциональный блок **CHECK_RIGHTS**).

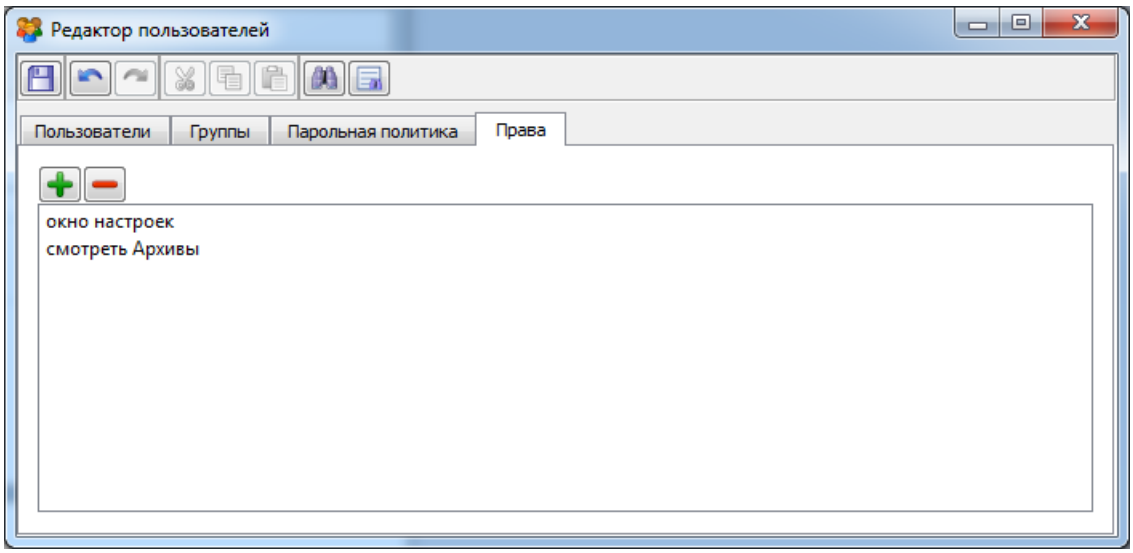


Рисунок 3.66 - Окно вкладки Права

В окне вкладки Права с помощью кнопок  ("Добавить") и  ("Удалить") можно добавить и удалить пользовательские права.

Созданные в данной вкладке права нужно самостоятельно реализовать. К примеру, если в проекте создать окно с инженерными настройками, которые должны быть доступны только определенному пользователю. То доступ к данному окну инженерных настроек нужно реализовать с помощью функционального блока **CHECK_RIGHTS**, на входе которого проверяется наличие у пользователя определённых прав. В нашем случае нужно прверять право **окно настроек**. У данного пользователя данное право должно быть включено в Права на проект (см. рис. 3.61).

3.2.2.11. Проверка проекта

По нажатию на кнопку  ("Проверить проект") на панели инструментов окна программы **ProjectManager** проводится проверка структуры проекта.

3.2.2.12. Компиляция проекта

По нажатию на кнопку  ("Откомпилировать проект") на панели инструментов окна программы **ProjectManager** проводится компиляция проекта и запись на диск конфигурационных файлов для запуска проекта.

На этапе компиляции в папке проекта создается папка **Runtime**, в которой создаются конфигурационные файлы для запуска приложений на узлах и в которую копируются необходимые для работы приложений файлы. При каждой компиляции проекта папка **Runtime** каждый раз создается заново.

3.2.2.13. Распространение (копирование) проекта на узлы

После удачной компиляции проекта его необходимо распространить (скопировать) на узлы проекта (контроллеры, рабочие станции и др.). Для этого используется кнопка на верхней панели инструментов  - Распространить.

По нажатию на эту кнопку открывается окно (см. рис. 3.67) и автоматически начинается распространение проекта на все узлы, включенные в данный проект.

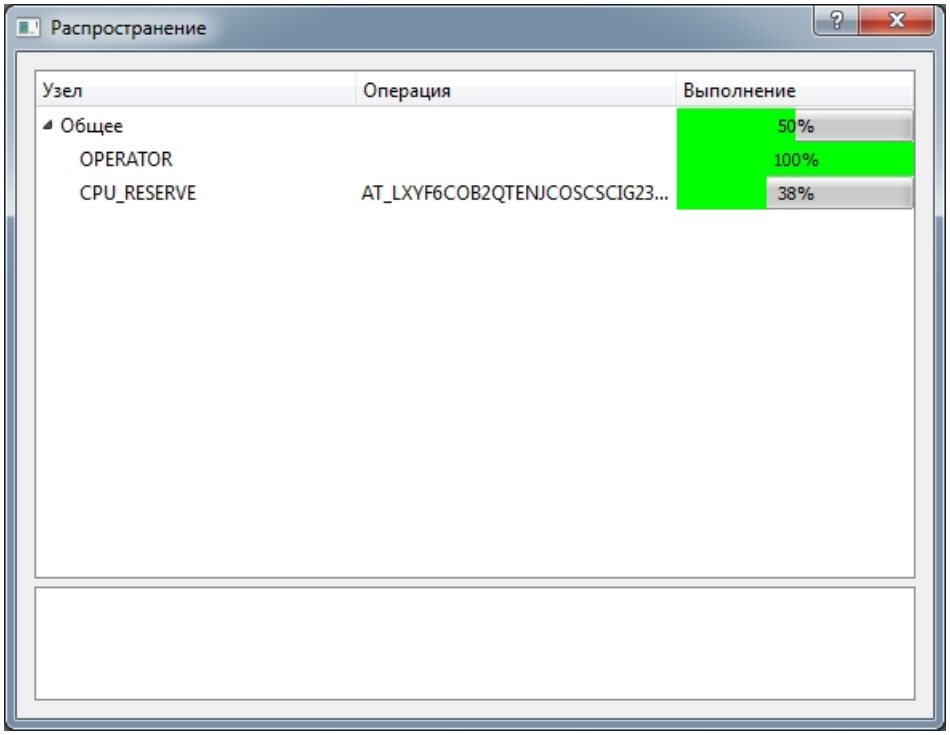


Рисунок 3.67 - Окно программы автоматического распространения проекта на узлы

При удачном завершении процесса распространения проекта будет выведено сообщение см. рис. 3.68.

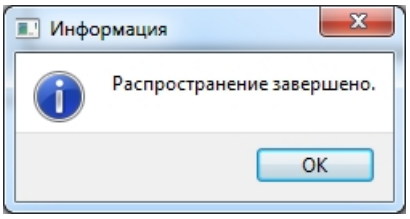


Рисунок 3.68 - Сообщение об удачном распространении проекта

Если процесс распространения проекта завершился с ошибками (или долго стоит на одном месте), то следует разобраться с возможными проблемами. Это отсутствие связи с узлом или узлами, нужно проверить, был ли ранее скомпилирован проект, проверить, запущена ли на узлах программа **Loader** (программа для загрузки проекта на узлах системы "Соната") и др. причины.

ВНИМАНИЕ! Распространение проекта запрещено при стартующем или останавливающимся узле. При попытке распространения Distributor (приложение, распространяющее проект) выдаст ошибку "Доступ запрещён (403)". Возможна ситуация, когда из-за какой-либо ошибки в приложениях, в ходе старта узла, он не запустился до конца. Это состояние также считается состоянием старта узла, при котором распространение проекта запрещено. Для выхода из этой ситуации нужно из центра управления подать команду на останов проекта.

3.2.2.14. Запуск проекта и контроль за его выполнением

По нажатию на кнопку  в панели инструментов главного окна **ProjectManager** (Менеджер проектов), будет запущена программа управления - **ControlCenter** (Центр управления).

Программа **ControlCenter** (Центр управления) считывает конфигурационные файлы и определяет структуру системы. После этого она начинает циклический опрос элементов системы: узлов (компьютеры и главные контроллеры) и выполняемых на них программ (драйвера, технологические алгоритмы, контроль уставок и т.п.). В данной программе осуществляется ручной запуск и останов проекта и его узлов, также можно осуществлять контроль и диагностику проекта в процессе работы (подробно см. раздел 3.4.4).

3.3. Описание типов приложений SCADA-системы "Соната"

3.3.1. Виды приложений в SCADA-системе «Соната»

Таблица 3.1 - Виды приложений

Вид приложения	**Описание**
Внешние приложения	
APPLICATION	Внешнее неуправляемое приложение
APPLICATION.CONSOLE	Внешнее консольное управляемое приложение
APPLICATION.WINDOW	Графическое управляемое приложение системы

Вид приложения	Описание
Приложения для построения технологических программ	
APPLICATION.IEC.CONSOLE	Консольное управляемое событийное приложение, написанное на языках стандарта IEC-61131 или IEC-61499
APPLICATION.LD.CONSOLE	Консольное управляемое циклическое приложение, написанное на языке LD (в данный момент не используется)
APPLICATION.ST.CONSOLE	Консольное управляемое циклическое приложение, написанное на языке ST
APPLICATION.LUA	Управляемое событийно/циклическое приложение, написанное на языке LUA
Приложения для построения человеко-машинного интерфейса	
APPLICATION.IEC.WINDOW	Графическое управляемое событийное приложение, написанное на языках стадарта IEC-61131 или IEC-61499
Приложения архивации	
ARCHIVE	Приложение-архив, для хранения значений сигналов системы
EVENT LOGGER	Архив событий проекта (автоматически регистрирует события проекта)
Приложения для работы с WEB	
WEB BROWSER	Приложение для просмотра WEB - страниц, как с диска, так и с удалённого сервера
WEB SERVER	Приложение-сервер для отдачи по HTTP - запросу статических файлов
Приложения-драйверы	
DRIVER.DTS	Приложение-драйвер, выполняющее функции шлюза СВБУ "ПОРТАЛ"
DRIVER.MODBUS	Универсальное приложение-драйвер для любых устройств, поддерживающих обмен данными по протоколу MODBUS RTU или TCP
DRIVER.OPCUA	Приложение-драйвер, предназначенное для взаимодействия внешних OPCUA-серверов со SCADA-системой "Соната"
DRIVER.OPCUA.SERVER	Приложение-драйвер, работающее как OPCUA-сервер и предназначенное для предоставления доступа к сигналам SCADA-системы "Соната" для внешних программ OPCUA-клиентов
DRIVER.PCI	Приложение-драйвер КМ-04 (контроллера многофункционального производства ЭЗАН)

Вид приложения	**Описание**
DRIVER.SNMP	Приложение-драйвер для работы с устройствами, предоставляющими доступ к данным по протоколу SNMP
DRIVER.SONET.FAILOVER	Приложение-драйвер для работы с модулем переключения резерва или БПР (блок переключения резерва)
DRIVER.SONET.LOCALBUS	Приложение-драйвер модулей ввода/ вывода контроллера СОНЕТ-Мастер (производства ЭЗАН)
DRIVER.SONET.MODBUS	Приложение-драйвер для опроса по MODBUS контроллеров удаленного ввода-вывода СОНЕТ (производства ЭЗАН)
DRIVER.TELECONT.MODBUS	Приложение-драйвер для опроса по протоколу MODBUS RTU контроллеров ТЕЛЕКОНТ (производства ЭЗАН) - в настоящее время мало применяется
DRIVER.TELECONT2.MODBUS	Приложение-драйвер для опроса по протоколу MODBUS RTU контроллеров ТЕЛЕКОНТ (производства ЭЗАН) - применяется в настоящее время
Прочие приложения	
BRIDGE	Приложение для настройки межпроектной связи (можно передавать значения сигналов, события и тревоги между разными проектами SCADA-системы "Соната")
EVENT VIEWER	Приложение для просмотра событий проекта (данное приложение уже не используется и присутствует в списке приложений для совместимости с АСУ ТП, которые работают с более старой версией SCADA-системы "Соната")
REPORT ENGINE	Приложение для формирования, просмотра и печати отчётов
SOUND	Приложение для воспроизведения звука
TREND VIEWER	Приложение для просмотра графиков (данное приложение уже не используется и присутствует в списке приложений для совместимости с АСУ ТП, которые работают с более старой версией SCADA-системы "Соната")

3.3.2. Внешние приложения

3.3.2.1. Описание приложения APPLICATION

Приложение **APPLICATION** является внешним неуправляемым приложением SCADA-системы "Соната". Данный вид приложения присваивается по умолчанию всем новым приложениям и используется для запуска внешних приложений, не входящих в среду разработки SCADA-системы "Соната".

3.3.2.1.1. Создание приложения APPLICATION

При создании нового приложения в SCADA-системе "Соната" ему присваивается вид приложения **APPLICATION** (см. раздел 3.2.2.4).

На рис. 3.69 изображено окно интерфейса приложения **APPLICATION**, в котором изменено имя приложения на **AppAPPLICATION** (см. раздел 3.2.2.4.3).

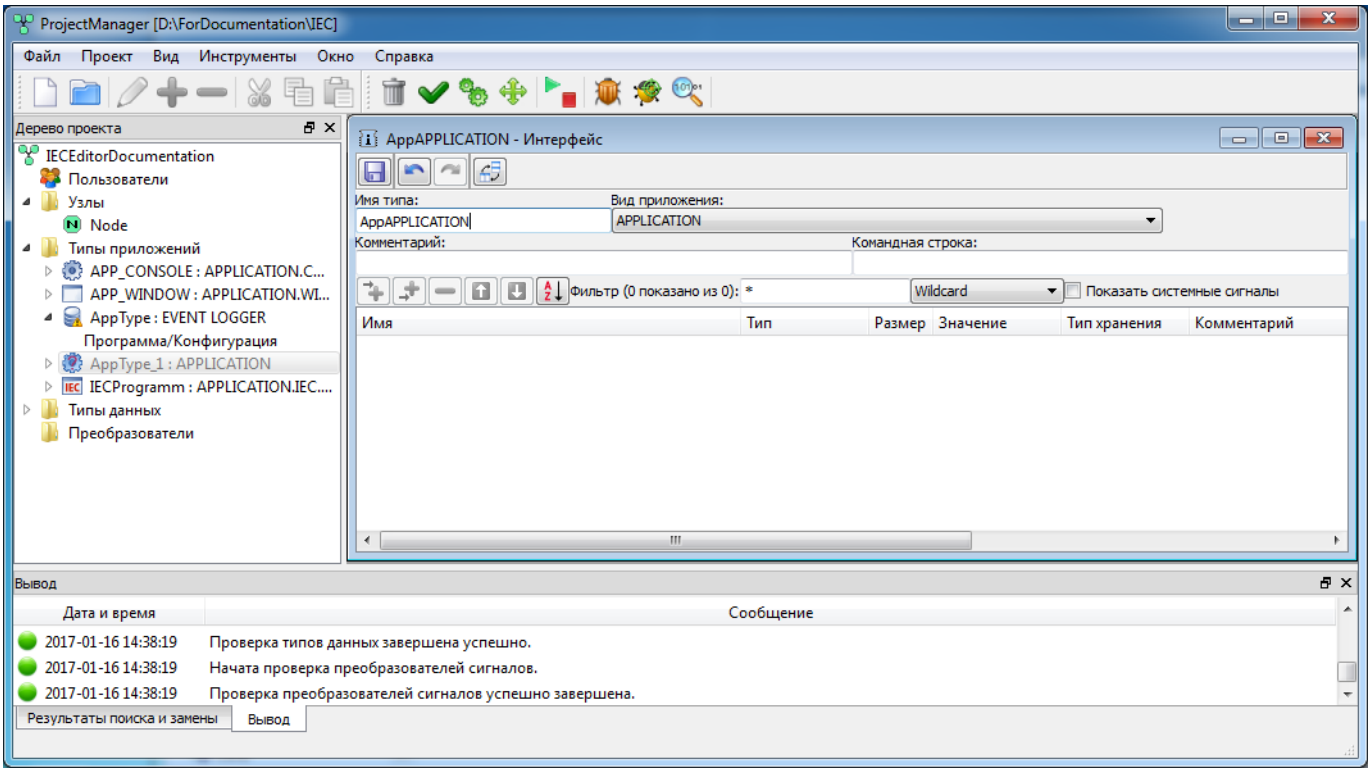


Рисунок 3.69 - Интерфейс приложения APPLICATION

3.3.2.1.2. Настройка интерфейса приложения APPLICATION

Основная настройка выполняется в поле **Командная строка**.

На рис. 3.70 в поле ввода **Командная строка** прописан исполняемый файл внешнего приложения **chip**, который будет запускаться при старте приложения **APPLICATION**. Путь к исполняемому файлу прописывается относительно расположения файлов проекта.

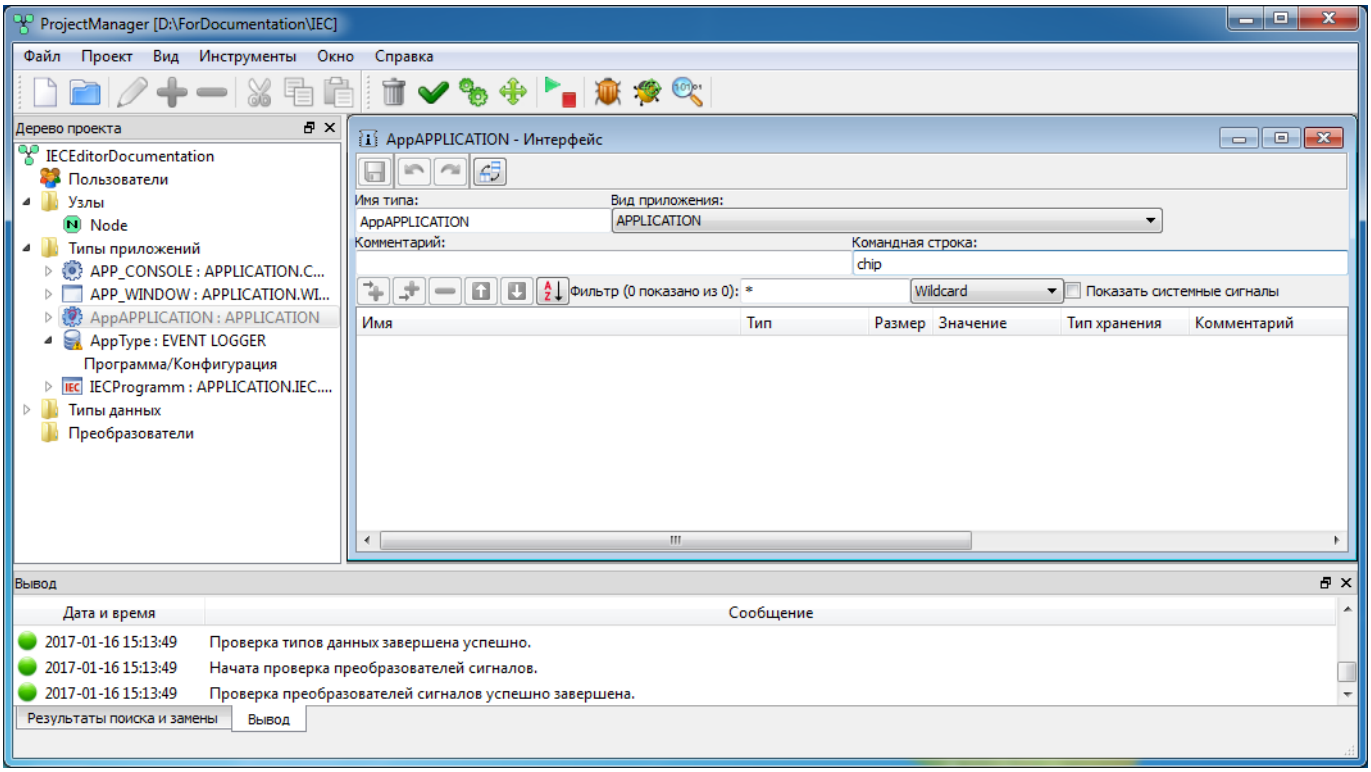


Рисунок 3.70 - Настройки в интерфейсе приложения APPLICATION

Изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

Для использования данного приложения в проекте, его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.2.2. Описание приложения APPLICATION.CONSOLE

Приложение **APPLICATION.CONSOLE** является внешним консольным управляемым приложением SCADA-системы "Соната". Данное приложение используется для запуска консольного приложения, написанного вне среды разработки SCADA-системы "Соната", но с помощью библиотек SCADA-системы "Соната".

3.3.2.2.1. Создание приложения APPLICATION.CONSOLE

Для создания приложения **APPLICATION.CONSOLE** сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.71 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** (необходимо выбрать **APPLICATION.CONSOLE**).

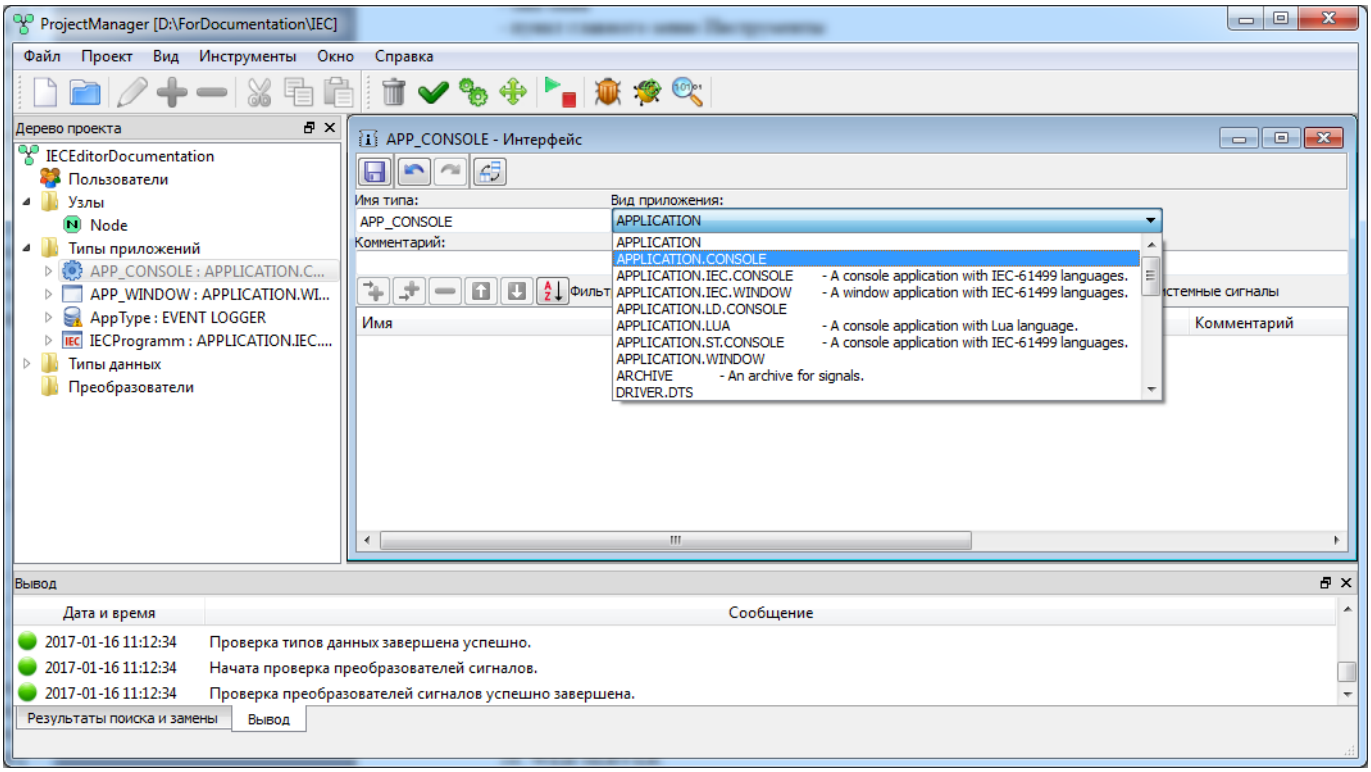


Рисунок 3.71 - Изменения вида приложения на APPLICATION.CONSOLE

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.72) – для подтверждения изменения необходимо нажать кнопку Да.

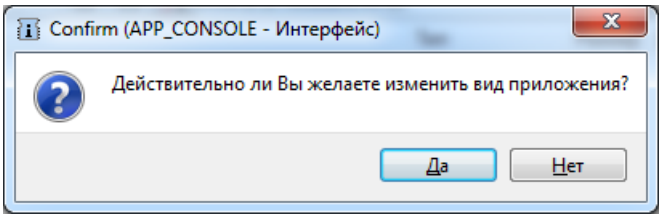


Рисунок 3.72 - Окно подтверждения для изменения вида типа приложения

3.3.2.2.2. Настройка интерфейса приложения APPLICATION.CONSOLE

Основная настройка выполняется в поле **Командная строка**.

На рис. 3.73 в поле ввода **Командная строка** прописан исполняемый файл консольного приложения **EventLogger**, который будет запускаться при старте приложения **APPLICATION.CONSOLE**. Путь к исполняемому файлу прописывается относительно расположения файлов проекта.

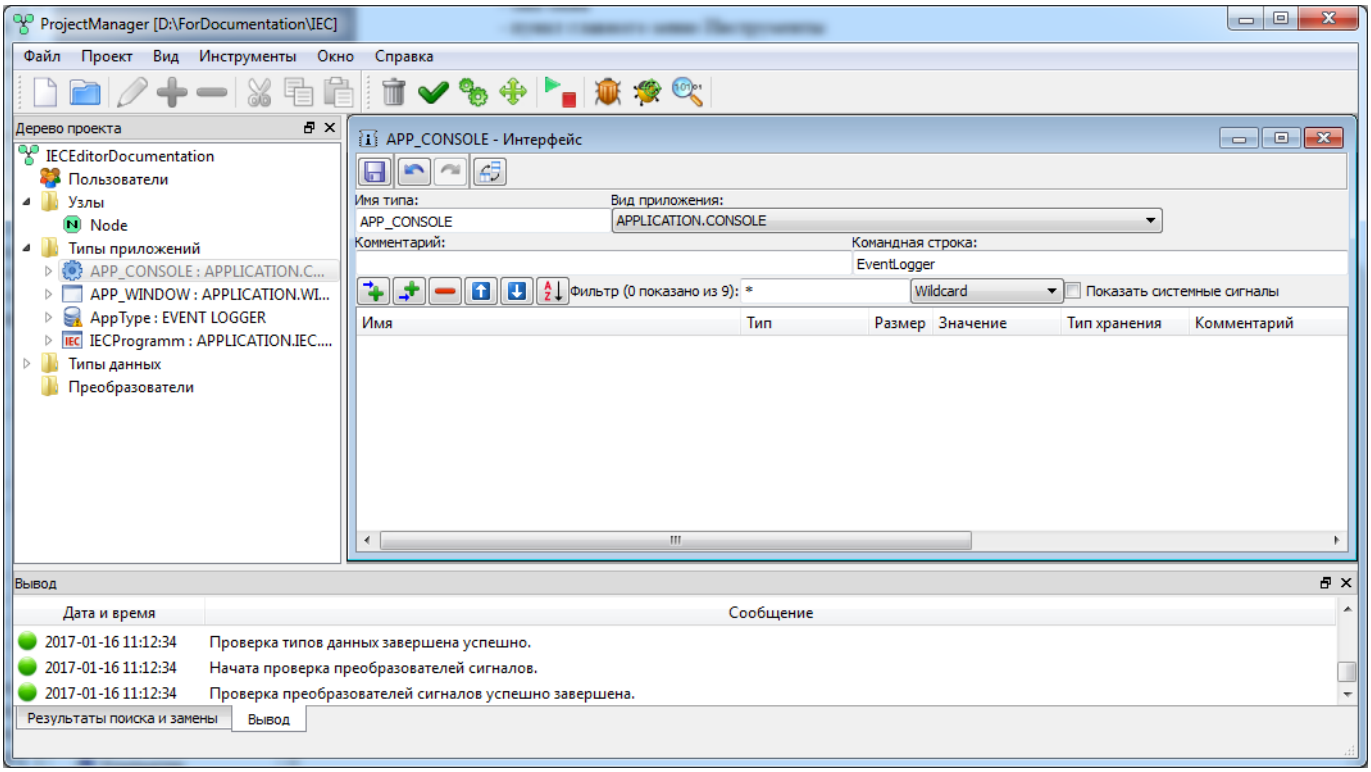


Рисунок 3.73 - Окно интерфейса приложения APPLICATION.CONSOLE

Изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

Для использования данного приложения в проекте, его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.3. Консольные приложения

3.3.3.1. Описание приложения APPLICATION.IEC.CONSOLE

Приложение **APPLICATION.IEC.CONSOLE** предназначено для создания консольных приложений, основанных на стандартах IEC 61131 и IEC 61499.

3.3.3.1.1. Создание приложения APPLICATION.IEC.CONSOLE

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.74 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **APPLICATION.IEC.CONSOLE**. Затем, в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **IECConsoleApp**

(допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

Также на рис. 3.74 в интерфейс приложения добавлены сигналы SCADA-системы "Соната", такие как Local, Local_1 и Local_2 (о создании и добавлении сигналов см. раздел 3.2.2.4.3).

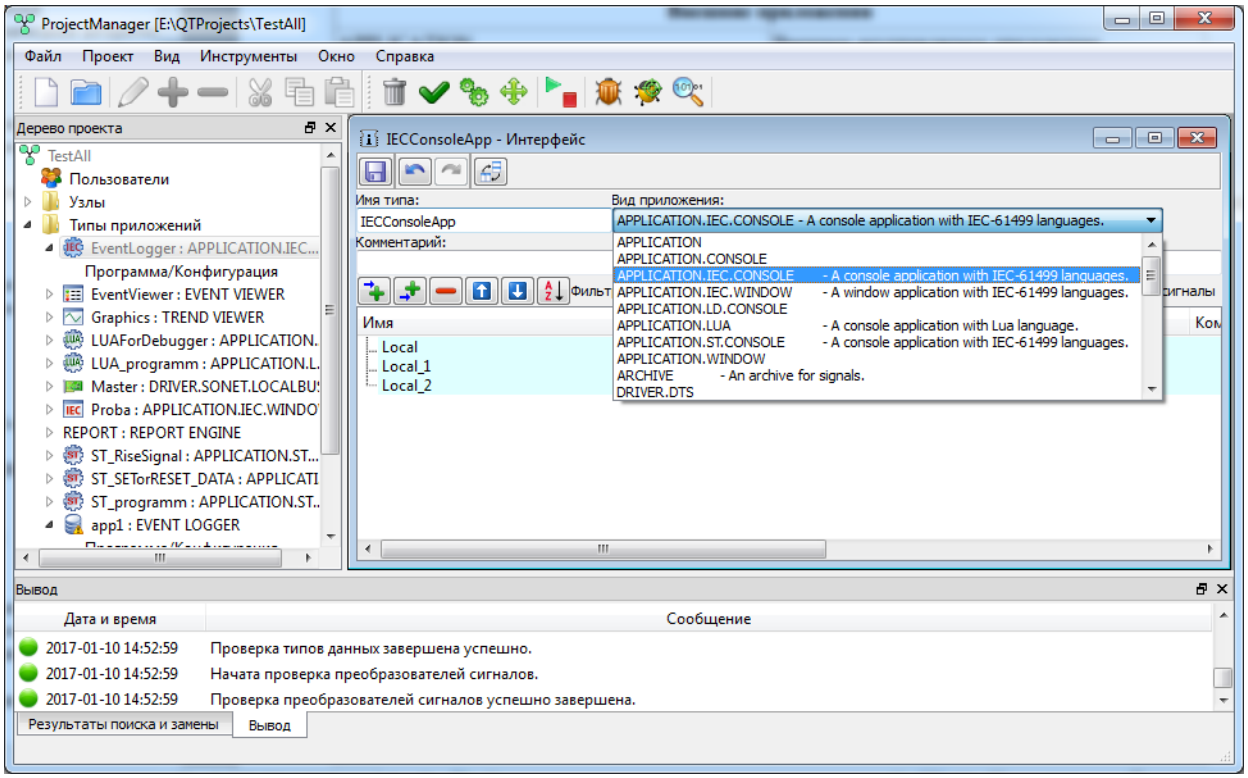


Рисунок 3.74 - Окно интерфейса приложения

По окончании редактирования интерфейса типа приложения необходимо сохранить изменения с помощью кнопки  («Сохранить»), после чего рекомендуется закрыть диалог редактирования крестиком  в верхнем правом углу окна.

3.3.3.1.2. Вызов специализированного редактора приложения APPLICATION.IEC.CONSOLE

В дереве проекта тип приложения отображается в виде двух позиций: верхняя позиция – интерфейс приложения, вложенная позиция – специализированный редактор приложения (далее редактор) или его конфигурационный файл. Для IEC-приложений редактором является программа **IECEditor**. Процесс вызова редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

Если окно редактирования интерфейса типа приложения не было закрыто на момент вызова программы **IECEditor**, то **Project Manager** сообщит о блокировке редактирования типа приложения (см. рис. 3.75) и не загрузит программу **IECEditor**.

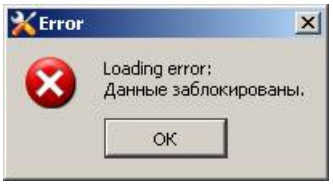


Рисунок 3.75 - Сообщение о блокировке элемента проекта

Для снятия блокировки необходимо закончить работу по редактированию интерфейса типа приложения и закрыть редактор способом, описанным выше. Возможно также обойти блокировку, используя команду Разблокировать элемент из раздела Проект главного меню (см. рис. 3.76). После этого можно повторно запустить **IECEditor** способами, описанными выше.

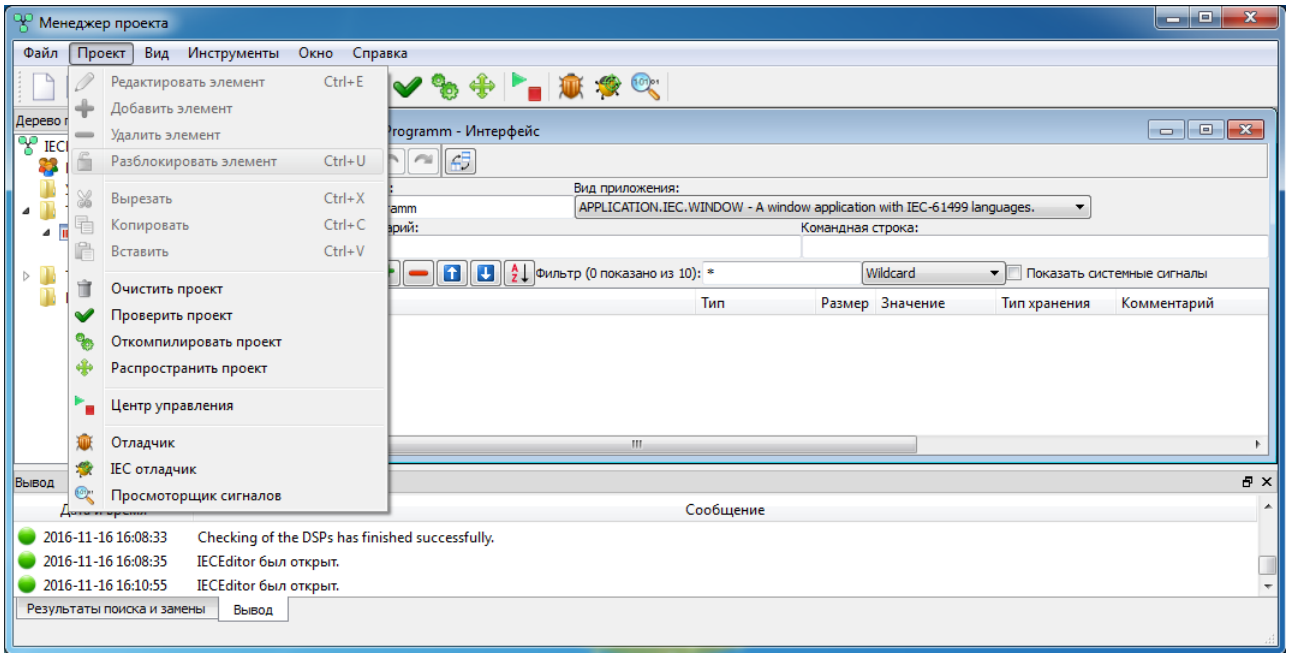


Рисунок 3.76 - Снятие блокировки элемента проекта с помощью команды главного меню

На рис. 3.77 изображено основное окно редактора IEC-приложений **IECEditor**.

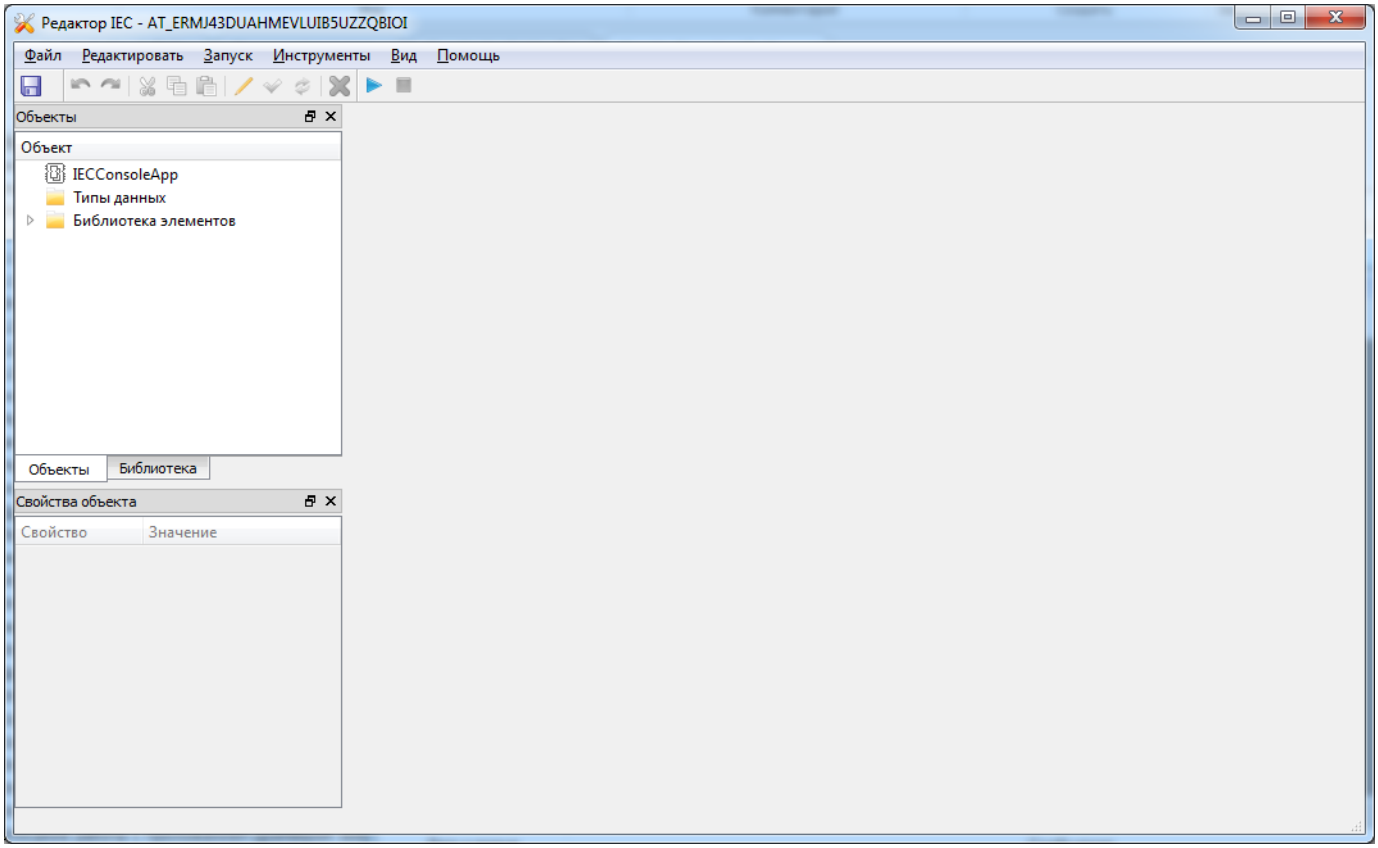


Рисунок 3.77 - Основное окно редактора IEC-приложений IECEditor

3.3.3.1.3. Описание интерфейса и работа со специализированным редактором приложения APPLICATION.IEC.CONSOLE

Описание специализированного редактора и принципов работы в нём представлено в раздел 3.4.8.

3.3.3.2. Описание приложения APPLICATION.ST.CONSOLE

Приложение **APPLICATION.ST.CONSOLE** применяется для написания алгоритмов на различных языках программирования ST (Structured Text - язык программирования стандарта IEC61131-3). Описание языка приведено в документе SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33. В данном разделе приведено описание по созданию приложения, работающего по алгоритму, написанному в специализированном редакторе языка ST.

3.3.3.2.1. Создание приложения APPLICATION.ST.CONSOLE

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению, осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.78 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **APPLICATION.ST.CONSOLE**. Затем, в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **ST_programm** (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

Также на рис. 3.78 в интерфейс приложения добавлены сигналы SCADA-системы "Соната", такие как RESET, Master_Modul1_IN_from_Mnemoshema, Master_Modul1_IN_to_Module и другие (о создании и добавлении сигналов см. раздел 3.2.2.4.3).

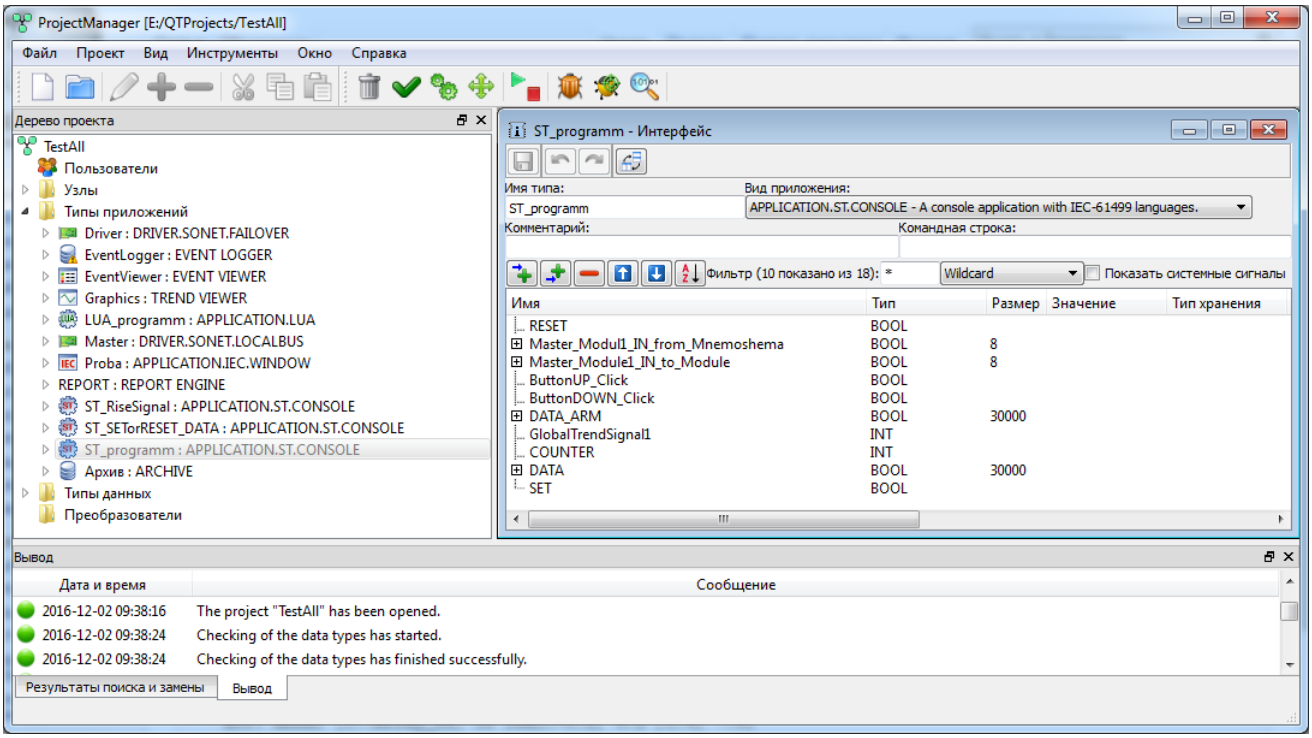


Рисунок 3.78 - Интерфейс приложения APPLICATION.ST.CONSOLE

По окончании редактирования интерфейса типа приложения необходимо сохранить изменения с помощью кнопки  («Сохранить»), после чего рекомендуется закрыть диалог редактирования крестиком  в верхнем правом углу окна.

3.3.3.2.2. Вызов специализированного редактора для написания алгоритмов на языке ST

В дереве проекта тип приложения отображается в виде двух позиций: верхняя позиция – интерфейс приложения, вложенная позиция – специализированный редактор приложения (далее редактор) или его конфигурационный файл. Процесс вызова редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

Если окно редактирования интерфейса типа приложения не было закрыто на момент вызова редактора алгоритмов ST, то **Project Manager** сообщит о блокировке редактирования типа приложения (см. рис. 3.79) и не загрузит редактор.

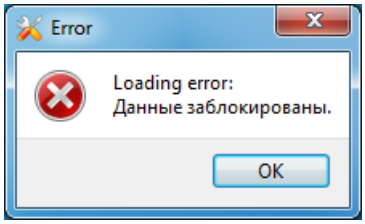


Рисунок 3.79 - Окно сообщения о невозможности открытия редактора

Для снятия блокировки необходимо закончить работу по редактированию интерфейса типа приложения и закрыть редактор способом, описанным выше. Возможно также обойти блокировку,

используя команду Разблокировать элемент из раздела Проект главного меню (см. рис. 3.80). После этого можно повторно запустить редактор алгоритмов ST.

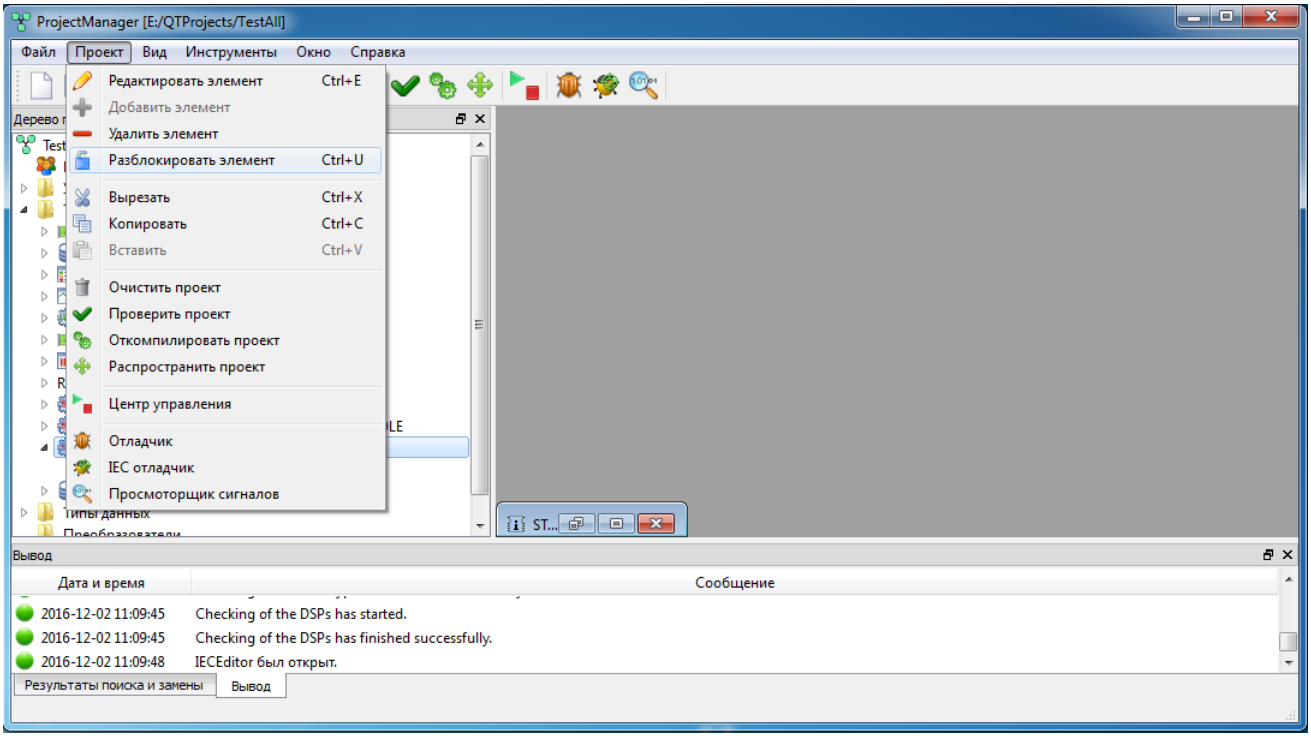


Рисунок 3.80 - Пункт для разблокировки приложения

На рис. 3.81 изображено окно специализированного редактора для написания алгоритмов на языке ST.

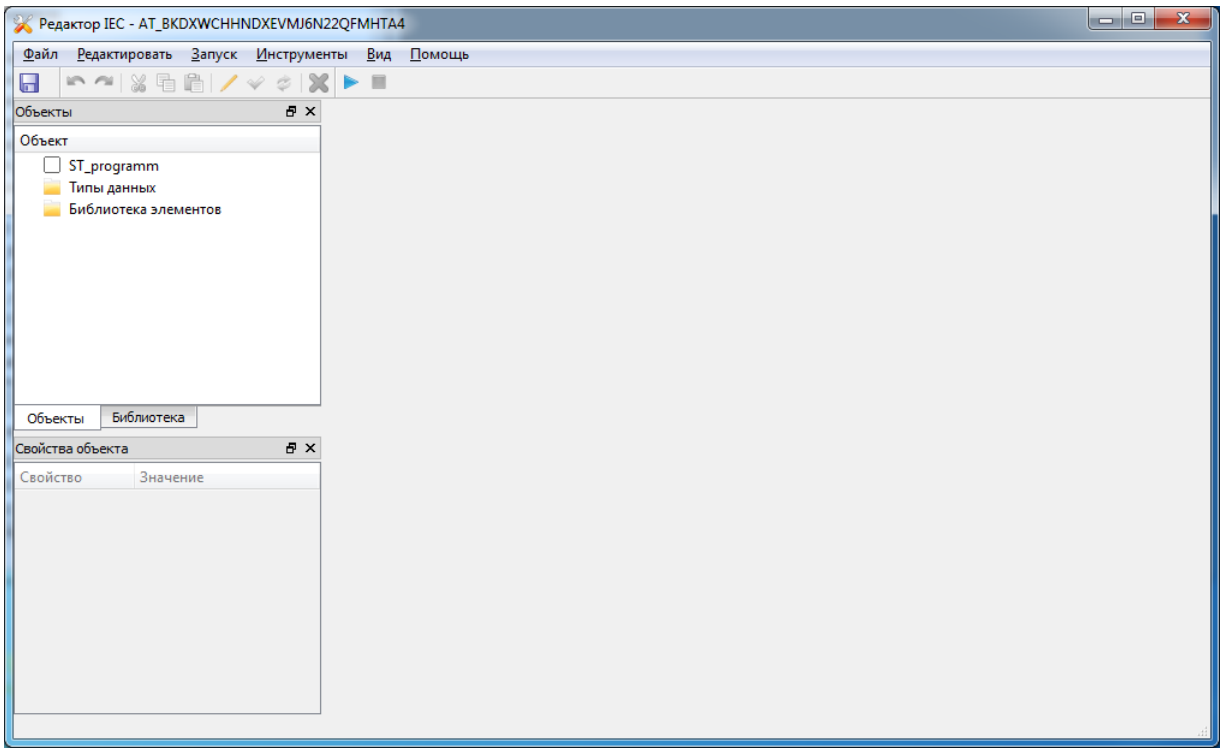


Рисунок 3.81 - Редактор алгоритмов языка ST

3.3.3.2.3. Описание интерфейса окна редактора алгоритмов языка ST

Редактор приложения **APPLICATION.ST.CONSOLE** открывается в отдельном окне. Главное окно программы содержит элементы, необходимые для формирования библиотеки функций и функциональных блоков, написанных на языке ST, написания основного алгоритма приложения и отладки данного алгоритма. Внешний вид главного окна приведен на рис. 3.82.

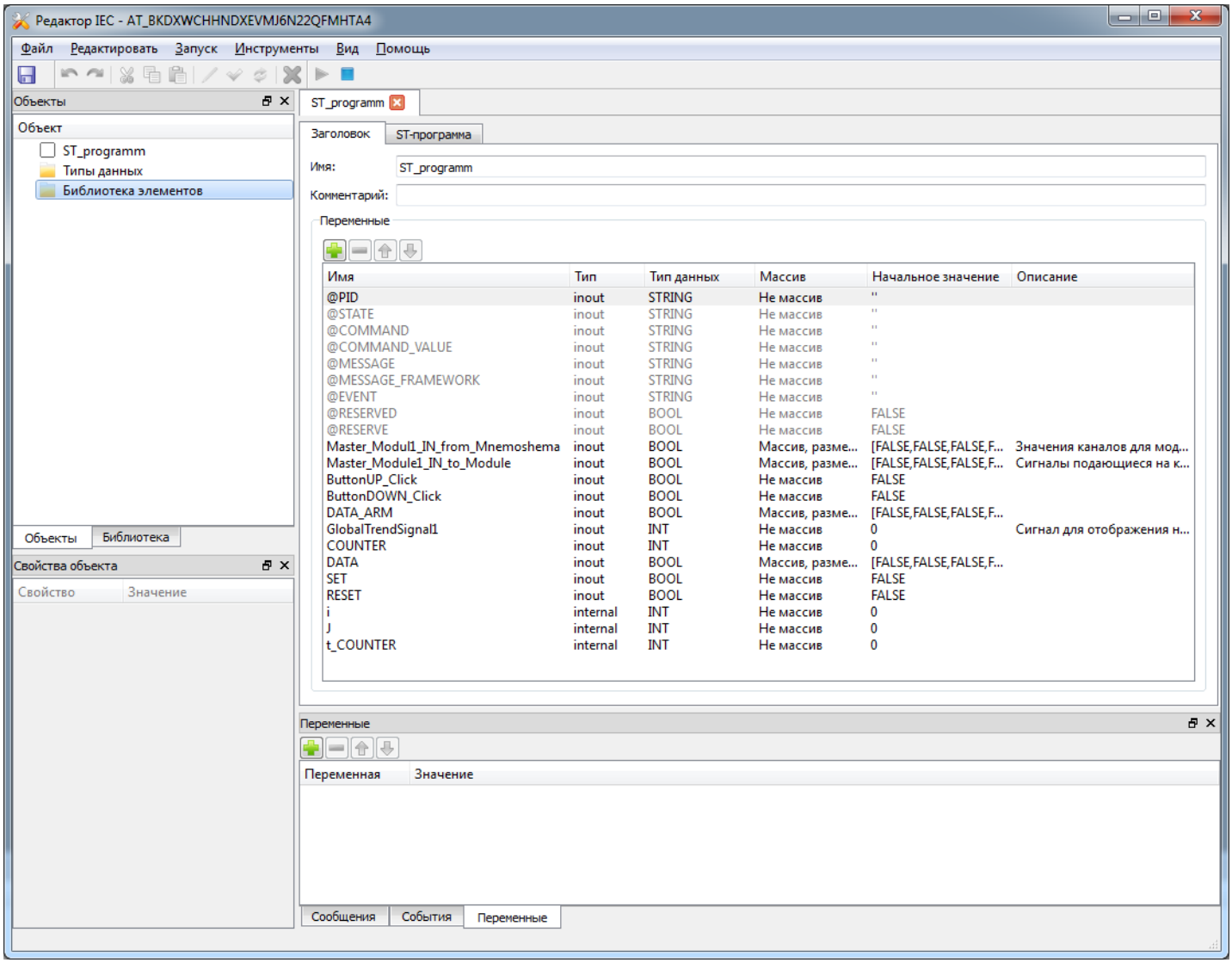


Рисунок 3.82 - Основное окно редактора APPLICATION.ST.CONSOLE

Главное окно редактора содержит следующие элементы:

- в верхней левой части имя редактора **Редактор IEC** и уникальное имя **AT_******;
- в верхней правой части находятся кнопки управления окном редактора  , соответственно свернуть, развернуть и закрыть окно редактора;
- главное меню (см. раздел 3.3.3.2.3.1);
- панель инструментов (см. раздел 3.3.3.2.3.2);
- окно дерева объектов (см. раздел 3.3.3.2.3.3);
- окно библиотеки типов функциональных блоков (см. раздел 3.3.3.2.3.4);
- окно редактора свойств (не используется в данном редакторе);

- окно редактора алгоритмов, функций и функциональных блоков (см. раздел 3.3.3.2.3.5);
- окно списка сообщений (см. раздел 3.3.3.2.3.6);
- окно списка событий (см. раздел 3.3.3.2.3.7);
- окно переменных отладчика (см. раздел 3.3.3.2.3.8).

3.3.3.2.3.1. Описание главного меню

Главное меню содержит полный список операций редактора приложения **APPLICATION.ST.CONSOLE** и делится на следующие разделы:

Файл - группа команд работы с файлами (см. раздел 3.3.3.2.3.1.1);

Редактировать - группа команд редактирования типа приложения (см. раздел 3.3.3.2.3.1.2);

Запуск - группа команд запуска и отладки приложения (см. раздел 3.3.3.2.3.1.3);

Инструменты - группа команд дополнительных функций (см. раздел 3.3.3.2.3.1.4);

Вид - группа команд, управляющих видимостью отдельных окон редактора (см. раздел 3.3.3.2.3.1.5);

Помощь - группа команд доступа к справочной информации (см. раздел 3.3.3.2.3.1.6);

3.3.3.2.3.1.1. Файл - группа команд работы с файлами

На рис. 3.83 представлено развернутое меню группы команд работы с файлами из раздела Файл.

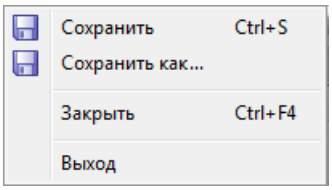


Рисунок 3.83 - Развернутое меню раздела Файл

Раздел Файл (группа команд работы с файлами) содержит следующие команды:

Сохранить Ctrl+S – сохранить внесённые изменения;

Выход – завершить работу с редактором.

3.3.3.2.3.1.2. Редактировать - группа команд редактирования типа приложения

На рис. 3.84 представлено развёрнутое меню группы команд Редактировать.

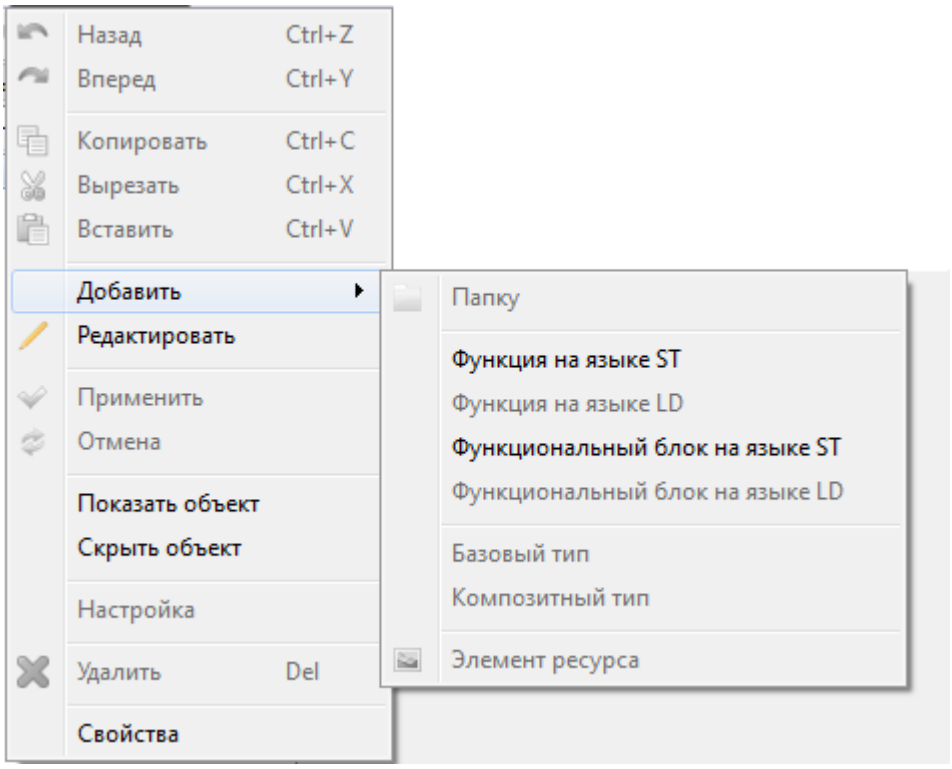


Рисунок 3.84 - Развёрнутое меню раздела Редактировать

Раздел Редактировать содержит следующие команды и подразделы (некоторые команды не описаны, так как не применяются в данном приложении):

Назад Ctrl+Z – отменить предыдущее действие по редактированию;

Вперед Ctrl+Y – повторить ранее отменённое действие;

Копировать Ctrl+C – копировать выделенный объект (объекты) в буфер памяти;

Вырезать Ctrl+X – вырезать выделенный объект (объекты) и сохранить его в буфере памяти;

Вставить Ctrl+V – вставить вырезанный или скопированный объект (объекты) из буфера памяти;

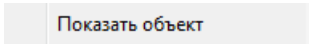
Добавить подраздел для создания новых типов функциональных блоков или папок библиотеки, содержит следующие команды:

- Папку – создать папку в дереве объектов;
- Функция на языке ST – создать объект - функцию на языке ST;
- Функциональный блок на языке ST – создать функциональный блок на языке ST;

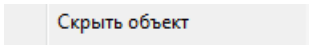
Редактировать – открыть тип функционального блока для просмотра или редактирования;

Применить – сохранить сразу все сделанные при редактировании функционального блока изменения;

Отмена – отменить сразу все сделанные при редактировании функционального блока изменения;

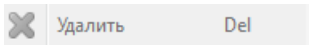


– показать скрытый графический объект в редакторе мнемосхем (не используется в данном приложении);

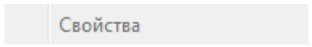


– скрыть видимый графический объект в редакторе мнемосхем (не используется в данном приложении);

– настроить объект (не используется в данном приложении);



– удалить выделенный объект или объекты (не используется в данном приложении);



– редактировать свойства объекта (папки или типа функционального блока).

3.3.3.2.3.1.3. Запуск – группа команд запуска и отладки приложения

На рис. 3.85 представлено развёрнутое меню группы команд Запуск для отладки приложения.

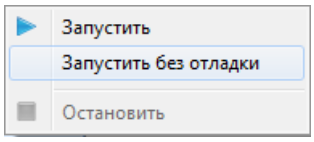
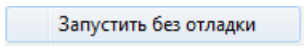


Рисунок 3.85 - Развёрнутое меню раздела Запуск

Раздел Запуск (группа команд запуска и отладки приложения) содержит следующие команды:

– запустить приложение для отладки;



– запустить приложение без отладки;

– остановить отладку приложения.

3.3.3.2.3.1.4. Инструменты - группа команд с различными полезными функциями

На рис. 3.86 представлено развёрнутое меню группы команд Инструменты.

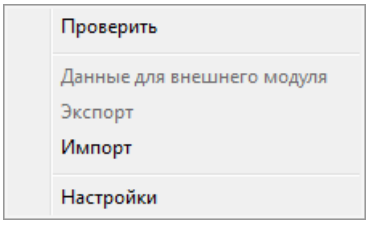


Рисунок 3.86 - Развёрнутое меню раздела Инструменты

Раздел Инструменты содержит следующие команды:

Проверить - проверить код на ошибки;

Данные для внешнего модуля - формирует данные для вынесения в отдельный модуль;

Экспорт - экспортировать данные в текстовый файл;

Импорт - импортировать данные из ранее сохранённых;

Настройки - запуск меню настроек по работе с редактором (см. раздел 3.3.3.2.3.9).

3.3.3.2.3.1.5. Вид – группа команд, управляющих видимостью отдельных окон редактора

На рис. 3.87 представлено развёрнутое меню группы команд Вид.

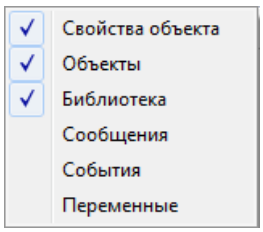


Рисунок 3.87 - Развёрнутое меню раздела Вид

Раздел Вид содержит следующие команды:

Свойства объекта - отобразить в редакторе окно свойств объекта;

Объекты - отобразить в редакторе окно дерева объектов;

Библиотека - отобразить в редакторе окно библиотеки типов функциональных блоков;

Сообщения - отобразить в редакторе окно сообщений;

События - отобразить в редакторе окно событий;

Переменные - отобразить в редакторе окно переменных для отладки.

3.3.3.2.3.1.6. Помощь – группа команд доступа к справочной информации

На рис. 3.88 представлено развёрнутое меню группы команд Помощь.

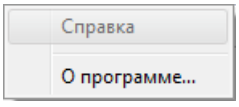
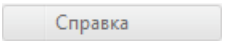
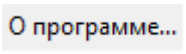


Рисунок 3.88 - Развёрнутое меню раздела Помощь

Раздел Помощь содержит следующие команды:



- не используется;



- отображает диалоговое окно с информацией о программе (см. рис. 3.89).

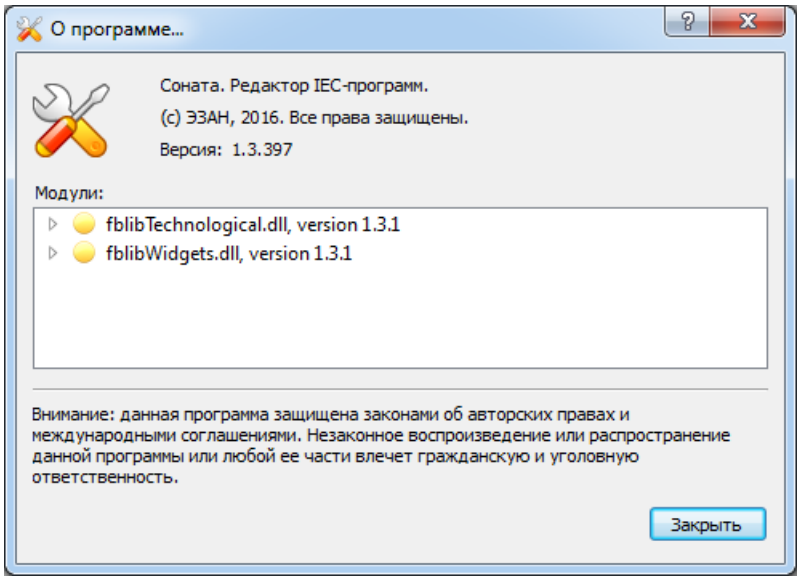


Рисунок 3.89 - Диалоговое окно с информацией о программе

3.3.3.2.3.2. Описание панели инструментов

Панель инструментов располагается под главным меню и содержит следующие кнопки:


("Сохранить") - команда для сохранения типа приложения (из раздела **Файл**);


("Назад") - отменить предыдущее действие по редактированию (из раздела **Редактирование**);


("Вперёд") - повторить ранее отменённое действие (из раздела **Редактирование**);


("Вырезать") - вырезать выделенный объект (объекты) и сохранить его в буфере памяти (из раздела **Редактирование**);


("Копировать") - копировать выделенный объект (объекты) в буфер памяти (из раздела **Редактирование**);


("Вставить") - вставить вырезанный или скопированный объект (объекты) из буфера памяти (из раздела **Редактирование**);


("Редактировать") - открыть тип функционального блока для просмотра или редактирования (из раздела **Редактирование**);


("Проверить и сохранить") - сохранить сразу все сделанные при редактировании функционального блока изменения (из раздела **Редактирование**);


("Отменить") - отменить сразу все сделанные при редактировании функционального блока изменения (из раздела **Редактирование**);

("Удалить") - удалить выделенный объект (объекты) (из раздела **Редактирование**);

("Запуск") - запустить приложение для отладки (из раздела **Запуск**);

("Останов") - остановить отладку приложения (из раздела **Запуск**).

3.3.3.2.3.3. Окно дерева объектов

На рис. 3.90 изображено окно дерева объектов приложения. Данное окно расположено в левой части окна редактора и открывается по нажатию на вкладку Объекты.

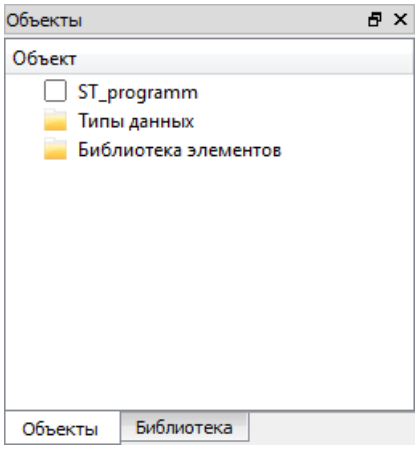


Рисунок 3.90 - Окно дерева объектов

Окно дерева объектов содержит узел редактируемого типа приложения. На рис. 3.90 узлом редактируемого типа приложения является узел **ST_programm**, где **ST_programm** - это имя приложения, заданное ранее в интерфейсе приложения. Именно в этом узле создаётся алгоритм приложения на языке ST, который может быть собран из различных функциональных элементов (функции на языке ST, функциональные блоки на языке ST).

Если выделить «мышью» узел редактируемого типа приложения и, вызвав контекстное меню (правой клавишей «мыши»), выбрать пункт **Свойства**, то откроется диалоговое окно редактирования свойств данного приложения (см. рис. 3.91).

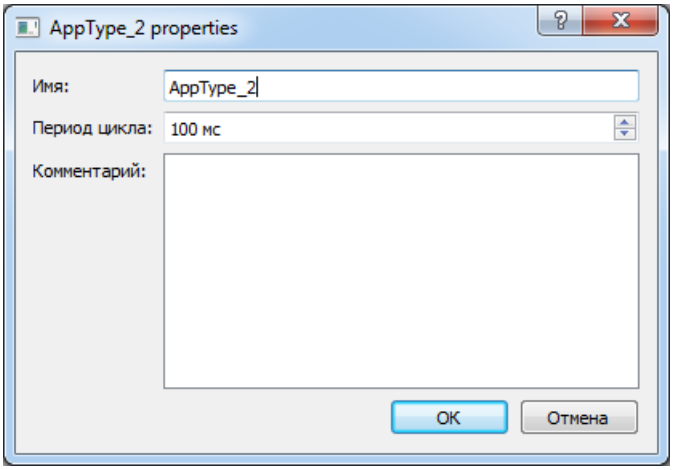


Рисунок 3.91 - Диалоговое окно редактирования свойств приложения

Описание полей ввода окна редактирования свойств приложения:

- **Имя** - имя приложения;

- **Период цикла** - время одного цикла ST программы;

- **Комментарий** - поле для полезных комментариев.

Папка Типы данных - это пользовательские типы данных, в которой хранятся созданные ранее в программе **ProjectManager** (Менеджер проектов) типы данных (см. раздел 3.2.2.7).

Папка Библиотека элементов содержит созданные в процессе работы с редактором функции и функциональные блоки.

Для создания функций и функциональных блоков на языке ST необходимо нажать правую клавишу «мыши» в окне дерева объектов редактора, вызвав тем самым контекстное меню (см. рис. 3.92), или воспользоваться разделом Редактировать главного меню редактора (см. раздел 3.3.3.2.3.1.2).

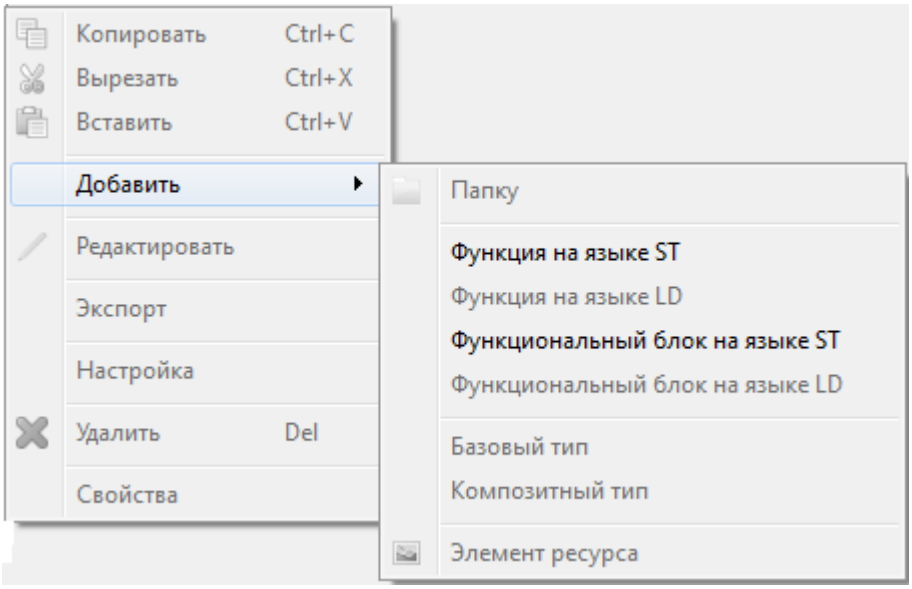


Рисунок 3.92 - Контекстное меню команд для дерева объектов

Группа команд контекстного меню аналогична группе команд раздела Редактировать главного меню редактора (см. раздел 3.3.3.2.3.1.2).

Если выбрать пункт **Добавить - Функция на языке ST**, то в папке **Библиотека элементов** будет создан элемент **NewFunction** - функция на языке ST (см. рис. 3.93). Имя функции можно будет изменить при редактировании.

Если выбрать пункт **Добавить - Функциональный блок на языке ST**, то в папке **Библиотека элементов** будет создан элемент **NewFB** - функциональный блок на языке ST (см. рис. 3.93). Имя функционального блока можно будет изменить при редактировании.

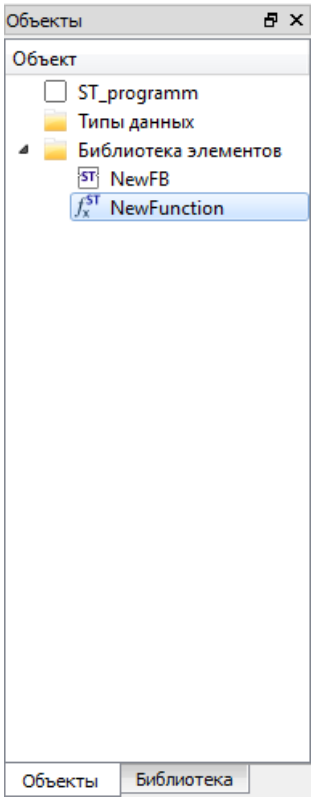


Рисунок 3.93 - Пользовательские функции и функциональные блоки в дереве объектов

Если выделить папку **Библиотека элементов** и выбрать пункт **Добавить - Папку**, то в папке **Библиотека элементов** будет создана вложенная папка. Имя данной папки можно изменить, если выделить данную папку, вызвать контекстное меню и выбрать пункт **Свойства**. При этом откроется диалоговое окно свойств папки (см. рис. 3.94).

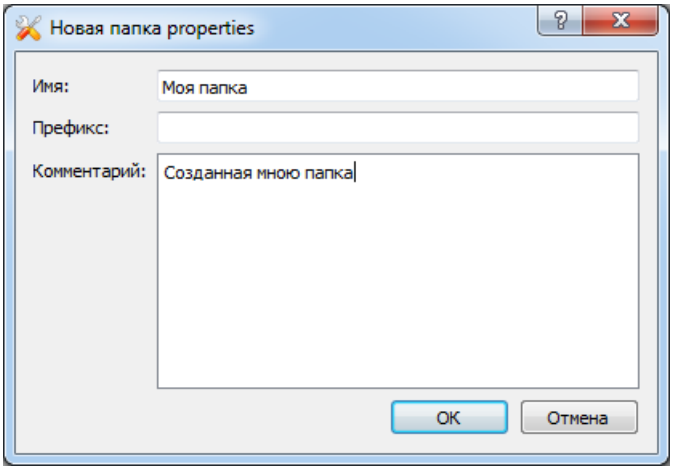


Рисунок 3.94 - Диалоговое окно свойств пользовательской папки

Диалоговое окно свойств папки содержит следующие поля ввода:

- **Имя** - задаётся имя папки;
- **Префикс** - задаётся префикс для имён элементов, которые будут созданы в этой папке;
- **Комментарий** - поле для ввода пользовательской информации.

Если выделить пользовательскую папку, вызвать контекстное меню и выбрать пункт **Добавить - Функция на языке ST**, то в пользовательской папке будет создан элемент функция на языке ST (см. рис. 3.95). Имя функции можно будет изменить при редактировании.

Если выделить пользовательскую папку, вызвать контекстное меню и выбрать пункт **Добавить - Функциональный блок на языке ST**, то в пользовательской папке будет создан элемент функциональный блок на языке ST (см. рис. 3.95). Имя функционального блока можно будет изменить при редактировании.

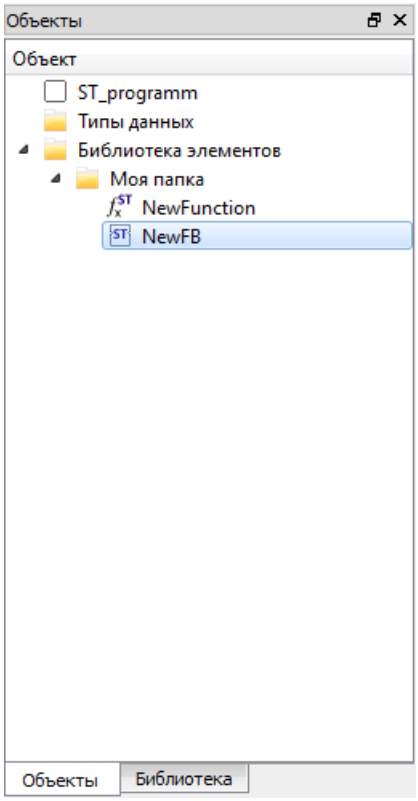


Рисунок 3.95 - Пользовательские функции и функциональные блоки в пользовательской папке

Если необходимо переместить созданный ранее элемент в пользовательскую папку, то необходимо выбрать данный элемент левой клавишей «мыши» и, не отпуская левую клавишу «мыши», переместить его в пользовательскую папку.

Если необходимо переместить созданный в пользовательской папке элемент в корень папки **Библиотека элементов**, то необходимо выбрать данный элемент левой клавишей «мыши» и, не отпуская левую клавишу «мыши», переместить его на узел **Библиотека элементов** в дереве объектов.

Удаление пользовательских папок и элементов осуществляется командой  , находящейся в разделе Редактировать главного меню, либо в панели инструментов, либо в контекстном меню дерева объектов.

3.3.3.2.3.4. Окно библиотеки типов функциональных блоков

На рис. 3.96 изображено окно библиотеки типов функциональных блоков приложения. Данное окно расположено в левой части окна редактора и открывается по нажатию на вкладку Библиотека.

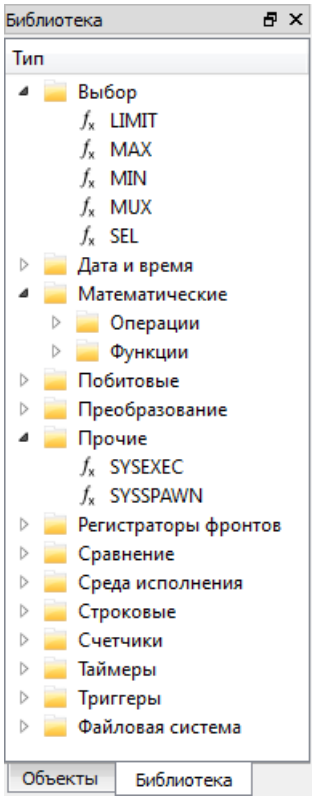


Рисунок 3.96 - Окно библиотеки типов функциональных блоков

В данном окне содержатся типы функциональных блоков, упорядоченные по группам.

3.3.3.2.3.5. Окно редактора алгоритмов, функций и функциональных блоков

В редакторе приложения **APPLICATION.ST.CONSOLE** можно создавать три вида элементов:

- узел основной программы-алгоритма приложения (не создаётся, а есть по умолчанию и имеет имя, которое было задано в интерфейсе приложения, см. раздел 3.3.3.2.3.5.3);
- функции на языке ST (см. раздел 3.3.3.2.3.5.2);
- функциональные блоки на языке ST (см. раздел 3.3.3.2.3.5.1).

3.3.3.2.3.5.1. Редактирование функционального блока на языке ST

Как создавать функциональные блоки, смотрите раздел 3.3.3.2.3.3.

Для редактирования функционального блока необходимо выделить данный блок и выполнить команду  ("Редактировать") из раздела Редактировать главного меню, либо из панели инструментов, либо из контекстного меню дерева объектов.

Для редактирования функционального блока в правой части редактора открывается окно редактирования, состоящее из двух вкладок:

- **Заголовок** - интерфейс функционального блока (см. рис. 3.97);
- **ST-программа** - алгоритм или код программы (см. рис. 3.98).

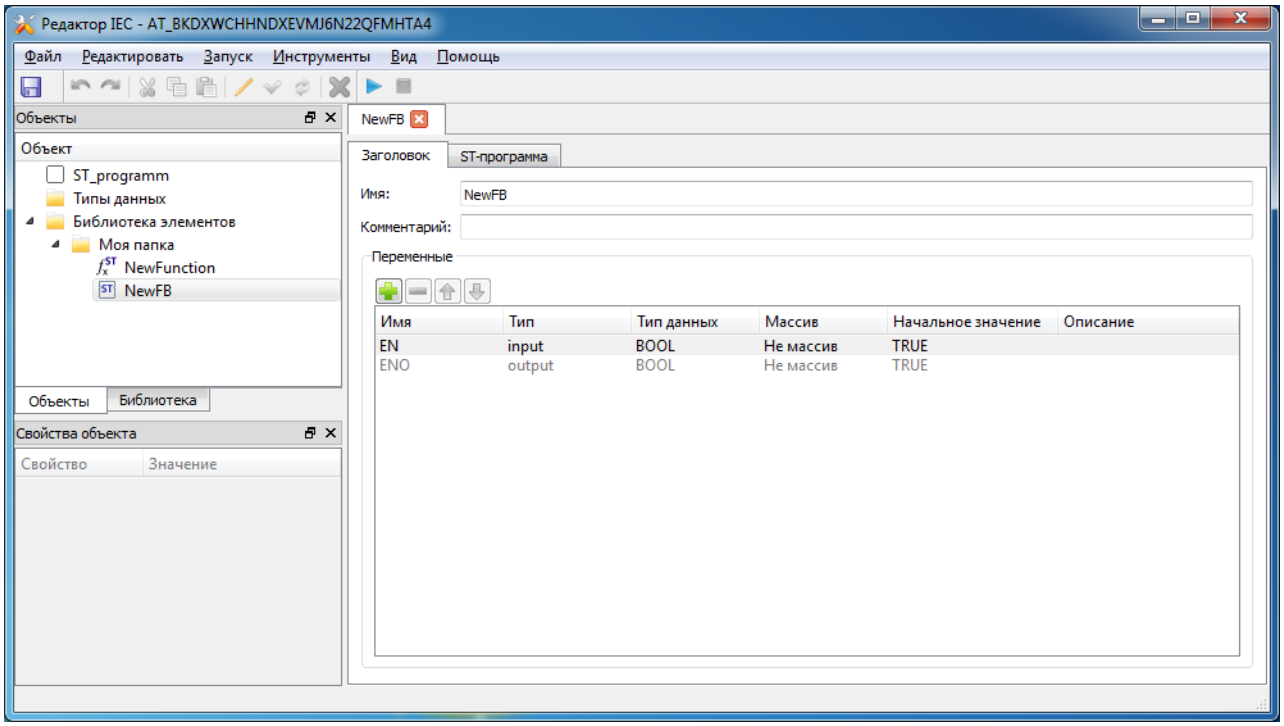


Рисунок 3.97 - Вкладка Заголовок окна редактирования функционального блока

Вкладка **Заголовок** состоит из следующих элементов:

- **Имя** - поле ввода имени функционального блока;

- **Комментарий** - пользовательское текстовое пояснение;

- **Таблица Переменные** - список переменных.

Таблица Переменные состоит из:

- **кнопки управления**:

 - добавить новую переменную;

 - удалить выделенные переменные;

 - переместить переменную вверх в таблице;

 - переместить переменную вниз в таблице.

- **столбцы таблицы**:

Имя - имя переменной;

Тип - типы переменных:

- **input** - входные;

- **output** - выходные;

- **inout** - входные и выходные;

- **internal** - внутренние.

Тип данных - тип данных переменной (типы данных, используемые в языке ST);

Массив - указать, является ли переменная массивом и, если является, то указать длину массива;

Начальное значение - значение, которое присваивается переменной при инициализации приложения;

Описание - текстовая строка, которую можно использовать для описания переменной.

Для редактирования значений столбцов таблицы переменных используется двойной щелчок левой клавиши «мыши» по области редактирования.

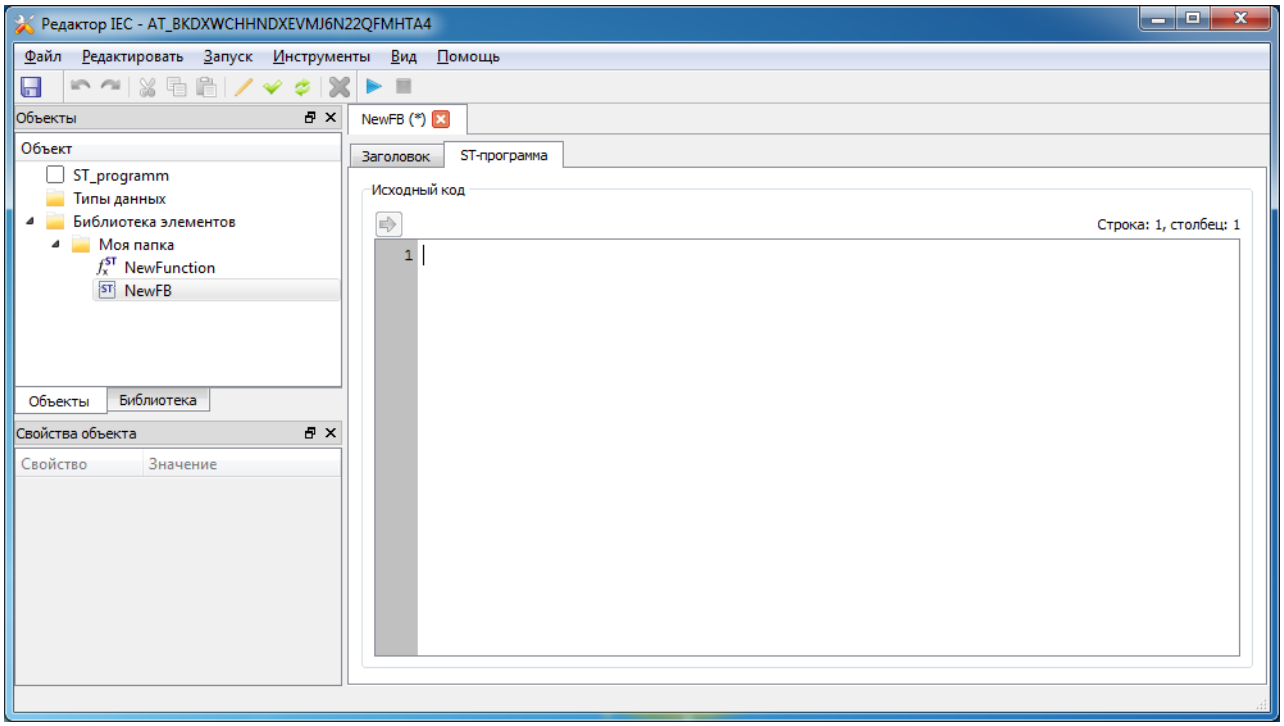
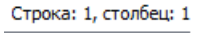


Рисунок 3.98 - Вкладка ST-программа окна редактирования функционального блока

Вкладка **ST-программа** состоит из:

 - кнопка проверки алгоритма;

 - область, отображающая номер строки и столбца, где располагается курсор;

- область написания кода на языке ST.

Закрытие окна редактора производится нажатием на кнопку  окна редактора функционального блока. Если никаких изменений не было, то окно редактора закроется, если же были внесены изменения, то будет выведено диалоговое окно сохранения изменений при выходе из окна редактирования функционального блока (см. рис. 3.99).

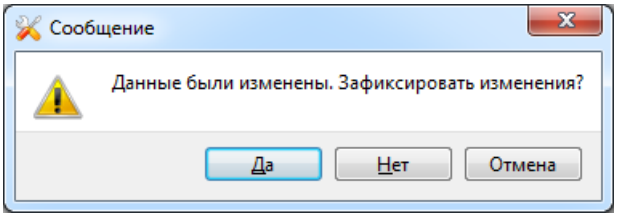
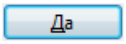


Рисунок 3.99 - Диалоговое окно сохранения изменений при выходе из окна редактирования функционального блока

По нажатию на кнопку  изменения будут сохранены и будет закрыто окно редактирования функционального блока. По нажатию на кнопку  изменения не будут сохранены, но окно редактирования функционального блока будет закрыто. По нажатию на кнопку  изменения не сохраняются и окно редактирования функционального блока не закрывается.

3.3.3.2.3.5.2. Редактирование функции на языке ST

Описание редактирования функции на языке ST аналогично описанию редактирования функционального блока на языке ST (см. раздел 3.3.3.2.3.5.1).

Отличие состоит во вкладке **Заголовок**.

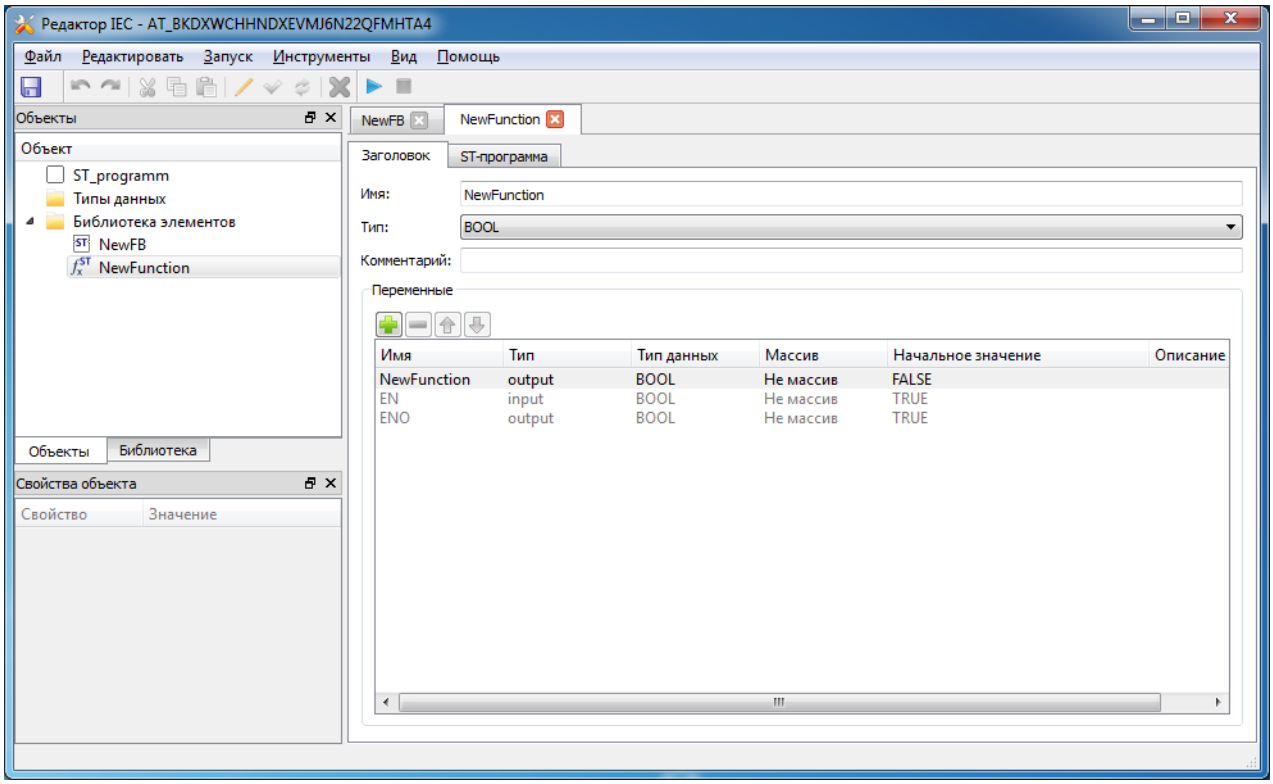


Рисунок 3.100 - Вкладка Заголовок окна редактирования функции на языке ST

На рис. 3.100 между полями ввода **Имя** и **Комментарий** находится поле выбора **Тип**, в котором указывается тип возвращаемого функцией значения.

3.3.3.2.3.5.3. Редактирование основного алгоритма приложения

Основной алгоритм приложения является верхним узлом в дереве объектов редактора приложения **APPLICATION.ST.CONSOLE**.

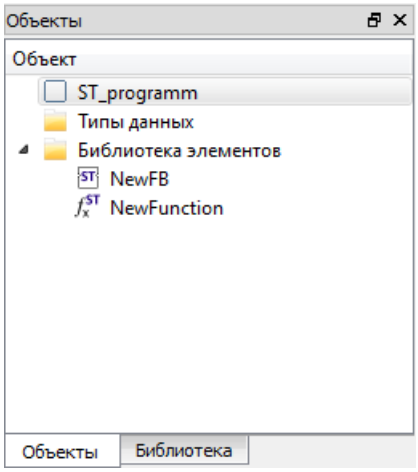


Рисунок 3.101 - Узел основной программы в дереве объектов

На рис. 3.101 узлом основной программы приложения является узел **ST_Programm**. При работе приложения **APPLICATION.ST.CONSOLE** в составе проекта будет работать алгоритм, написанный в данном узле.

Для редактирования функционального блока необходимо выделить данный блок и выполнить команду  ("Редактировать") из раздела Редактировать главного меню, либо из панели инструментов, либо из контекстного меню дерева объектов. По выполнению данной команды в правой части редактора приложения откроется редактор алгоритма узла главной программы (см. рис. 3.102).

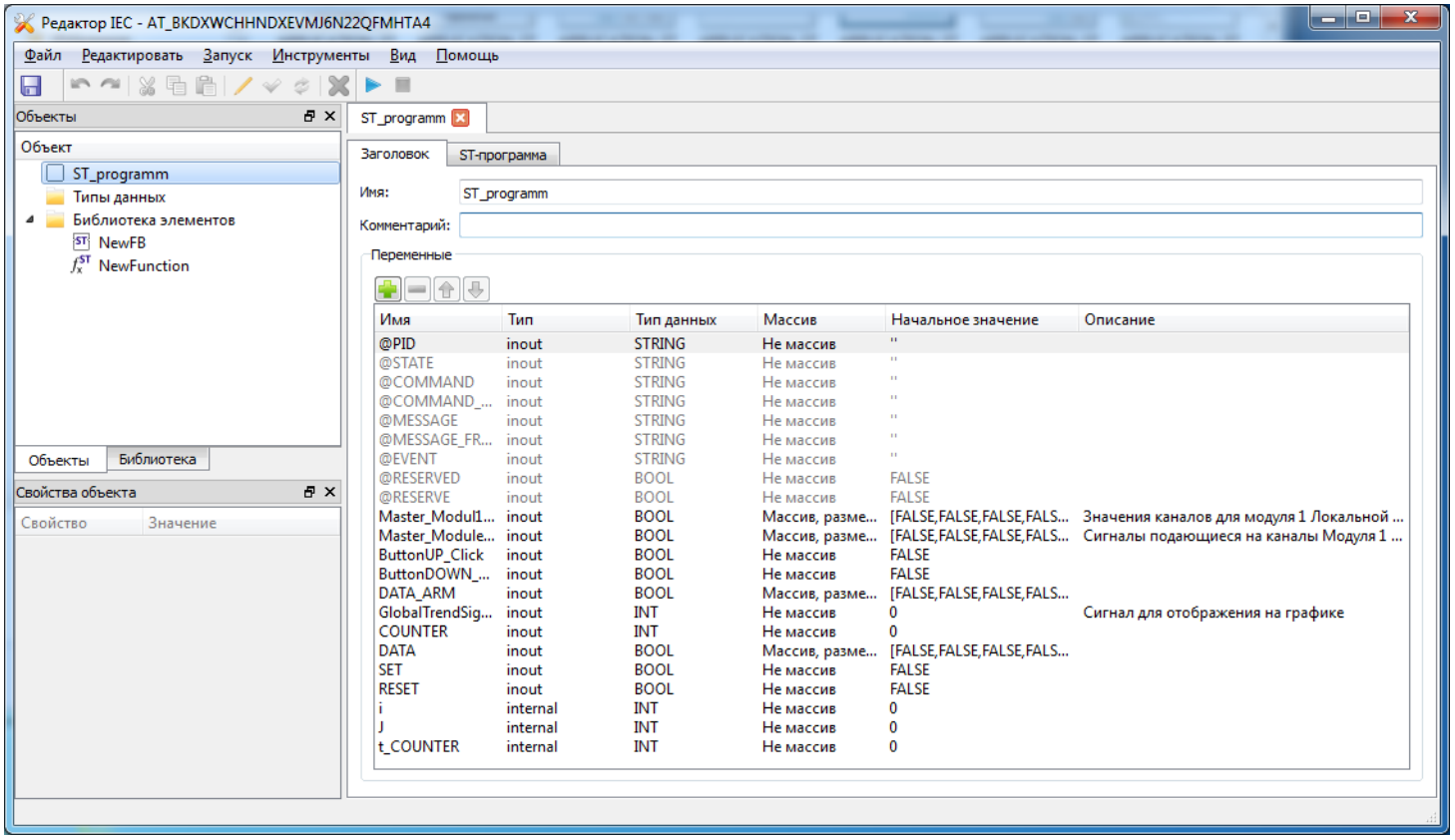


Рисунок 3.102 - Окно редактора основного алгоритма приложения

Описание окна редактора основного алгоритма приложения аналогично описанию окна редактора функционального блока (см. раздел 3.3.3.2.3.5.1).

Единственной особенностью редактора основного алгоритма приложения является то, что в закладке **Заголовок** в **Таблице Переменные** отображаются системные сигналы приложения, а также глобальные локальные сигналы приложения, которые были ранее добавлены в интерфейс приложения **APPLICATION.ST.CONSOLE**.

На рис. 3.103 изображена закладка ST-программа окна редактора основного алгоритма приложения.

Описание языка программирования ST смотрите в документе SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33.

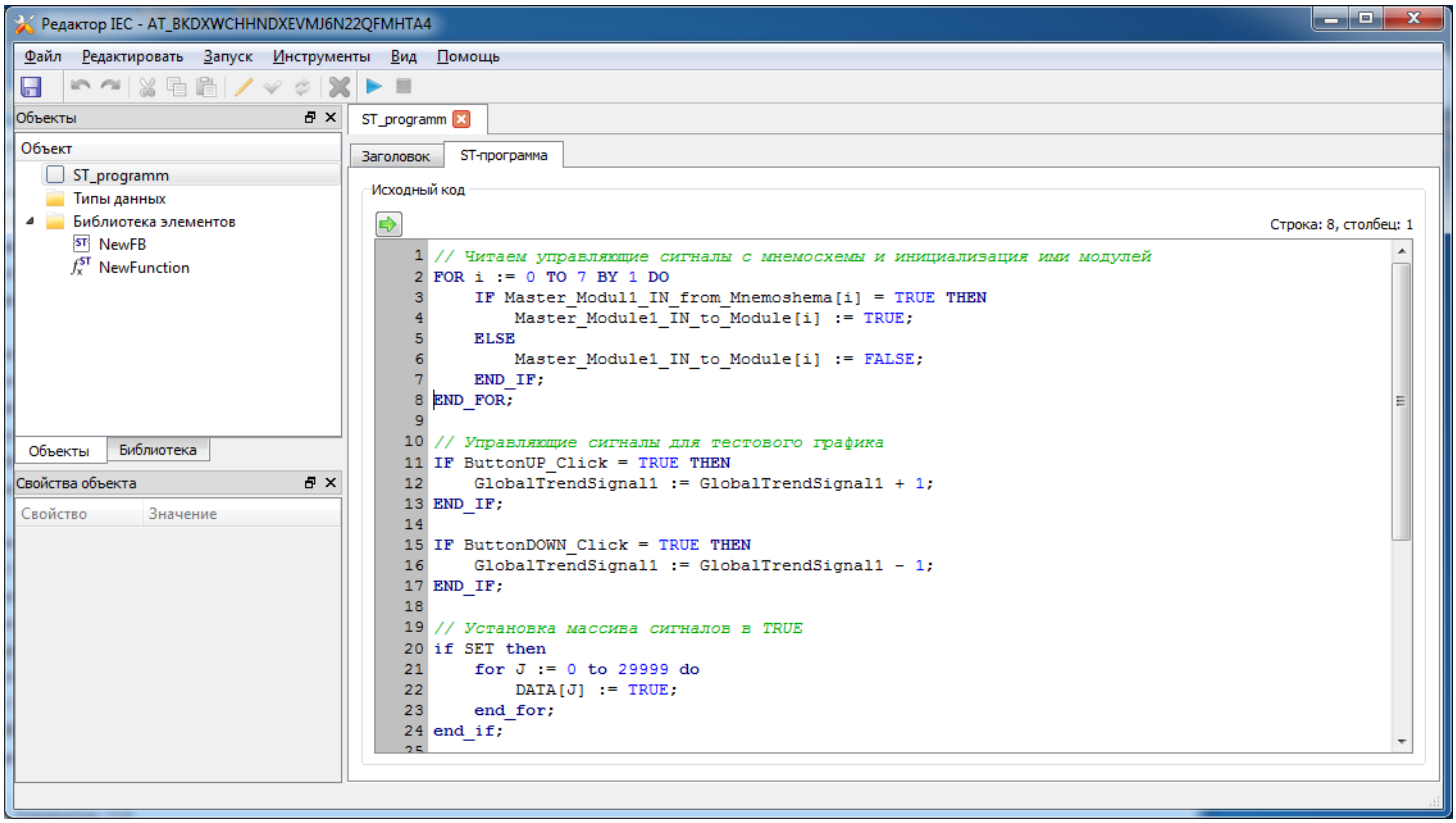


Рисунок 3.103 - Закладка ST-программа окна редактора основного алгоритма приложения

3.3.3.2.3.6. Окно списка сообщений

Окно списка сообщений расположено в нижней части главного окна редактора и выводится при проверке типа функционального блока, а также при открытии типа, если в ходе загрузки данных были обнаружены какие-либо ошибки. Окно списка сообщение приведено на рис. 3.104.

Рисунок 3.104 - Окно списка сообщений

На панели инструментов окна **Сообщения** располагаются следующие кнопки:

 ("Сохранить") - сохранить сообщения в файл;

 ("Очистить") - очистить все сообщения в окне;

Строки окна списка сообщений помечаются иконками разного цвета. Цвет зависит от категории сообщения:

(зеленый) – информационное сообщение, не влияет на работоспособность приложения;

(желтый) – предупреждение, на сообщения данного типа необходимо обращать внимание, поскольку они могут содержать сообщения о некритических ошибках – приложение с такими ошибками может работать;

(красный) – критические ошибки – приложение не будет работать.

На панели инструментов находятся кнопки, с помощью которых можно фильтровать сообщения в окне:

- оставить информационные сообщения;
- оставить предупреждения;
- оставить критические ошибки.

3.3.3.2.3.7. Окно списка событий

Окно списка событий расположено в нижней части главного окна редактора и выводится при запуске отладки типа приложения. Окно списка сообщение приведено на рис. 3.105.

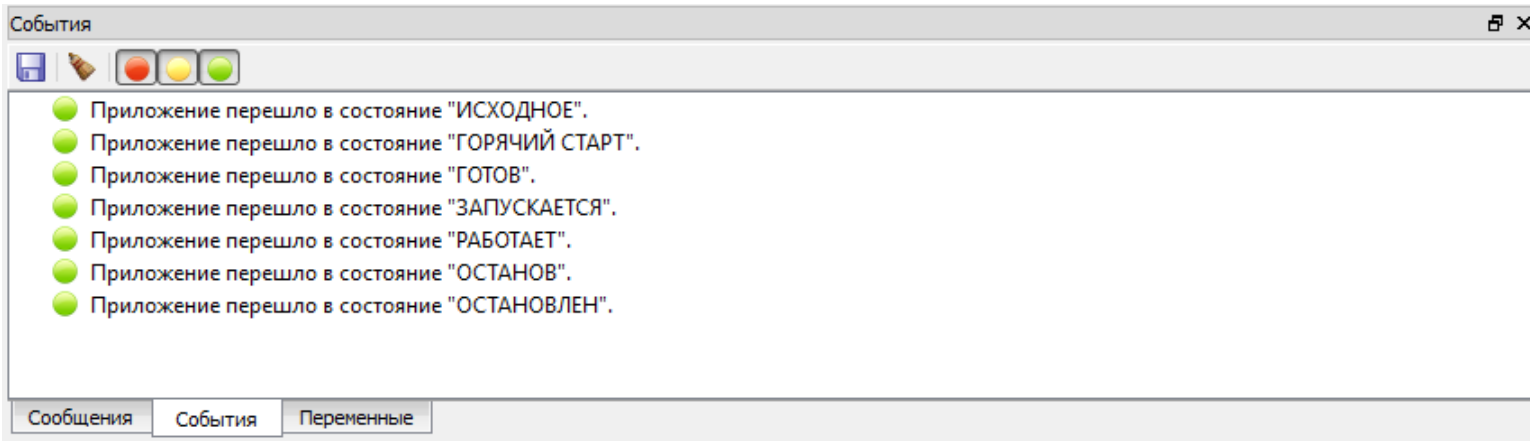


Рисунок 3.105 - Окно списка событий

На панели инструментов окна **События** располагаются следующие кнопки:

("Сохранить") - сохранить события в файл;

("Очистить") - очистить все события в окне.

Строки окна списка событий помечаются иконками разного цвета. Цвет зависит от категории события:

(зеленый) – информационное событие;

(желтый) – предупреждение;

(красный) – фатальное событие.

На панели инструментов находятся кнопки, с помощью которых можно фильтровать события в окне:

- оставить информационные события;
- оставить предупреждения;
- оставить фатальные события.

События можно разделить на две группы:

- события, формируемые самой системой;
- события, формируемые разработанными приложениями.

3.3.3.2.3.8. Окно переменных отладчика

Окно переменных отладчика расположено в нижней части главного окна редактора и выводится при запуске отладки типа приложения. Окно содержит список переменных, значения которых необходимо отслеживать и изменять в ходе отладки. Список переменных формируется оператором. Окно переменных отладчика приведено на рис. 3.106.

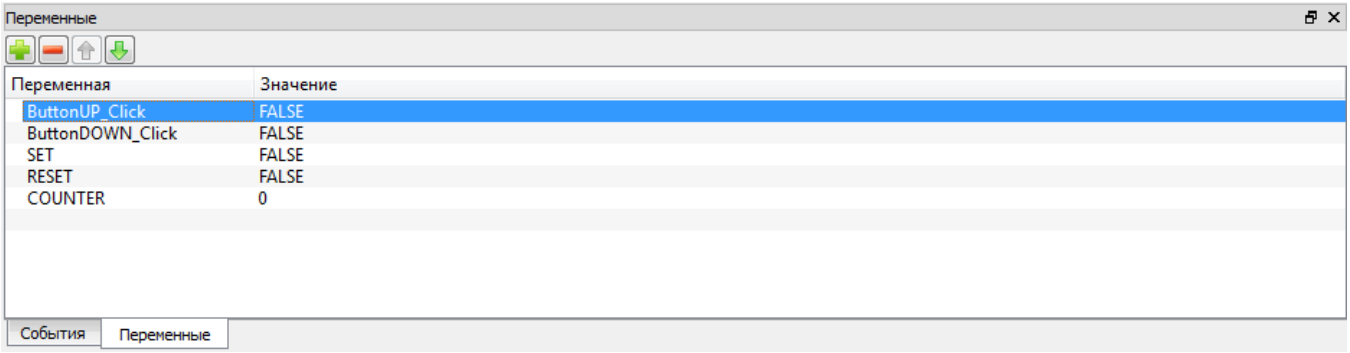


Рисунок 3.106 - Окно переменных отладчика

3.3.3.2.3.9. Меню настроек по работе с редактором

По нажатию в разделе Инструменты главного меню редактора на команду Настройки откроется диалоговое окно настроек по работе с редактором (см. рис. 3.107).

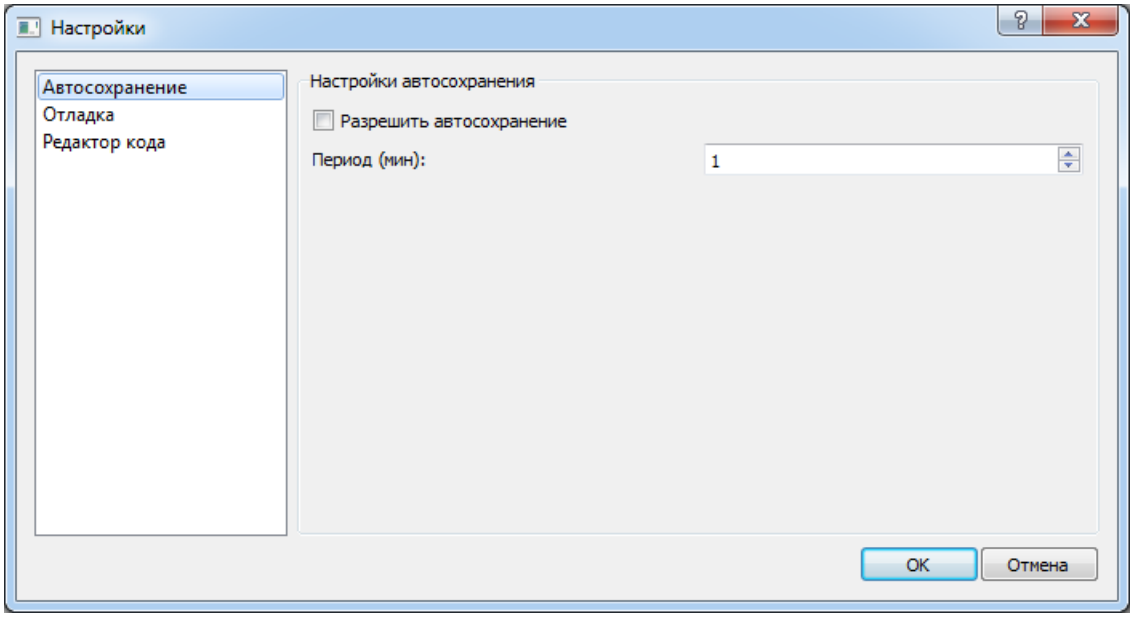


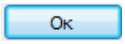
Рисунок 3.107 - Диалоговое окно настроек по работе с редактором

В левой части диалогового окна содержатся три пункта на выбор для настроек работы с редактором:

- **Автосохранение** - настройки автоматического сохранения изменений при работе в редакторе (см. раздел 3.3.3.2.3.9.1);
- **Отладка** - (см. раздел 3.3.3.2.3.9.2);

- **Редактор кода** - настройки для страницы с кодом алгоритма на языке ST (см. раздел 3.3.3.2.3.9.3).

В нижней части диалогового окна находятся следующие кнопки:



- принять изменения настроек и закрыть диалоговое окно;

- отменить изменения настроек и закрыть диалоговое окно.

3.3.3.2.3.9.1. Описание настроек Автосохранения

На рис. 3.107 открыт пункт настроек автосохранения изменений при работе в редакторе.

В правой части диалогового окна содержатся следующие кнопки и поля настроек:

- выполнять автоматическое сохранение изменений или не выполнять;

- период, через который происходит автосохранение изменений.

3.3.3.2.3.9.2. Описание настроек Отладки приложения

На рис. 3.108 открыт пункт настроек отладки приложения.

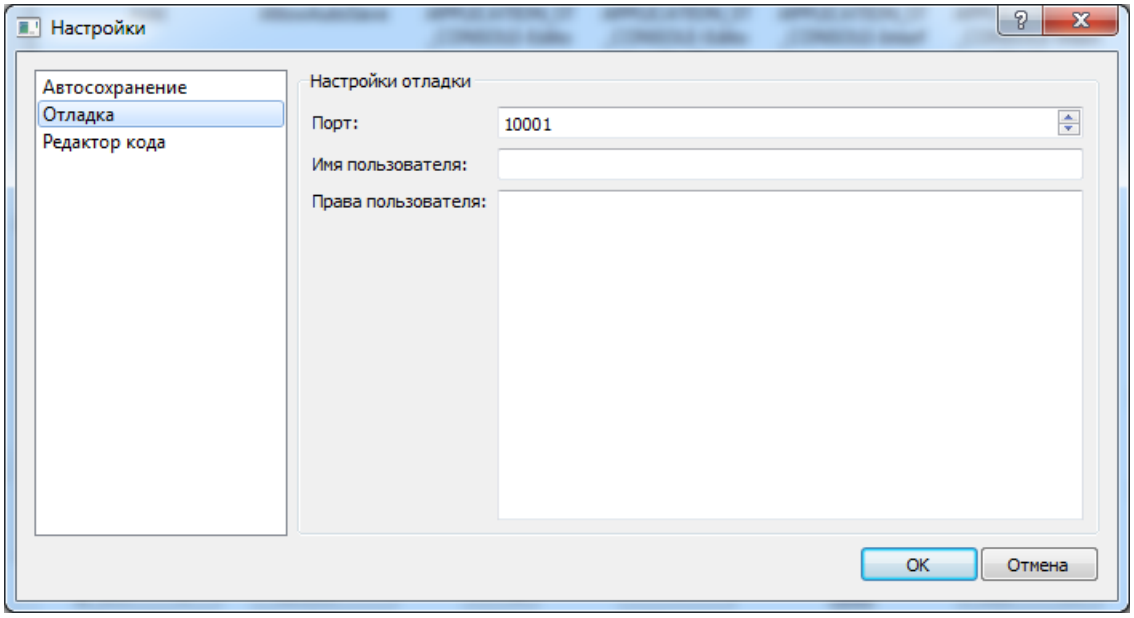


Рисунок 3.108 - Диалоговое окно настроек отладки приложения

Поле ввода порта предназначено для редактирования номера IP-порта, который будет использоваться запущенным приложением для взаимодействия с другими модулями системы.

Имя пользователя и его права не используются при отладке консольных приложений.

3.3.3.2.3.9.3. Описание настроек Редактора кода

На рис. 3.109 открыт пункт настроек страницы алгоритма на языке ST.

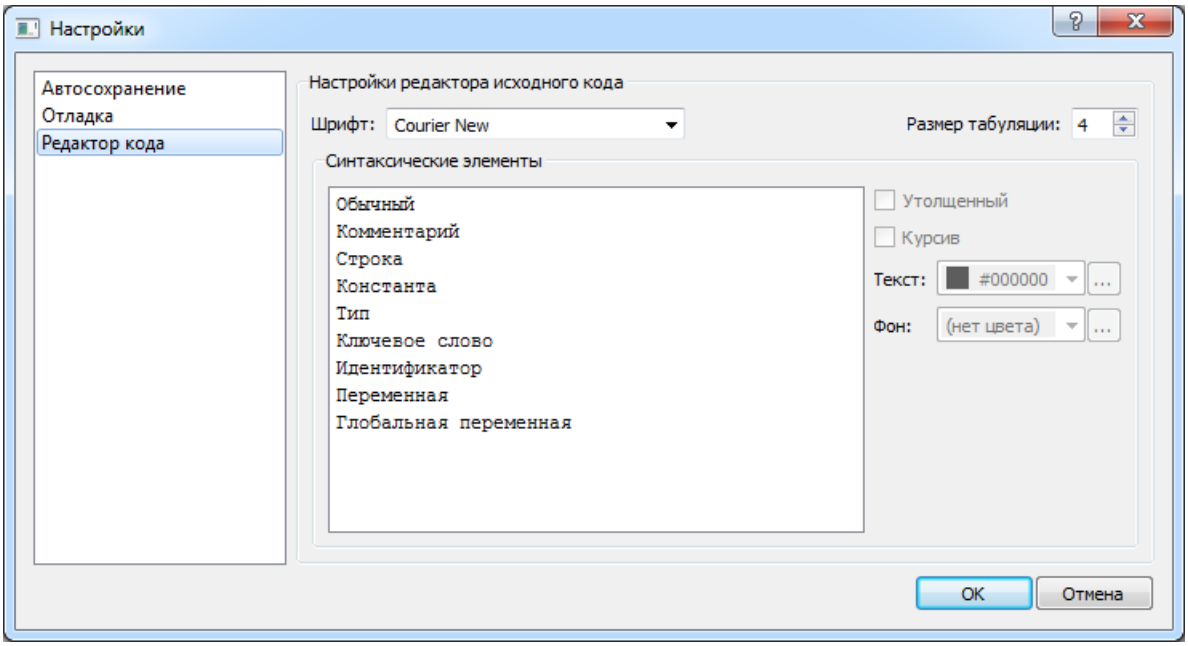
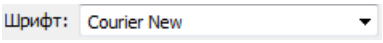
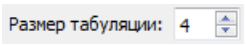


Рисунок 3.109 - Диалоговое окно настроек страницы алгоритма на языке ST

В верхней части диалогового окна находятся следующие поля и пункты настроек:

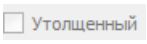


- поле выбора шрифта отображения набранного текста;

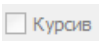


- размер отступа при нажатии на клавишу Tab на клавиатуре.

В правой части диалогового окна находятся настройки для отображения отдельных частей текста алгоритма, который разделён на группы (описание текстовых групп следует позже):



- утолщенный шрифт;



- наклонный шрифт;

- цвет шрифта;

- цвет заднего фона.

Группы текста разделяются на следующие:
- **Обычный** - основной текст алгоритма;
- **Комментарий** - текст, который по правилам языка ST является комментарием;
- **Строка** - текст, являющийся значением переменной STRING в языке ST;
- **Константа** - текст, описывающий константу;
- **Тип** - текст, описывающий тип;
- **Ключевое слово** - текст, описывающий ключевые слова;
- **Идентификатор** - текст, описывающий идентификатор;
- **Переменная** - текст, описывающий имя переменной;
- **Глобальная переменная** - текст, описывающий имя глобальной переменной.

3.3.3.2.4. Применение приложения APPLICATION.ST.CONSOLE

Все внесённые изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

Для использования данного приложения в проекте его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.3.3. Описание приложения APPLICATION.LUA

Приложение **APPLICATION.LUA** применяется для написания алгоритмов на различных языках программирования (на данный момент поддерживается язык LUA). Описание языка приведено в документе SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33. В данном разделе приведено описание редактора по созданию алгоритмов на примере языка LUA.

3.3.3.3.1. Создание приложения APPLICATION.LUA

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению, осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.110 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **APPLICATION.LUA**. Затем в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **InitBPR_LUA** (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

Также на рис. 3.110 в интерфейс приложения добавлены сигналы SCADA-системы "Соната", такие как GSTRING, main_in_work, reserve_in_work и другие (о создании и добавлении сигналов см. раздел 3.2.2.4.3).

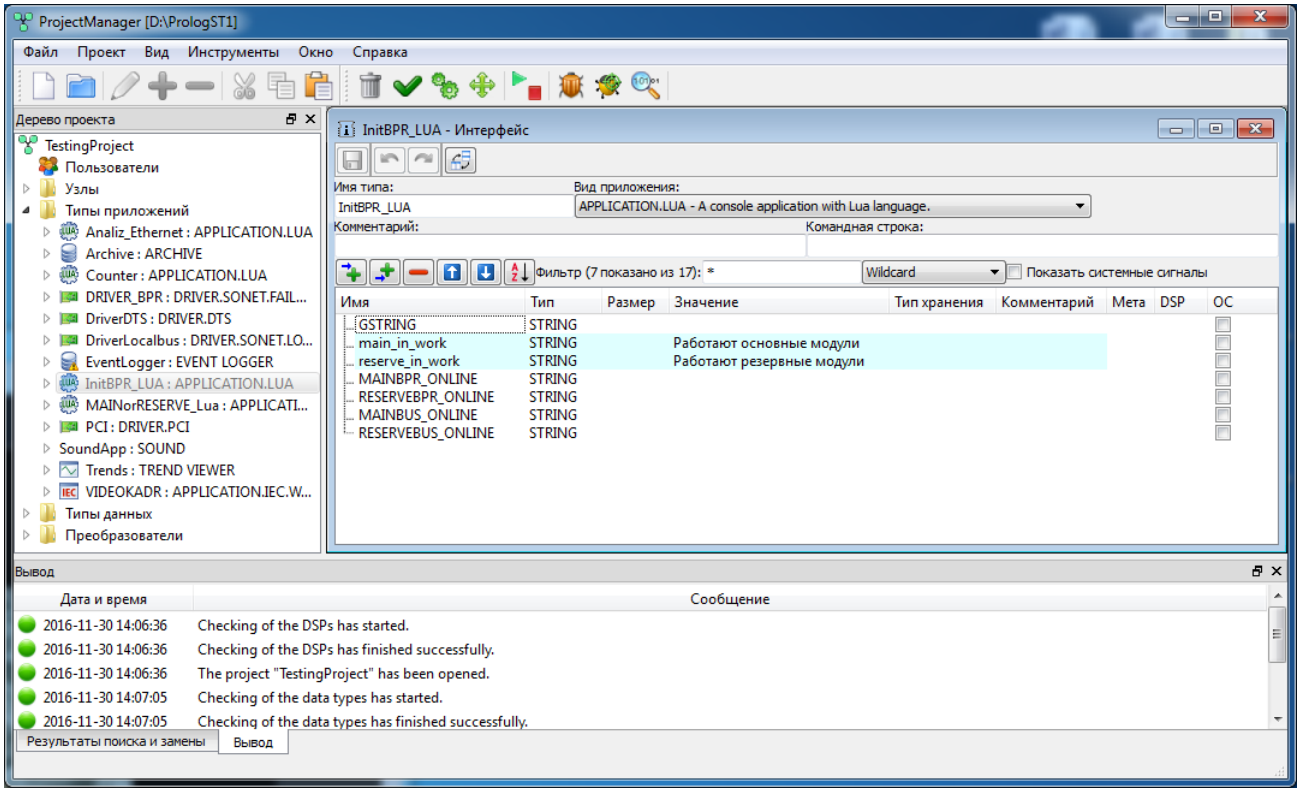


Рисунок 3.110 - Интерфейс приложения APPLICATION.LUA

По окончании редактирования интерфейса типа приложения необходимо сохранить изменения с помощью кнопки  («Сохранить»), после чего рекомендуется закрыть диалог редактирования крестиком  в верхнем правом углу окна.

3.3.3.3.2. Вызов специализированного редактора для написания алгоритмов

В дереве проекта тип приложения отображается в виде двух позиций: верхняя позиция – интерфейс приложения, вложенная позиция – специализированный редактор приложения (далее редактор) или его конфигурационный файл. Процесс вызова редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

На рис. 3.111 изображено окно специализированного редактора для написания алгоритмов.

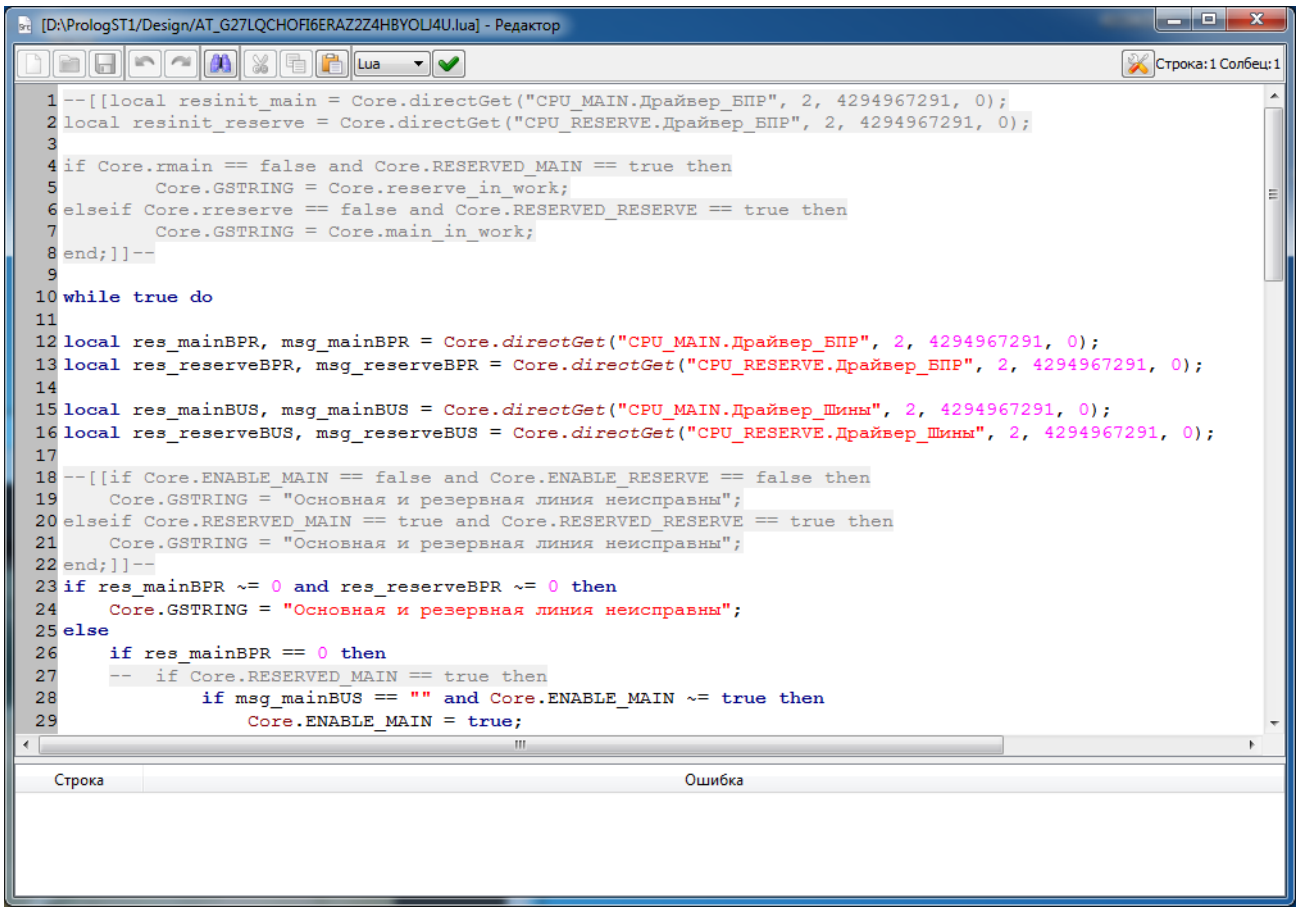


Рисунок 3.111 - Окно специализированного редактора для написания алгоритмов

3.3.3.3.3. Описание интерфейса окна специализированного редактора для написания алгоритмов на языке LUA

Окно специализированного редактора для написания алгоритмов содержит элементы, необходимые для написания и проверки алгоритмов. Внешний вид окна приведен на рис. 3.111.

Окно редактора содержит следующие элементы:

- в верхней левой части полный путь к конфигурационному файлу и его уникальное имя **AT_****;

- в верхней правой части находятся кнопки управления окном редактора  , соответственно свернуть, развернуть и закрыть окно редактора;

- далее ниже располагается панель инструментов;

- основную часть окна занимает область для написания алгоритма;

- в нижней части располагается окно информационных сообщений об ошибках и предупреждениях.

На панели инструментов располагаются следующие кнопки:

 ("Новый") - создать новый файл;

 ("Открыть") - открыть файл;

 ("Сохранить") - сохранить в файл;

 ("Отменить") - отменить последнее действие;

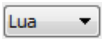
 ("Повторить") - повторить последнее действие;

 ("Найти/Заменить") - меню поиска и замены;

 ("Вырезать") - вырезать выделенный элемент;

 ("Копировать") - скопировать выделенный элемент;

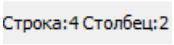
 ("Вставить") - вставить из буфера;



("Выбор языка программирования") - выбрать язык программирования для проверки;

 ("Проверить") - проверка синтаксиса;

 ("Настройки") - настройки редактора;



("Информационная строка") - информация о месте положении курсора в формате номер строки и номер столбца.

По нажатию на кнопку  ("Найти/Заменить") под панелью инструментов открывается дополнительное меню (см. рис. 3.112).

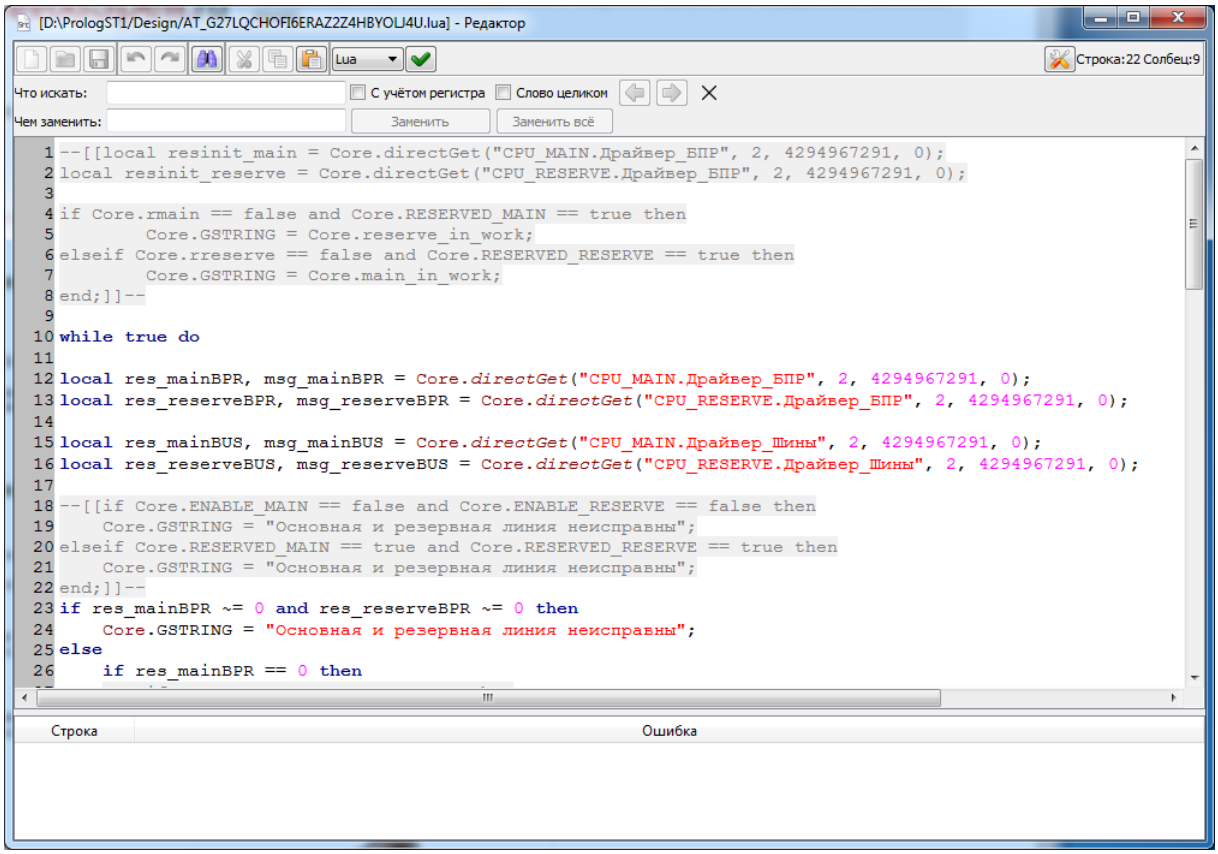


Рисунок 3.112 - Меню поиска и замены в редакторе алгоритмов

Меню поиска и замены состоит из следующих элементов:

Что искать: - поле ввода ключевого слова для поиска;

Чем заменить: - поле ввода слова, на которое будет заменён найденный элемент;

С учётом регистра - поиск будет осуществляться с учётом регистра;

Слово целиком - поиск будет осуществляться целиком введённого в строку поиска слова;

- шаг поиска в обратную сторону;

- шаг поиска вперёд по тексту;

- закрыть панель поиска и замены;

Заменить - выполнить замену найденного слова;

Заменить всё - выполнить замену всех найденных слов автоматически.

По нажатию на кнопку ("Проверить") выполняется проверка набранного алгоритма. По результатам проверки в окне информационных сообщений об ошибках и предупреждениях выводятся результаты проверки (см. рис. 3.113).

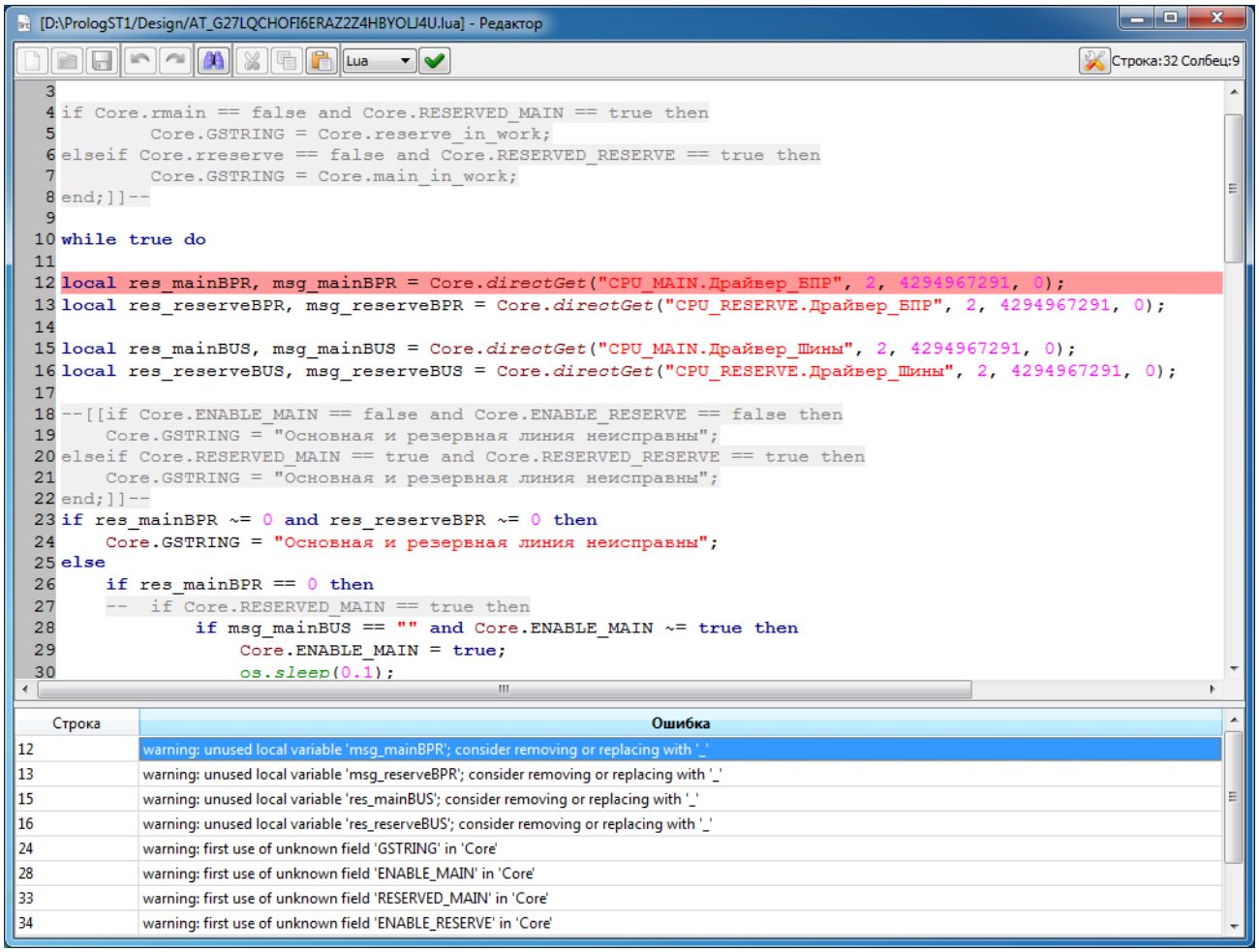


Рисунок 3.113 - Окно редактора алгоритмов и сообщения после проверки алгоритма

По нажатию на кнопку  ("Настройки") открывается диалоговое окно настроек (см. рис. 3.114).

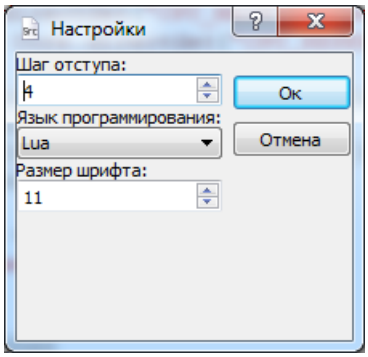
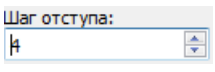
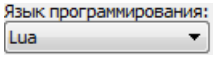


Рисунок 3.114 - Диалоговое окно настроек редактора алгоритмов

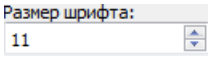
Диалоговое окно **Настройки** состоит из следующих элементов:



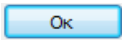
- шаг отступа при нажатии на клавишу **Tab** на клавиатуре;



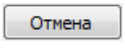
- выбор языка программирования;



- размер шрифта в окне редактора;



- принять изменения настроек и закрыть диалоговое окно;



- отказаться от изменений настроек и закрыть диалоговое окно.

По завершению написания и проверки алгоритма необходимо нажать на кнопку  ("Сохранить") и закрыть окно редактора кнопкой  .

Приложение **APPLICATION.LUA** настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.4. Графические приложения

3.3.4.1. Описание приложения APPLICATION.WINDOW

Приложение **APPLICATION.WINDOW** является графическим управляемым приложением SCADA-системы "Соната". Данное приложение используется для запуска графического приложения, написанного вне среды разработки SCADA-системы "Соната", но с помощью библиотек SCADA-системы "Соната".

3.3.4.1.1. Создание приложения APPLICATION.WINDOW

Для создания приложения **APPLICATION.WINDOW** сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.115 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** необходимо выбрать **APPLICATION.WINDOW**.

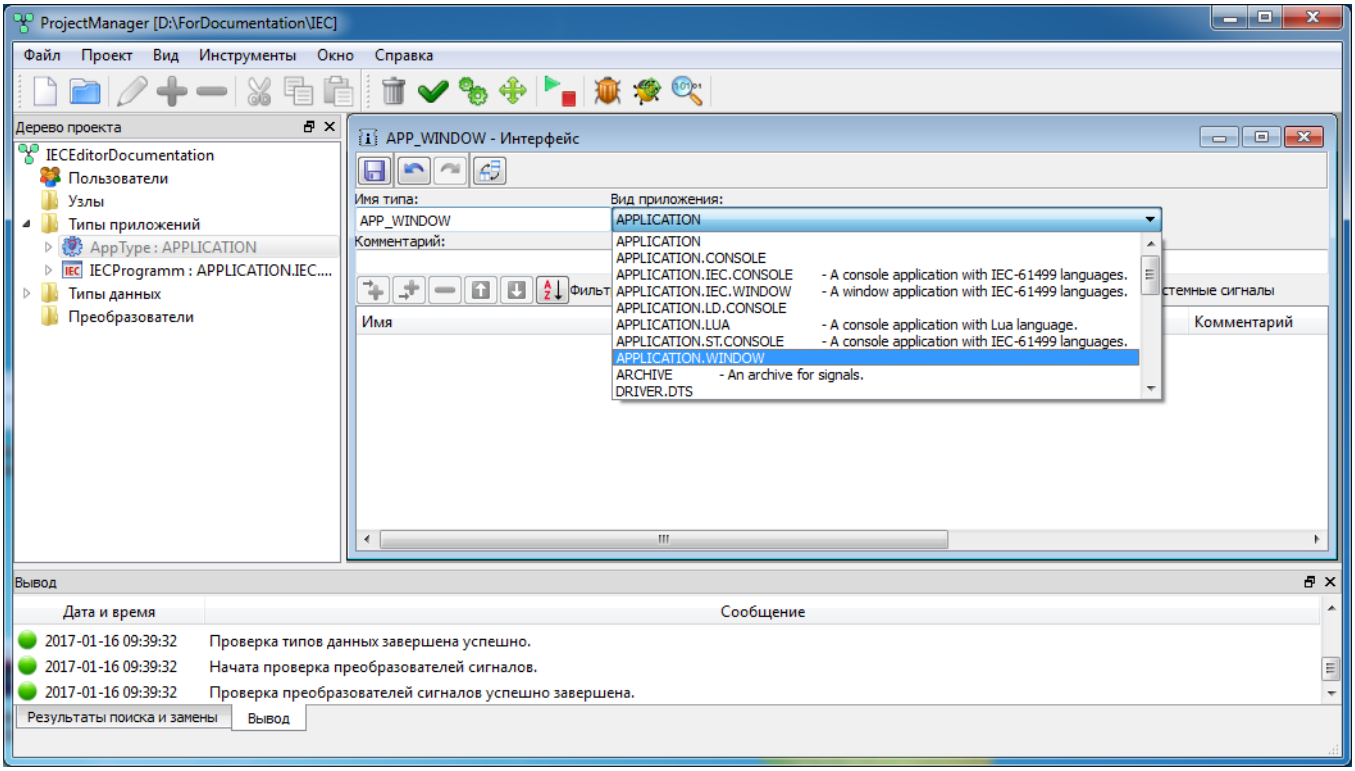


Рисунок 3.115 - Изменение вида приложения на APPLICATION.WINDOW

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.116) – для подтверждения изменения необходимо нажать кнопку Да.

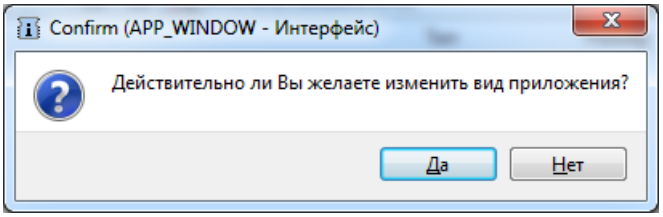


Рисунок 3.116 - Окно подтверждения для изменения вида типа приложения

3.3.4.1.2. Настройка интерфейса приложения APPLICATION.WINDOW

Основная настройка выполняется в поле **Командная строка**.

На рис. 3.117 в поле ввода **Командная строка** прописан исполняемый файл графического приложения **EventViewer**, который будет запускаться при старте приложения **APPLICATION.WINDOW**. Путь к исполняемому файлу прописывается относительно расположения файлов проекта.

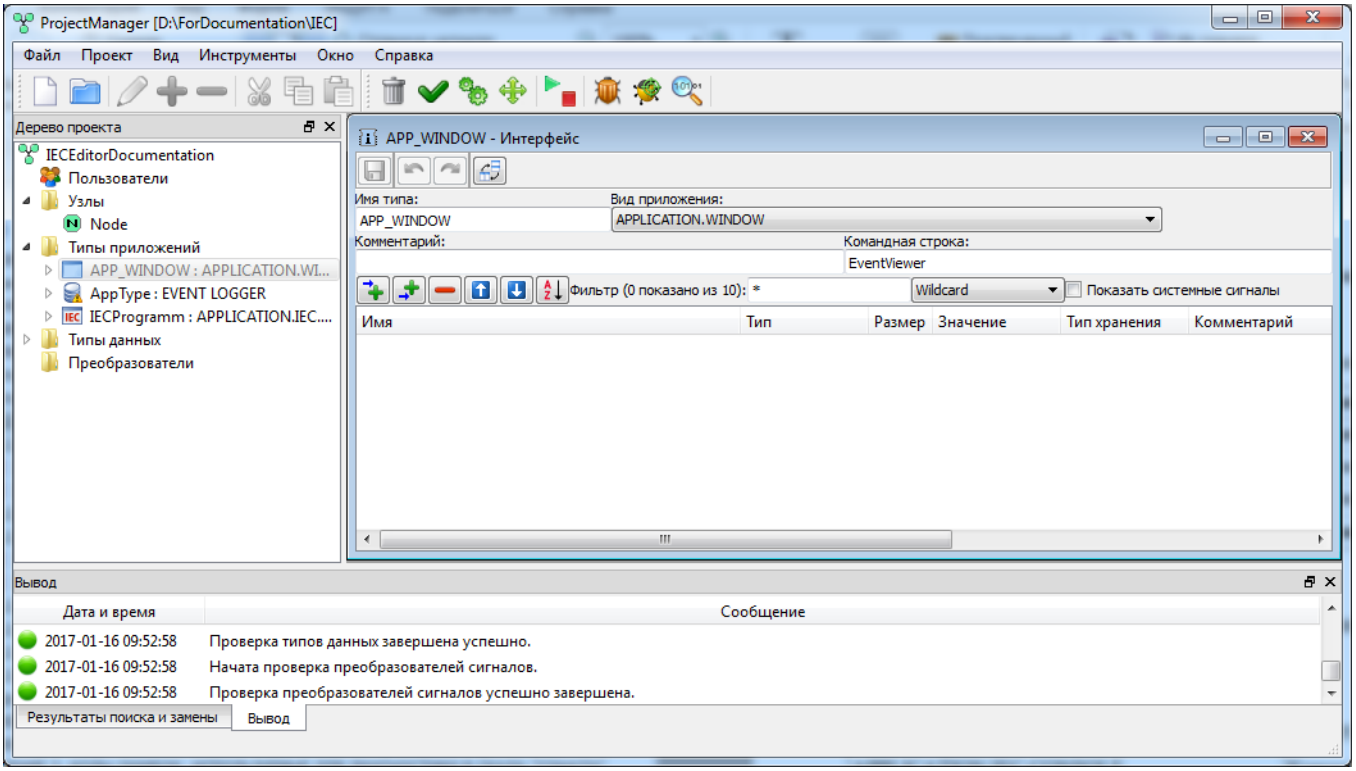


Рисунок 3.117 - Окно интерфейса приложения APPLICATION.WINDOW

Изменения следует зафиксировать с помощью кнопки ("Сохранить") в окне редактирования интерфейса типа приложения.

Для использования данного приложения в проекте, его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.4.2. Описание приложения APPLICATION.IEC.WINDOW

Приложение **APPLICATION.IEC.WINDOW** предназначено для создания графических приложений, основанных на стандартах IEC 61131 и IEC 61499.

3.3.4.2.1. Создание приложения APPLICATION.IEC.WINDOW

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению, осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.118 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **APPLICATION.IEC.WINDOW**. Затем в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **Proba** (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

Также на рис. 3.118 в интерфейс приложения добавлены сигналы SCADA-системы "Соната", такие как Trend, ButtonUP_Click, ButtonDOWN_CLick и другие (о создании и добавлении сигналов см. раздел 3.2.2.4.3).

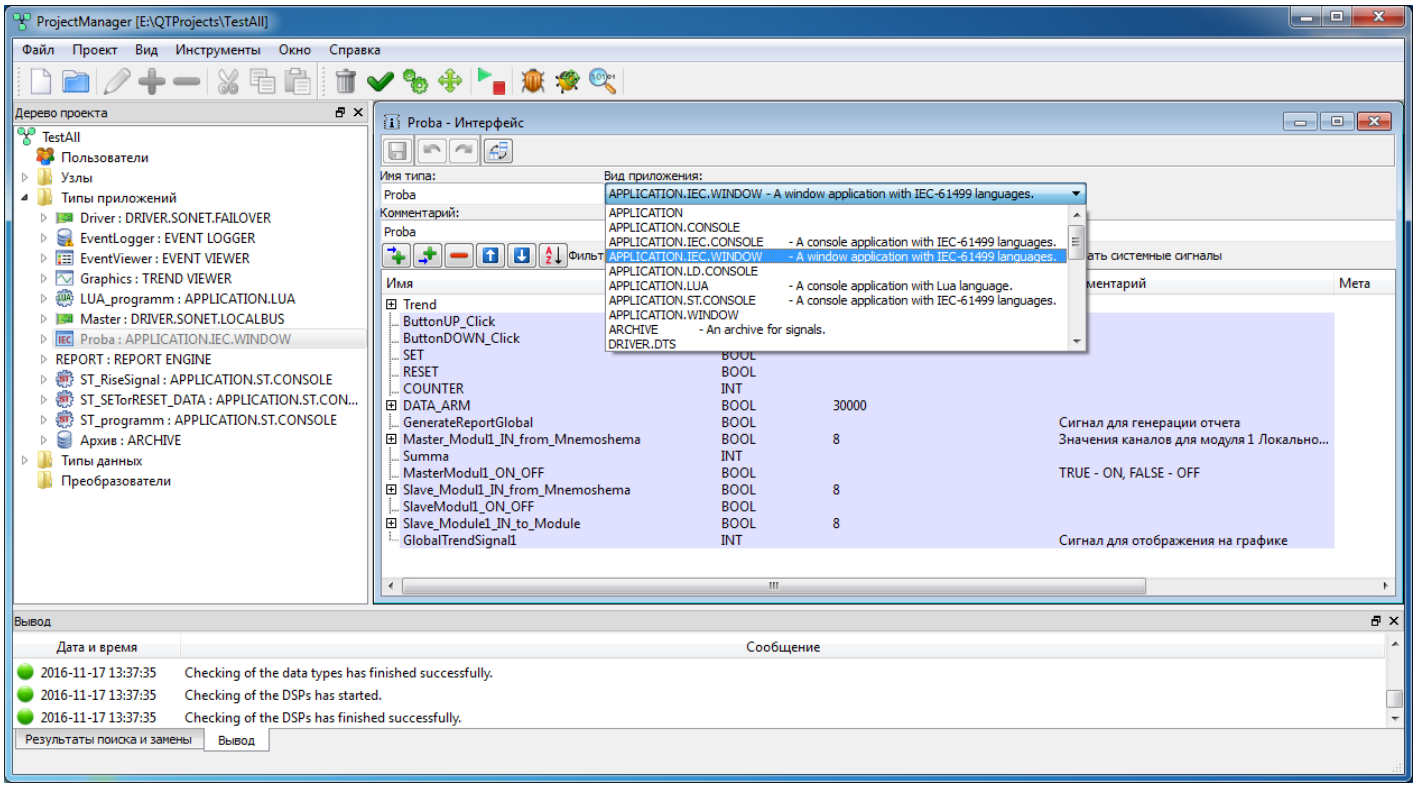


Рисунок 3.118 - Окно интерфейса приложения

По окончании редактирования интерфейса типа приложения необходимо сохранить изменения с помощью кнопки  («Сохранить»), после чего рекомендуется закрыть диалог редактирования крестиком  в верхнем правом углу окна.

3.3.4.2.2. Вызов специализированного редактора приложения APPLICATION.IEC.WINDOW

В дереве проекта тип приложения отображается в виде двух позиций: верхняя позиция – интерфейс приложения, вложенная позиция – специализированный редактор приложения (далее редактор) или его конфигурационный файл. Для IEC-приложений редактором является программа **IECEditor**. Процесс вызова редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

Если окно редактирования интерфейса типа приложения не было закрыто на момент вызова программы **IECEditor**, то **Project Manager** сообщит о блокировке редактирования типа приложения (см. рис. 3.119) и не загрузит программу **IECEditor**.

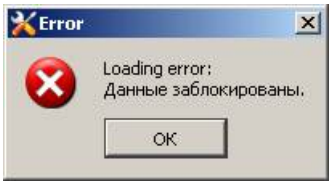


Рисунок 3.119 - Сообщение о блокировке элемента проекта

Для снятия блокировки необходимо закончить работу по редактированию интерфейса типа приложения и закрыть редактор способом, описанным выше. Возможно также обойти блокировку, используя команду Разблокировать элемент из раздела Проект главного меню (см. рис. 3.120). После этого можно повторно запустить **IECEditor** способами, описанными выше.

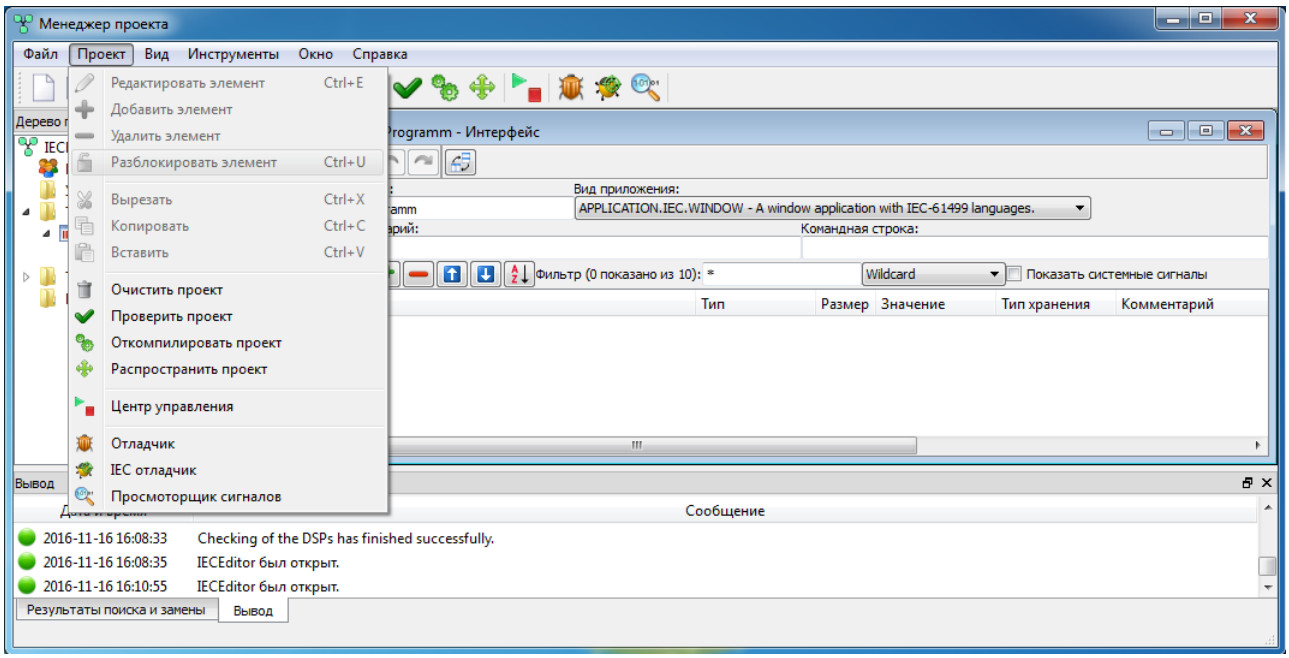


Рисунок 3.120 - Снятие блокировки элемента проекта с помощью команды главного меню

На рис. 3.121 изображено основное окно редактора IEC-приложений **IECEditor**.

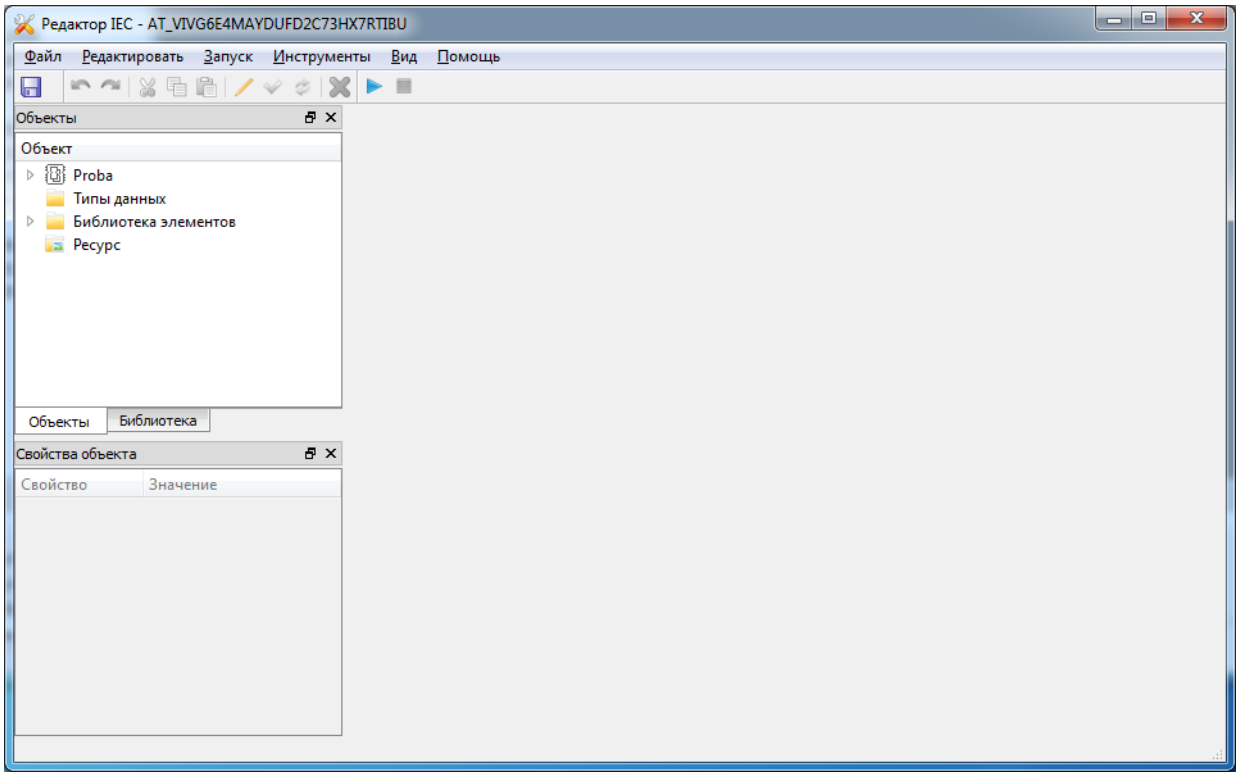


Рисунок 3.121 - Основное окно редактора IEC-приложений IECEditor

3.3.4.2.3. Описание интерфейса и работа со специализированным редактором приложения APPLICATION.IEC.WINDOW

Описание специализированного редактора и принципов работы в нём представлено в раздел 3.4.8.

3.3.5. Программы архивации

3.3.5.1. Описание приложения ARCHIVE

Для хранения различных величин, значений сигналов и других параметров, используемых в проектах автоматизации и управления, используется приложение **ARCHIVE**. Данное приложение сохраняет указанные ему сигналы.

Значения сигналов, сохраняемые приложением **ARCHIVE**, в дальнейшем используются для других приложений, таких как **TrendViewer** (Построение графиков) или **ReportEngine** (Построение отчётов).

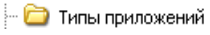
Если в проекте используются несколько приложений **ARCHIVE**, то значения их сигналов постоянно синхронизируются. Приложение, которому необходимы значения сигналов из архива, подключается только к одному из архивов. Если с данным архивом пропадает соединение, то приложение переключается на другой архив.

Для работы с архивированием сигналов нужно помнить несколько понятий:

- **Источник сигнала** - приложение, в котором получают данный сигнал. Для того чтобы указать в приложении, что оно является источником данного сигнала, необходимо в интерфейсе приложения у сигнала в поле **Направление** указать значение **output** (см. описание интерфейса приложения в ProjectManager);

- **Недостоверность (НОПР)** - если у архива пропадает связь с приложением источником, то значение сигнала считается недостоверным и помечается в проекте признаком НОПР. В данном случае на графиках значения будут отображаться другим цветом (это настраиваемый параметр) и данный признак можно получать в алгоритмах расчетов. Сигнал, у которого не указан источник, как описано ранее, не будет помечаться признаком НОПР.

3.3.5.1.1. Создание приложения ARCHIVE

Для создания нового типа приложения необходимо в дереве проекта главного окна программы **ProjectManager**выделить папку 

, а затем нажать кнопку  ("Добавить элемент") на главной панели инструментов (см. рис. 3.122).

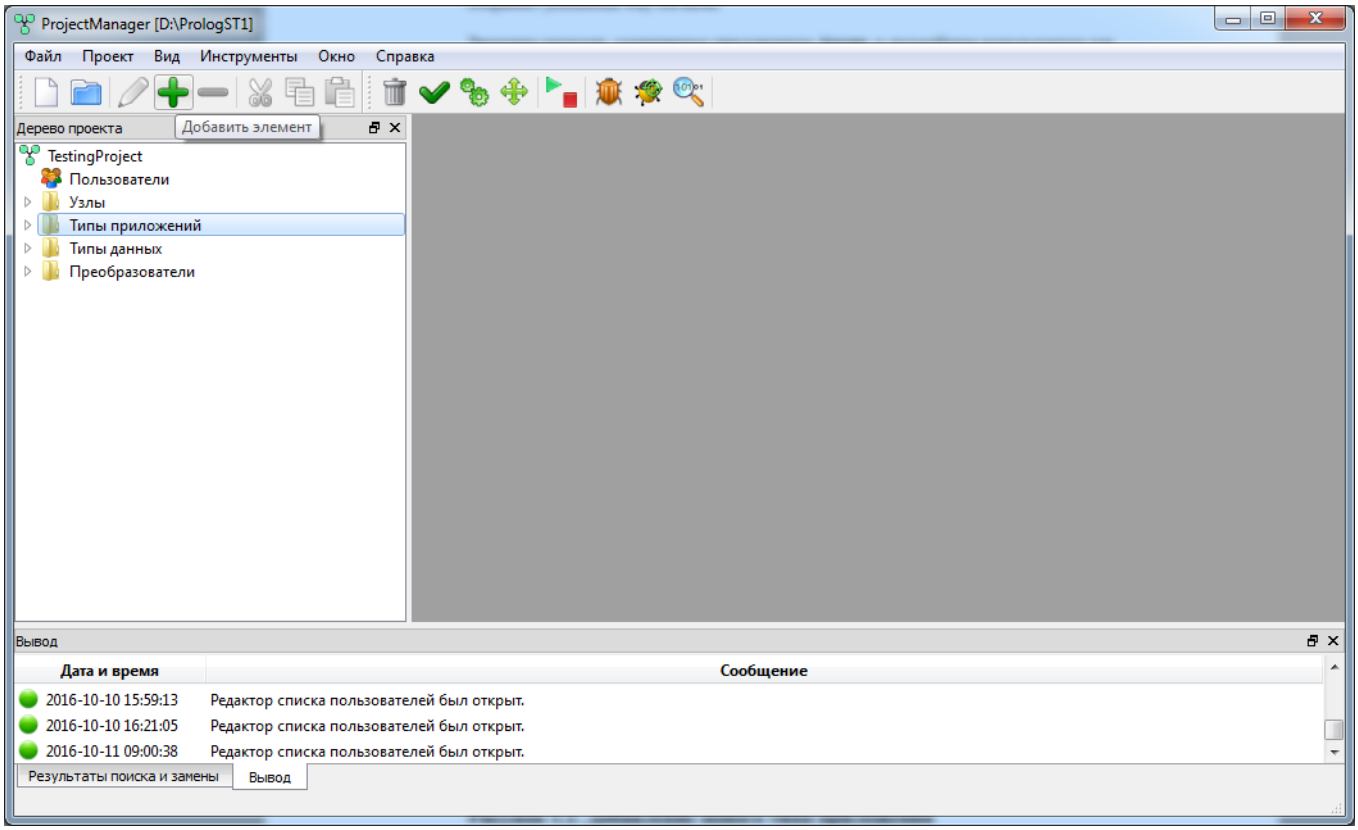


Рисунок 3.122 - Добавление нового типа приложения

Так как новому типу приложения задается вид **APPLICATION**, а имена новым типам приложений присваиваются автоматически по шаблону **AppType_N** (где **N** - уникальное число), то **имя типа** и **вид типа** необходимо изменить. Для этого необходимо отредактировать интерфейс типа приложения. Т. е. нужно в дереве проекта главного окна **ProjectManager** выделить позицию, содержащую **имя типа** и **вид приложения** (в данном случае - строку 

), а затем нажать кнопку  ("Редактировать элемент") на главной панели инструментов (см. рис. 3.123) - откроется диалог редактирования интерфейса типа приложения (см. рис. 3.124).

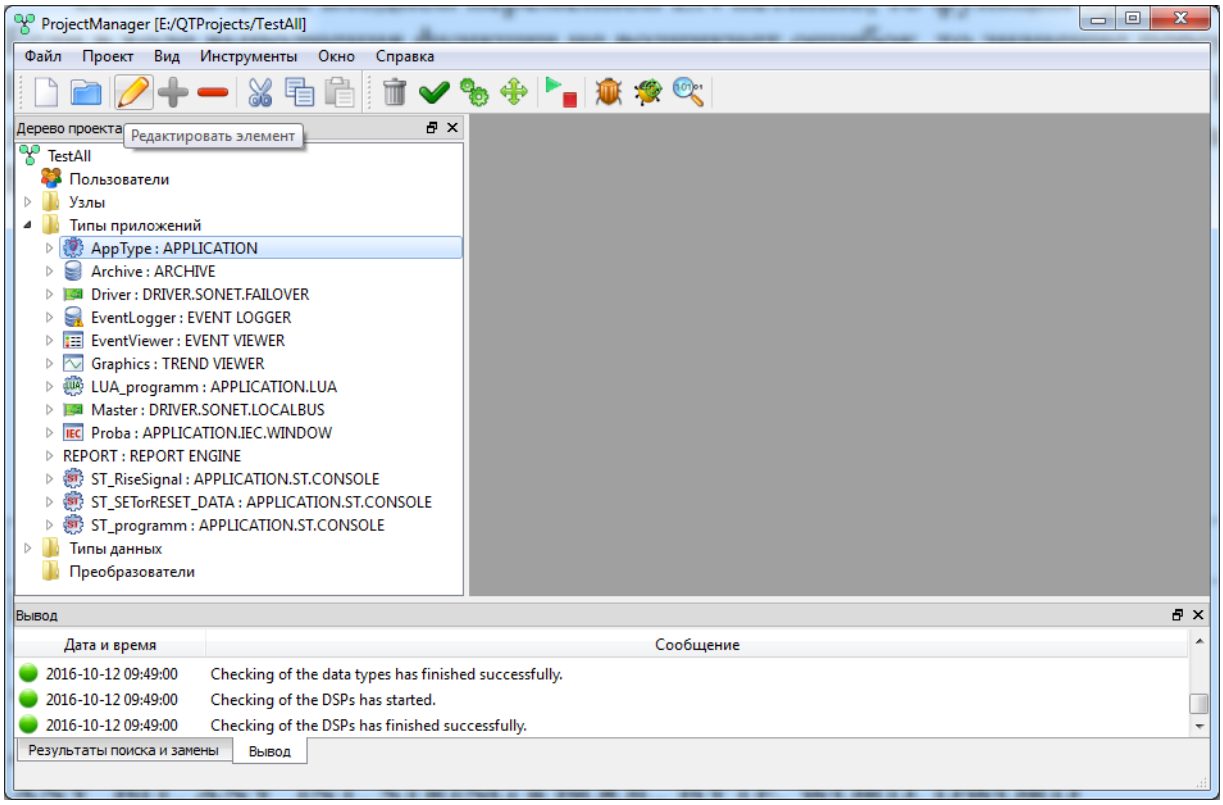


Рисунок 3.123 - Открытие редактора интерфейса типа приложения

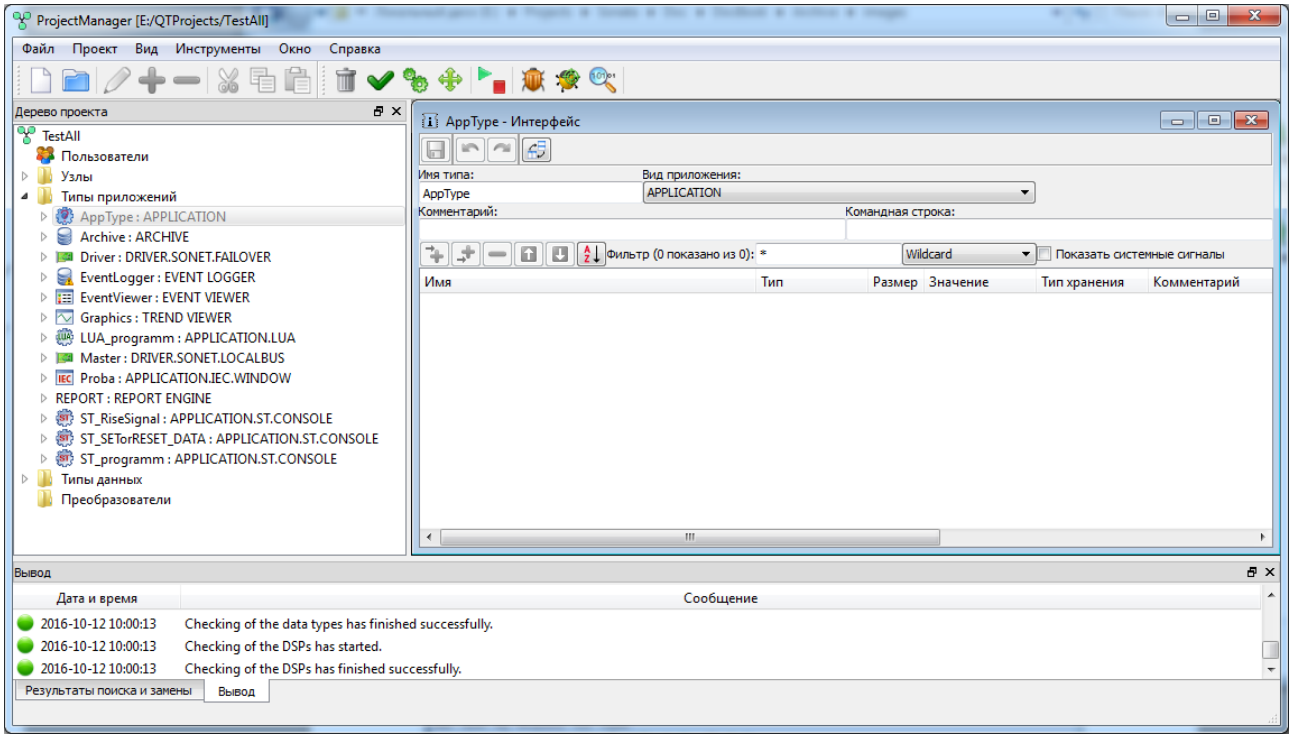


Рисунок 3.124 - Диалог редактирования интерфейса типа приложения

В поле **Имя типа** можно написать удобное для дальнейшего использования имя приложения, в поле **Вид приложения**, необходимо выбрать **ARCHIVE** (см. рис. 3.125).

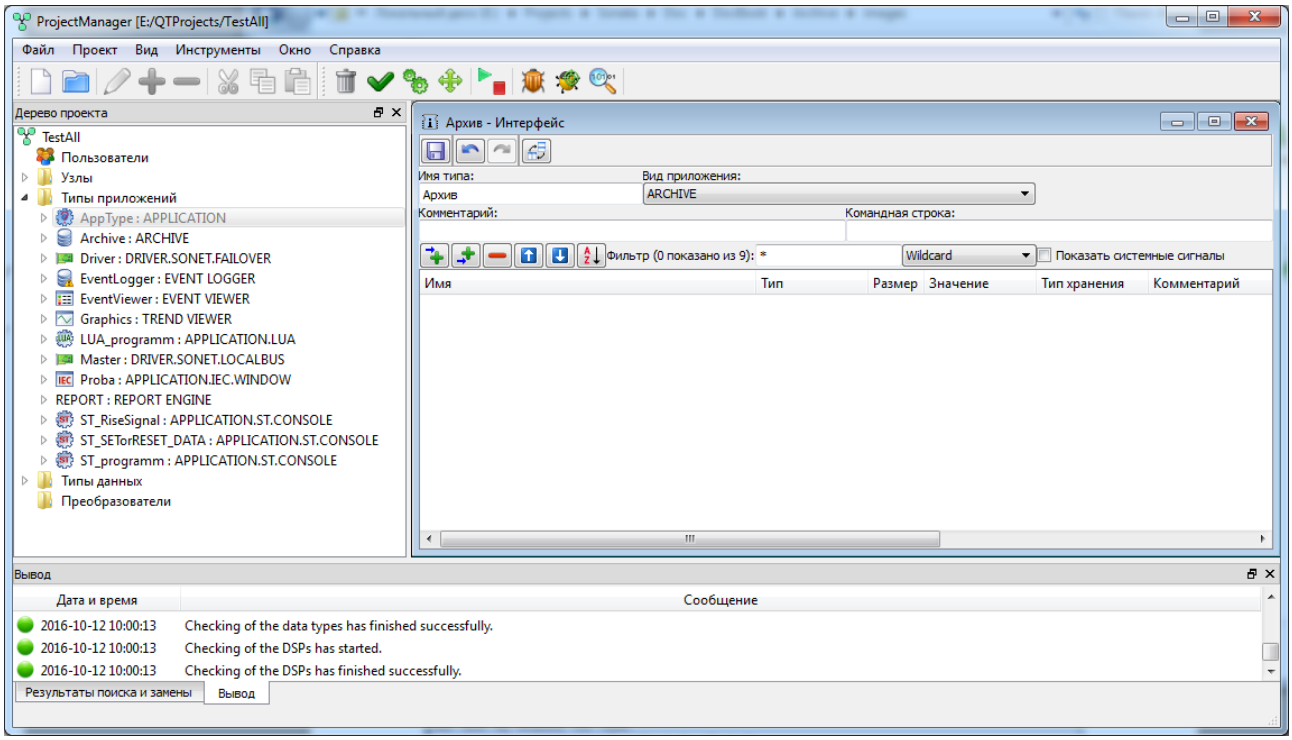


Рисунок 3.125 - Изменение Имя типа и Вида приложения

Изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

3.3.5.1.2. Создание интерфейса приложения ARCHIVE

После выбора вида приложения (в данном случае **ARCHIVE**) необходимо добавить в интерфейс приложения сигналы SCADA-системы "Соната", которые и будут сохраняться приложением **ARCHIVE**. Как создаются сигналы SCADA-системы "Соната", смотрите в раздел 3.2.

Добавление сигналов осуществляется с помощью кнопок редактирования таблицы интерфейса  ("Вставить сверху") и  ("Вставить снизу"). Добавленные сигналы смотрите на рис. 3.126.

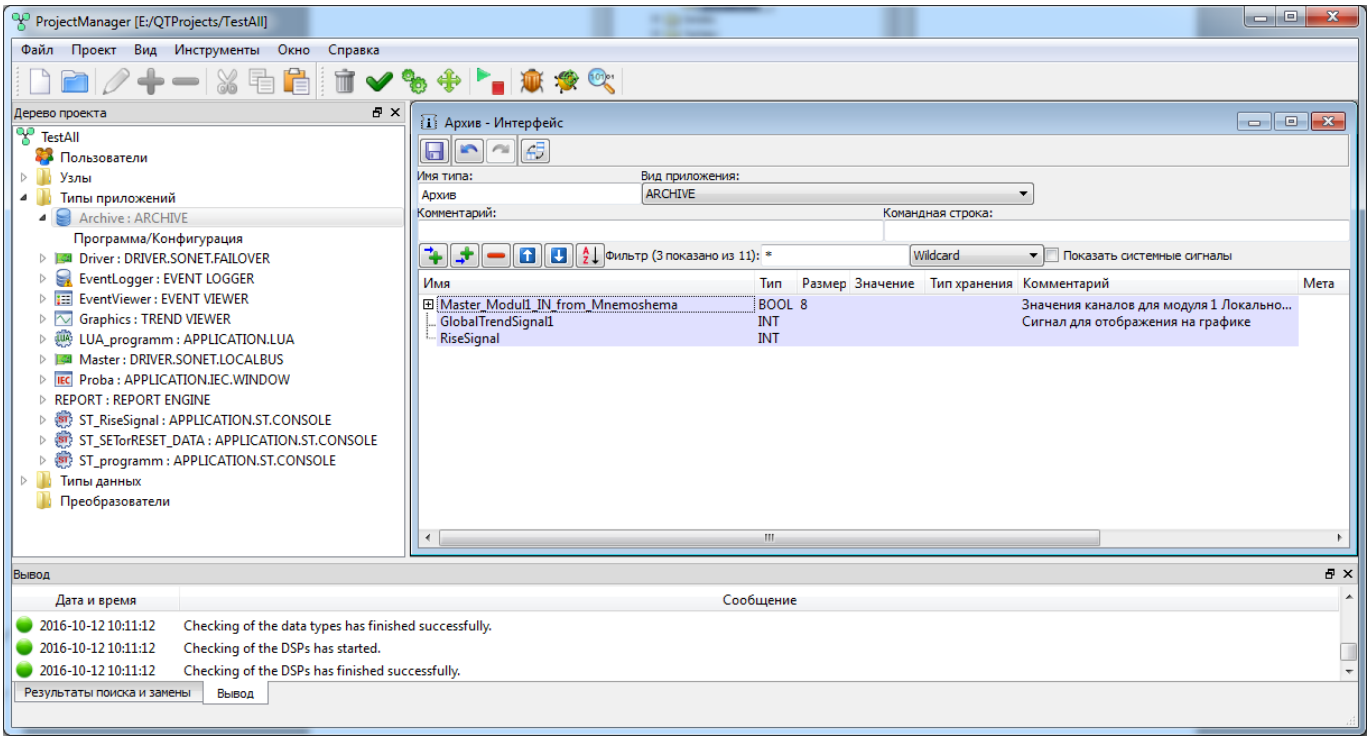


Рисунок 3.126 - Таблица сигналов приложения ARCHIVE

3.3.5.1.3. Внутренние настройки приложения ARCHIVE

Для вызова специализированного редактора приложения **ARCHIVE** необходимо раскрыть папку типа приложения в дереве проекта, а затем установить курсор на строку Программа/Конфигурация , выделить её и нажать кнопку  ("Редактировать элемент") (см. рис. 3.127) - откроется редактор приложения ARCHIVE (см. рис. 3.128).

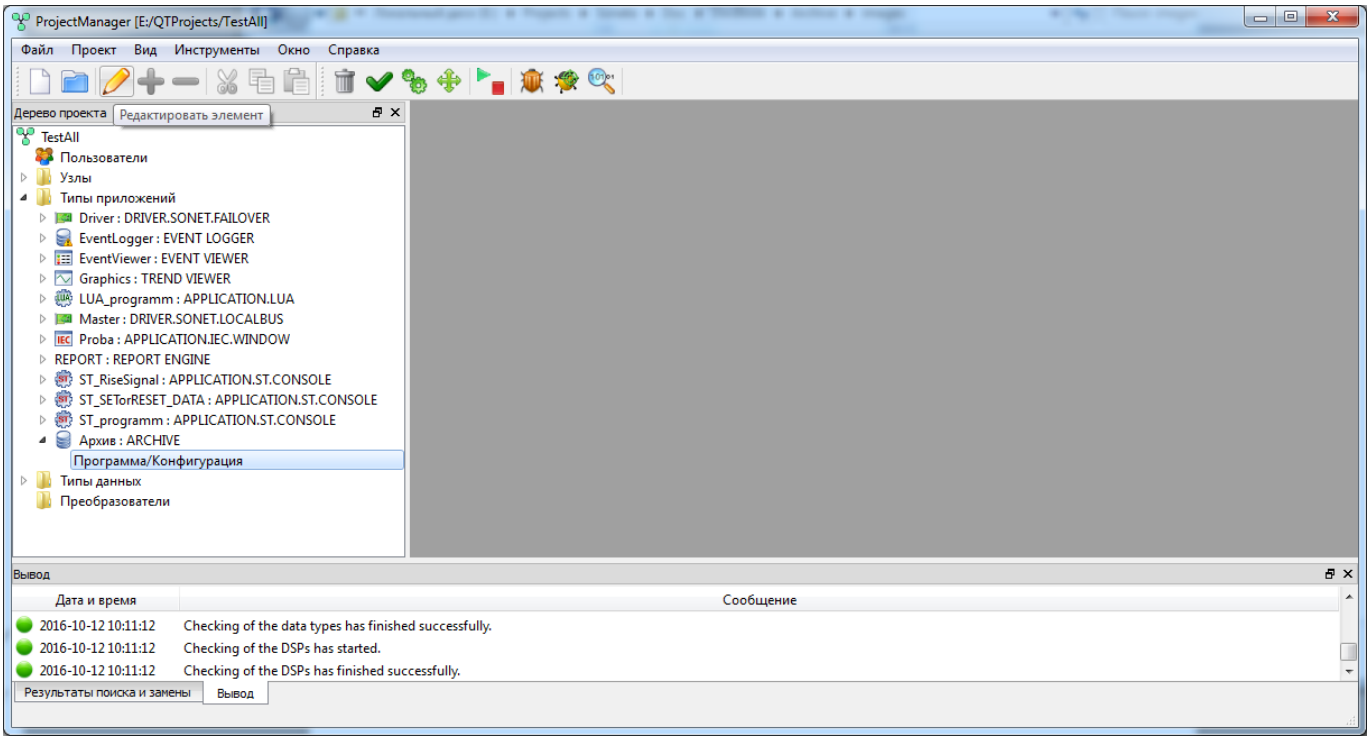


Рисунок 3.127 - Вызов редактора приложения ARCHIVE

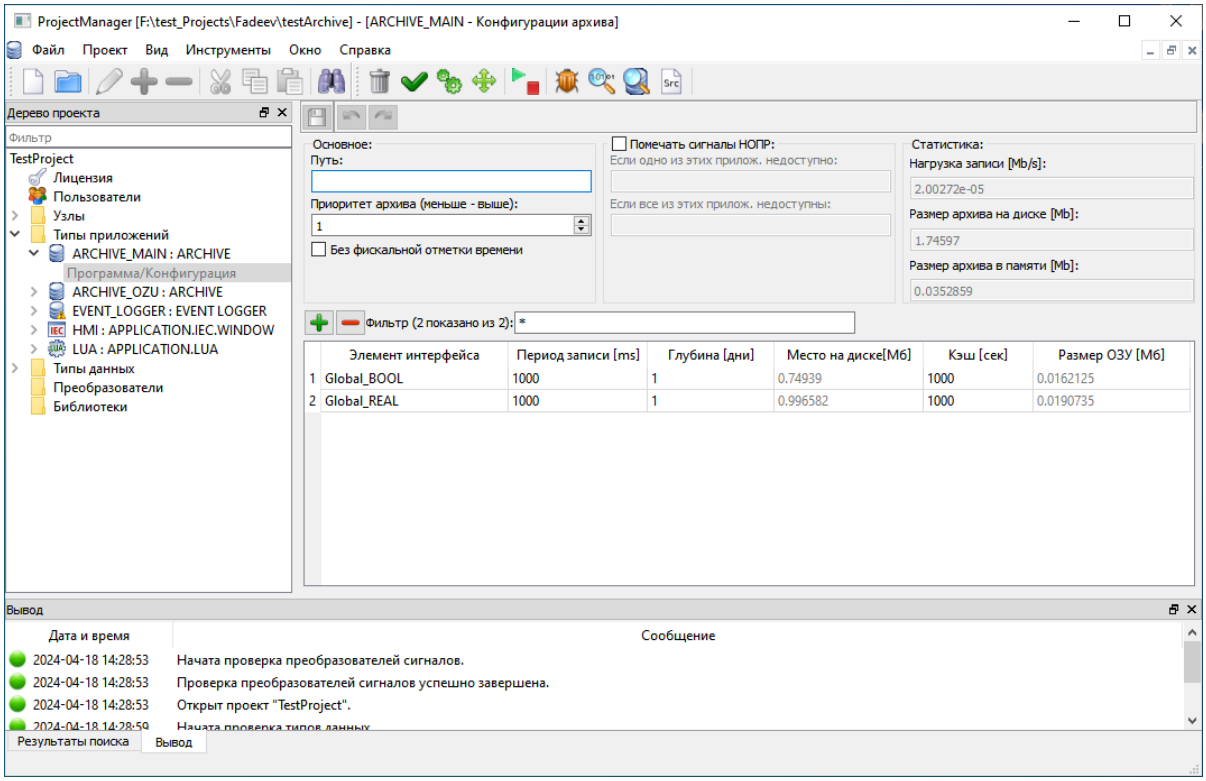


Рисунок 3.128 - Редактор приложения ARCHIVE

Описание элементов специализированного редактора приложения **ARCHIVE**:

- **Путь** - путь, где будут храниться файлы с архивными данными (по умолчанию файлы хранятся на узле в специальной папке в папке среды исполнения);

ВНИМАНИЕ! Если данный архив сигналов будет использоваться в работе приложения Аварийный архив (**ALERT ARCHIVE**), то путь (папка), где будут храниться файлы архива сигналов, не должен быть вне папки среды исполнения. Это связано с тем, что приложение Аварийный архив должно работать с файлами архива сигналов, а из требований информационной безопасности

следует, что приложения среды исполнения не могут работать с файлами вне среды исполнения. Для решения данной проблемы можно использовать символические ссылки на папки. Пример решения данной задачи рассмотрен в разделе раздел 3.3.5.4.

- **Приоритет архива (меньше - выше)** - чем выше это число, тем ниже приоритет у данного архива. Графики в HMI будут сначала обращаться к более приоритетному архиву (меньше число). Данное поле приоритета архивов позволит создать, например, такой проект:
Постановка задачи.
В проекте требуется просмотр оперативных данных технологических параметров с высоким разрешением по времени. Глубина хранения один месяц. Также требуется иметь возможность просмотра годовых данных по технологическим параметрам за год, но с более низким временным разрешением. В этом случае мы в проекте организуем два архива.
Первый архив (с приоритетом 1) сохраняет параметры каждые 100 мс в течение месяца, а второй архив (с приоритетом 2) сохраняет эти же параметры раз в 5 секунд, но уже в течение года. Пока данные есть в первом архиве графики HMI берут их оттуда с высоким разрешением по времени, если же пользователь запросил исторические данные за прошлое время, то в виду их отсутствия в первом архиве, графики запросят их у второго. Переключение между архивами будет выполнятся Сонатой автоматически;

- **Без фискальной отметки времени** - включение этой "галочки" позволяет сократить размер архива в два раза. Данная галочка указывает архиву не сохранять дополнительно для каждой точки каждого сигнала время его реального изменения. Иными словами, архив в Сонате строго периодический и записывает значения сигнала через указанный интервал времени, вне зависимости от того, менялось ли значение или нет. При отключенной данной "галочке" дополнительно сохраняется фискальная отметка времени (время когда сигнал был последний раз изменён), что может быть использовано при разборе неисправностей драйверов или аварийных ситуаций. При обычной эксплуатации системы хранение фискальной отметки времени не требуется;
ВНИМАНИЕ! При выставленной в конфигурации архива галочки "Без фискальной отметки времени" новые архивные файлы не будут читаться старыми компонентами: AlerArchive, Archive, ArchiveViewer из сборок Сонаты меньших r13450, так как изменилась структура файлов.

- **Помечать сигналы НОПР** - данная галочка включает функцию, которая помечает все сигналы архива как недостоверные (НОПР), если нет связи с приложениями, которые указаны в полях: **Если одно из этих прилож. недоступно** или **Если все из этих прилож. недоступны**. Данная галочка позволяет не настраивать приложения источники данных сигналов через интерфейс приложений, а указать приложения источник непосредственно в настройках архива (но нужно настроить для каждого архива в проекте);

- **Если одно из этих прилож. недоступно** - в данном поле через запятую указываются приложения (Имя_узла.Имя_приложения), при отсутствии связи с любым из которых, все сигналы архива помечаются как недостоверные (НОПР);

- **Если все из этих прилож. недоступны** - в данном поле через запятую указываются приложения (Имя_узла.Имя_приложения) и при отсутствии связи со всеми из указанных, все сигналы архива помечаются как недостоверные (НОПР);

- **Нагрузка записи [Mb/s]** - суммарная нагрузка на операции записи на жесткий диск, которую накладывает приложение архив;

- **Размер архива в памяти [Mb]** - суммарный объем памяти, которую занимает приложение архив в ОЗУ;

- **Размер архива на диске [Mb]** - суммарный объем памяти, которую занимает приложение архив на жестком диске;

- **Элемент интерфейса** - сигналы SCADA-системы "Соната", которые будут храниться в архиве;

- **Период записи [ms]** - периодичность записи сигналов SCADA-системы "Соната";

- **Глубина [дни]** - количество дней, в течении которого храняться значения сигналов;

- **Место на диске [Мб]** - объем памяти, который занимают данные о сигнале на жестком диске;

- **Кэш [сек]** - время в секундах, в течении которого архив хранит значения сигнала в кэше или ОЗУ узла;

- **Размер ОЗУ [Мб]** - объем памяти ОЗУ, который занимают значения сигнала, хранящиеся в кэше.

На рис. 3.129 приведен пример настроенного приложения **ARCHIVE**. Настройки архива сигналов задаются для каждого сигнала индивидуально.

К примеру, сигнал SCADA-системы "Соната" **GlobalTrendSignal** будет сохраняться каждые 1000 мс или каждую секунду в течении 10 дней. Файлы с данными, хранящими значения сигнала, будут занимать на жестком диске 8,25 Мб. Данный сигнал так же хранится в кэше в течении 1000 секунд и занимает 0,017 Мб ОЗУ.

Сигнал SCADA-системы "Соната" **Master_Modul_IN_from_Mnemosheme[0]** будет сохраняться каждые 0,5 секунды в течении 1 дня и его файлы данных будут занимать на жестком диске 1,49 Мб, а в кэше он храниться не будет.

Сигнал SCADA-системы "Соната" **RiseSignal** будет сохраняться каждую секунду, но только в Кэш в течении 1000 секунд. В результате показано, что данный сигнал будет занимать 0,017 Mb в ОЗУ.

Нагрузка чтения/записи на жесткий диск при этом 0,00011 Мб/с.

Суммарный размер архива в кэше 0,034 Мб.

Суммарный размер архива на жестком диске 9,74 Мб.

Для хранения данных следует учесть существующие ограничения по размеру файла, хранящего архивные данные:

- в ОС QNX файловая система не поддерживает файлы объёмом более 2 ГБ;

- в ОС Windows файловая система FAT32 не поддерживает файлы объёмом более 2 ГБ, для файловой системы NTFS такого ограничения нет.

Все сигналы в архиве собираются в группы, у которых одинаковые значения периода записи и глубины хранения. Данные сигналы помещаются в один файл из расчета чтобы объем информации за 1 час не превысил 2 ГБ. Каждый час создается новый файл. По истечении значения глубины хранения старые файлы будут удаляться.

В настройках в поле с названием **Если один из них недоступен** указаны имена приложений **ARM.Driver1** и **ARM.Algoritm**. Если архив потеряет связь с одним из данных приложений, то все сигналы архива будут помечены признаком недостоверности (НОПР).

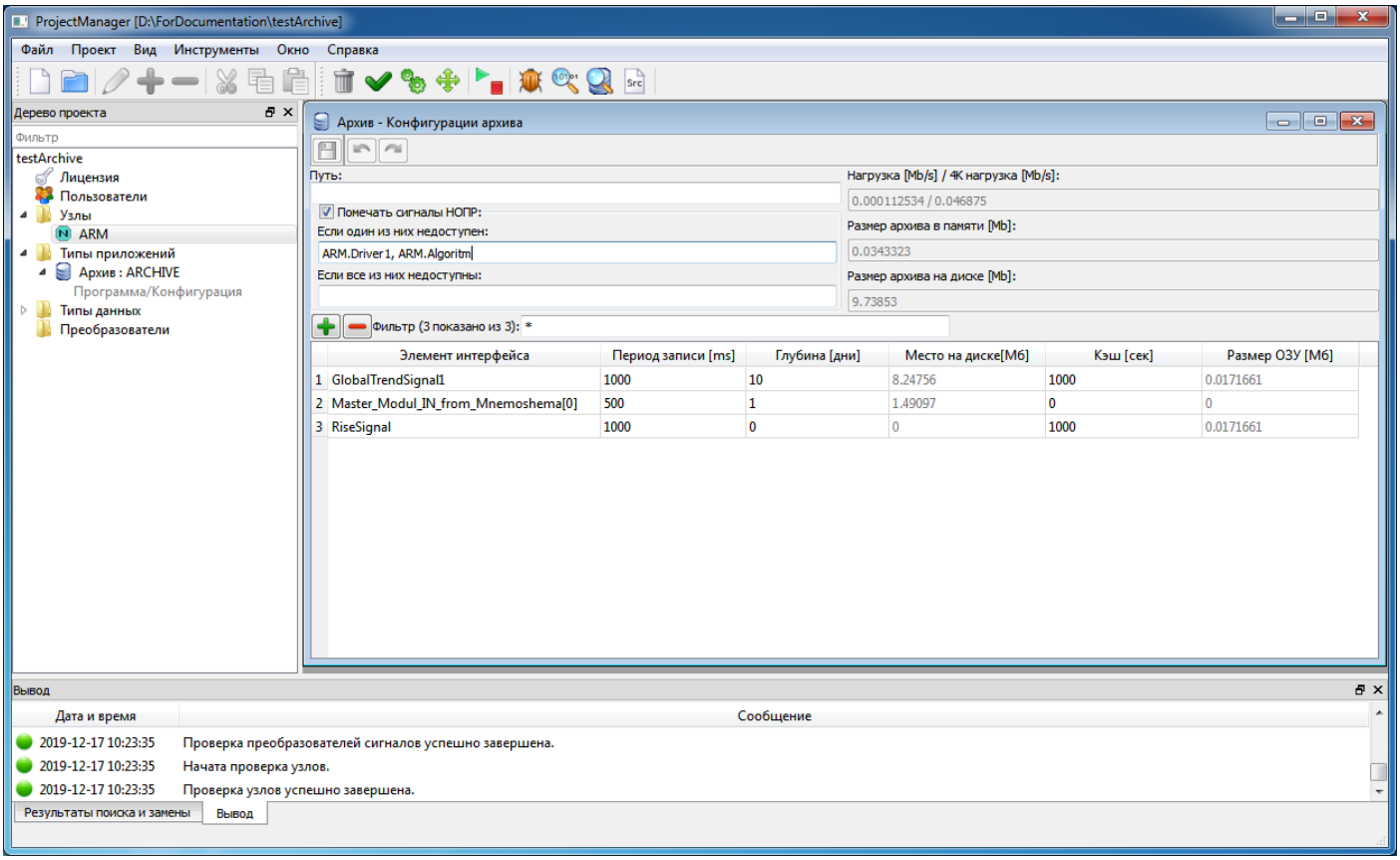


Рисунок 3.129 - Настроенный редактор приложения ARCHIVE

Приложение **ARCHIVE** настроено и готово к работе. Для использования необходимо добавить приложение **ARCHIVE** в любой из **Узлов** проекта (см. рис. 3.130).

При добавлении приложения **ARCHIVE** в узел проекта, рекомендуется настроить поле **Таймаут останова** равным 10000, что соответствует 10 секундам.

Для просмотра и работы с архивными значениями вне проекта используется приложение **ArchiveViewer**.

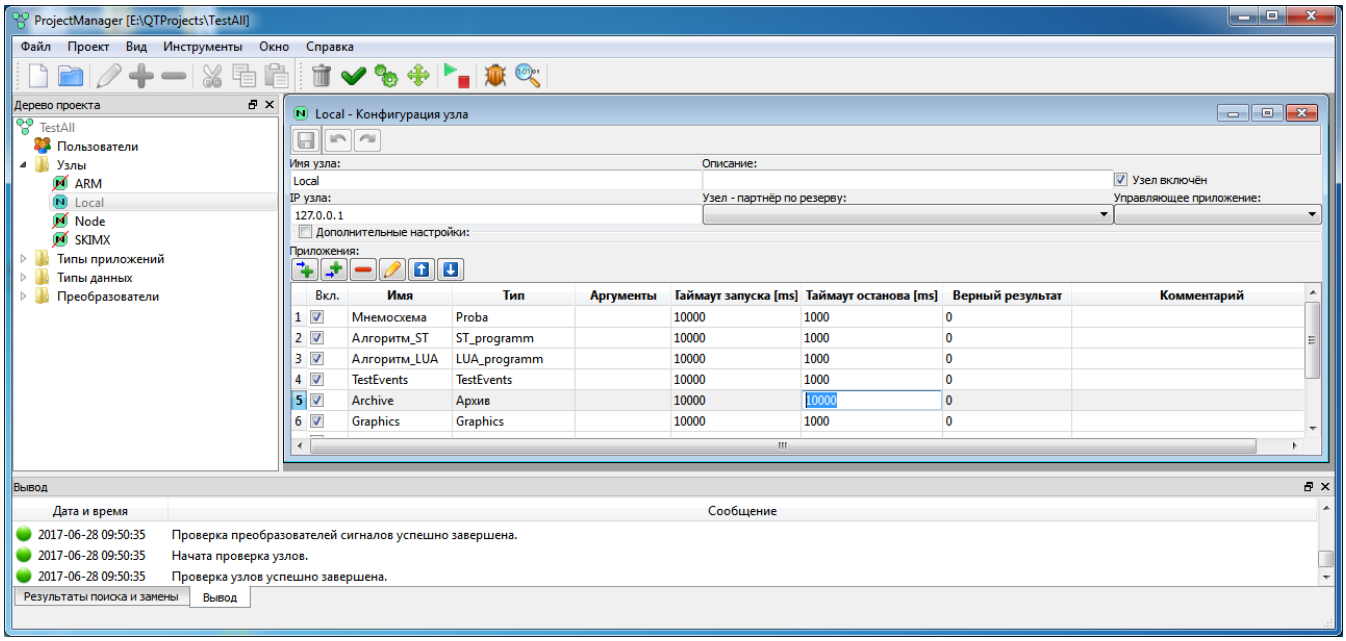


Рисунок 3.130 - Конфигурация узла проекта

3.3.5.2. Описание приложения EVENT LOGGER (Архив событий)

Приложение **EVENT LOGGER** (далее Архив событий) является приложением SCADA-системы "Соната", которое автоматически регистрирует события и архивирует их.

3.3.5.2.1. Создание приложения EVENT LOGGER

Для создания приложения **Архив событий** сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.131 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** (необходимо выбрать **EVENT LOGGER**).

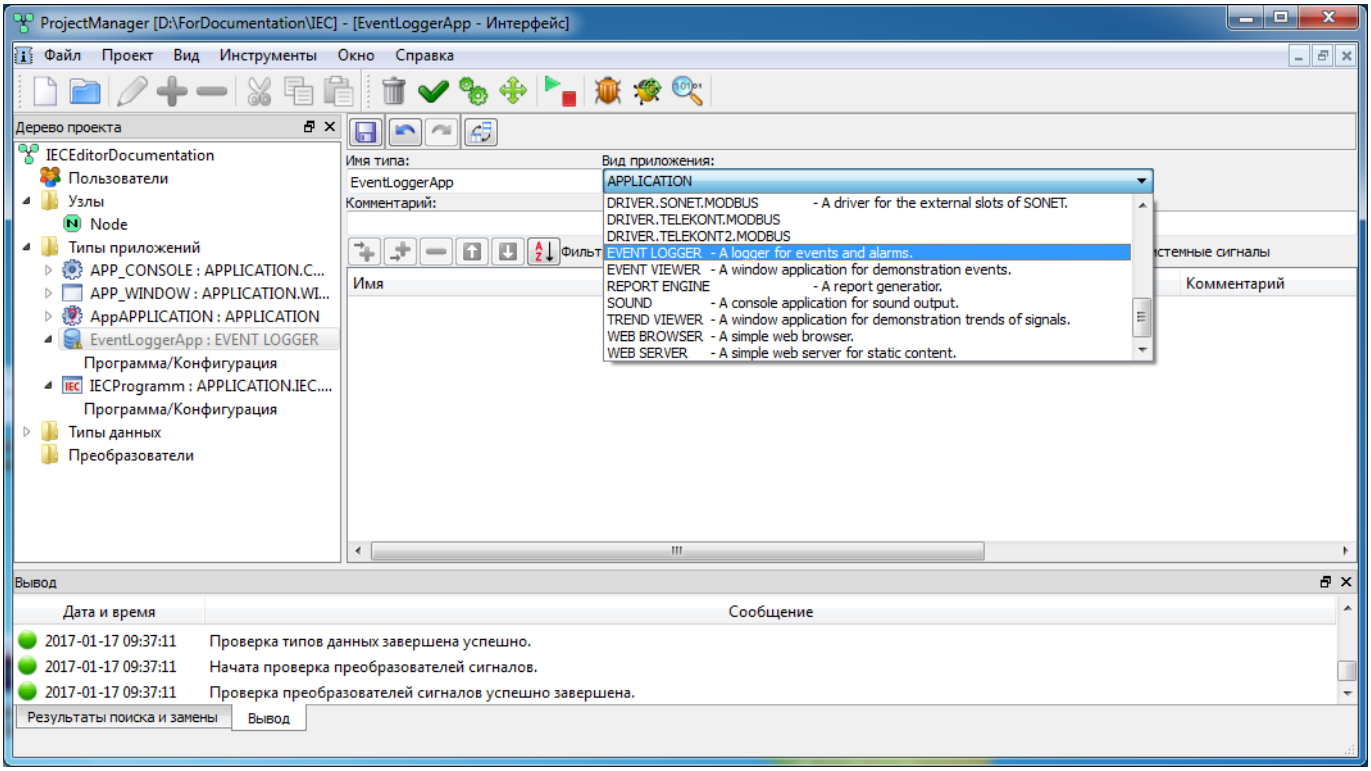


Рисунок 3.131 - Изменение Имени типа и Вида приложения

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.132) – для подтверждения изменения необходимо нажать кнопку Да.

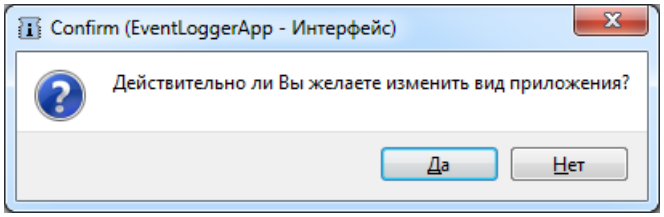
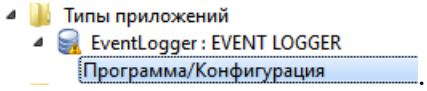


Рисунок 3.132 - Окно подтверждения для изменения вида типа приложения

Других настроек для приложения **EVENT LOGGER** не требуется. Изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

3.3.5.2.2. Применение приложения EVENT LOGGER (Архив событий)

Для использования данного приложения в проекте его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1). Данное приложение работает автономно и занимается регистрацией и архивированием всех событий в проекте.

Приложение **Архив событий** имеет несколько конфигурационных параметров. Данные параметры находятся в окне Конфигурация регистратора событий (см. рис. 3.133), которое вызывается по нажатию кнопки  (Редактировать) на меню 

.

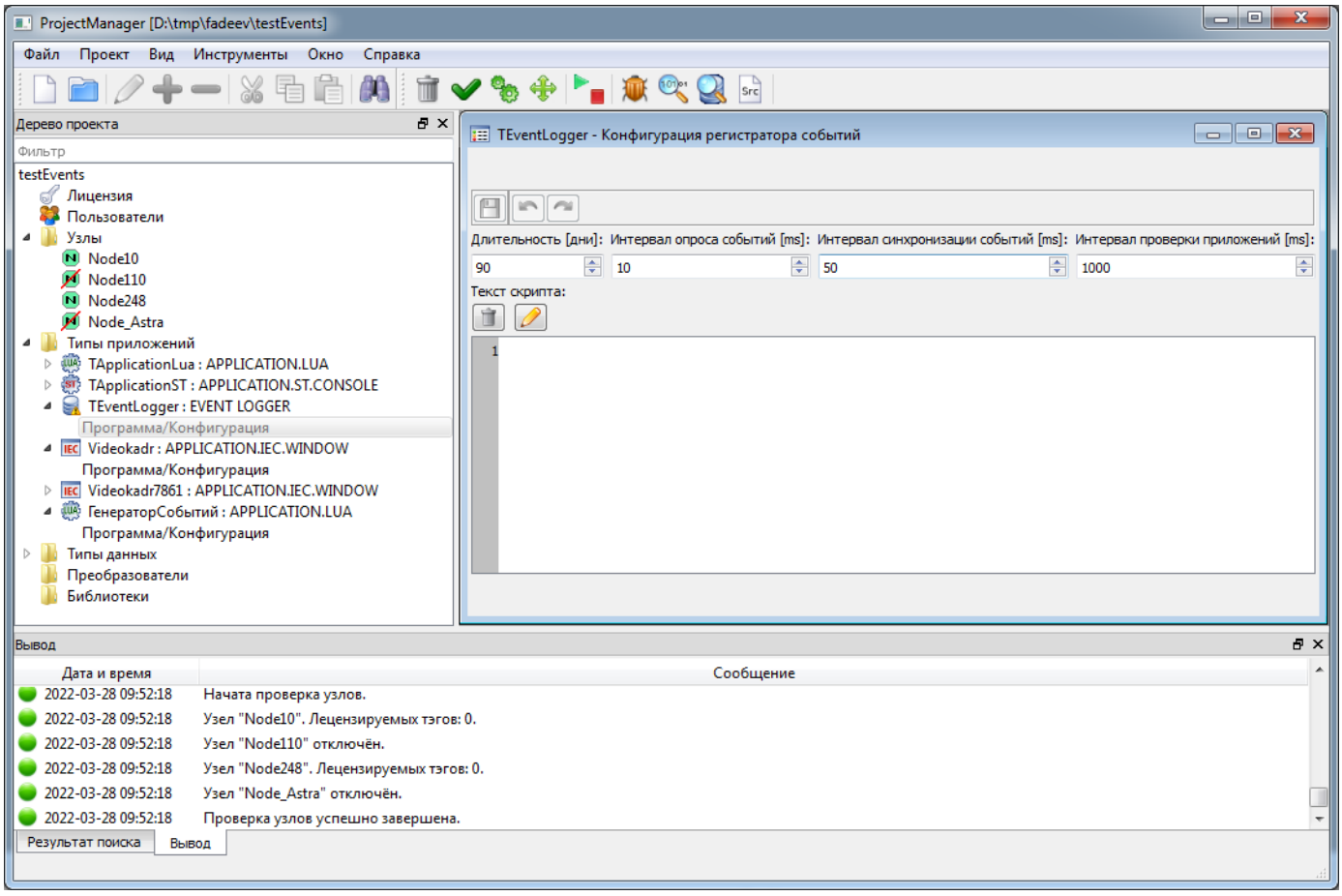


Рисунок 3.133 - Окно Конфигурация регистратора событий

В окне Конфигурация регистратора событий располагаются следующие параметры:

- **Длительность [дни]** - количество дней, в течении которого хранятся события (события, которые устарели, будут удалены);

- **Интервал опроса событий [ms]** - время между опросом событий у всех приложений. Если события у приложений возникают чаще данного интервала, то данный интервал нужно уменьшить;

- **Интервал синхронизации событий [ms]** - интервал синхронизации соседних **EVENT LOGGER**;

- **Интервал проверки приложений [ms]** - данный интервал указывает как часто **EVENT LOGGER** опрашивает все приложения на доступность, т.е. отвечает оно или нет.

Ниже располагается область ввода под названием **Текст скрипта**. В данной области можно написать скрипт на языке ЛУА. Данный скрипт даёт возможность выполнять различные действия с полученными событиями. К примеру, сохранить событие в другую базу данных или отправить по ftp и т.д. Более подробно будет описано ниже.

При работе приложения **Архив событий** создаются файлы ГГГГММДД_ЧЧЧЧ.evl, где ГГГГ - год, ММ - месяц, ДД - число, ЧЧЧЧ - время (пример 20220328_1900.evl). Один такой файл хранит события за один час работы системы.

Для просмотра событий необходимо воспользоваться функциональными блоками **TEventViewer** и **TAlarmViewer** (см. документ SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33), которые входят в библиотеку функциональных блоков приложения **APPLICATION.IEC.WINDOW** (см. раздел 3.3.4.2).

Есть возможность преобразовать файл *.evl в формат txt. Для этого необходимо скопировать файл EventLogger.exe в папку с файлом *.evl и выполнить команду:

EventLogger.exe -d имя_файла.evl

В результате появится дополнительный файл имя_файла.txt, который можно открыть текстовым редактором.

3.3.5.2.3. Описание поля Текст скрипта

В области **Текст скрипта** можно писать скрипты на языке ЛУА (см. рис. 3.134). Данные скрипты позволяют выполнять различные действия с полученными событиями.

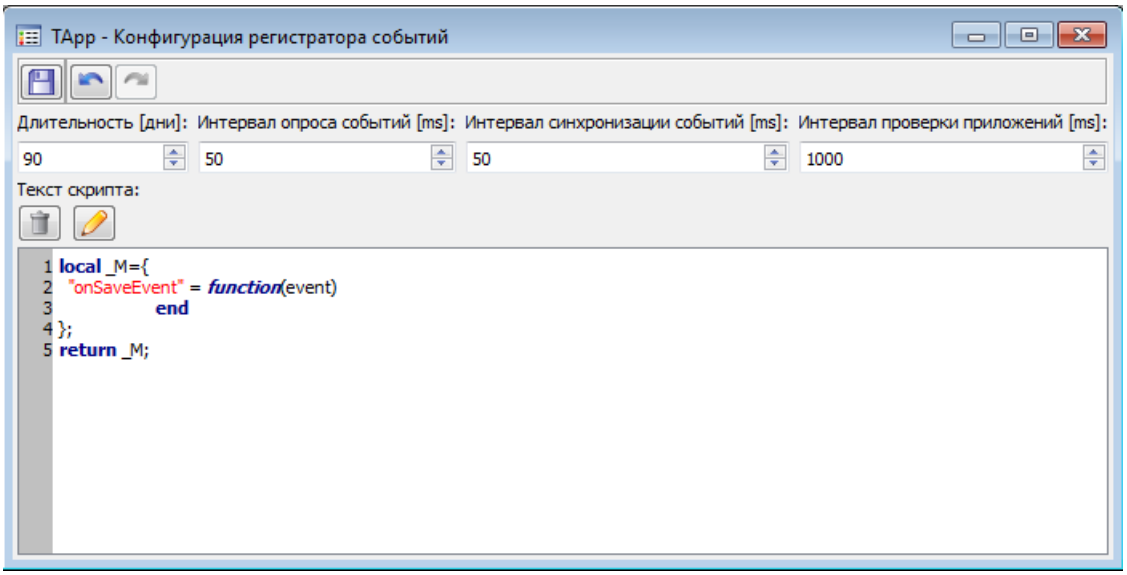


Рисунок 3.134 - Область Текст скрипта

По нажатию на кнопку  ("Редактировать скрипт") откроется окно для редактирования скрипта и появится основная рабочая функция скрипта, на базе которой пишется весь скрипт (см. рис. 3.135). Данная функция позволяет выполнять дополнительные действия с событиями, которые в данный момент поступили на сохранение в архив.

По нажатию на кнопку  ("Очистить скрипт") весь текст скрипта удаляется.

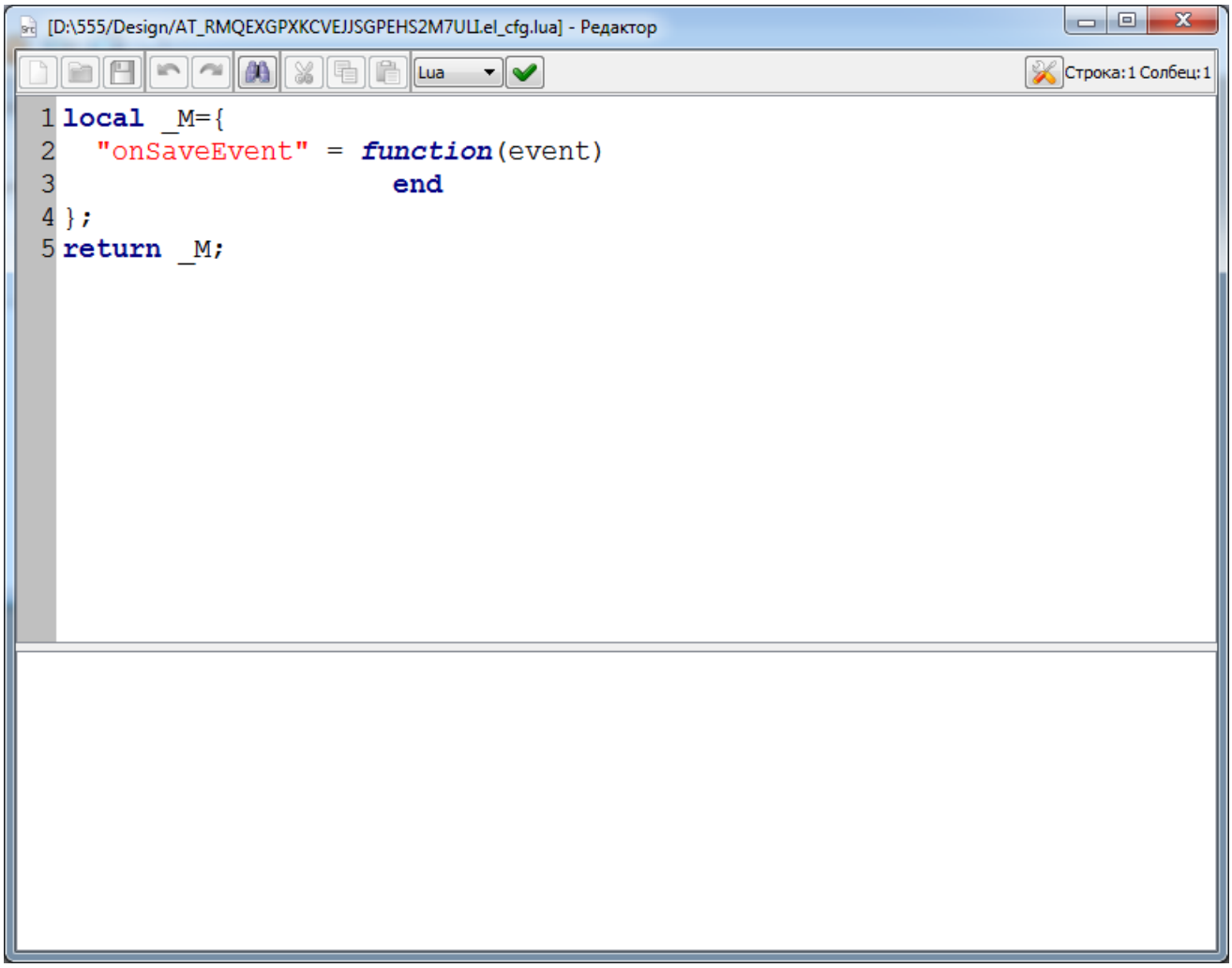


Рисунок 3.135 - Редактор скрипта

3.3.5.2.4. Пример скрипта для Архива событий

Для написания скриптов применяется язык программирования Lua.

Данный скрипт для всех сохраняемых событий выполняет две функции:

- Формирует данные в JSON формате (num записей, одно событие - одна запись со всеми данными о событии) и отправляет их в виде файла по FTP;

- Подключается к базе postgreSQL и записывает события с категорией 3 в таблицу SecurityTable, а события с категорией 5 в таблицу AdministratorTable.

```
local pgsql = require ("pgsql") ----- Библиотека работы с PostgreSQl
local socket = require('socket') ---- Библиотека обработки сокета Lua
local ftp = require('socket.ftp')
local host = '192.168.10.201'
local port = '5432'
local db = 'testDB'
local user = 'user'
local password = 'userPassword'

---- Функция для подключения к postgreSQL базе данных
```

```
local function getConnect(h,p,dbname,user,password)
    return pgsql.connectdb(string.format('host=%s port=%s dbname=%s
                                          user=%s password=%s',h,p,
                                          dbname,user,password))
end

local conn
local tExtSecurity = false
local tExtAdministrator = false

---- Функция, которая создает таблицу SecurityTable в базе
local function getTableSecurity()
    return conn:exec('CREATE TABLE IF NOT EXISTS SecurityTable (
                      pos INTEGER,dt NUMERIC,userDt NUMERIC,
                      uuid VARCHAR(64),groupUuid VARCHAR(64),
                      category INTEGER,state INTEGER,
                      application VARCHAR(64),source VARCHAR(64),
                      us VARCHAR(64),msg VARCHAR(254),
                      meta VARCHAR(254));')
end

---- Функция, которая создает таблицу AdministratorTable в базе
local function getTableAdministrator()
    return conn:exec('CREATE TABLE IF NOT EXISTS AdministratorTable (
                      pos INTEGER,dt NUMERIC,
                      userDt NUMERIC,uuid VARCHAR(64),
                      groupUuid VARCHAR(64),
                      category INTEGER,state INTEGER,
                      application VARCHAR(64),
                      source VARCHAR(64),us VARCHAR(64),
                      msg VARCHAR(254),
                      meta VARCHAR(254));')
end

---- Функция, которая добавляет строку с данными события в таблицу
---- SecurityTable
local function setValueSecurity(event)
    return conn:execParams('INSERT INTO SecurityTable (pos,dt,userDt,
                            uuid,groupUuid,category,state,application,
                            source,us,msg,meta) values ($1,$2,$3,$4,
                            $5,$6,$7,$8,$9,$10,$11,$12);',
                            event["pos"],event["dt"],event["userDt"],
                            event["uuid"],event["groupUuid"],
                            event["category"],event["state"],
                            event["application"],event["source"],
                            event["user"],event["msg"],
                            event["meta"])
end

---- Функция, которая добавляет строку с данными события в таблицу
```

```
---- AdministratorTable
local function setValueSAdministrator(event)
    return conn:execParams('INSERT INTO AdministratorTable (pos,dt,
                           userDt,uuid,groupUuid,category,state,
                           application,source,us,msg,meta) values
                           ($1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12);'
                           ,event["pos"],event["dt"],event["userDt"],
                           event["uuid"],event["groupUuid"],
                           event["category"],event["state"],
                           event["application"],event["source"],
                           event["user"],event["msg"],event["meta"])
end

local iterator = 0
local dt = 0 -- дата и время начала формирования файла
local num = 100 -- Количество записей в сформированном JSON файле
local addr = "ftp://10.10.12.250/" -- Адрес FTP сервера
local dump = "" -- строка с данными

---- Данная функция формирует JSON файл из num записей с информацией о
---- событиях и отправляет данный файл по FTP
local function putJSONtoFTP(event,prefix,dt,iterator,quantity,address)
    -- Получаем текущее время.
    if iterator == 0 then
        dt = os.time();
    end
    -- записываем в строку объект JSON
    dump = dump..string.format("{%q:%s,\n%q:%s,\n%q:%q,\n%q:%q,
           \n%q:%s,\n%q:%s,\n%q:%q,\n%q:%q,\n%q:%q,\n%q:%q,
           \n%q:%q},\n","dt",event["dt"],"userDt",event["userDt"],
           "uuid",event["uuid"],"groupUuid",event["groupUuid"],
           "category",event["category"],"state",event["state"],
           "application",event["application"],"source",
           event["source"],"user",event["user"],"msg",event["msg"],
           "meta",event["meta"])
    iterator = iterator + 1
    -- если накоплено необходимое количество записей в строку, то сохраняем
    -- его на ftp-сервере
    if iterator >= quantity then
        -- Формируем имя файла.
        local fileName = prefix..os.date("_%Y%m%d_%H%M%S.json", dt);
        local f, e = ftp.put(address..fileName, dump)
        if f == nil then
            Core.addLogMsg(e) -- печать ошибки загрузки файла
        else
            iterator = 0
        end
    end
    return iterator,dt
end
```

```
----- Основная функция
local _M={
    ["onSaveEvent"] = function(event)
          -- Собираем данные в JSON файл и отправляем по FTP
          iterator,dt = putJSONtoFTP(event,'IB',dt,iterator,num,addr)
          Core.addLogMsg(tostring(iterator)..' '..tostring(dt))
          -- Работа с базой данных postgreSQL
          if conn == nil then
              conn=getConnect(host,port,db,user,password)
          end
          if conn:status() ~= pgsql.CONNECTION_OK then
              Core.addLogMsg('Cant connect to db '
                             ..conn:errorMessage())
              getConnect(host,port,db,user,password)
          end
          if conn:status() == pgsql.CONNECTION_OK then
              Core.addLogMsg('Connect to db '..conn:errorMessage())
              local res
              if tExtSecurity == false then
                  res = getTableSecurity()
                  if res:status() == pgsql.PGRES_COMMAND_OK then
                      tExtSecurity = true
                      Core.addLogMsg('Create table Security')
                  else
                      Core.addLogMsg('Failed to create table Security'
                                     ..res:errorMessage())
                  end
              end
              if tExtAdministrator == false then
                  res = getTableAdministrator()
                  if res:status() == pgsql.PGRES_COMMAND_OK then
                      tExtAdministrator = true
                      Core.addLogMsg('Create table administrator')
                  else
                      Core.addLogMsg(
                      'Failed to create table administrator'
                      ..res:errorMessage())
                  end
              end
              if tExtSecurity == true and event["category"] == 3 then
                  res = setValueSecurity(event)
                  if res:status() ~= pgsql.PGRES_COMMAND_OK then
                      Core.addLogMsg('Failed to insert value'
                                     ..res:errorMessage())
                  end
              end
              if tExtAdministrator == true
                 and event["category"] == 5 then
                  res = setValueSAdministrator(event)
```

```
                if res:status() ~= pgsql.PGRES_COMMAND_OK then
                    Core.addLogMsg('Failed to insert value'
                                   ..res:errorMessage())
                end
            end
        end
};
return _M;
```

3.3.5.3. Описание работы с приложением ALERT ARCHIVE (Аварийный архив сигналов и событий)

Приложение **ALERT ARCHIVE** (далее Аварийный архив) предназначено для получения значений сигналов и событий за указанный промежуток времени до и после аварии.

Для реализации данной возможности в проекте необходимо настроить:

- приложение для архивирования требуемых сигналов (ARCHIVE);

- приложение для архивирования событий и тревог (EVENT LOGGER);

- приложение Аварийный архив (ALERT ARCHIVE), которое использует информацию из двух ранее указанных архивов.

3.3.5.3.1. Создание приложения Аварийный архив

Приложение Аварийный архив в СКАДА системе "Соната" создается стандартным способом (см. раздел 3.2.2.4.1).

Далее в интерфейсе приложения в поле **Вид приложения** нужно выбрать **ALERT ARCHIVE**, в поле **Имя типа приложения** ввести нужное имя (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем имя типа не может начинаться с цифры), добавить глобальные или локальные сигналы с помощью кнопок  (см. рис. 3.136).

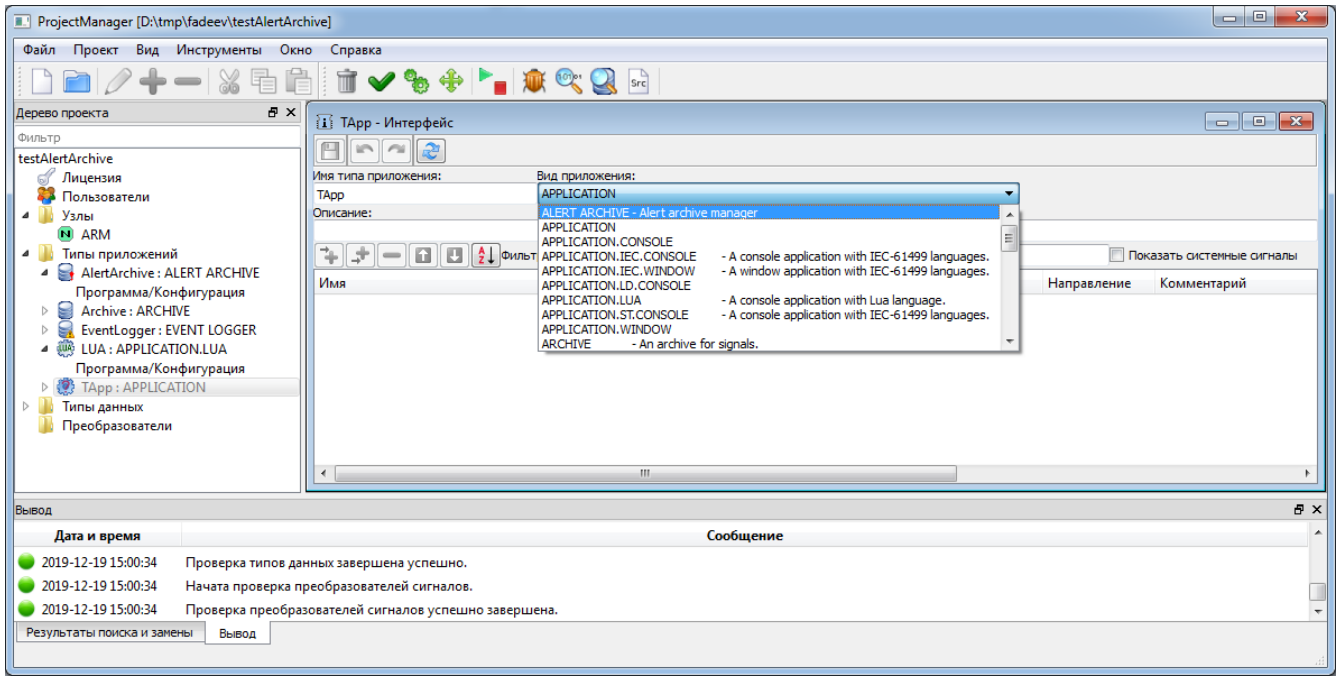


Рисунок 3.136 - Настройка интерфейса приложения Аварийный архив

В результате будет получен настроенный интерфейс приложения Аварийный архив. В нашем случае мы добавили следующие глобальные сигналы: Signal_WriteAlertArchive:BOOL, Temperature:REAL, G1_INT:INT и G2_BOOL:BOOL (см. рис. 3.137). В дальнейшем будет описано применение данных сигналов.

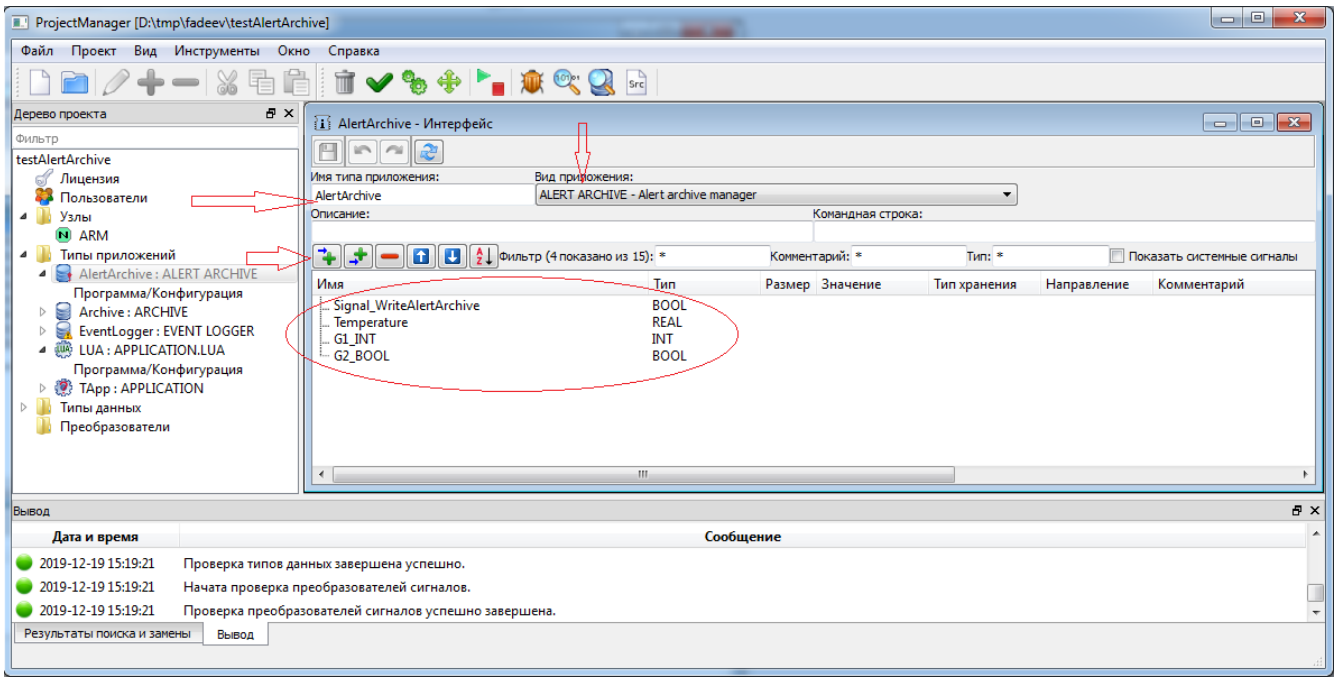


Рисунок 3.137 - Настроенный интерфейс приложения Аварийный архив

3.3.5.3.2. Настройка конфигурации приложения Аварийный архив

Для настройки конфигурации приложения Аварийный архив необходимо открыть специализированный редактор данного приложения. Для этого нужно выделить поле

AlertArchive : ALERT ARCHIVE
Программа/Конфигурация и нажать кнопку (Редактировать). Откроется специализированный редактор приложения Аварийный архив (см. рис. 3.138).

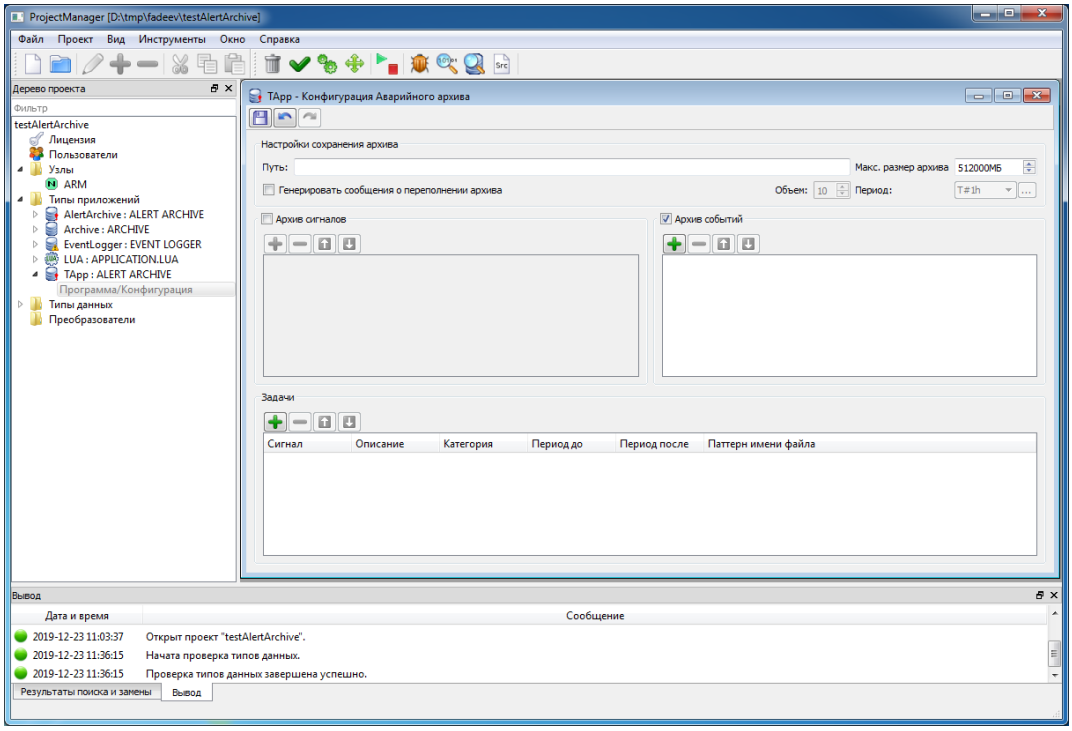


Рисунок 3.138 - Окно конфигурации Аварийного архива

В верхней части окна конфигурации Аварийного архива располагаются следующие кнопки:

- - кнопка сохранения сделанных изменений;
- - кнопка возврата сделанных изменений назад на один шаг;
- - кнопка возврата отмененных изменений на один шаг.

Ниже располагаются настройки сохранения архива:

- **Путь** - путь, по которому будут сохраняться файлы аварийного архива;
- **Макс. размер архива** - максимальный размер, который могут занимать файлы Аварийного архива;
- **Генерировать сообщения о переполнении архива** - если поставить данную галочку, то будут генерироваться сообщения о превышении установленного максимального размера архива. Данные сообщения можно увидеть в функциональном блоке **TEventViewer**;
- **Объём** - количество сообщений, которое будет сгенерированно при превышении установленного максимального размера архива;
- **Период** - периодичность, с которой будут генерироваться сообщения.

В окне **Архив сигналов** необходимо прописать все приложения архивы сигналов, из которых Аварийный архив сможет получать данные. Если архив сигналов нерезервированный, то строка-описатель источника данных содержит единственное имя архива. Если архив сигналов резервированный, то строка должна содержать несколько имен архивов, перечисленных через запятую.

Каждое имя архива задается в формате **Имя_узла.Имя_приложения**. **Имя_приложения** - данное имя указывается в настройках узла для данного приложения.

При срабатывании сигнала-триггера (описано ниже) аварийный архив получает данные из каждого источника. При этом источник данных должен содержать в своем списке хотя бы один доступный архив сигналов, не находящийся в резерве. При возникновении ситуации, когда источник

данных не содержит подходящих архивов, получение данных аварийным архивом будет отложено (через 60 секунд будет повторная попытка).

Управление в окне Архив сигналов осуществляется с помощью следующих кнопок:

-  - добавить новую запись в таблицу архивов сигналов;

-  - удалить выбранную запись из таблицы архивов сигналов;

-  - поднять выделенную запись вверх;

-  - опустить выделенную запись вниз.

Справа от окна со списком архивов сигналов находится окно со списком архивов событий. Данное окно имеет аналогичное описание, но применительно к архивам событий, которые будут использованы в проекте.

Ниже располагается окно списка задач аварийного архива. Оно содержит следующие кнопки и столбцы:

-  - кнопки добавления задач, удаления задач, перемещения вверх и вниз задач в списке;

- **Сигнал** - сигнал (тип BOOL), по переходу которого в значение TRUE, будет отрабатывать задача;

- **Описание** - текст, который будет описывать данную аварию, в аварийном архиве. К Аварийному архиву можно будет подключиться с помощью программы просмотра архивов (Archive Viewer) из дистрибутива СКАДА "Соната". В списке аварий будет отображаться данный текст;

- **Категория** - на данный момент можно не заполнять (не используется);

- **Период до** - время до аварии, за которое в аварийный архив будут помещены значения всех сигналов из архива сигналов и событий из архива событий;

- **Период после** - время после аварии, за которое в аварийный архив будут помещены значения всех сигналов из архива сигналов и событий из архива событий;

- **Паттерн имени файла** - паттерн имени файла, в который будет сохранена информация о данной аварии.

Пример выполненных настроек Аварийного архива представлен на рис. 3.139.

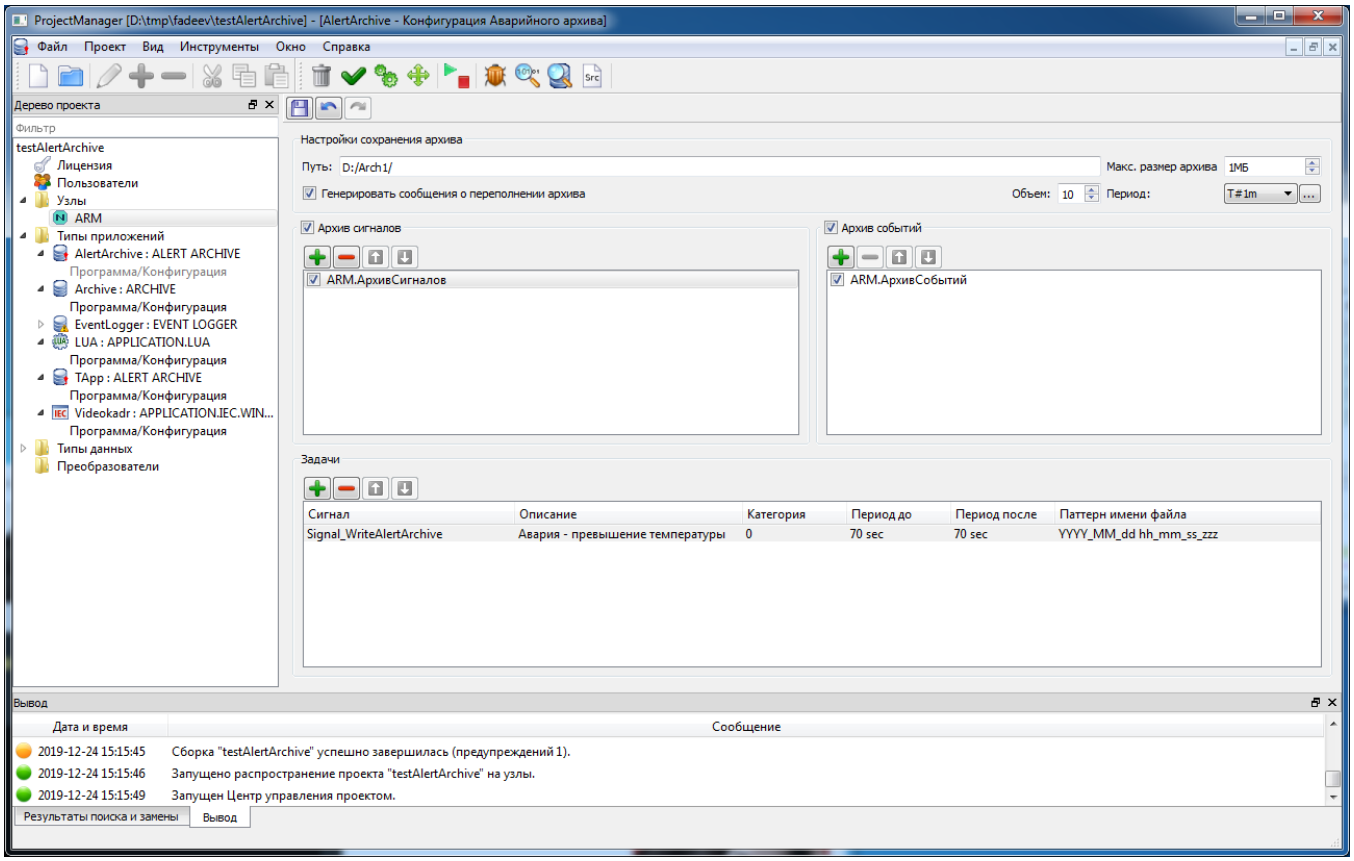


Рисунок 3.139 - Пример настроек Аварийного архива

Обратите внимание! Если сработала определенная задача и в период, который отведен для сохранения данных данной задачи, эта же задача сработает еще раз, то будет получен один общий результат по её отработке. Но в данном результате будут сохранены данные, которые охватывают весь необходимый период, т.е. начало данных будет от первой задачи, а концовка данных от второй задачи. И т.д.

3.3.5.3.3. Пример применения приложения Аварийный архив

В данном разделе описан пример настроек в проекте АСУ ТП с применением Аварийного архива. С датчика температуры снимаются значения и при превышении температуры 100 градусов цельсия будет срабатывать Аврийный архив. В результате мы сможем подключиться к данным аварийного архива и проанализировать ситуацию в момент аварии.

Последовательность настройки проекта:

- завести глобальные сигналы в таблицу глобальных сигналов проекта;
- добавить и настроить приложение архив сигналов (ARCHIVE);
- добавить и настроить приложение архив событий (EVENT LOGGER);
- добавить и настроить узел в проекте;
- добавить и настроить приложение аварийный архив (ALERT ARCHIVE);
- написать алгоритм обработки значений в проекте;
- добавить приложение с графическим отображением событий и аварий в системе (APPLICATION.IEC.WINDOW).

В глобальные переменные проекта добавлены следующие переменные (см. рис. 3.140):

- **Temperature:REAL** - отображает значение температуры с предполагаемого датчика;

- **G1_INT:INT** - сигнал для значений в проекте;
- **G2_BOOL:BOOL** - сигнал для значений в проекте;
- **Signal_WriteAlertArchive:BOOL** - сигнал, по переходу которого в значение TRUE, будет выполняться задание в приложении Аварийный архив.

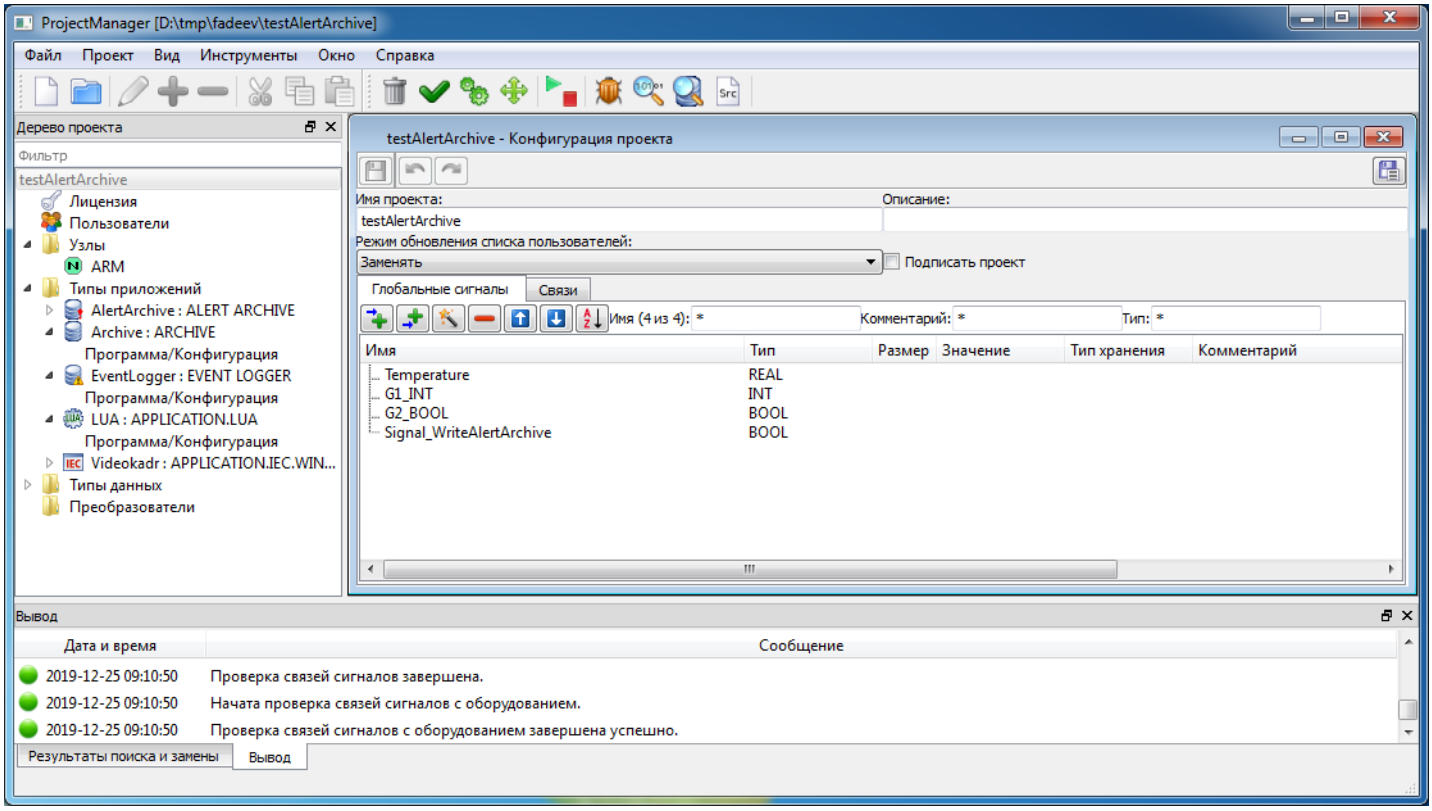


Рисунок 3.140 - Глобальные сигналы в нашем проекте

Вторым шагом в проект добавлено и настроено приложение архив сигналов (ARCHIVE, см. рис. 3.141).

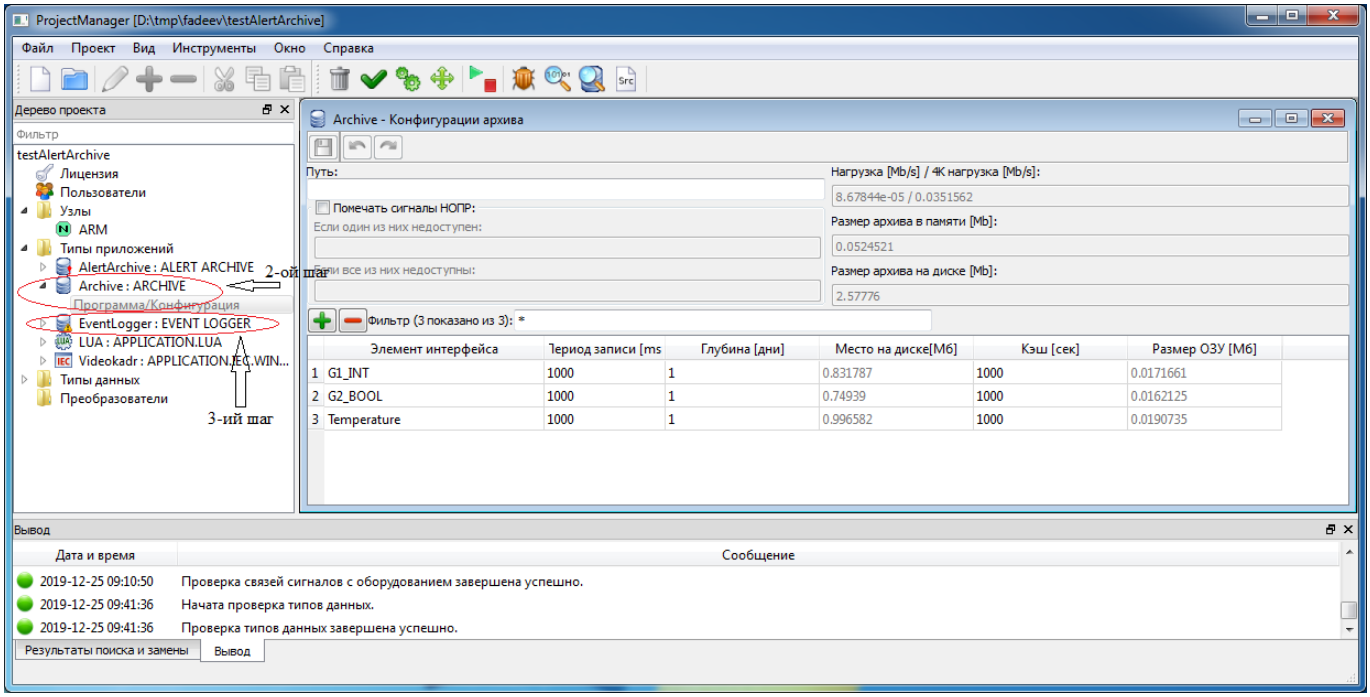


Рисунок 3.141 - Настройки приложения архив сигналов

Третьим шагом в проект добавлено приложение архив событий (EVENT LOGGER). Данное приложение не будем настраивать и оставим с настройками по-умолчанию.

Четвертым шагом добавим и настроим узел в нашем проекте (см. рис. 3.142). В настройках узла обратите внимание на имя узла и имена из столбца Имя в списке приложений. Эти данные будут использоваться при настройке аварийного архива.

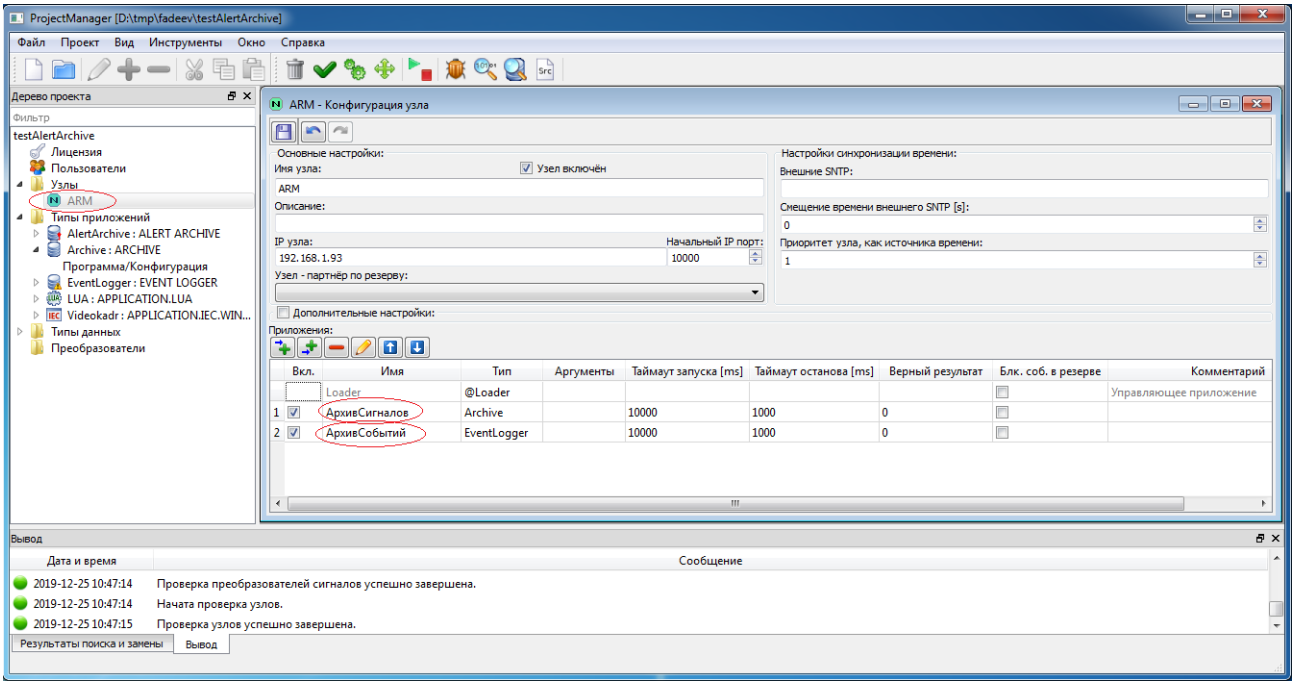


Рисунок 3.142 - Настройки узла проекта

Пятым шагом добавим и настроим приложение аварийный архив. В интерфейс приложения добавим сигнал **Signal_WriteAlertArchive:BOOL**, по его переходу в значение TRUE будет срабатывать наше задание в аварийном архиве. Настроенный Аварийный архив смотрите на рис. 3.143.

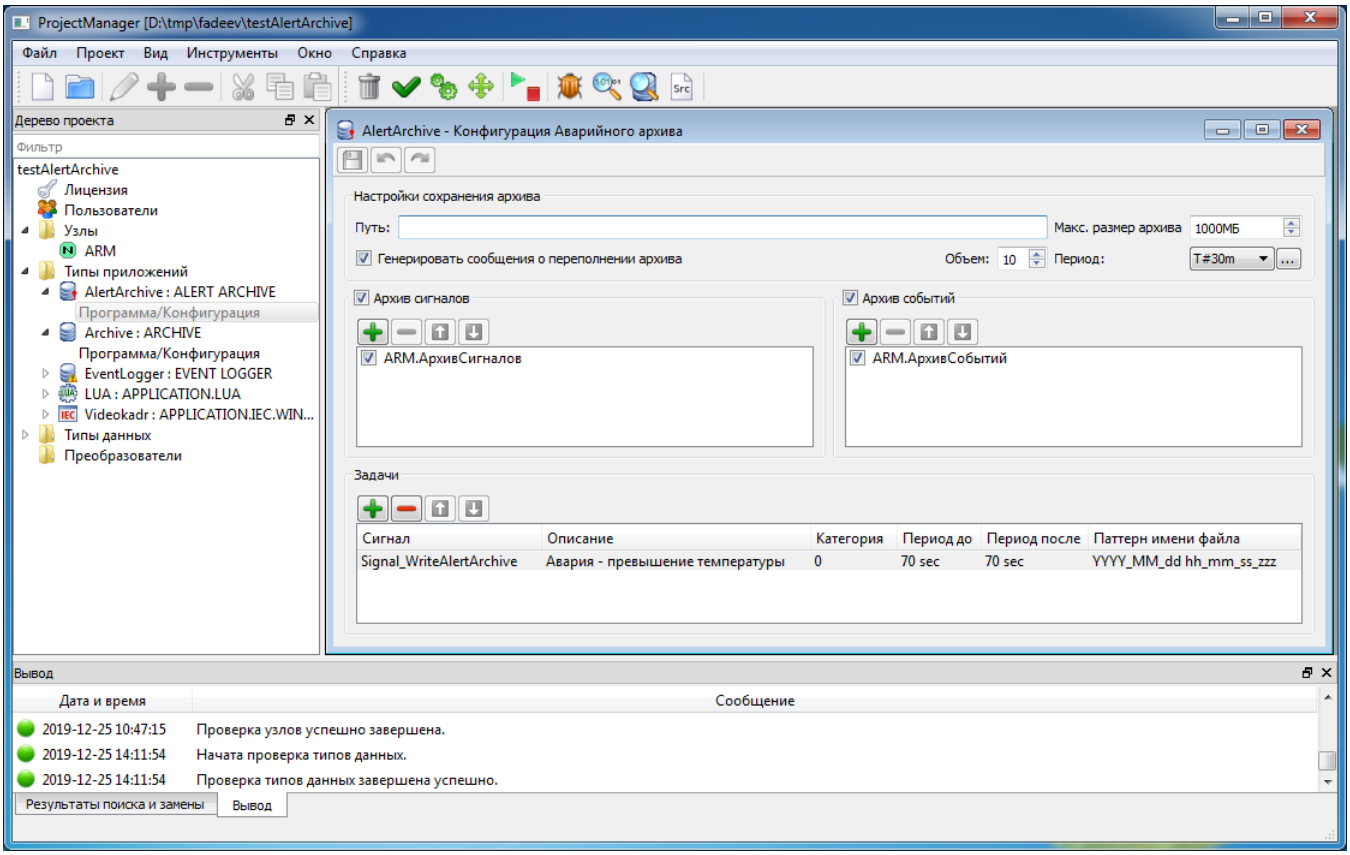


Рисунок 3.143 - Настроенный Аварийный архив

Данный Аварийный архив будет сохранять свои файлы в директории по умолчанию, т.е в папке с дистрибутивом СКАДА "Соната". В данной папке будет создана вложенная папка Node_UUID.NNN, где UUID - уникальный идентификационный номер узла (можно посмотреть во всплывающей подсказке, если навести курсор мыши на узел в дереве проекта), NNN - порядковый номер приложения Аварийный архив в списке приложений на узле. Далее смотрите вложенную папку Storage.

Допустимый максимальный размер данного архива мы установили 1000 МБ и настроили, что при превышении данного значения будут генерироваться информационные события в системе. Будут сгенерированы 10 событий с периодичностью 30 минут.

Значения сигналов будут браться из архива сигналов **АРМ.АрхивСигналов**, а события будут браться из архива событий **АРМ.АрхивСобытий**. Так мы их назвали в списке приложений узла АРМ.

В списке задач настроена одна задача. По переходу сигнала **Signal_WriteAlertArchive:BOOL** в значение TRUE будет срабатывать Аварийный архив. Исходя из наших настроек это следующие действия - в файл аварийного архива, с именем в соответствии с заданным нами паттерном, будет сохранена информация о всех сигналах и событиях из **АРМ.АрхивСигналов** и **АРМ.АрхивСобытий**. Информация будет сохранена за промежуток времени 70 секунд до аварии и 70 секунд после аварии.

Шестым шагом добавим алгоритм обработки значений в наш проект. Пусть это будет приложение **APPLICATION.LUA** (см. рис. 3.144). В данном алгоритме каждый цикл изменяются значения глобальных сигналов **G1_INT** и **G2_BOOL**, так же написано условие, что при выходе значения сигнала **Temperature** за 100 будет взводиться сигнал **Signal_WriteAlertArchive:BOOL** в TRUE, а значит сработает задача в аварийном архиве. Дополнительно в данном условии будет генерироваться событие с текстом "Превышение температуры".

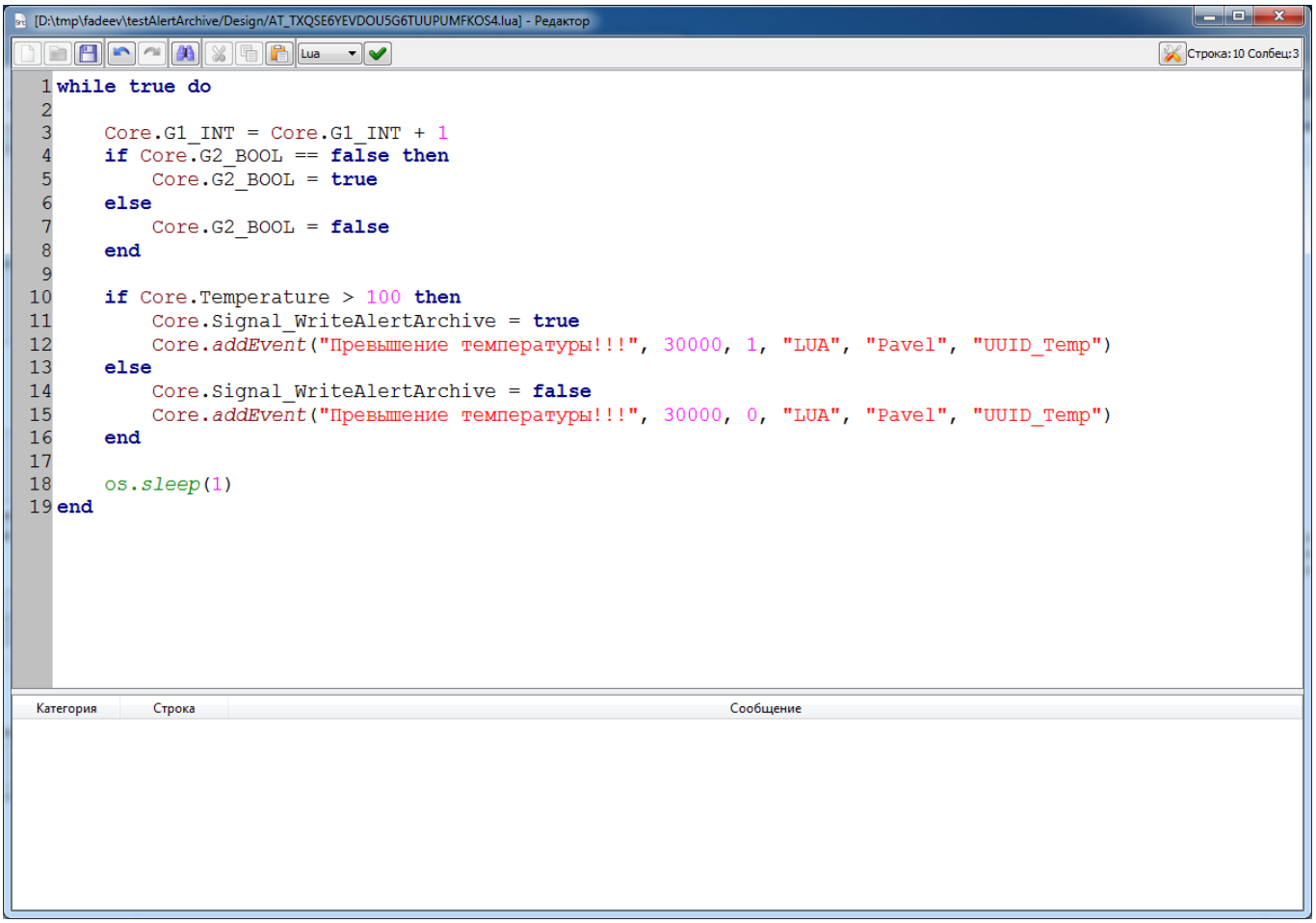


Рисунок 3.144 - Алгоритм LUA по обработке значений в нашем проекте

Седьмым шагом можно добавить приложение с графическим отображением аварий и событий в проекте (**APPLICATION.IEC.WINDOW**). Как его настраивать в данном примере рассматривать не будем.

Следующим шагом добавим все наши приложения в список приложений на узел (см. рис. 3.145).

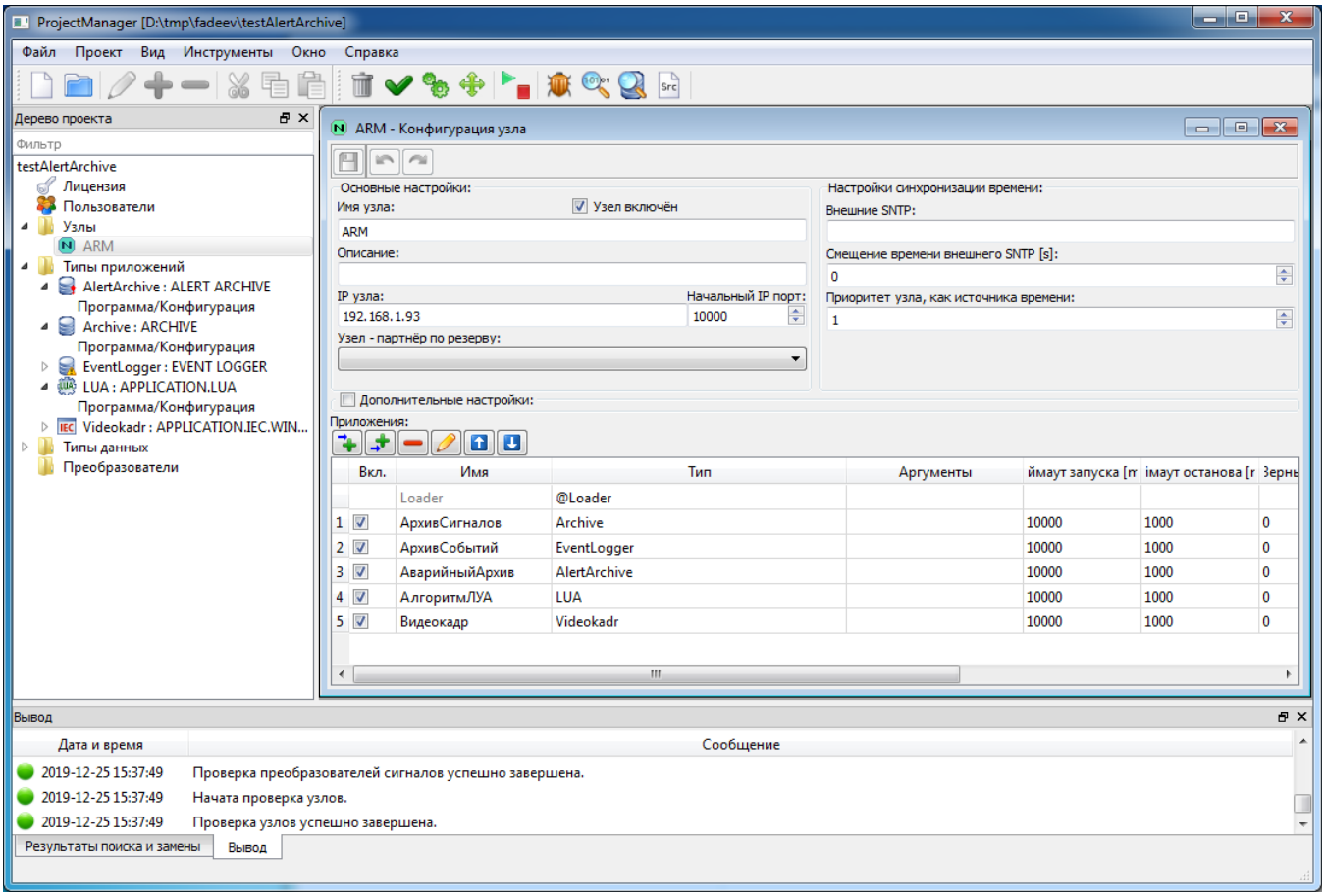


Рисунок 3.145 - Узел с полным списком приложений нашего проекта

Далее компилируем , рассылаем и запускаем проект .

Для имитации выхода нашей температуры за 100 градусов цельсия необходимо запустить приложение просмотра и управления сигналами СКАДА "Соната" (**Signal Viewer**). Это делается нажатием на кнопку . Далее добавляем сигнал **Temperature** и ставим значение больше 100 градусов. При этом сработает наше задание в аварийном архиве.

Просмотр результа работы аварийного архива доступен через приложение просмотра архивов СКАДА "Соната" (**Archive Viewer**). Данное приложение запускается кнопкой (см. рис. 3.146).

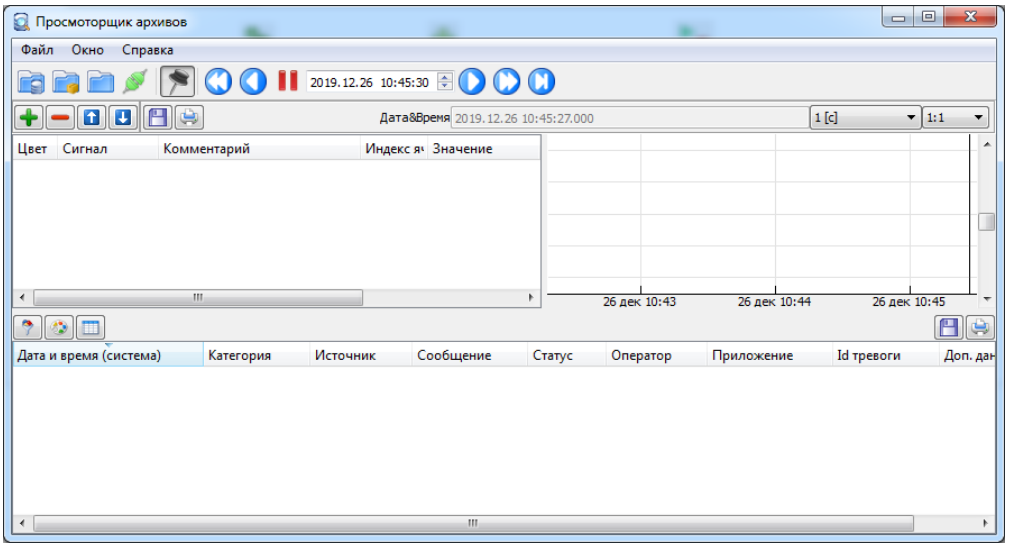


Рисунок 3.146 - Окно приложения просмотра архивов (ARCHIVE VIEWER)

Далее в окне приложения просмотра архивов нужно нажать на кнопку  . Откроется окно, в котором нужно выбрать папку с местоположением файла аварийного архива. После этого откроется окно со списком аварий, зафиксированных аварийным архивом (см. рис. 3.147).

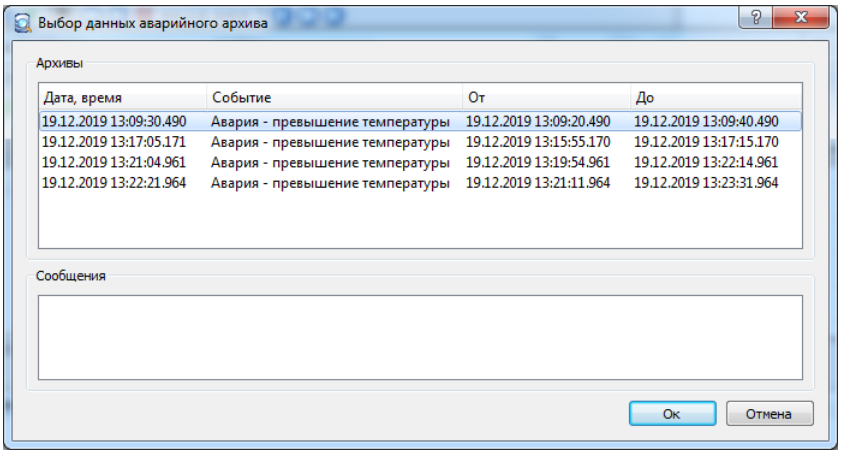


Рисунок 3.147 - Окно выбора аварии из списка аварий аварийного архива

Необходимо выбрать нужную аварию и нажать кнопку ОК. После этого в окне просмотра архивов подгрузится информация о сигналах и отобразятся события и аварии за заданный в нашем задании промежуток времени.

3.3.5.4. Особенности настройки приложения Архив сигналов (ARCHIVE), которое используется в работе приложения Аварийный архив (ALERT ARCHIVE)

В настройках приложения Архив сигналов есть возможность указать папку, в которой будут храниться архивные файлы. По умолчанию архивные файлы хранятся в специальной папке, которая находится внутри директории Среды исполнения. Пусть это будет дирректория F:\Sonata-1-4 (папка Среды исполнения) и специальная папка Node_DTZ2L4EVCYJUHBP5N67WWEUREY.002 (имя_узла_в_системе.порядковый_номер_приложения_Архив_сигналов_на_узле). На рис. 3.148) указано, что архивные файлы будут сохраняться в папку F:\ArchiveOfSignals.

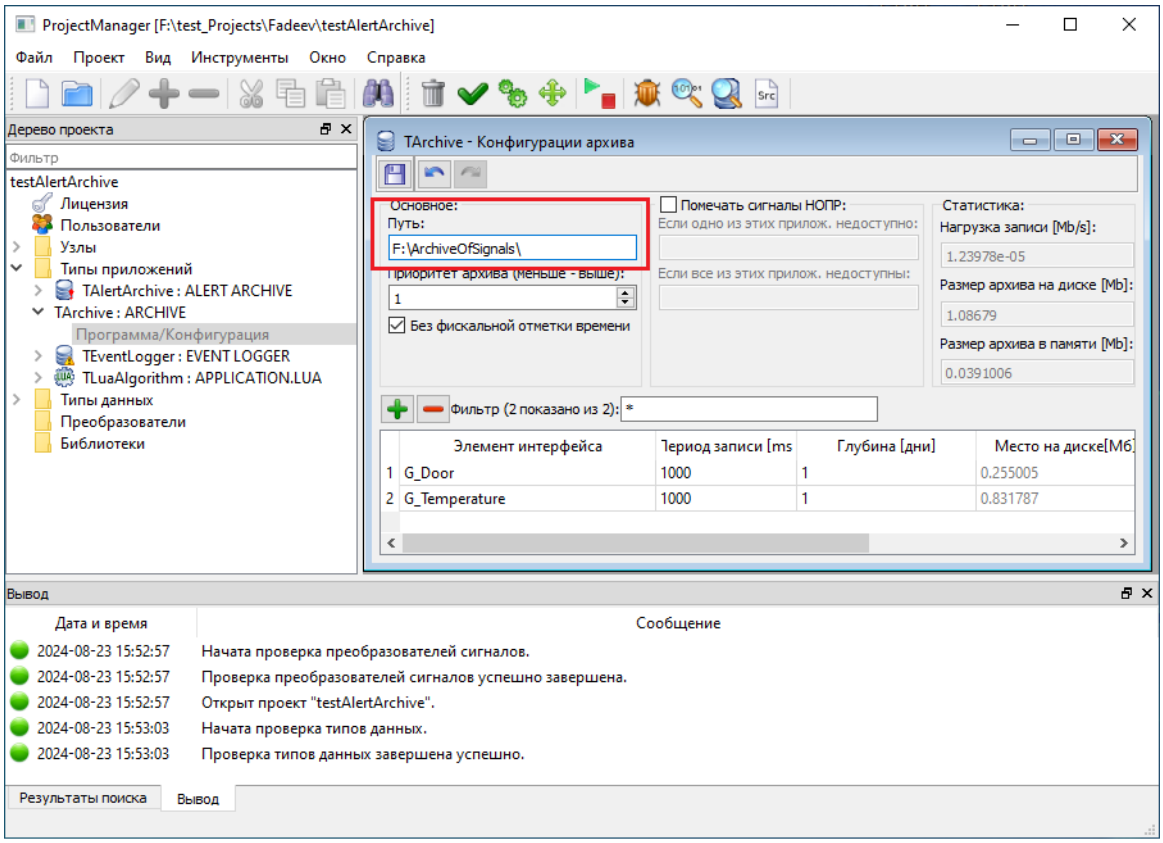


Рисунок 3.148 - Настройки приложения Архив сигналов

ВНИМАНИЕ! Если приложение Архив сигналов хранит архивные файлы вне директории Среды исполнения, то приложение Аварийный архив не сможет работать с данным Архивом сигналов. Это связано с ограничениями по Информационной безопасности.

Если вам всё же необходимо хранить архивные файлы приложения Архив сигналов вне директории Среды исполнения, то для этого можно использовать символические ссылки на папку.

К примеру, чтобы приложение Архив сигналов сохраняло архивные файлы в директории F:\ArchiveOfSignals и при этом данный Архив сигналов можно было указывать в настройках приложения Аварийный архив, необходимо:

1) Создать директорию F:\ArchiveOfSignals;

2) Создать символическую ссылку на данную директорию. При этом символическая ссылка должна находиться внутри директории Среды исполнения. В операционной системе Windows это можно сделать командой:

mklink \j "F:\Sonata-1-4\ArchiveOfSignals" "F:\ArchiveOfSignals", где "F:\Sonata-1-4\ArchiveOfSignals" - ссылка на директорию "F:\ArchiveOfSignals". В результате выполнения данной команды будет создана директория F\Sonata-1-4\ArchiveOfSignals, которая является ссылкой на директорию F:\ArchiveOfSignals.

В операционной системе семейства Linux подобную операцию можно выполнить командой:

ln -s "/home/user/ArchiveSignalsFiles" "/home/user/sonata/Runtime/ArchiveSignalsFiles", где /home/user/ArchiveSignalsFiles - директория, в которой мы хотим хранить архивные файлы, /home/user/sonata/Runtime/ArchiveSignalsFiles - символическая ссылка, которая будет создана внутри директории Среды исполнения.

3) В настройках приложения Архив сигналов в поле Путь нужно указать директорию, которая является ссылкой на основную директорию (см. рис. 3.149).

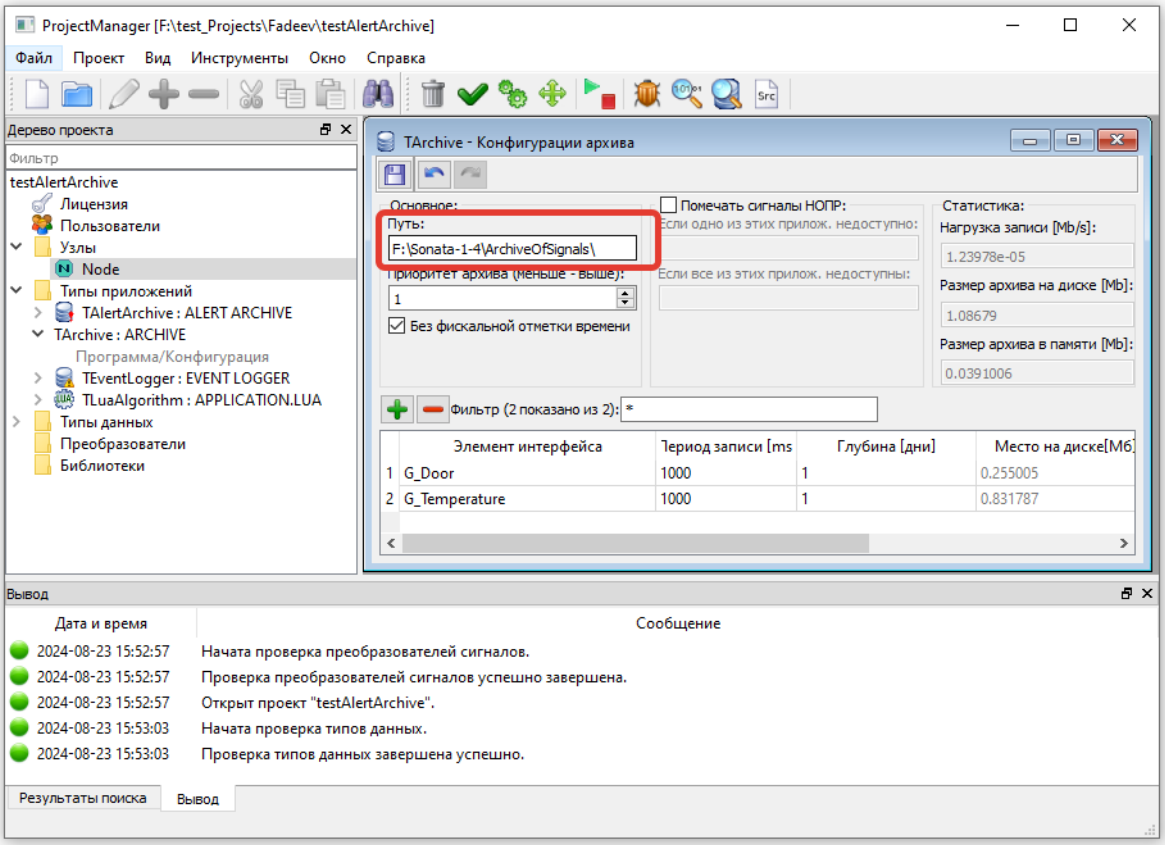


Рисунок 3.149 - Настройки приложения Архив сигналов для возможности работать с приложением Аварийный архив

Данное приложение Архив сигналов будет сохранять свои архивные файлы по ссылке F:\Sonata-1-4\ArchiveOfSignals и они будут доступны в директории F:\ArchiveOfSignals.

3.3.6. Технологические программы

3.3.6.1. Описание приложения REPORT ENGINE (управления отчетами и создания шаблонов отчетов)

SCADA-система "Соната" может генерировать отчеты по наступлению определенных событий. Для этого используется приложение **REPORT ENGINE**, которое предназначено для настройки конфигурации отчетов, задания условий генерации, а также создания и редактирования шаблонов отчетов.

3.3.6.1.1. Создание приложения REPORT ENGINE

Для создания приложения **REPORT ENGINE** сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.150 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** (необходимо выбрать **REPORT ENGINE**).

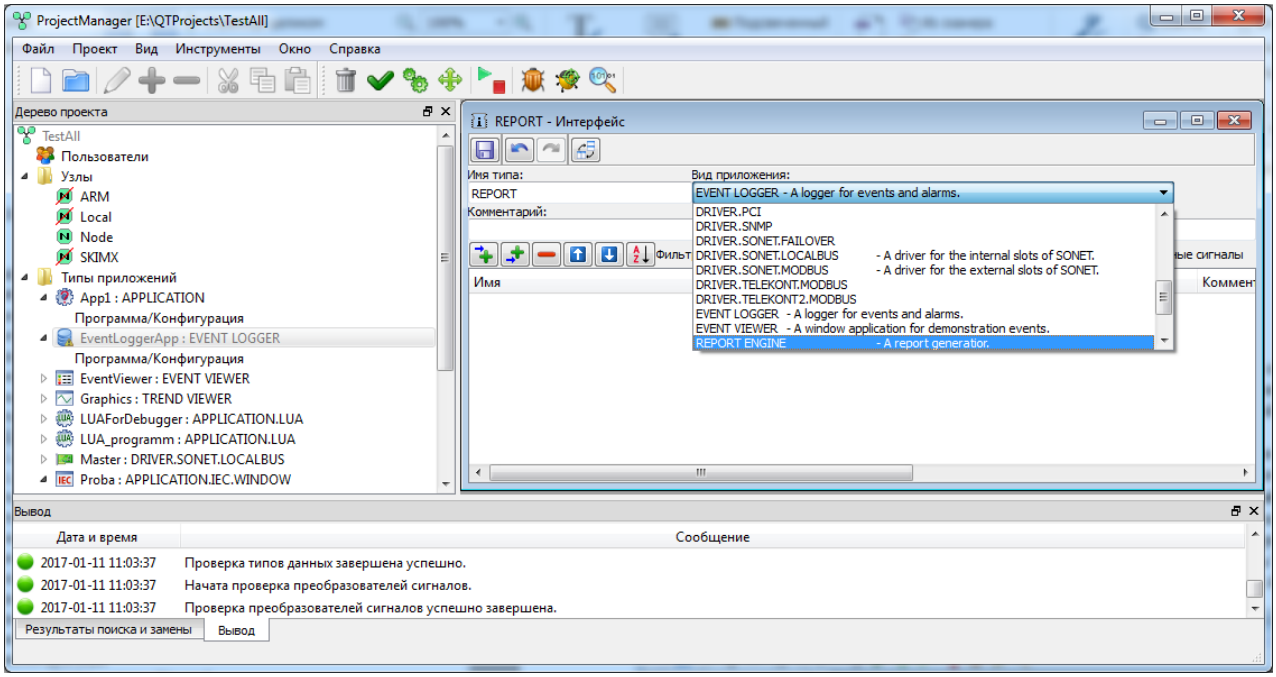


Рисунок 3.150 - Изменение Имени типа и Вида приложения

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.151) – для подтверждения изменения необходимо нажать кнопку Да.

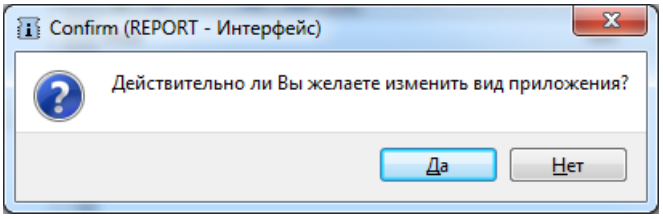


Рисунок 3.151 - Окно подтверждения для изменения вида типа приложения

3.3.6.1.2. Настройка интерфейса приложения REPORT ENGINE

После выбора вида приложения (в данном случае **REPORT ENGINE**) необходимо добавить в интерфейс приложения сигналы SCADA-системы "Соната", которые будут использоваться приложением **REPORT ENGINE**. Процесс добавления сигналов в интерфейс приложения подробно описан ранее (см. раздел 3.2.2.4.3).

На рис. 3.152 в интерфейс приложения добавлены сигналы **GenerateReport** и **RiseSignal**.

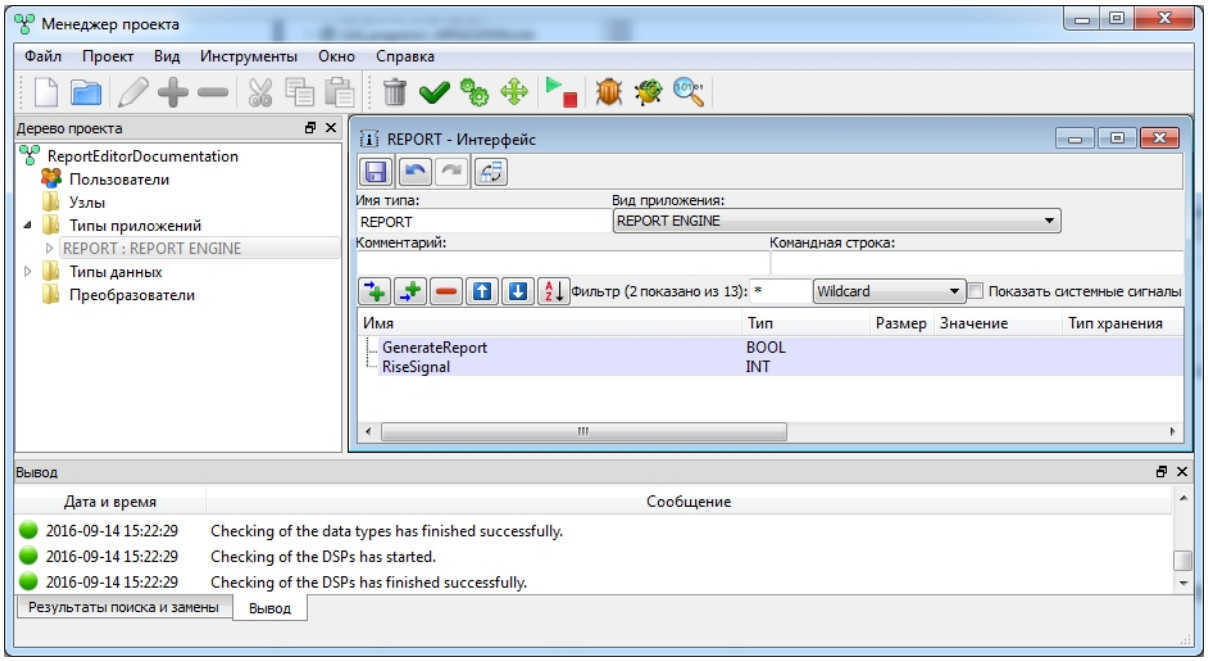


Рисунок 3.152 - Интерфейс приложения REPORT ENGINE

Изменения следует зафиксировать с помощью кнопки  ("Сохранить") в окне редактирования интерфейса типа приложения.

3.3.6.1.3. Вызов и описание специализированного редактора приложения REPORT ENGINE

Для дальнейшей настройки приложения **REPORT ENGINE** необходимо вызвать специализированный редактор. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

Главное окно редактора приложения **REPORT ENGINE** представлено на рис. 3.153.

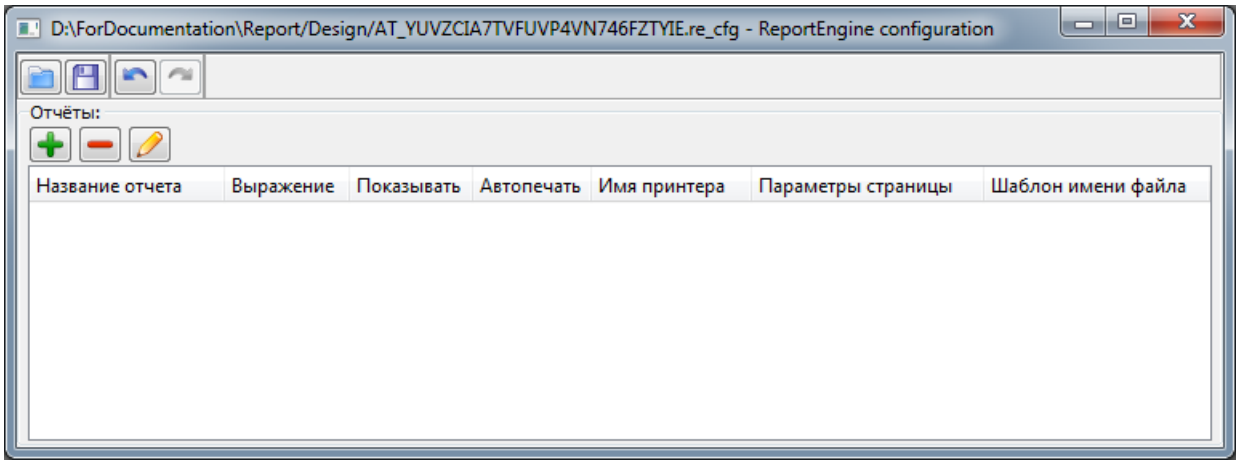


Рисунок 3.153 - Редактор приложения REPORT ENGINE

В верхней части окна редактора приложения **REPORT ENGINE** расположена панель инструментов со следующими кнопками:

 - сохранить внесённые в данный элемент проекта изменения;

- отменить предыдущее действие;

- повторить отменённое действие.

Ниже находятся кнопки для изменения конфигурации:

- добавить отчет (с таким же эффектом можно нажать клавишу **Insert** на клавиатуре);

- удалить отчет, будут удалены все выделенные строки (нажатие этой кнопки равносильно нажатию клавиши **Delete** на клавиатуре);

- редактировать отчет.

Ниже располагается таблица с настроенными пользователем отчетами, которые может генерировать данное приложение. Поля в данной таблице будут заполнены автоматически после создания нового отчета или редактирования имеющегося.

В данной таблице находятся следующие столбцы:

- Название отчета - имя отчета, которое используется только в данной таблице;

- Выражение - описывает условие, при котором будет сгенерирован данный отчет;

- Показывать - в данном столбце находится элемент чекбокс, активное состояние которого сообщает, что при генерации отчета он будет выведен на экран для предварительного просмотра, в противном случае не будет выведен для предварительного просмотра;

- Автопечать - в данном столбце находится элемент чекбокс, активное состояние которого сообщает, что при генерации отчет будет автоматически отправлен на печать на настроенный принтер;

- Имя принтера - имя принтера по умолчанию, на который происходит печать;

- Параметры страницы - параметры страницы для печати и сохранения отчета;

- Шаблон имени файла - шаблон имени файла для сохранения отчета.

При нажатии на кнопку появляется окно диалога добавления и конфигурации нового отчета (см. рис. 3.154).

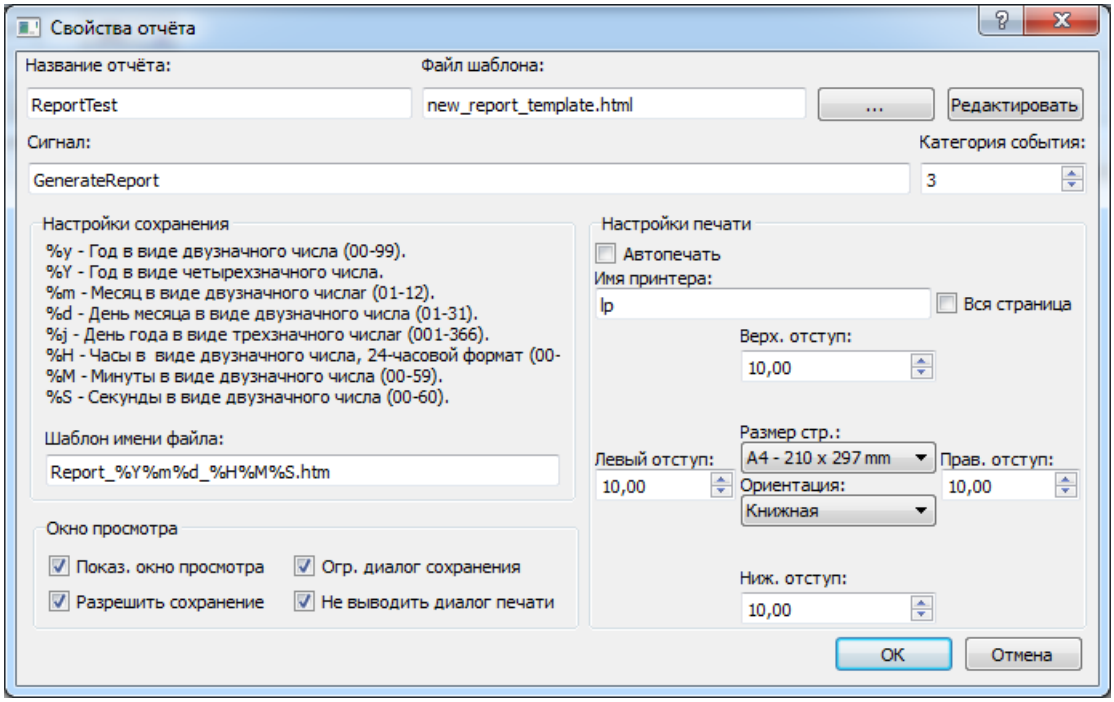


Рисунок 3.154 - Окно добавления и редактирования конфигурации отчета

На форме диалога присутствуют следующие поля:

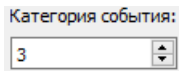
- **Название отчета** – поле с названием отчета, которое будет отображаться в таблице со всеми настроенными пользователем отчетами;

- **Файл шаблона -** имя файла шаблона отчета;

-  - выбор файла шаблона на диске;

-  - редактирование выбранного файла шаблона или создание нового;

- **Сигнал –** имя сигнала, который запускает генерацию данного отчета (тип BOOL);

-  - категория событий, к которой будет относится событие о генерации отчета;

На рис. 3.155 представлена левая часть окна редактирования конфигурации отчета, на нём описаны варианты настройки формата имени отчёта при сохранении.

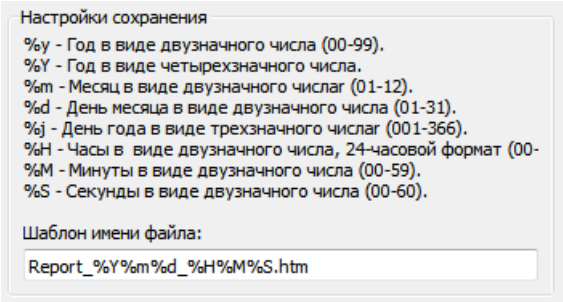


Рисунок 3.155 - Настройки сохранения

Шаблон имени файла - шаблон, по которому будет формироваться имя файла отчета. Обычные символы в названии файла не преобразуются. Символы, определяющие преобразования, предваряются символом %. В итоговом названии файла их заменят следующие символы:

- %a (аббревиатура названия дня недели в зависимости от текущей локали);

- %A (полное название дня недели в зависимости от текущей локали);

- %b (аббревиатура названия месяца в зависимости от текущей локали);

- %B (полное название месяца в зависимости от текущей локали);

- %c (предпочтительный формат даты и времени для текущей локали);

- %C (век (год/100) как целое двухразрядное число);

- %d (день месяца в десятичной форме (от 01 до 31));

- %D (формат даты в %м/%д/%г (только для американцев, в других странах обычная форма - %д/%м/%г .);

- %e (то же, что и %d: десятичное число, обозначающее номер дня месяца, но вместо начального нуля ставится пробел);

- %E (модификатор: использовать другой формат, см. далее);

- %F (эквивалент %Y-%m-%d (формат даты ISO 8601));

- %G (указывает год как четырехзначное число (по стандарту ISO 8601). Имеет тот же формат и значение, что и %y, но если неделя входит также в прошедший (или последующий) год (в соответствии со стандартом ISO о номерах недель), то в этом случае отображается прошедший год .);

- %g (то же, что и %G, но без первых двух чисел, то есть двухразрядное число, 00-99);

- %h (аналогично %b);

- %H (показывает час как десятичное число от 00 до 23);

- %I (показывает час как десятичное число от 01 до 12);

- %j (показывает день года как десятичное число от 001 до 366);

- %k (показывает час как десятичное число от 0 до 23; начальные нули заменяются пробелами (см. также %H,);

- %l (показывает час, как десятичное число (от 1 до 12); начальные нули заменяются пробелами (см. также %I);

- %m (показывает месяц как десятичное число от 01 до 12);

- %M (показывает минуты как десятичное число от 00 до 59);

- %n (символ новой строки);

- %O (модификатор: использовать другой формат, см. далее);

- %p (показывает `AM' или `PM', в зависимости от времени суток или от соответствующей локали. Полдень обозначен как `pm', а полночь - как 'am'.);

- %P (то же, что и %p, но в нижнем регистре (например, `pm'));

- %r (сообщает о времени с указанием a.m. или p.m. В локали POSIX это соответствует `%I:%M:%S %p' .);

- %R (показывает время в формате 24-х часов (%H:%M). В варианте %T, описанном ниже, указаны еще и секунды .);

- %s (показывает количество секунд с начала 1970-01-01 00:00:00 UTC);

- %S (отображает секунды в десятичной форме от 00 до 61);

- %t (символ табуляции);

- %T (показывает время в 24-часовом формате %H:%M:%S);

- %u (показывает день недели как десятичное число от 1 до 7. Понедельник считается равным 1. См. также %w .);

- %U (показывает номер недели текущего года в виде десятичного числа от 00 до 53, начиная с первого воскресенья как первого дня первой недели. См. также %V и %W.);

- %V (по стандарту ISO 8601:1988 номер недели в году отображается в виде десятичного числа (от 01 до 53), где первая неделя - это неделя, минимум 4 дня которой находятся в текущем году. Первым днем недели считается понедельник. См. также %U и %W.);

- %w (показывает день недели как десятичное число от 0 до 6, и воскресенье считается равным нулю. См. также %u.);

- %W (показывает номер недели в году как десятичное число от 00 до 53, первый понедельник считается первым днем недели номер 01);

- %x (показывает дату в формате, указанном в текущей локали, без времени);

- %X (показывает время в формате, указанном в текущей локали, без даты);

- %y (показывает год как двухразрядное число от 00 до 99, без указания века);

- %Y (показывает год как четырехразрядное десятичное число (с указанием века));

- %z (показывает часовой пояс как смещение от GMT (Гринвича). Требует совместимости с RFC822 (форматы: "%a, %d %b %Y %H:%M:%S %z");

- %Z (показывает часовой пояс, или его название, или аббревиатуру);

- %+ (показывает дату и время в формате date(1));

- %% (символ `%').

Ниже располагаются настройки для окна предварительного просмотра отчета (см. рис. 3.156).

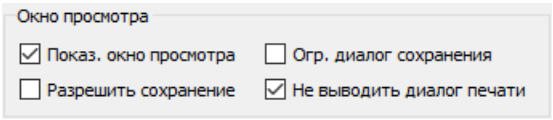


Рисунок 3.156 - Настройки для окна предварительного просмотра отчета

- **Показ. окно просмотра** - данный элемент чекбокс в активном состоянии сообщает, что при генерации отчета он будет выведен на экран для предварительного просмотра, в противном случае не будет выведен для предварительного просмотра;

- **Разрешить сохранение** - данный элемент чекбокс в активном состоянии сообщает, что в окне предварительного просмотра будет кнопка сохранения отчета, в противном случае кнопки сохранения не будет;

- **Огр. диалог сохранения** - данный элемент чекбокс в активном состоянии сообщает, что при сохранении отчета диалог сохранения будет настроен на рабочую папку проекта и не даст сохранить отчет в другое место;

- **Не выводить диалог печати** - данный элемент чекбокс в активном состоянии сообщает, что при нажатии на кнопку печать в окне просмотра отчета, не будет выведен диалог печати. Печать будет произведена на принтер, который настроен в редакторе отчётов.

В правой части диалогового окна находятся настройки, используемые при печати (см. рис. 3.157):

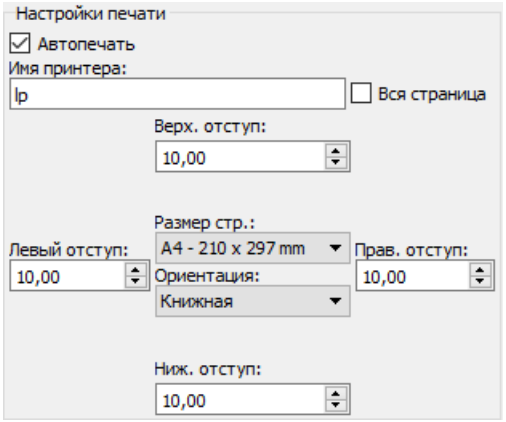


Рисунок 3.157 - Настройки печати отчета

- **Автопечать** – автоматически отправлять на печать. Если стоит данная галка, то отчет после генерации будет отправлен на печать на принтер, указанный в поле **Имя принтера**;

- **Имя принтера** – принтер, на который посылается отчет, если стоит галка **Автопечать**;

- **Вся страница** - данный элемент чекбокс в активном состоянии сообщает, что вся страница будет растянута на область печати;

- **Ориентация** – ориентация страницы (альбомная / книжная);

- **Левый отступ** – отступ слева;

- **Верх. отступ** – отступ сверху;

- **Прав. отступ** – отступ справа;

- **Ниж. отступ** – отступ снизу;

- **Размер стр.** – размер листа бумаги, задается стандартными обозначениями (A3, A4, …).

По нажатию на кнопку Редактировать, если не был выбран файл шаблона на диске кнопкой ..., будет предложено создать новый файл шаблон. При этом появится окно, в котором надо ввести имя нового файла (см. рис. 3.158)

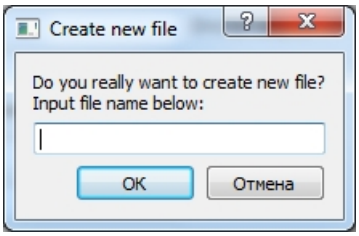


Рисунок 3.158 - Окно создания нового файла шаблона

Если файл был выбран или введено имя нового файла – откроется окно редактора шаблонов (см. рис. 3.159).

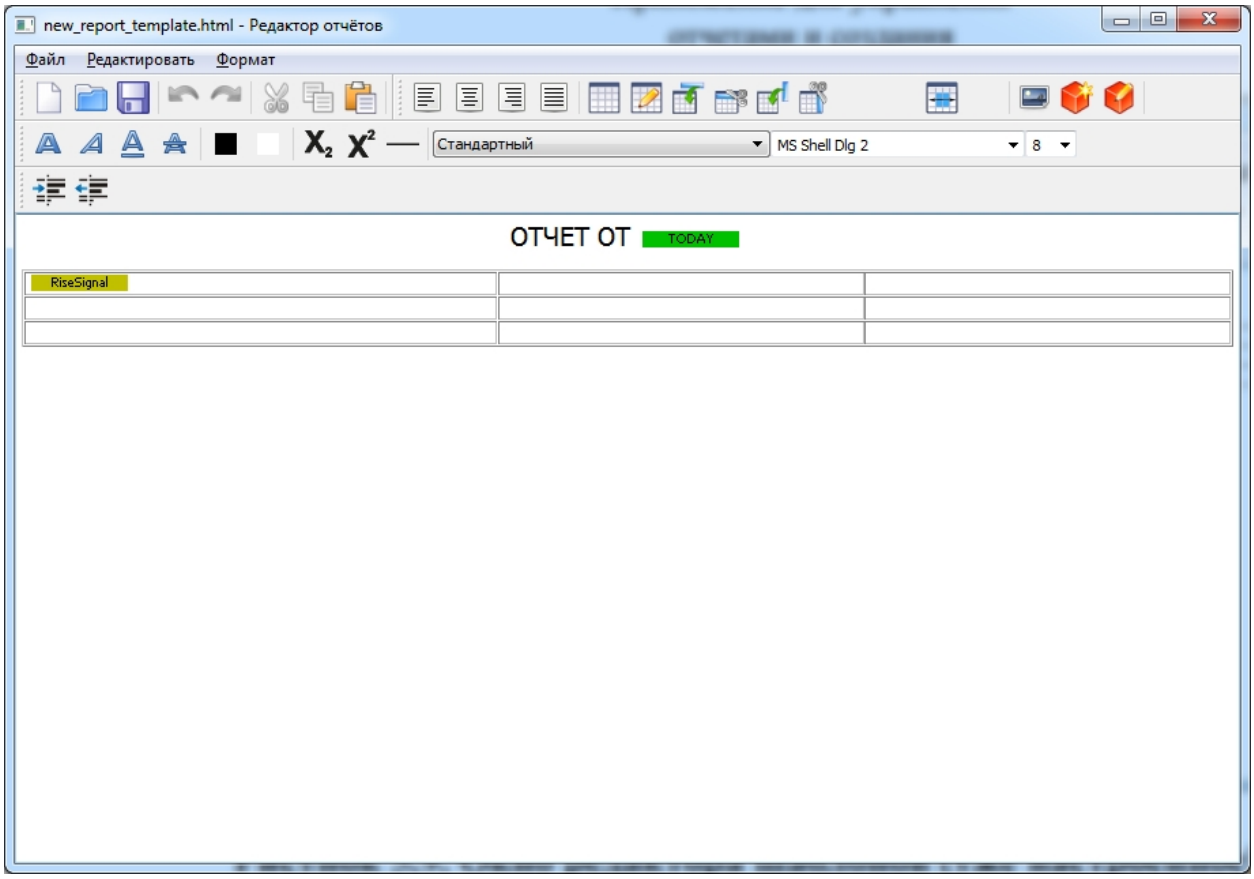


Рисунок 3.159 - Окно редактора шаблонов (уже настроенное)

В верхней части окна расположены следующие инструменты для управления и редактирования шаблонов:

- создать новый шаблон;
- открыть файл шаблона;
- сохранить внесённые изменения;
- отменить предыдущее действие;
- повторить отменённое действие;
- копировать;
- вырезать;

 - вставить.

Далее идут кнопки главной панели инструментов:

 - выравнивание по правому краю;

 - выравнивание по левому краю;

 - выравнивание по ширине;

 - выравнивание по центру.

Далее идут инструменты для работы с таблицами:

 - добавить таблицу;

 - вставить строку;

 - удалить строку;

 - вставить столбец;

 - удалить столбец;

 - ширина столбца;

 - объединить выбранные ячейки;

 - разъединить объединенные ячейки (при нажатии этой кнопки откроется окно, в котором надо задать новое количество столбцов и строк в объединении);

 - цвет ячейки.

Далее находится панель инструментов для работы с изображениями и служебными токенами:

 - вставить изображение;

 - вставить служебный токен;

 - редактировать служебный токен.

При одинарном нажатии на картинку или изображение служебного токена в тексте шаблона произойдет выделение объекта. После этого объект можно копировать, перетаскивать, зажав левую кнопку мышки. Двойной щелчок мышки по картинке или изображению служебного токена приведет к открытию диалога редактирования данного объекта.

Ниже расположена панель инструментов для редактирования текста:

 - жирный текст;

 - наклонный текст;

 - подчеркнутый текст;

 - зачеркнутый текст.

Далее расположены две кнопки для редактирования цвета. Первая – цвет текста, вторая – цвет фона. За ними идут 2 кнопки расположения текста относительно строки:

 - подстрочный символ;

 - надстрочный символ;

 - горизонтальная черта.

Следом расположены выпадающий список стилей списков, список шрифтов и размер шрифта. Далее следуют две кнопки изменения отступа:

 - увеличение отступа;

 - уменьшение отступа;

Все перечисленные выше функции доступны через контекстное меню, которое вызывается нажатием правой кнопки «мышки».

По нажатию на кнопку  - **Вставить служебный токен** откроется окно **Редактировать объект**, в котором в поле **Тип токена** выбираются 4 варианта объектов: **Значение сигнала**, **Дата/время**, **Выражение Lua** и **Код на Lua**.

На рис. 3.159 добавлены два значения: TODAY(см. рис. 3.160) и RiseSignal(см. рис. 3.161).

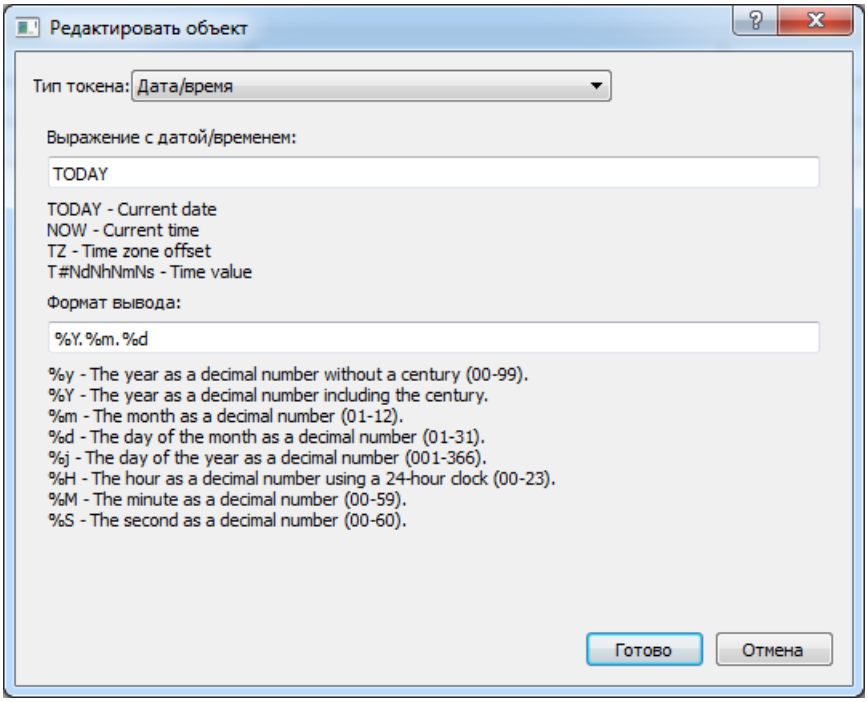


Рисунок 3.160 - Настройки поля для даты и времени

Настройки типа объекта отчета - **Дата/время**: (см. рис. 3.160)

- **Выражение с датой/временем** - Дата и время;

TODAY - подставляет текущую дату по UTC;

NOW - подставляет текущее время по UTC;

TZ - подставляет смещение текущей временной зоны от UTC;

T#NdNhNmNs - подставляет интервал времени.

Пример:

"TODAY + NOW - TZ" - дата и время текущей временной зоны;

"TODAY + NOW - TZ - T#5d" - дата и время текущей временной зоны 5 суток назад.

- **Формат вывода** – формат вывода даты и времени

Пример (%Y:%M:%D %h:%m:%s) Y - год, M – месяц, D – день, h – часы, m – минуты, s – секунды.

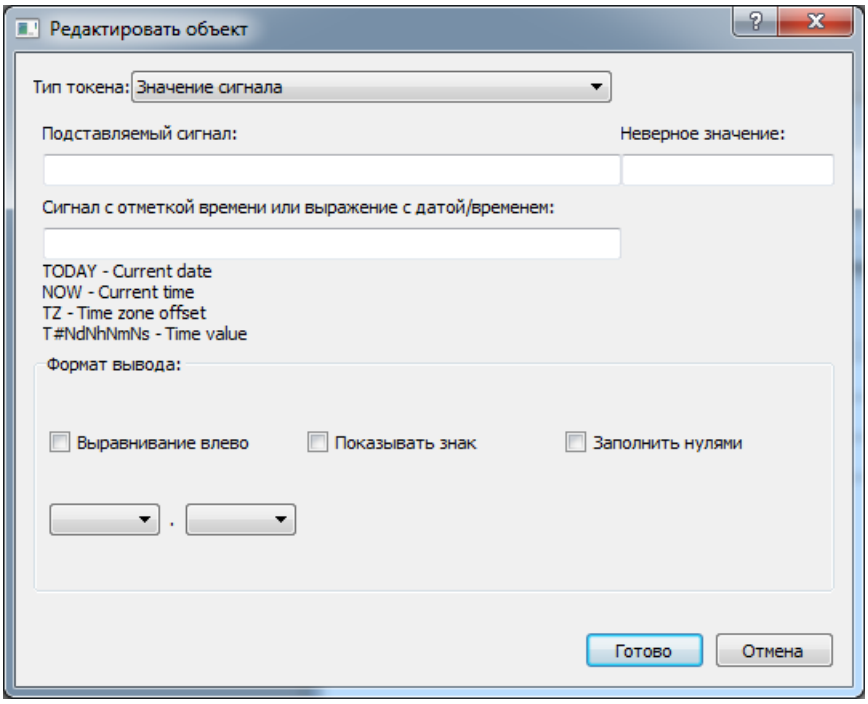


Рисунок 3.161 - Настройки поля для значения сигнала RiseSignal (данная переменная является Глобальной переменной СКАДА СОНАТА и её обязательно нужно добавить в интерфейс приложения)

Настройки типа объекта отчета - **Значение сигнала**: (см. рис. 3.161). Значение сигнала может быть текущим или из архива, в зависимости от значения поля "Сигнал с отметкой времени или выражение с датой/временем". Если поле оставить пустым, то будет браться текущее значение сигнала.

- **Подставляемый сигнал** - имя сигнала;

- **Неверное значение** - значение, которое будет подставляться в сигнал, который прописан в поле **Подставляемый сигнал**, в случае когда не удалось запросить данный сигнал;

- **Сигнал с отметкой времени или выражение с датой/временем** - данное поле служит для указания времени значения сигнала в архиве. В данное поле можно указывать, либо выражение для вычисления отметки времени, либо сигнал, значение которого будет интерпретироваться, как отметка времени для запроса значения сигнала, указанного выше в поле "Подставляемый сигнал". Отметка времени указывается с помощью операторов TODAY, NOW и T#NdNhNmNs;

TODAY - подставляет текущую дату;

NOW - подставляет текущее время;

TZ - подставляет текущую временную зону (**!не нужно использовать!**);

T#NdNhNmNsNms - подставляет интервал времени.

Пример:

"TODAY + NOW - T#00h30m" - получить значение сигнала, которое было 30 минут назад.

- **Формат вывода** - формат вывода значения сигнала:

Выравнивание влево - выравнивание результата влево в пределах заданной ширины;

Показывать знак - принудительное отображение знака числа + или -;

Заполнить нулями - заполнение нулями выводимого числа до достижения указанной ширины;


:

- первая ячейка - ширина выводимого значения;

- вторая ячейка - количество знаков после запятой выводимого значения.

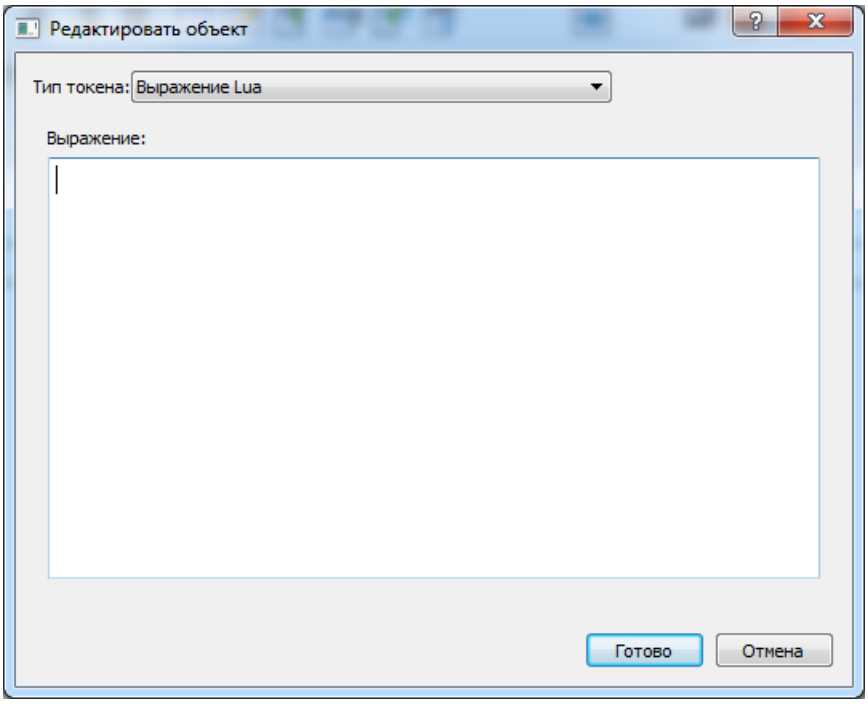


Рисунок 3.162 - Окно настроек типа объекта отчета - Выражение Lua

В окне настроек типа объекта **Выражение Lua** пишется текст на языке LUA, который автоматически вычисляется и выводит значение в отчет (см. рис. 3.162).

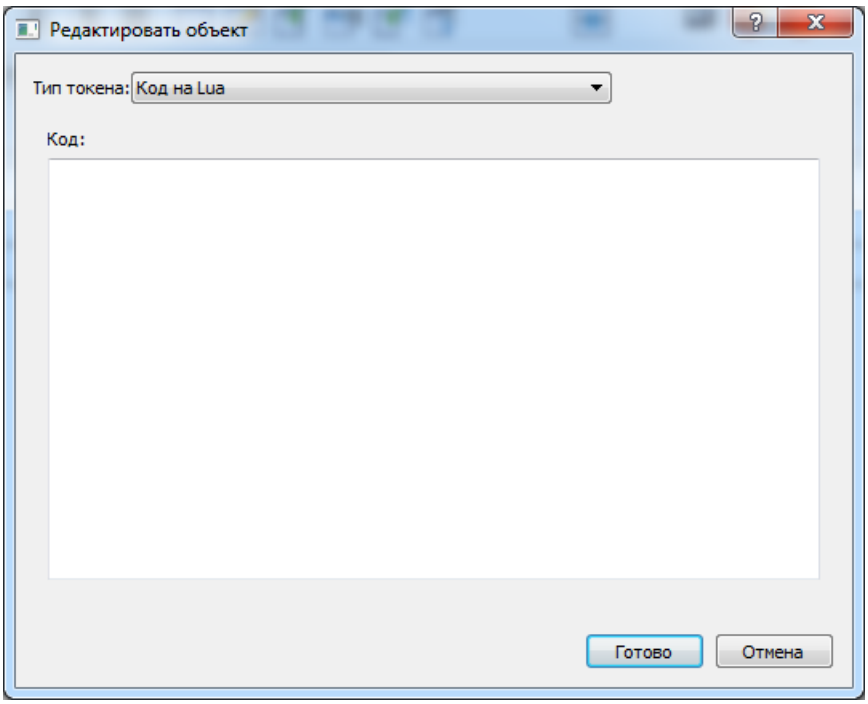


Рисунок 3.163 - Окно настроек типа объекта отчета - Код на Lua

В окне настроек типа объекта **Код на Lua** пишется программа на языке LUA, выполняющая действия в зависимости от её текста (см. рис. 3.163).

После настроек в редакторе необходимо нажать кнопку ("Сохранить") и закрыть окно редактора. Приложение **REPORT ENGINE** настроено и готово к работе. Для использования данного приложения в проекте, его необходимо добавить на нужный вам узел проекта (см. раздел 3.2.2.5.1).

3.3.6.2. Описание приложения SOUND (управления звуковыми сигналами системы)

Данная программа предназначена для добавления в проект звуковых файлов при пуско-наладочных работах.

3.3.6.2.1. Создание приложения SOUND

Создание приложения **SoundEditor** осуществляется аналогично другим приложениям SCADA-системы "Соната" (см. раздел 3.2.2.4).

3.3.6.2.2. Создание интерфейса приложения SOUND

На рис. 3.164 изображено окно интерфейса приложения SCADA-системы "Соната". Чтобы данное приложение стало приложением **SoundEditor**, необходимо в поле **Вид приложения** выбрать **SOUND**, а в поле **Имя типа** написать удобное для дальнейшего использования в проекте имя данного приложения (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

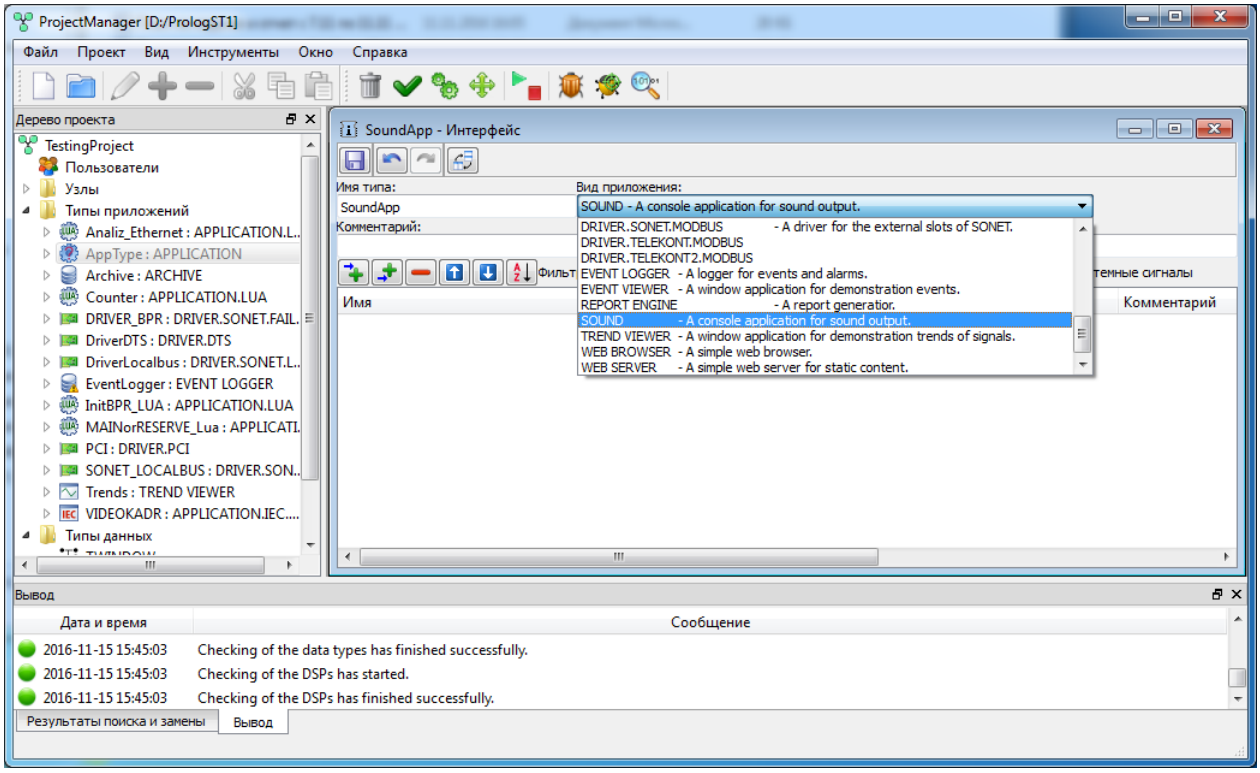


Рисунок 3.164 - Окно редактора интерфейса приложения

После выбора вида приложения необходимо добавить в интерфейс приложения сигналы, значения которых будут использоваться в приложении. Процесс добавления сигналов в интерфейс приложения подробно описан в раздел 3.2.2.4.3. На рис. 3.165 в интерфейс приложения **SoundEditor** добавлен сигнал **StartSound** - сигнал логического типа.

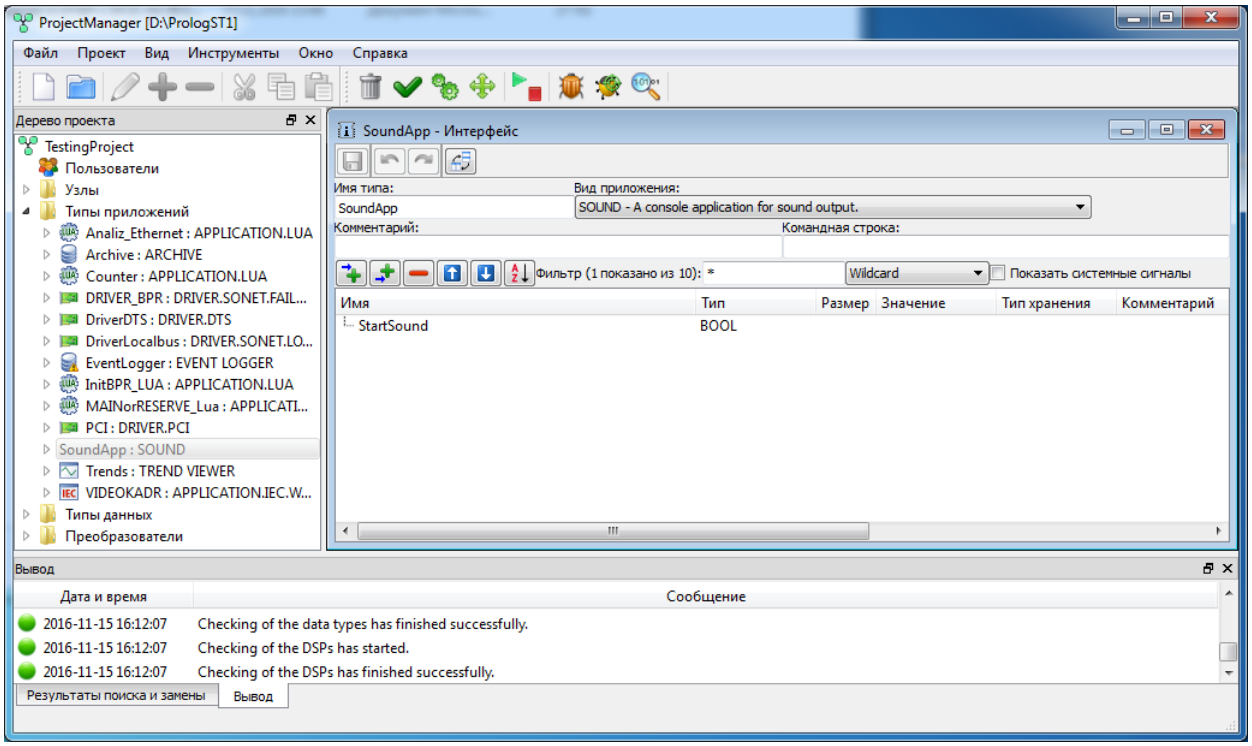


Рисунок 3.165 - Добавление сигнала в интерфейс приложения

После добавления сигналов в интерфейс приложения необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Интерфейс приложения настроен и теперь необходимо настроить связь сигналов из интерфейса приложения со звуковым файлом.

3.3.6.2.3. Настройка приложения SOUND

Для привязки сигналов из интерфейса приложения к звуковым файлам необходимо вызвать специализированный редактор приложения **SoundEditor**. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4). На рис. 3.166 изображено окно специализированного редактора (далее просто редактор) приложения **SoundEditor**.

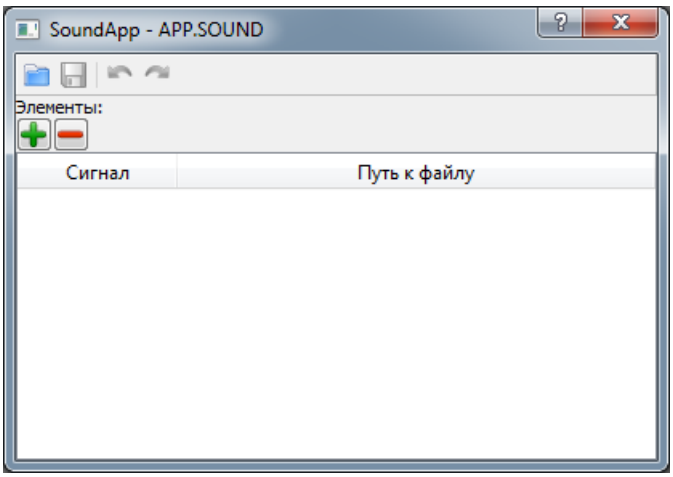


Рисунок 3.166 - Окно редактора приложения SoundEditor

В верхней части окна расположена панель инструментов со следующими кнопками:

("Открыть") - открыть файл с ранее сохраненными настройками;

("Сохранить") - сохранить внесённые в данный элемент проекта изменения;

("Отменить") - отменить предыдущее действие;

("Повторить") - повторить отменённое действие.

Ниже расположена таблица **Элементы** и кнопки управления:

(Добавить элемент) - добавить строку в таблицу **Элементы**;

(Удалить элемент) - удалить строку из таблицы **Элементы**;

- столбец **Сигнал** – имя сигнала SCADA-системы "Соната";

- столбец **путь к файлу** – путь к звуковому файлу.

На рис. 3.167 в столбце **Путь к файлу** указано имя звукового файла **alarm.wav**, а в столбце **Сигнал** указано имя сигнала из интерфейса приложения **StartSound**.

При выборе звукового файла он автоматически скопируется в директорию проекта, о чём будет выдано сообщение (см. рис. 3.168).

Сигнал **StartSound** может принимать значения TRUE и FALSE. По значению TRUE будет запускаться звуковой файл alarm.wav, по значению FALSE проигрывание файла будет остановлено.

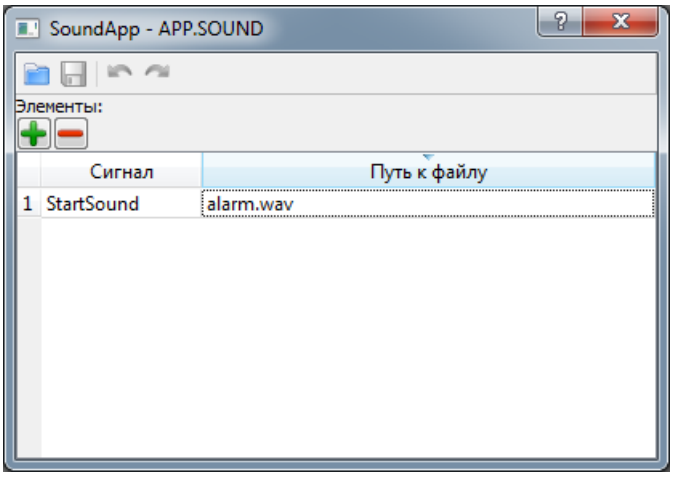


Рисунок 3.167 - Настроенное окно редактора приложения SoundEditor

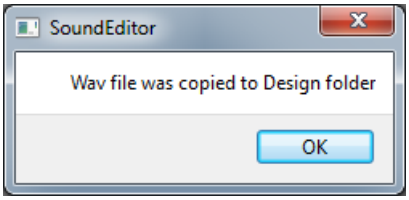


Рисунок 3.168 - Сообщение о копировании звукового файла в директорию проекта

Важно:Допускается использование только несжатых **wav** файлов. Частота дискретизации **wav** файла не должна превышать 44 100 Гц, количество каналов должно быть 1 или 2, количество бит на сэмпл 8 или 16.

3.3.7. Приложения для работа с WEB

3.3.7.1. Описание работы с приложением WEB SERVER

Приложение **WEB SERVER** позволяет принимать HTTP-запросы от клиентов, обычно веб-браузеров, и выдаёт им HTTP-ответы, которые представлены HTML-страницей с различными данными (изображения, медиа-потоки, значения сигналов SCADA-системы "Соната" и др.).

3.3.7.1.1. Создание приложения WEB SERVER

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению, осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.169 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **WEB SERVER**. Затем в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **Web_Server** (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

Также на рис. 3.169 в интерфейс приложения добавлен сигнал SCADA-системы "Соната" Global1 (о создании и добавлении сигналов см. раздел 3.2.2.4.3).

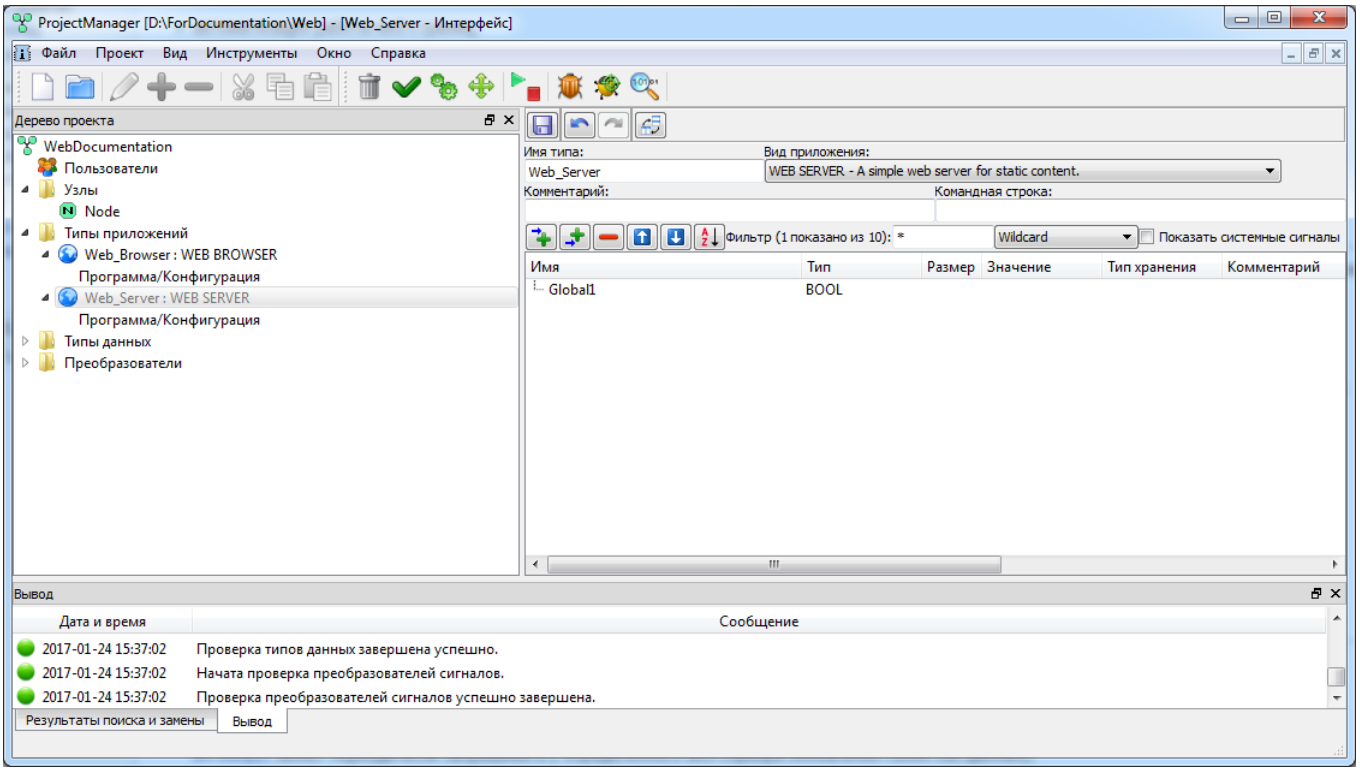


Рисунок 3.169 - Интерфейс приложения WEB SERVER

По окончании редактирования интерфейса типа приложения необходимо сохранить изменения с помощью кнопки  («Сохранить»), после чего рекомендуется закрыть диалог редактирования крестиком  в верхнем правом углу окна.

Приложение **WEB SERVER** не требует дальнейшего конфигурирования в дереве проекта, но, если зайти в строку **Программа/Конфигурация**, то можно увидеть его настройки и изменить установленные значения по умолчанию (см. рис. 3.170).

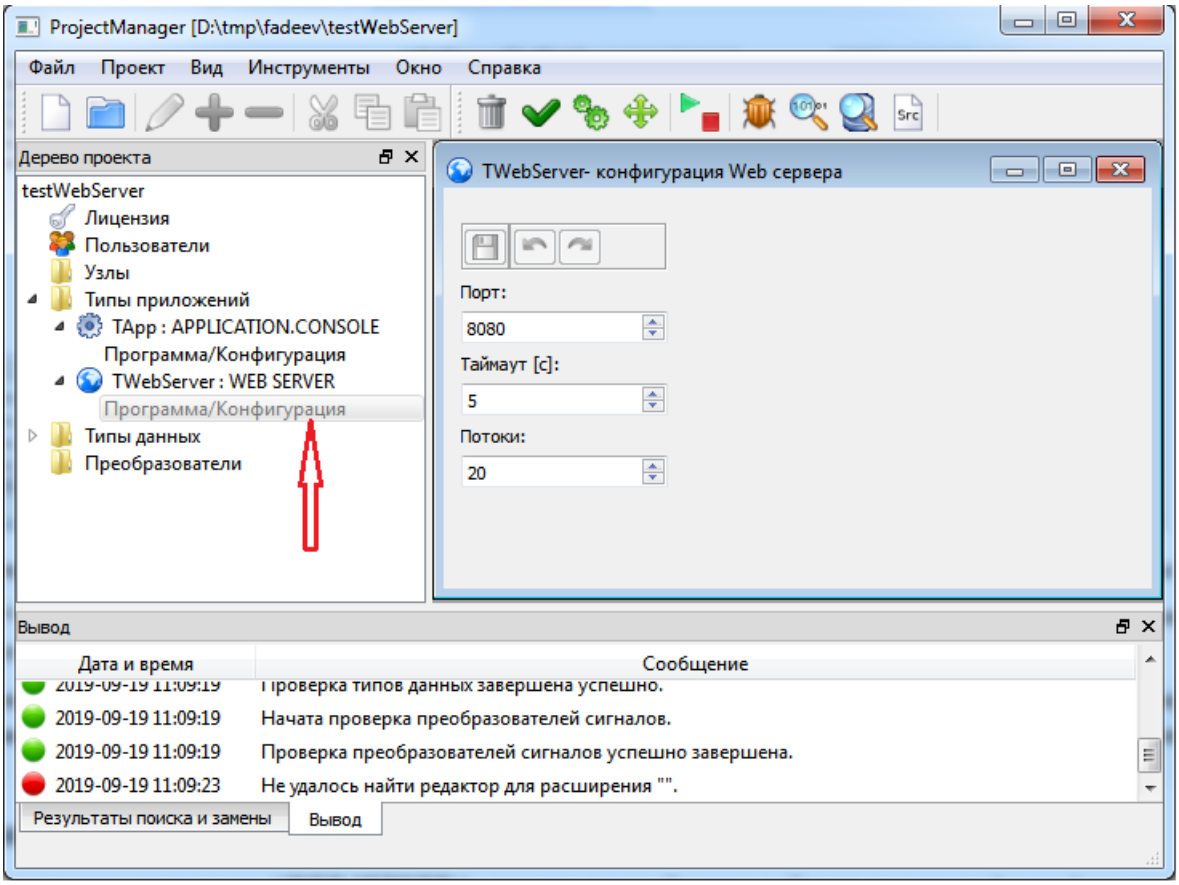


Рисунок 3.170 - Окно конфигурации приложения WEB SERVER

В окне конфигурации приложения **WEB SERVER** располагаются следующие кнопки и поля для ввода настроек:

-  (Сохранить) - кнопка сохранения внесенных изменений;
-  (Отменить) - кнопка отмены последних внесенных изменений;
-  (Повторить) - кнопка возврата отмененных изменений;
- **Порт** - порт, на котором будет работать **WEB SERVER** (по умолчанию 8080);
- **Таймаут** [с] - таймаут ожидания запроса от клиента;
- **Потоки** - максимальное количество подключений к **WEB SERVER**.

3.3.7.1.2. Работа с приложением WEB SERVER

Для использования приложения **WEB SERVER** в проекте его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1). На данном узле в папке с проектом должен храниться файл **index.htm**. Данный файл создаётся отдельно от SCADA-системы "Соната" и является

HTML-страницей, написанной с использованием средств веб-программирования. Приложение **WEB SERVER** работает с использованием данного файла.

Команды приложения **WEB SERVER**:

- **http://ip-адрес:8080/** - данная команда выводит в веб-браузере содержимое страницы **index.htm** (**ip-адрес:8080** - ip-адрес узла, на котором находится файл, 8080 - прописанный для данного приложения порт);

- **http://ip-адрес:8080/$Core** - данная команда выводит в веб-браузер список всех сигналов данного приложения **WEB SERVER**;

- **http://ip-адрес:8080/$Core/Name** - данная команда выводит в веб-браузер значение сигнала **Name** SCADA-системы "Соната" из интерфейса данного приложения, где **Name** - имя сигнала;

- **http://ip-адрес:8080/$Core/Name=Value** - данная команда изменит значение сигнала с именем **Name** на значение **Value**.

Существует два эквивалетных формата URL для запроса или установки значения сигнала:

1. **http://IP-адрес:8080:/$Core/SignalName [:ArchiveTime] [= Value],** где **ArchiveTime** - запрашиваемое архивное время сигнала (микросекунды от 01.01.1970); **Value** - записываемое значение сигнала. Если **Value** не указывается, то производится считывание сигнала, иначе запись с последующим считыванием. В случае массива структур имя переменной задается полностью, без приведения к одномерному массиву, т.е. в виде Struct[i].var[j].

2. **http://IP-адрес:8080:/$Core/SignalId [[CellIndex]] [:ArchiveTime] [= Value],** где **SignalId** - Id сигнала; **CellIndex** - индекс ячейки сигнала, в случае массивов структур каждое поле структуры разварачивается в отдельный одномерный массив (Struct[10].var[17] -> Struct.var[170], т.е. массив из 10-ти элементов Struct с массивом из 17-ти элементов в поле Var разварачивается в одномерный массив из 170 элементов); **ArchiveTime** - запрашиваемое архивное время сигнала (микросекунды от 01.01.1970); **Value** - записываемое значение сигнала. Если **Value** не указывается, то производится считывание сигнала, иначе запись с последующим считыванием.

3. **http://IP-адрес:8080:/$Core?SignalName=....[&ArchiveTime=]...[&Value=]**, где **SignalName** - имя сигнала, для массивов указывается ячейка, в случае массива структур имя переменной задается полностью, без приведения к одномерному массиву, т.е. в виде Struct[i].var[j]; **ArchiveTime** - запрашиваемое архивное время сигнала (микросекунды от 01.01.1970); **Value** - записываемое значение сигнала. Если **Value** не указывается, то производится считывание сигнала, иначе запись с последующим считыванием.

4. **http://IP-адрес:8080:/$Core?SignalId=....[&CellIndex=]...[&ArchiveTime=]...[&Value=]**, где **SignalId** - Id сигнала; **CellIndex** - индекс ячейки сигнала, в случае массивов структур каждое поле структуры разварачивается в отдельный одномерный массив (Struct[10].var[17] -> Struct.var[170], т.е. массив из 10-ти элементов Struct с массивом из 17-ти элементов в поле Var разварачивается в одномерный массив из 170 элементов); **ArchiveTime** - запрашиваемое архивное время сигнала (микросекунды от 01.01.1970); **Value** - записываемое значение сигнала. Если **Value** не указывается, то производится считывание сигнала, иначе запись с последующим считыванием.

Пример 1. Считать значение второй ячейки сигнала Signal_1 или Struct_1[0].Var[1] (Id =2).

http://IP-адрес:8080/$Core/Signal_1[2]

http://IP-адрес:8080/$Core/Struct_1[0].Var[1]

http://IP-адрес:8080/$Core/2[1]

или

http://IP-адрес:8080/$Core?SignalName=Signal_1[2]

http://IP-адрес:8080:/$Core?SignalName=Struct_1[0].Var[1]

http://IP-адрес:8080/$Core?SignalId=2&CellIndex=1

Пример 2. Записать во вторую ячеку сигнала Signal_1 или Struct_1[0].Var[1] (Id =2) число 3.

http://IP-адрес:8080/$Core/Signal_1[2]=3

http://IP-адрес:8080/$Core/Struct_1[0].Var[1]=3

http://IP-адрес:8080/$Core/2[1]=3

или

http://IP-адрес:8080/$Core?SignalName=Signal_1[2]&Value=3

http://IP-адрес:8080/$Core?SignalName=Struct_1[0].Var[1]&Value=3

http://IP-адрес:8080/$Core?SignalId=2&CellIndex=1&Value=3

Пример 3. Запросить архивное значение второй ячейки сигнала Signal_1 или Struct_1[0].Var[1] (Id =2) от 10.07.2019 12:19.

http://IP-адрес:8080/$Core/Signal_1[2]:1562761179000000

http://IP-адрес:8080/$Core/Struct_1[0].Var[1]:1562761179000000

http://IP-адрес:8080/$Core/2[1]:1562761179000000

или

http://IP-адрес:8080/$Core?SignalName=Signal_1[2]&ArchiveTime=1562761179000000

http://IP-адрес:8080/$Core?SignalName=Struct_1[0].Var[1]&ArchiveTime=1562761179000000

http://IP-адрес:8080/$Core?SignalId=2&CellIndex=1&ArchiveTime=1562761179000000

3.3.7.1.3. Пример работы с приложением WEB SERVER

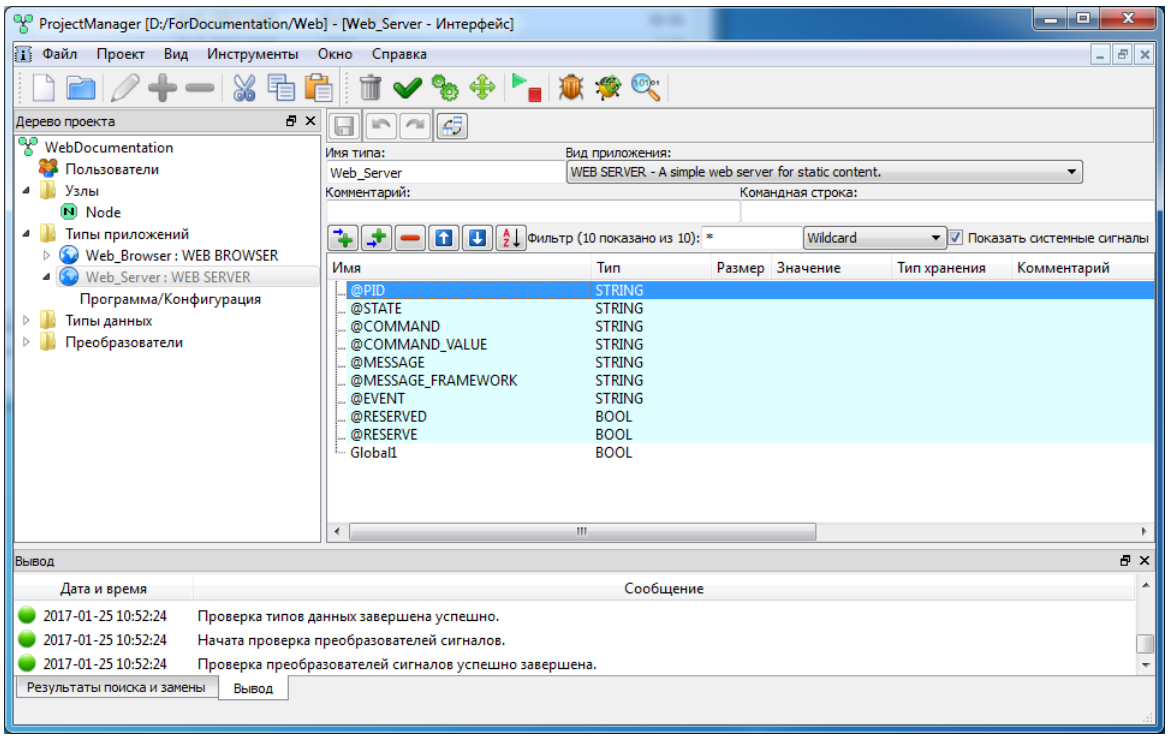


Рисунок 3.171 - Интерфейс приложения WEB SERVER с системными сигналами приложения

На рис. 3.171 изображён интерфейс приложения **WEB SERVER**, в котором отображаются системные сигналы приложения, так как стоит галочка ☑ Показать системные сигналы и сигнал **Global1** - глобальный сигнал данного проекта.

В примере будет описано, как получить значения всех сигналов интерфейса через интернет и отобразить в браузере.

В первую очередь необходимо создать файл **index.htm**. Данный файл пока создаётся без использования средств разработки SCADA-системы "Соната". Файл из данного примера представлен на рис. 3.172. Файл необходимо положить в папку Design вашего проекта.

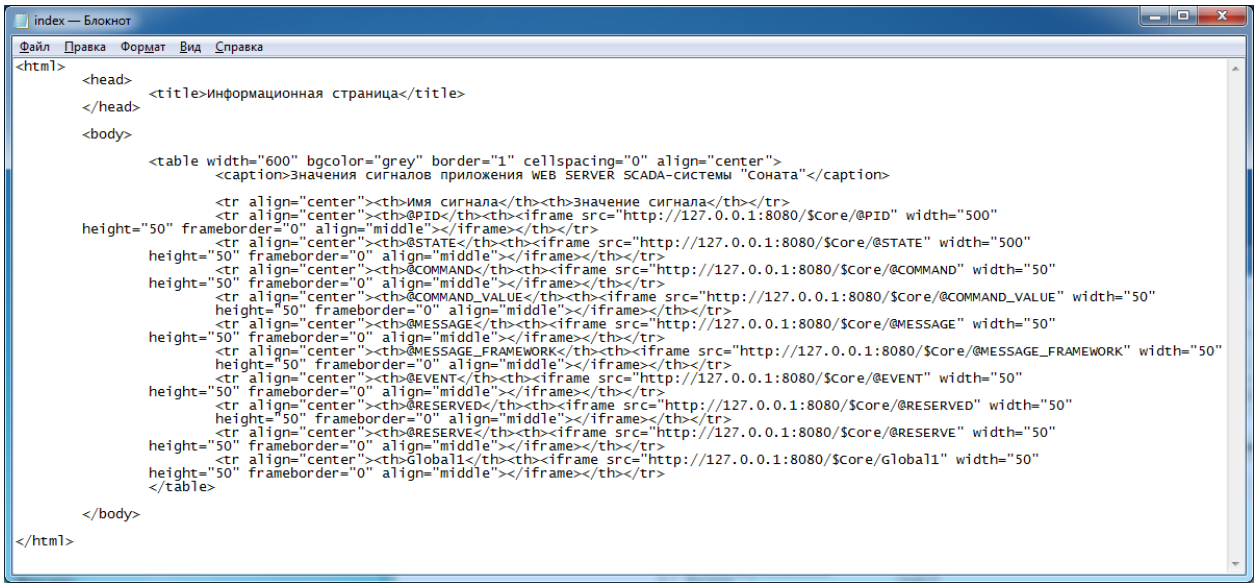
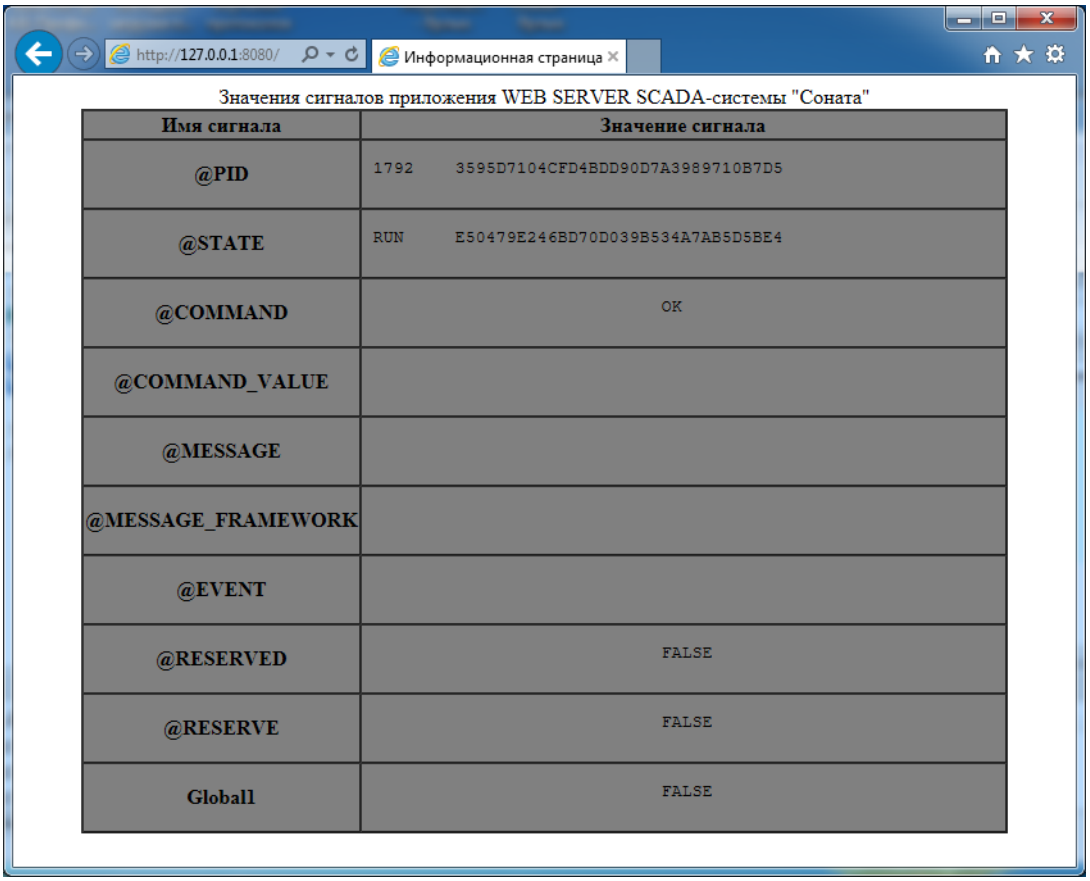


Рисунок 3.172 - Текст файла index.htm

Файл **index.htm** написан с использованием языка программирования HTML, но можно использовать любые средства веб-программирования. Значения сигналов выводятся с помощью html тега **iframe** - фрейм, который позволяет загружать данные извне и команды приложения **WEB SERVER** http://127.0.0.1:8080/$Core/Name. В данной команде указывается ip-адрес сервера (адрес 127.0.0.1, так как данный проект работает локально на компьютере), стандартный порт 8080 и имя сигнала (в данном случае Name - @PID, @STATE и др.).

В соответствии с принципами работы в SCADA-системе "Соната" далее необходимо создать узел, добавить на него нужные приложения, скомпилировать проект, разослать проект и запустить на выполнение (см. раздел 3.2).

Когда проект запущен, открываем веб-браузер и вводим в адресную строку команду http://127.0.0.1:8080/ - данная команда отображает обработанный результат файла **index.htm** (см. рис. 3.173).



Значения сигналов приложения WEB SERVER SCADA-системы "Соната"

Имя сигнала	Значение сигнала
@PID	1792 3595D7104CFD4BDD90D7A3989710B7D5
@STATE	RUN E50479E246BD70D039B534A7AB5D5BE4
@COMMAND	OK
@COMMAND_VALUE	
@MESSAGE	
@MESSAGE_FRAMEWORK	
@EVENT	
@RESERVED	FALSE
@RESERVE	FALSE
Globall	FALSE

Рисунок 3.173 - Интернет страница, полученная с нашего веб-сервера

Если ввести в адресную строку веб браузера команду http://127.0.0.1:8080/$Core, то будут отображены все сигналы данного приложения **WEB SERVER** (см. рис. 3.174).

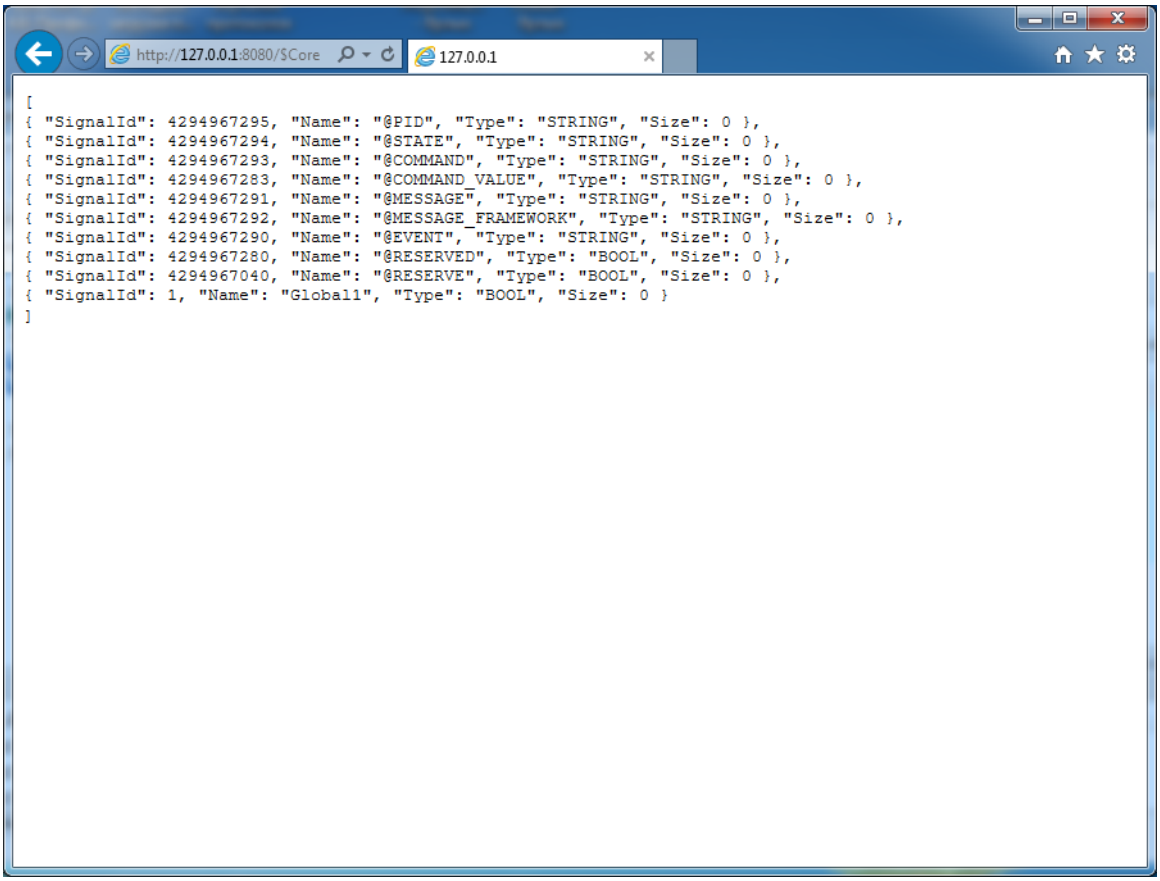


Рисунок 3.174 - Интернет страница, отображающая
все сигналы из интерфейса приложения WEB SERVER

Если ввести в адресную строку веб-браузера команду http://127.0.0.1:8080/$Core/Global1, то будет отображена страница со значением сигнала **Global1** из интерфейса приложения **WEB SERVER** (см. рис. 3.175).

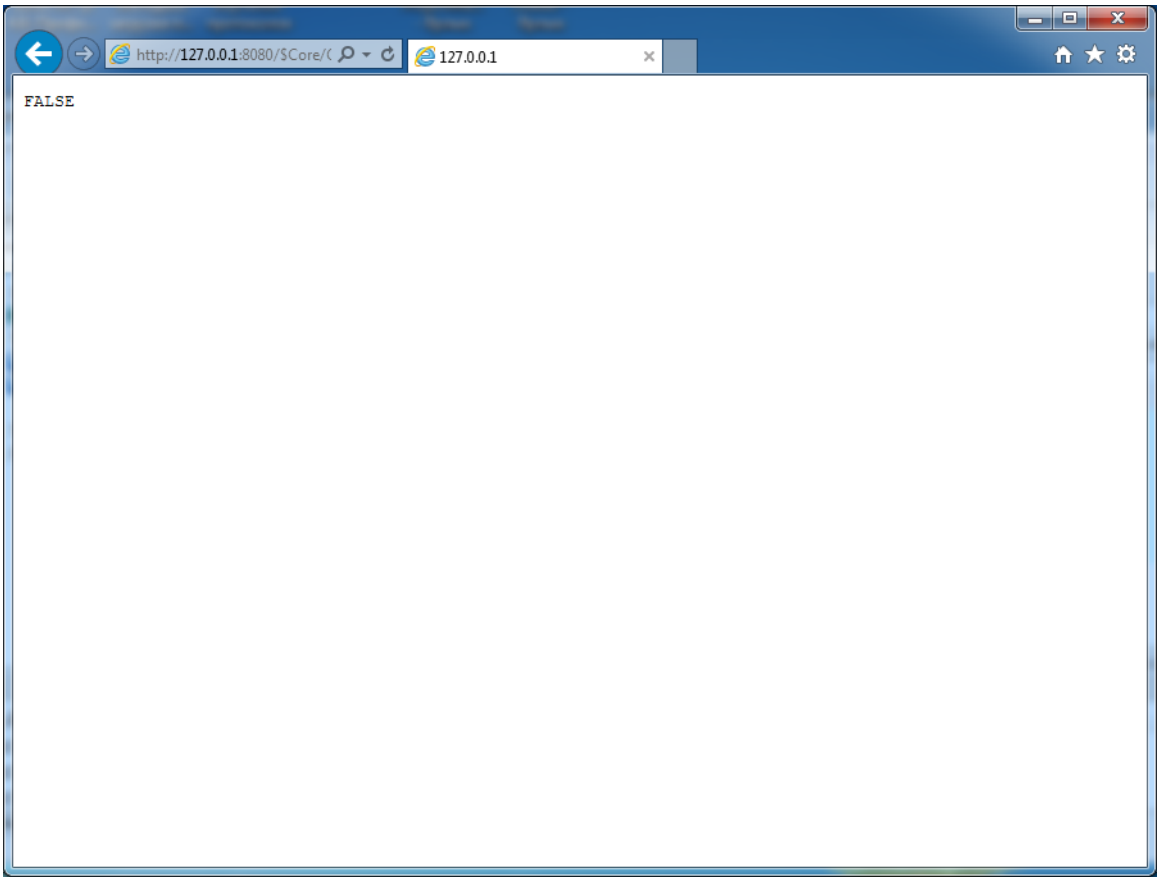


Рисунок 3.175 - Интернет страница, отображающая
значение сигнала из интерфейса приложения WEB SERVER

3.3.7.2. Описание работы с приложением WebBrowser

3.3.8. Приложения-драйвера

Различные виды приложений-драйверов служат для преобразования физических сигналов с модулей ввода/вывода в сигналы проекта SCADA-системы "СОНАТА". При редактировании приложений-драйверов вначале создается интерфейс приложения, содержащий список сигналов проекта, с которыми работает приложение-драйвер, а затем в специализированном редакторе производится привязка сигналов проекта к физическим сигналам (каналам модулей ввода/вывода).

3.3.8.1. Описание работы с приложением-драйвером DTS-клиент (шлюз с СВУ "ПОРТАЛ")

Приложение-драйвер DTS используется в SCADA-системе "СОНАТА" для реализации обмена данными со SCADA-системой "ПОРТАЛ". SCADA-система "ПОРТАЛ" рассматривается, как DTS-сервер, а SCADA-система "СОНАТА", как DTS-клиент. На данный момент приложение-драйвер DTS работает только в ОС Linux.

Создание приложения-драйвера DTS осуществляется аналогично другим приложениям-драйверам SCADA-системы "СОНАТА" (см. раздел 3.2.2.4).

Далее описан упрощенный порядок действий при создании приложения-драйвера DTS-клиент и подробно описаны важные моменты настройки данного приложения-драйвера:

1. Необходимо создать новое приложение в программе **ProjectManager** (см. раздел 3.2.2.4) и провести редактирование его имени типа и вида приложения (см. рис. 3.176).

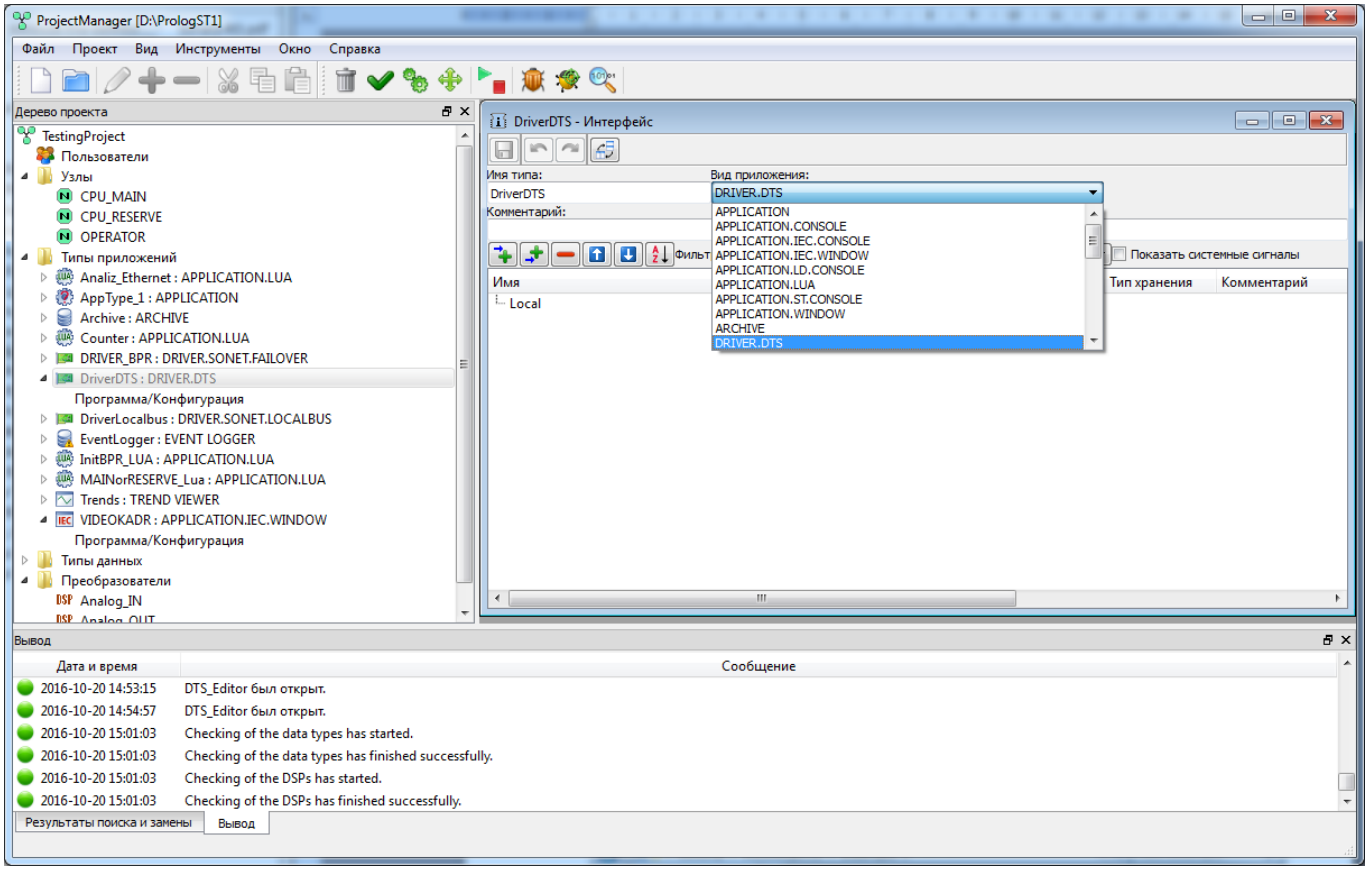


Рисунок 3.176 - Окно редактирования интерфейса приложения

2. Необходимо добавить сигналы SCADA-системы "СОНАТА", которые будут связаны с сигналами SCADA-системы "ПОРТАЛ". В интерфейс можно либо добавить глобальные сигналы, созданные ранее (см. раздел 3.2.2.8.1), либо создать локальные сигналы приложения (см. раздел 3.2.2.4.3). На рис. 3.177 в интерфейс приложения-драйвера DTS добавлены два сигнала SCADA-системы "СОНАТА": **LocalVar** - локальный сигнал приложения и **GlobalVar** - глобальный сигнал SCADA-системы "СОНАТА". По окончанию настройки интерфейса, необходимо нажать кнопку  ("Сохранить") и закрыть окно кнопкой  ("Закрыть").

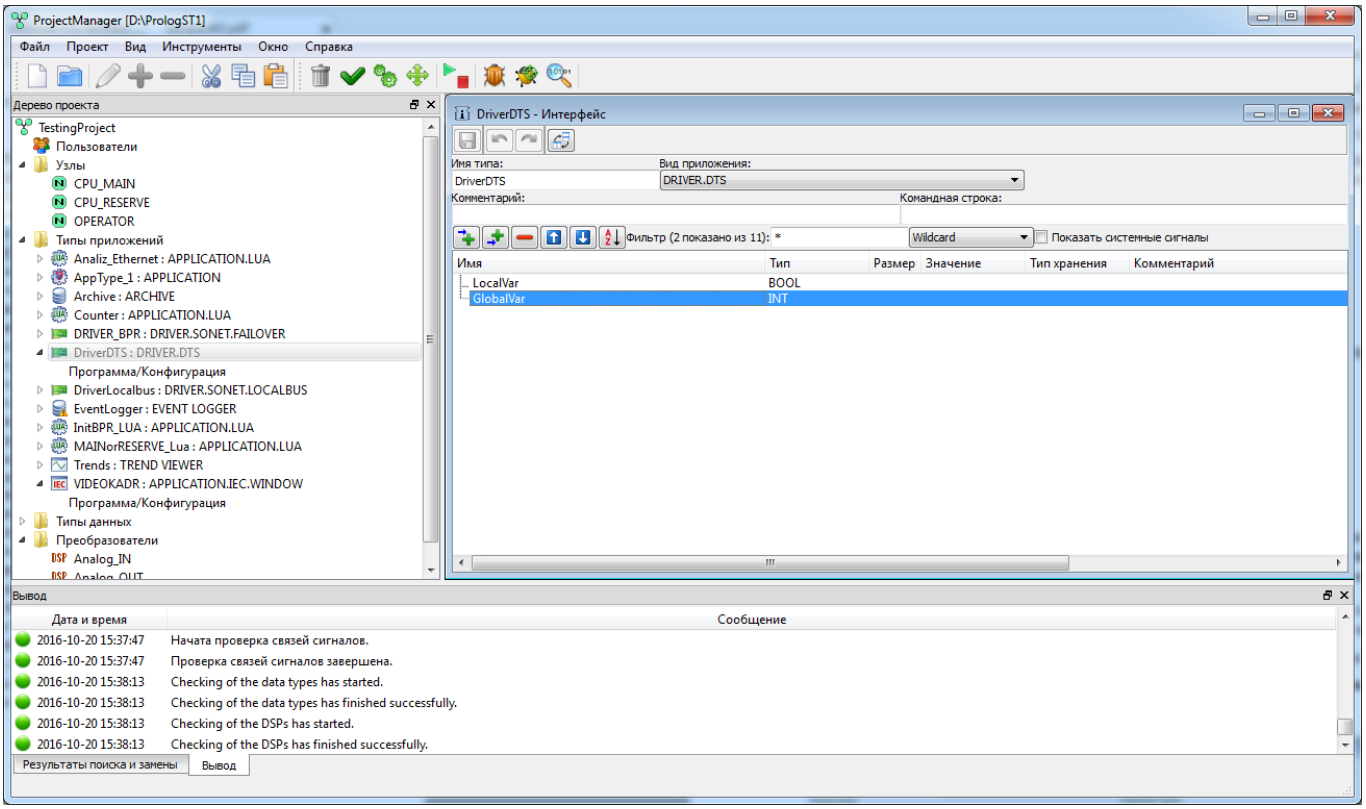


Рисунок 3.177 - Настроенный интерфейс приложения-драйвера DTS-клиент

3. Необходимо провести настройку приложения-драйвера DTS-клиент. Для этого нужно открыть редактор приложения (см. раздел 3.2.2.4.4). На рис. 3.178 изображено окно редактора приложения-драйвера DTS-клиент.

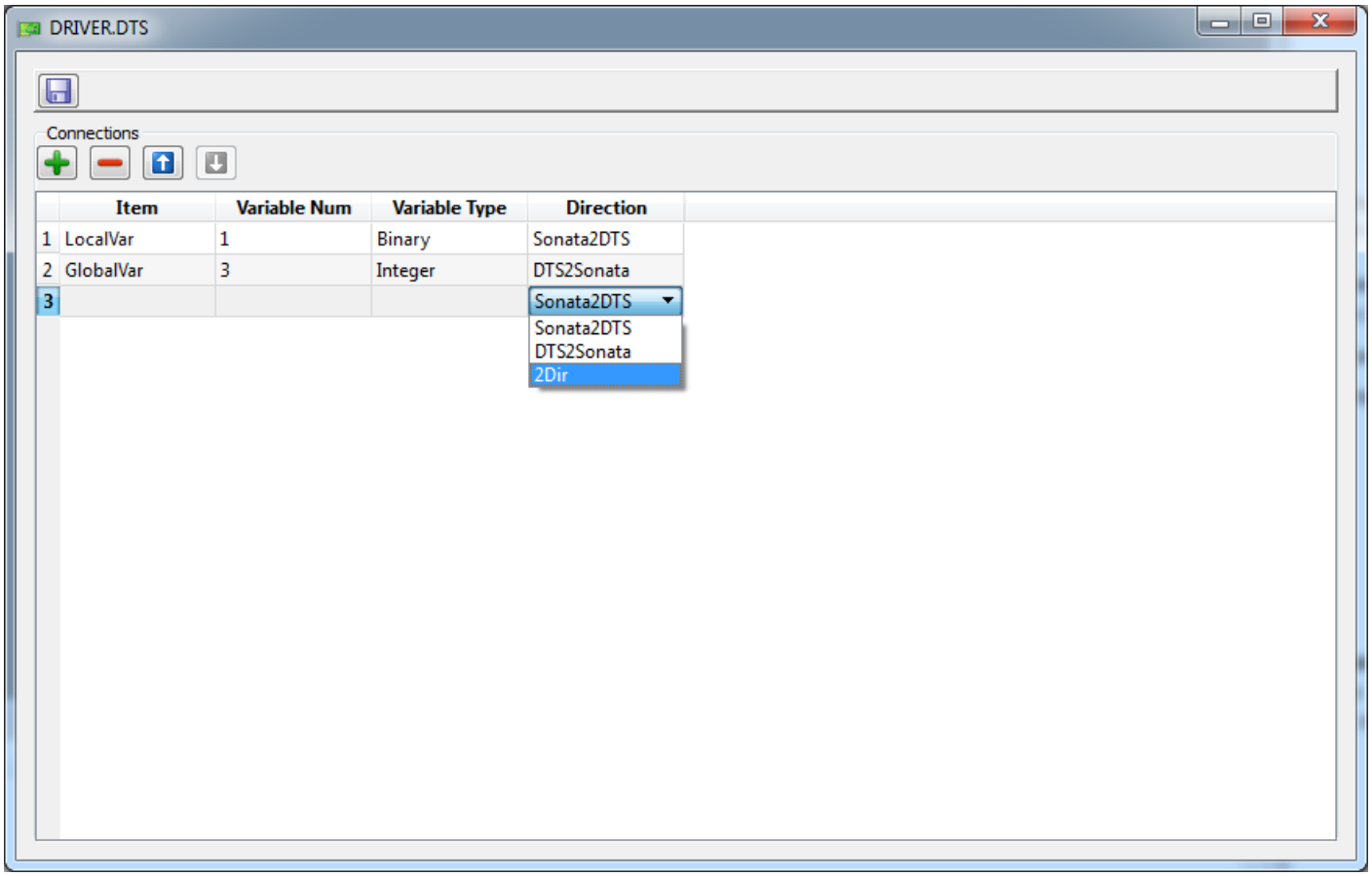


Рисунок 3.178 - Окно настроек кросс-коннекта приложения-драйвера DTS-клиент

В верхней части окна слева располагается кнопка  ("Сохранить") - сохраняются все настройки, сделанные в редакторе.

Ниже располагаются кнопки управления таблицей **Connections** ("Связи"):

 ("Добавить") - добавляет строку в таблицу **Connections** ("Связи");

 ("Удалить") - удаляет строку из таблицы **Connections** ("Связи");

 ("Передвинуть вверх") - перемещает вверх строку в таблице **Connections** ("Связи");

 ("Передвинуть вниз") - перемещает вниз строку в таблице **Connections** ("Связи").

Каждая строка в таблице **Connections** ("Связи") состоит из следующих столбцов:

- **Item** - в данном столбце находятся сигналы SCADA-системы "СОНАТА", выбор сигнала осуществляется двойным щелчком «мыши» по ячейке данного столбца (откроется диалог выбора сигналов интерфейса приложения см. рис. 3.179);
- **Variable Num** - номер сигнала в SCADA-системе "ПОРТАЛ";
- **Variable Type** - тип сигнала в SCADA-системе "ПОРТАЛ", есть три варианта:
  - **Analog** - значения с плавающей точкой;
  - **Binary** - 0 или 1;
  - **Integer** - целочисленные значения;
- **Direction** - направление взаимодействия между SCADA-системой "СОНАТА" и SCADA-системой "ПОРТАЛ", есть три варианта:
  - **Sonata2DTS** - однонаправленная передача данных из SCADA-системы "СОНАТА" во вне (в SCADA-систему "ПОРТАЛ");
  - **DTS2Sonata** - однонаправленная передача извне (из SCADA-системы "ПОРТАЛ") в SCADA-систему "СОНАТА";

- **2Dir** - двухсторонний обмен.

Важно

Тип сигнала SCADA-системы "СОНАТА", добавленного в поле **Item**, должен соответствовать типу сигнала SCADA-системы "ПОРТАЛ" в поле **Variable Type**.

Основная задача приложения-драйвера DTS-клиент заключается в настройке данных полей таблицы связей, когда все поля настроены, необходимо сохранить настройки кнопкой и закрыть окно кнопкой . Драйвер готов к работе. Для использования данного приложения в проекте его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

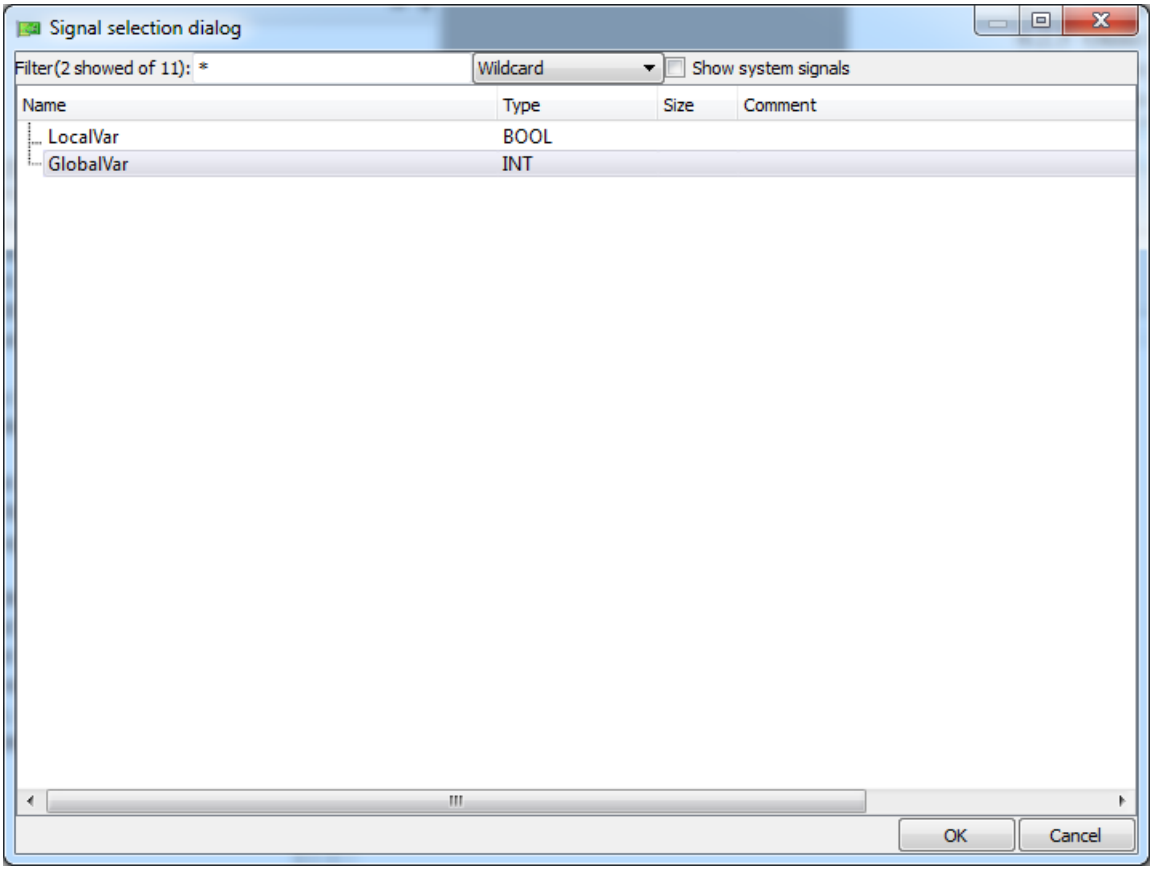


Рисунок 3.179 - Диалог выбора переменных из интерфейса приложения

Подробное описание интерфейса рис. 3.179 не приводится, так как он аналогичен описанному в раздел 3.2.2.4.3.2.

3.3.8.2. Описание работы с приложением-драйвером MODBUS

Приложение-драйвер Modbus может быть использован для любых устройств, поддерживающих протокол Modbus.

Основные характеристики:

- Драйвер поддерживает работу, как через TCP, так и через RS-485;
- Драйвер может работать в режиме ведущего (master) и ведомого (slave);
- В режиме RS-485 поддерживаются адреса устройств от 1 до 255 на шине;

- Поддерживаются следующие MODBUS функции: READ_COIL_STATUS=1, READ_DISCRETE_INPUTS=2, READ_HOLDING_REGISTERS=3, READ_INPUT_REGISTERS=4, FORCE_SINGLE_COIL=5, RESET_SINGLE_REGISTER=6, FORCE_MULTIPLY_COILS=15, PRESET_MULTIPLY_REGISTERS=16, READ_WRITE_MULTIPLY_REGISTERS=23;
- Поддерживаются следующие скорости передачи данных: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 Бод;
- Поддерживаются следующие контроли чётности: четный, нечетный, без контроля;
- Количество бит данных ModBus RTU: 8 бит;
- Количество стоповых битов ModBus RTU: 1 или 2 бита;
- Допустимый период ожидания ответа на запрос до 5000 мс;
- Поддерживаемые форматы цифровых данных: BIT, BYTE, USHORT_AB, USHORT_BA, SHORT_AB, SHORT_BA, ULONG_ABCD, ULONG_CDAB, ULONG_BADC, ULONG_DCBA, LONG_ABCD, LONG_CDAB, LONG_BADC, LONG_DCBA, FLOAT_ABCD, FLOAT_CDAB, FLOAT_BADC, FLOAT_DCBA, DOUBLE_ABCDEFGH, DOUBLE_GHEFCDAB, DOUBLE_BADCFEHG, DOUBLE_HGFEDCBA.

3.3.8.2.1. Создание приложения-драйвера MODBUS

Создание приложения-драйвера Modbus осуществляется аналогично другим приложениям-драйверам SCADA-системы "Соната" (см. раздел 3.2.2.4).

На рис. 3.180 изображено стандартное окно интерфейса приложения SCADA-системы "Соната". Чтобы данное приложение стало приложением-драйвером Driver.Modbus, необходимо в поле **Вид приложения** выбрать **DRIVER.MODBUS** и в поле **Имя типа** написать удобное для дальнейшего использования в проекте имя данного приложения (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем имя типа не может начинаться с цифры).

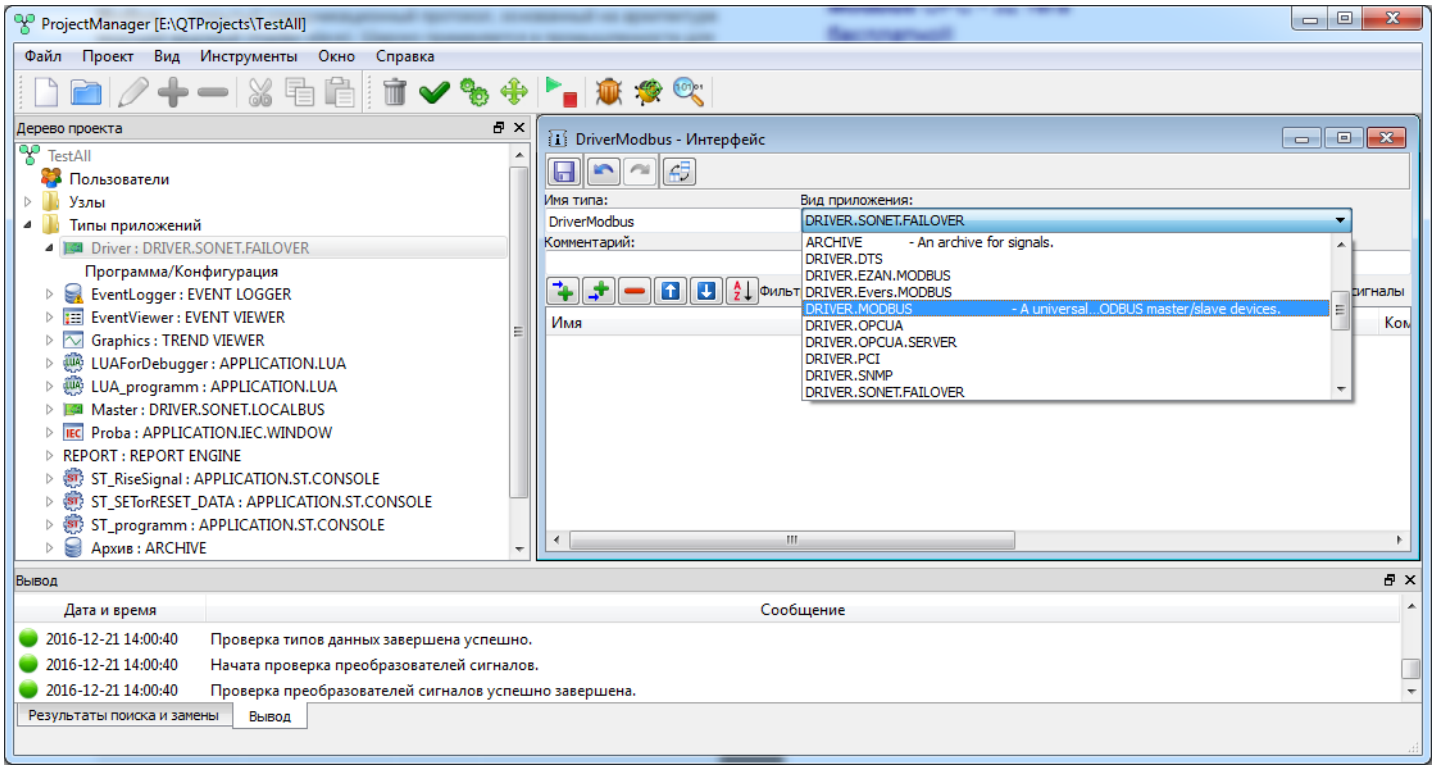


Рисунок 3.180 - Изменение приложения на приложение-драйвер Driver.Modbus

3.3.8.2.2. Создание интерфейса приложения-драйвера MODBUS

После выбора вида драйвера необходимо добавить в интерфейс приложения сигналы, значение которых будут связаны с информацией от Modbus устройств. Для этого нужно нажать одну из кнопок редактирования таблицы интерфейса:  («Вставить сверху») или  («Вставить снизу») – на экране появится окно выбора (см. рис. 3.181), в котором нужно выбрать вариант Локальный или Глобальный.

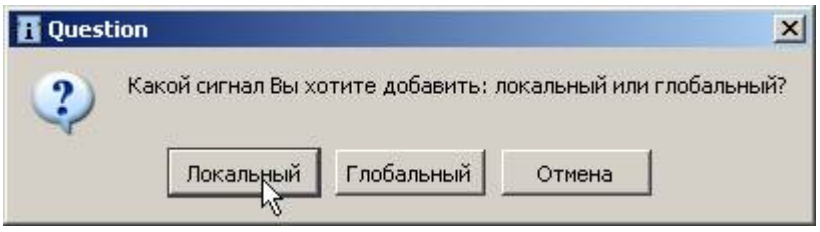


Рисунок 3.181 - Выбор признака Локальный/Глобальный при добавлении сигнала в интерфейс приложения

У **Локальных сигналов** можно редактировать поля **Имя** и другие. Так как имя новому сигналу задается по шаблону, то его можно изменить, вначале выделив строку сигнала, а затем дважды «кликнуть» «мышью» по графе Имя в выделенной строке – строка станет доступной для редактирования (см. рис. 3.182).

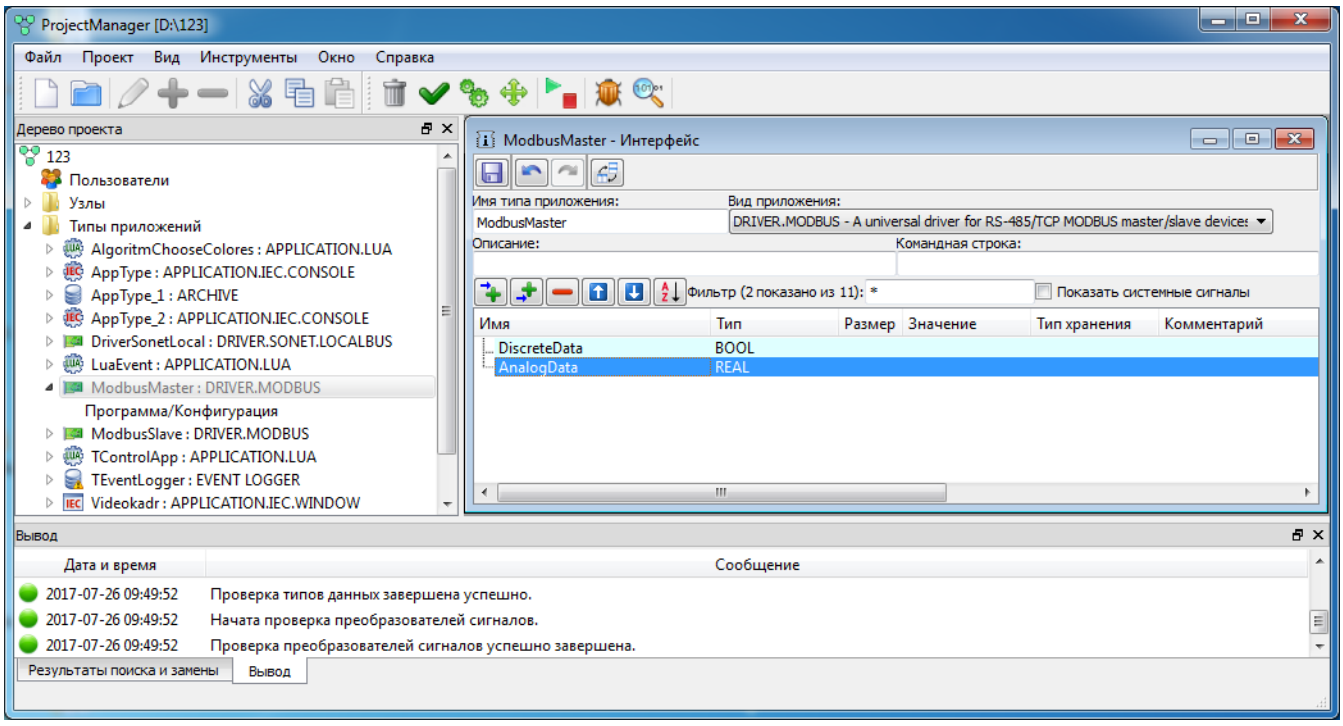


Рисунок 3.182 - Редактирование имени локального сигнала в интерфейсе приложения

Можно также поменять тип сигнала: для этого необходимо дважды «кликнуть» «мышью» по графе **Тип**, а затем кликнуть по кнопке – появится выпадающее меню, в котором можно изменить тип локального сигнала. Процесс добавления сигналов в интерфейс приложения подробно описан в раздел 3.2.2.4.3.

По завершении редактирования интерфейса типа приложения необходимо сохранить изменения кнопкой  в левой верхней части окна интерфейса приложения. Далее окно нужно

закрыть крестиком в верхнем правом углу окна редактирования. Если внесенные изменения не были сохранены, то на экране появится диалог (см. рис. 3.183), в котором нужно выбрать либо сохранение изменений и выход из редактора (кнопка Сохранить) , либо отказ от внесенных изменений и выход из редактора (кнопка Отклонить) , либо отказаться от выхода из процедуры редактирования (кнопка Отмена).

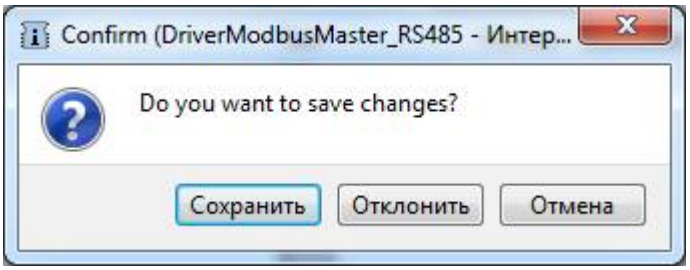


Рисунок 3.183 - Диалог подтверждения выхода из редактора интерфейса приложения

3.3.8.2.3. Настройка драйвера MODBUS и привязка сигналов интерфейса драйвера к каналам ввода/вывода

Для привязки сигналов из интерфейса приложения-драйвера к каналам модулей ввода/вывода необходимо вызвать специализированный редактор приложения-драйвера **Driver.Modbus**. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4). На рис. 3.286 изображено окно специализированного редактора (далее просто редактор) приложения-драйвера DRIVER.MODBUS.

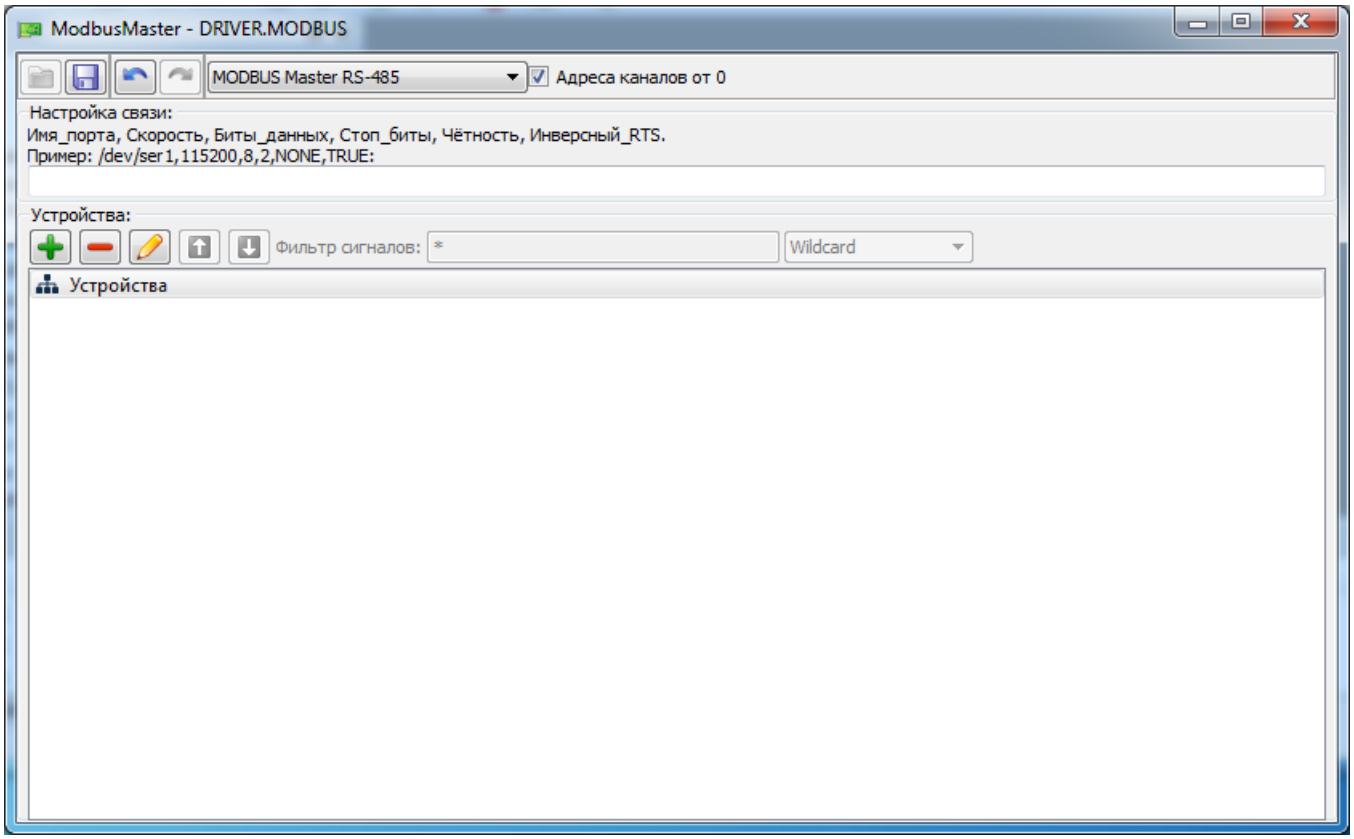


Рисунок 3.184 - Редактор приложения-драйвера Driver.Modbus

Окно редактора состоит из панели инструментов, поля **Настройка связи** и поля **Устройства**.

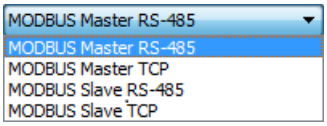
Панель инструментов расположена в верхней части окна – на этой панели расположены следующие значки (кнопки):

-  («Открыть файл») – команда для открытия уже сформированного файла, содержащего конфигурацию приложения-драйвера;

-  («Сохранить») – команда для сохранения конфигурации приложения-драйвера;

-  («Назад») – отменить предыдущее действие по редактированию;

-  («Вперед») – повторить ранее отменённое действие;

- 

- поле выбора разновидности Modbus протокола (RS485 или TCP) и Master(хозяин) или Slave(подчинённый);

- поле **Адреса каналов от 0** - с нуля или с 1 вести счет каналов ввода/вывода.

При выборе различных Modbus протоколов изменяется внешний вид редактора. Так при выборе Modbus протокола MODBUS Master RS-485 или MODBUS Slave RS-485 редактор принимает вид, как на рис. 3.185.

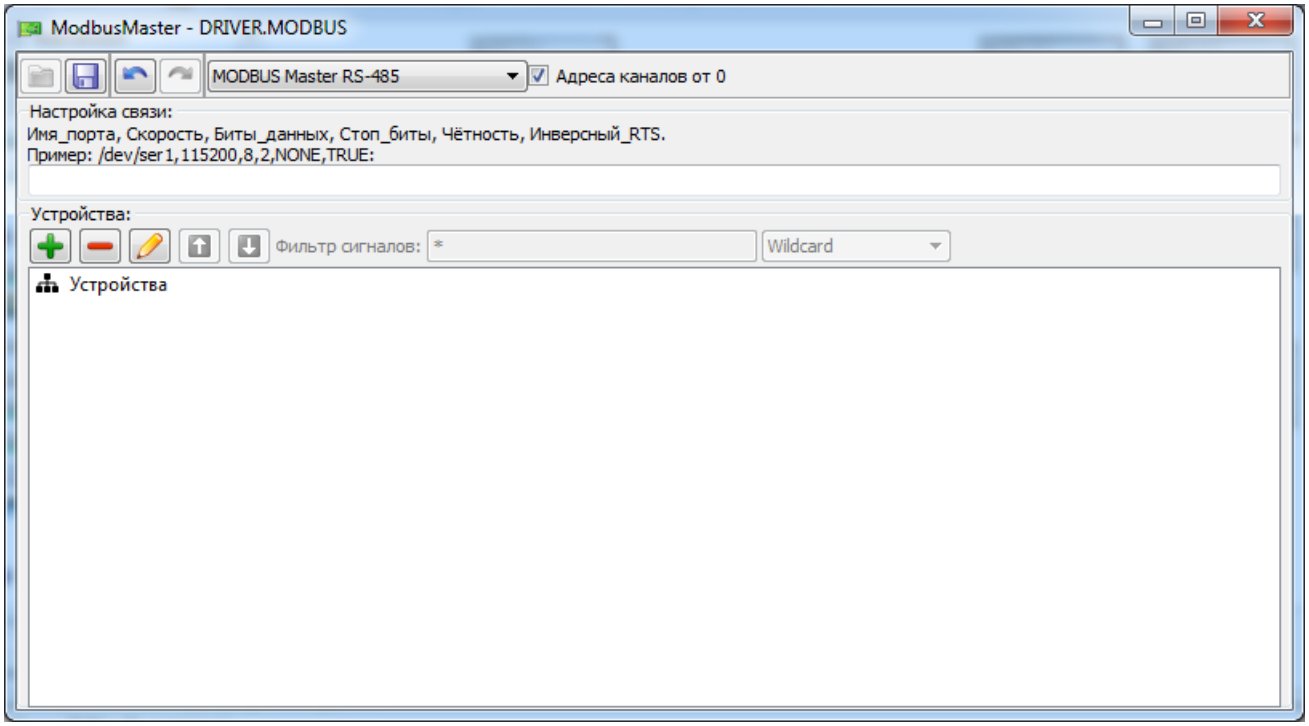


Рисунок 3.185 - Редактор Modbus протокола RS-485

Если же выбрать протокол MODBUS Master TCP или MODBUS Slave TCP, то редактор примет вид, как на рис. 3.186.

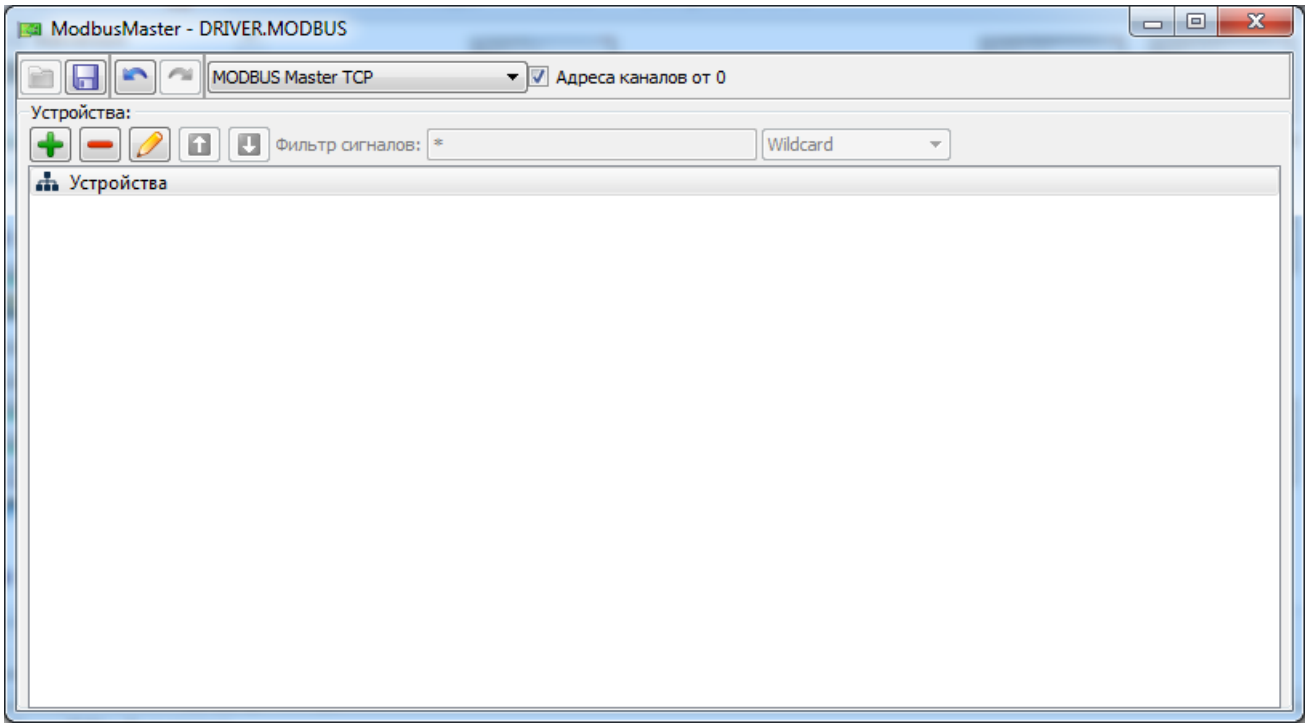
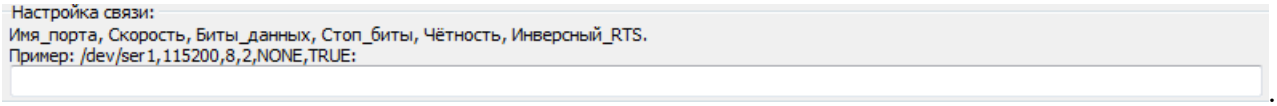


Рисунок 3.186 - Редактор Modbus протокола TCP

Различия данных редакторов в том, что для протокола Modbus RS-485 (Master и Slave) присутствует поле Настройка связи -



.

В поле **Настройка связи** нужно описать **Имя порта**, **Скорость связи**, **Количество бит данных**, **Количество стоп битов**, **Четность** и **Инверсный RTS** (за данными настройками обращайтесь к описанию протокола Modbus на устройстве).

ВАЖНО! Настройка **Инверсный RTS** была введена для работы с контроллером Сонет, которому требуется выставить значение данного параметра в TRUE. В основном для всех остальных устройств значение данного параметра FALSE.

В остальном редакторы для протоколов Modbus RS-485 и Modbus TCP идентичны, поэтому подробно рассмотрим редактор на примере протокола Modbus RS-485.

Имя порта соответствует порту, по которому вы организуете связь. В примере на рис. 3.187 с рабочей станции Windows по порту COM2 организована связь с модулями ввода/вывода. Зная значение скорости передачи, количества бит данных и другие настройки, которые настроены для Modbus со стороны контроллера с модулями ввода/вывода, мы указываем их такими же (см. рис. 3.187). Важно понимать, что на контроллере с модулями ввода/вывода должен также работать драйвер протокола Modbus RS-485, либо из драйверов СКАДА-системы Соната (по протоколу Modbus Slave RS-485), либо сторонний драйвер.

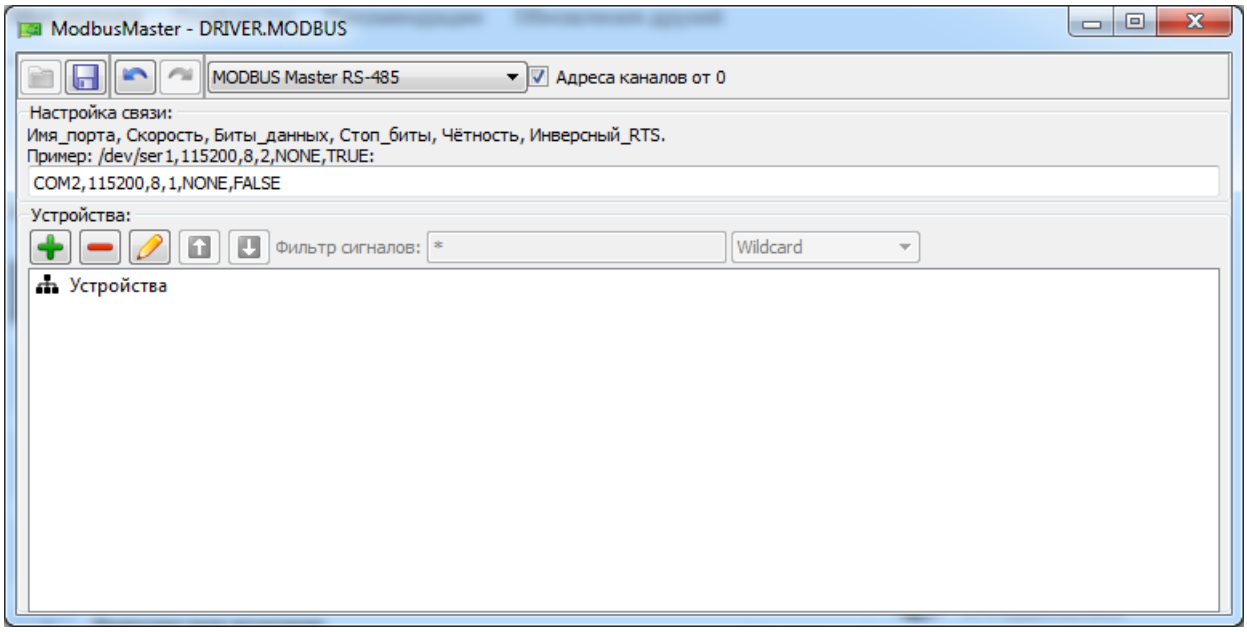


Рисунок 3.187 - Настройки связи в редакторе Modbus

Поле **Устройства** содержит методы работы с сопряженными устройствами:

 - добавление устройства или функции (чтение, запись и др.), как это делается, смотри далее;

 - удаление устройства или др.;

 - редактирование записей;

 - переместить по списку вверх;

 - переместить по списку вниз;

- **Фильтр сигналов -** произвести выборку сигналов по критериям.

** Если посмотреть на рис. 3.187, то видно, что в данном примере настроено устройство Modbus Master (рабочая станция Windows) по протоколу Modbus RS485.*

Для того чтобы добавить периферийное устройство, с которым необходимо взаимодействовать, необходимо выбрать поле устройства и нажать  (см. рис. 3.188).

Далее необходимо выделить курсором «мыши» добавленную строку устройства и нажать  для редактирования (см. рис. 3.189). Откроется окно редактирования свойств устройства, с которым налаживается взаимодействие (см. рис. 3.190).

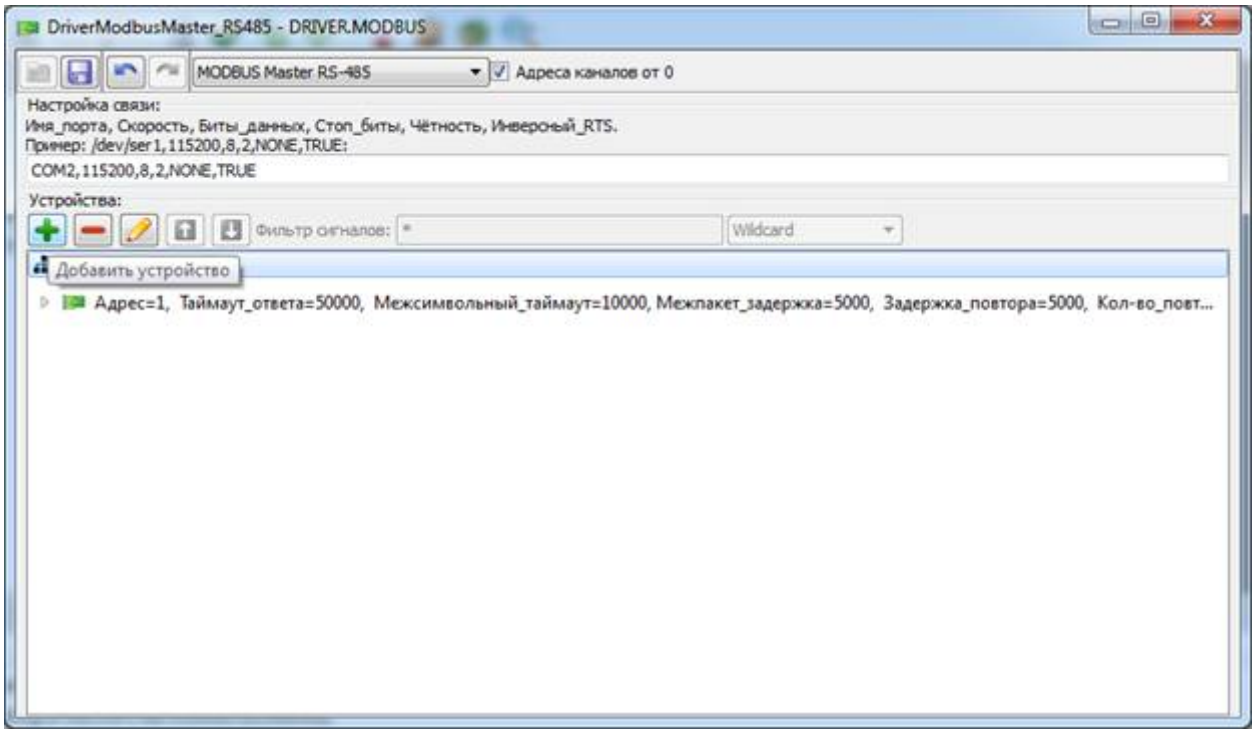


Рисунок 3.188 - Добавление устройства

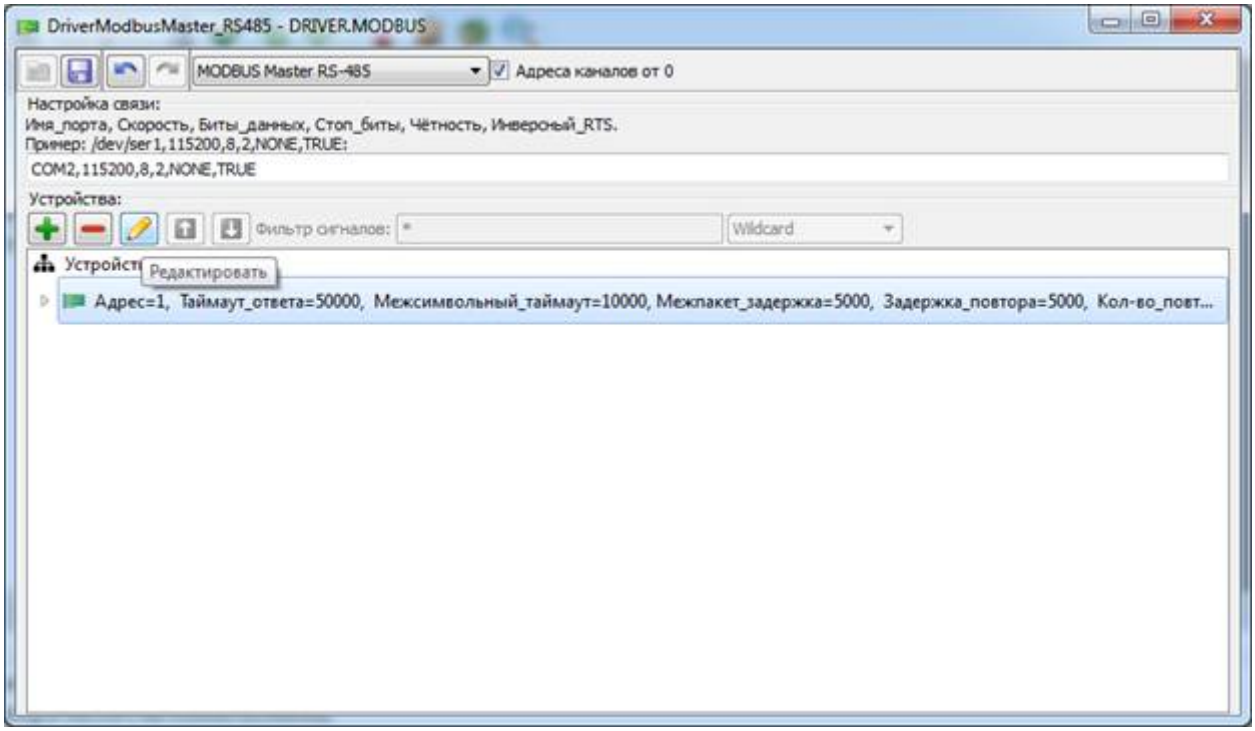


Рисунок 3.189 - Кнопка редактирования

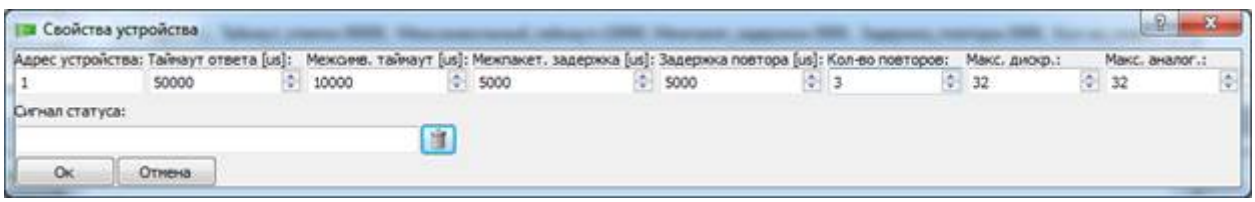


Рисунок 3.190 - Окно редактирования свойств устройства

Адрес устройства - это modbus адрес подчиненного(slave) устройства, который ему задан, в соответствии с проектной документацией. На рис. 3.190 настроен адрес устройства 1, это адрес

процессорного модуля взятый из документации на проект. Для драйвера MODBUS Master TCP данное поле может выглядеть как IP-адрес:порт, где IP-адрес - IP адрес подчиненного (slave) устройства, порт - TCP порт для Modbus (обычно 502).

Внимание! Протокол MODBUS TCP в режиме Slave работает по 502 порту. В один момент времени порт TCP может быт занят только одним приложением, то есть на одном узе не могут одновременно работать более одно драйвера MODBUS TCP в режиме Slave.

Далее описаны параметры, которые отвечают за передачу данных.

Таймаут ответа - время, после которого считается, что ответ не получен;

Межсимвольный таймаут - максимальное время между символами в пакете (по спецификации считается, что 3,5*Межсимвольный таймаут - это признак конца пакета);

Межпакетная задержка - минимальное время, которое выдержит наше устройство после отправки запроса, перед отправкой следующего запроса, т.е. запросы будут отправляться не чаще, чем данное время;

Задержка повтора - пауза, которую выдержит наше устройство, после того, как пройдет время ожидания ответа на посланный запрос (т.е. принято решение, что ответ не получен), перед повторной отправкой данного запроса;

Количество повторов - количество повторных запросов, после которых считается, что устройство, к которому мы обращались, неисправно;

Макс. дискретных - максимальное количество дискретных значений, которое можно посылать одним пакетом;

Макс. аналог - максимальное количество аналоговых значений, которое можно посылать одним пакетом.

Параметр для диагностики устройства в ваших алгоритмах:

Сигнал статуса - можно прикрепить сигнал (из ранее заведенных), который будет показывать исправно устройство или нет (см. Приложение В), данный параметр работает только для приложений-драйверов с настройкой Master.

ВАЖНО! При пропаже связи устройству и всем группам выставляется статус DEFECTIVE(2). При возобновлении связи при опросе одной из групп данной группе устанавливается признак OK(0), остальным группам устанавливается статус UNDEFINED(128). Контроллер в целом получает статус UNDEFINED(128). По мере опроса групп они меняют статус с UNDEFINED(128) на OK(0). Как только у контроллера исчезнут все группы со статусом UNDEFINED(128) он изменит свой статус на OK(0). Если в ходе работы драйвера какая-то группа будет читаться с ошибкой PARTIALLY_DEFECTIVE(1) или DEFECTIVE(2), но у устройства не будет групп со статусом UNDEFINED(128), то устройство получит статус PARTIALLY_DEFECTIVE(1).

После того, как настроены свойства устройства, нужно настроить операции (чтение, запись и др.) - функции, которые необходимы для взаимодействия с устройством. Для этого нужно выделить курсором мыши строку в таблице **Устройства** и нажать кнопку  (см. рис. 3.191). На описании устройства появится раскрывающееся меню и, если его раскрыть, то там будет строка для настройки функции взаимодействия (см. рис. 3.192).

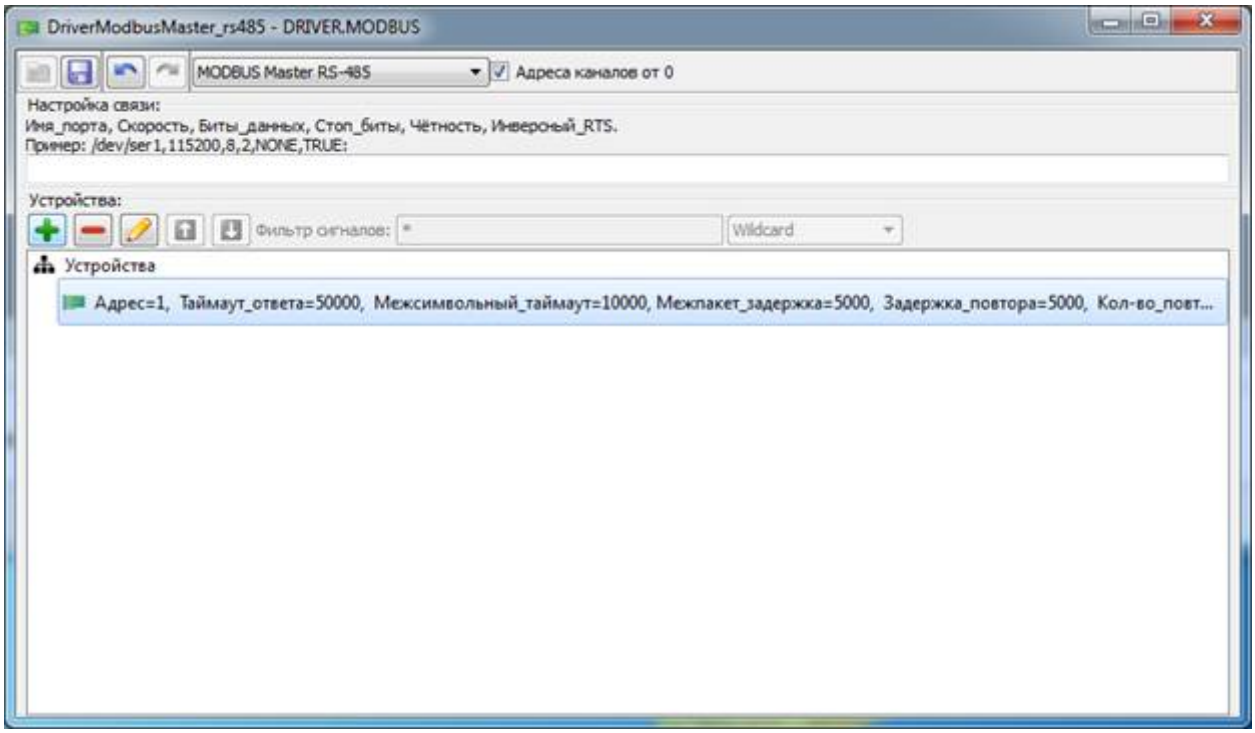


Рисунок 3.191 - Добавление функций взаимодействия с периферийным устройством

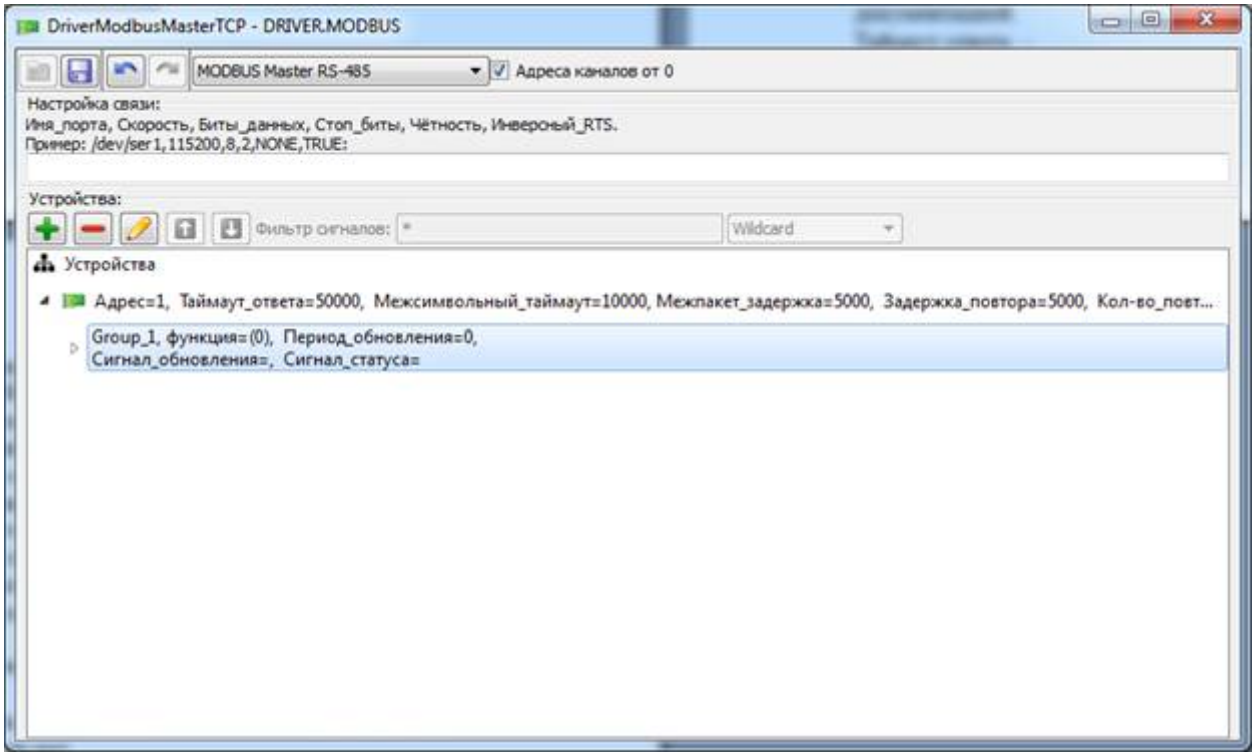


Рисунок 3.192 - Отображение строки функции взаимодействия с периферийным устройством

Теперь необходимо настроить данную функцию. Для этого нужно выделить данную строку и нажать  (см. рис. 3.193). Откроется окно **Свойства группы,** где и настраиваются необходимые операции (запись, чтение и др.) (см. рис. 3.194).

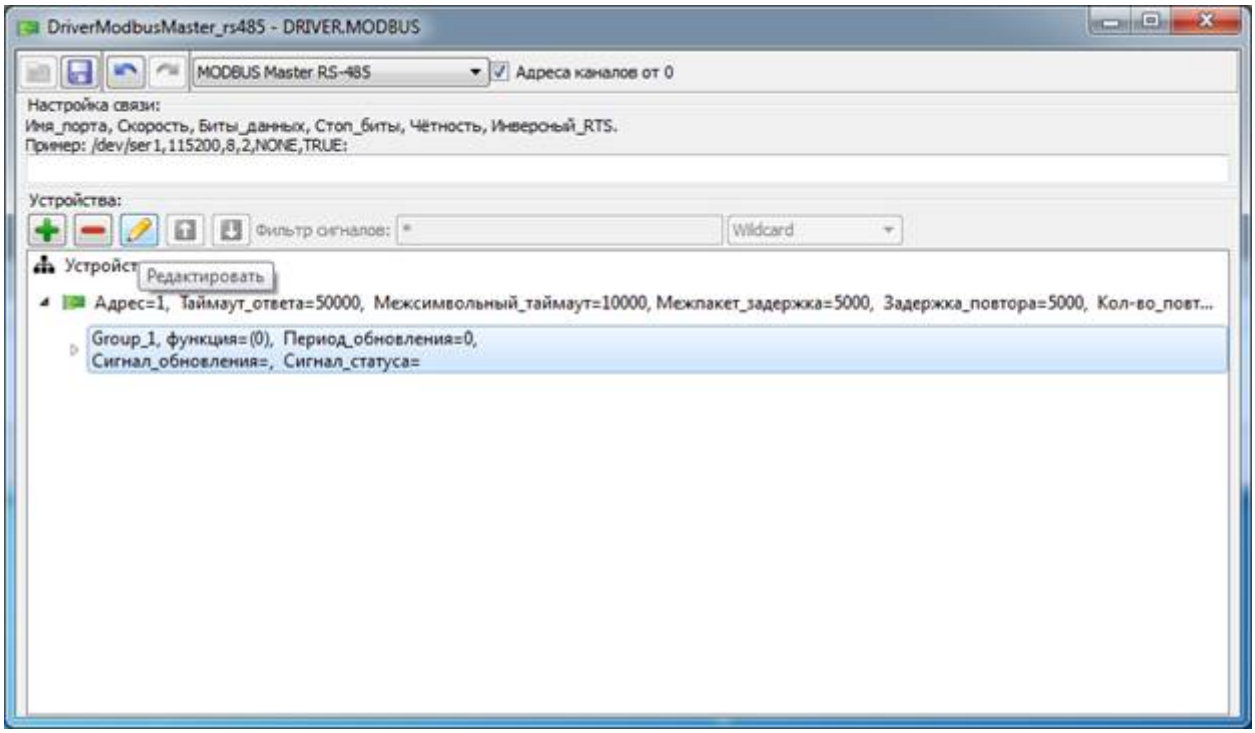


Рисунок 3.193 - Редактирование строки функций взаимодействия

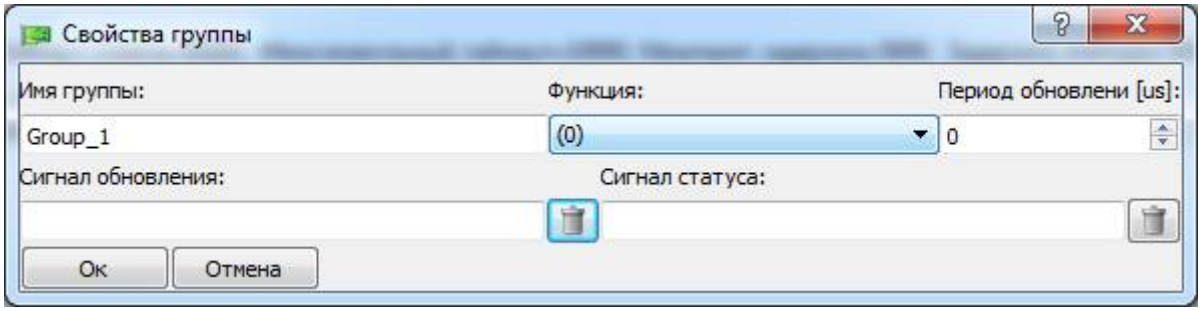


Рисунок 3.194 - Окно Свойства группы

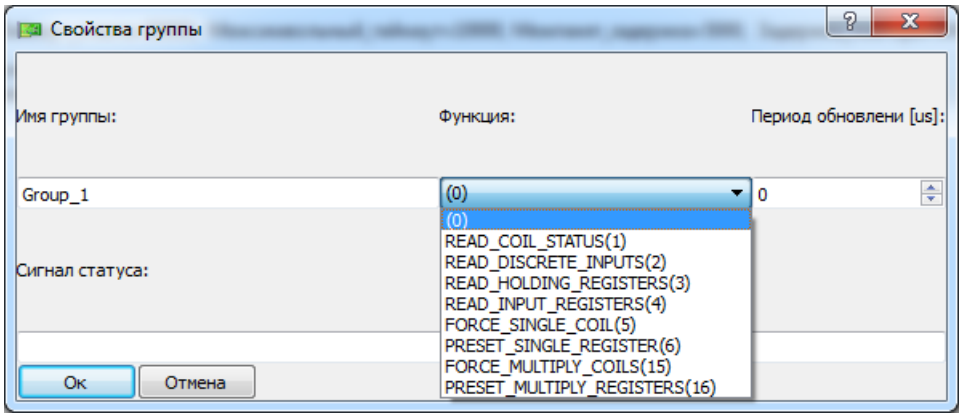


Рисунок 3.195 - Доступные для использования функции

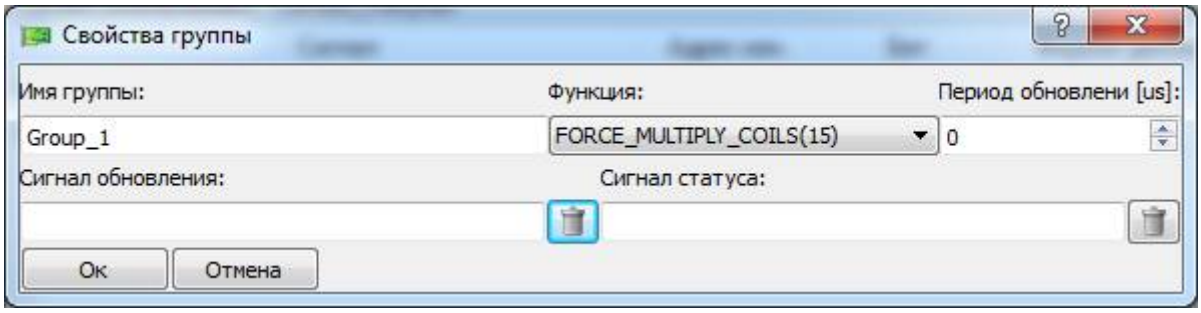


Рисунок 3.196 - Выбрана функция запись значений в несколько дискретных сигналов за раз

Окно **Свойства группы** состоит из следующих полей:

- **Имя группы** - задается для удобства использования пользователем в дальнейшем;

- **Функция** - используемые функции для работы с устройством по протоколу Modbus

* (0) - пустое значение;

* READ_COIL_STATUS(1) - читать значение 1 бита(дискретный сигнал) за раз;

* READ_DISCRETE_INPUTS(2) - читать значение нескольких битов(дискретных сигналов) за раз;

* READ_HOLDING_REGISTERS(3) - читать множество аналоговых сигналов;

* READ_INPUT_REGISTERS(4) - читать множество аналоговых сигналов;

* FORCE_SINGLE_COIL(5) - изменить(отправить) 1 бит(дискретный сигнал) за раз;

* PRESET_SINGLE_REGISTER(6) - изменить(отправить) один аналоговый сигнал(выход) за раз;

* FORCE_MULTIPLY_COILS(15) - изменить(отправить) множество дискретных сигналов за раз;

* PRESET_MULTIPLY_REGISTERS(16) - изменить множество аналоговых сигналов(выходов);

* READ_WRITE_MULTIPLY_REGISTERS(23) - изменить/читать множество аналоговых сигналов.

- **Период обновления [мкс]** - это время, не чаще которого будет опрашиваться данная группа сигналов. Значение равное 0 - опрос с максимально возможной частотой;
Внимание! Задавайте период опроса сопоставимый с типичным временем изменения значений технологических параметров. Например, при опросе температуры не следует выставлять период опроса меньше 1 секунды.

- **Сигнал обновления** - можно привязать сигнал, по изменению которого в TRUE будет однократно вызван опрос данной группы. По завершению опроса сигнал автоматически сбросится в FALSE;

- **Сигнал статуса** - можно привязать сигнал, в который будет записываться результат опроса группы.

ВАЖНО! При пропаже связи с устройством всем группам выставляется статус DEFECTIVE(2). При удачном опросе группы выставляется статус OK(0). Если в группе не удалось опросить часть сигналов, а часть удалось, выставляется статус PARTIALLY_DEFECTIVE(1). Если группа ни разу не опрашивалась, то выставляется статус UNDEFINED(128).

На рис. 3.196 выбрана функция записи значений, которая позволяет управлять каналами на модулях ввода/вывода. Под описанием функции (в раскрывающемся меню) находится таблица для связывания сигналов из интерфейса приложения и данных с устройств (см. рис. 3.197).

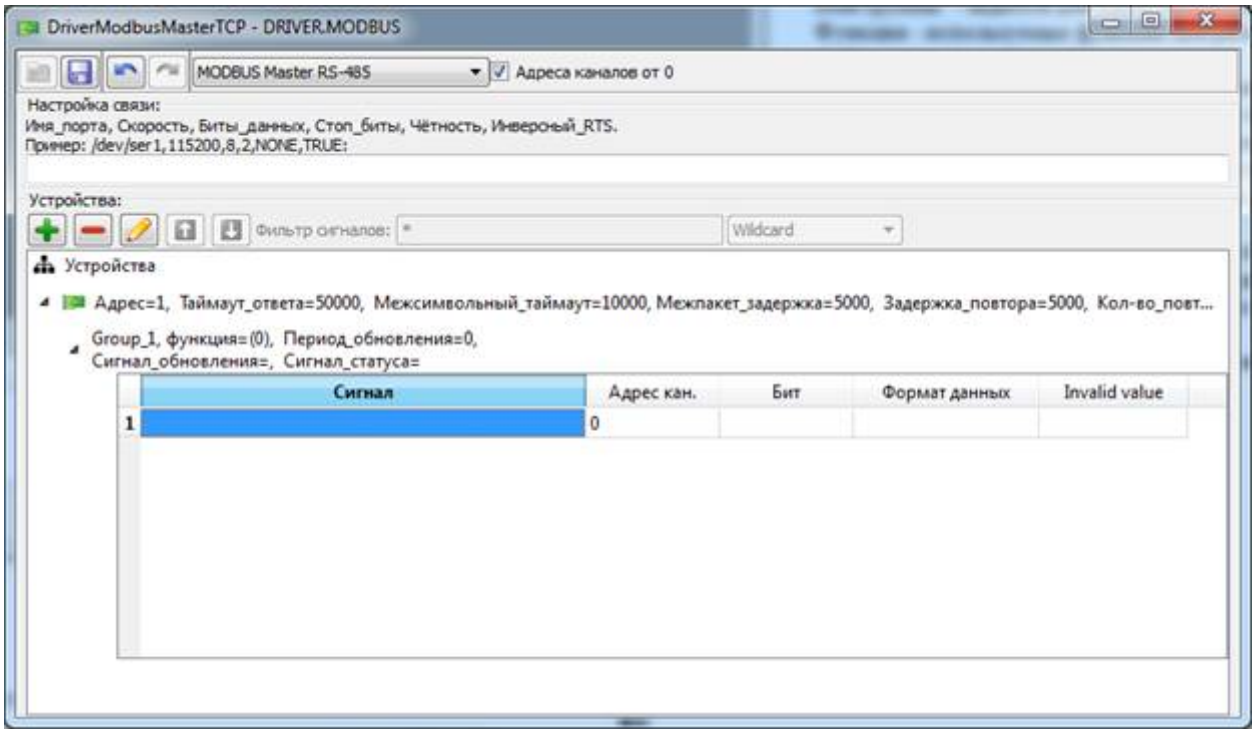


Рисунок 3.197 - Таблица для связывания сигналов
из интерфейса приложения и данных с устройств

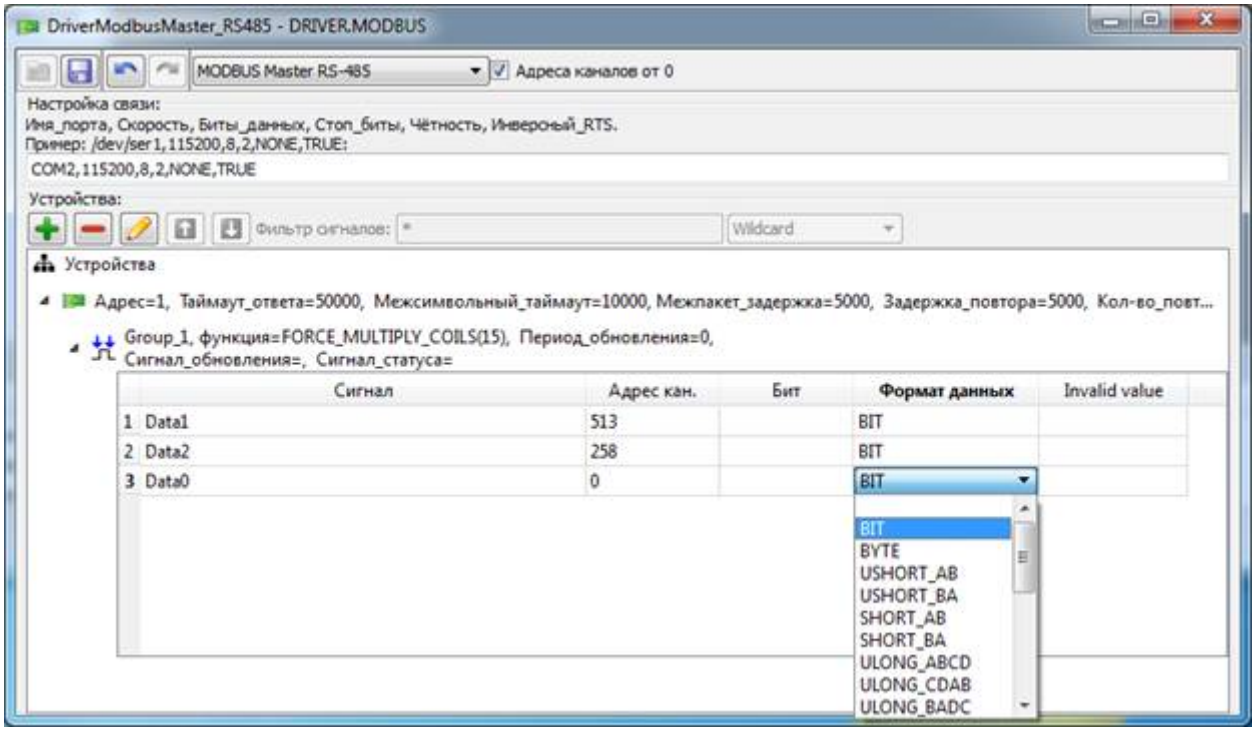


Рисунок 3.198 - Форматы данных

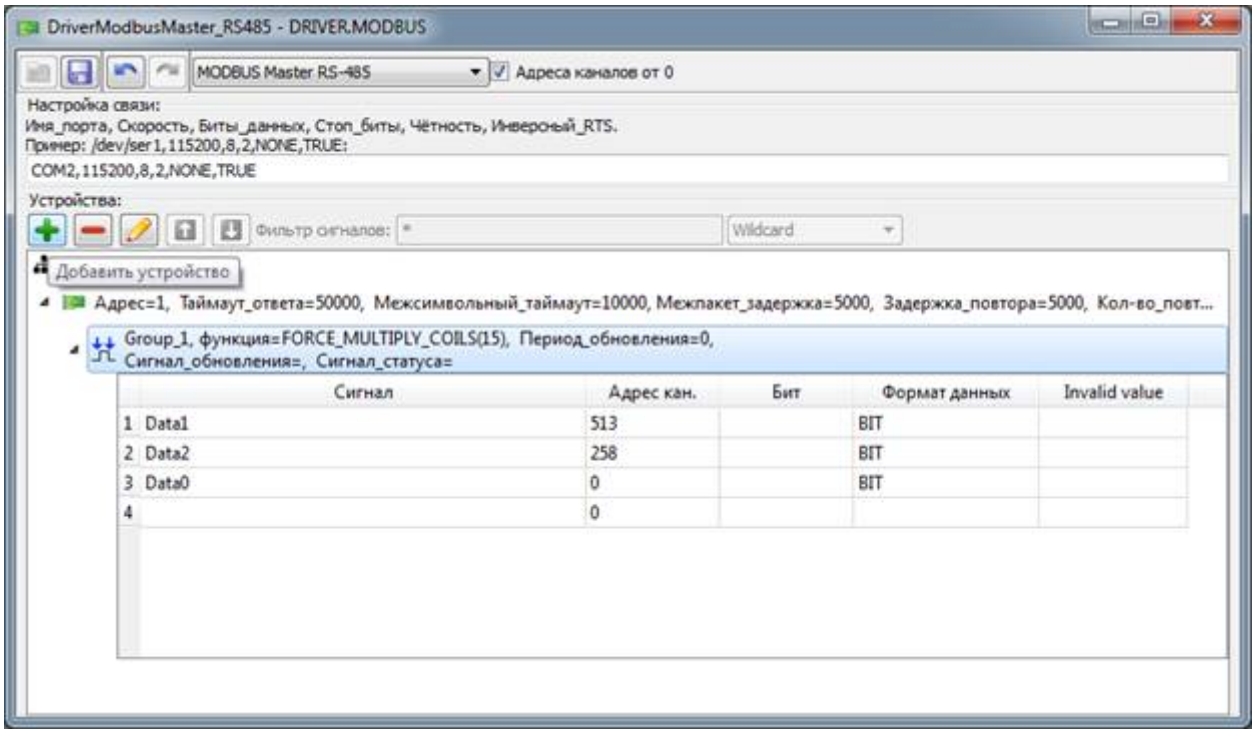


Рисунок 3.199 - Добавление сигналов

На рис. 3.199 показано, как добавить новую строку в таблицу сигналов. Для этого нужно выделить строку описания функции и нажать  .

Таблица сигналов состоит из следующих полей:

- **Сигнал** - привязывается сигнал из интерфейса приложения, в котором будут храниться данные, в соответствии с настройками далее;
- **Адрес канала** - адрес-номер Modbus канала в соответствии с проектной документацией.

Внимание!

В случае работы драйвера MODBUS, как MASTER, разрешено использование одинаковых MODBUS адресов в разных группах запросов для разных сигналов. Запрещено использование одного и того же адреса в пределах одного запроса (т.е. в пределах одной группы);

- **Бит** - можно указать конкретный бит от 0 до 15, значение которого будет храниться в переменной, которая указана в поле сигнал;
- **Формат данных** - в данном поле представлены различные форматы данных;
- **Invalid value** - можно указать значение, которое будет принимать переменная из поля **Сигнал**, при невыполнении функции.

На рис. 3.199 можно видеть несколько сигналов из интерфейса приложения (Data0, Data1 и Data2), которым настроены определенные Modbus адреса сопряжённого устройства.

Значение Modbus адресов нужно смотреть в документации на устройство.

Обратите внимание на нижестоящую таблицу зависимости статуса устройства от стаусов опрашиваемых групп.

Статус группы	**Статус устройства**
Все исправны	Устройство исправно (0)
Часть исправна, остальные НОПР (UNDEFINED)	Устройство исправно (0)
Часть исправнв, остальные неисправны	Устройство частично неисправно (1)
Все неисправны	Устройство неисправно (2)

Статус группы	Статус устройства
Все НОПР (UNDEFINED)	Устройство НОПР (128)

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер **Driver.Modbus** настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на узел проекта (см. раздел 3.2.2.5.1), который имеет связь с Modbus устройством по настроенному ранее порту.

3.3.8.2.4. Примеры работы с приложением-драйвером DRIVER.MODBUS

В данном разделе будут описаны примеры работы с приложением-драйвером DRIVER.MODBUS по двум протоколам: Modbus TCP и Modbus RS-485. В примерах не будет подробно описано как пользоваться приложением-драйвером, так как это описано ранее (см. раздел 3.3.8.2), но будут описаны важные моменты при настройке конкретного оборудования.

3.3.8.2.4.1. Пример работы с приложением-драйвером Modbus по протоколу RS-485

В данном разделе будет разобран пример настройки приложения-драйвера по прототколу RS-485. В примере используются следующие устройства:

- микропроцессорный модуль СОНЕТ СН-МП-ВК с операционной системой QNX (данный модуль используется, как управляющий);
- микропроцессорный модуль СОНЕТ СН-МП-1-3-Р с, подключёнными к нему контроллерами ввода/вывода (данный модуль используется, как подчинённый).

На рис. 3.200 представлена схема подключения модулей в нашем примере. Способ опроса модулей ввода/вывода, которые находятся в корзине с микропроцессорным модулем СН-МП-ВК, в данном примере не будет рассмотрен. Для работы с ними используется драйвер DRIVER.SONET.LOCALBUS. В данном примере будет рассмотрен вариант работы с удаленным микропроцессорным модулем СН-МП-1-3-Р и, подключёнными к нему контроллерами ввода/вывода. Для решения этой задачи удобней использовать DRIVER.SONET.MODBUS, но здесь будет описан вариант решения данной задачи с помощью DRIVER.MODBUS по протоколу RS-485, так как на рисунке к той же шине MODBUS1 подключено ещё другое устройство (в данном примере не важно какое) и чтобы иметь возможность опрашивать и модули ввода/вывода и данное устройство одним приложением-драйвером, необходимо использовать именно DRIVER.MODBUS. Если использовать DRIVER.SONET.MODBUS, то мы сможем опросить только модули ввода/вывода в корзине СН-МП-1-3-Р, а для опроса другого устройства ещё понадобится DRIVER.MODBUS.

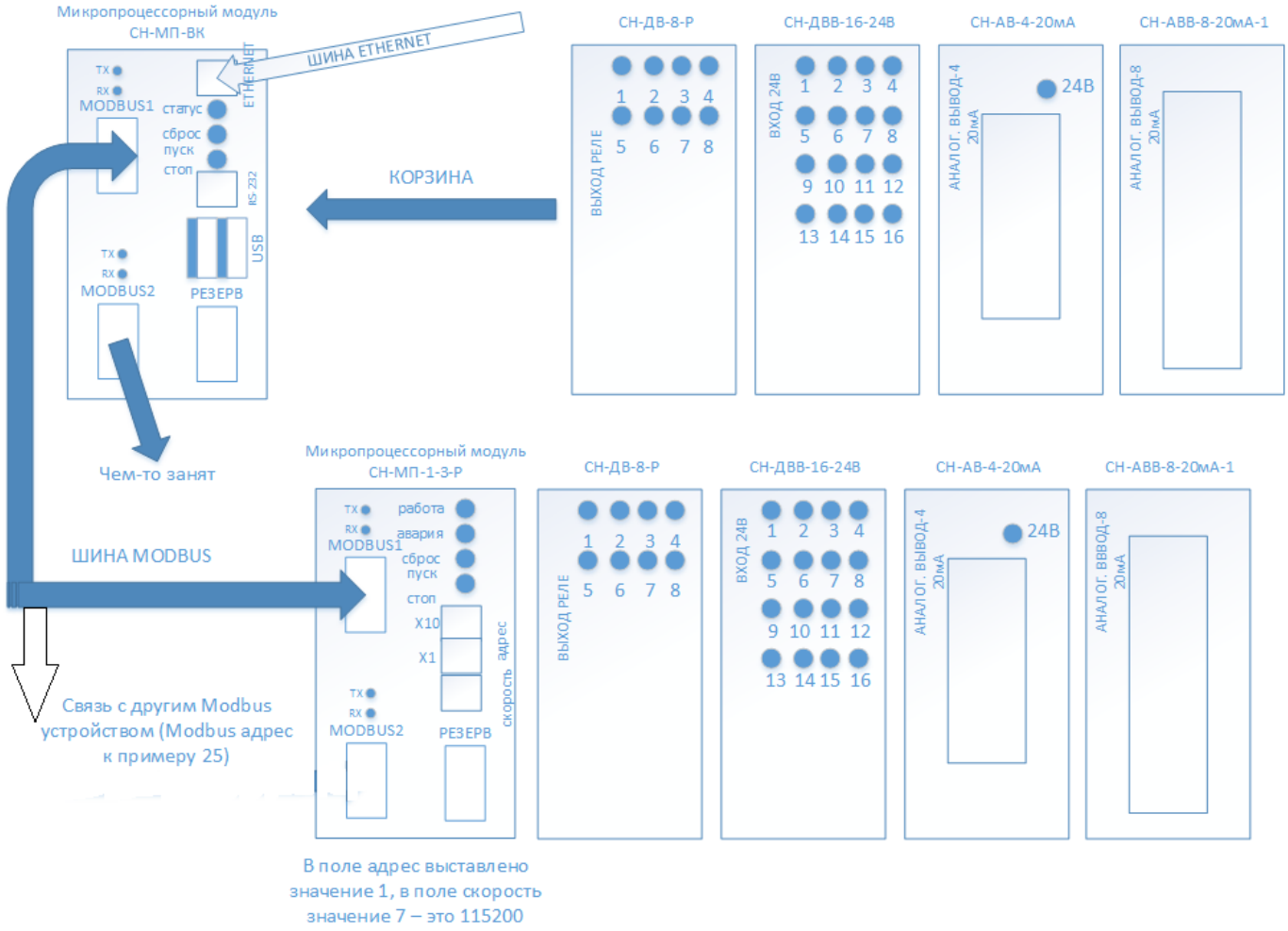


Рисунок 3.200 - Схема подключения модулей в нашем примере

На рис. 3.201 представлено дерево нашего проекта в программе Менеджер проекта. Из рисунка видно, что в проекте настроены следующие элементы:

- Узлы - создан узел CPU20;

- Типы приложений - создано приложение с именем DriverModbusRS485_Master и типом DRIVER.MODBUS;

- Преобразователи - создан преобразователь DSP с именем DSP_AnalogIO.

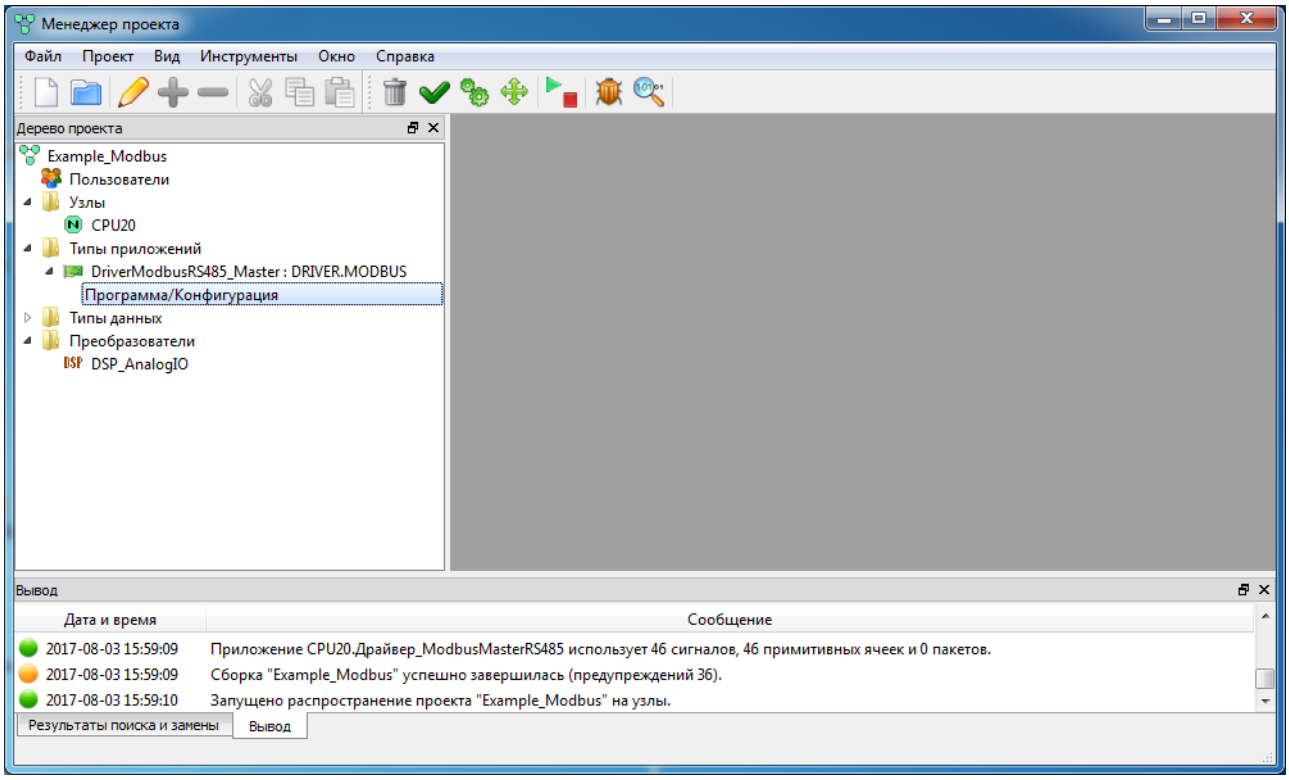


Рисунок 3.201 - Дерево проекта

В первую очередь рассмотрим преобразователь DSP с именем DSP_AnalogIO. Необходимость в данном преобразователе возникает по той причине, что аналоговые модули ввода/вывода для представления токов и напряжений используют внутренние коды и, если просто считывать значения с модуля аналогового входа, то мы получим значение именно во внутренних кодах. В нашем примере используется модуль аналогового вывода СН-АВ-4-20мА(4 выхода по 0-20мА) и аналогового ввода СН-АВВ-8-20мА(8 входов по 0-20мА). Для данных модулей значения выглядят следующим образом: 0мА соответствует 0 во внутреннем коде и 20мА соответствует 65535 во внутреннем коде. Преобразователь используется чтобы в переменные сохранять нужные нам значения токов и напряжений.

На рис. 3.202 представлен настроенный преобразователь DSP для аналоговых модулей ввода/вывода.

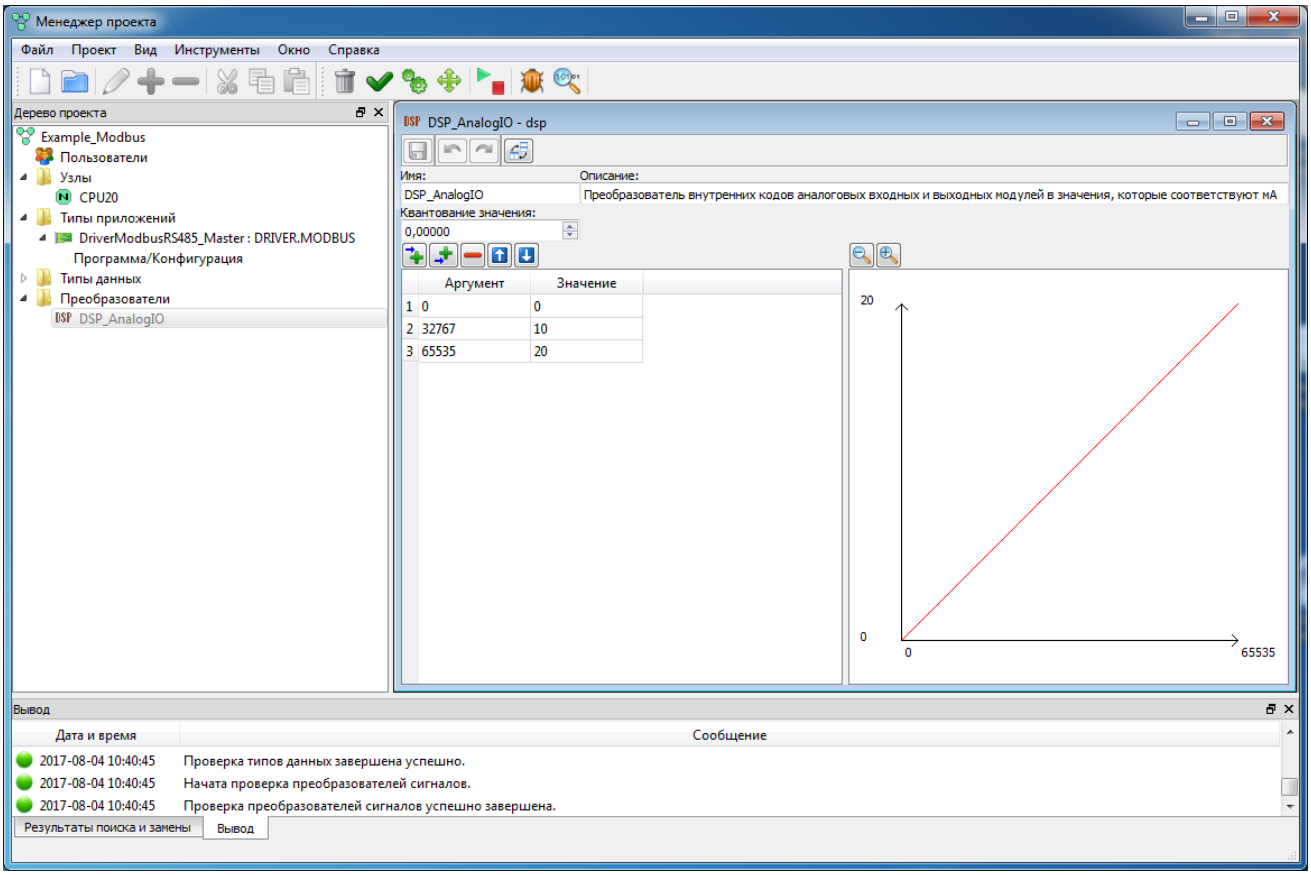


Рисунок 3.202 - Настройка преобразователя DSP

Преобразователь строится по точкам, где Аргумент - это внутренние коды аналогового модуля, а Значение - это то, что мы получим в нашу переменную.

Далее рассмотрим интерфейс приложения-драйвера DRIVER.MODBUS (см. рис. 3.203).

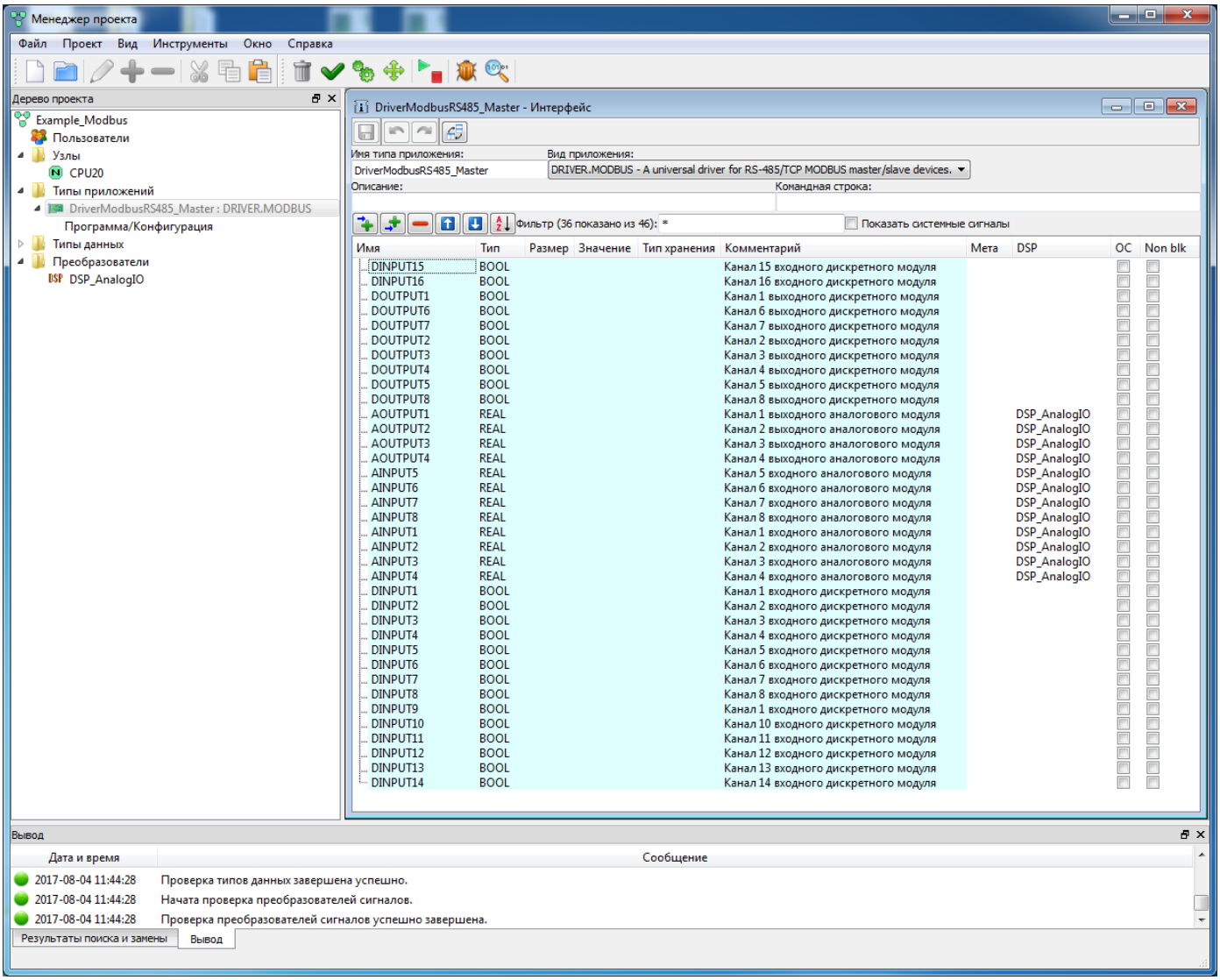


Рисунок 3.203 - Настроенный интерфейс приложения-драйвера DRIVER.MODBUS

В интерфейс приложения-драйвера заведены необходимые сигналы:

- DOUTPUT1..DOUTPUT8 - сигналы дискретного выходного модуля;
- DINPUT1..DINPUT16 - сигналы дискретного входного модуля;
- AOUTPUT1..AOUTPUT4 - сигналы аналогового выходного модуля;
- AINPUT1..AINPUT8 - сигналы аналогового входного модуля.

Так как дискретные значения могут принимать только два значения - это 0 и 1, то им соответствует тип BOOL. Важно правильно определиться с типом для аналоговых сигналов. Передача аналогового сигнала в нашем случае выглядит как передача числа в диапазоне 0 - 65535, а это 4 байта. Значит нам нужно выбирать 4 байтный тип. В интерфейсе поставлен тип REAL, что соответствует 4 байтам и даёт возможность видеть значение после запятой.

Так же обратите внимание в интерфейсе на колонку DSP, в которой стоит значение DSP_AnalogIO. Таким образом используются преобразователи и в результате в сигнале будет храниться преобразованное значение, в нашем случае 0-20мА.

Самым важным моментом является правильная настройка приложения драйвера. На рис. 3.204 изображены общие настройки соединения для приложения-драйвера и настройки для работы с дискретными модулями.

На рисунке видно, что выбран протокол MODBUS Master RS-485, т.к. наше приложение-драйвер будет работать на микропроцессорном модуле СН-МП-ВК, а оно является мастером для модуля СН-МП-1-3-Р.

Далее стоит галочка Адреса каналов от 0, значит нумерация каналов на модулях ввода/вывода идёт не с 1, а с 0.

Настройки связи прописаны для соответствующего выхода с микропроцессорного модуля СН-МП-ВК. По схеме подключения мы используем порт MODBUS1, а это в ОС QNX и в прошивке данного модуля соответствует значению /dev/ser4 (или /dev/ser3 в новых прошивках). Если бы мы использовали порт MODBUS2, то данное значение было бы равно /dev/ser/3 (или /dev/ser2 в новых прошивках). Скорость передачи ставится в соответствии с выставленной на модуле СН-МП-1-3-Р (на рис. 3.200 видно, что это 115200). По документации контроллеры COHET мы узнаем, что посылка состоит и 8 битов данных, 2 стоп битов, без проверки чётности и с инверсным RTS.

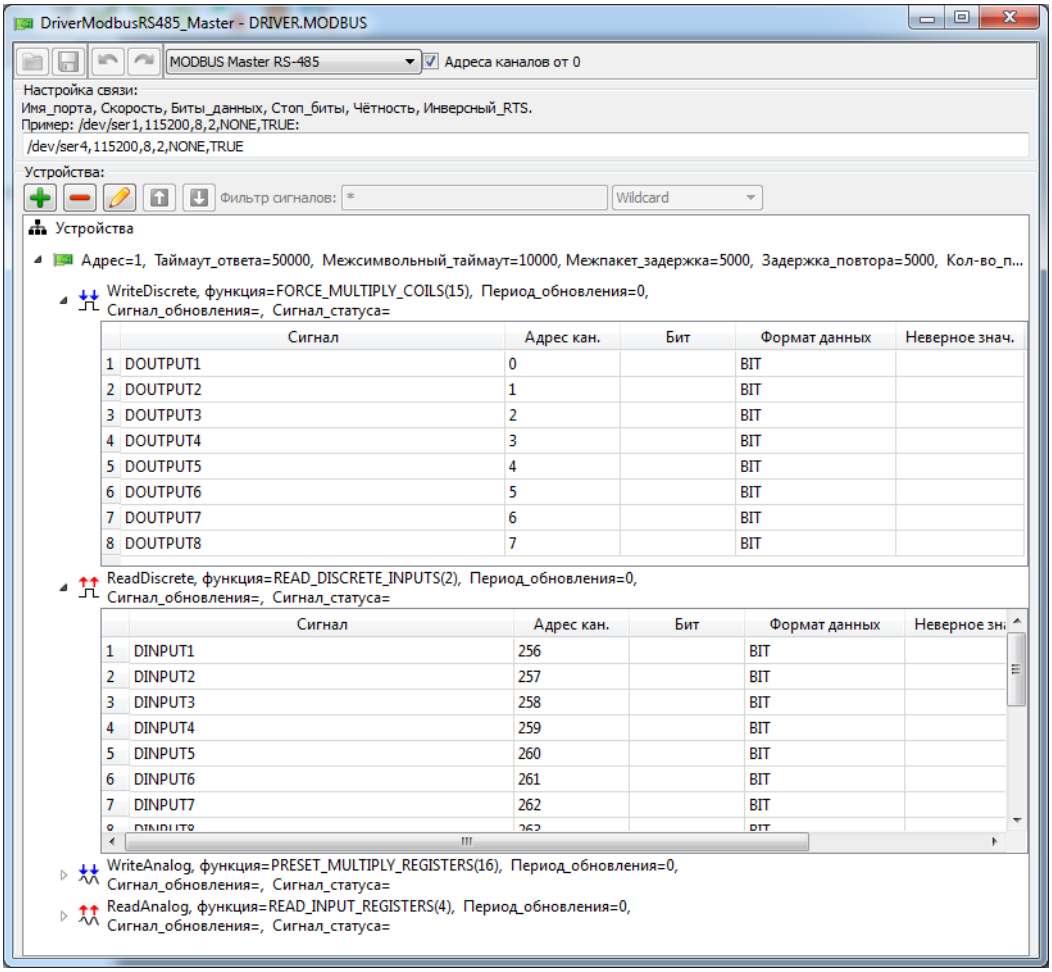


Рисунок 3.204 - Настройка для работы с дискретными сигналами

На рис. 3.205 отображены настройки удалённого-подчинённого устройства, в нашем случае модуля СН-МП-1-3-Р. В поле Адрес устройства в данном случае пишется Modbus адрес устройства (на рис. 3.200 видно, что он выставлен 1). Остальные настройки можно не менять.

ВНИМАНИЕ! Если Modbus адрес устройства задать равным 0 (широковещательный), то драйвер будет отвечать на любой запрос от ведущего устройства по любому адресу ведомого устройства.

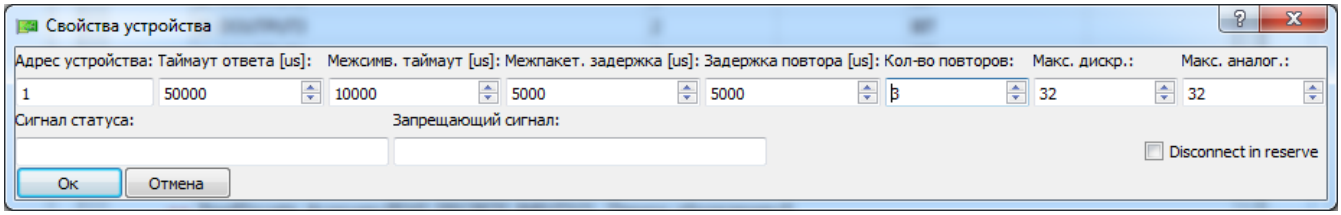


Рисунок 3.205 - Свойства подчинённого устройства

Так же по нашей схеме подключения на рис. 3.200 у нас есть еще одно устройство, подключённое к MODBUS1 модуля СН-МП-ВК и его Modbus адресс равен 25 (см. рис. 3.206). Данное устройство не будем рассматривать в примере, так как оно было приведено для общей картины понимания работы с приложением-драйвером.

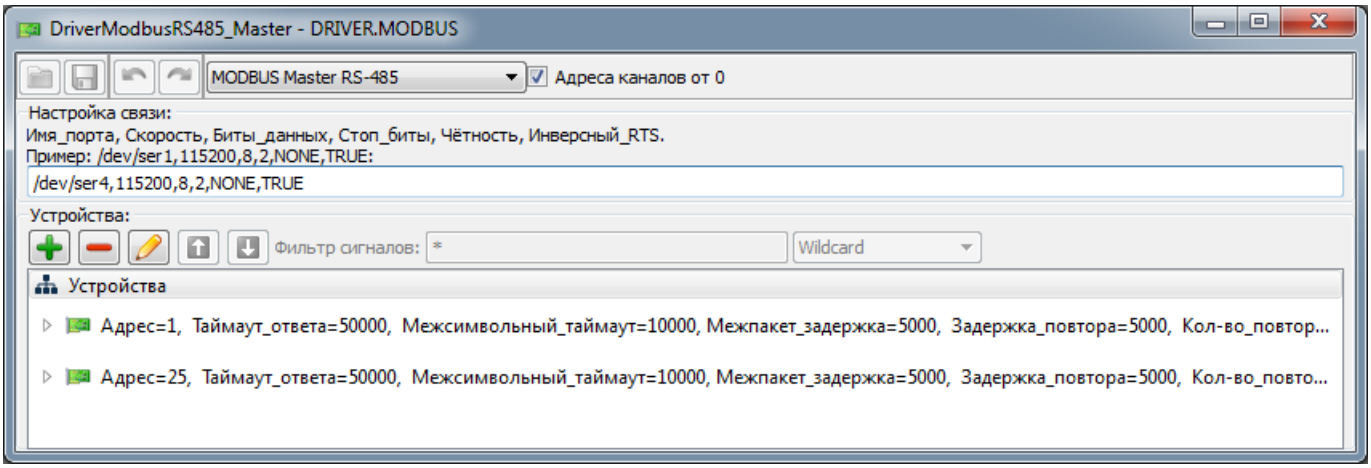


Рисунок 3.206 - Отображение всех Modbus устройств в настройках драйвера

На рис. 3.204 для дискретных модулей настроены две функции:

- FORCE_MULTIPLY_COILS(15) - функция записи множества дискретных значений за раз, используется в данном случае для дискретного выходного модуля;

- READ_DISCRETE_INPUTS(2) - функция чтения нескольких дискретных сигналов за раз, используется в данном случае для дискретного входного модуля.

Формат данных для дискретных значений необходимо поставить BIT.

Адреса каналов, чтобы работать с дискретными значениями, рассчитываются следующим образом. Данный метод используется только для модулей Сонет, что связано с их внутренней прошивкой. Адрес каждого канала состоит из двух байт: старший байт хранит номер позиции модуля ввода/вывода в корзине (позиции считаются от 0), младший байт хранит номер канала модуля (номер канала считается от 0). Таким образом каналы нашего дискретного выходного модуля, который по схеме визуально стоит в первой позиции(но при счёте от 0, он в нулевой позиции), показаны в таблице ниже (каналы тоже считаются от 0).

Старший байт(позиция модуля)	**Младший байт(номер канала)**	**Адрес канала(в десятичном формате)**	**Описание**
00000000	00000000	0	Первый канал модуля в 0 позиции
00000000	00000001	1	Второй канал модуля в 0 позиции
00000000	00000010	2	Третьий канал модуля в 0 позиции
00000000	00000011	3	Четвёртый канал модуля в 0 позиции
00000000	00000100	4	Пятый канал модуля в 0 позиции
00000000	00000101	5	Шестой канал модуля в 0 позиции

Старший байт(позиция модуля)	Младший байт(номер канала)	Адрес канала(в десятичном формате)	Описание
00000000	00000110	6	Седьмой канал модуля в 0 позиции
00000000	00000111	7	Восьмой канал модуля в 0 позиции

Адреса каналов дискретного входного модуля, который стоит визуально во второй позиции, но при счёте от 0 он будет иметь позицию 1, показаны в таблице ниже(каналы тоже считаются от 0).

Старший байт(позиция модуля)	Младший байт(номер канала)	Адрес канала(в десятичном формате)	Описание
00000001	00000000	256	Первый канал модуля в 1 позиции
00000001	00000001	257	Второй канал модуля в 1 позиции
00000001	00000010	258	Третьий канал модуля в 1 позиции
00000001	00000011	259	Четвёртый канал модуля в 1 позиции
00000001	00000100	260	Пятый канал модуля в 1 позиции
00000001	00000101	261	Шестой канал модуля в 1 позиции
00000001	00000110	262	Седьмой канал модуля в 1 позиции
00000001	00000111	263	Восьмой канал модуля в 1 позиции
-	-	-	-
00000001	00010000	271	Шестнадцатый канал модуля в 1 позиции

Далее рассмотрим настройку аналоговых модулей.

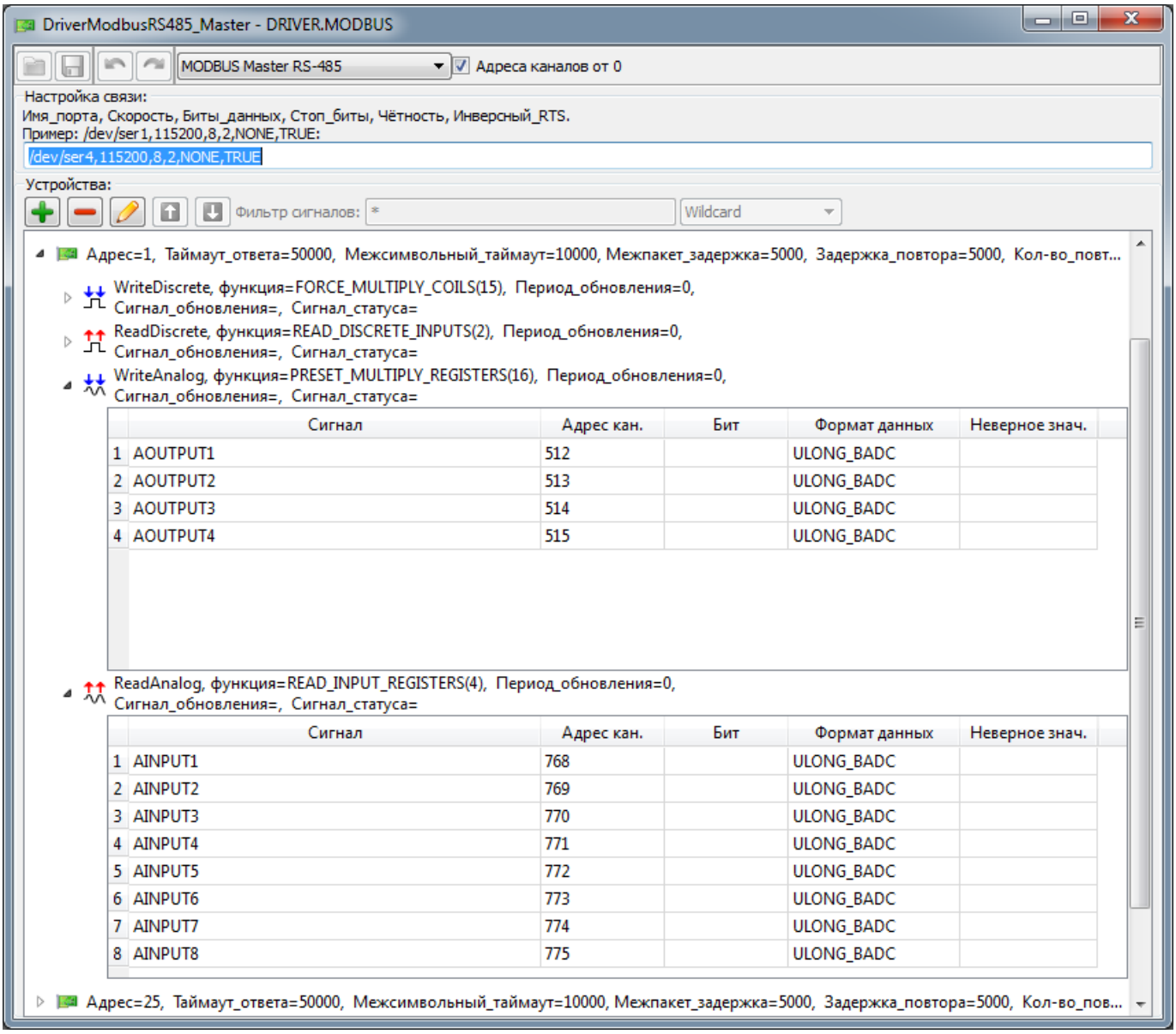


Рисунок 3.207 - Настройки для работы с аналоговыми сигналами

На рис. 3.207 представлены настройки приложения-драйвера Modbus для работы с аналоговыми сигналами. Для этого настроены две функции:

- PRESET_MULTIPLY_REGISTERS(16) - изменить множество аналоговых сигналов(выходов), используется в данном случае для аналогового выходного модуля;

- READ_INPUT_REGISTERS(4) - читать множество аналоговых сигналов, используется в данном случае для аналогового входного модуля.

Формат данных следует выбирать со знанием тех устройств, с которыми вы работаете. В данном примере мы работаем с контроллерами Сонет и, читая документацию на эти устройства, узнаём, что формат передачи данных у них младшим битом вперёд и посылка занимает 4 байта + данные, которые у нас сами занимают 4 байта (т.к. значения от 0 до 65535), в итоге 8 байт. Этому соответствует формат данных ULONG_BADC.

Адреса каналов рассчитываются аналогично дискретным модулям.

На таблице ниже представлены адреса каналов для аналогового выходного модуля, который визуально стоит 3 по порядку, но так как счёт от 0, то у него позиция 2. Каналы так же считаем от 0.

Старший байт(позиция модуля)	Младший байт(номер канала)	Адрес канала(в десятичном формате)	Описание
00000010	00000000	512	Первый канал модуля во 2 позиции
00000010	00000001	513	Второй канал модуля во 2 позиции
00000010	00000010	514	Третьий канал модуля во 2 позиции
00000010	00000011	515	Четвёртый канал модуля во 2 позиции

На таблице ниже представлены адреса каналов для аналогового входного модуля, который визуально стоит 4 по порядку, но так как счёт от 0, то у него позиция 3. Каналы так же считаем от 0.

Старший байт(позиция модуля)	Младший байт(номер канала)	Адрес канала(в десятичном формате)	Описание
00000011	00000000	768	Первый канал модуля в 3 позиции
00000011	00000001	769	Второй канал модуля в 3 позиции
00000011	00000010	770	Третьий канал модуля в 3 позиции
00000011	00000011	771	Четвёртый канал модуля в 3 позиции
00000011	00000100	772	Пятый канал модуля в 3 позиции
00000011	00000101	773	Шестой канал модуля в 3 позиции
00000011	00000110	774	Седьмой канал модуля в 3 позиции
00000011	00000111	775	Восьмой канал модуля в 3 позиции

Приложение-драйвер Modbus настроен по нашей схеме подключения.

Для того чтобы приложение-драйвер Modbus включить в проект, необходимо создать узел, в нашем случае это CPU20 и добавить его (см. рис. 3.208).

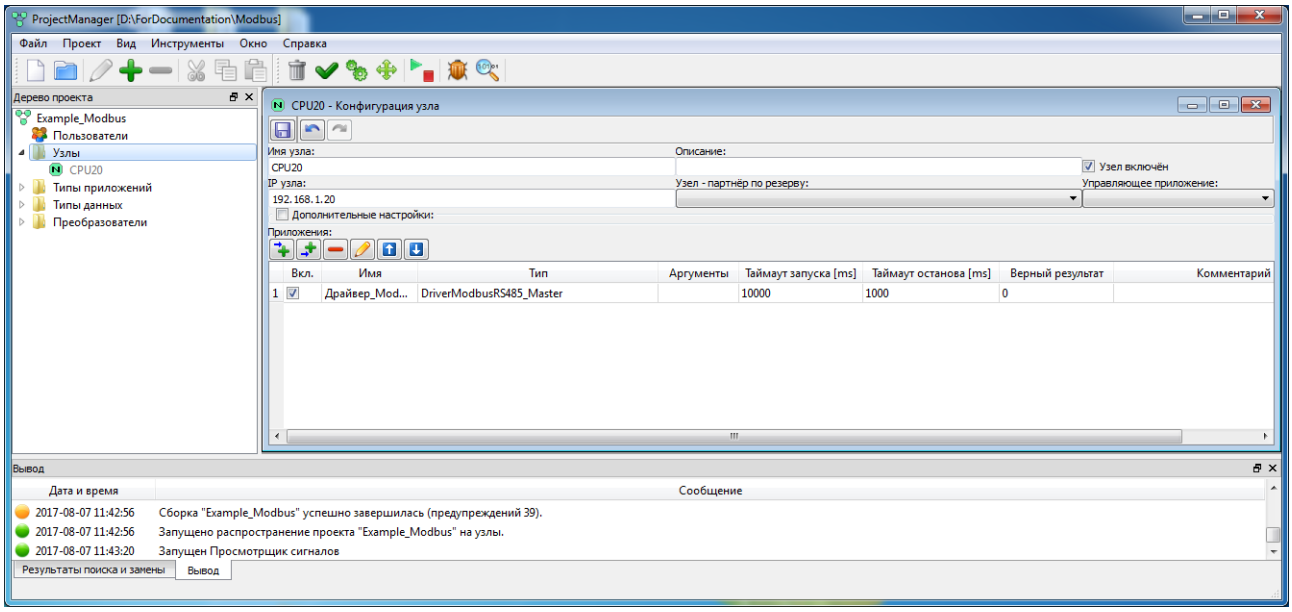


Рисунок 3.208 - Настройки узла проекта

Таким образом мы получаем информацию о наших Modbus устройствах и можем использовать её в любом месте нашего проекта.

3.3.8.2.4.2. Пример работы с приложением-драйвером Modbus по протоколу TCP

В данном разделе описывается пример настройки приложения-драйвера Modbus по протоколу TCP. На рис. 3.209 представлена схема, на которой компьютер с ip-адресом 192.168.1.93 (далее компьютер) по сети Ethernet соединён с программируемым контроллером Сонет СН-МП-ВК с ip-адресом 192.168.1.30 (далее контроллер). Приём и передача данных будет осуществляться по протоколу Modbus TCP. В данном примере будут использоваться приложения-драйвера SCADA-системы "Соната" и на компьютере будет настроен Master(управляющий), а на контроллере Slave(подчинённый).

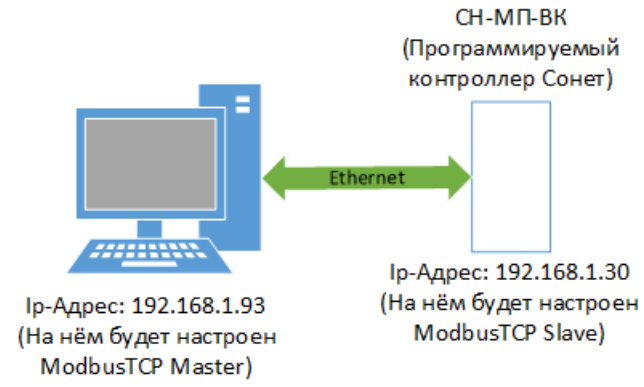


Рисунок 3.209 - Схема, разбираемая в данном примере

В данном примере мы рассмотрим чтение/запись дискретных значений и чтение/запись аналоговых значений.

1. Создаём приложение-драйвер Modbus TCP Master.

а) создаём новое приложение;

б) данному приложению задаём тип DRIVER.MODBUS;

в) добавляем сигналы в интерфейс приложения: M_DOUT (сигнал дискретного выхода на мастере), M_DIN (сигнал дискретного входа на мастере), M_AOUT (сигнал аналогового выхода на мастере) и M_AIN (сигнал аналогового входа на мастере), как на рис. 3.210;

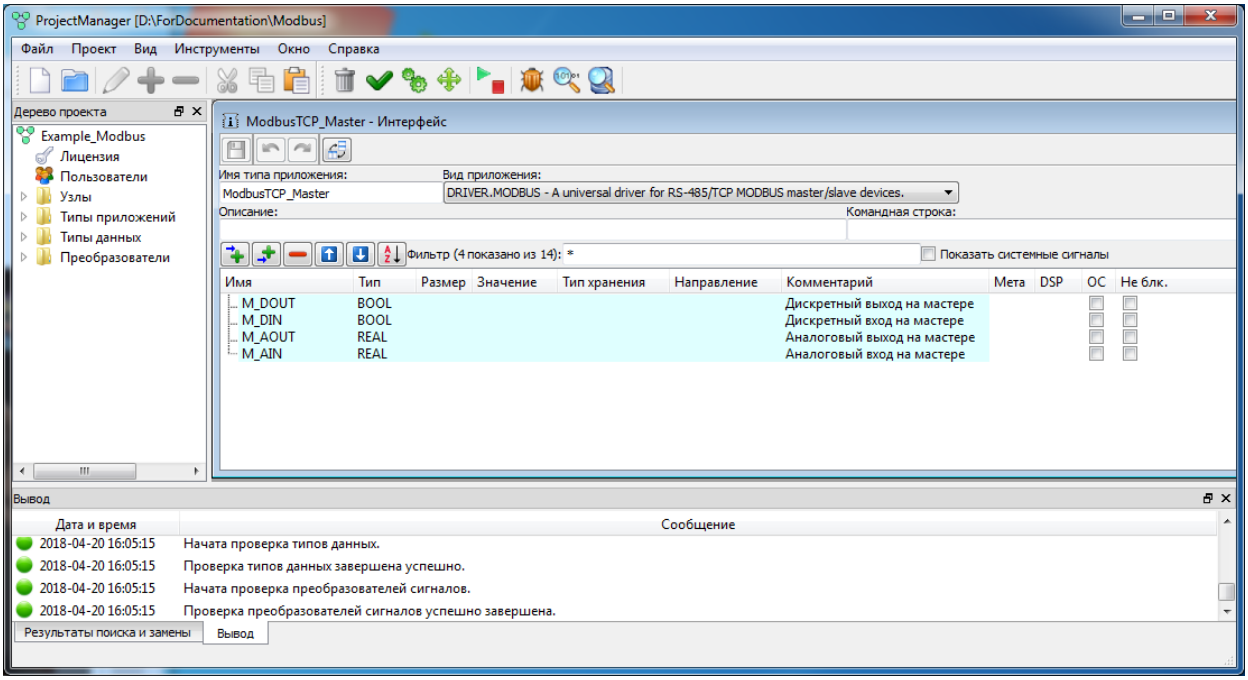


Рисунок 3.210 - Интерфейс приложения-драйвера Modbus (планируем его сделать Master)

г) далее настраиваем конфигурацию приложения-драйвера, как на рисунках ниже

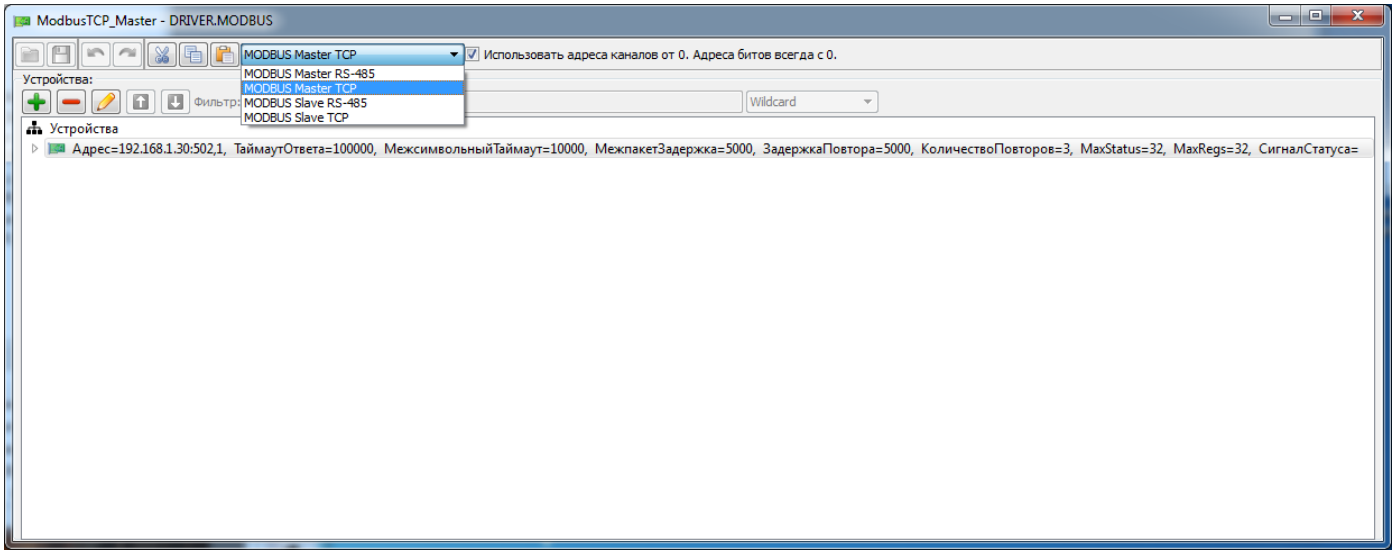


Рисунок 3.211 - Выбор протокола Modbus Master TCP

В поле **Адрес устройства** указываем ip-адрес подчинённого устройства, далее 502 - стандартный порт для Modbus TCP и Modbus адрес подчинённого устройства.

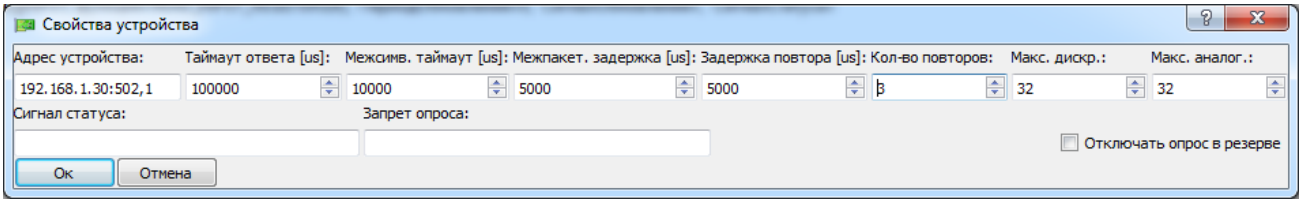


Рисунок 3.212 - Настройки свойств устройства Master

Для записи в дискретные выхода воспользуемся стандартной функцией протокола Modbus номер 15.

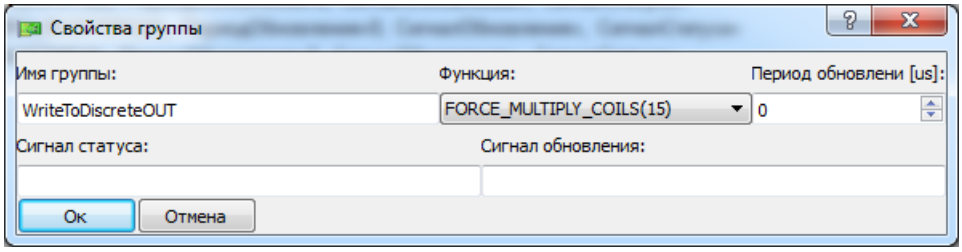


Рисунок 3.213 - Настройки функции записи в дискретные выхода

Далее настраиваем кроссконнект, в котором для дискретного выхода в поле **Адрес кан.** указываем значение 1. Данное значение мы должны узнать у подчинённого устройства из его документации. В данном примере, предположим, что по данному адресу мы обращаемся к одному из дискретных выходов на подчинённом устройстве.

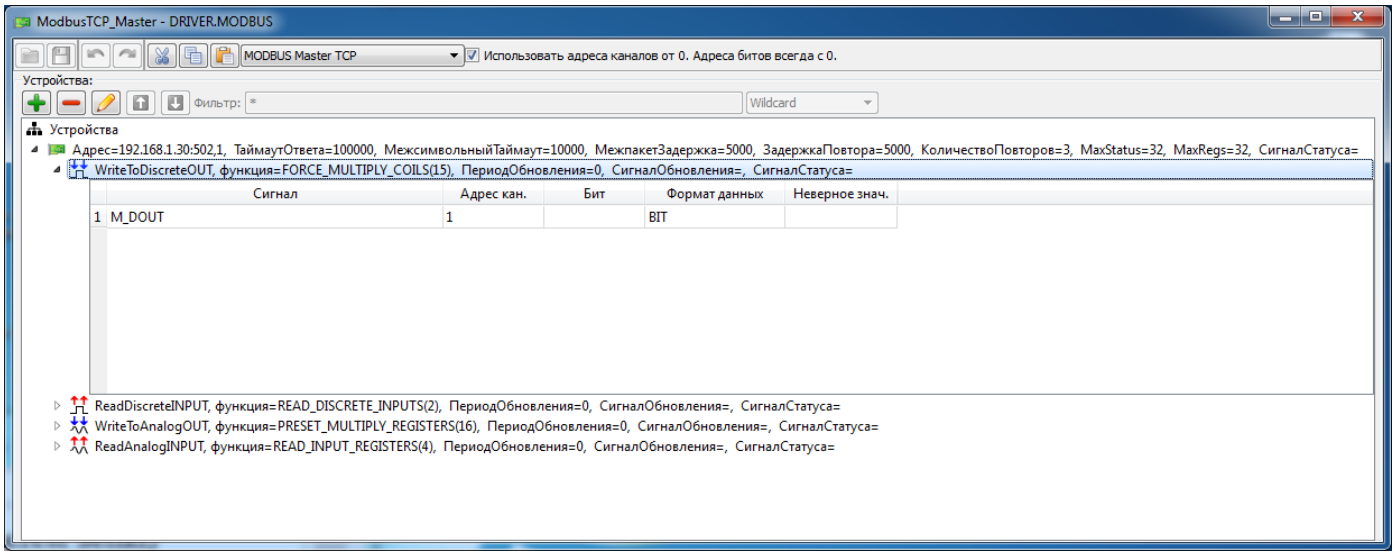


Рисунок 3.214 - Настройка кроссконнекта для функции записи в дискретные выхода

Аналогичным образом настраиваем другие функции (см. рис. 3.215):

- для чтения дискретных входов используем функцию номер 2 (READ_DISCRETE_INPUTS) и настроим кроссконнект, читая значение по адресу 2;

- для записи в аналоговые значения используем функцию номер 16 (PRESET_MULTIPLY_REGISTERS) и настроим кроссконнект на запись в регистр с адресом 1, порядок передачи байт прямой;

- для чтения аналоговых значений используем функцию 4 (READ_INPUT_REGISTERS) и настроим кроссконнект на чтение регистра по адресу 1, но это другое адресное пространство и поэтому мы не пересечёмся с адресом 1, в который мы пишем ранее функцией номер 16, порядок передачи байт прямой.

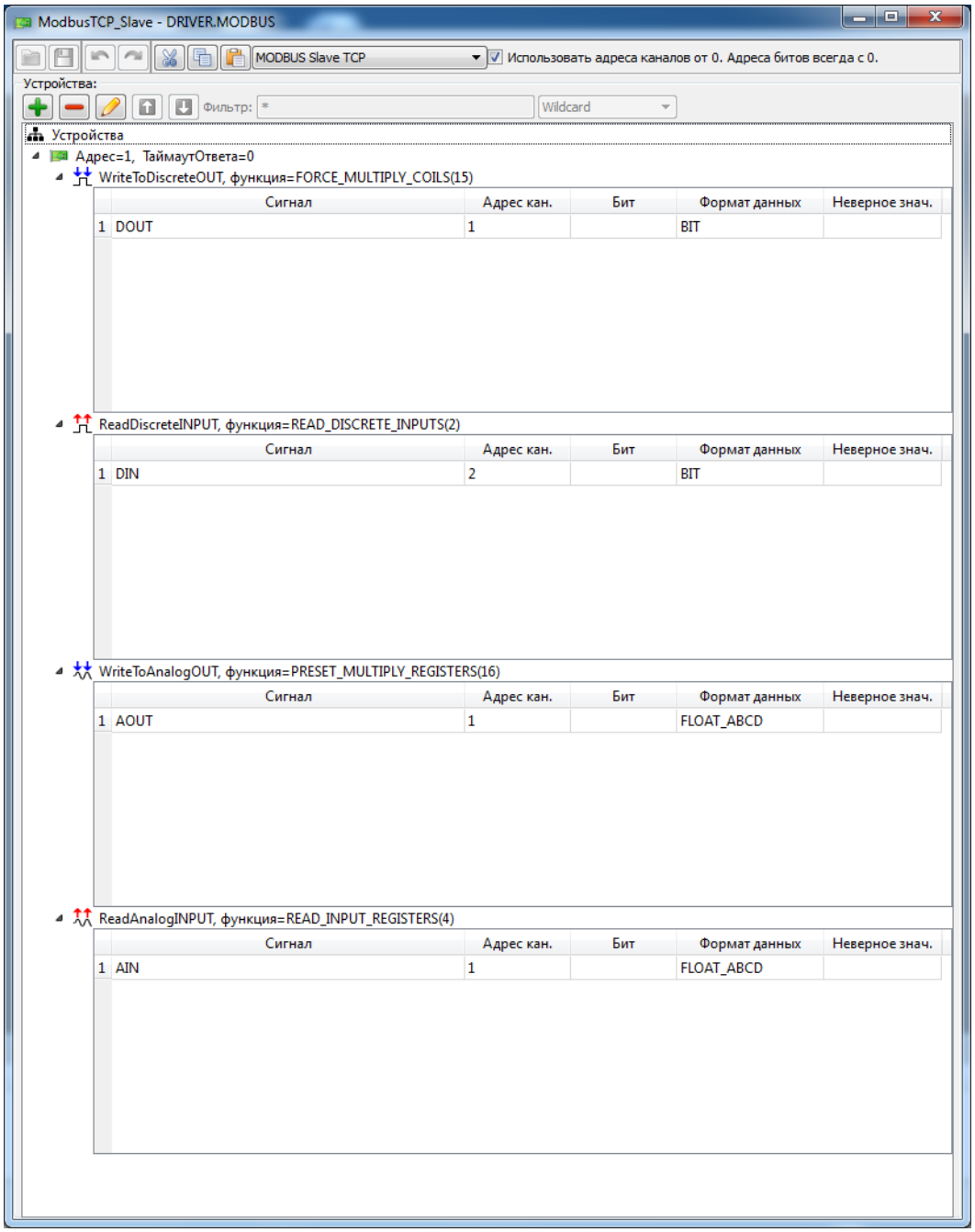


Рисунок 3.215 - Вся конфигурация для приложения-драйвера Modbus TSP Master

2. Создаём приложение-драйвер Modbus TCP Slave.

а) создаём новое приложение;

б) данному приложению задаём тип DRIVER.MODBUS;

в) добавляем сигналы в интерфейс приложения: DOUT (сигнал дискретного выхода на слэйве), DIN (сигнал дискретного входа на слэйве), AOUT (сигнал аналогового выхода на слэйве) и AIN (сигнал аналогового входа на слэйве), как на рис. 3.210;

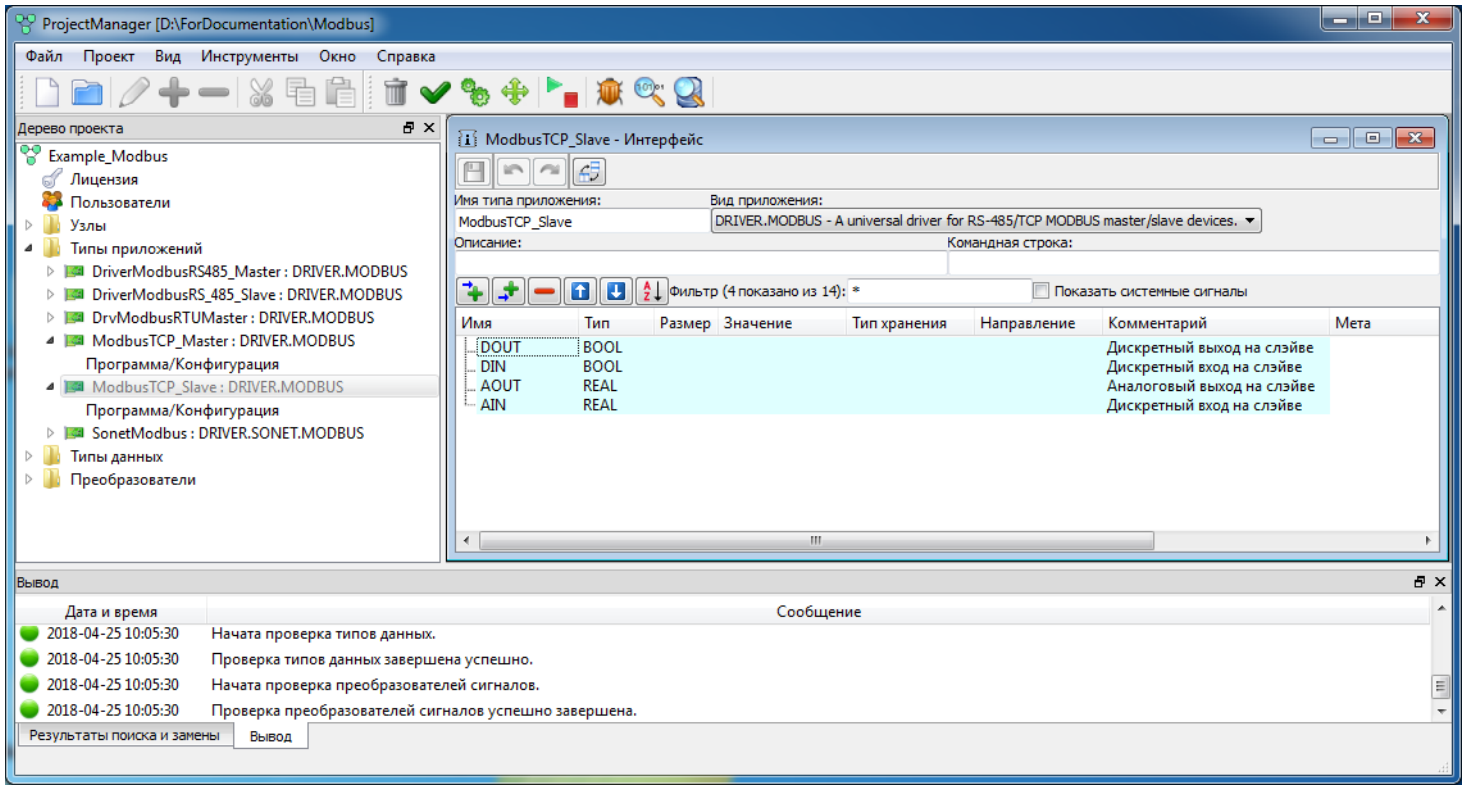


Рисунок 3.216 - Интерфейс приложения-драйвера Modbus TCP Slave

г) далее настраиваем конфигурацию приложения-драйвера, как на рисунках ниже

ВНИМАНИЕ! Если Modbus адрес устройства задать равным 0 (широковещательный), то драйвер будет отвечать на любой запрос от ведущего устройства по любому адресу ведомого устройства.

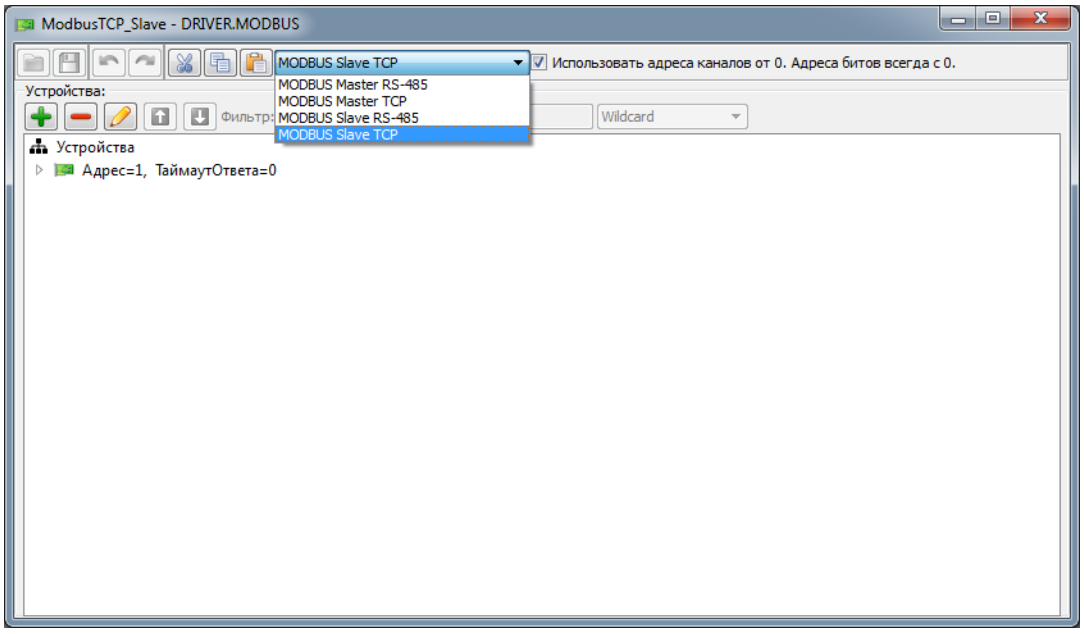


Рисунок 3.217 - Выбор протокола Modbus Slave TCP

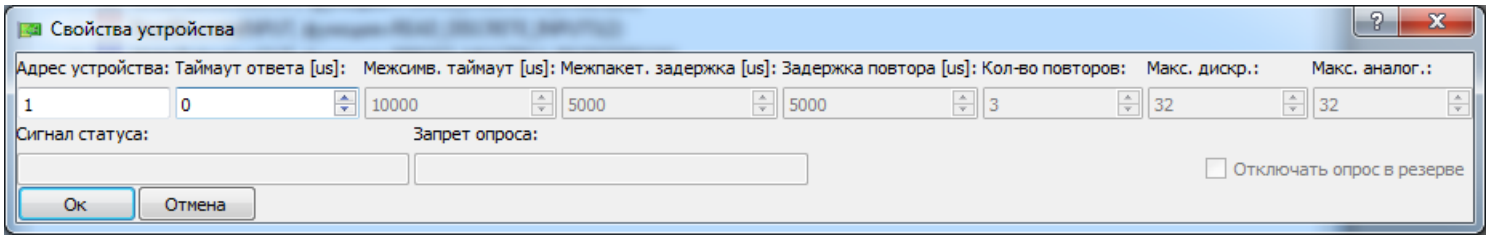


Рисунок 3.218 - Настройка свойств устройства Slave

На рис. 3.218 в поле **Адрес устройства** ставим Modbus адрес равный 1. Данное значение используется при настройке Master, когда мы настраиваем свойства опрашиваемых устройств.

Далее на рис. 3.219 представлена конфигурация всех функций для приложения-драйвера Modbus TCP Slave. Конфигурация функций для Slave совпадает с конфигурацией функций для Master.

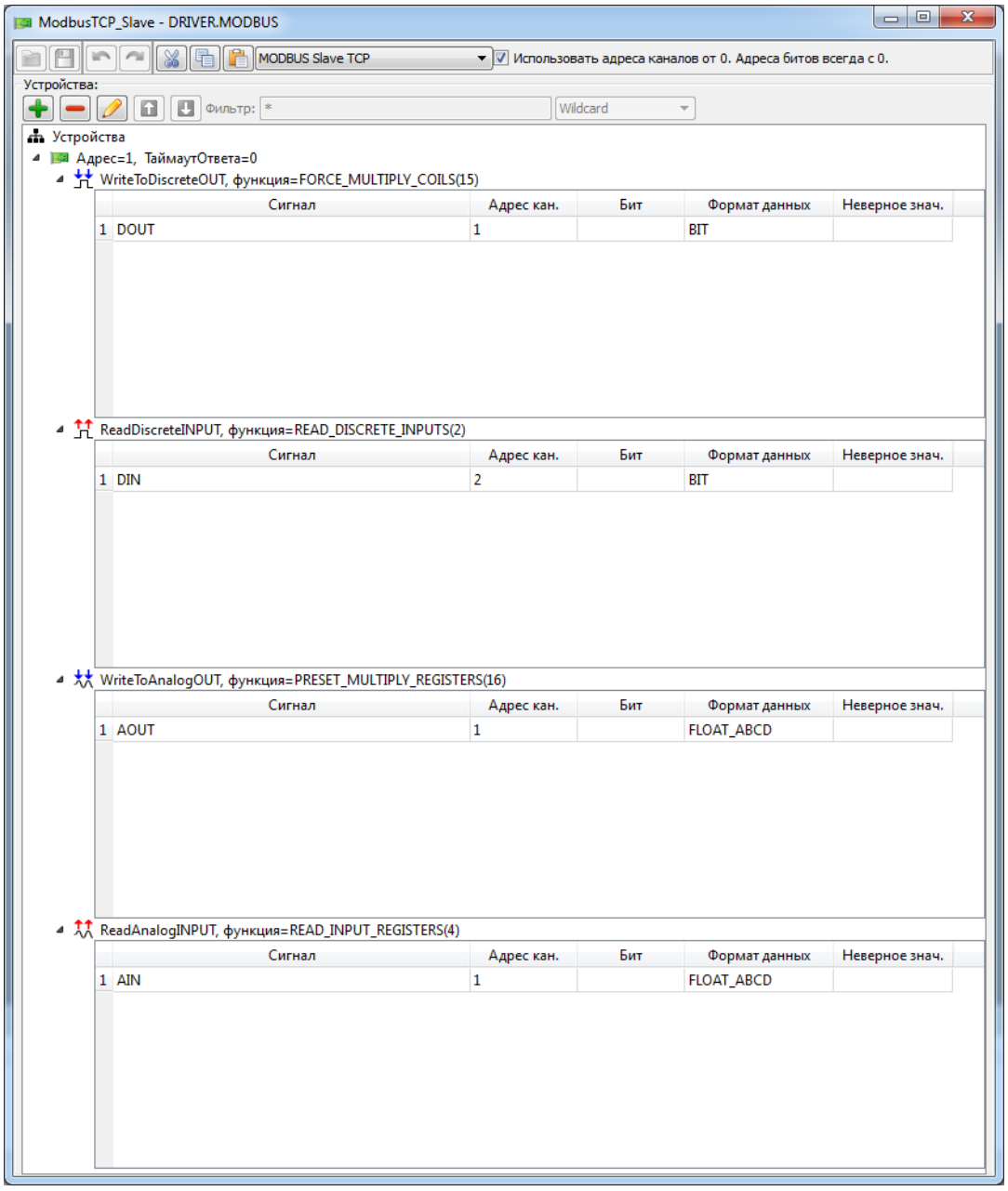


Рисунок 3.219 - Вся конфигурация для приложения-драйвера Modbus TCP Slave

Если далее эти приложения-драйвера добавить на разные узлы, Master на компьютер, а Slave на контроллер, то с помощью программы Просмотрщик сигналов можно будет убедиться в том, что

с компьютера можно изменять сигналы M_DOUT и M_AOUT, т.е. писать в выхода и, изменяя на контроллере сигналы DIN и AIN, получать значения со входов.

Данным образом работают в паре приложения-драйвера DRIVER.MODBUS Master и Slave по протоколу TCP. При настройке работы со сторонними устройствами, учитывайте, что функции с обоих сторон должны быть одинаковыми и стройте логику функций, исходя из того, что будет делать Master, а Slave дублирует функции.

3.3.8.3. Создание приложения-драйвера OPCUA-клиент

Приложение-драйвер OPCUA-клиент применяется для обеспечения доступа SCADA-системы "СОНАТА" к OPCUA-серверам.

Создание приложения-драйвера OPCUA-клиент осуществляется аналогично другим приложениям-драйверам SCADA-системы "СОНАТА" (см. раздел 3.2.2.4).

Далее описан упрощенный порядок действий при создании приложения-драйвера OPCUA-клиент и подробно описаны важные моменты настройки данного приложения-драйвера:

1. В программе **ProjectManager** необходимо создать новый проект (см. раздел 3.2.2.1).

2. Добавить новое приложение и указать тип приложения **DRIVER.OPCUA**(см. рис. 3.220)

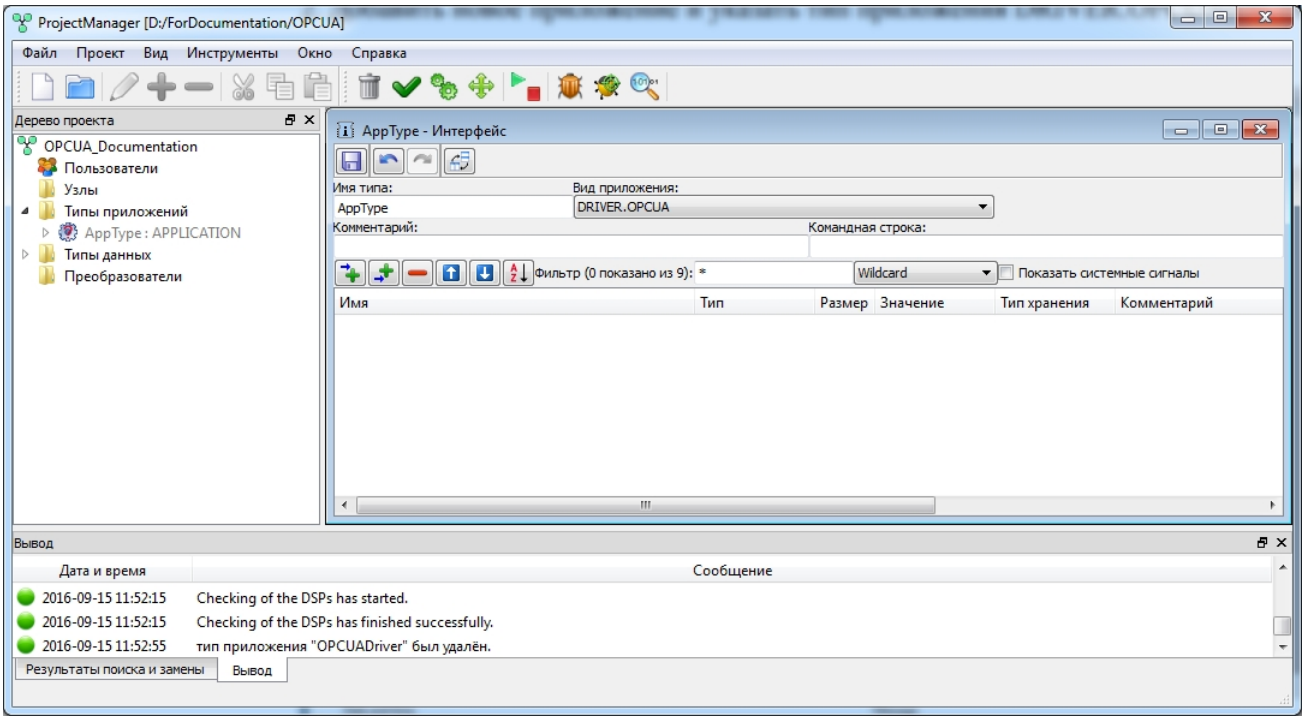


Рисунок 3.220 - Изменение типа приложения в интерфейсе приложения

В поле **Имя** при помощи кнопок  и  можно добавить локальные или глобальные сигналы SCADA-системы "СОНАТА" (см. раздел 3.2.2.4.3.2), которые могут понадобиться в дальнейшем при настройке приложения-драйвера(см. рис. 3.221).

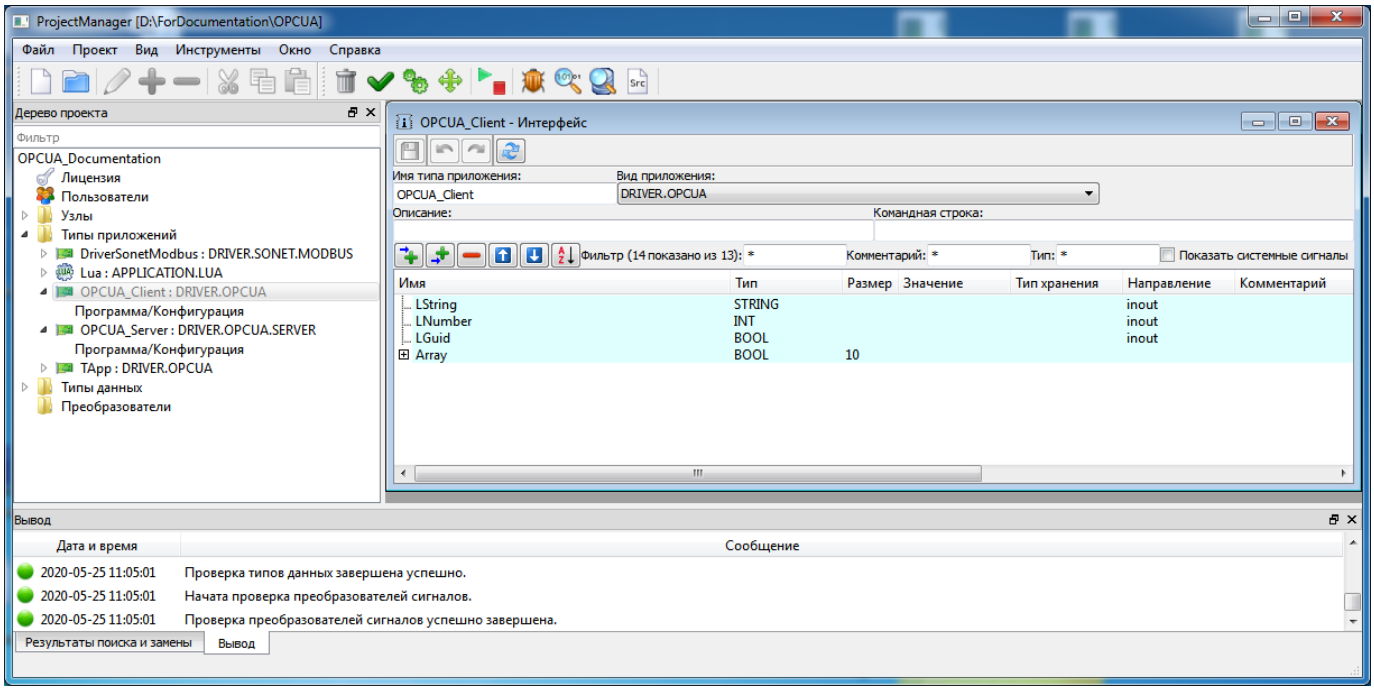


Рисунок 3.221 - Интерфейс приложения-драйвера OPCUA-клиент

На вопрос о разрешении замены типа приложения ответить “Да”(см. рис. 3.222).

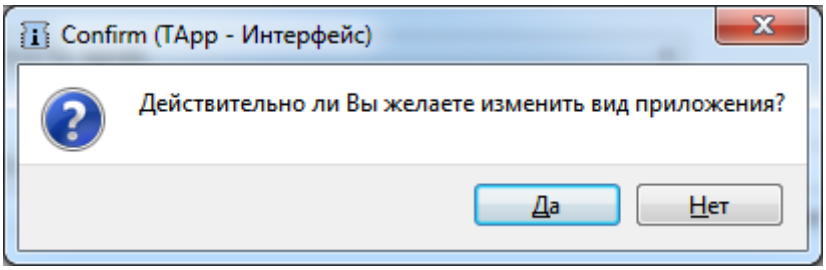


Рисунок 3.222 - Окно подтверждения изменений в интерфейсе приложения

3. Дальнейшие действия не отличаются от аналогичных действий для других приложений-драйверов.

Заходим в меню Программа/Конфигурация и производим настройку приложения-драйвера OPCUA-клиент (см. раздел 3.2.2.4.4).

Для редактирования будет вызван специфичный для приложения-драйвера OPCUA-клиента редактор(см. рис. 3.223).

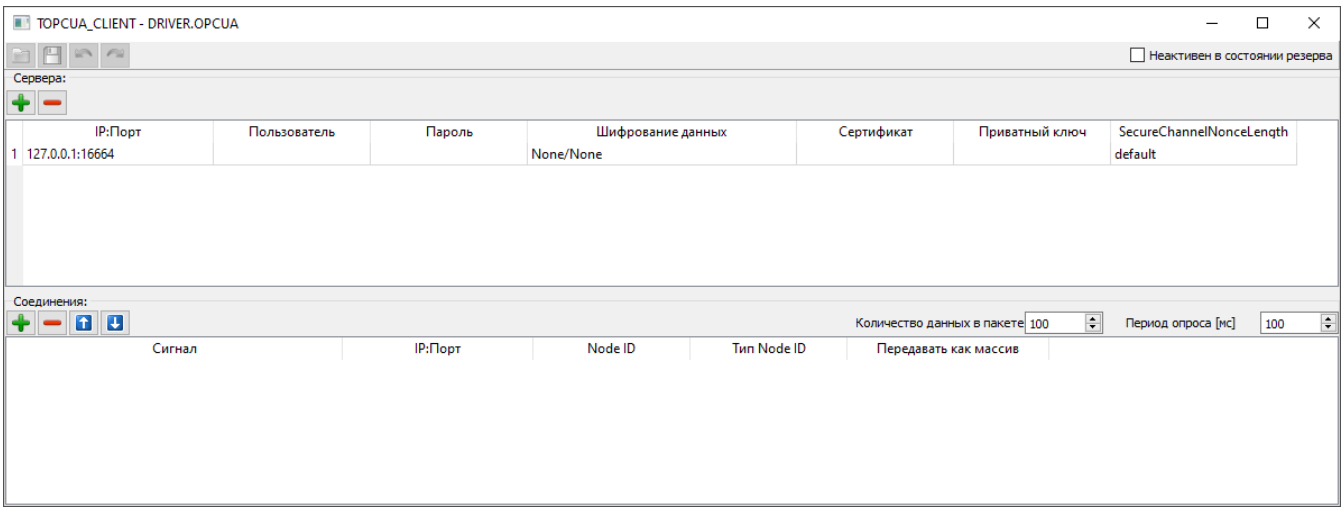


Рисунок 3.223 - Редактор приложения-драйвера OPCUA-клиент

В верхней части окна расположены следующие кнопки:

 ("Открыть") - открыть файл конфигурации драйвера;

 ("Сохранить") - сохранить выполненные настройки в файл конфигурации драйвера;

 ("Отменить") - откатить выполненные изменения на один шаг назад;

 ("Вернуть") - вернуть отменённое изменение;

 - стандартные для оконных приложений кнопки управления окном, имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

Так же в правом верхнем углу редактора находится настройка **Неактивен в состоянии резерва**. Если включить данную настройку, то приложение-драйвер OPCUA-клиент, находясь в состоянии резерва, не будет посылать запросы OPCUA-серверу. Иначе, будет посылать запросы OPCUA-серверу, даже находясь в состоянии резерва.

Так как приложение-драйвер OPCUA-клиент позволяет работать с несколькими OPCUA-серверами, в верхней таблице **Сервера** необходимо указать IP-адреса и номера портов всех серверов. По умолчанию номер порта OPCUA-сервера равен 16664 (смотрите в настройках приложения-драйвера OPCUA-сервер). Эти данные указываются в поле **IP:Port** в формате **IP-адрес:порт** (см. рис. 3.226).

В полях **Пользователь** и **Пароль** нужно настроить данные для аутентификации, которые требуются для подключения к OPCUA-серверу. Если пользователь и пароль не требуются, то данные поля не заполняются.

В поле **Шифрование данных** указывается режим передачи данных между клиентом и сервером:

- **None/None** - без использования шифрования/без использования шифрования;
- **Basic128Rsa15Sign** - с шифрованием RSA128/защита сообщений цифровой подписью;
- **Basic128Rsa15/SignAndEncrypt** - с шифрованием RSA128/защита сообщений цифровой подписью и шифрованием;
- **Basic256/Sign** - с шифрованием Basic256/защита сообщений цифровой подписью;
- **Basic256/SignAndEncrypt** - с шифрованием Basic256/защита сообщений цифровой подписью и шифрованием;
- **Basic256Sha256/Sign** - с шифрованием Basic256Sha256/защита сообщений цифровой подписью;
- **Basic256Sha256/SignAndEncrypt** - с шифрованием Basic256Sha256/защита сообщений цифровой подписью и шифрованием;
- **Aes128Sha256RsaOaep/Sign** - с шифрованием Aes128Sha256RsaOaep/защита сообщений цифровой подписью и шифрованием;
- **Aes128Sha256RsaOaep/SignAndEncrypt** - с шифрованием Aes128Sha256RsaOaep/защита сообщений цифровой подписью и шифрованием.

Если в поле **Шифрование данных** выбрать один из режимов шифрования, кроме None/None, то станут доступны и необходимо будет настроить следующие два поля: **Сертификат** и **Приватный ключ**:

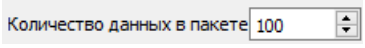
- **Сертификат** - по нажатию на данное поле откроется стандартное окно операционной системы для выбора файла. В данном случае нужно выбрать файл сертификата безопасности клиента (формат *.der). После того как файл будет выбран, он будет скопирован в папку Design проекта;
- **Приватный ключ** - по нажатию на данное поле откроется стандартное окно операционной системы для выбора файла. В данном случае нужно выбрать файл приватного ключа безопасности клиента (формат *.der, *.key, *.pem). После того как файл будет выбран, он будет скопирован в папку Design проекта;

- **SecureChannelNonceLength** - длина токена Nonce. Значение default использует настройки по умолчанию из протокола. С данными значениями работают большинство устройств.

ВНИМАНИЕ! При создании сертификата и приватного ключа для клиента OPCUA SCADA-системы "СОНАТА", который будет подключаться к серверу OPCUA SCADA-системы "СОНАТА" необходимо обязательно настроить поле URL = urn:Scada.Sonata.OPCUA.Server. Для работы со сторонними серверами этого делать не нужно.

В нижней таблице **Соединения** указываются соответствия между сигналами SCADA-системы "СОНАТА" (поле **Сигнал**), которые были добавлены ранее(см. ???), и сигналами OPCUA-сервера, которые нужно смотреть в документации на устройство (эти данные у клиента и сервера должны быть идентичны).

В правой части над таблицей **Соединения** располагаются настройки, связанные с передачей данных:



- максимальное количество сигналов, которое можно передать одним пакетом данных;

- минимальный интервал времени между обменом данными с сервером.

В верхней части таблицы **Соединения** располагаются следующие кнопки и настройки:

 - добавить строку в таблицу;

 - удалить выбранные записи из таблицы;

 - переместить строку в таблице на одну позицию вверх;

 - переместить строку в таблице на одну позицию вниз;

- таймаут между циклами опроса. OPCUA клиент работает в циклическом режиме. В каждом цикле клиент производит чтение и запись изменённых данных на сервер.

В поле **Сигнал** указывается сигнал SCADA-системы "СОНАТА".

Поле **IP:Port** используется для выбора узла, указанного в верхней таблице.

Поле **Node ID** - это сигнал OPCUA-сервера, который указывается в формате **namespace:variableID**. По умолчанию **namespace**=2, т.е., если не указать **namespace**, его значение будет считаться равным 2. **variableID** нужно смотреть в документации на устройство. Данные в данном поле у клиента должны совпадать с данными в данном поле у сервера.

Внимание! В имени тегов OPCUA клиента и OPCUA сервера следует избегать символов '@' и ':'. Данные символы используются Сонатой в служебных целях. Соната, по возможности, обработает их корректно, но возможно существуют сценарии, когда их использование приведёт к некорректному поведению драйвера. Обратите внимание, для задания индекса пространства имён тега OPCUA (namespace index) в начале имени тега нужно, как раз, писать слитно номер индекса и двоеточие. Пример. Тег имеет имя ABCD и относится к пространству имён 2, тогда полное имя тега в таблице редактора нужно указывать как 2:ABCD.

Поле **Тип Node ID** - тип **variableID**, может быть **Numeric**, **String**, **GUID** или **Opaque**.

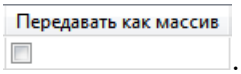
- если **Numeric**, то в качестве **VariableID** следует указать число (пример: 2:145);

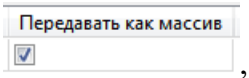
- если **GUID**, то в качестве **VariableID** значение в формате {NNNNNNNN-NNNN-NNNN-NNNN-NNNNNNNNNNNN}, где каждому N соответствует какая-то шестнадцатеричная цифра (пример: 2:1111111A-1111-AAAA-BBBB-111122223333CCCC);

- если **String**, то в качестве **VariableID** следует указать значение в формате строка (пример: 2:сигнал123);

- если **Opaque**, то в качестве **VariableID** следует указать значение в формате байтовый массив (пример: 2:12a234!/dfd123).

Поле **Передавать как массив** - данное поле позволяет настроить передачу массива данных одним пакетом.

Если сигнал SCADA-системы "СОНАТА" является массивом, то для приема/передачи данных одним пакетом необходимо выбрать именно нулевой элемент данного массива, тогда в поле **Передавать как массив** появиться активный элемент 

. Если в данном поле поставить галочку 

, то данный массив будет передаваться одним пакетом (см. рис. 3.224). Если не поставить галочку, то данный элемент (нулевой элемент массива) будет считаться скалярным и передаваться, как отдельный сигнал.

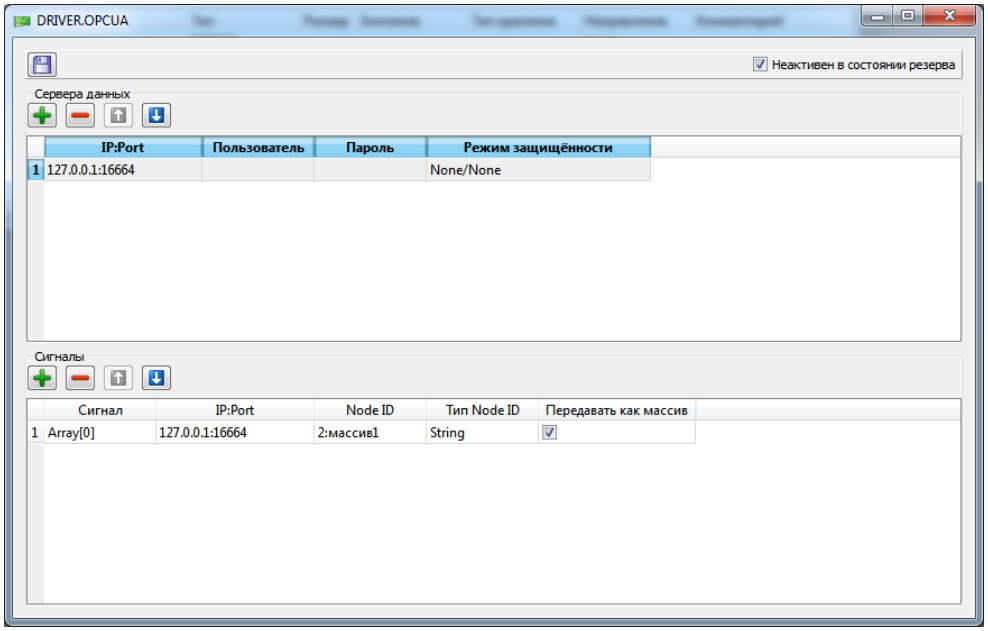


Рисунок 3.224 - Настройки передачи массива одним пакетом

Массив можно передавать либо одним пакетом (настраивается только для нулевого элемента), либо любые его элементы по отдельности (см. рис. 3.225). Одновременно два варианта передачи применять нельзя. Т.е., если добавить в редакторе несколько элементов одного массива, то поле **Передавать как массив** станет неактивным. Так же, если уже настроена передача массива одним пакетом, то редактор подсветит красным все ненулевые элементы данного массива и при сохранении сообщит об ошибке в настройках.

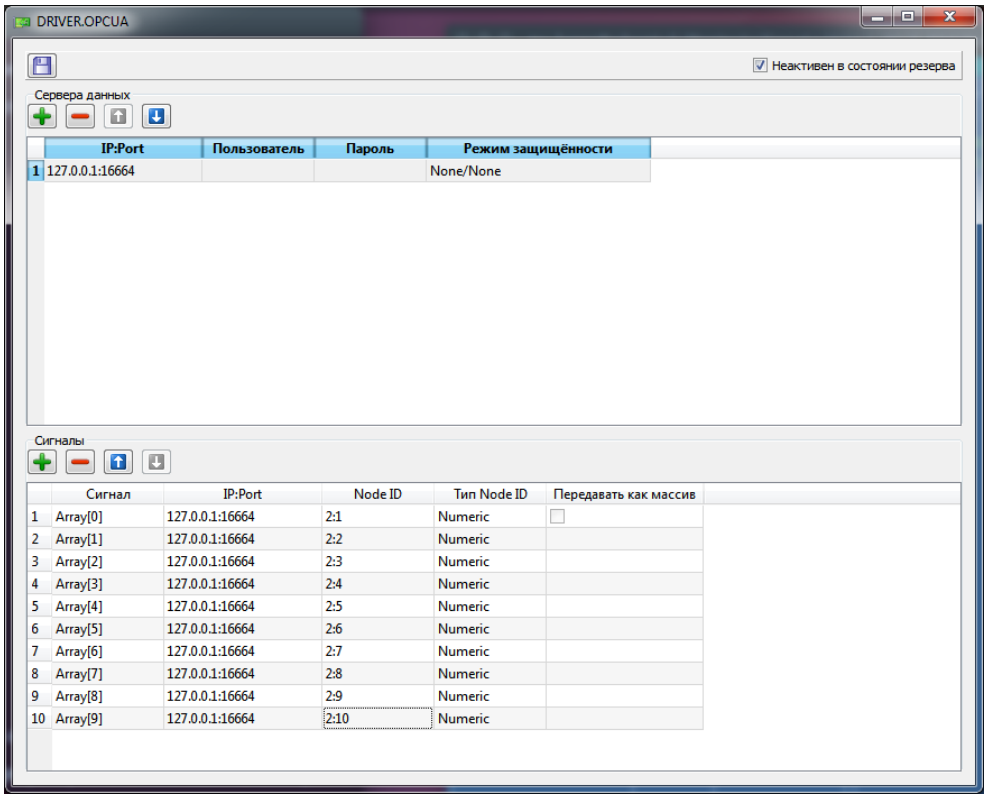


Рисунок 3.225 - Настройки для передачи элементов массива по отдельности

На рис. 3.226 представлен простой вариант настроенного приложения-драйвера OPCUA-клиент.

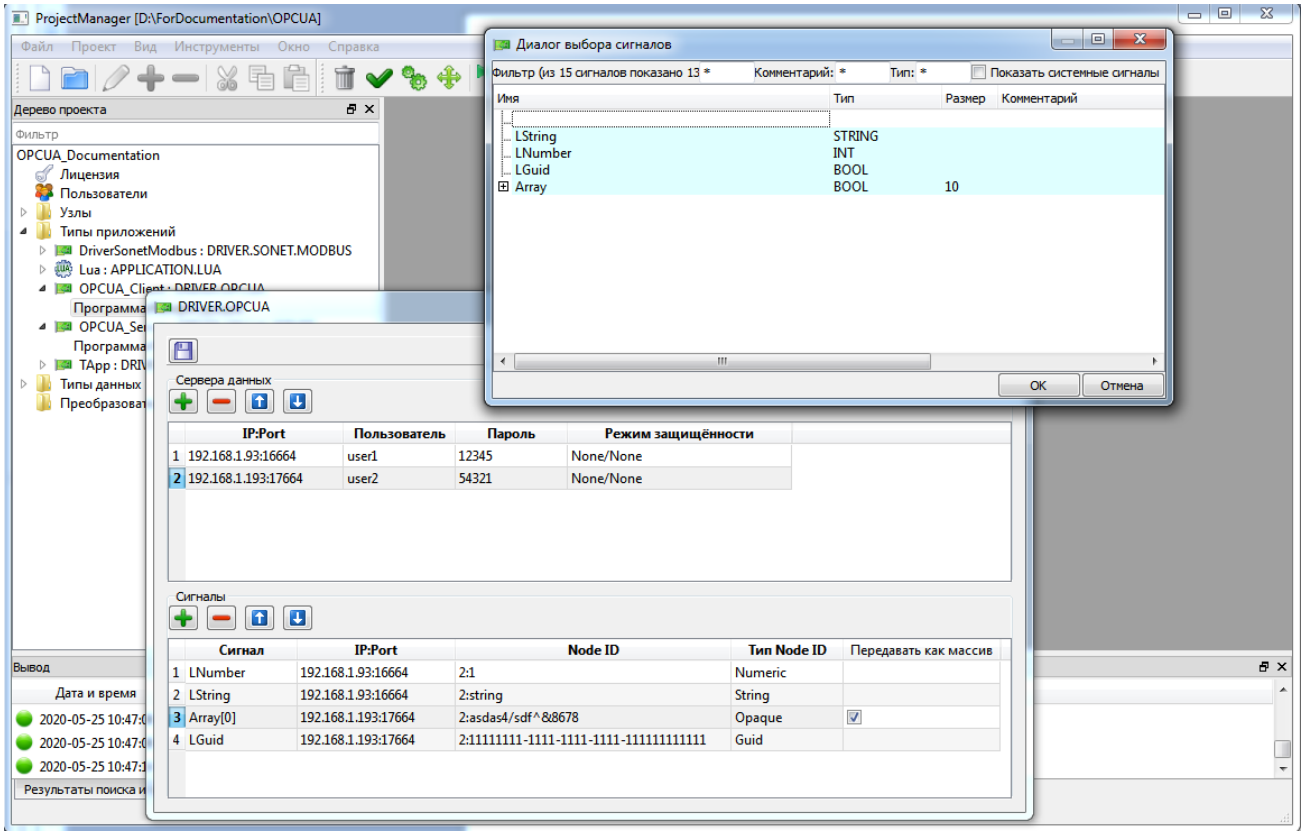


Рисунок 3.226 - Настроенное приложение-драйвер OPCUA-клиент

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер OPCUA-клиент настроено и готово к работе. Для использования

данного приложения в проекте, его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1).

3.3.8.4. Создание приложения-драйвера OPCUA-сервер

Приложение-драйвер OPCUA-сервер применяется для предоставления доступа к сигналам SCADA-системы "СОНАТА" для сторонних программ OPCUA-клиентов.

Создание приложения-драйвера OPCUA-сервер осуществляется аналогично другим приложениям SCADA-системы "СОНАТА" (см. раздел 3.2.2.4).

Далее описан упрощенный порядок действий при создании приложения-драйвера OPCUA-сервер и подробно описаны важные моменты настройки данного приложения-драйвера:

1. В программе **ProjectManager** необходимо создать новый проект (см. раздел 3.2.2.1).

2. Далее необходимо создать новое приложение (см. раздел 3.2.2.4) и провести настройки его интерфейса (см. рис. 3.227). Для этого в поле **Вид приложения** необходимо выбрать **DRIVER.OPCUA.SERVER**, поменять в поле **Имя типа** название приложения на удобное для дальнейшей работы и провести добавление локальных и глобальных сигналов SCADA-системы "СОНАТА" в интерфейс приложения (см. раздел 3.2.2.4.3.2). На рис. 3.227 добавлены локальный сигнал (массив) **S_Array** и глобальные сигналы **GSTRING**, **NUMBER** и **Global**.

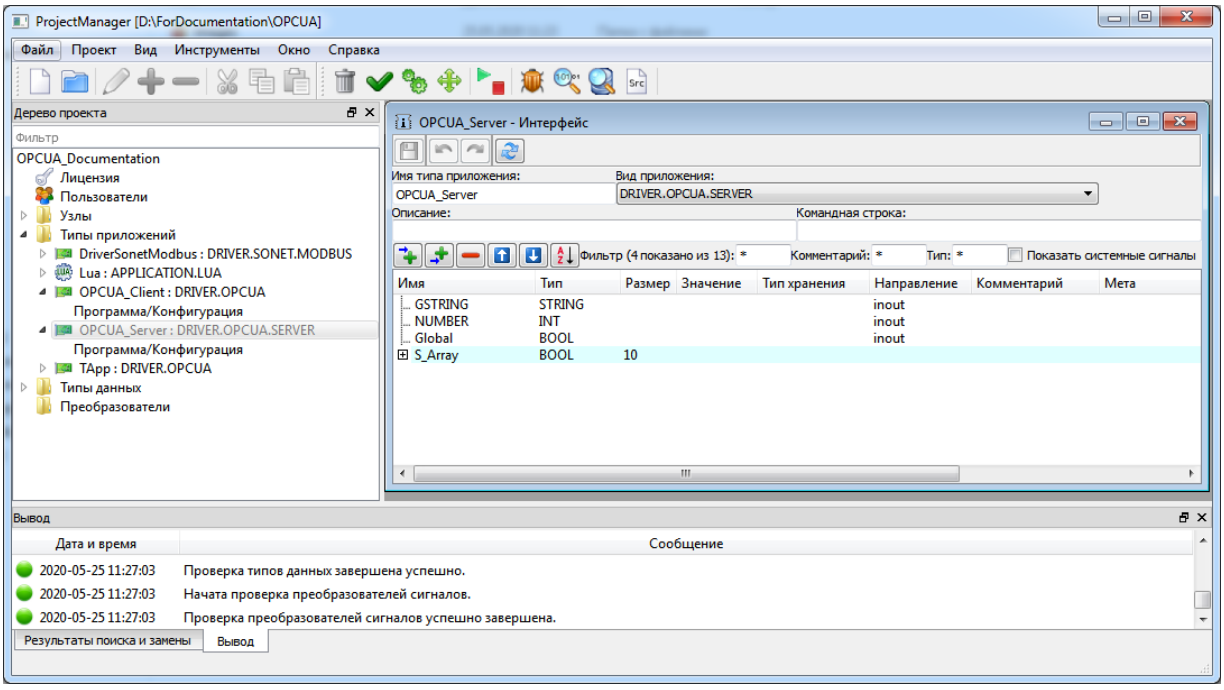


Рисунок 3.227 - Интерфейс приложения-драйвера OPCUA-сервер

3. Далее необходимо запустить редактор приложения-драйвера (см. раздел 3.2.2.4.4) и настроить таблицу кросс-коннекта. На рис. 3.228 изображено настроенное окно редактора.

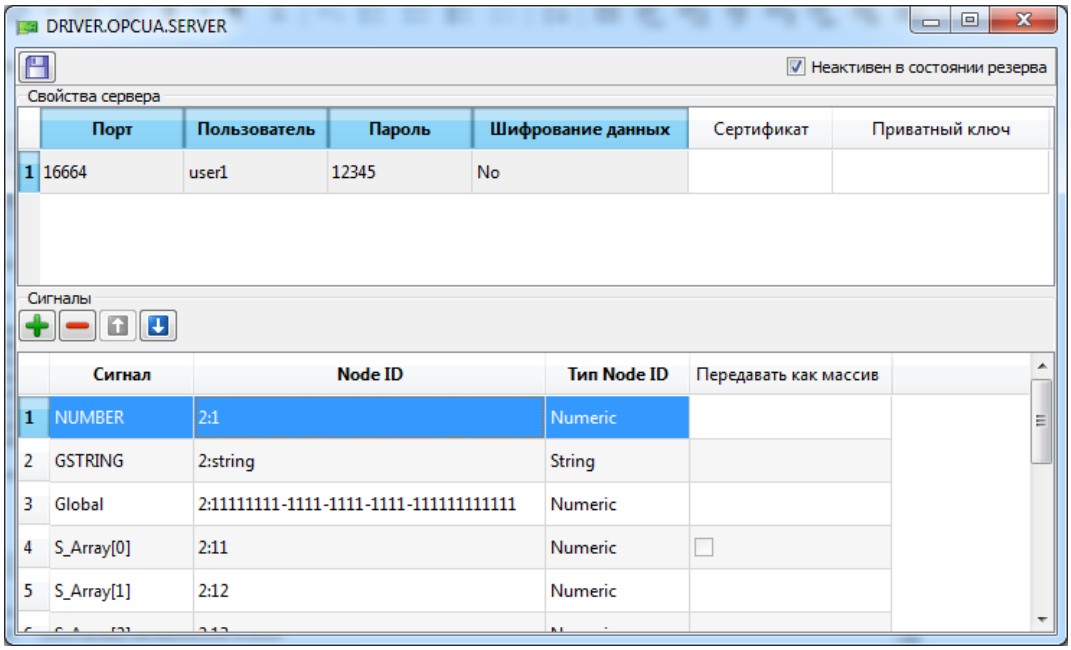


Рисунок 3.228 - Окно редактора приложения-драйвера OPCUA-сервер

В верхней части окна расположена кнопка  ("Сохранить") и стандартные для оконных приложений кнопки управления окном  , имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

В правом верхнем углу редактора находится настройка **Неактивен в состоянии резерва**. Если включить данную настройку, то приложение-драйвер OPCUA-сервер, находясь в состоянии резерва, не будет принимать запросы от OPCUA-клиентов. Иначе, будет принимать запросы от OPCUA-клиентов, даже находясь в состоянии резерва.

Ниже располагается таблица **Свойства сервера**, отвечающая за настройки параметров подключения к OPCUA-серверу. Таблица состоит из следующих столбцов:

- **Порт** - порт, по которому происходят подключения (по умолчанию 16664);
- **Пользователь** - имя пользователя для подключения к серверу;
- **Пароль** - пароль для данного пользователя;
- **Шифрование данных** - в данном поле необходимо указать: **Yes** - применять шифрование данных, **No** - не применять шифрование данных. В случае значения **Yes** станут доступны и необходимо будет настроить следующие два поля: **Сертификат** и **Приватный ключ**;
- **Сертификат** - по нажатию на данное поле откроется стандартное окно операционной системы для выбора файла. В данном случае нужно выбрать файл сертификата безопасности сервера (формат *.der). После того как файл будет выбран, он будет скопирован в папку Design проекта;
- **Приватный ключ** - по нажатию на данное поле откроется стандартное окно операционной системы для выбора файла. В данном случае нужно выбрать файл приватного ключа безопасности сервера (формат *.der, *.key, *.pem). После того как файл будет выбран, он будет скопирован в папку Design проекта.

ВНИМАНИЕ! При создании сертификата и приватного ключа для сервера OPCUA SCADA-системы "СОНАТА" необходимо обязательно настроить поле URL = urn:Scada.Sonata.OPCUA.Server.

В таблице **Сигналы** указываются соответствия между сигналами SCADA-системы "СОНАТА" и сигналами OPCUA-сервера, которые будут доступны для сторонних приложений OPCUA-клиентов.

Таблица **Сигналы** состоит из следующих столбцов:

- **Сигнал** - сигнал SCADA-системы "СОНАТА", который был ранее добавлен в интерфейс приложения-драйвера (см. рис. 3.227);

- **Node ID** - имя сигнала для OPCUA-сервера или имя сигнала, которое будет доступно сторонним приложениям OPCUA-клиентам, указывается в формате **namespace:variableID**, по умолчанию **namespace** = 2, если не указать **namespace**, то его значение будет считаться равным 2, формат записи **VariableID** зависит от **Node ID Type**;

- Поле **Тип Node ID** - тип данных для сигналов OPCUA-сервера. Поддерживаются следующие типы:

  - **Numeric** - числовой, в качестве **VariableID** следует указать число (пример: 2:1);

  - **String** - строка, в качестве **VariableID** следует указать строку символов (пример: 2:строка №1);

  - **Guid** - в качестве **VariableID** следует указать значение в формате {NNNNNNNN-NNNN-NNNN-NNNN-NNNNNNNNNNNN}, где N - шестнадцатеричная цифра (пример: 2:1111111A-BAAC-1111-1221-111122223333CCCC);

  - **Opaque** - в качестве **VariableID** следует указать значение в формате байтовый массив (пример: 2:asdas4/sdf^&8678).

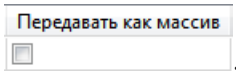
Важно

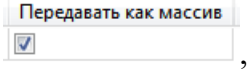
Для OPCUA-сервера SCADA-системы "Соната" принято значение **namespace** = 2. Другие значения в данный момент не используются.

Важно

В имени тегов (столбец Node ID) OPCUA клиента и OPCUA сервера следует избегать символов '@' и ':'. Данные символы используются Сонатой в служебных целях. Соната, по возможности, обработает их корректно, но возможно существуют сценарии, когда их использование приведёт к некорректному поведению драйвера. Обратите внимание, для задания индекса пространства имён тега OPCUA (namespace index) в начале имени тега нужно, как раз, писать слитно номер индекса и двоеточие. Пример. Тег имеет имя ABCD и относится к пространству имён 2, тогда полное имя тега в таблице редактора нужно указывать как 2:ABCD.

Поле **Передавать как массив** - данное поле позволяет настроить передачу массива данных одним пакетом.

Если сигнал SCADA-системы "СОНАТА" является массивом, то для приема/передачи данных одним пакетом необходимо выбрать именно нулевой элемент данного массива, тогда в поле **Передавать как массив** появиться активный элемент 

. Если в данном поле поставить галочку 

, то данный массив будет передаваться одним пакетом (см. рис. 3.228). Если не поставить галочку, то данный элемент (нулевой элемент массива) будет считаться скалярным и передаваться, как отдельный сигнал.

Массив можно передавать либо одним пакетом (настраивается только для нулевого элемента), либо любые его элементы по отдельности (см. рис. 3.229). Одновременно два варианта передачи применять нельзя. Т.е., если добавить в редакторе несколько элементов одного массива, то поле **Передавать как массив** станет неактивным. Так же, если уже настроена передача массива одним пакетом, то редактор подсветит красным все ненулевые элементы данного массива и при сохранении сообщит об ошибке в настройках.

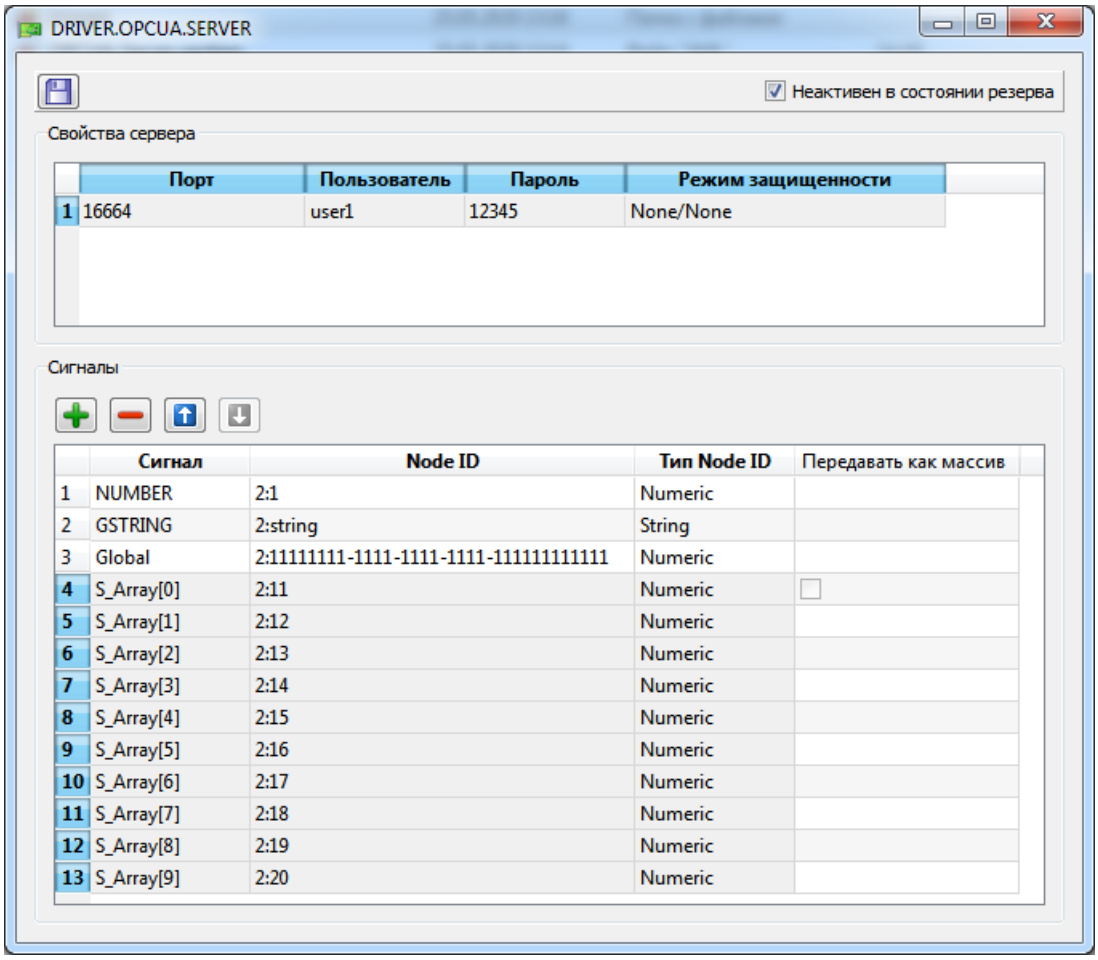


Рисунок 3.229 - Настройка для передачи элементов массива по отдельности

После настроек в редакторе необходимо нажать кнопку ("Сохранить") и закрыть окно редактора. Приложение-драйвер OPCUA-сервер настроено и готово к работе. Для использования данного приложения в проекте, его необходимо добавить на один из узлов проекта (см. раздел 3.2.2.5.1). Сигналы SCADA-системы "СОНАТА" будут видны для сторонних приложений OPCUA-клиентов под именами, указанными в столбце **Node ID**.

3.3.8.5. Пример настройки приложений OPCUA-сервер и OPCUA-клиент

В данном разделе будет разобран пример настройки приложений OPCUA-сервер и OPCUA-клиент.

На рис. 3.230 изображена схема, которая будет разобрана в примере. Данная схема отображает основные особенности работы OPCUA-сервера и OPCUA-клиента в SCADA-системе "СОНАТА".

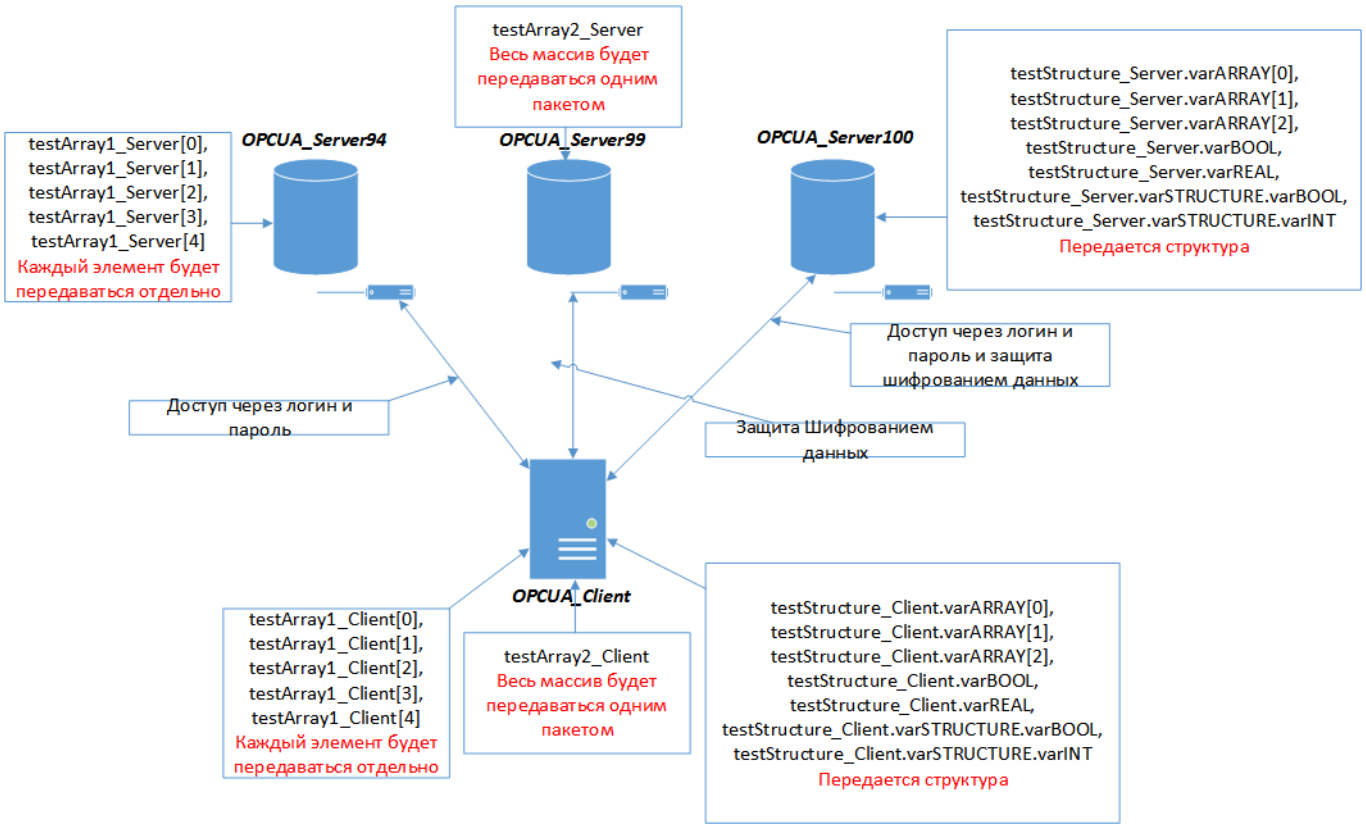


Рисунок 3.230 - Общая схема примера

В нашем примере OPCUA-клиент будет обмениваться данными с тремя OPCUA-серверами: OPCUA_Server94, OPCUA_Server99 и OPCUA_Server100. С каждым сервером защита соединения будет организована по разному:

- к OPCUA_Server94 подключение будет происходить по настроенному логину и паролю;

- с OPCUA_Server99 будет настроена защита с помощью шифрования данных;

- с OPCUA_Server100 будут настроены подключение по логину и паролю и защита с помощью шифрования данных.

В нашем примере будут передаваться следующие данные (обмен двухсторонний):

- массив testArray1_Server, состоящий из пяти элементов, будет обмениваться данными поэлементно с массивом из пяти элементов testArray1_Client (каждый элемент передается отдельно);

- массив testArray2_Server, состоящий из пяти элементов, будет передаваться одним пакетом в массив из пяти элементов testArray2_Client (при изменении любого элемента будет передаваться весь массив);

- структура testStructure_Server, состоящая из массива varArray, поля varBOOL, поля varREAL и вложенной структуры varSTRUCTURE, будет обмениваться данными с аналогичной структурой testStructure_Client (поля структуры и вложенной структуры передаются как отдельные элементы, поле-массив можно передавать как одним пакетом или как отдельные элементы).

1. Создадим проект, который будет содержать четыре узла (см. рис. 3.231):

- OPCUA_Client - на данном узле будет работать приложение OPCUA-клиент (OPCUA_Client : DRIVER.OPCUA);

- OPCUA_Server94 - на данном узле будет работать приложение OPCUA-сервер (OPCUA_Server94 : DRIVER.OPCUA.SERVER) с настройкой доступа по логину и паролю;

- OPCUA_Server99 - на данном узле будет работать приложение OPCUA-сервер (OPCUA_Server99 : DRIVER.OPCUA.SERVER) с использованием шифрования данных;

- OPCUA_Server100 - на данном узле будет работать приложение OPCUA-сервер (OPCUA_Server100 : DRIVER.OPCUA.SERVER) с настройкой доступа по логину и паролю и с использованием шифрования данных.

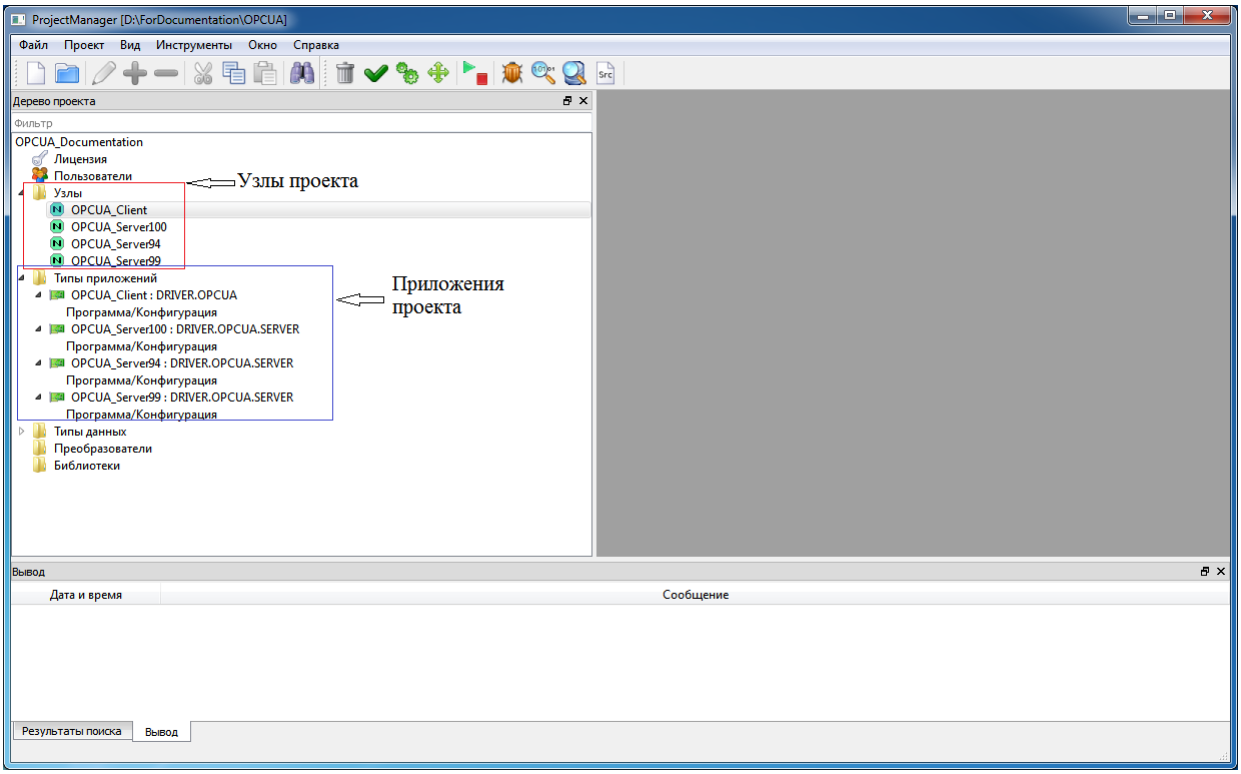
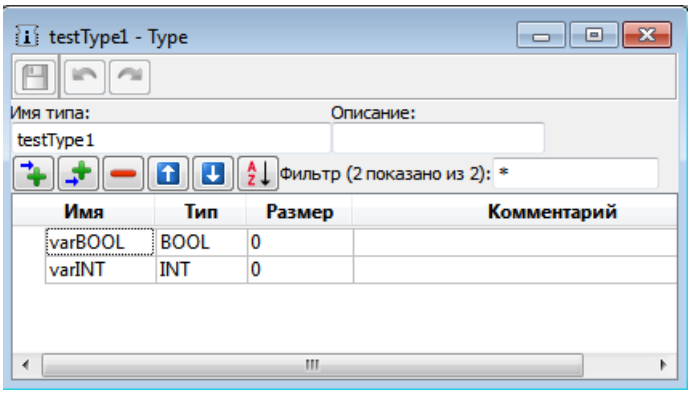
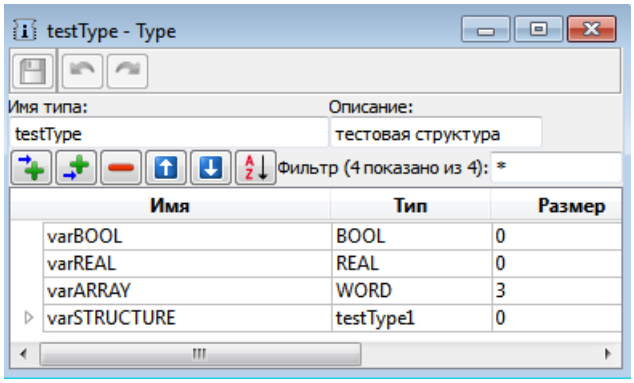


Рисунок 3.231 - Дерево проекта в Менеджере проектов

2. В проекте используются следующие структурные типы данных (они созданы для того чтобы далее показать как настраивать передачу подобных структур в приложениях OPCUA):

- testType1 

- будет одним из полей другого структурного типа testType;

- testType 

- будет использоваться в передаваемых данных.

3. Для передачи данных используются следующие сигналы (см. рис. 3.232):

- testArray1_Server - массив из пяти элементов, тип данных LREAL. Это будут данные для приложения OPCUA_Server94;

- testArray1_Client - массив из пяти элементов, тип данных LREAL. Это будут данные для приложения OPCUA_Client;

- testArray2_Server - массив из пяти элементов, тип данных DWORD. Это будут данные для приложения OPCUA_Server99;

- testArray2_Client - массив из пяти элементов, тип данных DWORD. Это будут данные для приложения OPCUA_Client;

- testStructure_Server - структура типа testType. Это будут данные для приложения OPCUA_Server100;

- testStructure_Client - структура типа testType. Это будут данные для приложения OPCUA_Client.

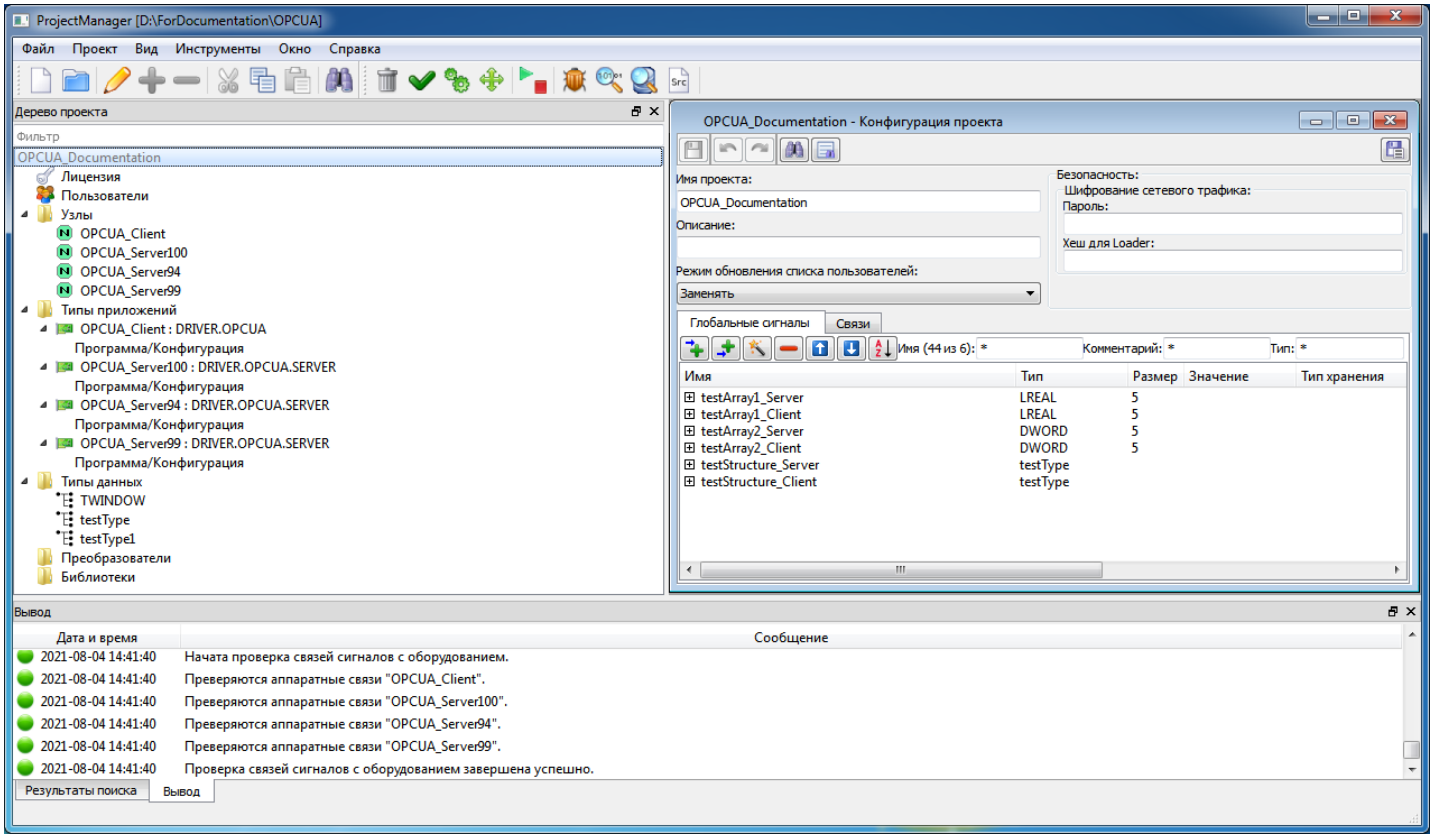


Рисунок 3.232 - Окно конфигурации проекта с глобальными сигналами проекта

4. Далее посмотрим как настроены наши приложения.

4.1 Интерфейс приложения OPCUA_Server94 представлен на рис. 3.233.

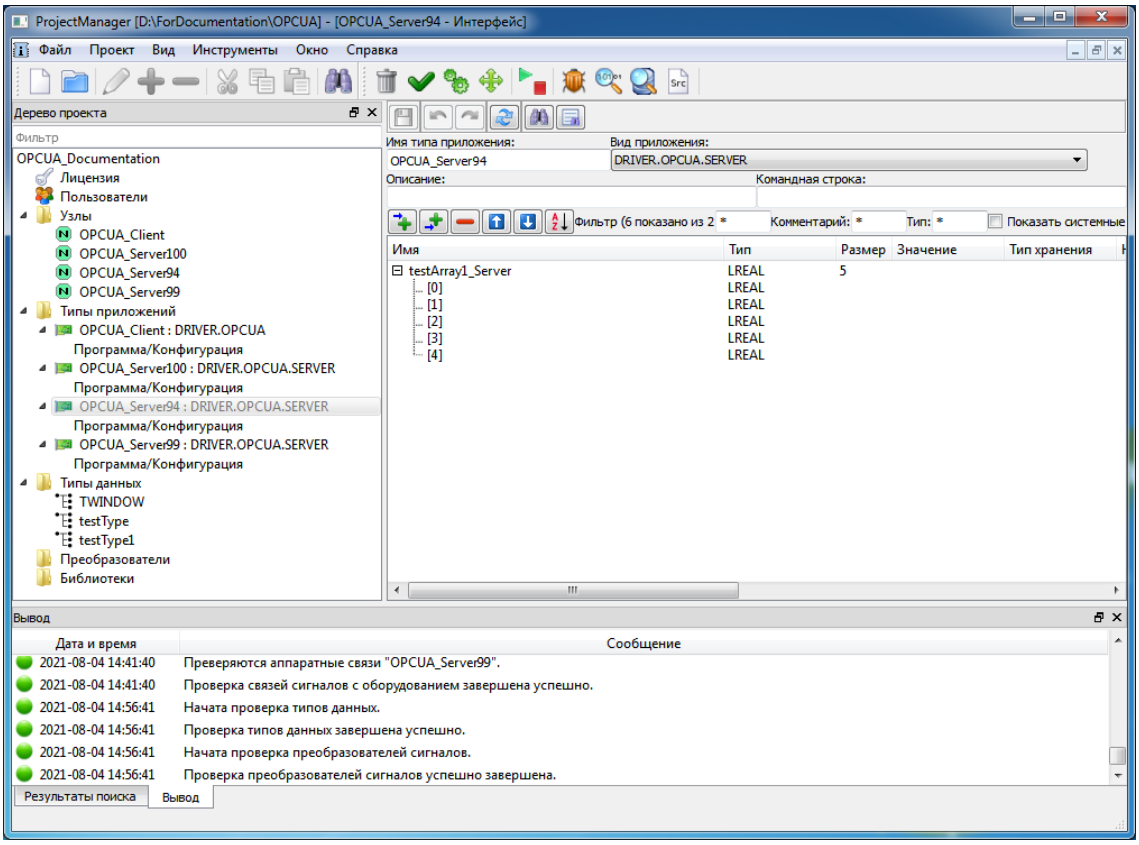


Рисунок 3.233 - Интерфейс приложения OPCUA_Server94

Настройки приложения OPCUA_Server94 представлены на рис. 3.234.

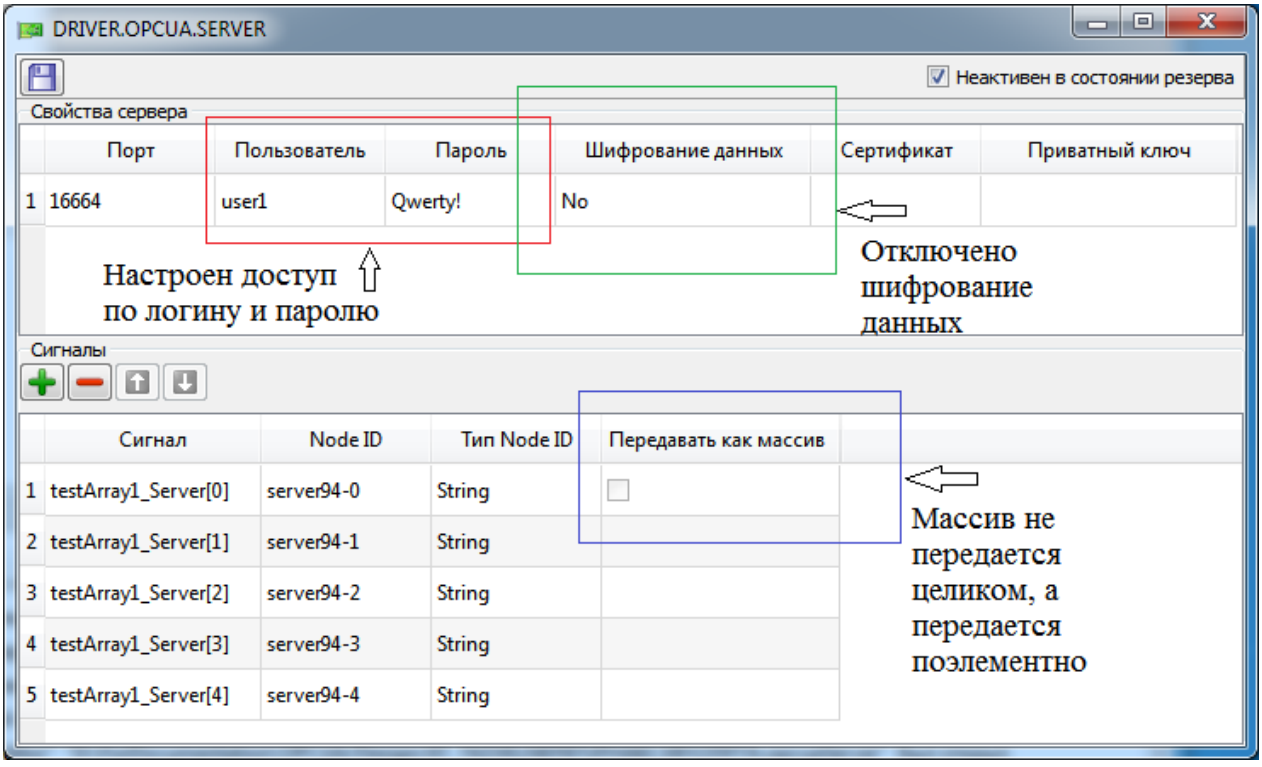


Рисунок 3.234 - Настройки приложения OPCUA_Server94

В данном приложении настроен доступ по паролю и логину, оключено шифрование данных и будет поэлементно передаваться массив данных testArray1_Server.

Обратите внимание, что для передачи массива поэлементно не нужно ставить галочку в поле "Передавать как массив" и нужно каждому элементу присвоить уникальный Node ID.

4.2 Интерфейс приложения OPCUA_Server99 представлен на рис. 3.235.

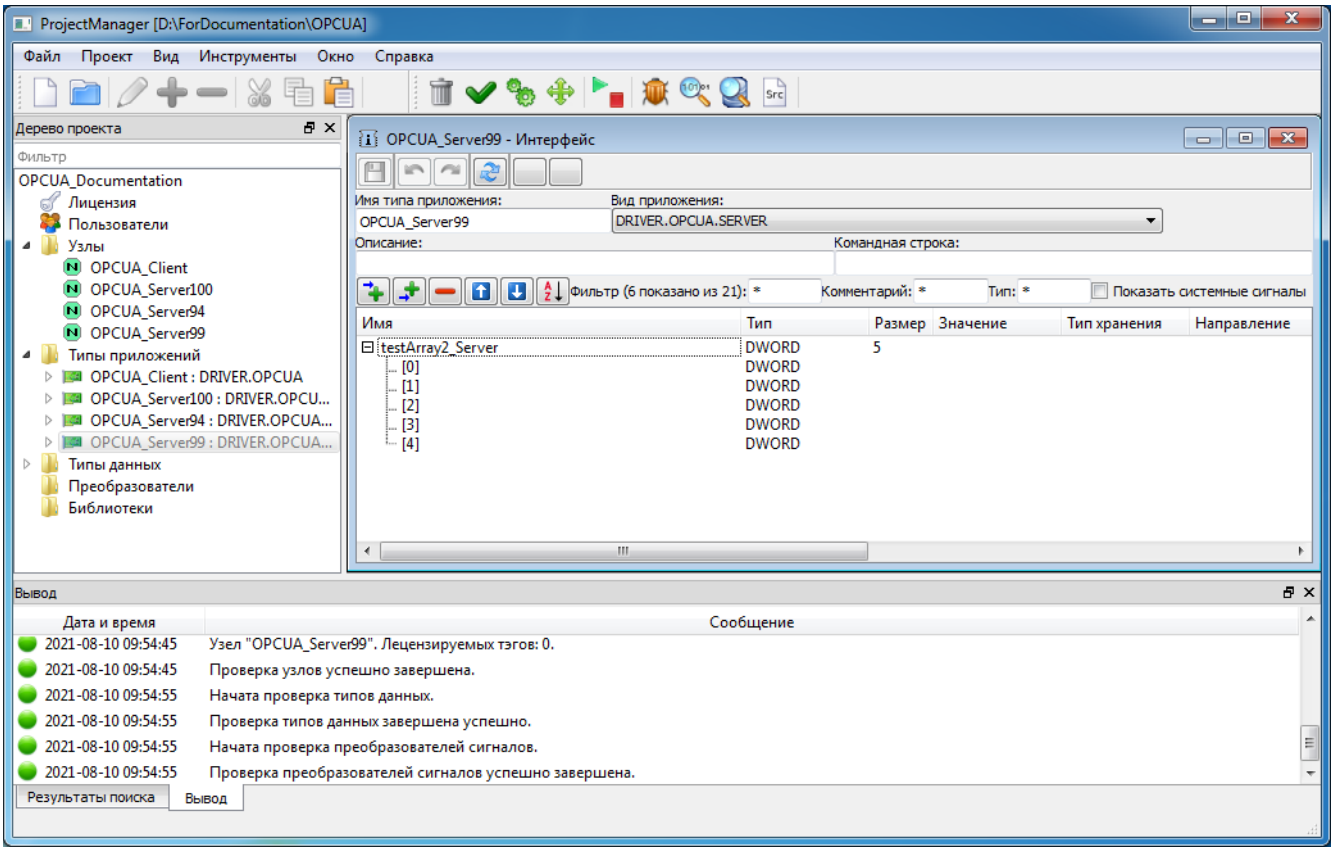


Рисунок 3.235 - Интерфейс приложения OPCUA_Server99

Настройки приложения OPCUA_Server99 представлены на рис. 3.236.

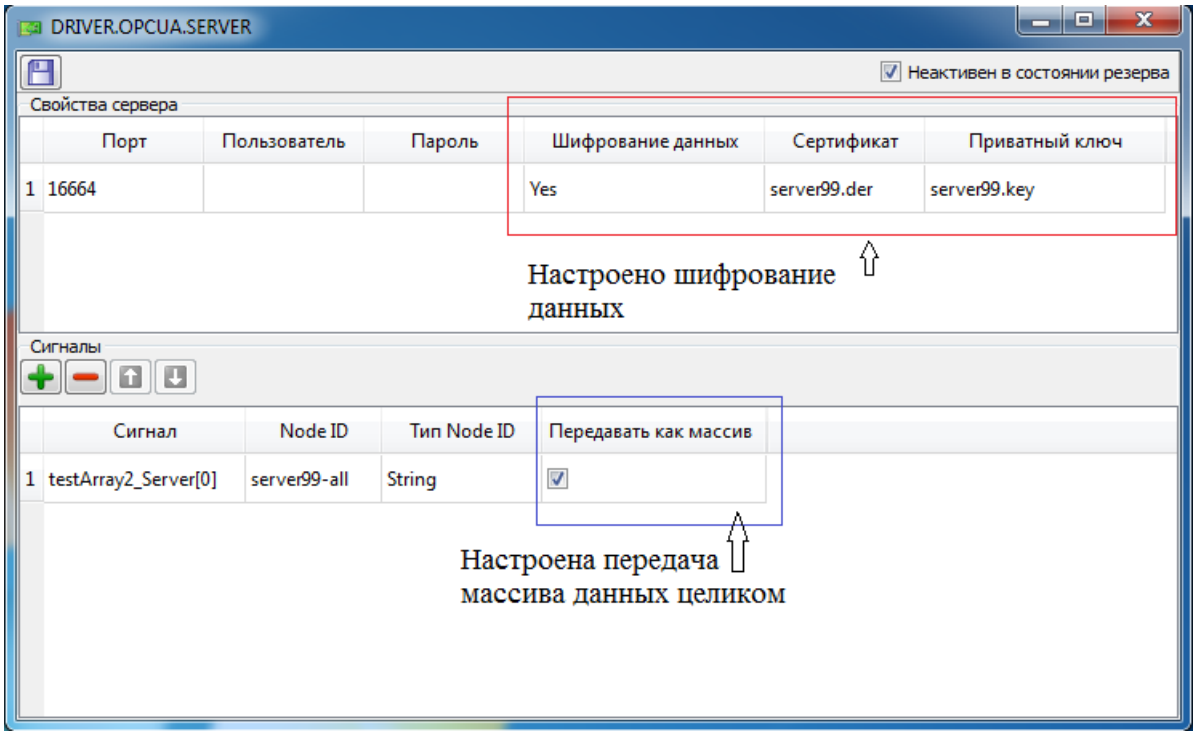


Рисунок 3.236 - Настройки приложения OPCUA_Server99

В данном приложении выполнены настройки шифрования данных при передаче (как создавать сертификат и приватный ключ описано ниже) и массив данных testArray2_Server будет передаваться целиком.

Обратите внимание, что для передачи массива данных целиком нужно поставить галочку в поле "Передавать как массив". Данная галочка доступна только для нулевого элемента массива.

4.3 Интерфейс приложения OPCUA_Server100 представлен на рис. 3.237.

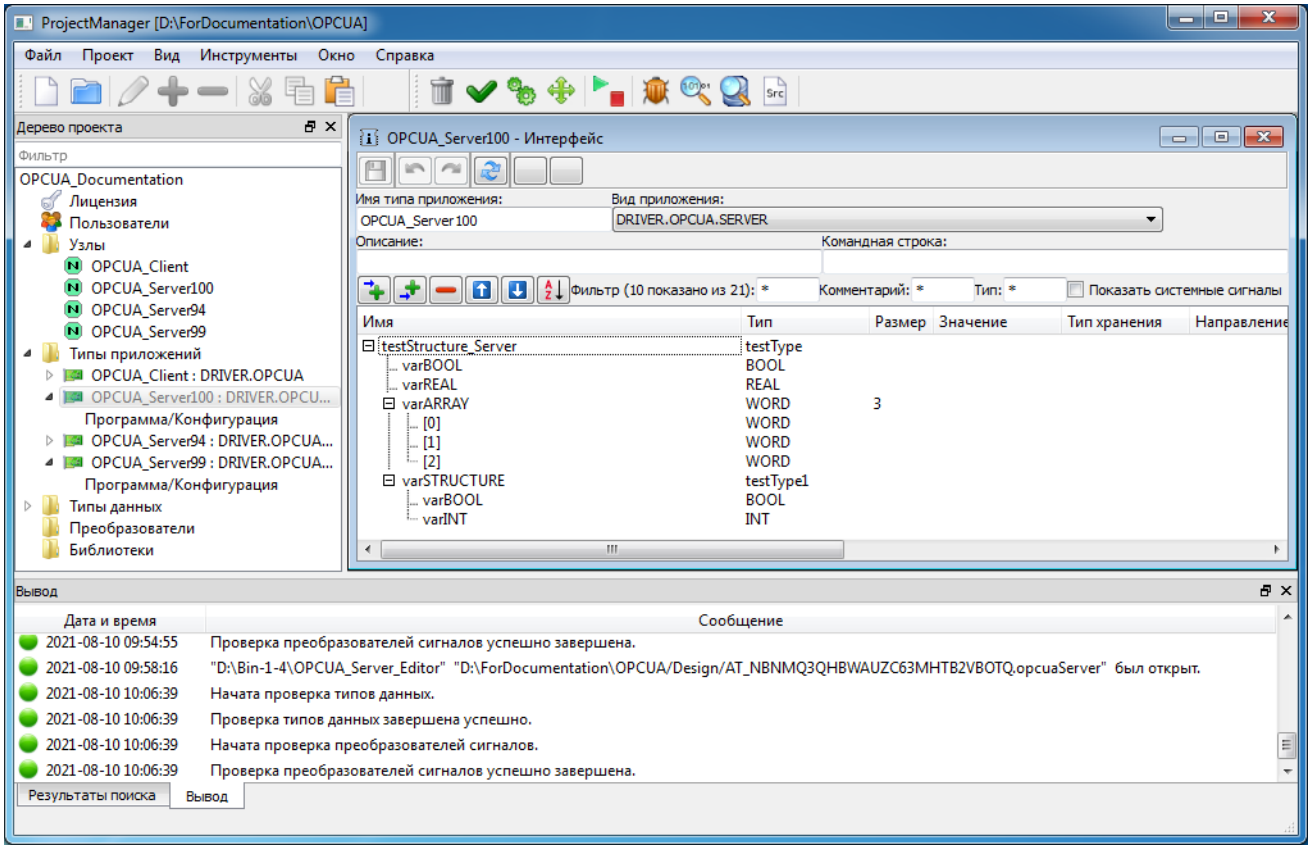


Рисунок 3.237 - Интерфейс приложения OPCUA_Server100

Настройки приложения OPCUA_Server100 представлены на рис. 3.238.

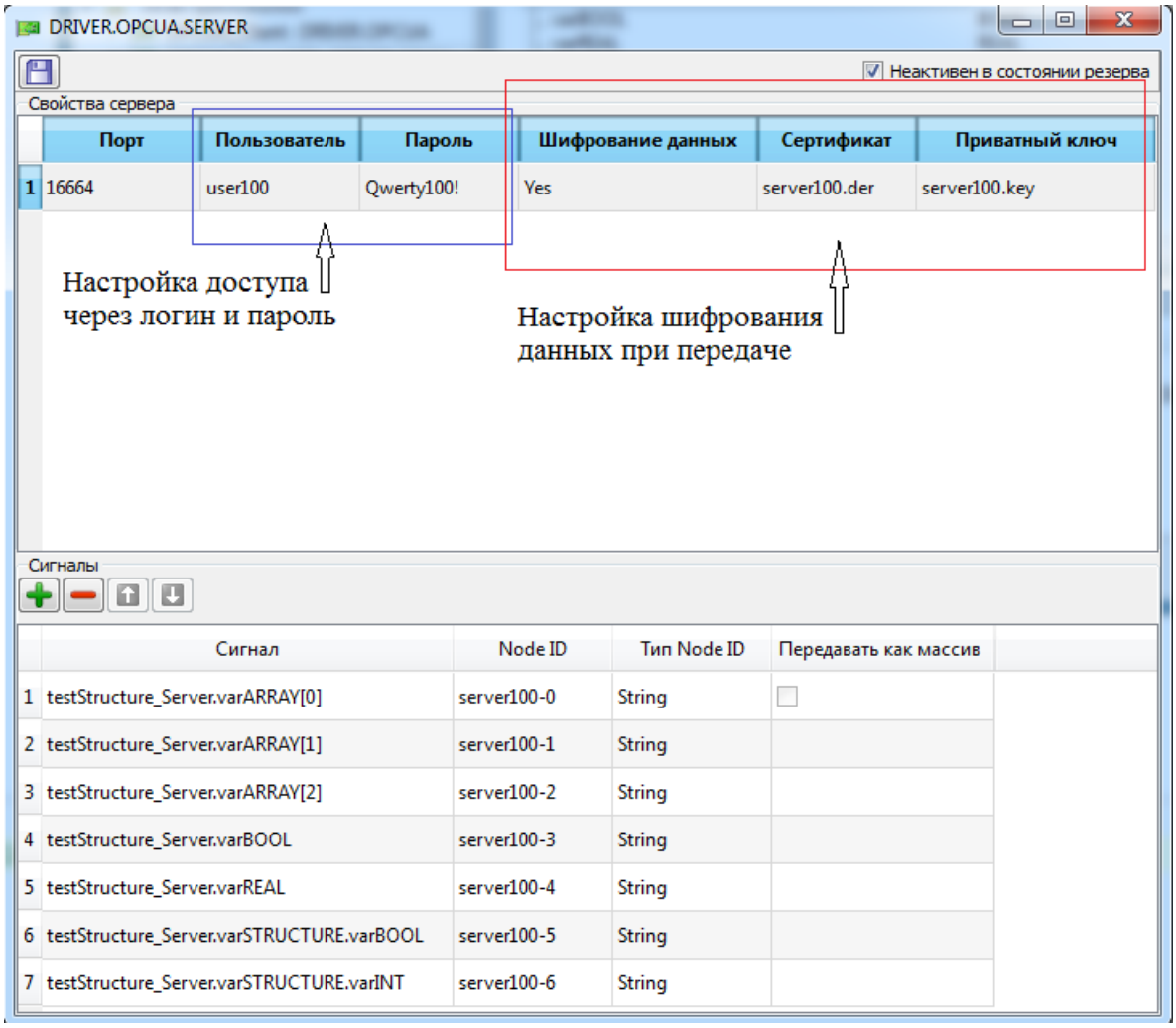


Рисунок 3.238 - Настройки приложения OPCUA_Server100

В данном приложении выполнены настройки доступа по логину и паролю и настроено шифрование данных при передаче (как создавать сертификат и приватный ключ описано ниже). В качестве передаваемых данных используется структура testStructure_Server, полями которой являются массив varARRAY, два простых поля varBOOL и varREAL и структура varSTRUCTURE, состоящая из двух простых полей varBOOL и varINT. Передача поля varARRAY, которое является массивом, настроено поэлементно, но можно настроить и передачу данного поля целиком.

4.4 Интерфейс приложения OPCUA_Client представлен на рис. 3.239.

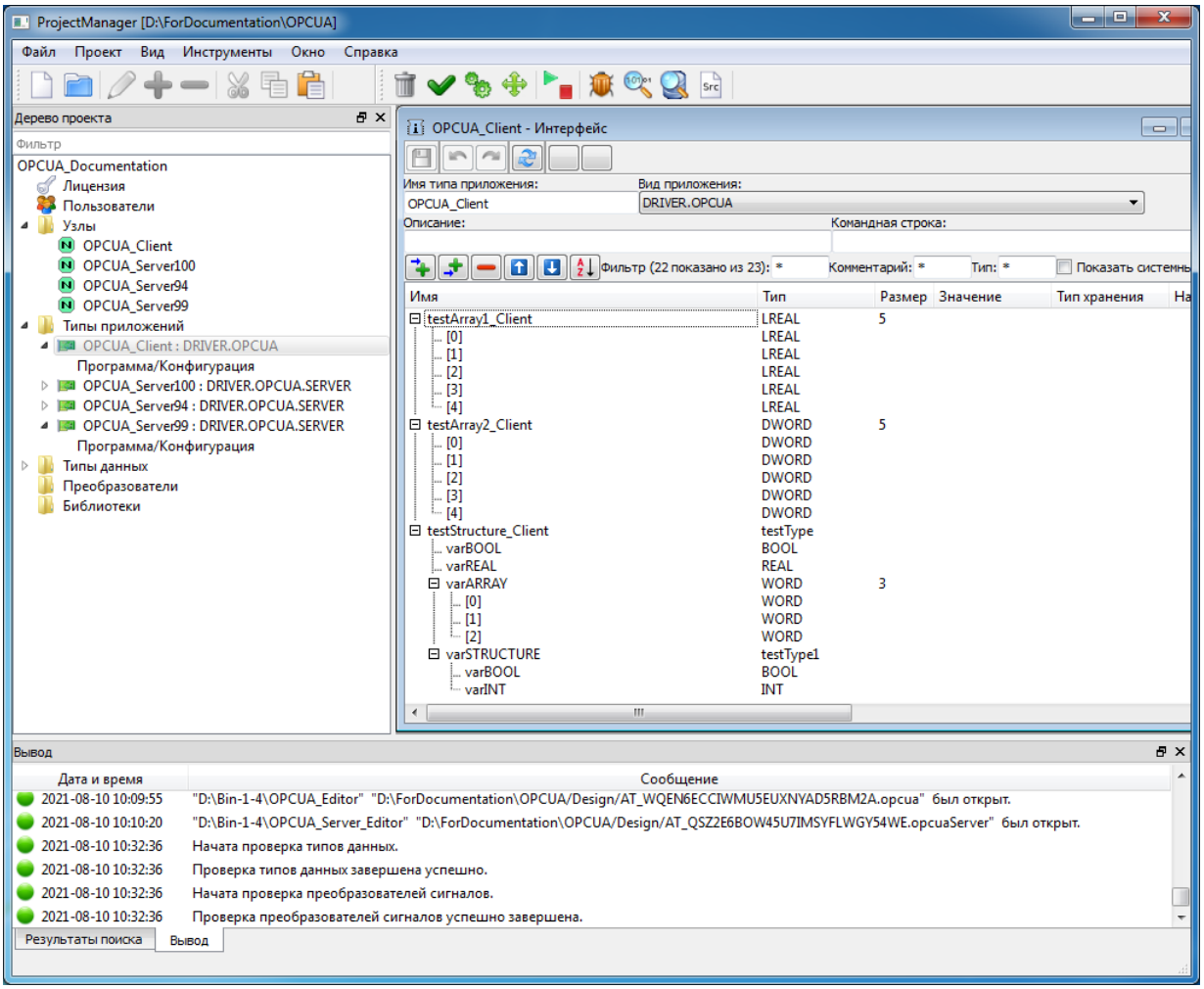


Рисунок 3.239 - Интерфейс приложения OPCUA_Client

Настройки приложения OPCUA_Client представлены на рис. 3.240.

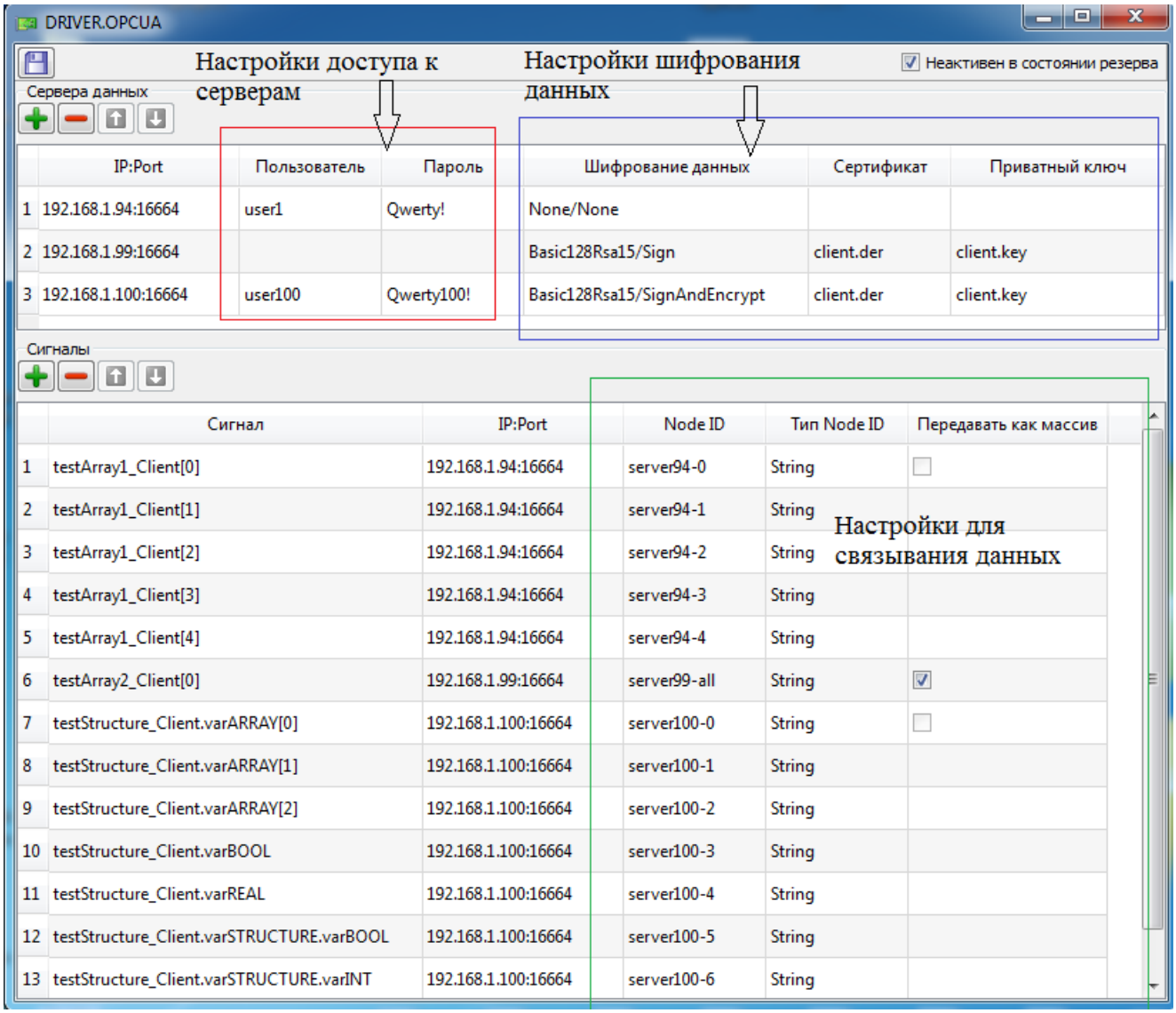


Рисунок 3.240 - Настройки приложения OPCUA_Client

В настройках OPCUA-клиента можно указать несколько OPCUA-серверов, с которыми он будет обмениваться данными. Настройки выполняются в колонке IP:Port и необходимо указать ip-адресс сервера и порт.

Далее идут настройки доступа к серверам через логин и пароль.

Далее идут настройки шифрования данных. В данных настройках у OPCUA-клиента можно выбрать вариант шифрования данных (смотри в описании OPCUA-клиента) и указать ему сертификат и приватный ключ клиента. Обратите внимание, что для настройки шифрования данных у приложения OPCUA-сервер только указывается, что он работает с шифрованием, а именно у OPCUA-клиента указывается вариант шифрования данных.

В настройках передаваемых данных (нижняя таблица) в поле IP:Port нужно правильно указать нужный OPCUA-сервер, а колонки Node ID, Тип Node ID и Передавать как масив нужно заполнить такими же данными, как и в соответствующих OPCUA-серверах.

5. Настройка шифрования данных

5.1 Общая информация

В нашем OPCUA используется ассиметричное шифрование, это означает, что для шифрования данных используется один ключ, а для расшифровки - второй ключ. Эти ключи называются: приватный и публичный ключ.

Как можно понять из названия, публичный ключ не является секретным. Он свободно распространяется и используется для шифрования данных, которые можно расшифровать только приватным ключом.

Публичный и приватный ключ генерируются вмести и криптографически связаны.

Ещё данная пара ключей может использоваться для подписи данных и проверки подписи. Эта подпись подтверждает то, что данные удостоверены владельцем приватного ключа и в последствии эти данные не были изменены. Подписываются данные приватным ключом (которые имеет одно определённое лицо), а проверить подпись можно публичным ключом, который может получить каждый.

Любой может сгенерировать пару ключей, поэтому возникает проблема идентификации — как проверить, что публичный ключ выпущен определённым лицом?

Это можно было бы сделать например так: владелец сайт mysite.org генерирует пару публичный-приватный ключ и просит третью сторону подписать его публичный ключ. В результате публичный ключ распространяется с цифровой подписью, которую можно проверить публичным ключом третьей стороны. На самом деле, всё именно так и происходит, а подписанный публичный ключ, вместе с дополнительной информацией (например, название домена, для которого он подписан) упаковываются в сертификат.

Сертификат, по сути, это публичный ключ, а также информация о домене и другая сопутствующая информация, подписанная электронной подписью.

В результате процедура создания сертификата выглядит так:

1) Владельцем генерируется пара приватный и публичный ключ;

2) Публичный ключ вместе с другой информацией для подписи (например, название доменного имени) упаковывается в файл в специальном формате. Он называется — Certificate Signing Request (CSR), то есть «запроса на подпись сертификата»;

3) Данный запрос на подпись (CSR) отправляется в Центр Сертификации (CA), который, используя свой приватный корневой ключ, создаёт подпись для этих данных и всё это упаковывается в другой специальный файл, называемый сертификат.

В результате получается сертификат со следующими свойствами:

1) Он может зашифровать данные (в нём есть публичный ключ), которые способен расшифровать только приватный ключ составляющий пару этому сертификату;

2) Сертификат может быть проверен на подлинность (у него есть цифровая подпись) с помощью сертификата Центр Сертификации (CA), который его создал.

5.2 Создание сертификата и приватного ключа при помощи openssl

Шаг 1. Создадим приватный и публичный ключи, которые будем использовать как ключи центра сертификации. Данные ключи будут храниться только у нас и с помощью них мы сможем создавать сертификаты для наших OPCUA-серверов и клиентов. Назовем их rootCA.key - корневой приватный ключ и rootCA.crt - корневой публичный ключ.

Для этого нужно выполнить две команды:

а) **openssl genrsa -out rootCA.key**

б) **openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt**

Шаг 2. Создадим приватные ключи для наших серверов, которые будут использовать шифрование, и для клиента.

Для этого нужно выполнить следующие команды:

а) **openssl genrsa -out server99.key** - для OPCUA_Server99;

б) **openssl genrsa -out server100.key** - для OPCUA_Server100;

в) **openssl genrsa -out client.key** - для OPCUA_Client.

Шаг 3. Создадим запросы (csr) на получение сертификата для наших серверов, которые будут использовать шифрование, и для клиента.

Для этого нужно выполнить следующие команды:

а) **openssl req -new -key server99.key -out server99.csr** - для OPCUA_Server99;

б) **openssl req -new -key server100.key -out server100.csr** - для OPCUA_Server100;

в) **openssl req -new -key client.key -out client.csr** - для OPCUA_Client.

Шаг 4. Создадим сертификаты на основе ранее созданных запросов (csr).

ВНИМАНИЕ! Наш OPCUA-сервер требует чтобы в сертификате обязательно было настроено поле subjectAltName, его значение должно быть URI:urn:Scada.Sonata.OPCUA.Server. Для этого необходимо создать файл 1.ext и внутри него написать следующую строку:

subjectAltName = URI:urn:Scada.Sonata.OPCUA.Server

Для создания сертификатов нужно выполнить следующие команды:

а) **openssl x509 -req -in server99.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out server99.crt -days 1024 -sha256 -extfile 1.ext** - для OPCUA_Server99;

б) **openssl x509 -req -in server100.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out server100.crt -days 1024 -sha256 -extfile 1.ext** - для OPCUA_Server100;

в) **openssl x509 -req -in client.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out client.crt -days 1024 -sha256 -extfile 1.ext** - для OPCUA_Client.

Обратите внимание на параметр -days 1024 - это значит, что сертификат будет действительным 1024 дня. Данный параметр можно выставлять таким как нужно вам.

Шаг 5. Наши OPCUA-сервер и клиент работают с сертификатами в формате .der (простой двоичный вид сертификата). А на шаге 4 мы получили сертификаты в формате .crt. Это так называемый PEM формат. И нам его нужно перекодировать.

Для перекодировки сертификатов из .crt в .der нужно выполнить следующие команды:

а) **openssl x509 -outform der -in server99.crt -out server99.der** - для OPCUA_Server99;

б) **openssl x509 -outform der -in server100.crt -out server100.der** - для OPCUA_Server100;

в) **openssl x509 -outform der -in client.crt -out client.der** - для OPCUA_Client.

В итоге мы получим набор пар сертификат плюс приватный ключ для наших серверов и клиента:

а) server99.key (приватный ключ) и server99.der (сертификат) - для OPCUA_Server99;

б) server100.key (приватный ключ) и server100.der (сертификат) - для OPCUA_Server100;

в) client.key (приватный ключ) и client.der (сертификат) - для OPCUA_Client.

6. Добавим наши приложения на узлы.

Приложение OPCUA_Server100 добавим на узел OPCUA_Server100 (см. рис. 3.241).

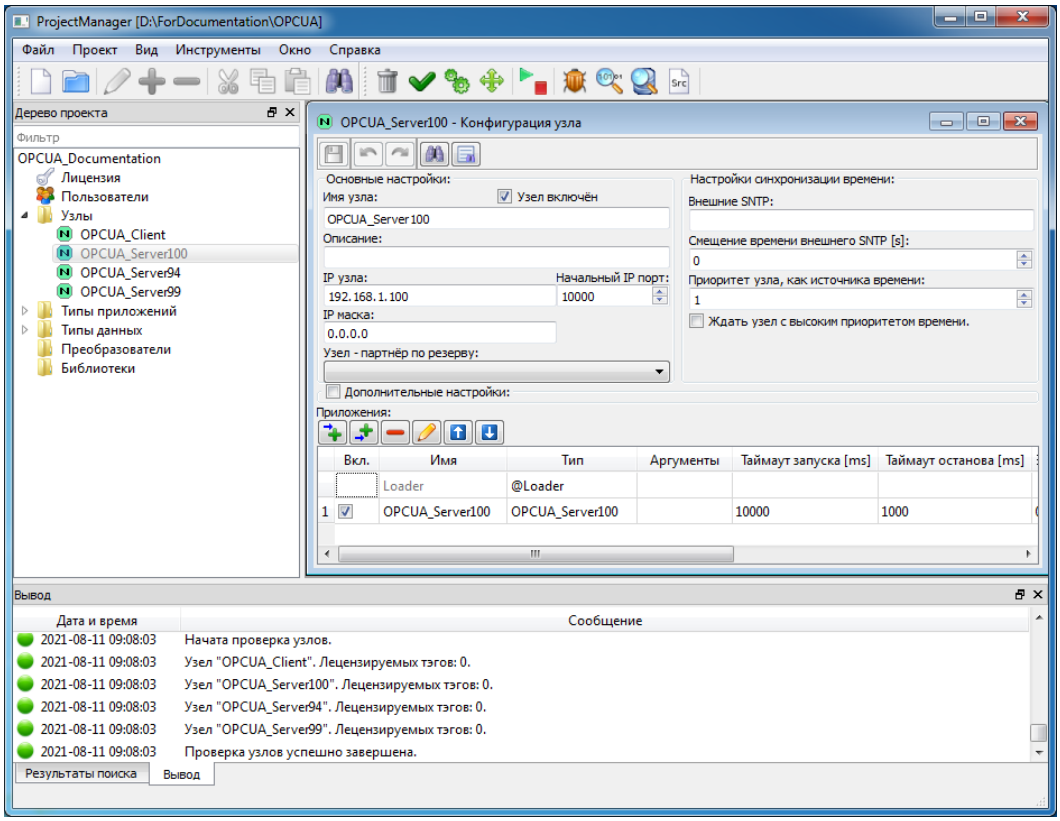


Рисунок 3.241 - Настройки узла OPCUA_Server100

Приложение OPCUA_Server99 добавим на узел OPCUA_Server99 (см. рис. 3.242).

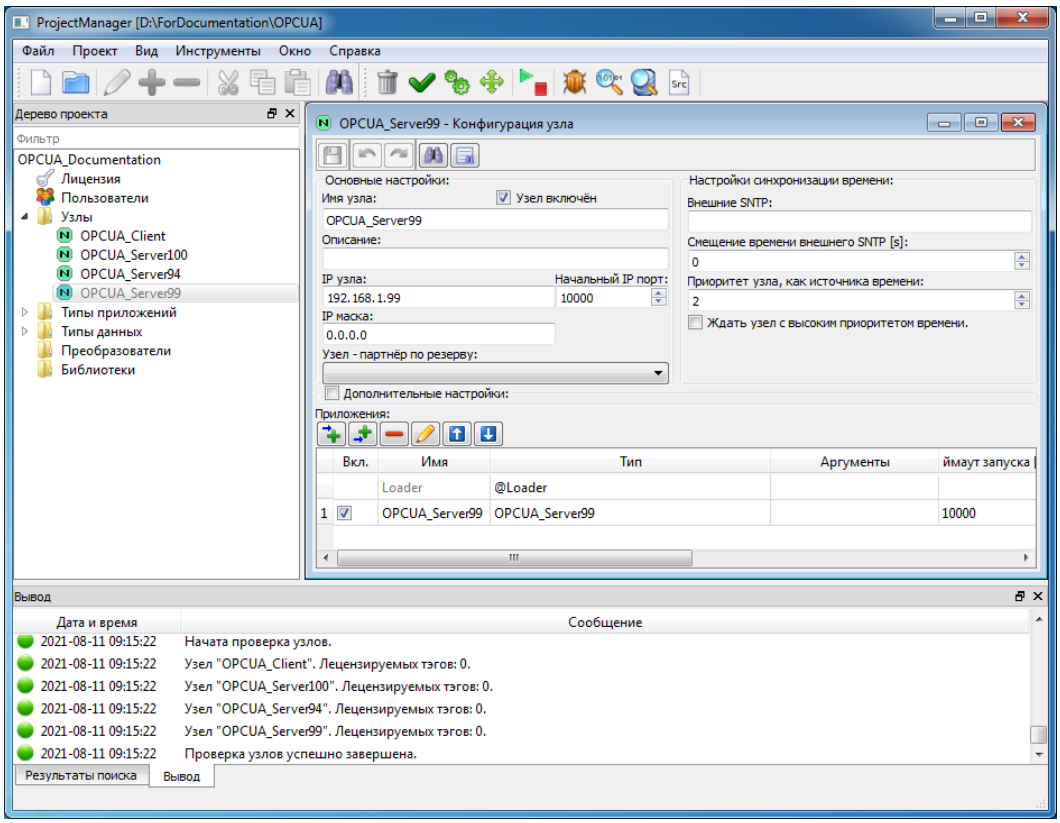


Рисунок 3.242 - Настройки узла OPCUA_Server99

Приложение OPCUA_Server94 добавим на узел OPCUA_Server94 (см. рис. 3.243).

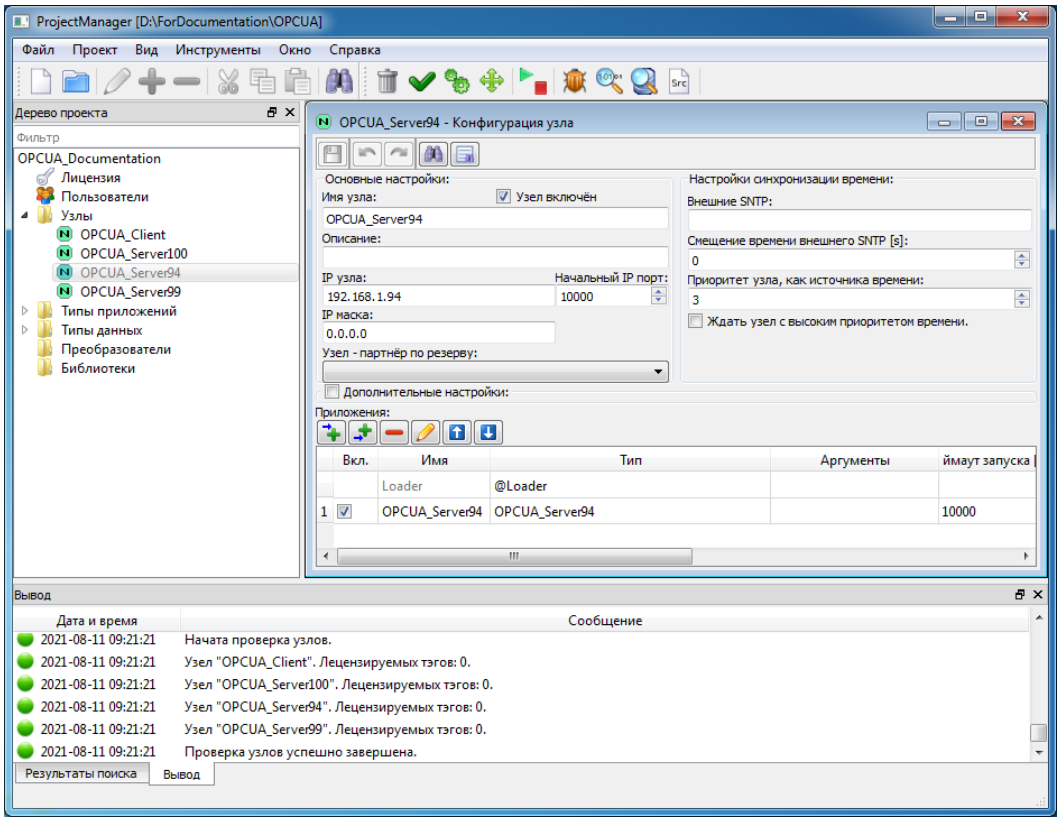


Рисунок 3.243 - Настройки узла OPCUA_Server94

Приложение OPCUA_Client добавим на узел OPCUA_Client (см. рис. 3.244).

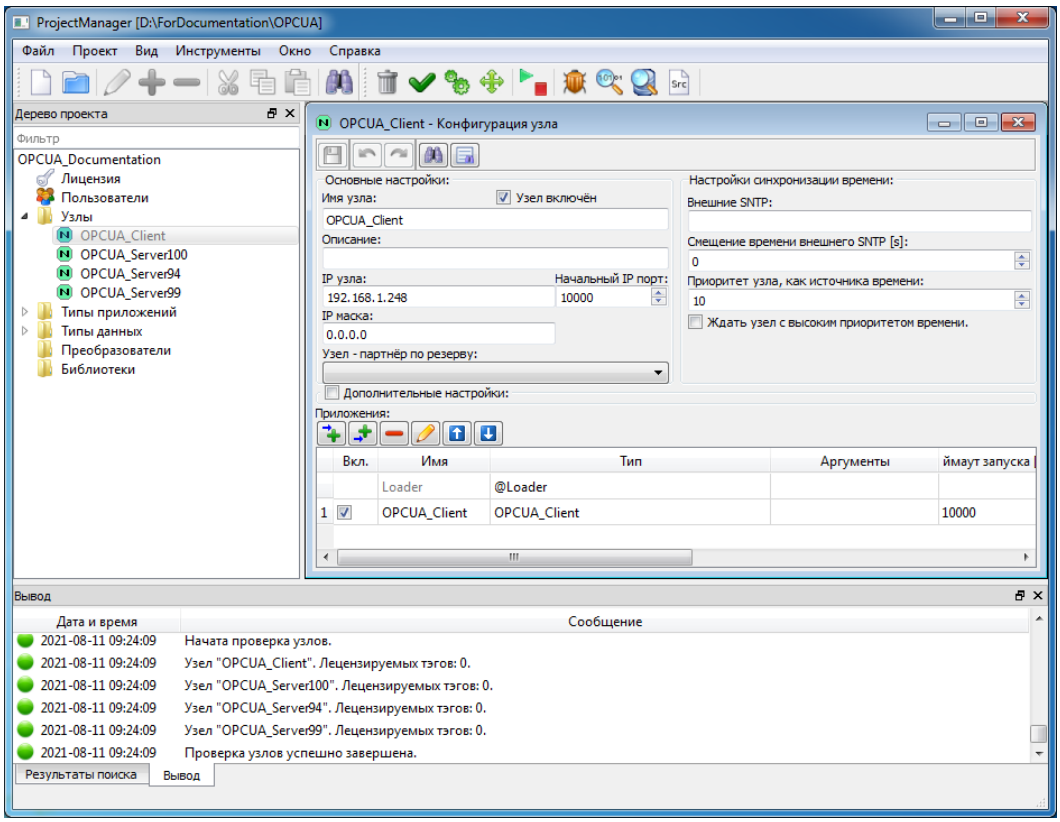


Рисунок 3.244 - Настройки узла OPCUA_Client

7. Наш проект готов. Необходимо его скомпилировать, разослать по узлам и можно запускать. Центр управления с запущенным проектом представлен на рис. 3.245.

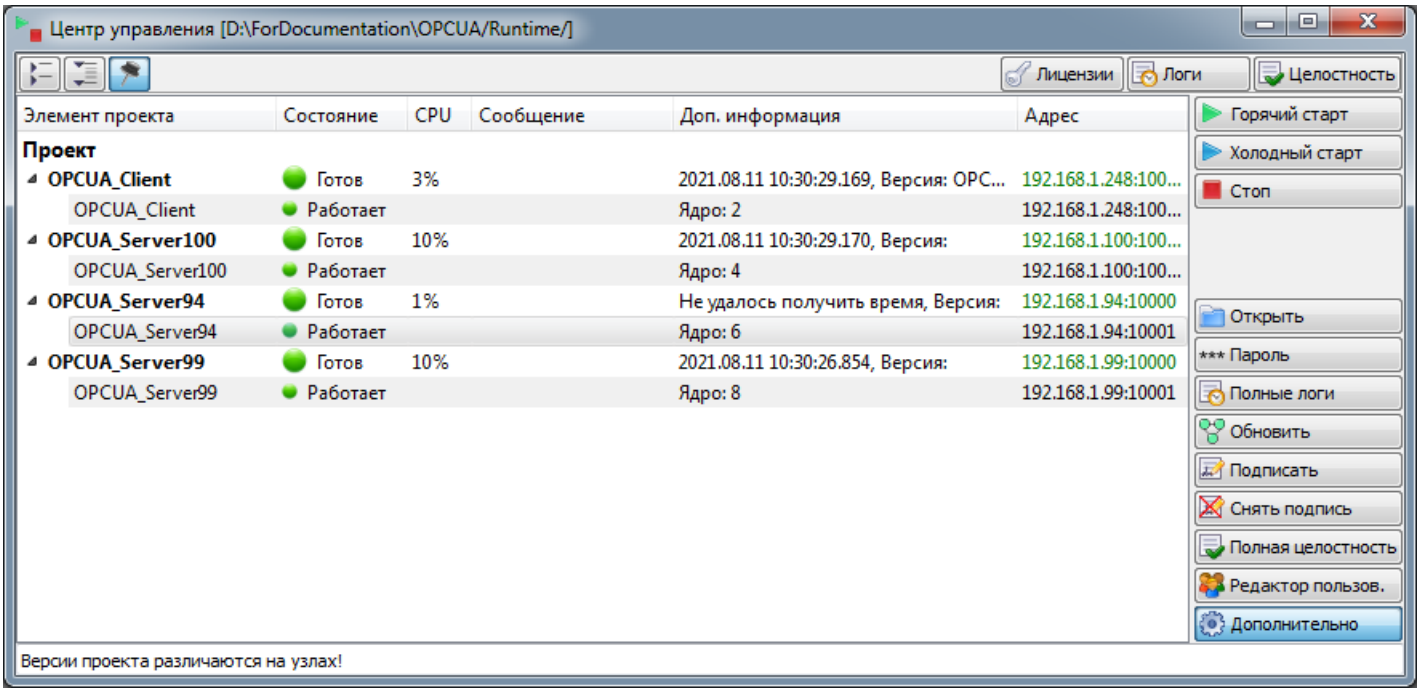
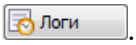


Рисунок 3.245 - Работающий проект в Центре управления

8. Из логов наших приложений OPCUA-сервер и клиент можно получить разную полезную информацию.

Для получения логов приложения необходимо выделить нужное приложение в Центре управления и справа сверху нажать кнопку 

.

Посмотрим логи приложения OPCUA_Server94 (см. рис. 3.246).

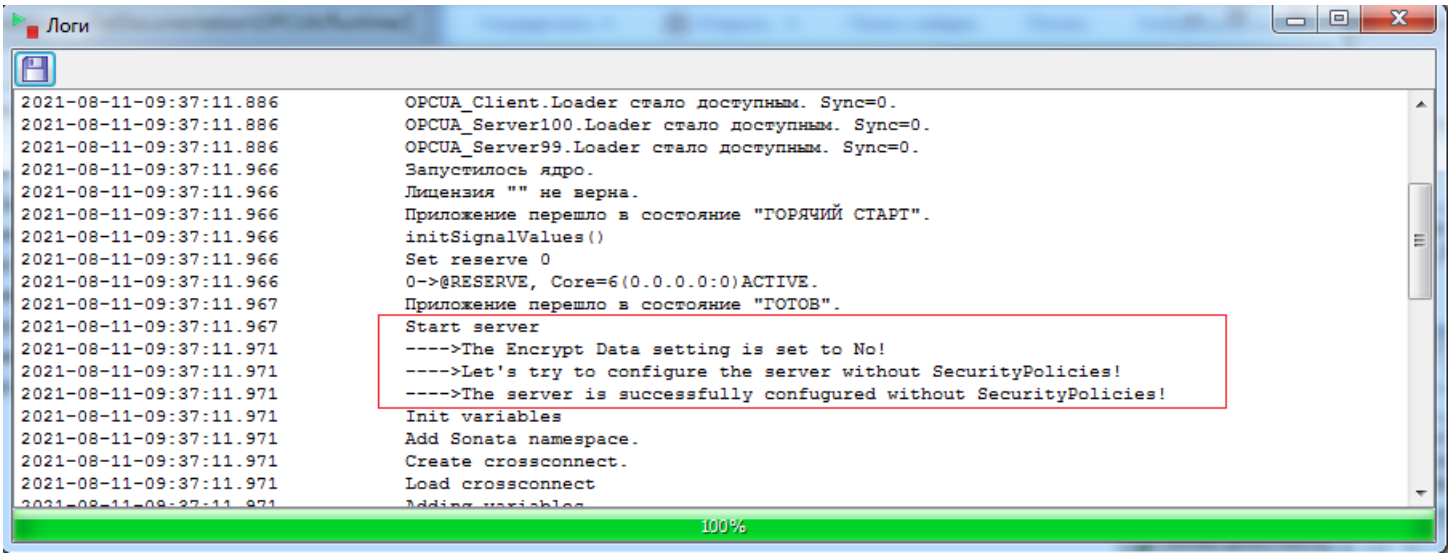


Рисунок 3.246 - Логи приложения OPCUA_Server94

Выделенная информация сообщает, что не включен режим шифрования передачи данных и поэтому сервер настроен без возможности использования политик шифрования.

Посмотрим логи приложения OPCUA_Server99 (см. рис. 3.247).

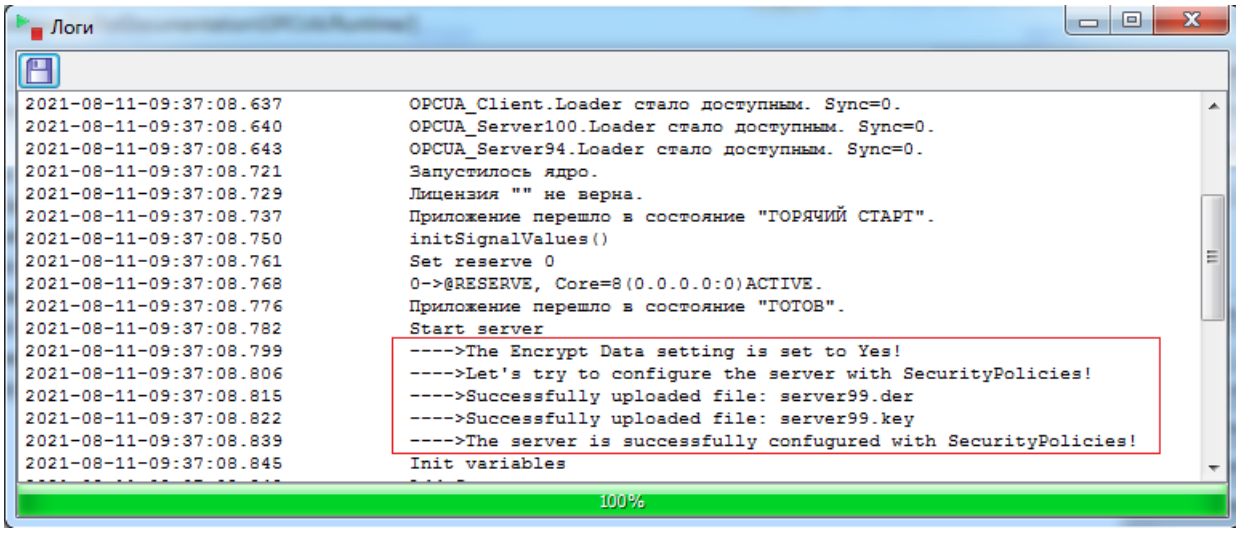


Рисунок 3.247 - Логи приложения OPCUA_Server99

Выделенная информация сообщает следующее:

- включен режим шифрования данных;

- удачно загружен файл server99.der;

- удачно загружен файл server99.key;

- сервер сконфигурирован для работы с использованием политик шифрования.

Аналогичную информацию мы сможем увидеть в логах приложения OPCUA_Server100.

Посмотрим логи приложения OPCUA_Client (см. рис. 3.248).

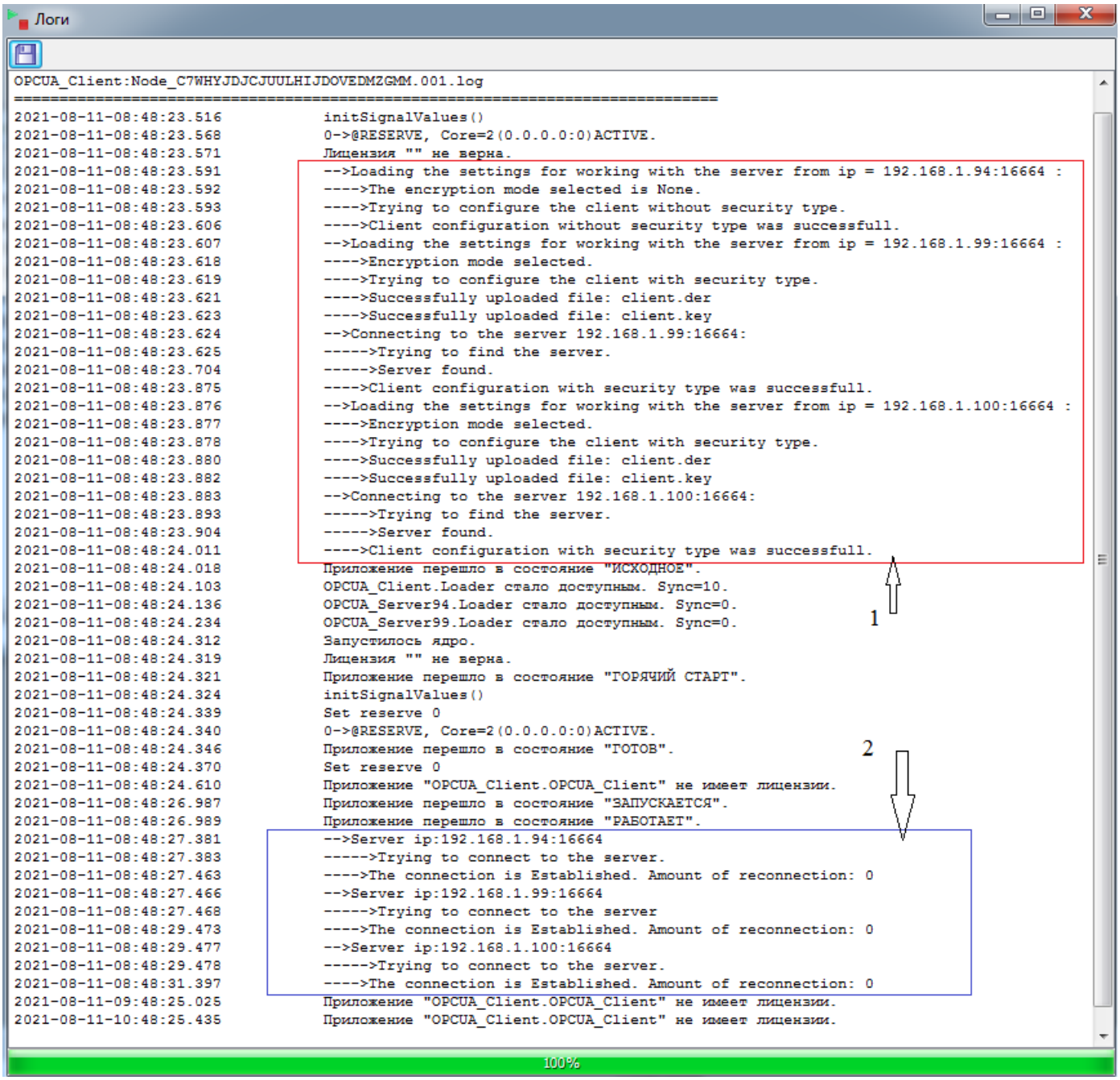


Рисунок 3.248 - Логи приложения OPCUA_Client

В данных логах можно получить следующую информацию:

- под цифрой 1 - конфигурирование клиента для подключения к каждому серверу и проверка связи с сервером;

- под цифрой 2 - информация о подключении к каждому серверу, количество разрывов связи и сообщение об ошибке (см. рис. 3.249).

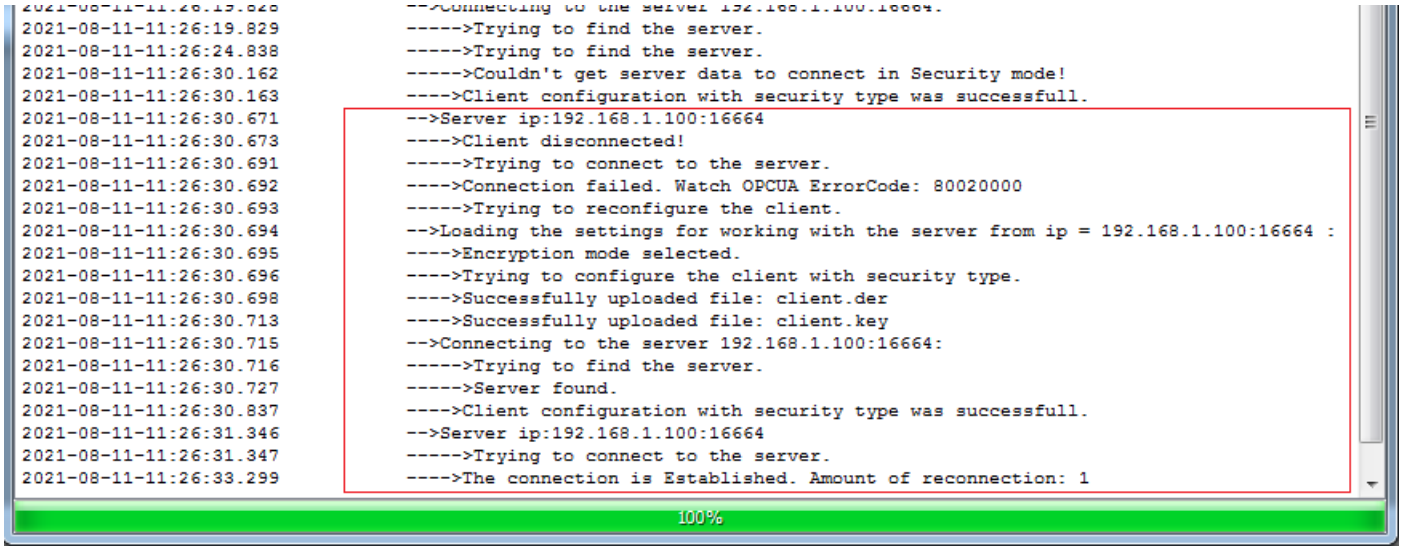


Рисунок 3.249 - Логи приложения OPCUA_CLient с информацией об ошибке

Это все основные моменты, которые нужно знать при использовании приложений OPCUA-сервер и OPCUA-клиент SCADA-системы "СОНАТА". Для того чтобы проверить передачу данных между серверами и клиентом воспользуйтесь программой Просмотрщик сигналов.

3.3.8.6. Описание работы с приложением-драйвером PCI

Приложение-драйвер PCI применяется для работы с цифровыми и аналоговыми модулями ввода/вывода и модулями измерения частоты, которые располагаются на специальной плате-носителе с разъемом PCI.

3.3.8.6.1. Создание приложения-драйвера PCI

Создание приложения-драйвера PCI осуществляется аналогично другим приложениям-драйверам SCADA-системы "Соната" (см. раздел 3.2.2.4).

На рис. 3.250 изображено стандартное окно интерфейса приложения SCADA-системы "Соната". Чтобы данное приложение стало приложением-драйвером PCI, необходимо в поле **Вид приложения** выбрать **DRIVER.PCI**, а в поле **Имя типа** написать удобное для дальнейшего использования в проекте имя данного приложения (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем имя типа не может начинаться с цифры).

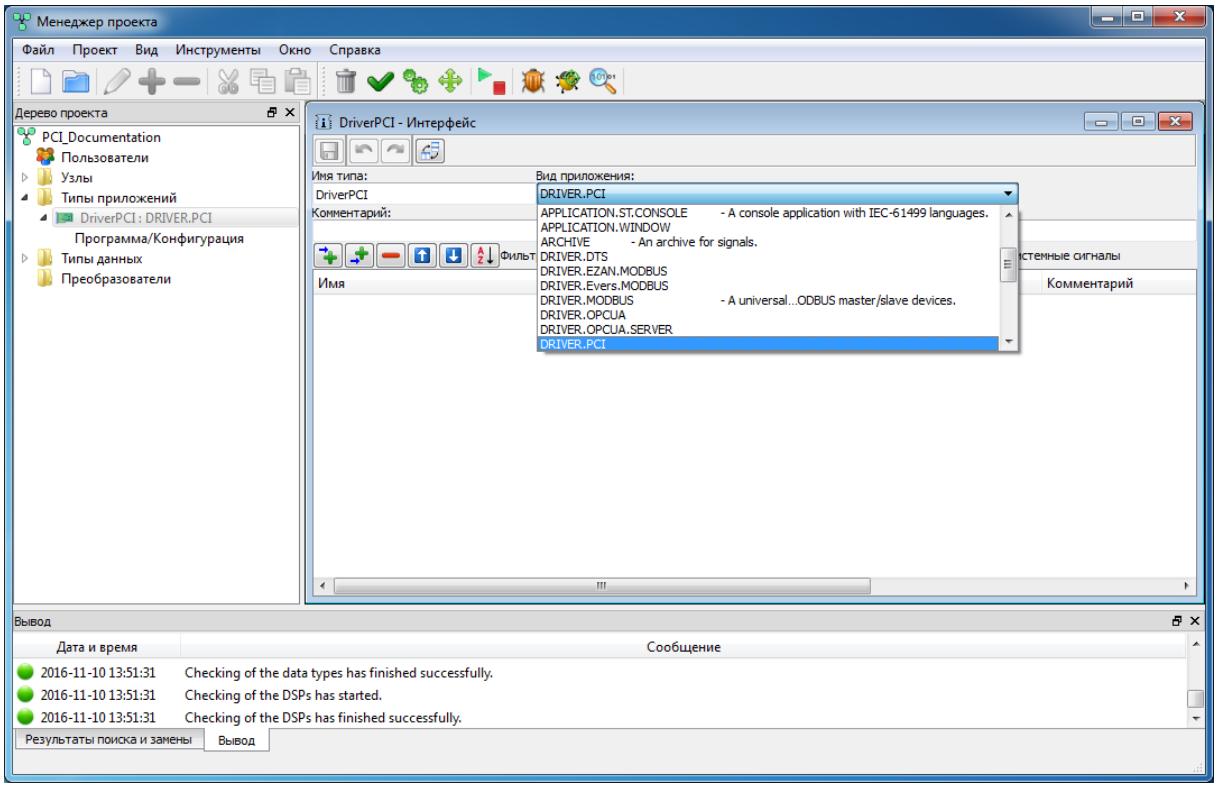


Рисунок 3.250 - Окно редактора интерфейса приложения

3.3.8.6.2. Создание интерфейса приложения-драйвера PCI

После выбора вида приложения-драйвера необходимо добавить в интерфейс приложения сигналы, значения которых будут связаны с каналами модулей ввода/вывода. Процесс добавления сигналов в интерфейс приложения подробно описан в раздел 3.2.2.4.3. На рис. 3.251 в интерфейс приложения-драйвера добавлено 6 сигналов: **DINPUT1** - сигнал дискретного ввода, **DOUTPUT1** - сигнал дискретного вывода, **AINPUT1** - сигнал аналогового ввода, **AOUTPUT1** - сигнал аналогового вывода, сигнал **INFO** - состояние микропроцессорного модуля, **Module_Status** - состояние одного из модулей (см. далее) и **Canal_Status** - состояние одного из каналов (см. далее). Значения кодов и статусов для сигналов **INFO**, **Module_Status** и **Canal_Status** смотри в Приложение В.

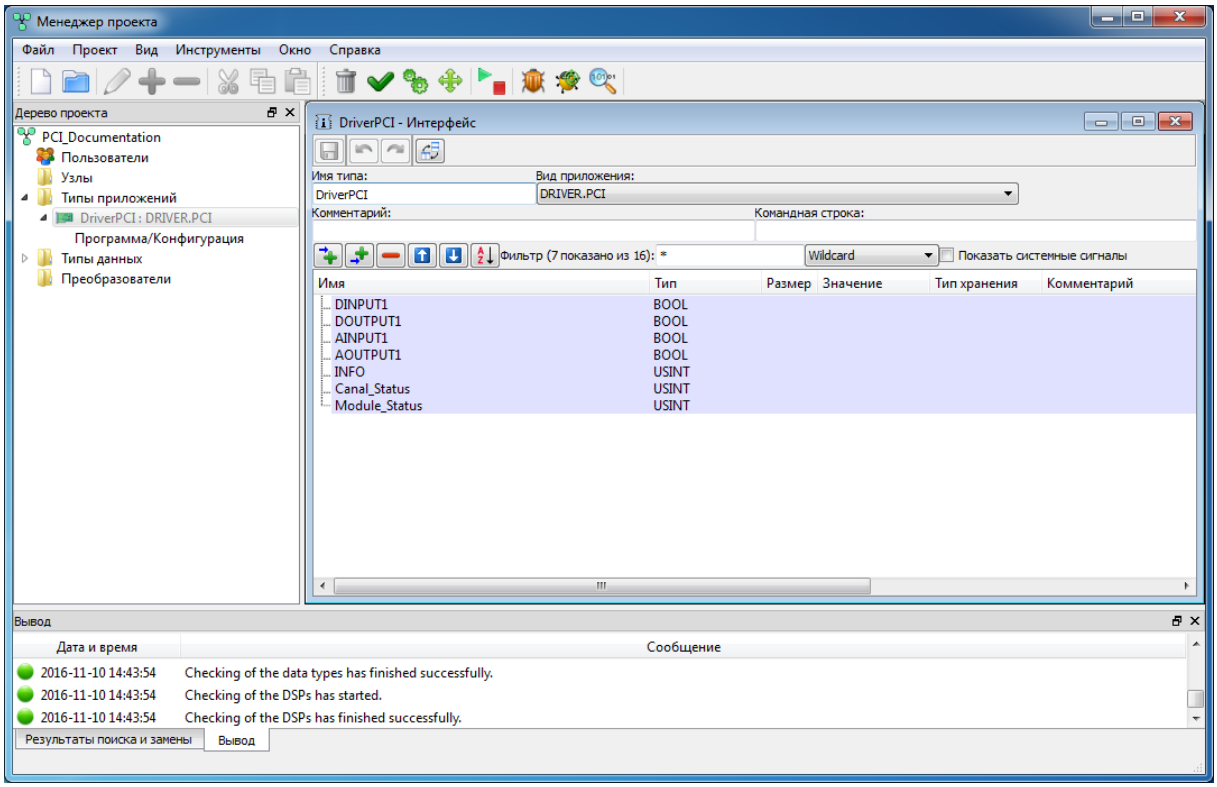


Рисунок 3.251 - Интерфейс приложения-драйвера PCI

После добавления сигналов в интерфейс приложения-драйвера необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Интерфейс приложения настроен и теперь необходимо настроить связь сигналов из интерфейса приложения с каналами модулей ввода/вывода.

3.3.8.6.3. Привязка сигналов приложения-драйвера PCI к каналам ввода/вывода

Для привязки сигналов из интерфейса приложения-драйвера к каналам модулей ввода/вывода необходимо вызвать специализированный редактор приложения-драйвера PCI. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4). На рис. 3.252 изображено окно специализированного редактора (далее просто редактор) приложения-драйвера PCI.

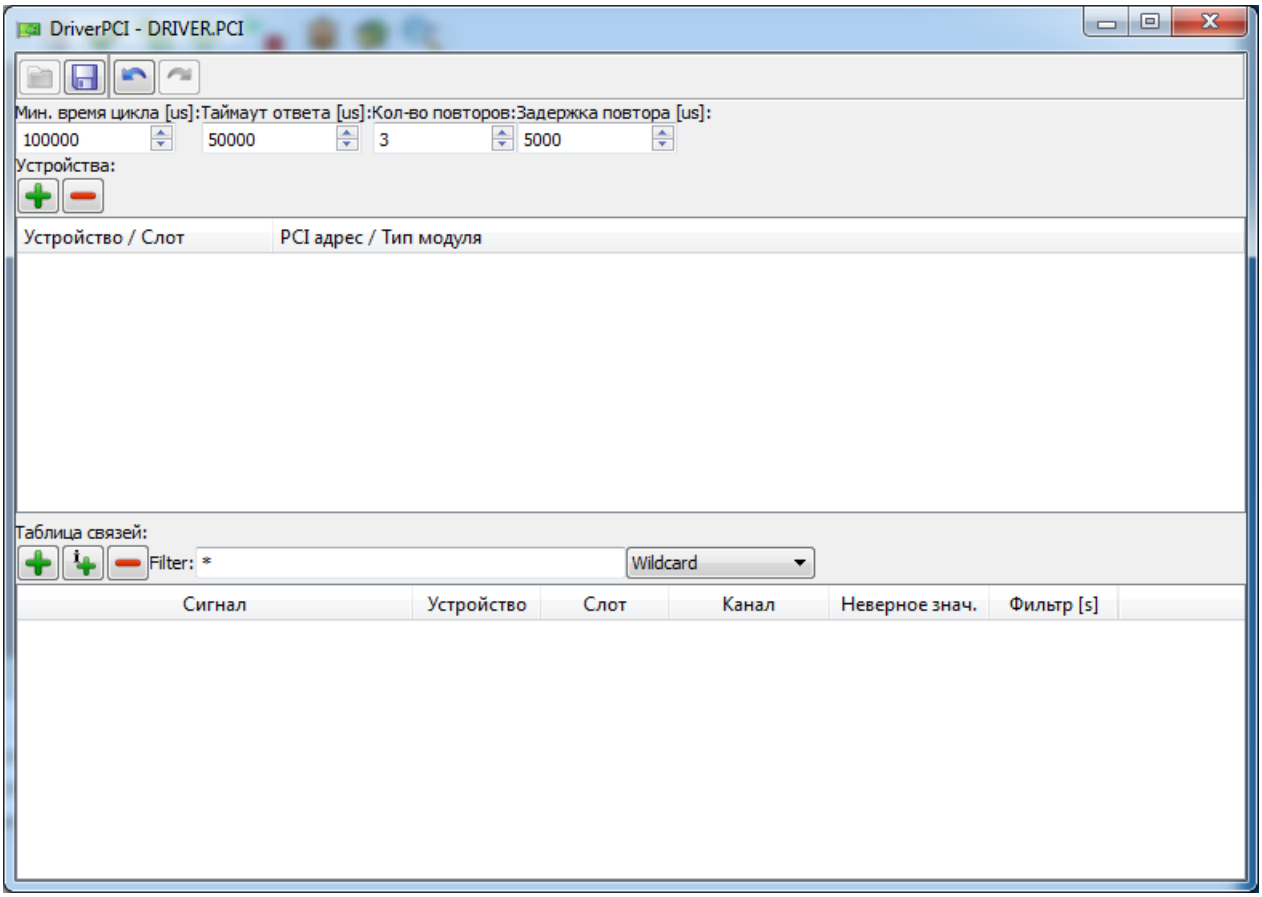


Рисунок 3.252 - Специализированный редактор приложения-драйвера PCI

В верхней части окна редактора находится строка с полным именем приложения, состоящая из имени типа и вида приложения. В данном случае **DriverPCI**(имя типа) - **DRIVER.PCI**(вид приложения). Правее располагаются стандартные для оконных приложений кнопки управления окном  , имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

Ниже располагаются следующие кнопки и поля ввода:

 ("Открыть") - открыть файл с готовыми настройками;

 («Сохранить») – сохранить внесённые в редакторе изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

- **Мин. время цикла [us]** - время цикла опроса, если опрос проходит быстрее данного времени, то следующий опрос начнётся не сразу, а когда выйдет данное время до конца, если опрос пройдёт медленнее, то следующий опрос начнётся сразу;

- **Таймаут ответа [us]** - время ожидания прихода первого байта ответа;

- **Кол-во повторов [us]** - кол-во попыток отправки повторных запросов;

- **Задержка повтора [us]** - задержка перед началом повторного запроса.

Далее расположено окно **Устройства**, которое состоит из следующих кнопок управления и столбцов:

 («Добавить устройство») – команда для добавления нового устройства в список;

(«Удалить устройство») – команда для удаления выделенного устройства.

- **Устройство/Слот** - порядковый номер устройства на шине PCI и позиции-слоты модулей ввода/вывода;

- **PCI адрес/Тип модуля** - PCI адрес платы-носителя и известные SCADA-системе "Соната" для приложения-драйвера PCI типы модулей.

Для добавления нового контроллера в список необходимо нажать кнопку («Добавить контроллер») – на экране появится диалог (см. рис. 3.253), в котором следует указать адрес устройства (порядковый номер), т.е. платы-носителя (в этом поле данное значение является абстрактным и лучше всего задавать номер, который будет понятен для определения места положения платы-носителя). После задания нужного адреса необходимо в поле **Идентификатор** выбрать нужную модель платы-носителя: плата-носитель для дискретных и аналоговых модулей или плата-носитель для модулей измерения частоты. Далее нажать в диалоге кнопку OK – в таблице **Устройства** отобразится контроллер с введённым адресом (см. рис. 3.254), для которого необходимо задать состав модулей ввода/вывода.

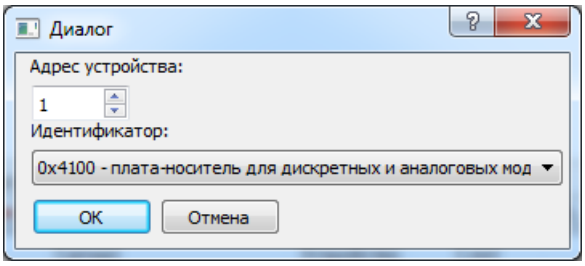


Рисунок 3.253 - Диалог добавления нового устройства в редакторе

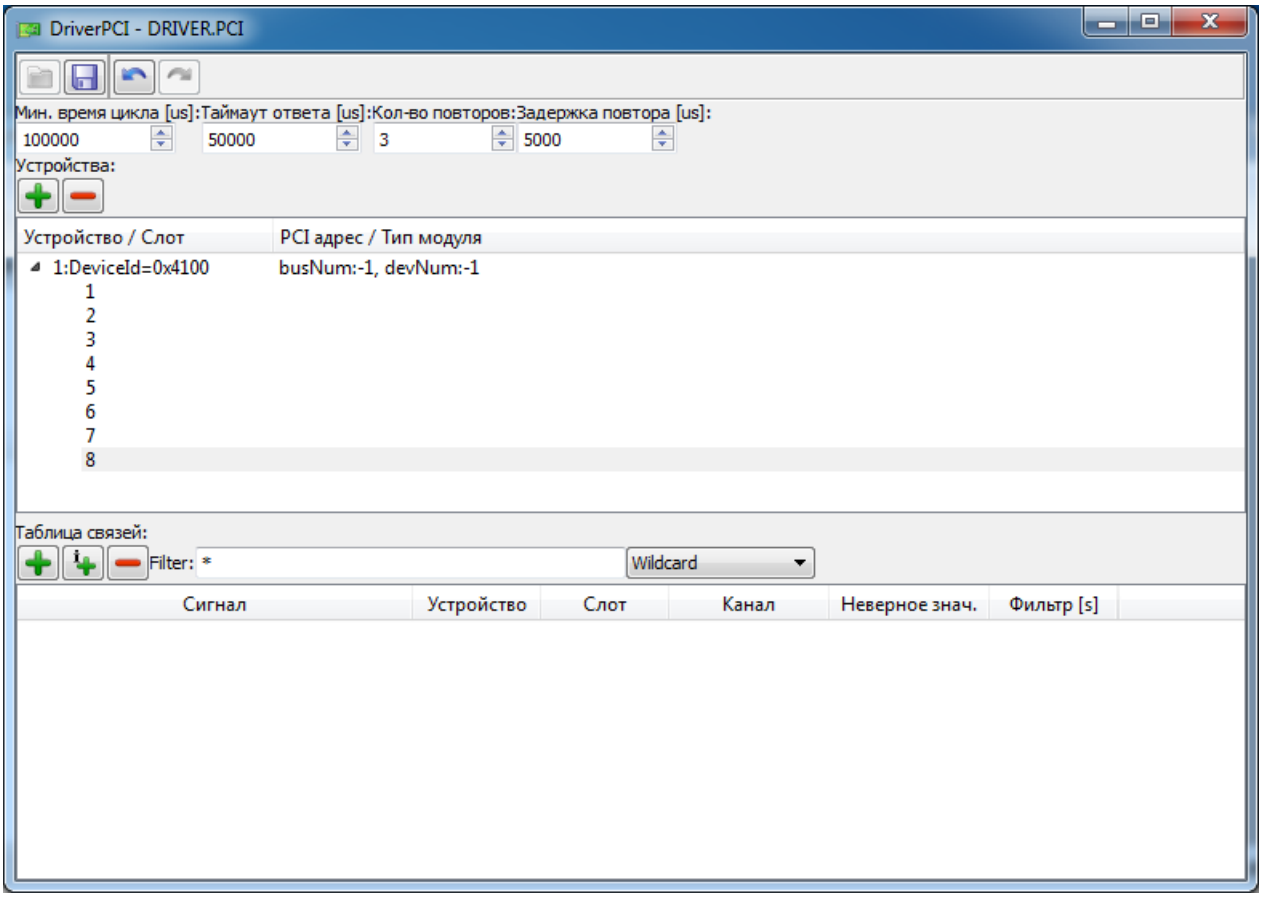


Рисунок 3.254 - Добавленная плата-носитель с порядковым номером устройства 1

Чтобы задать состав модулей ввода/вывода нужно сначала двойным щелчком мыши выделить поле **busNum:-1** (номер шины), **devNum:-1** (номер устройства) в столбце **PCI адрес/Тип модуля**, оно преобразуется в поля настроек **Bus** (шина) и **Device** (устройство) (см. рис. 3.255). Данные поля настраиваются в соответствии с тем, как определилась плата-носитель на шине PCI.

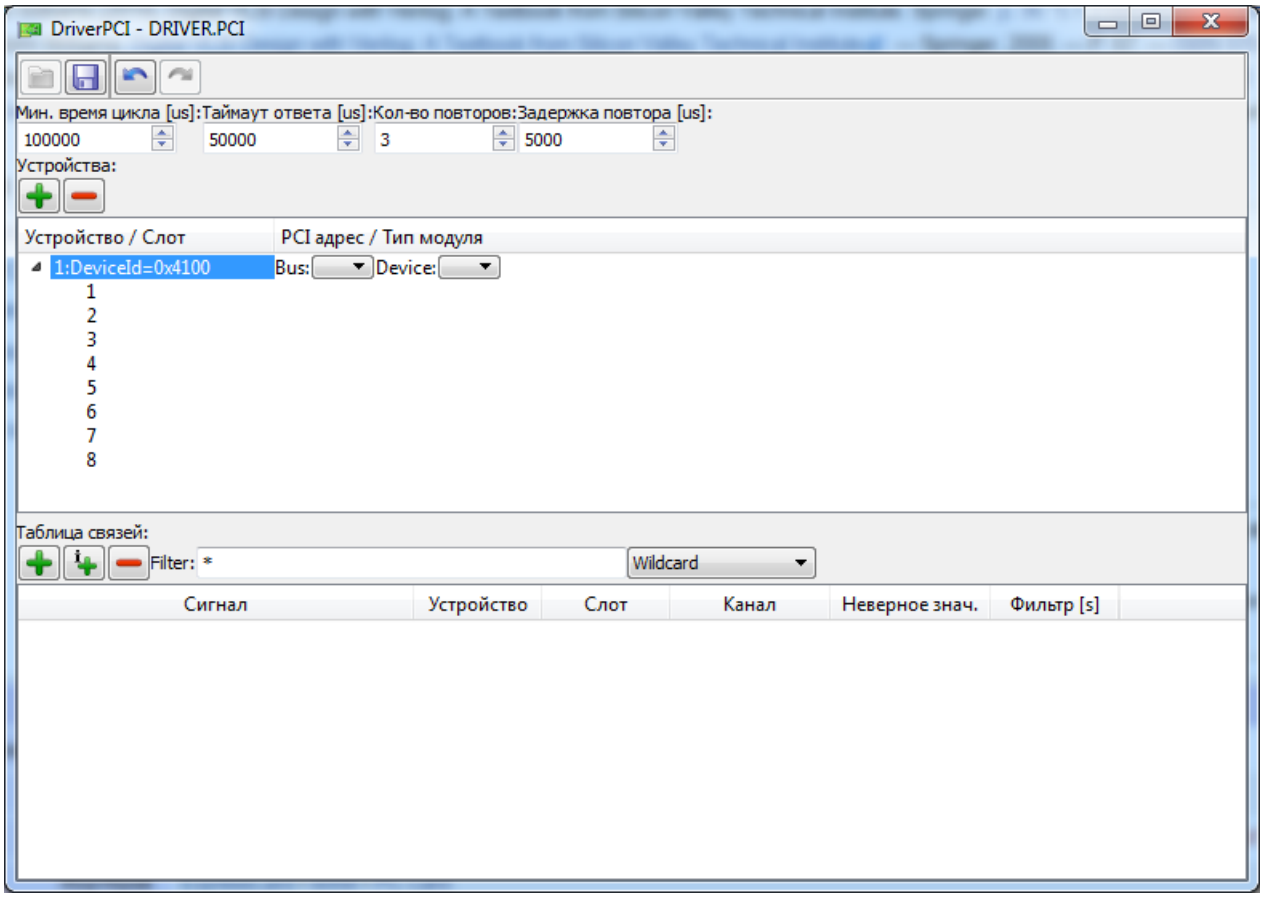


Рисунок 3.255 - Настройка PCI адреса устройства

Далее необходимо сконфигурировать расположение модулей ввода/вывода на плате-носителе. Расположение характеризуется полем **Слот**.Как видно на рис. 3.255 плата-носитель имеет 8 слотов. Выбор нужного модуля производится двойным щелчком мыши в столбце **PCI адрес/Тип модуля** напротив нужного номера слота (см. рис. 3.256). Модули ввода/вывода могут занимать от одной до четырёх позиций-слотов на плате-носителе. На рис. 3.256 в позиции-слоте 1 выбран цифровой модуль на 8 входов, который занимает один слот, в позиции 2 выбран цифровой модуль на 8 выходов, который занимает один слот, в 3 позиции выбран аналоговый модуль на 4 входа, который занимает 2 слота - третий и четвёртый, в позиции 5 выбран аналоговый модуль на 16 выходов, который занимает 2 слота - пятый и шестой.

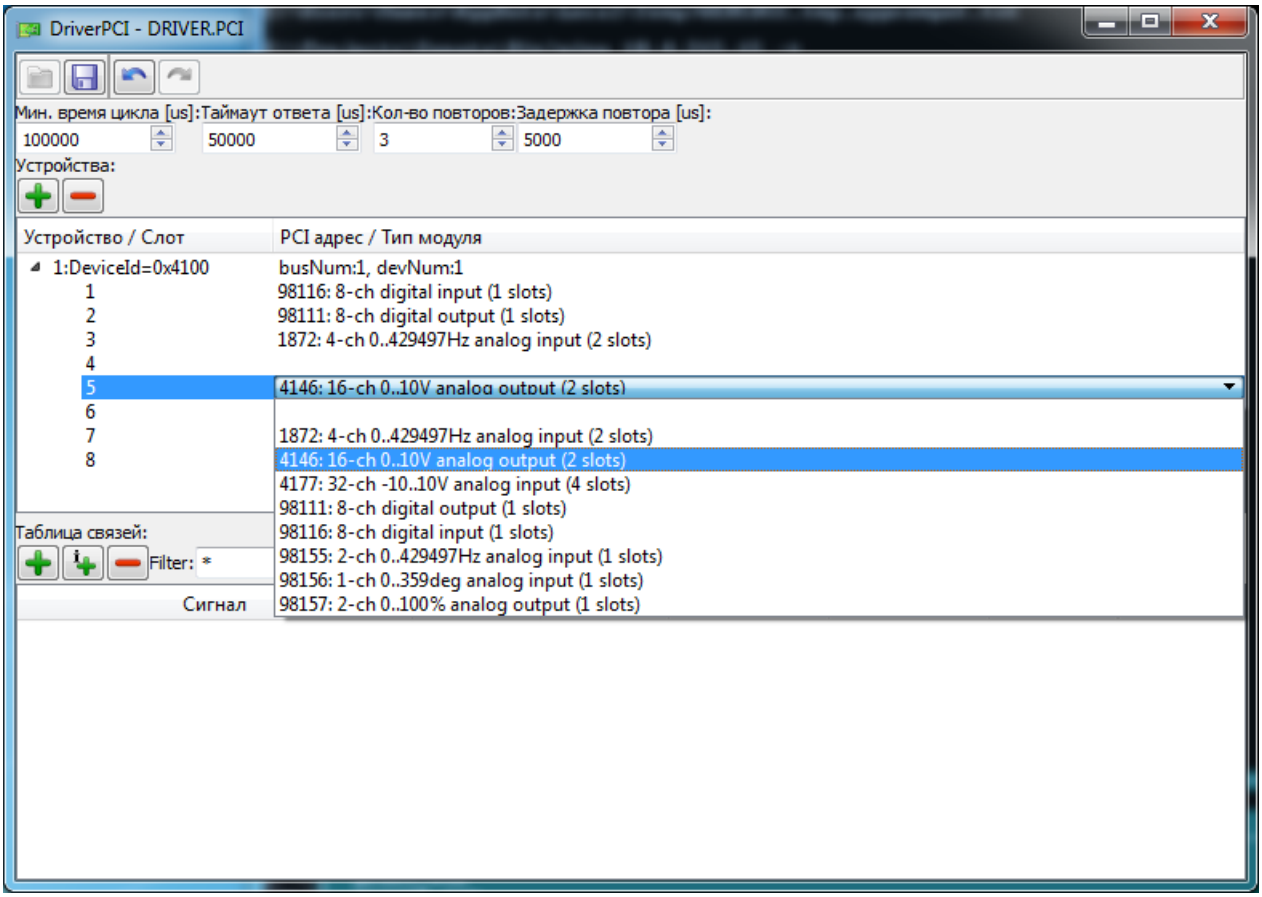


Рисунок 3.256 - Выбор модулей ввода/вывода в нужной позиции-слоте

После настройки конфигурации устройства необходимо настроить связь сигналов SCADA-системы "Соната" с каналами модулей ввода/вывода. Данные настройки проводятся в **Таблице связей** (см. рис. 3.257).

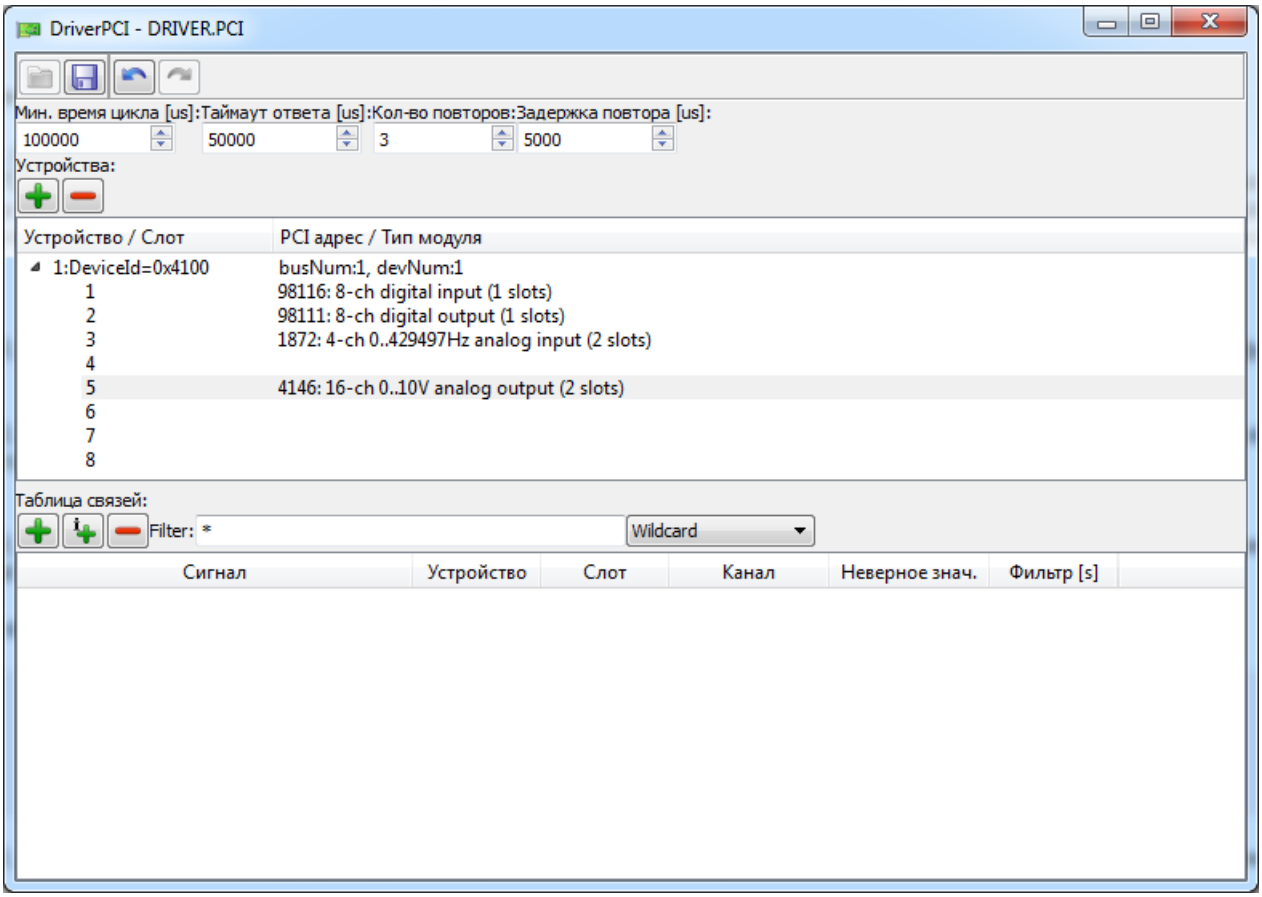


Рисунок 3.257 - Окно редактора с настроенной конфигурацией устройства

Таблица связей состоит из следующих кнопок, столбцов и элементов управления:

 ("Добавить элемент") - добавить новую строку в **Таблицу связей**;

 ("Импортировать элементы из интерфейса") - добавить все сигналы из интерфейса приложения-драйвера;

 ("Удалить элемент") - удалить выбранный элемент из таблицы связей;

- поле настройки фильтров отображения сигналов;

- **Сигнал** - сигнал из интерфейса приложения-драйвера;

- **Устройство** - номер устройства из таблицы **Устройства**;

- **Слот** - позиция модуля ввода/вывода;

- **Канал** - номер канала на модуле ввода/вывода;

- **Неверное знач.** - значение, которое будет принимать сигнал, когда канал неисправен;

- **Фильтр [s]** - постоянная времени фильтра, выраженная в секундах.

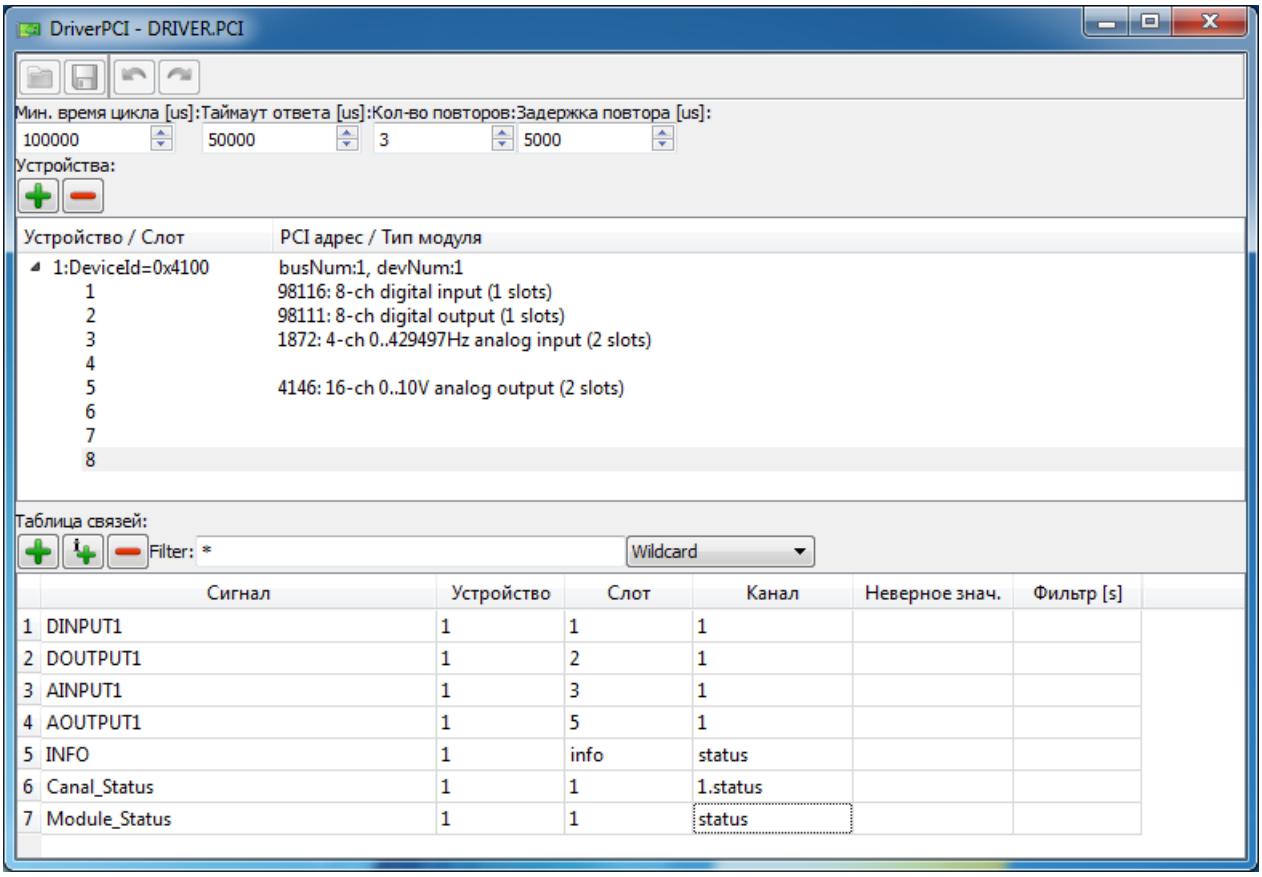


Рисунок 3.258 - Настроенное окно редактора приложения-драйвера PCI

Настройка **Таблицы связей** осуществляется следующим образом.

С помощью кнопок  ("Добавить элемент") и  ("Импортировать элементы из интерфейса") в столбец **Сигнал** добавляем нужные сигналы из интерфейса приложения-драйвера. В столбце **Устройство** выбираем номер нужного устройства, в столбце **Слот** нужно выбрать соответствующий номер позиции-слота, в которой в соответствии с ранее настроенной таблицей **Устройства** находится нужный модуль ввода/вывода. Если выбрать в столбце **Слот** значение **info**, то данный сигнал будет сообщать о состоянии платы-носителя. В столбце **Канал** выбирается номер канала ввода/вывода и так же есть дополнительные значения, которые рассмотрим на примере (см. рис. 3.258).

Для сигнала **DINPUT1** выставлены значения **Устройство** = 1, **Слот** = 1 - это цифровой модуль на 8 входов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **DOUTPUT1** выставлены значения **Устройство** = 1, **Слот** = 2 - это цифровой модуль на 8 выходов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **AINPUT1** выставлены значения **Устройство** = 1, **Слот** = 3 - это аналоговый модуль на 4 входа и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **AOUTPUT1** выставлены значения **Устройство** = 1, **Слот** = 5 - это аналоговый модуль на 16 выходов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **INFO** выставлены значения **Устройство** = 1, **Слот** = **info** - это получение информации о плате-носителе и **Канал** = **status** - это статус модуля, а как видно по значению **Слот** = **info**, то это статус платы-носителя. Статусы каналов и модулей описаны в Приложение В.

Для сигнала **Canal_Status** выставлены настройки **Устройство** = 1, **Слот** = 1 - это цифровой модуль на 8 входов и **Канал** = **1.status** - это статус первого канала данного модуля.

Для сигнала **Module_Status** выставлены значения **Устройство** = 1, **Слот** = 1 - это цифровой модуль на 8 входов и **Канал** = **status** - это статус модуля, соответственно получаем информацию о работоспособности модуля в первом слоте.

В столбце **Неверное знач.** можно выставить значение для сигнала, которое он всегда будет принимать, в случае любой неисправности канала.

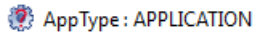
Для дискретных сигналов в столбце **Фильтр [s]** настраивается фильтрация от помех, т.е. значение времени в секундах, в пределах которого изменения сигнала не регистрируются (считаются помехой). Для аналоговых сигналов это константа времени для апериодического фильтра первого порядка.

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер PCI настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на узел проекта, который будет опрашивать данные устройства (см. раздел 3.2.2.5.1).

3.3.8.7. Описание работы с драйвером SNMP

Приложение-драйвер SNMP предназначено для взаимодействия с устройствами на основе протокола SNMP, который даёт возможность проверки функционирования сетевых устройств и многих другий действий.

3.3.8.7.1. Создание приложения - драйвера SNMP

Для создания нового типа приложения необходимо в дереве проекта главного окна программы **ProjectManager** выделить папку **Типы приложений**, а затем нажать кнопку («Добавить элемент») на главной панели инструментов (см. рис. 3.259) – в перечень типов приложений в дереве проекта будет добавлена новая позиция. Новая позиция будет иметь такой вид 

(см. рис. 3.260).

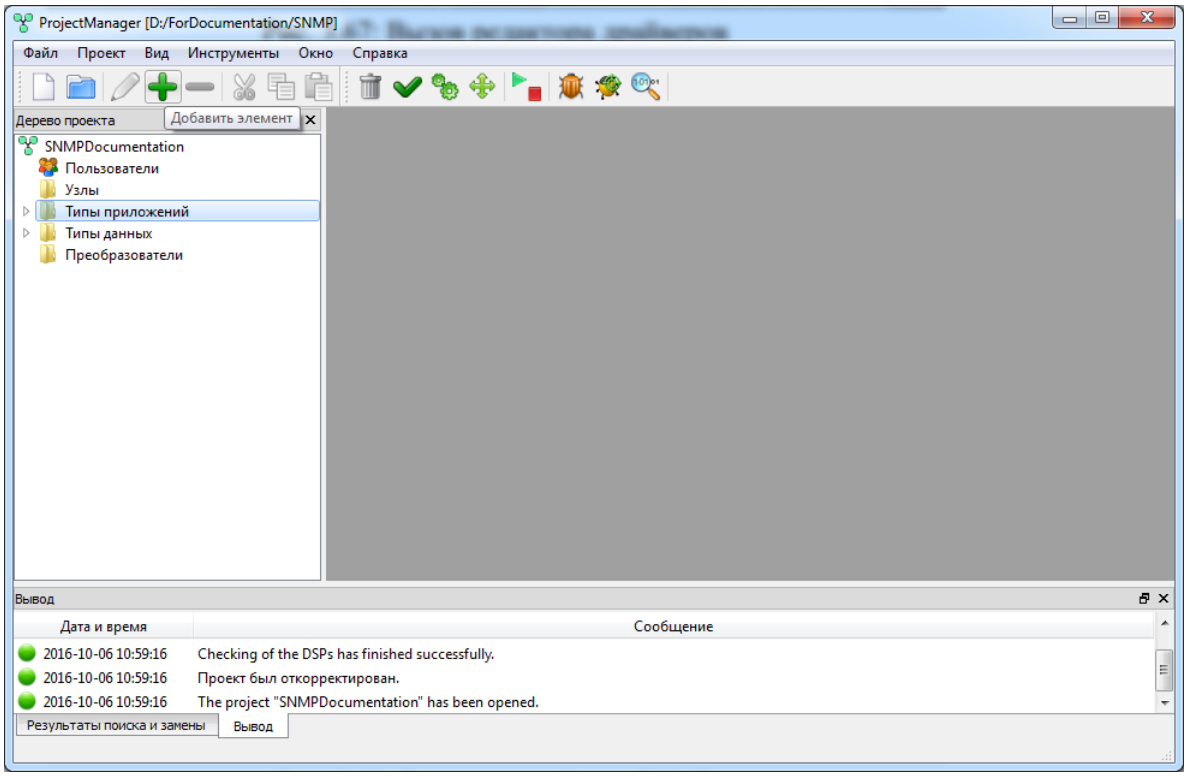


Рисунок 3.259 - Добавление нового типа приложения с помощью кнопки («Добавить элемент»)

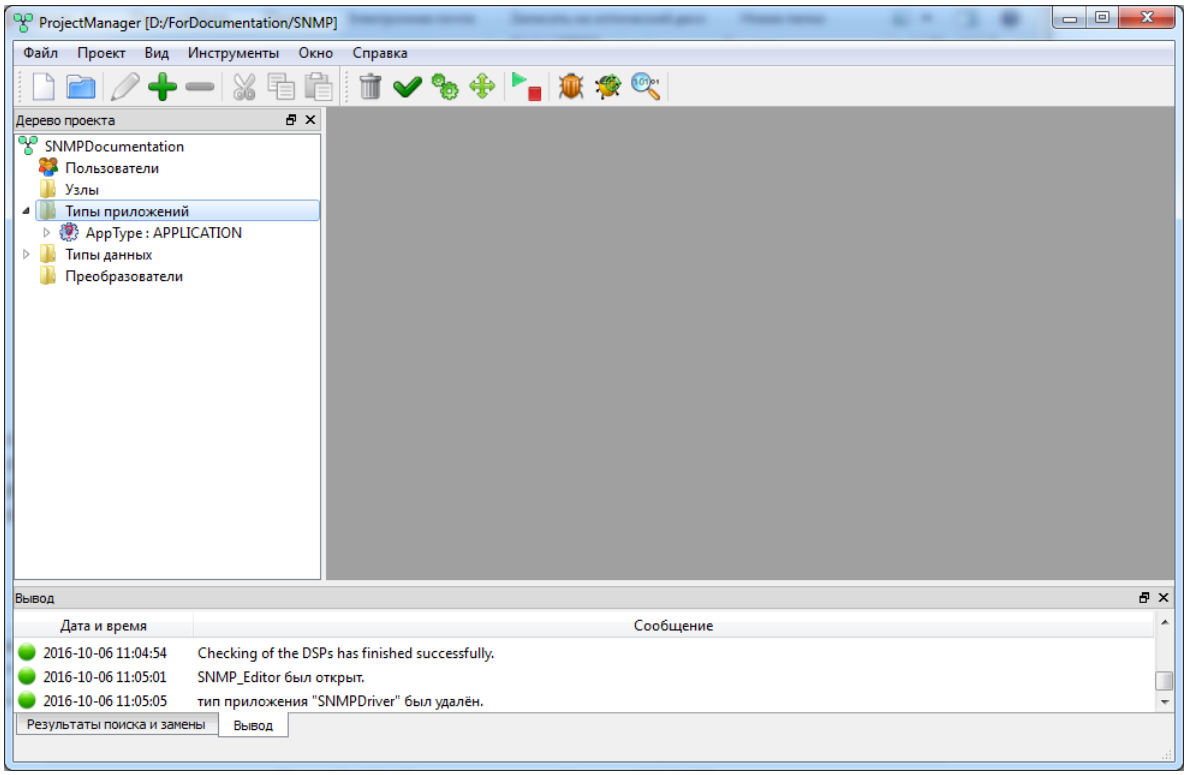


Рисунок 3.260 - Новая позиция в поле Типы приложений

Так как новому типу приложения задается вид **APPLICATION**, а имена новым типам приложений присваиваются автоматически по шаблону **AppType_N** (где N – уникальное число), то имя типа и вид типа необходимо изменить. Для этого необходимо отредактировать **интерфейс типа приложения**. Т.е. нужно в дереве проекта главного окна **ProjectManager** выделить позицию, содержащую имя типа и вид приложения (в данном случае – строку) , а затем нажать кнопку («Редактировать элемент») на главной панели инструментов (см. рис. 3.261) – откроется диалог редактирования интерфейса типа приложения (см. рис. 3.262).

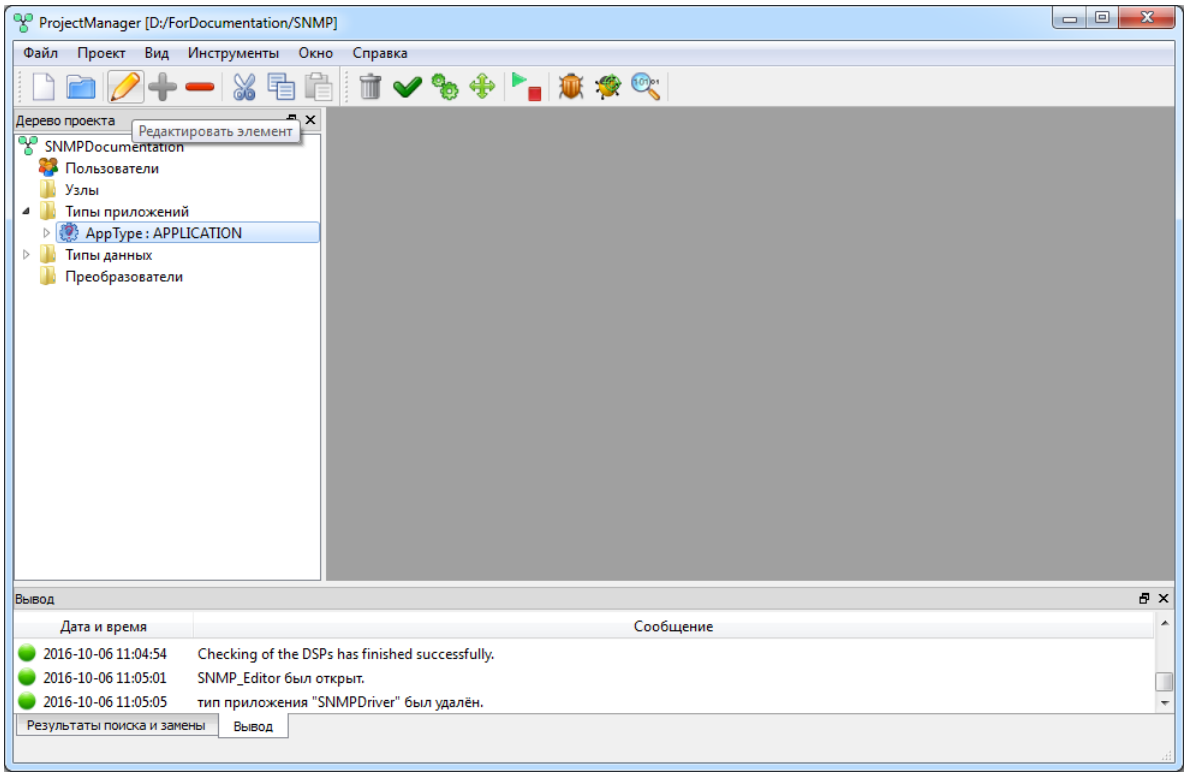


Рисунок 3.261 - Открытие редактора интерфейса типа приложений с помощью кнопки («Редактировать элемент»)

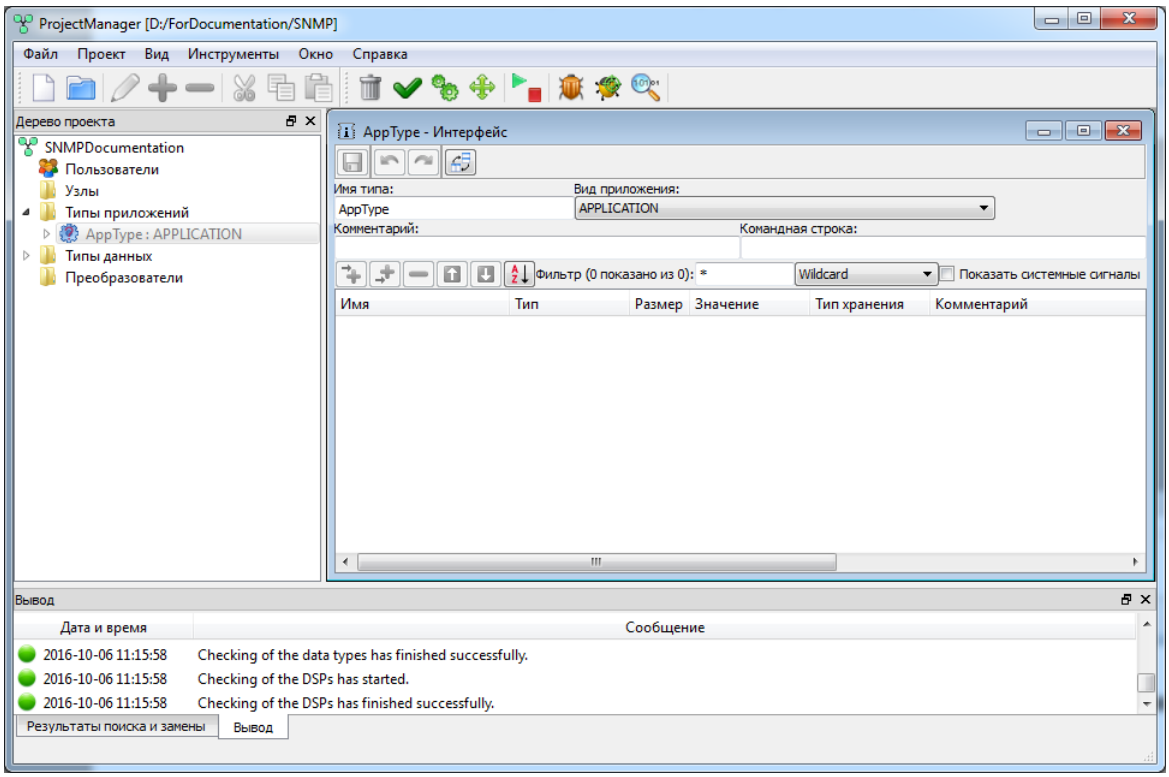


Рисунок 3.262 - Диалог редактирования интерфейса типа приложения

В этом диалоге в поле **Имя типа** можно задать удобное имя для вашего приложения, которое будет понятным вам для использования в дальнейшем. В поле ввода **Вид приложения** необходимо изменить вид **APPLICATION** (присвоенный по умолчанию) на **DRIVER.SNMP** (см. рис. 3.263). После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.264) – для подтверждения изменения необходимо нажать кнопку **<Да>**.

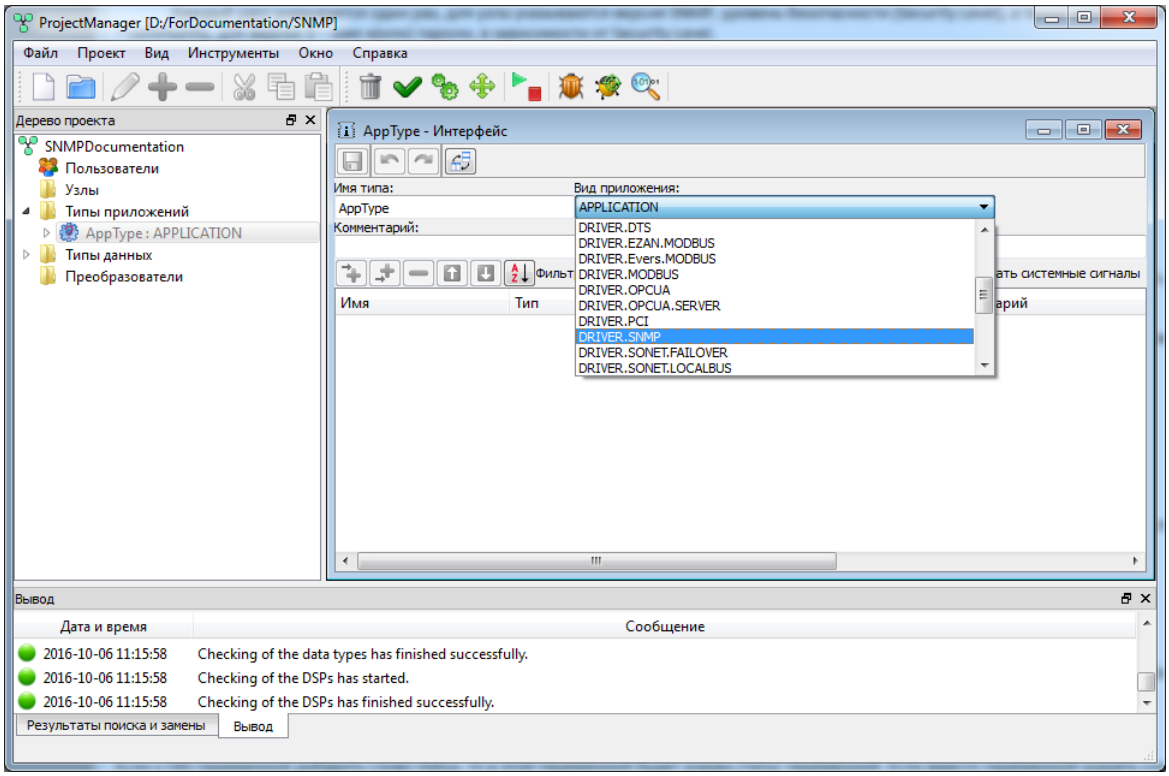


Рисунок 3.263 - Редактирование интерфейса приложения

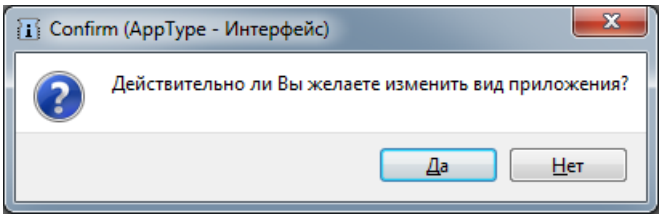


Рисунок 3.264 - Окно подтверждения для изменения вида типа приложения

Изменения следует зафиксировать с помощью кнопки («**Сохранить**») в окне редактирования интерфейса типа приложения (см. рис. 3.265).

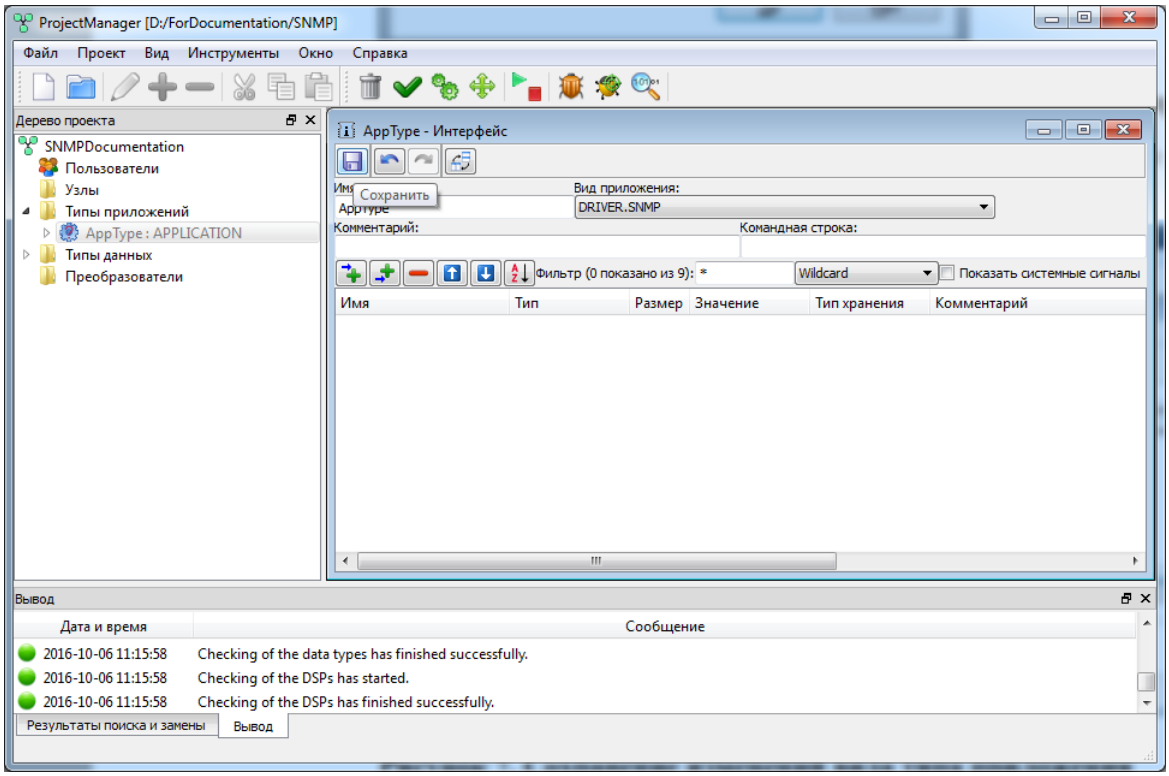


Рисунок 3.265 - Сохранение изменений вида типа приложения

3.3.8.7.2. Создание интерфейса приложения драйвера SNMP

После выбора вида драйвера необходимо добавить в интерфейс приложения сигналы, значение которых будут связаны с каналами модулей ввода/вывода. Для этого нужно нажать одну из кнопок редактирования таблицы интерфейса:  («Вставить сверху») или  («Вставить снизу») – на экране появится окно выбора (см. рис. 3.266), в котором нужно выбрать вариант Локальный или Глобальный.

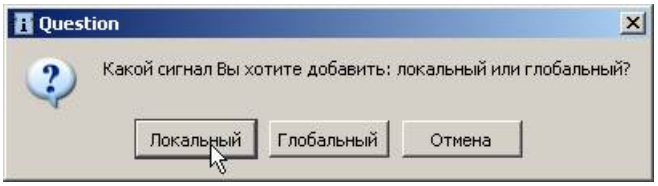


Рисунок 3.266 - Выбор признака Локальный/Глобальный
при добавлении сигнала в интерфейс приложения

У локальных сигналов можно редактировать поля **Имя** и другие. Так как имя новому сигналу задается по шаблону, то его можно изменить, вначале выделив строку сигнала, а затем дважды «кликнуть» «мышью» по ячейке в столбце **Имя** в выделенной строке – строка станет доступной для редактирования (см. рис. 3.267).

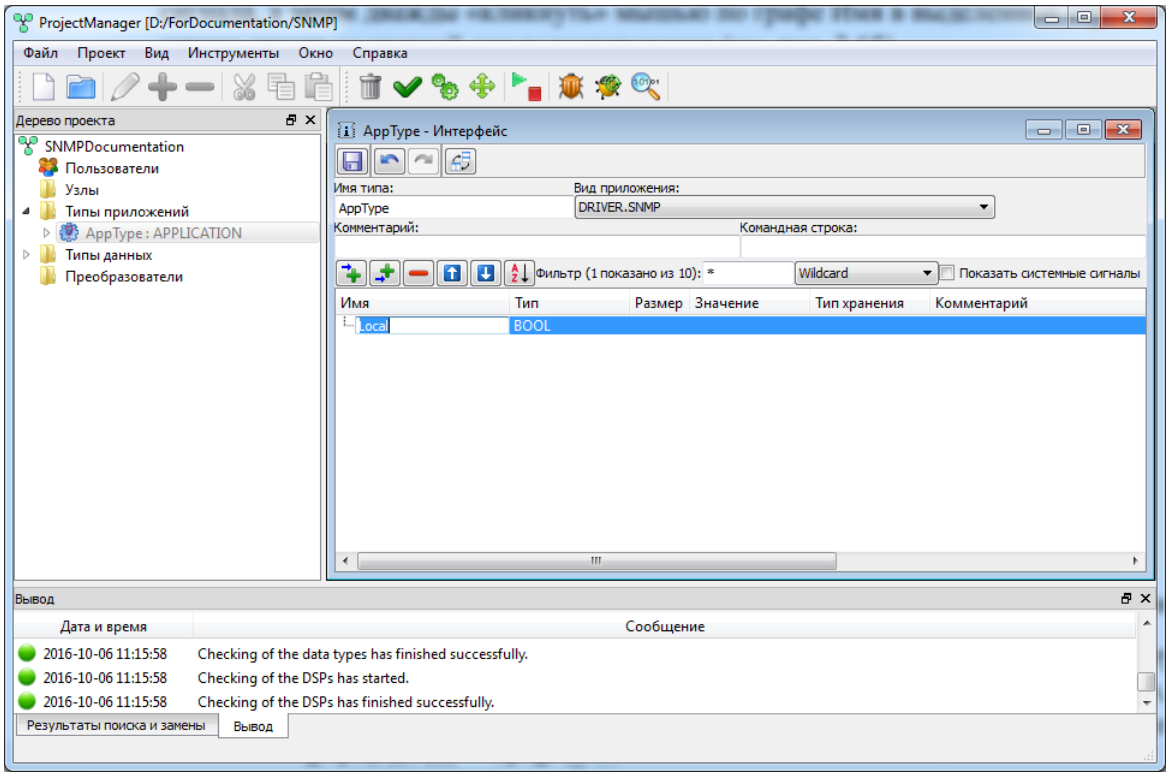


Рисунок 3.267 - Редактирование имени локального сигнала в интерфейсе приложения

Можно также поменять тип сигнала: для этого необходимо дважды «кликнуть» «мышью» по графе **Тип**, а затем кликнуть по кнопке – появится выпадающее меню, в котором можно изменить тип локального сигнала. Подробное описание полей сигнала смотрите раздел 3.2.2.4.3.

Об использовании и добавлении глобальных сигналов смотрите раздел 3.2.2.4.3.2.

.

По завершении редактирования интерфейса типа приложения диалог нужно закрыть крестиком в верхнем правом углу окна редактирования. Если внесенные изменения не были сохранены, то на экране появится диалог (см. рис. 3.268), в котором нужно выбрать либо сохранение изменений и выход из редактора (кнопка <**Сохранить**>) , либо отказ от внесенных изменений и выход из редактора (кнопка <**Отклонить**>) , либо отказаться от выхода из процедуры редактирования (кнопка <**Отмена**>).

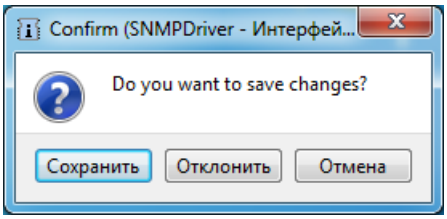


Рисунок 3.268 - Диалог подтверждения выхода из редактора интерфейса приложения

3.3.8.7.3. Настройка драйвера SNMP и привязка сигналов интерфейса драйвера к каналам ввода/вывода

Для вызова специализированного редактора сначала необходимо раскрыть папку в дереве проекта, а затем установить курсор на строку Программа/Конфигурация, выделить ее, и нажать кнопку (

«Редактировать элемент») (см. рис. 3.269) на главной панели инструментов – в результате в новом окне будет открыт специализированный редактор драйверов (см. рис. 3.270).

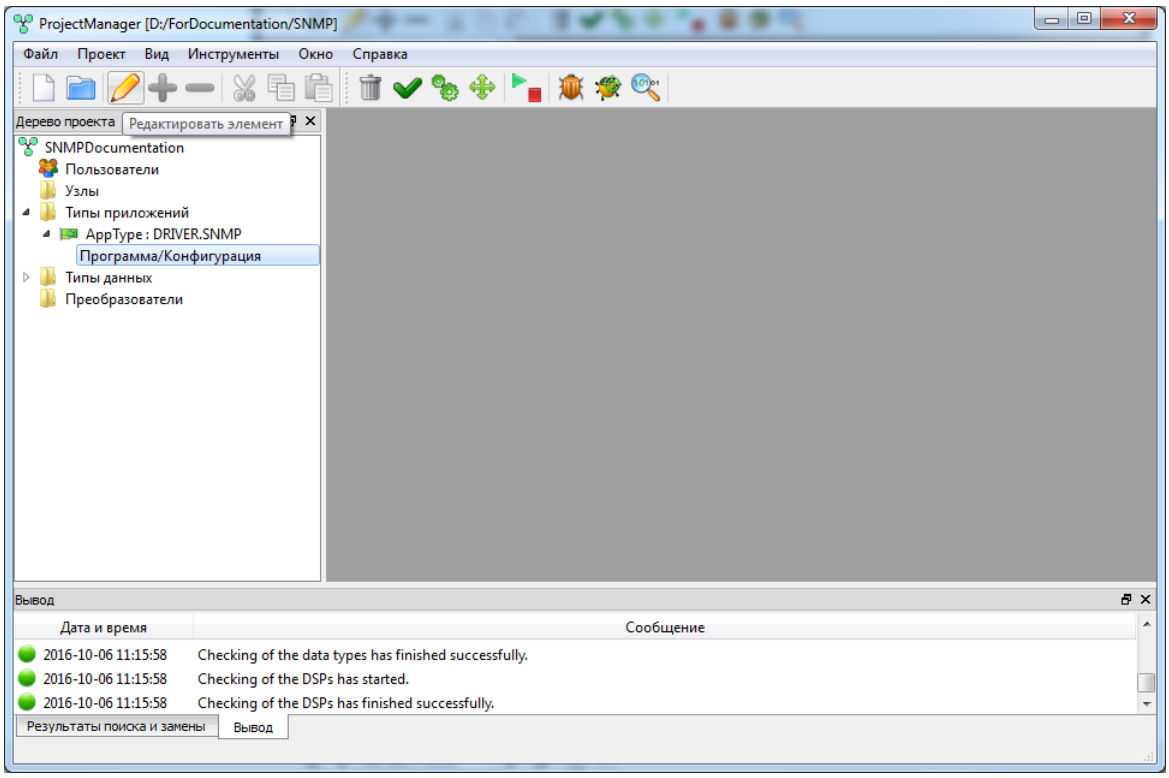


Рисунок 3.269 - Вызов редактора драйверов

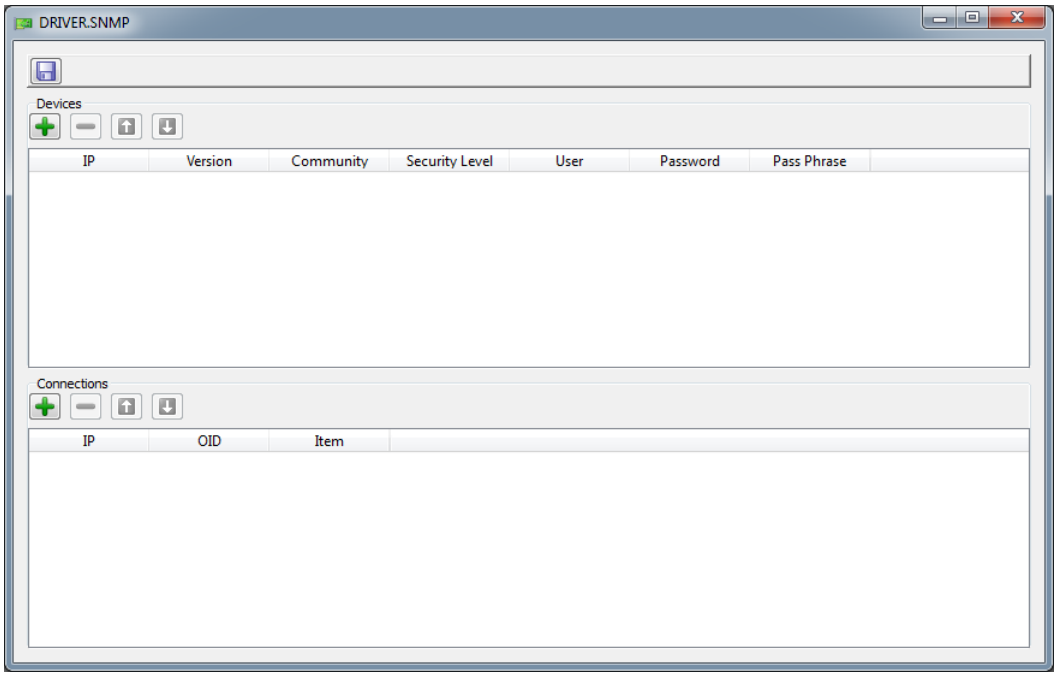


Рисунок 3.270 - Редактор драйвера SNMP

В верхней части располагается окно **Devices**. В данном окне выполняется настройка устройств, которые будут опрашиваться по протоколу SNMP.

В нижней части располагается окно **Connections**. В данном окне описываются связи сигналов SCADA-системы "Соната" и переменных протокола SNMP с опрашиваемого устройства.

Для работы со списком узлов(окно **Devices**) используются кнопки:

- добавить узел;

- удалить узел;

- поднять строку вверх;

- опустить строку вниз.

В поле **IP** указывается ip-адрес устройства, которое будет опрашиваться по протоколу SNMP. Для устройства указываются версия SNMP в поле **Version**, уровень безопасности в поле **Security Level**, а также необходимые параметры авторизации: для версий SNMP 1 и 2c – поле **Community**, для версии SNMP 3 – имя в поле **User** и(или) пароли в поле **Password**, в зависимости от **Security Level**. В поле **Pass Phrase** указывается напоминание для пароля.

Для работы со списком переменных(окно **Connections**) используются кнопки:

- добавить переменную;

- удалить переменную;

- поднять строку вверх;

- опустить строку вниз.

Поле **OID** - это переменные драйвера SNMP, которые нужно смотреть в документации на драйвер. Поле **Item** - сигналы SCADA-системы "Соната", которые были добавлены в интерфейс приложения нашего драйвера на рис. 3.266 и рис. 3.267. Поле **IP** - это IP-адрес устройства, который был добавлен в верхнем окне **Devices**.

Если к **OID** переменной добавить слово **status**, то в этой переменной будет указан статус переменной. Если вместо переменной указать слово **status**, то переменная соответствует статусу устройства (см. статусы Приложение В). Нулевое значение статуса обозначает отсутствие ошибок.

Сохранение происходит автоматически при выходе из редактора и файл сохраняется в каталог, содержащий .int файл. Кроме того, можно сохранить файл при помощи кнопки в верхней части окна.

- сохранить внесённые изменения.

3.3.8.7.4. Драйвер SNMP - пример. Проверка работы каналов связи.

В проекте тестируются на работоспособность каналы связи и два устройства связи (MOXAEDS-408A-MM-CC: IP – 192.168.1.193 и IP – 192.168.1.134). Результаты отображаются на АРМе IP:192.168.1.93.

В окне Конфигурации проекта созданы глобальные сигналы, которые будут значениями каналов и статусами-состояниями устройств связи(см. рис. 3.271).

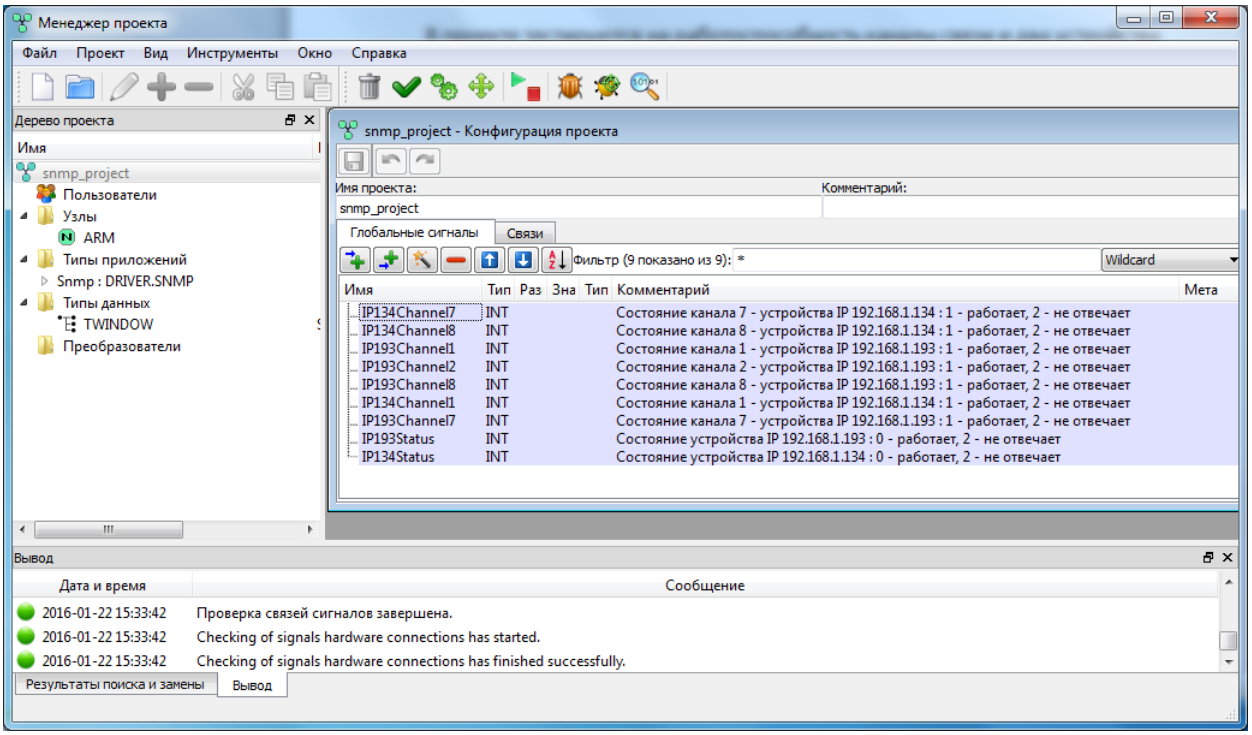


Рисунок 3.271 - Глобальные сигналы проекта

К примеру, IP134Channel1 - это канал 1 устройства связи с IP – 192.168.1.134, а IP134Status – это состояние самого устройства связи с IP – 192.168.1.134 (0 – работает, 2 – не отвечает). Аналогично остальные переменные.

Устройства связи должны поддерживать протокол SNMP.

Далее настраиваем приложение-драйвер SNMP на АРМе с IP – 192.168.1.93.

1. Создаем тип приложения и настраиваем его, как на рис. 3.272

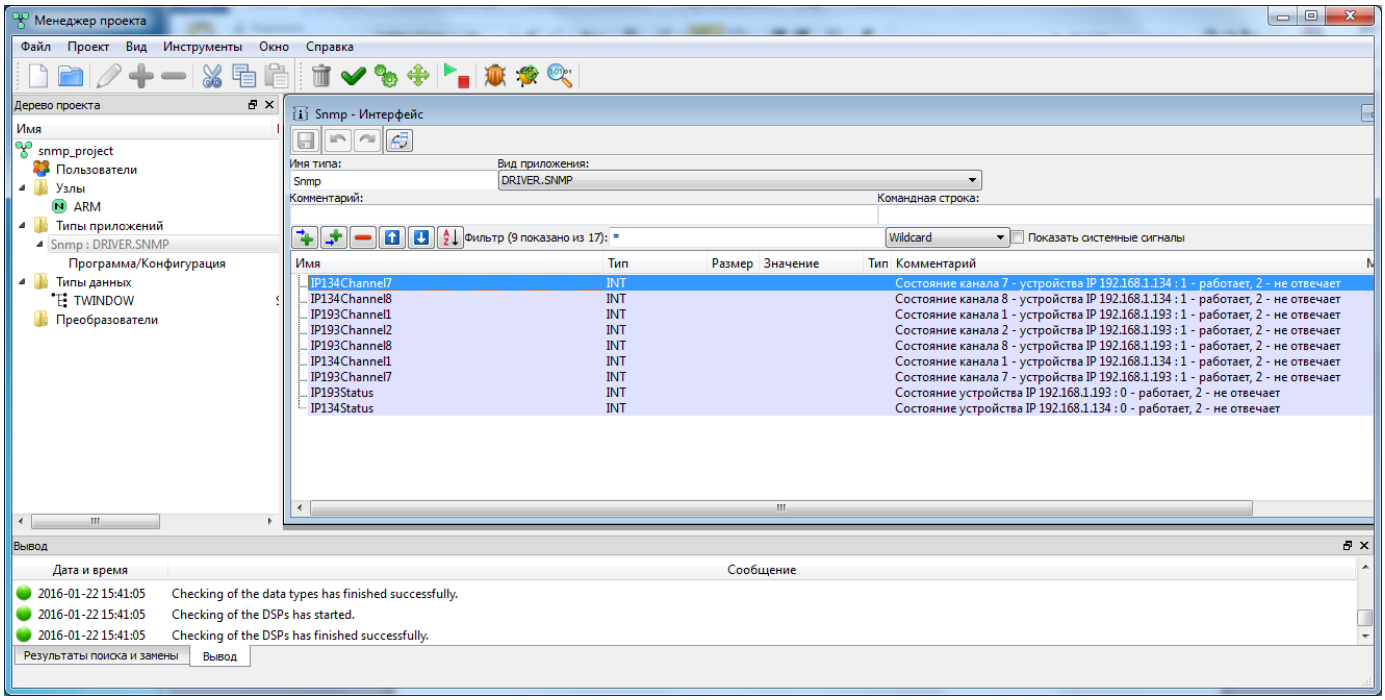


Рисунок 3.272 - Интерфейс приложения

2. Заходим в программу конфигурации приложения и настраиваем как на рис. 3.273

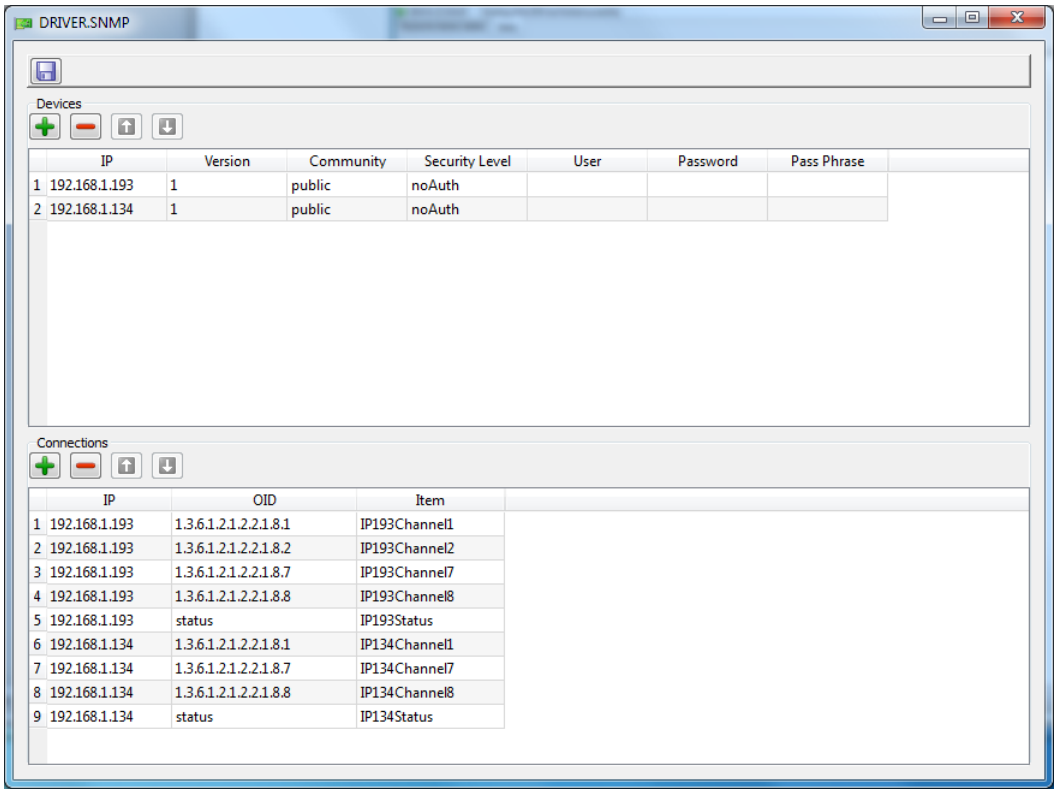


Рисунок 3.273 - Редактор приложения-драйвера SNMP

- **Devices** – устройства, с которыми будем общаться;

- **Version**, **Community**, **SecurityLevel** – настройки на SNMP устройствах (необходимо смотреть в документации на устройства);

Настраиваем **Connections**:

IP – это IP-адрес устройства;

OID–стандартные переменные протоколов SNMP, соответственно:

1 канал – 1.3.6.1.2.1.2.2.1.8.1

2 канал – 1.3.6.1.2.1.2.2.1.8.2

8 канал – 1.3.6.1.2.1.2.2.1.8.8

И т.д.

status – переменная статуса-состояния устройства (см. Приложение В);

Item – привязываем соответствующие сигналы SCADA-системы "Соната";

Наш драйвер настроен. Сохраняем.

3. Создаем узел, на котором будем всё контролировать (см. рис. 3.274)

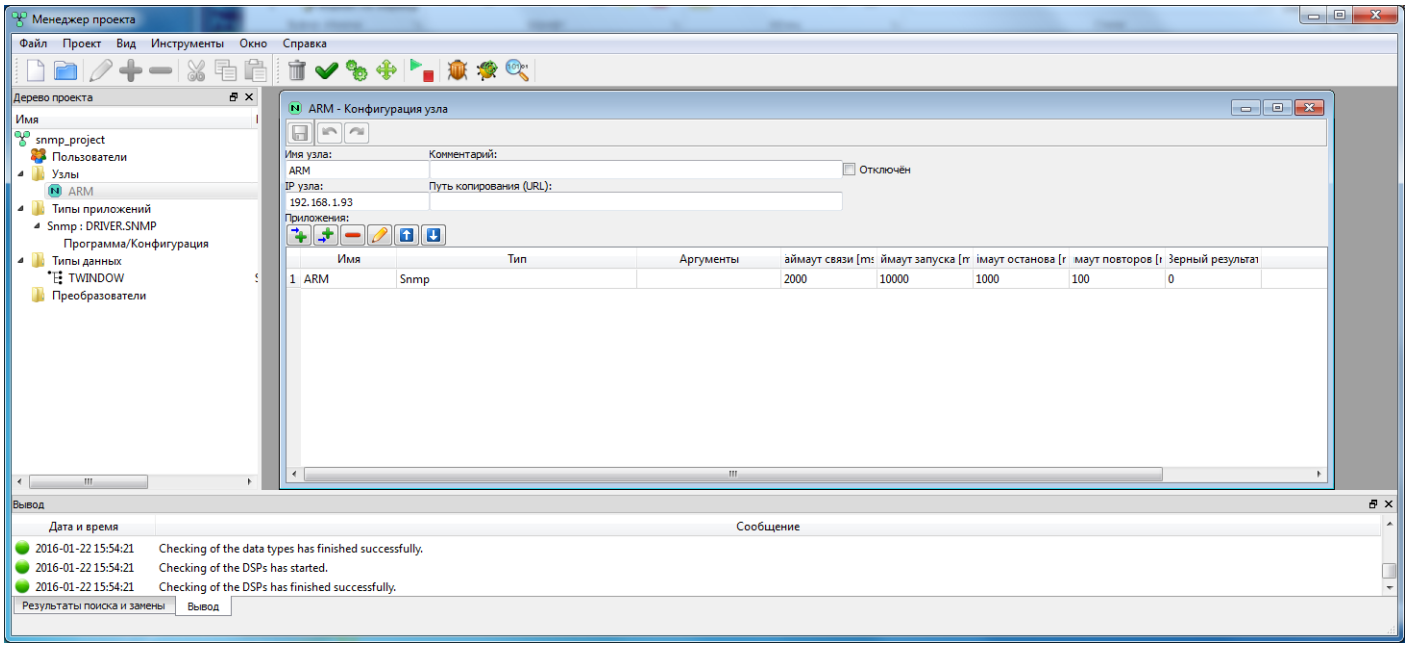


Рисунок 3.274 - Конфигурация узла

4. Проект готов. Осталось выполнить пункты **«Проверить проект»**, **«Скомпилировать»** и **«Разослать»**. Далее запускаем проект. (Это все соответствующие кнопки на верхней панели **Менеджера проектов**)

5. Для отображения результатов запускаем **«Просмотрщик сигналов»** (это значок **Увеличительное стекло** на верхней панели **Менеджера проектов**).

В открывшемся окне нажимаем «+»(см. рис. 3.275)

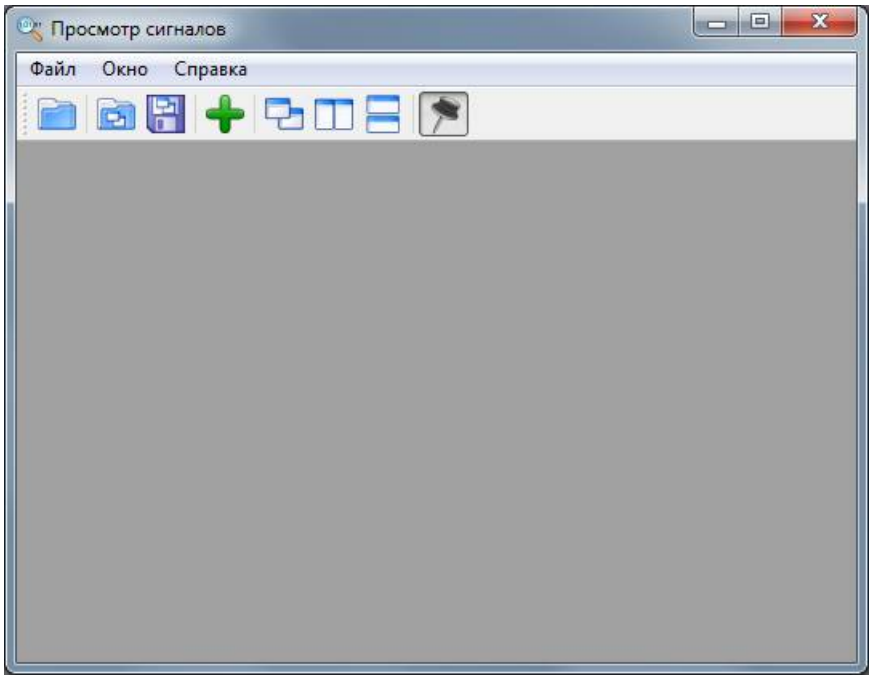


Рисунок 3.275 - Окно SignalViewer (Просмотрщик сигналов)

Далее нажимаем +, но слева ниже(см. рис. 3.276)

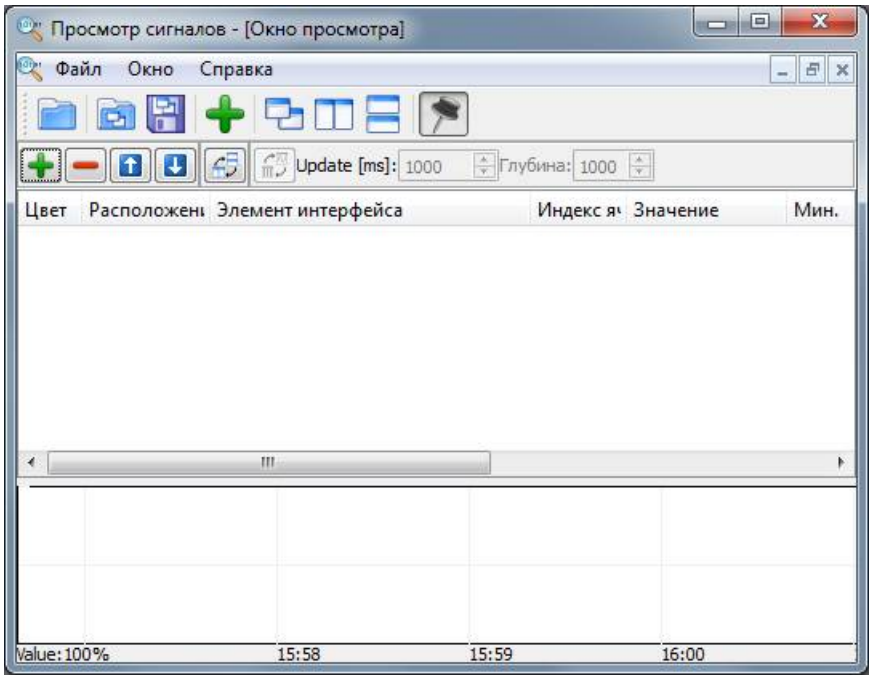


Рисунок 3.276 - Следующее окно Просмотрщика сигналов

В открывшемся окне выбираем IP– 192.168.1.93 и нажимаем кнопку **Получить (см. рис. 3.277)**

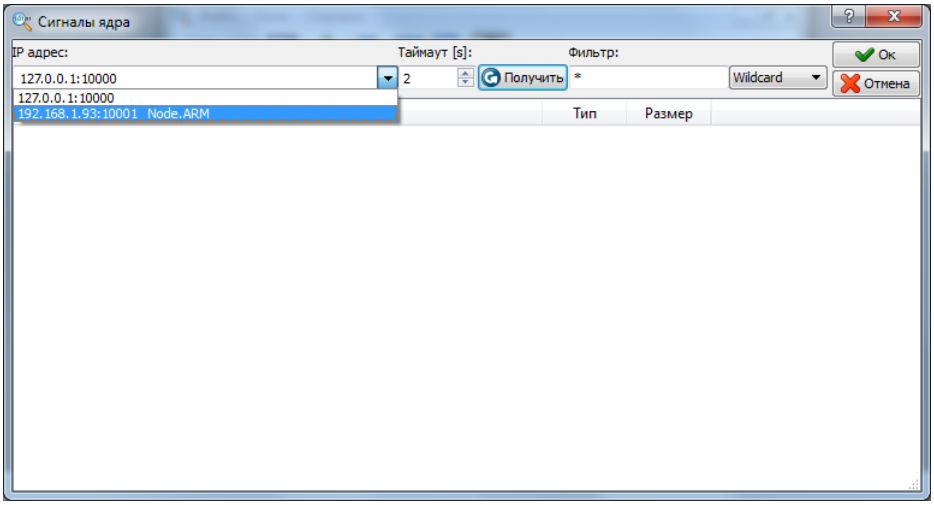
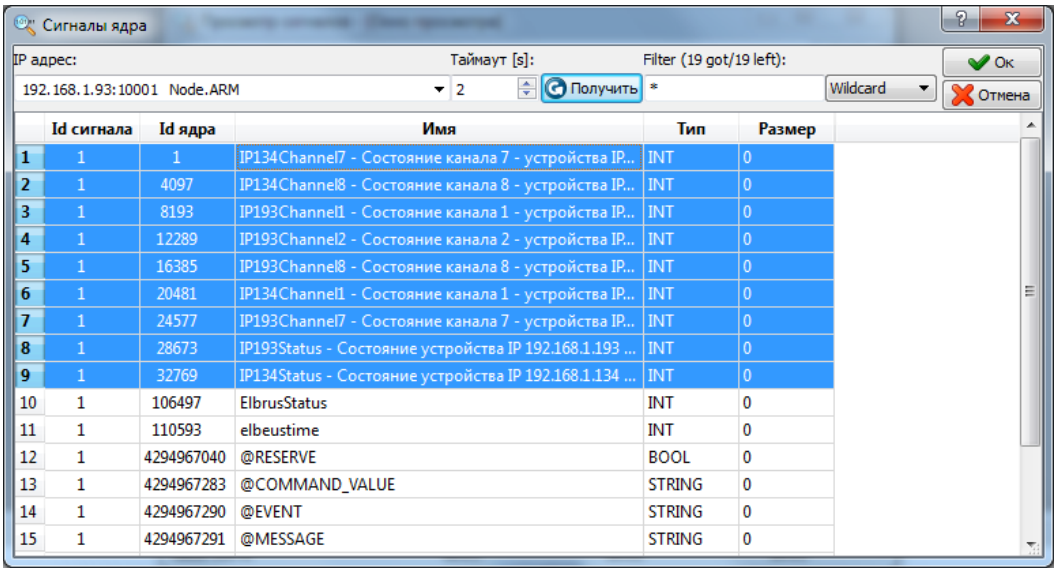


Рисунок 3.277 - Выбор узла в окне Просмотрщика сигналов

Далее выбираем нужные переменные и нажимаем **OK**(см. рис. 3.278)



	Id сигнала	Id ядра	Имя	Тип	Размер
1	1	1	IP134Channel7 - Состояние канала 7 - устройства IP...	INT	0
2	1	4097	IP134Channel8 - Состояние канала 8 - устройства IP...	INT	0
3	1	8193	IP193Channel1 - Состояние канала 1 - устройства IP...	INT	0
4	1	12289	IP193Channel2 - Состояние канала 2 - устройства IP...	INT	0
5	1	16385	IP193Channel8 - Состояние канала 8 - устройства IP...	INT	0
6	1	20481	IP134Channel1 - Состояние канала 1 - устройства IP...	INT	0
7	1	24577	IP193Channel7 - Состояние канала 7 - устройства IP...	INT	0
8	1	28673	IP193Status - Состояние устройства IP 192.168.1.193 ...	INT	0
9	1	32769	IP134Status - Состояние устройства IP 192.168.1.134 ...	INT	0
10	1	106497	ElbrusStatus	INT	0
11	1	110593	elbeustime	INT	0
12	1	4294967040	@RESERVE	BOOL	0
13	1	4294967283	@COMMAND_VALUE	STRING	0
14	1	4294967290	@EVENT	STRING	0
15	1	4294967291	@MESSAGE	STRING	0

Рисунок 3.278 - Выбор переменных для просмотра в окне Просмотрщика сигналов

Получаем нужные переменные, которые говорят о состоянии каналов и устройств.

На рис. 3.279 можно видеть, что: (в данном примере нет реально подключенных устройств)

- IP193Status и IP134Status равны 2 – это значит, эти устройства не отвечают и все другие переменные этих устройств (переменные каналов) не надо анализировать, устройство просто не видно и в переменной канала отображается последнее значение до выхода всего устройства из рабочего состояния - отключения.

IP134Channel1 – мы видим значение 0, это инициализирующее значение (имеет смысл поставить его 2 – не отвечает, в процессе создания **Глобальных переменных**)

Соответственно, если бы устройство было на связи, то мы бы увидели IP193Status и IP134Status, равные 1. В этом случае о состоянии каналов нас проинформируют наши переменные:

IP134Channel1 = 1 – канал работает;

IP134Channel1 = 2 – канал не отвечает.

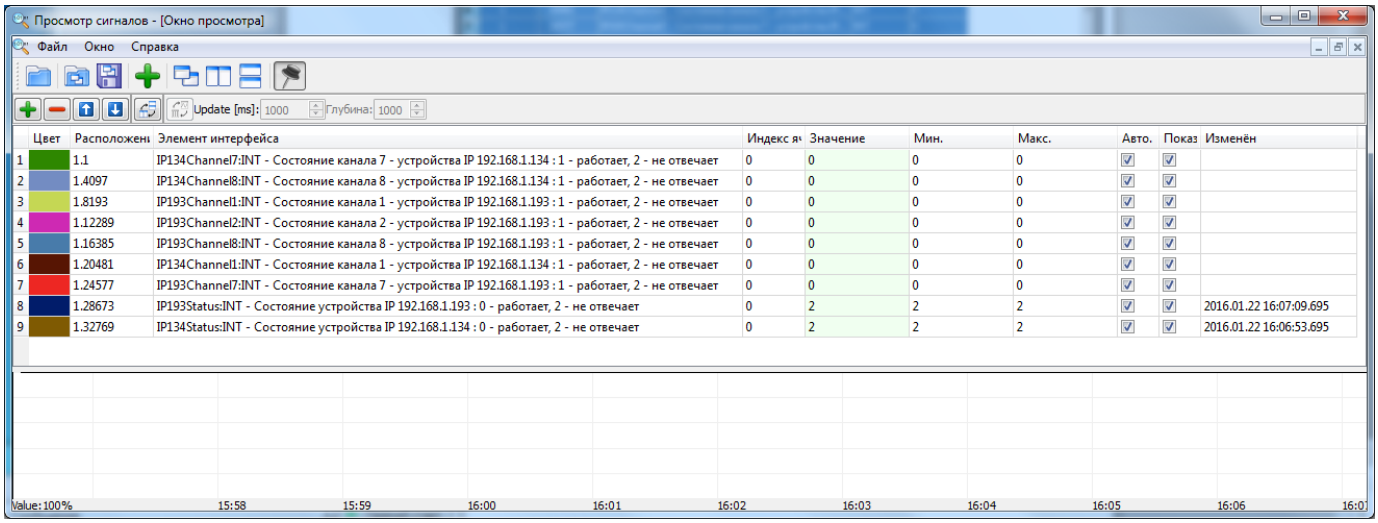


Рисунок 3.279 - Работа в Просмотрщике сигналов

3.3.8.8. Описание работы с драйвером Sonet.Failover

Устройство БПР (блок переключения на резерв) предназначено для обеспечения бесперебойной работы системы путем переключения основного процессорного модуля на резервный в случае возникновения неполадок.

3.3.8.8.1. Создание приложения-драйвера Sonet.Failover

Для создания приложения-драйвера Sonet.Failover сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.280 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** (необходимо выбрать **DRIVER.SONET.FAILOVER**).

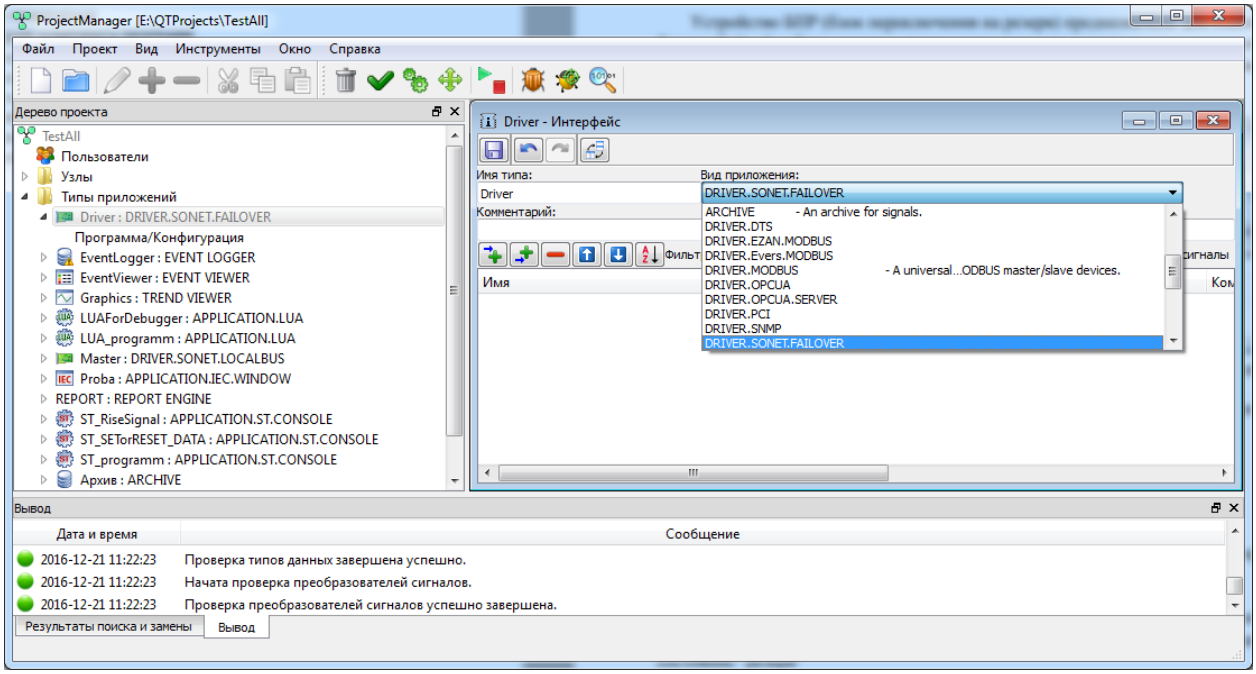


Рисунок 3.280 - Интерфейс приложения-драйвера Sonet.Failover

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.281) – для подтверждения изменения необходимо нажать кнопку Да.

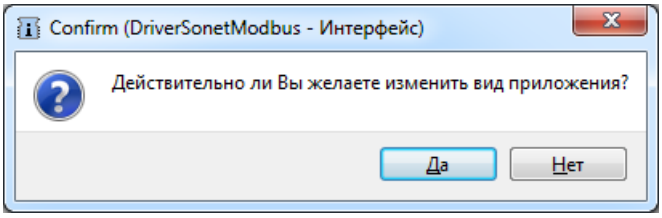


Рисунок 3.281 - Окно подтверждения для изменения вида типа приложения

3.3.8.8.2. Создание интерфейса приложения-драйвера Sonet.Failover

После выбора вида приложения-драйвера необходимо добавить в интерфейс приложения сигналы, значения которых будут использоваться в дальнейшем для настройки приложения. Процесс добавления сигналов в интерфейс приложения подробно описан ранее (см. раздел 3.2.2.4.3).

На рис. 3.282 в интерфейс приложения-драйвера добавлены три сигнала LSTATUS, LENABLE и LRESERVED, которые будут использованы в этом приложении-драйвере.

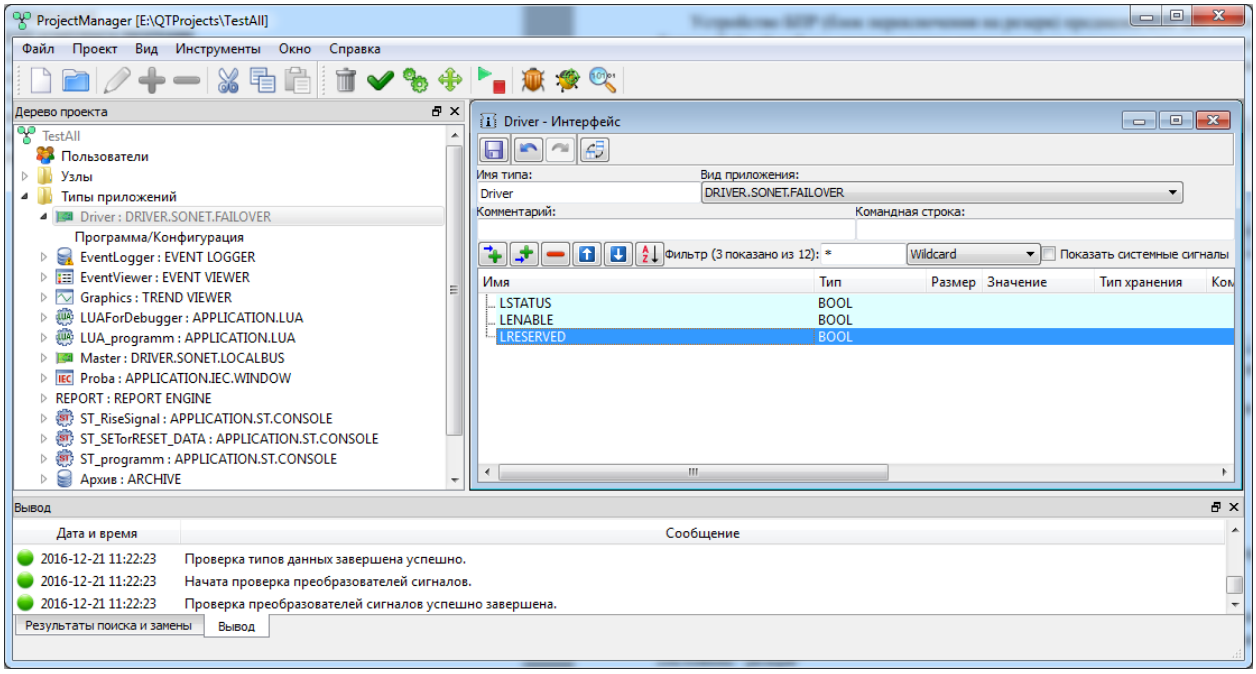


Рисунок 3.282 - Настроенный интерфейс приложения-драйвера Sonet.Failover

После добавления сигналов в интерфейс приложения-драйвера необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Интерфейс приложения настроен и теперь необходимо провести настройки самого приложения-драйвера Sonet.Failover.

3.3.8.8.3. Настройка приложения-драйвера Sonet.Failover

Для настройки приложения-драйвера Sonet.Failover необходимо вызвать специализированный редактор. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

Главное окно редактора приложения-драйвера Sonet.Failover представлено на рис. 3.283.

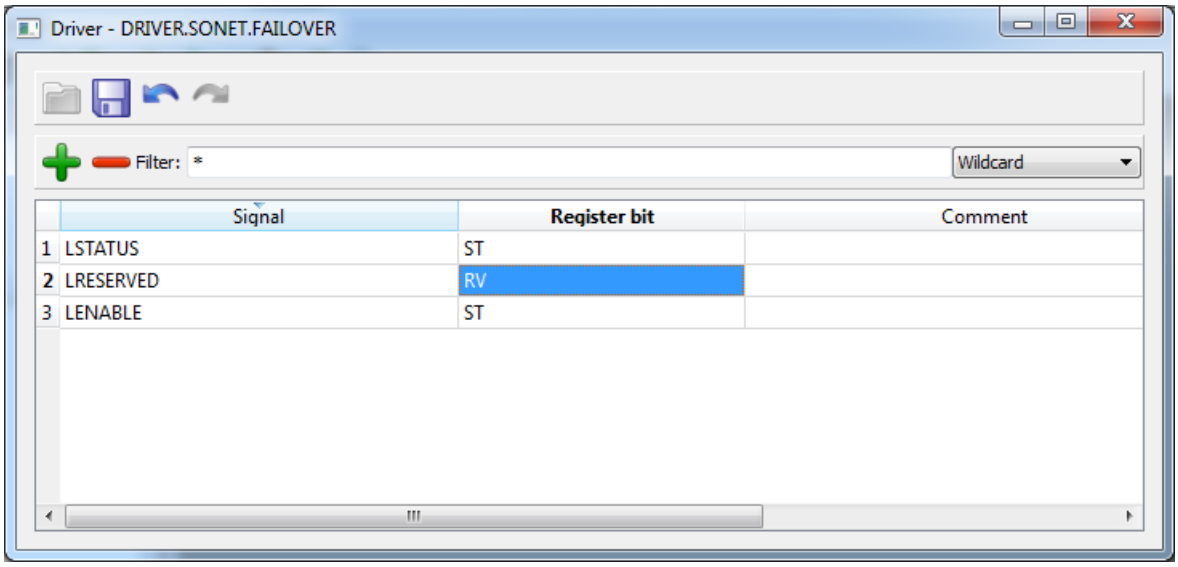


Рисунок 3.283 - Главное окно редактора приложения-драйвера Sonet.Failover

В верхней части окна расположена панель инструментов со следующими кнопками:

- открыть файл конфигурации драйвера;

- сохранить внесённые в данный элемент проекта изменения;

- отменить предыдущее действие;

- повторить отменённое действие.

Ниже находятся кнопки для изменения конфигурации:

- добавить сигнал;

- удалить сигнал. Будут удалены все выделенные строки;

Filter – поле для фильтрации элементов интерфейса. Выпадающий список для выбора вариантов фильтрации содержит следующие варианты:

Wildcard – фильтрация по шаблону, где ? – любой символ, * – любая последовательность символов, [...] – любой символ из набора;

Regular expression - фильтрация по регулярному Perl-like выражению;

Fixed string - точная формулировка.

Ниже расположена таблица конфигурации драйвера, состоящая из трех столбцов:

Signal – переменная (сигнал), которая привязывается к определенному биту;

Register bit – управляющий бит. У устройства БПР есть 3 управляющих бита:

ST – возвращает информацию об исправности БПР. Неисправным БПР считается, если за 1 секунду не было считано ни одного значения, равного “1” бита **ST**;

RV – этот бит показывает, является ли контроллер основным или резервным (0 – основной, 1- резервный);

EN – запись “1” в этот бит разрешает работу с БПР, “0” – запрещает.

Кроме основных значений управляющих битов, есть возможность добавить в конфигурацию инвертированные биты (ST-inv, MS-inv, EN-inv).

Comment – комментарий.

На рис. 3.283 проведены следующие настройки приложения-драйвера Sonet.Failover:

- сигнал LSTATUS - значение, равное 1, сообщает, что устройство БПР исправно, значение, равное 0, что неисправно;

- сигнал LRESERVED - значение, равное 1, сообщает, что узел, на котором работает данное приложение-драйвер, находится в резерве, значение, равное 0, что узел является основным;

- сигнал LENABLED - данным сигналом можно управлять из других приложений (по умолчанию оно равно 1), выставляя значение 1 - приложение-драйвер Sonet.Failover работает с

устройством БПР, выставляя значение 0 - приложение-драйвер Sonet.Failover перестаёт работать с устройством БПР.

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер Sonet.Failover настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на нужный вам узел проекта (см. раздел 3.2.2.5.1).

3.3.8.9. Описание работы с приложением-драйвером Sonet.Localbus

Приложение-драйвер Sonet.Localbus применяется для работы с цифровыми и аналоговыми модулями ввода/вывода под управлением микропроцессорного модуля Sonet.

3.3.8.9.1. Создание приложения-драйвера Sonet.Localbus

Создание приложения-драйвера Sonet.Localbus осуществляется аналогично другим приложениям-драйверам SCADA-системы "Соната" (см. раздел 3.2.2.4).

На рис. 3.284 изображено стандартное окно интерфейса приложения SCADA-системы "Соната". Чтобы данное приложение стало приложением-драйвером Sonet.Localbus, необходимо в поле **Вид приложения** выбрать **DRIVER.SONET.LOCALBUS** и в поле **Имя типа** написать удобное для дальнейшего использования в проекте имя данного приложения (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

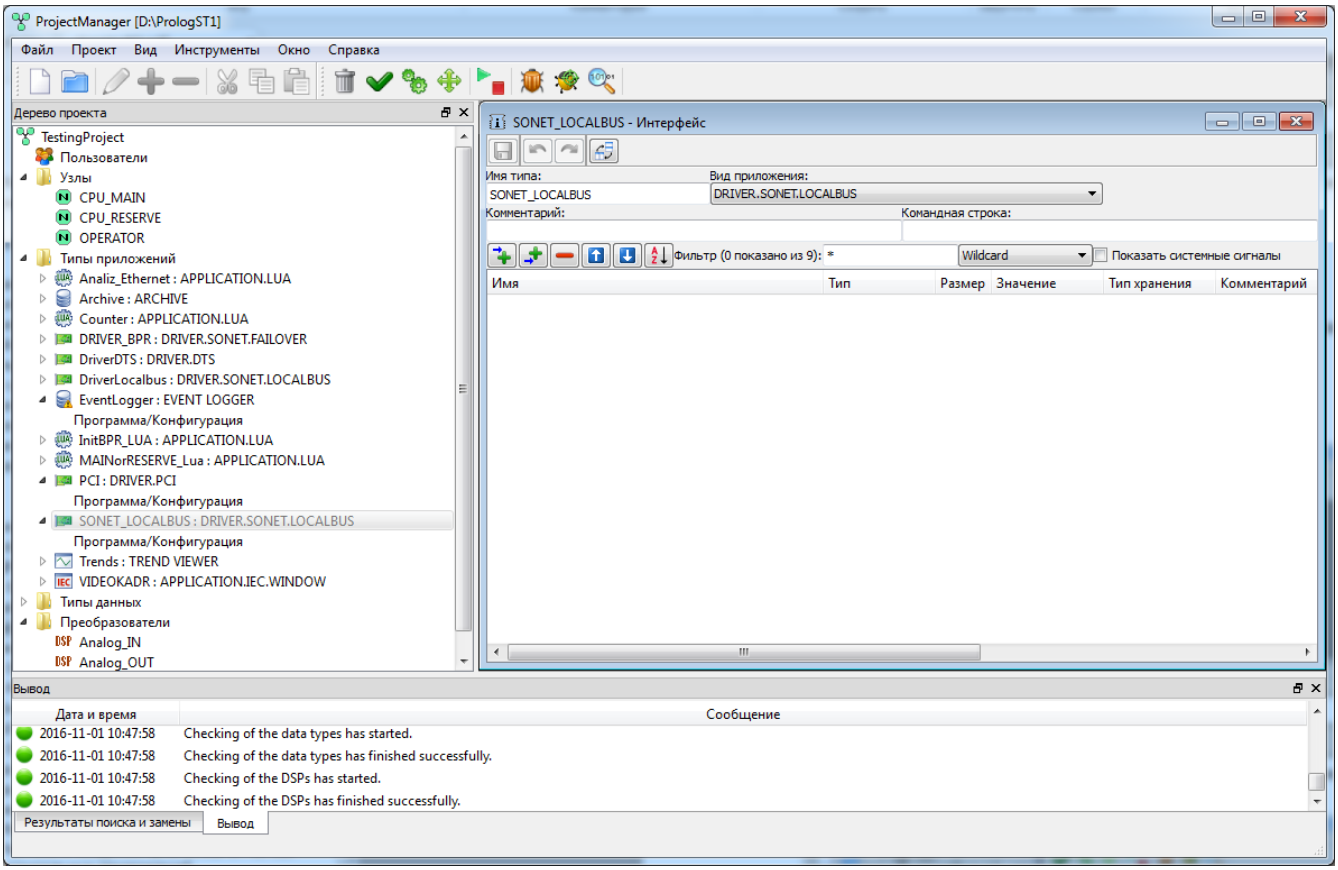


Рисунок 3.284 - Окно редактора интерфейса приложения

3.3.8.9.2. Создание интерфейса приложения-драйвера Sonet.Localbus

После выбора вида приложения-драйвера необходимо добавить в интерфейс приложения сигналы, значения которых будут связаны с каналами модулей ввода/вывода. Процесс добавления сигналов в интерфейс приложения подробно описан в раздел 3.2.2.4.3. На рис. 3.285 в интерфейс приложения-драйвера добавлено 9 сигналов: **DINPUT1** и **DINPUT2** - сигналы дискретного ввода, **DOUTPUT1** и **DOUTPUT2** - сигналы дискретного вывода, **AINPUT1** - сигнал аналогового ввода, **AOUTPUT1** - сигнал аналогового вывода, сигнал **INFO** - состояние микропроцессорного модуля, **Module_Status** - состояние одного из модулей (см. далее) и **Canal_Status** - состояние одного из каналов (см. далее). Значения кодов и статусов для сигналов **INFO**, **Module_Status** и **Canal_Status** смотри в Приложение В.

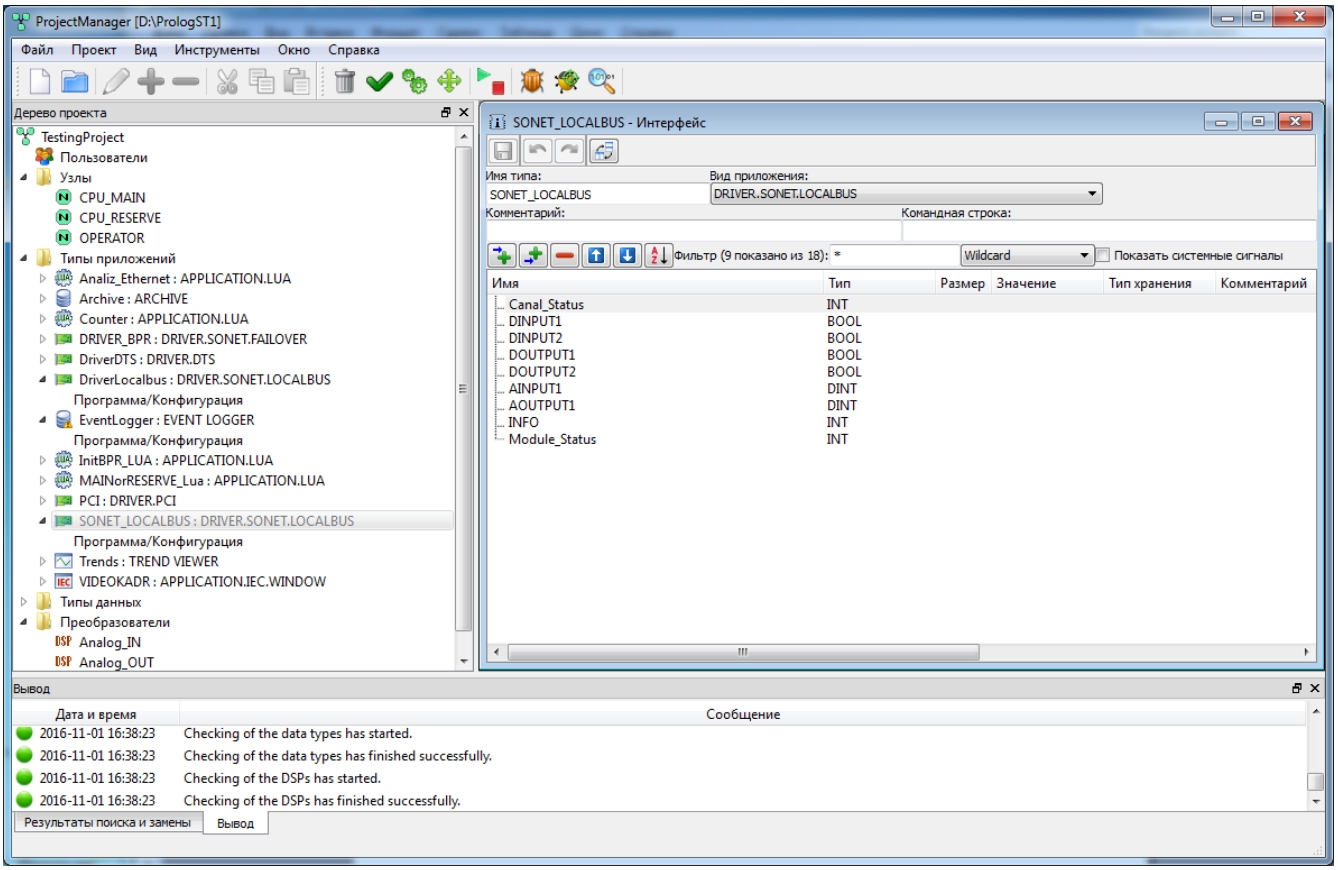


Рисунок 3.285 - Интерфейс приложения-драйвера Sonet.Localbus

После добавления сигналов в интерфейс приложения-драйвера необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Интерфейс приложения настроен и теперь необходимо настроить связь сигналов из интерфейса приложения с каналами модулей ввода/вывода.

3.3.8.9.3. Привязка сигналов приложения-драйвера Sonet.Localbus к каналам ввода/вывода

Для привязки сигналов из интерфейса приложения-драйвера к каналам модулей ввода/вывода необходимо вызвать специализированный редактор приложения-драйвера Sonet.Localbus. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4). На рис. 3.286 изображено окно специализированного редактора (далее просто редактор) приложения-драйвера Sonet.Localbus.

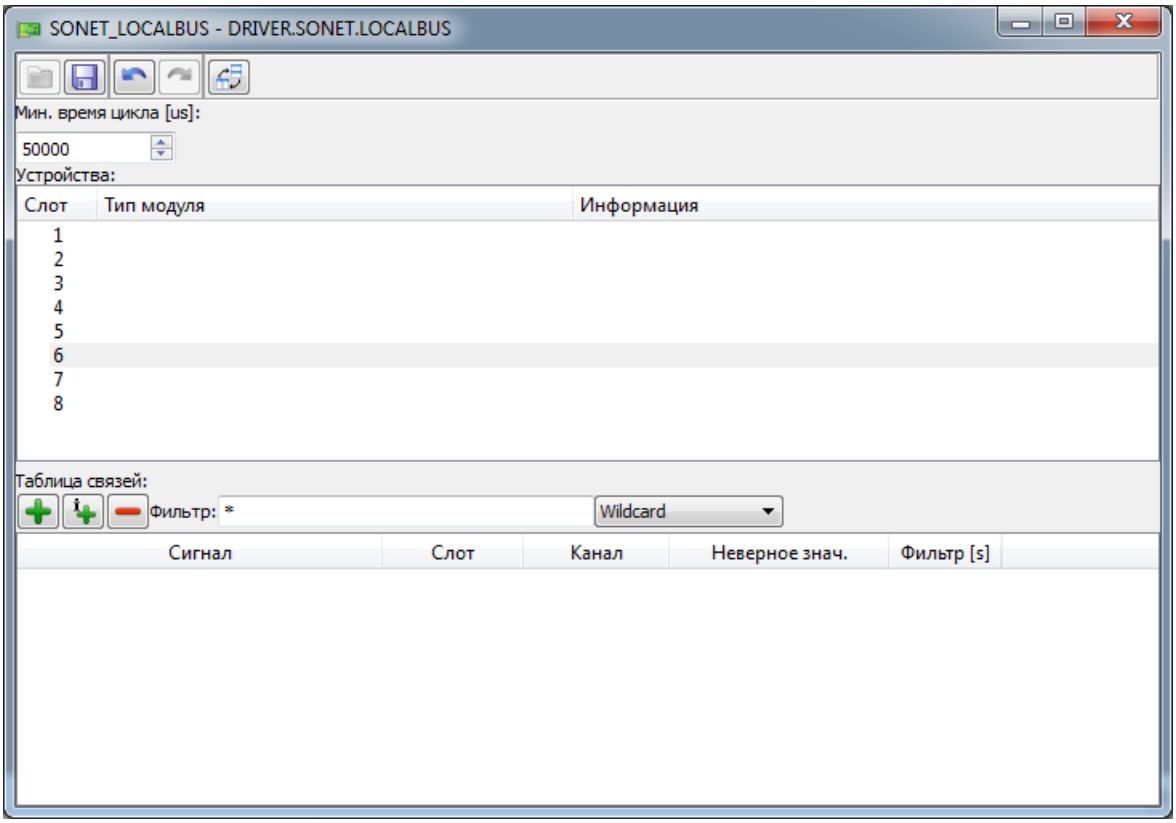


Рисунок 3.286 - Специализированный редактор приложения-драйвера Sonet.Localbus

В верхней части окна редактора находится строка с полным именем приложения, состоящая из имени типа и вида приложения. В данном случае **SONET_LOCALBUS**(имя типа) - **DRIVER.SONET.LOCALBUS**(вид приложения). Правее располагаются стандартные для оконных приложений кнопки управления окном  , имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

Ниже располагаются следующие кнопки и поля ввода:

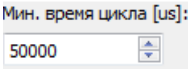
 ("Открыть") - открыть готовый файл с конфигурацией;

 («Сохранить») – сохранить внесённые в редакторе изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

 («Изменить ориентацию») – изменить ориентацию окон;

 - минимальное время цикла опроса.

Далее расположено окно **Устройства**, которое состоит из следующих столбцов:

- **Слот** - порядковый номер разъема для модуля ввода/вывода;
- **Тип модуля** - известные SCADA-системе "Соната" типы модулей (см. Приложение Е);
- **Информация** - информация о выбранном модуле.

Ниже располагается **Таблица связей**, которая состоит из следующих столбцов и элементов управления:

 ("Добавить элемент") - добавить новую строку в **Таблицу связей**;

 ("Импортировать элементы из интерфейса") - добавить все сигналы из интерфейса приложения-драйвера;

 ("Удалить элемент") - удалить выбранный элемент из таблицы связей;

- поле настройки фильтров отображения сигналов;

- **Сигнал** - сигнал из интерфейса приложения-драйвера;
- **Слот** - разъем для модуля ввода/вывода;
- **Канал** - номер канала на модуле ввода/вывода;
- **Неверное знач.** - значение, которое будет принимать сигнал, когда канал неисправен;
- **Фильтр [s]** - постоянная времени фильтра, выраженная в секундах.

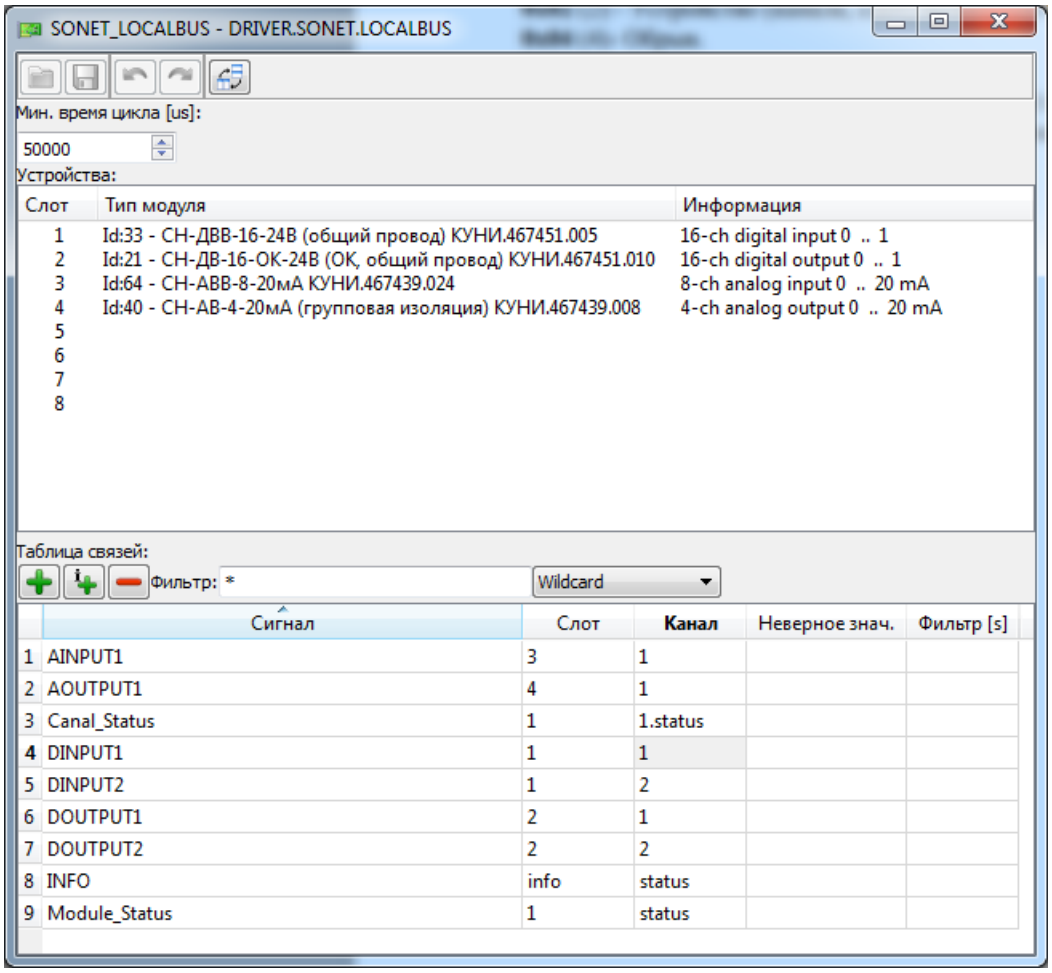
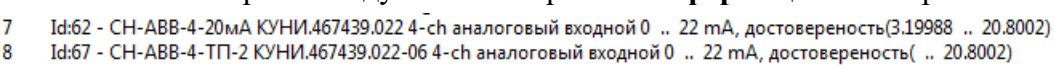


Рисунок 3.287 - Настроенное окно редактора приложения-драйвера Sonet.Localbus

Настройка приложения-драйвера Sonet.Localbus начинается с настроек в окне **Устройства**.

В столбце **Слот** по порядку с 1 до 8 пронумерованы позиции модулей ввода/вывода в общей корзине с микропроцессорным модулем **Sonet** (8 - максимальное количество модулей ввода/вывода в одной корзине). На рис. 3.287 в первой позиции-слоте находится цифровой модуль на 16 входов, во второй позиции-слоте находится цифровой модуль на 16 выходов, в третьей позиции-слоте находится аналоговый модуль на 8 входов и в четвёртой позиции-слоте находится аналоговый модуль на 4 выхода. Каждый модуль оснащён своей маркировкой, в соответствии с которой в столбце **Тип модуля** выбирается нужное значение (см. Приложение Е). После выбора типа модуля в строке **Информация** отобразится дополнительная информация о выбранном модуле.

ВАЖНО! У некоторых модулей в строке **Информация** отобразятся следующие данные



. Запись **достоверность** сообщает о том, что каналы данного модуля в указанном диапазоне (от 3.19988 до 20.8002 или у другого модуля .. до 20.8002, что значит левая граница достоверности равна левой

границе диапазона каналов модуля, а диапазон написан перед достоверностью), будут считаться исправными и у получаемых данных будет стоять признак достоверности. Вне данного диапазона каналы будут считаться неисправными и данным, получаемым по ним, будет выставляться признак недостоверности (**НОПР**). Данные диапазоны считаются диапазонами по умолчанию. Но в **Таблице связей** есть два столбца для настройки данных диапазонов вручную: **Нижняя граница** и **Верхняя граница** (см. рис. 3.288). Границы диапазонов достоверности указываются в единицах измерения канала.

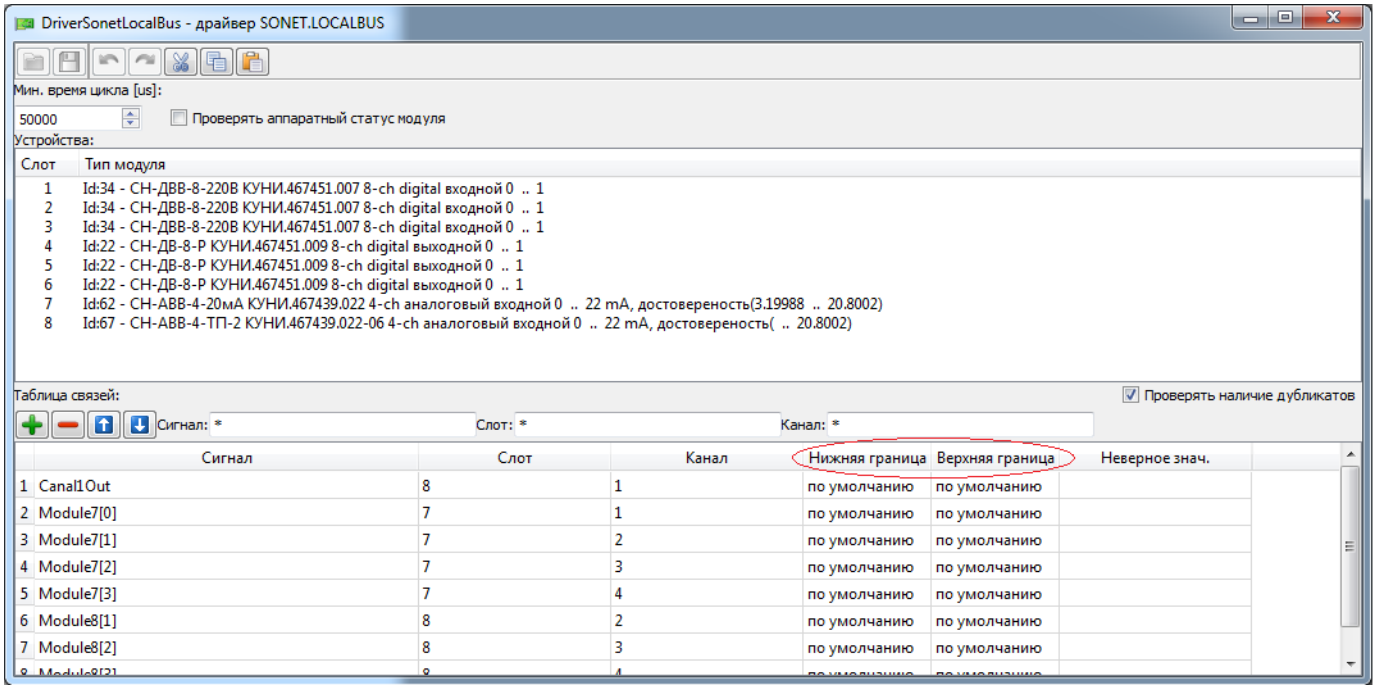


Рисунок 3.288 - Окно Настроек драйвера Sonet.Localbus - поля настройки диапазонов

Далее нужно настроить связи сигналов SCADA-системы "Соната" с каналами модулей ввода/вывода. Данные настройки проводятся в **Таблице связей**.

С помощью кнопок  ("Добавить элемент") и  ("Импортировать элементы из интерфейса") в столбец **Сигнал** добавляем нужные сигналы из интерфейса приложения-драйвера. В столбце **Слот** нужно выбрать соответствующий номер позиции-слота, в которой, в соответствии с ранее настроенным окном **Устройства**, находится нужный модуль ввода/вывода. Если выбрать в столбце **Слот** значение **info**, то данный сигнал будет сообщать о состоянии микропроцессорного модуля в данной корзине. В столбце **Канал** выбирается номер канала ввода/вывода и так же есть дополнительные значения, которые рассмотрим на примере (см. рис. 3.287).

Для сигнала **AINPUT1** выставлены значения **Слот** = 3 - это аналоговый модуль на 8 входов и **Канал** = 1 - это первый канал данного модуля. Аналогично настроены сигналы **AOUTPUT1**, **DINPUT1**, **DINPUT2**, **DOUTPUT1** и **DOUTPUT2**.

Для сигнала **Canal_Status** выставлены настройки **Слот** = 1 - это цифровой модуль на 16 входов и **Канал** = **1.status** - это статус первого канала данного модуля. Статусы каналов и модулей описаны в Приложение В.

Для сигнала **INFO** выставлены значения **Слот** = **info** - это получение информации о микропроцессорном модуле и **Канал** = **status** - это статус модуля, а как видно по значению **Слот** = **info**, то это статус микропроцессорного модуля. Так же значение **Канал** для значения **Слот** = **info**, может быть равно **temp** - это значение температуры микропроцессорного модуля.

Для сигнала **Module_Status** выставлены значения **Слот** = 1 - это цифровой модуль на 16 входов и **Канал** = **status** - это статус модуля, соответственно получаем информацию о работоспособности модуля в первом слоте.

В столбце **Неверное знач.** можно выставить значение для сигнала, которое он всегда будет принимать, в случае любой неисправности канала.

Для дискретных сигналов в столбце **Фильтр [s]** настраивается фильтрация от помех, т.е. значение времени в секундах, в пределах которого изменения сигнала не регистрируются (считаются помехой). Для аналоговых сигналов это константа времени для апериодического фильтра первого порядка.

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер Sonet.Localbus настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на узел проекта, который является микропроцессорным модулем **Sonet** с соответствующей настройкам корзиной модулей ввода/вывода (см. раздел 3.2.2.5.1).

3.3.8.10. Описание работы с приложением-драйвером SonetModbus

Приложение-драйвер SonetModbus применяется для объединения нескольких микропроцессорных модулей Sonet и их модулей ввода/вывода в одно целое.

3.3.8.10.1. Создание приложения-драйвера SonetModbus

Для создания приложения-драйвера SonetModbus сначала необходимо создать новое приложение (см. раздел 3.2.2.4).

Далее необходимо изменить имя типа приложения и вид приложения (см. раздел 3.2.2.4.3).

На рис. 3.289 проведены изменения в поле ввода **Имя типа** и в поле выбора **Вид приложения** (необходимо выбрать **DRIVER.SONET.MODBUS**).

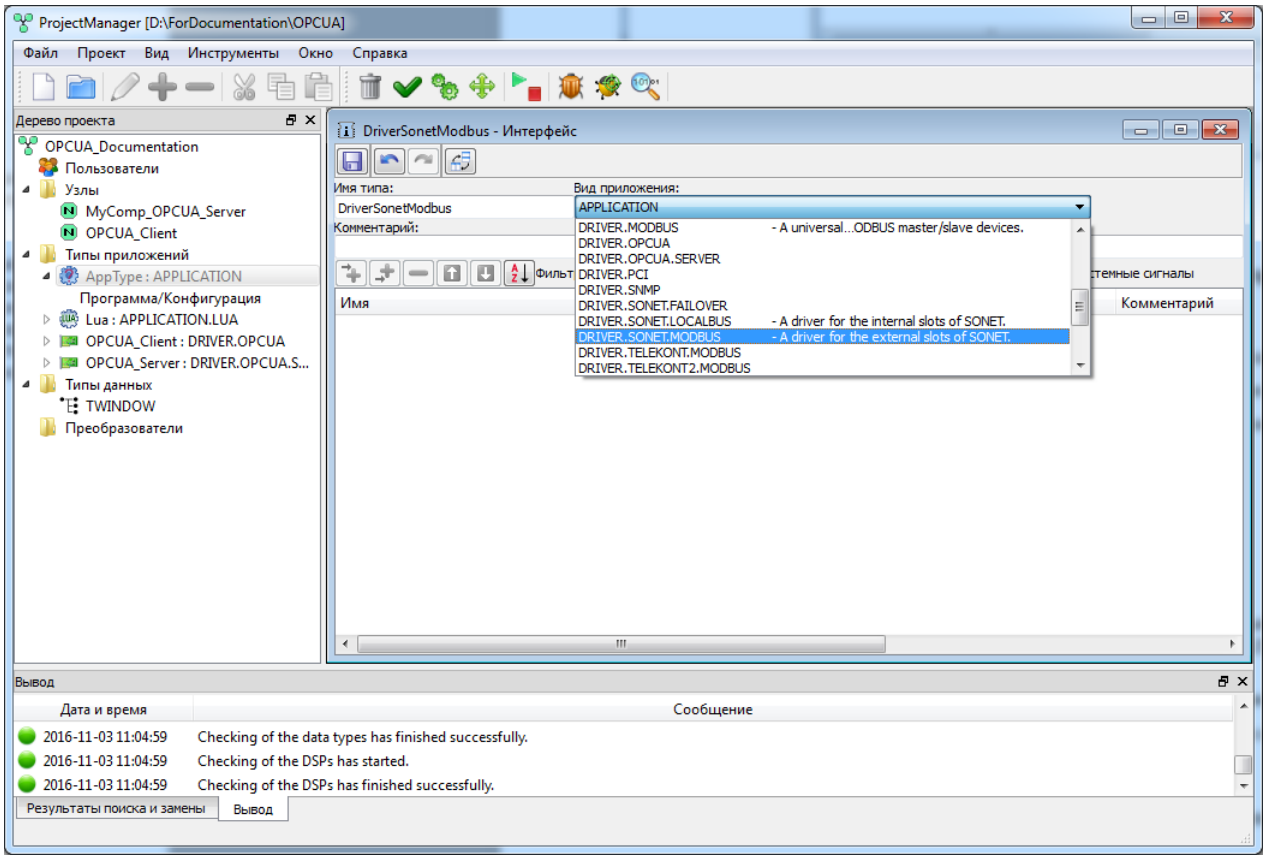


Рисунок 3.289 - Изменение имени типа и вида приложения

После изменения вида типа приложения на экран будет выведено окно подтверждения (см. рис. 3.290) – для подтверждения изменения необходимо нажать кнопку Да.

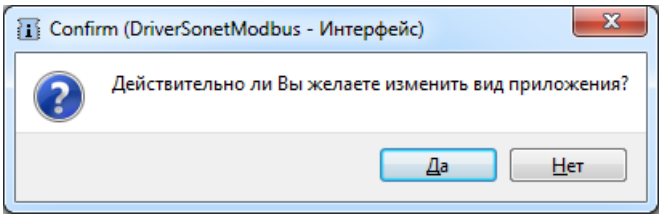


Рисунок 3.290 - Окно подтверждения для изменения вида типа приложения

Изменения следует зафиксировать с помощью кнопки («Сохранить») в окне редактирования интерфейса типа приложения (см. рис. 3.291).

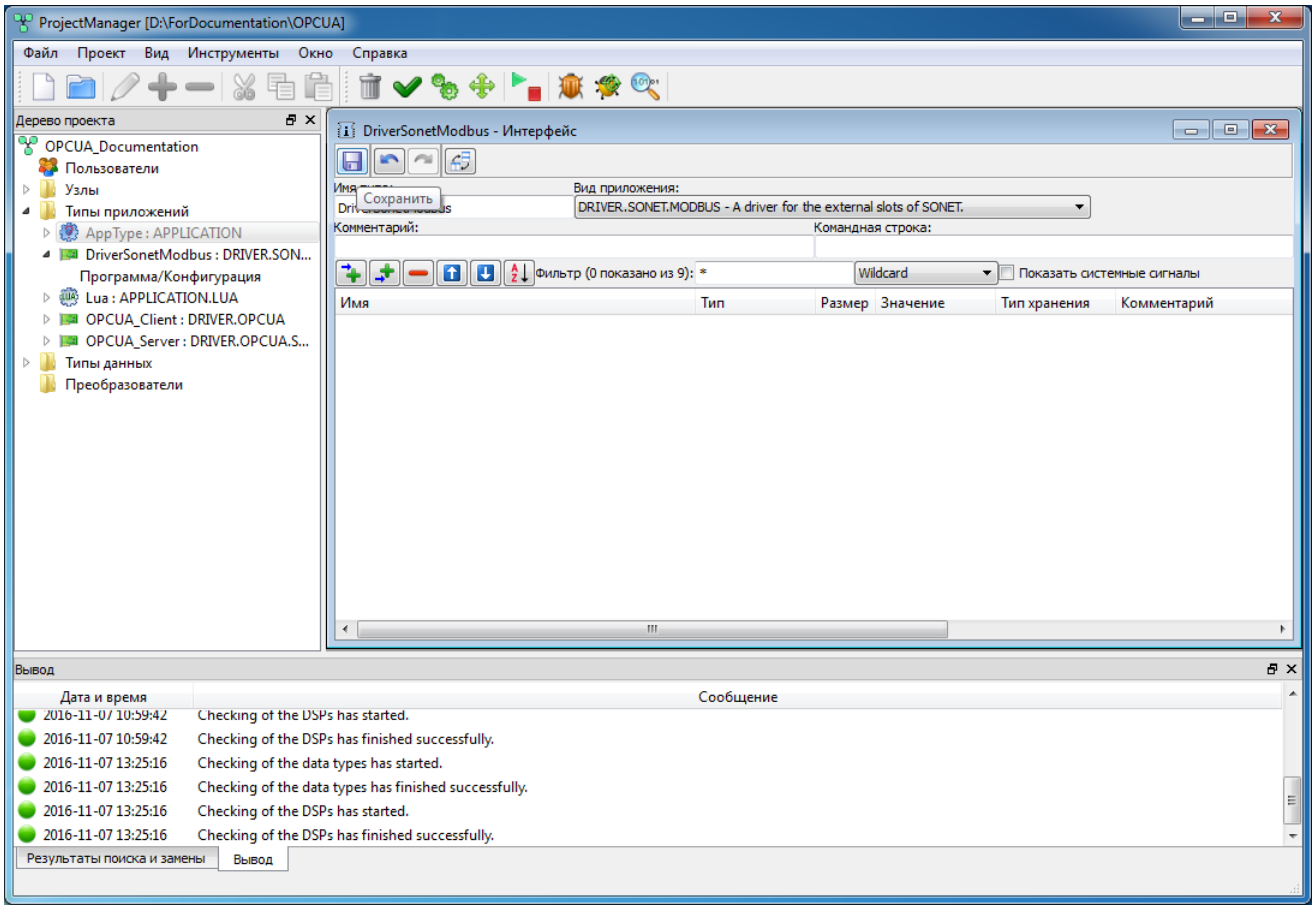


Рисунок 3.291 - Сохранение изменений вида типа приложения

3.3.8.10.2. Создание интерфейса приложения-драйвера SonetModbus

После выбора вида приложения-драйвера необходимо добавить в интерфейс приложения сигналы, значение которых будут связаны с каналами модулей ввода/вывода. Для этого нужно нажать одну из кнопок редактирования таблицы интерфейса:  («Вставить сверху») или  («Вставить снизу») – на экране появится окно выбора (см. рис. 3.292), в котором нужно выбрать вариант Локальный или Глобальный (подробно см. раздел 3.2.2.4.3.2).

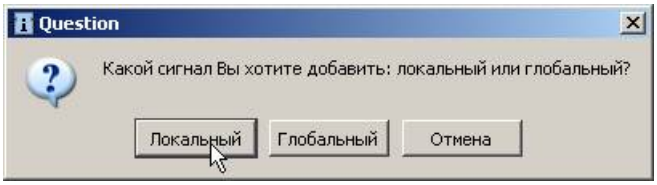


Рисунок 3.292 - Выбор признака Локальный/Глобальный
при добавлении сигнала в интерфейс приложения

Локальный сигнал добавляется с помощью кнопок редактирования таблицы интерфейса ( («Вставить сверху») и  («Вставить снизу» – см. рис. 3.293). Имя новому сигналу задается по шаблону , для изменения имени необходимо вначале выделить строку сигнала, а затем дважды «кликнуть» «мышью» по графе **Имя** в выделенной строке – строка станет доступной для редактирования.

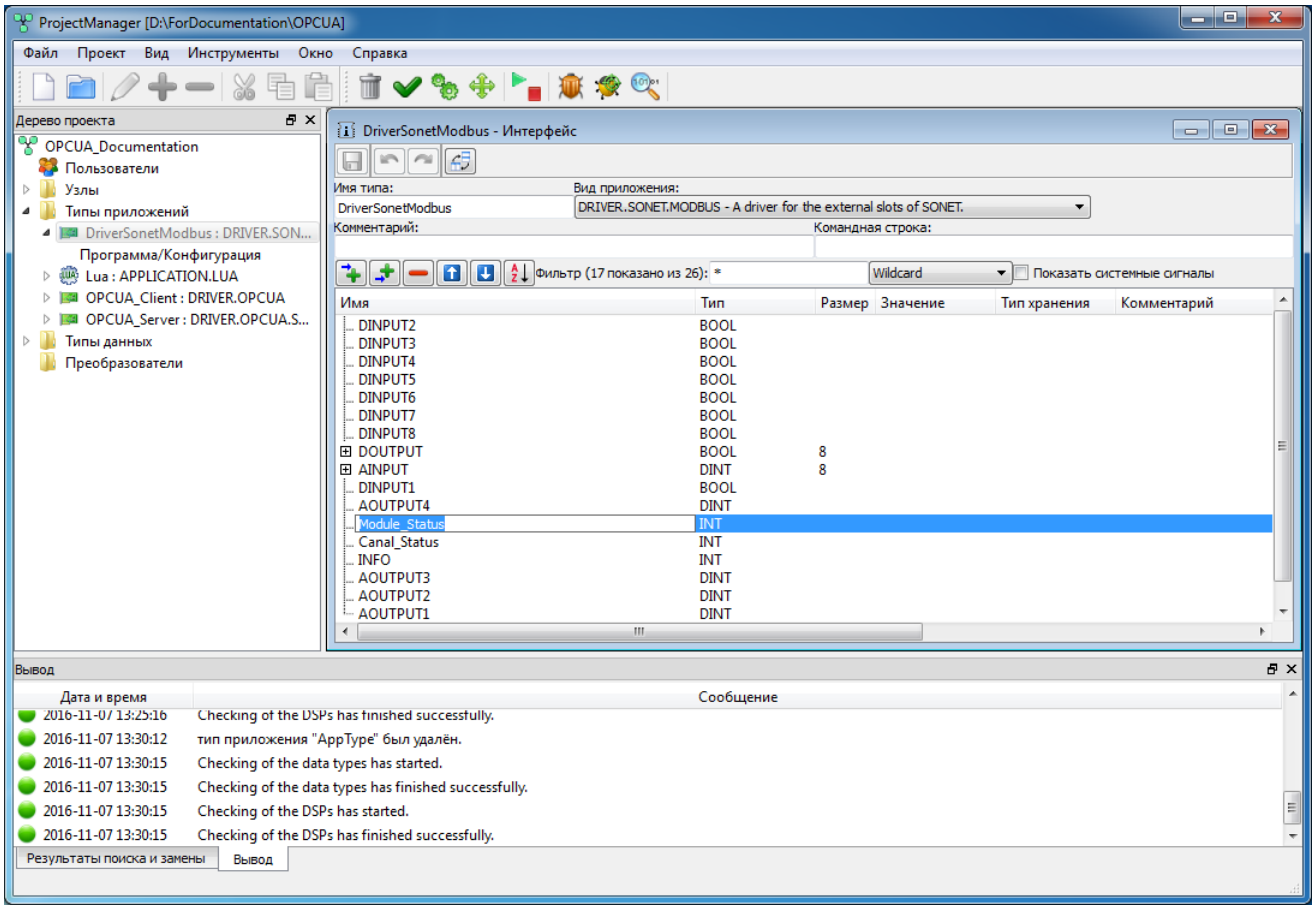


Рисунок 3.293 - Редактирование имени локального сигнала в интерфейсе приложения

Можно также поменять тип сигнала: для этого необходимо дважды «кликнуть» «мышью» по графе **Тип**, а затем кликнуть по кнопке – появится выпадающее меню, в котором можно изменить тип локального сигнала (см. рис. 3.294).

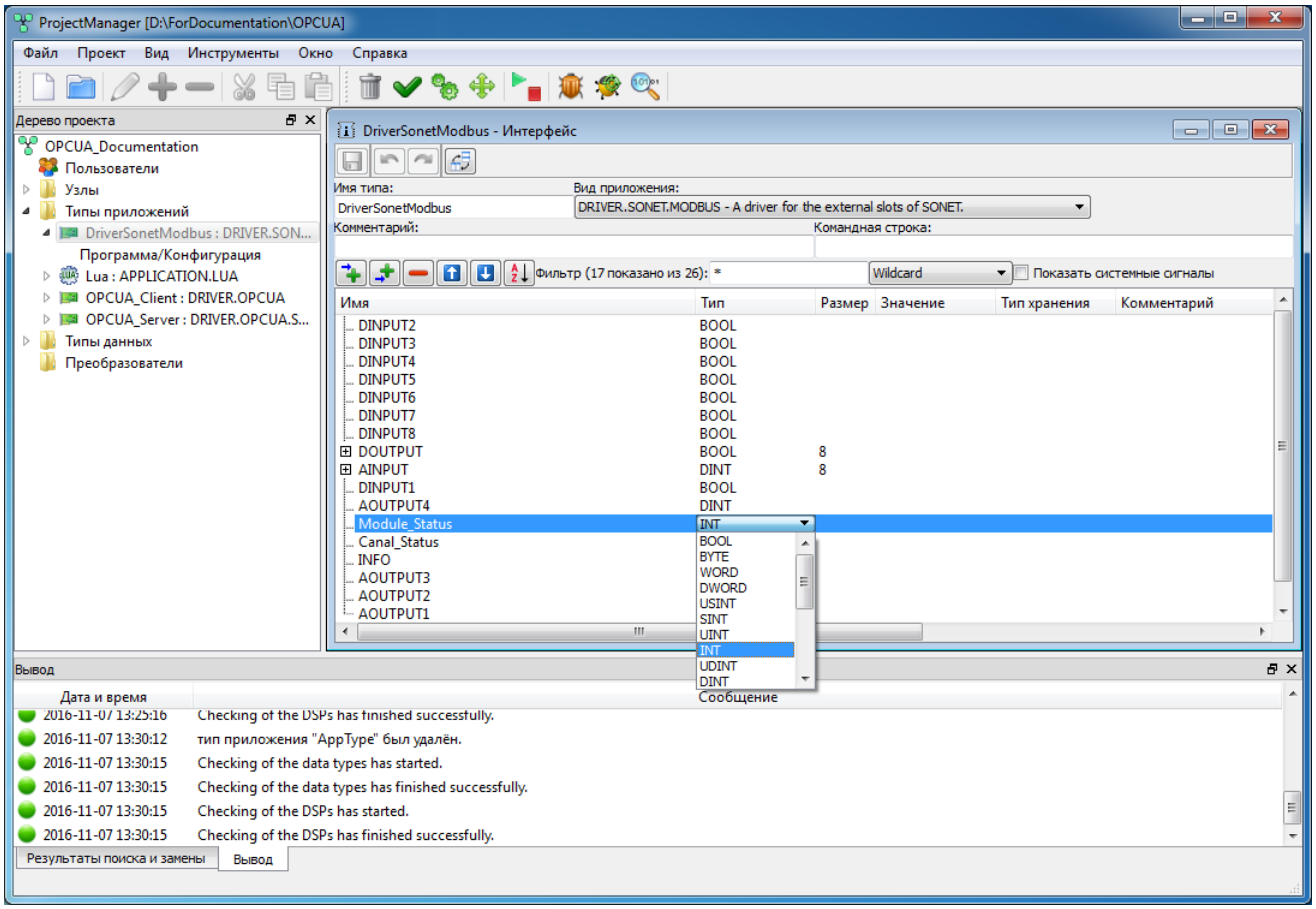


Рисунок 3.294 - Изменение типа локального сигнала в интерфейсе приложения

Для добавления глобального сигнала открывается специальный диалог (см. рис. 3.295), добавление сигнала возможно, если он уже был создан при редактировании непосредственно самого проекта.

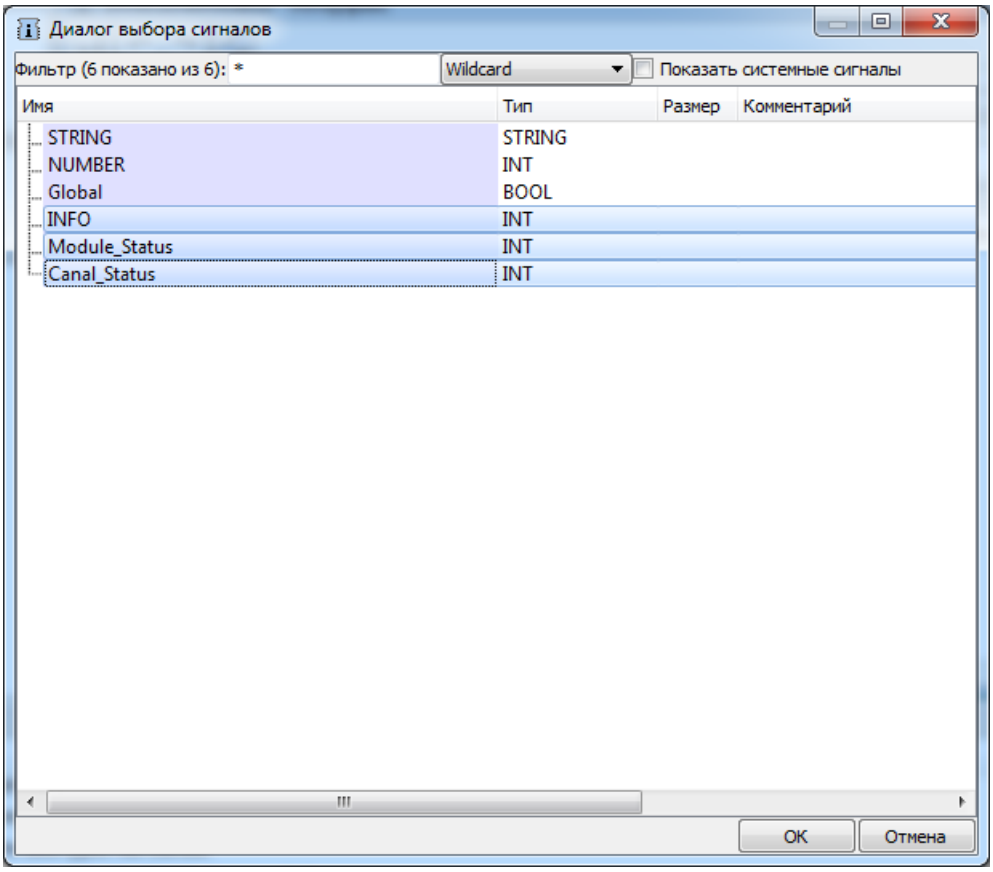


Рисунок 3.295 - Диалог добавления глобального сигнала в интерфейс приложения

После выбора нужного глобального сигнала необходимо нажать кнопку OK , либо дважды «кликнуть» «мышью» на строчку с нужным глобальным сигналом. Для выхода из диалога без добавления глобального сигнала в интерфейс следует нажать кнопку Отмена.

Окончательные изменения в интерфейсе необходимо сохранить с помощью кнопки («Сохранить»). По завершении редактирования интерфейса типа приложения диалог нужно закрыть крестиком в верхнем правом углу окна редактирования. Если внесенные изменения не были сохранены, то на экране появится диалог (см. рис. 3.296), в котором нужно выбрать либо сохранение изменений и выход из редактора (кнопка Сохранить) , либо отказ от внесенных изменений и выход из редактора (кнопка Отклонить) , либо отказаться от выхода из процедуры редактирования (кнопка Отмена).

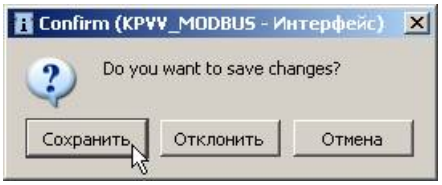


Рисунок 3.296 - Диалог подтверждения выхода из редактора интерфейса приложения

3.3.8.10.3. Привязка сигналов приложения-драйвера SonetModbus к каналам ввода/вывода

Для дальнейшей настройки приложения-драйвера SonetModbus необходимо запустить специализированный редактор приложения-драйвера SonetModbus. Процесс запуска идентичен для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4).

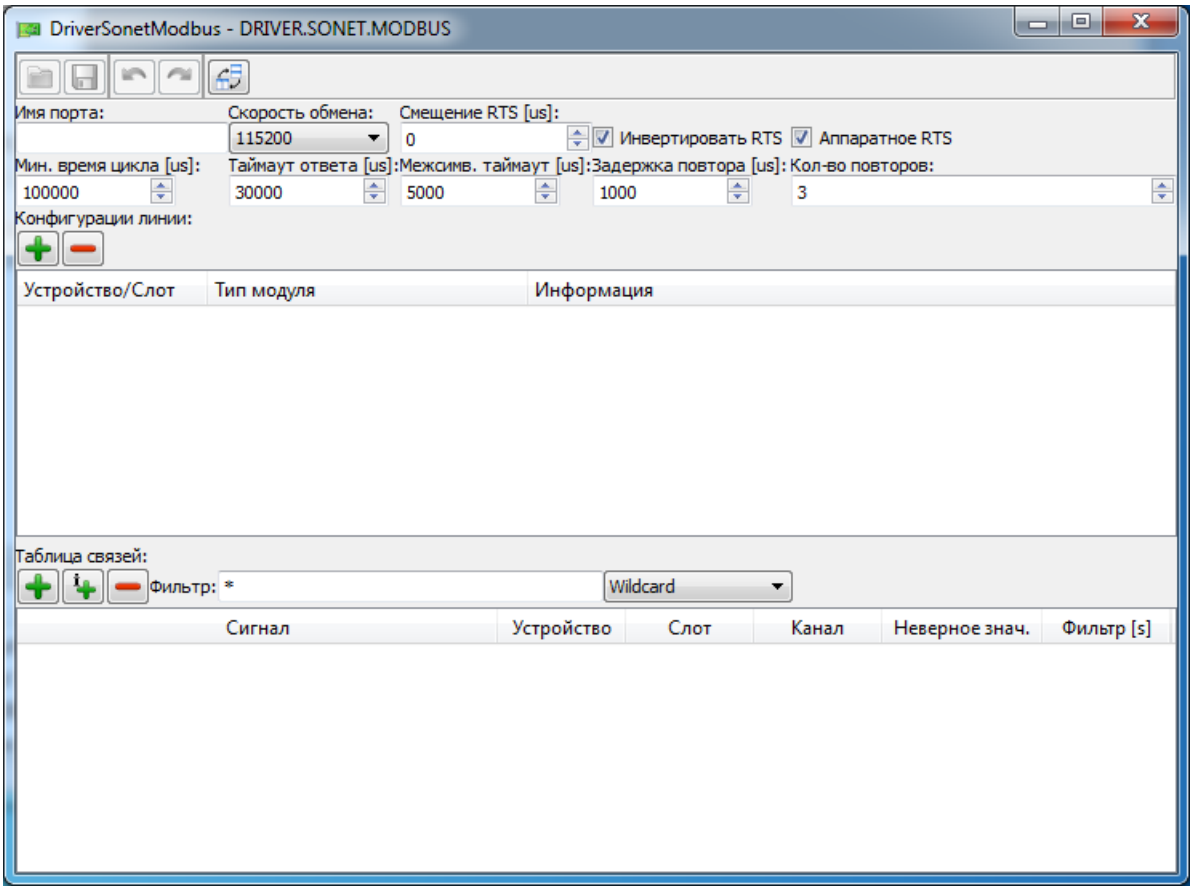


Рисунок 3.297 - Окно специализированного редактора драйверов

Окно редактора драйверов состоит из панели инструментов, полей ввода для настраиваемых параметров, и двух таблиц: **Конфигурации линии** и **Таблица связей**.

Панель инструментов расположена в верхней части окна – на этой панели расположены следующие значки (кнопки):

 («Открыть файл») – команда для открытия уже сформированного файла, содержащего конфигурацию приложения-драйвера;

 («Сохранить») – команда для сохранения конфигурации приложения-драйвера;

 («Назад») – отменить предыдущее действие по редактированию;

 («Вперед») – повторить ранее отменённое действие;

 («Переставить») – поменять расположение вкладок в окне с вертикального на горизонтальное (или обратно).

Далее располагаются следующие поля ввода:

- **Имя порта** - имя порта, по которому контроллер Sonet-мастер будет опрашивать другие контроллеры Sonet.

Для контроллеров СН-МП-ВК значения данных портов следующие:

* в старой прошивке (уже не должна использоваться) MODBUS1 называется /dev/ser4, а MODBUS2 называется /dev/ser3;

* в новой прошивке (должна быть на данных контроллерах) MODBUS1 называется /dev/ser2, а MODBUS2 называется /dev/ser3.

Для контроллеров СН-МП-PC104 значения данных портов следующие:

* MODBUS1 называется /dev/ser1, а MODBUS2 называется /dev/ser2.

Чтобы проверить имя порта можно подключиться к контроллеру и выполнить команду:

cp имя_файла_для_передачи имя_порта.

К примеру отправим файл MODBUS из дистрибутива в порт /dev/ser1:

cp /home/Sonata/Runtime/MODBUS /dev/ser1

Если имя порта указано правильно, то на контроллере у порта должна загореться лампочка TX.

- **Скорость обмена** - скорость обмена данными, устанавливается в соответствии с выставленной на микропроцессорном модуле Sonet-slave (дополнительный модуль);

- **Смещение RTS [us]** - задержка отключения сигнала RTS (Request To Send - запрос на отправку) после передачи данных;

- **Инвертировать RTS** - использовать инвертированные значения сигнала RTS (для контроллеров Сонет СН-МП-ВК нужно использовать инвертированные значения, т.е. включить данную настройку);

- **Аппаратное RTS** - использовать аппаратный RTS;

- **Мин. время цикла [us]** - время цикла опроса, если опрос проходит быстрее данного времени, то следующий опрос начнётся не сразу, а когда выйдет данное время до конца, если опрос пройдёт медленнее, то следующий опрос начнётся сразу;

- **Таймаут ответа [us]** - время ожидания прихода первого байта ответа;

- **Межсимв. таймаут [us]** - предельное время между соседними символами в одном пакете;

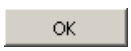
- **Задержка повтора [us]** - задержка перед началом повторного запроса;

- **Кол-во повторов [us]** - кол-во попыток отправки повторных запросов.

Далее располагаются следующие значки (кнопки) управления таблицей **Конфигурации линии**:

 («Добавить устройство») – команда для добавления нового контроллера в список (в список включаются контроллеры, соединенные по сети Modbus);

 («Удалить устройство») – команда для удаления выделенного контроллера.

Для добавления нового контроллера в список необходимо нажать кнопку  («Добавить устройство») – на экране появится диалог (см. рис. 3.298), в котором следует указать адрес контроллера в сети Modbus. После задания нужного адреса необходимо нажать в диалоге кнопку  – в таблице **Конфигурации линии** отобразится контроллер с адресом 22 (см. рис. 3.299), для которого необходимо задать состав модулей ввода/вывода.

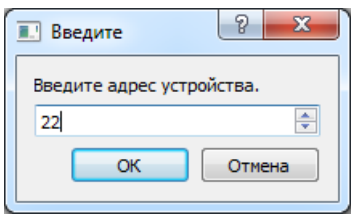


Рисунок 3.298 - Диалог добавления контроллера с указанием адреса в сети modbus

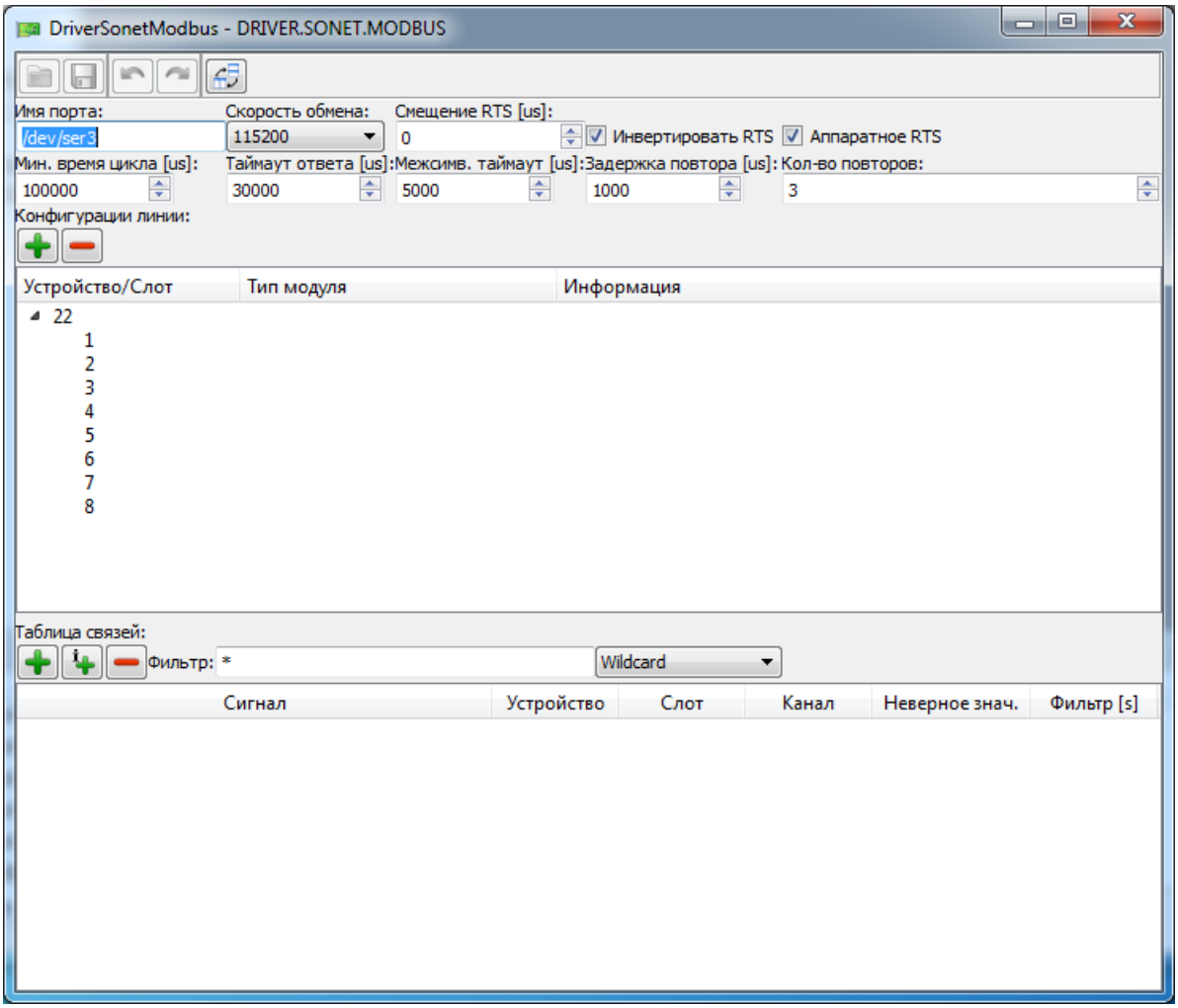


Рисунок 3.299 - Добавление нового устройства в конфигурацию

Таблица **Конфигурации линии** состоит из следующих столбцов:

- **Устройство/Слот** - Modbus адрес устройства и номер слота модуля ввода/вывода;

- **Тип модуля** - известные SCADA-системе "Соната" типы модулей (см. Приложение Е);

- **Информация** - информация о выбранном модуле.

Для конфигурирования контроллера необходимо вначале выбрать его в списке по номеру адреса (столбец **Устройство/Слот**) и раскрыть его структуру, «кликнув» «мышью» по значку ⊞, а затем выделить щелчком «мыши» номер позиции в крейте (с 1 по 8) и двойным щелчком по ячейке в графе **Тип модуля** вызвать выпадающее меню, содержащее список модулей ввода/вывода, в котором наконец можно выбрать требуемый тип модуля (см. рис. 3.300).

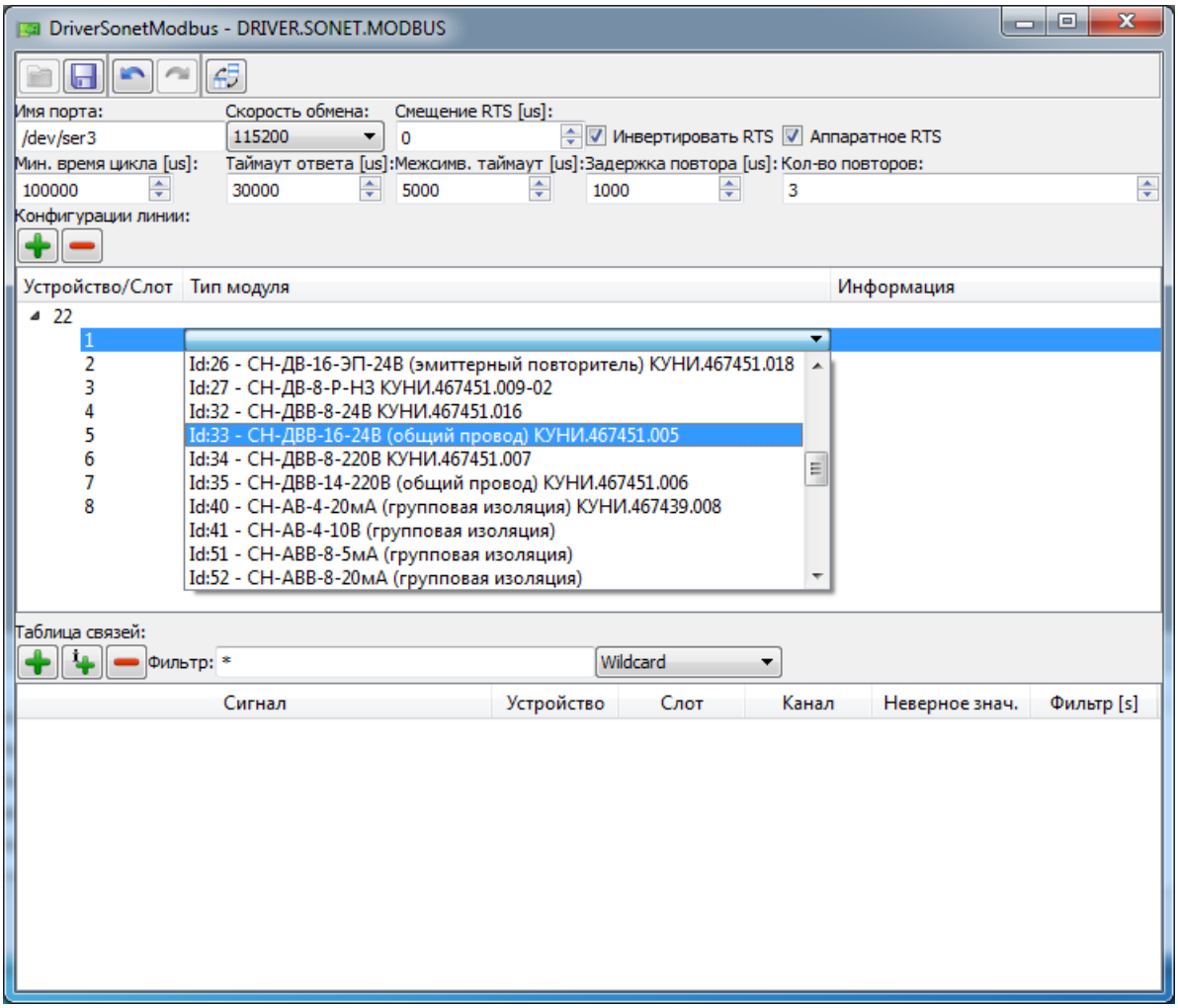


Рисунок 3.300 - Настройки драйвера в редакторе

На рис. 3.302 настроена корзина с микропроцессорным модулем, имеющем Modbus адрес 22 и 4 модуля ввода/вывода. В первой позиции-слоте находится цифровой модуль на 16 входов, во второй позиции-слоте находится цифровой модуль на 16 выходов, в третьей позиции-слоте находится аналоговый модуль на 8 входов и в четвёртой позиции-слоте находится аналоговый модуль на 4 выхода. Каждый модуль оснащён своей маркировкой, в соответствии с которой в столбце **Тип модуля** выбирается нужное значение (см. Приложение Е). После выбора типа модуля в строке **Информация** отобразится дополнительная информация о выбранном модуле.

ВАЖНО! У некоторых модулей в строке **Информация** отобразятся следующие данные

4	Id:64 - CH-АВВ-8-20мА КУНИ.467439.024	8-ch аналоговый входной 0 .. 20 mA, достоверность(3.19982 ..)
5	Id:62 - CH-АВВ-4-20мА КУНИ.467439.022	4-ch аналоговый входной 0 .. 22 mA, достоверность(3.19988 .. 20.8002)
6	Id:63 - CH-АВВ-4-Р500 КУНИ.467439.023	4-ch аналоговый входной 0 .. 500 Ohm, достоверность(..)
7	Id:67 - CH-АВВ-4-ТП-2 КУНИ.467439.022-06	4-ch аналоговый входной 0 .. 22 mA, достоверность(.. 20.8002)

Запись **достоверность** сообщает о том, что каналы данного модуля в указанном диапазоне будут считаться исправными и у получаемых данных будет стоять признак достоверности. Вне данного диапазона каналы будут считаться неисправными и данным, получаемым по ним, будет выставляться признак недостоверности (**НОПР**). Данные диапазоны считаются диапазонами по умолчанию. Но в **Таблице связей** есть два столбца для настройки данных диапазонов вручную: **Нижняя граница** и **Верхняя граница** (см. рис. 3.301).

Пояснения к диапазонам достоверности (рабочие диапазоны указаны перед диапазоном достоверности в строке **Информация**):

- **..** - диапазон достоверности включает весь рабочий диапазон;
- **.. 20.8002** - диапазон достоверности начинается от левой границы рабочего диапазона и до 20.8002;

- **3.19988 ..** - диапазон достоверности начинается от 3.19988 и до правой границы рабочего диапазона;

- **3.19988 .. 20.8002** - диапазон достоверности начинается от 3.19988 и до 20.8002.

Границы диапазонов достоверности указываются в единицах измерения канала.

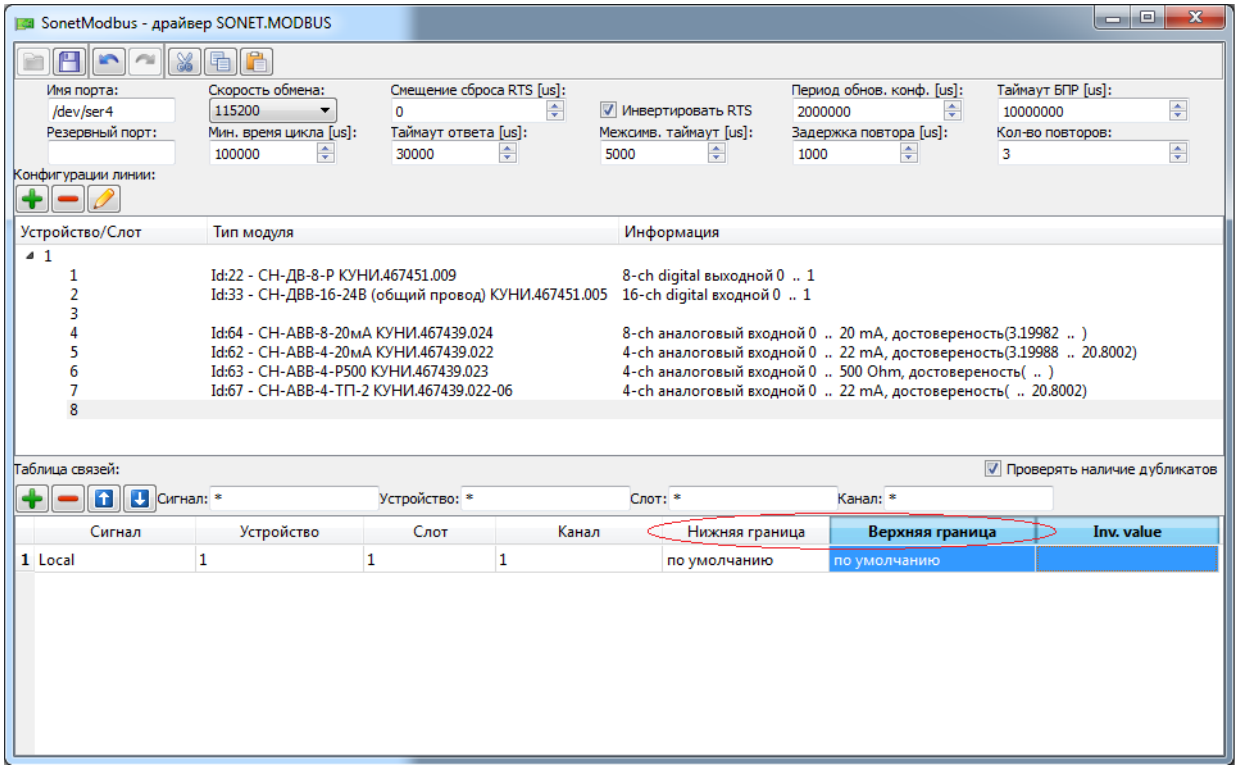


Рисунок 3.301 - Окно Настроек драйвера Sonet.Modbus - поля настройки диапазонов

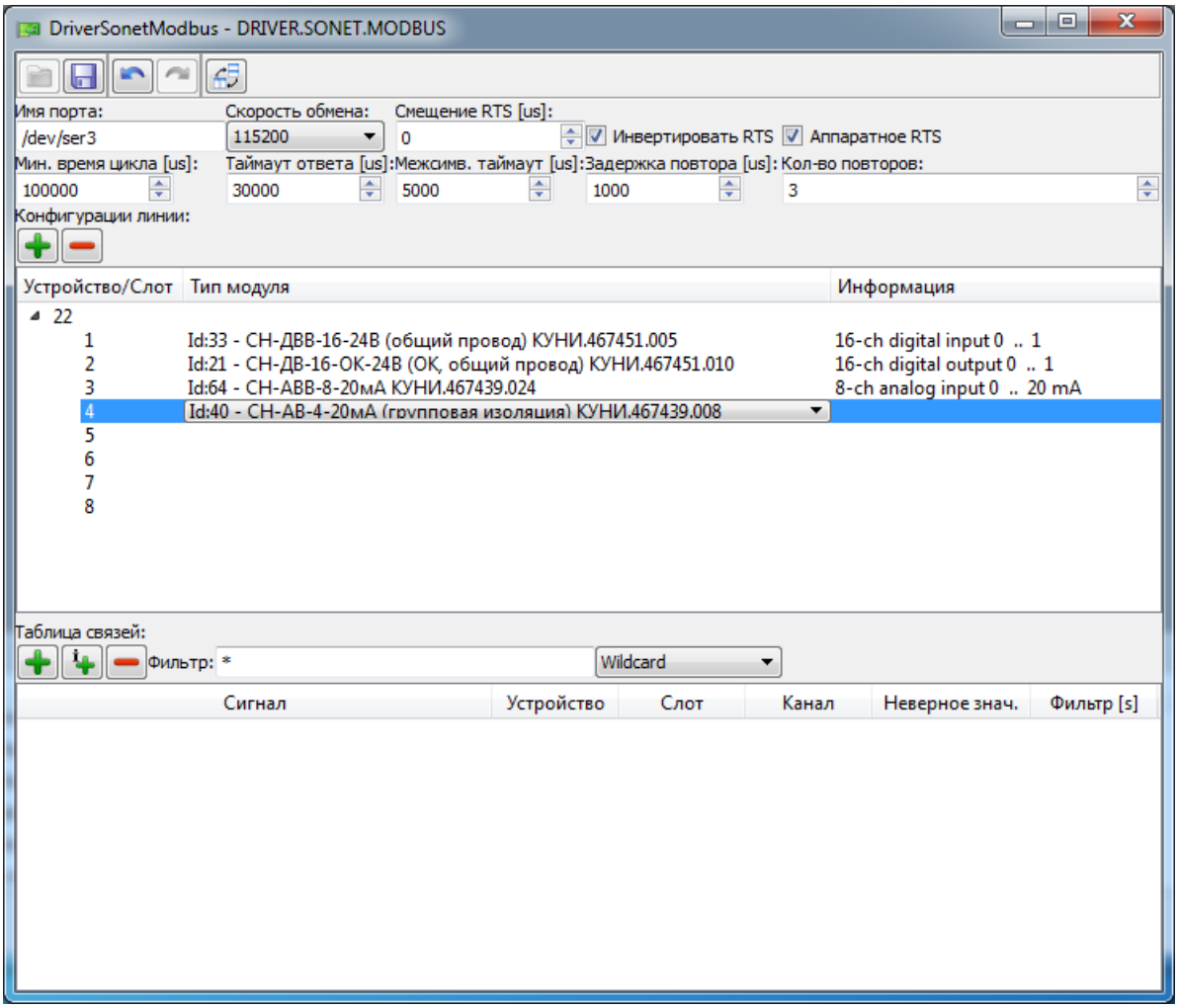


Рисунок 3.302 - Несколько модулей в конфигурации

Для привязки сигналов SCADA-системы "Соната" к каналам ввода/вывода служит **Таблица связей**.

В верхней части расположены следующие значки (кнопки) и поле ввода:

 («Добавить элемент») – добавить новую строку в **Таблицу связей**;

 («Импорт из интерфейса») – команда для импорта сразу всех сигналов из интерфейса приложения в схему коммутации;

 («Удалить элемент») – удалить выбранный элемент из таблицы связей;

- поле настройки фильтров отображения сигналов.

Таблица связей состоит из следующих столбцов:

- **Сигнал** - сигнал из интерфейса приложения-драйвера;
- **Устройство** - Modbus адрес устройства;
- **Слот** - разъем для модуля ввода/вывода;
- **Канал** - номер канала на модуле ввода/вывода;
- **Неверное знач.** - значение, которое будет принимать сигнал, когда канал неисправен;
- **Фильтр [s]** - постоянная времени фильтра, выраженная в секундах.

С помощью кнопок  ("Добавить элемент") и  ("Импортировать элементы из интерфейса") в столбец **Сигнал** добавляем нужные сигналы из интерфейса приложения-драйвера. В столбце **Устройство** выбирается нужный Modbus адрес устройства. В столбце **Слот** нужно выбрать

соответствующий номер позиции-слота, в которой, в соответствии с ранее настроенной таблицей **Конфигурации линии**, находится нужный модуль ввода/вывода. Если выбрать в столбце **Слот** значение **info**, то данный сигнал будет сообщать о состоянии микропроцессорного модуля в данной корзине. В столбце **Канал** выбирается номер канала ввода/вывода и также есть дополнительные значения, которые рассмотрим на примере (см. рис. 3.303).

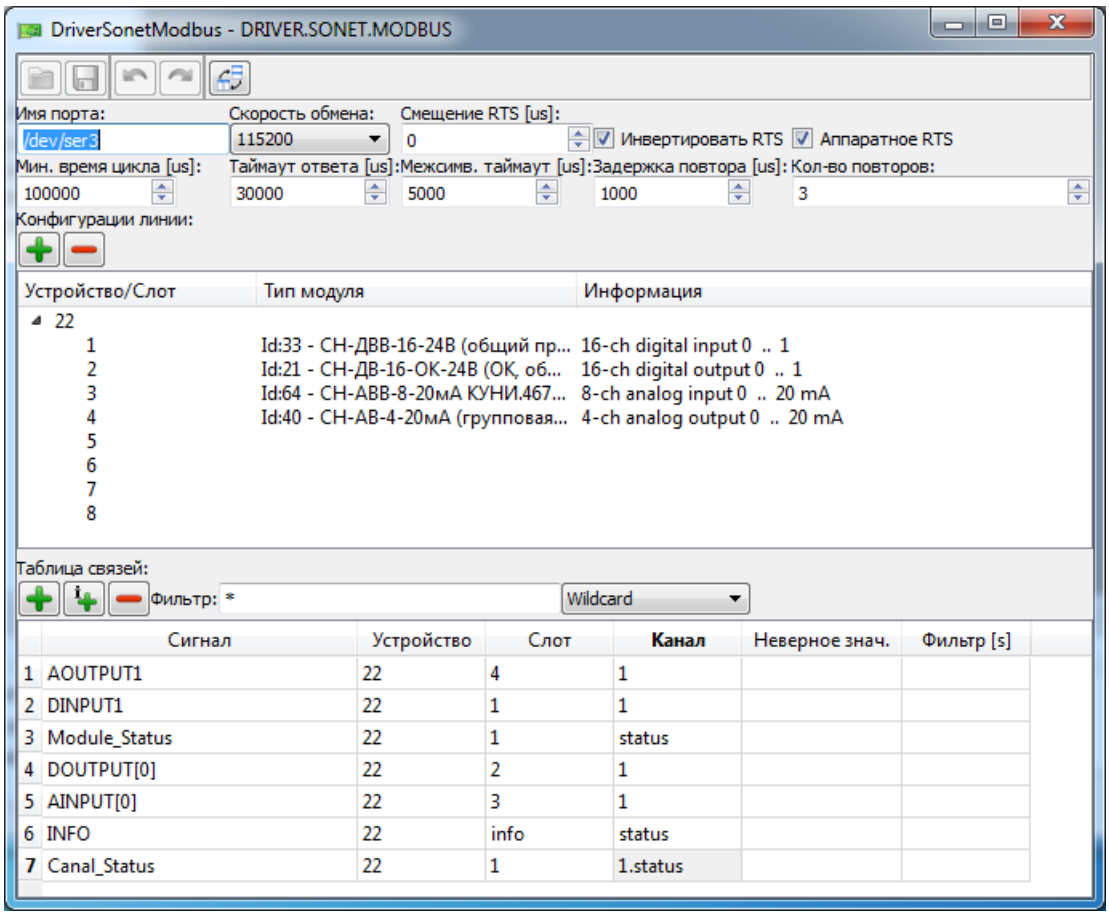


Рисунок 3.303 - Настройки Таблицы связей конфигурации приложения-драйвера SonetModbus

Для сигнала **AINPUT[0]** выставлены значения **Слот** = 3 - это аналоговый модуль на 8 входов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **AOUTPUT1** выставлены значения **Слот** = 4 - это аналоговый модуль на 4 выхода и **Канал** = 1 - это первый канал данного модуля.

Аналогично настроены сигналы **DINPUT1**, **DOUTPUT[0]**.

Для сигнала **Canal_Status** выставлены настройки **Слот** = 1 - это цифровой модуль на 16 входов и **Канал** = **1.status** - это статус первого канала данного модуля. Статусы каналов и модулей описаны в Приложение В.

Для сигнала **INFO** выставлены значения **Слот** = **info** - это получение информации о микропроцессорном модуле и **Канал** = **status** - это статус модуля, а как видно по значению **Слот** = **info**, то это статус микропроцессорного модуля. Также значение **Канал** для значения **Слот** = **info** может принимать следующие значения:

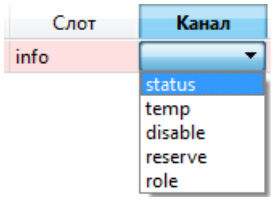


Рисунок 3.304 - Значения поля Канал для Слот = info

- **temp** - это значение температуры микропроцессорного модуля;

- **disable** - это параметр, который позволяет отключить опрос данного микропроцессорного модуля драйвером, т.е. по умолчанию он равен FALSE и драйвер опрашивает данный микропроцессорный модуль, если присвоить TRUE, то драйвер перестанет опрашивать данный микропроцессорный модуль;

- **reserve** - этот сигнал служит для управления состоянием выбранного контроллера:

- если сигнал = 0, то драйвер автоматически управляет переключением между основным и резервным КУВВ;

- если сигнал = 1, то основной контроллер принудительно становится активным, а резервный пассивным;

- если сигнал = 2, то основной контроллер принудительно становится пассивным, а резервный активным;

- **role** - это значение связано с БПР, т.е. БПР сообщает микропроцессорному модулю о его состоянии основной/резервный. Если данное значение равно 2, то значит модуль в резерве, если равно 1, то значит модуль основной(активный), если значение равно 0, то значит данный микропроцессорный модуль не подключён к БПР.

Для сигнала **Module_Status** выставлены значения **Слот** = 1 - это цифровой модуль на 16 входов и **Канал** = **status** - это статус модуля, соответственно получаем информацию о работоспособности модуля в первом слоте.

В столбце **Неверное знач.** можно выставить значение для сигнала, которое он всегда будет принимать, в случае любой неисправности канала.

Для дискретных сигналов в столбце **Фильтр [s]** настраивается фильтрация от помех, т.е. значение времени в секундах, в пределах которого изменения сигнала не регистрируются (считаются помехой). Для аналоговых сигналов это константа времени для апериодического фильтра первого порядка.

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер Sonet.Modbus настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на узел проекта, который является микропроцессорным модулем **Sonet** с соответствующей настройкам корзиной модулей ввода/вывода (см. раздел 3.2.2.5.1).

3.3.8.11. Описание работы с приложением-драйвером Telekont2.Modbus

Приложение-драйвер **Telecont2.Modbus** применяется для работы с цифровыми и аналоговыми модулями ввода/вывода под управлением микропроцессорного модуля **ТелеКонт**. В SCADA-системе "СОНАТА" в списке приложений-драйверов присутствуют два приложения-драйвера **Telekont**: **Telecont.Modbus** и **Telekont2.Modbus**. В настоящее время применяется только приложение-драйвер **Telekont2.Modbus**, но описание работы и рабочих окон аналогичны.

3.3.8.11.1. Создание приложения-драйвера Telekont2.Modbus

Создание приложения-драйвера **Telekont2.Modbus** осуществляется аналогично другим приложениям-драйверам SCADA-системы "СОНАТА" (см. раздел 3.2.2.4).

На рис. 3.305 изображено стандартное окно интерфейса приложения SCADA-системы "Соната". Чтобы данное приложение стало приложением-драйвером **Telekont2.Modbus**,

необходимо в поле **Вид приложения** выбрать **DRIVER.TELEKONT2.MODBUS** и в поле **Имя типа** написать удобное для дальнейшего использования в проекте имя данного приложения (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

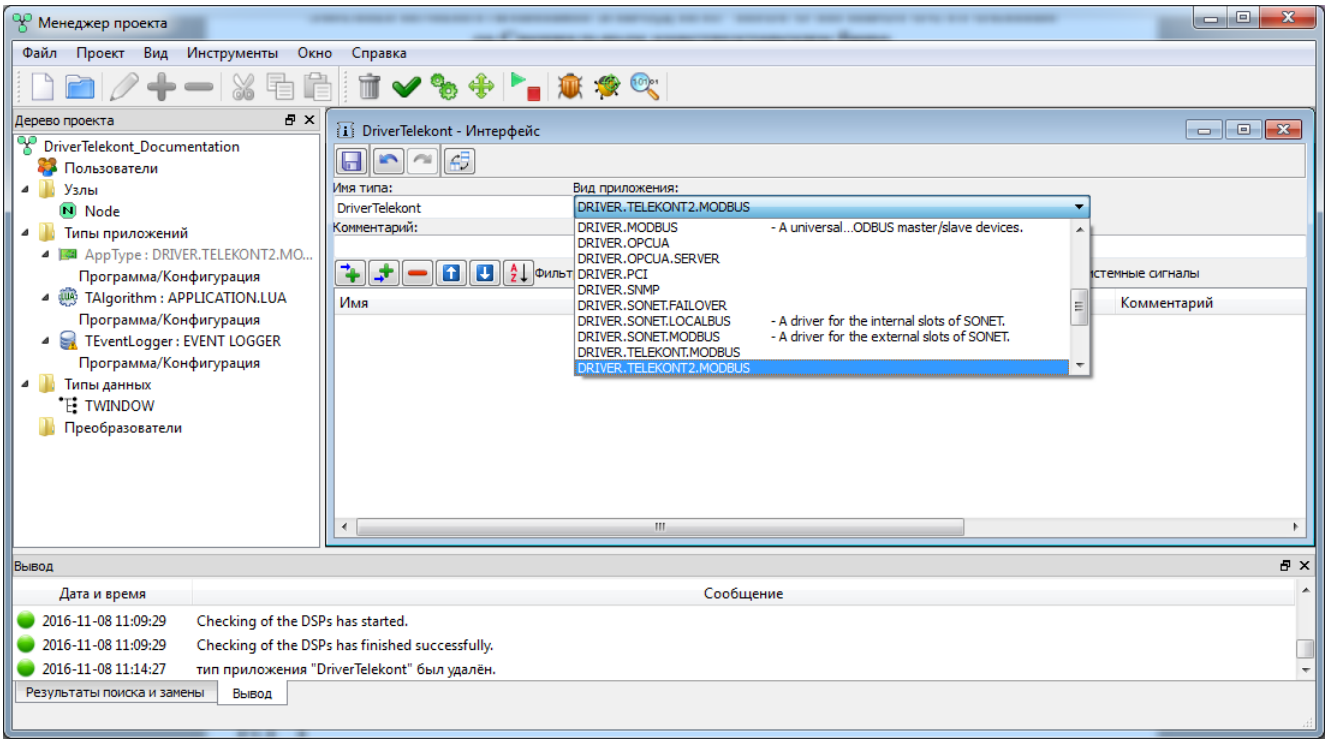


Рисунок 3.305 - Изменение имени типа и вида приложения-драйвера

3.3.8.11.2. Создание интерфейса приложения-драйвера Telekont2.Modbus

После выбора вида приложения-драйвера необходимо добавить в интерфейс приложения сигналы, значения которых будут связаны с каналами модулей ввода/вывода. Процесс добавления сигналов в интерфейс приложения подробно описан в раздел 3.2.2.4.3. На рис. 3.306 в интерфейс приложения-драйвера добавлено 7 сигналов: **DINPUT1** - сигнал дискретного ввода, **DOUTPUT1** - сигнал дискретного вывода, **AINPUT1** - сигнал аналогового ввода, **AOUTPUT1** - сигнал аналогового вывода, сигнал **INFO** - состояние микропроцессорного модуля, **Module_Status** - состояние одного из модулей (см. далее) и **Canal_Status** - состояние одного из каналов (см. далее). Значения кодов и статусов для сигналов **INFO**, **Module_Status** и **Canal_Status** смотри в Приложение В.

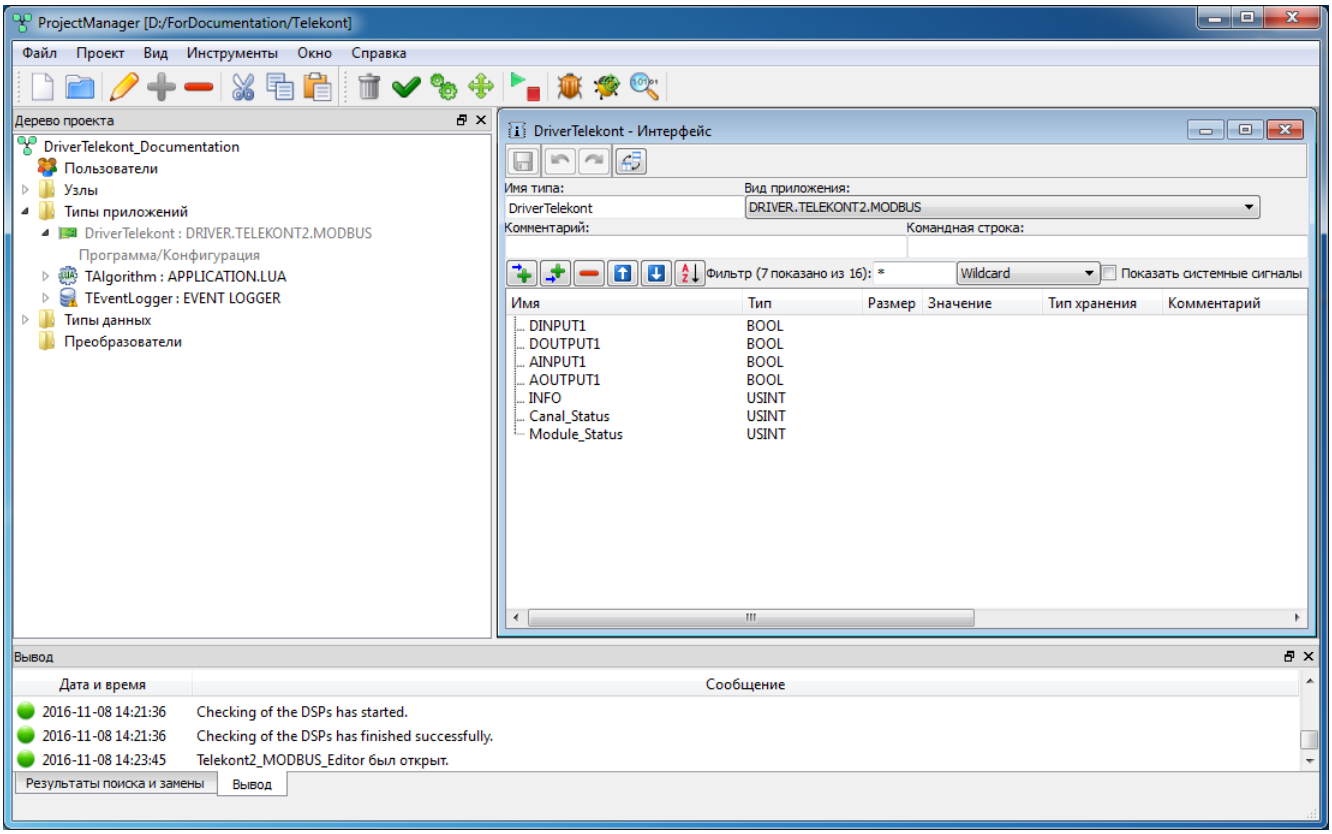


Рисунок 3.306 - Сигналы в интерфейсе приложения-драйвера

После добавления сигналов в интерфейс приложения-драйвера необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Интерфейс приложения настроен и теперь необходимо настроить связь сигналов из интерфейса приложения с каналами модулей ввода/вывода.

3.3.8.11.3. Привязка сигналов приложения-драйвера Telekont2.Modbus к каналам ввода/вывода

Для привязки сигналов из интерфейса приложения-драйвера к каналам модулей ввода/вывода необходимо вызвать специализированный редактор приложения-драйвера **Telekont2.Modbus**. Процесс вызова специализированного редактора приложения одинаков для всех приложений SCADA-системы "Соната" (см. раздел 3.2.2.4.4). На рис. 3.307 изображено окно специализированного редактора (далее просто редактор) приложения-драйвера **Telekont2.Modbus**.

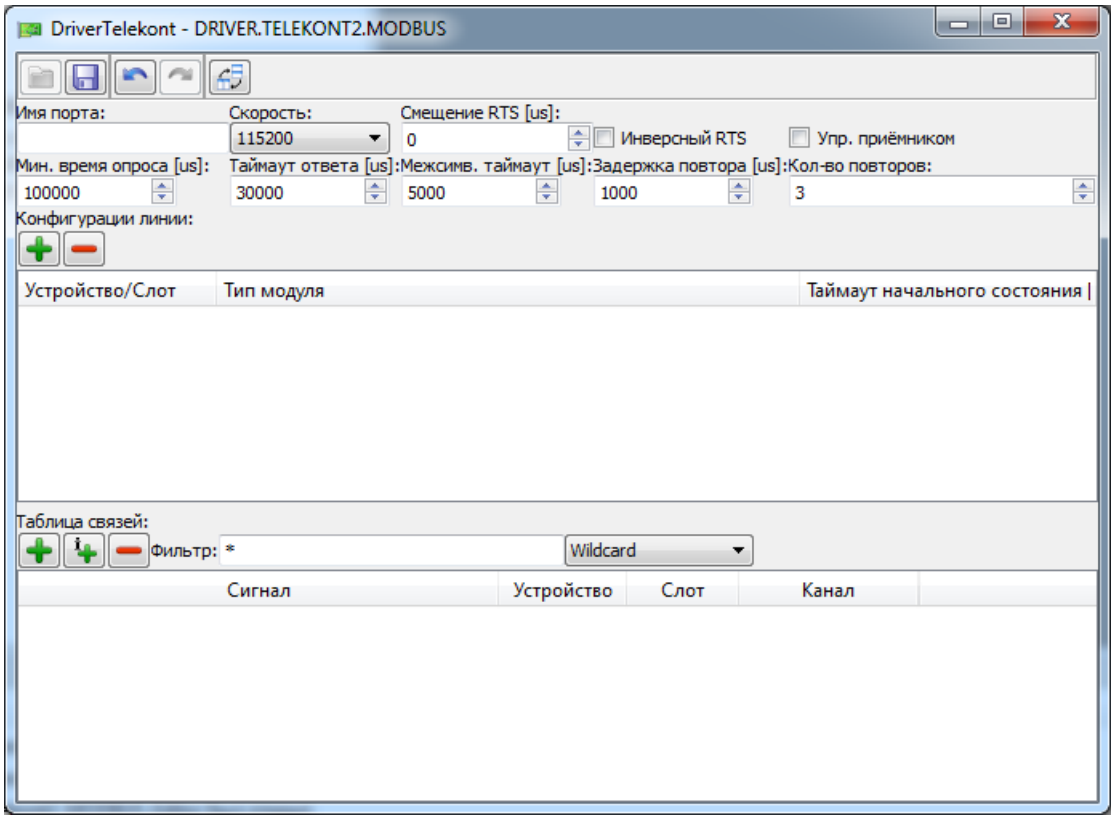


Рисунок 3.307 - Окно редактора приложения-драйвера Telekont2.Modbus

В верхней части окна расположена кнопка  ("Сохранить") и стандартные для оконных приложений кнопки управления окном  , имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

Ниже располагаются следующие кнопки:

 ("Открыть") - открыть уже имеющийся файл с конфигурацией;

 («Сохранить») – сохранить внесённые в редакторе изменения;

 («Отменить») – отменить предыдущее действие по редактированию;

 («Повторить») – повторить ранее отменённое действие;

 («Изменить ориентацию») – изменить ориентацию окон.

Далее располагаются следующие поля ввода:

- **Имя порта** - имя порта, по которому узел с приложением-драйвером **Telekont2.Modbus** будет взаимодействовать с микропроцессорным модулем **ТелеКонт** (к примеру Com-порт для ОС Windows или /dev/serN, где N - номер порта для ОС QNX);

- **Скорость обмена** - скорость обмена данными, устанавливается в соответствии с выставленной на микропроцессорном модуле **ТелеКонт**;

- **Смещение RTS [us]** - задержка отключения сигнала RTS после передачи данных;

- **Инверсный RTS** - использовать инвертированные значения сигнала RTS (для контроллеров Сонет);

- **Управление приёмником** - отключение приемника на момент передачи;

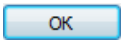
- **Мин. время опроса [us]** - время цикла опроса, если опрос проходит быстрее данного времени, то следующий опрос начнётся не сразу, а когда выйдет данное время до конца, если опрос пройдёт медленнее, то следующий опрос начнётся сразу;

- **Таймаут ответа [us]** - время ожидания прихода первого байта ответа;
- **Межсимв. таймаут [us]** - предельное время между соседними символами в одном пакете;
- **Задержка повтора [us]** - задержка перед началом повторного запроса;
- **Кол-во повторов [us]** - кол-во попыток отправки повторных запросов.

Далее располагаются следующие значки (кнопки) управления таблицей **Конфигурации линии**:

 («Добавить контроллер») – команда для добавления нового контроллера в список (в список включаются контроллеры, соединенные по сети Modbus);

 («Удалить контроллер») – команда для удаления выделенного контроллера.

Для добавления нового контроллера в список необходимо нажать кнопку  («Добавить контроллер») – на экране появится диалог (см. рис. 3.308), в котором следует указать адрес контроллера в сети Modbus (можно узнать в настройках микропроцессорного модуля **ТелеКонт**). После задания нужного адреса необходимо нажать в диалоге кнопку  – в таблице **Конфигурации линии** отобразится контроллер с адресом 33 (см. рис. 3.309), для которого необходимо задать состав модулей ввода/вывода.

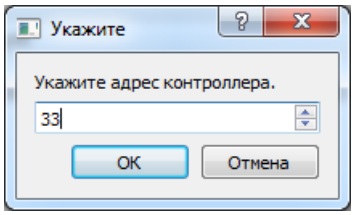


Рисунок 3.308 - Диалог добавления нового контроллера

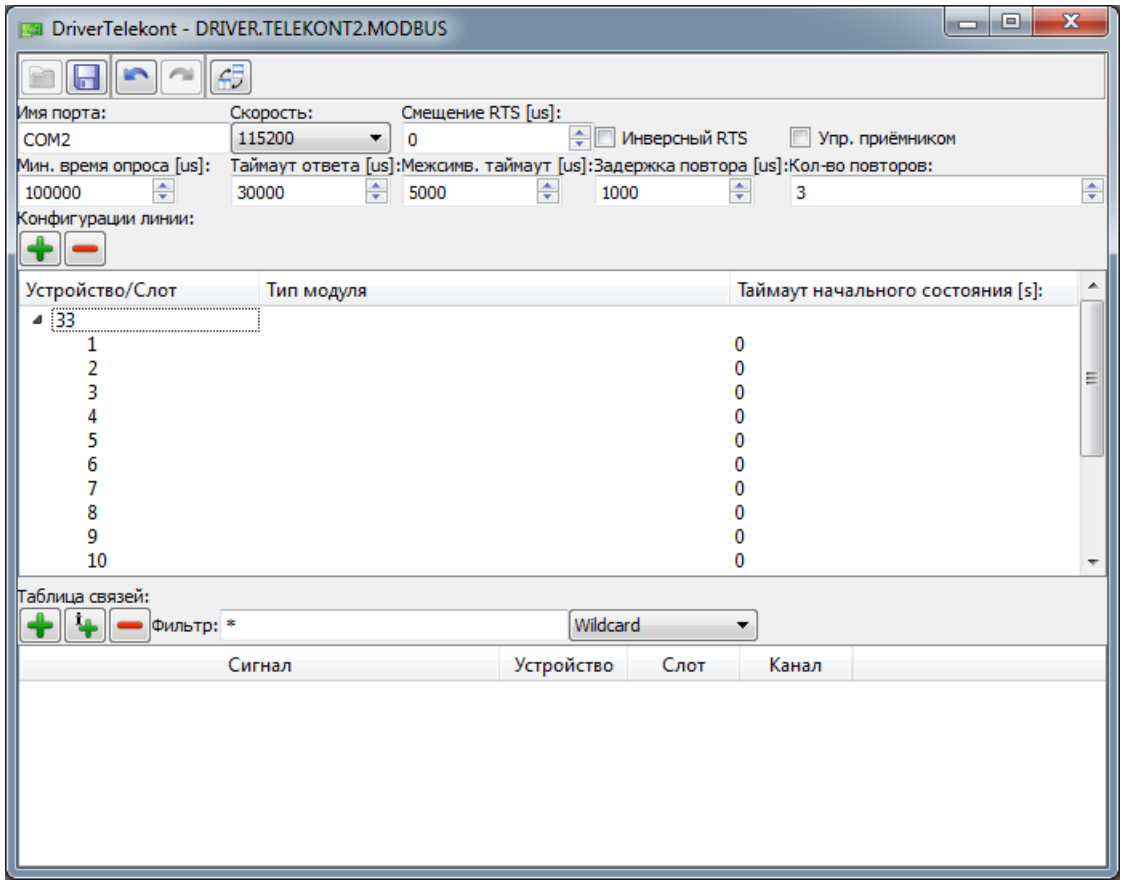


Рисунок 3.309 - Добавленный контроллер с Modbus адресом 33 в редакторе

Таблица **Конфигурации линии** состоит из следующих столбцов:

- **Устройство/Слот** - Modbus адрес устройства и номер слота модуля ввода/вывода;

- **Тип модуля** - известные SCADA-системе "Соната" типы модулей ввода/вывода **ТелеКонт**;

- **Таймаут начального состояния [s]** - время, через которое модуль ввода/вывода переводит свои каналы в исходное состояние, в случае отсутствия связи с контроллером **ТелеКонт**.

Для конфигурирования контроллера необходимо вначале выбрать его в списке по номеру Modbus адреса (столбец **Устройство/Слот**) и раскрыть его структуру, «кликнув» «мышью» по значку , а затем выделить щелчком «мыши» номер позиции в крейте (с 1 по 16) и двойным щелчком по ячейке в графе **Тип модуля** вызвать выпадающее меню, содержащее список модулей ввода/вывода, в котором наконец можно выбрать требуемый тип модуля (см. рис. 3.310).

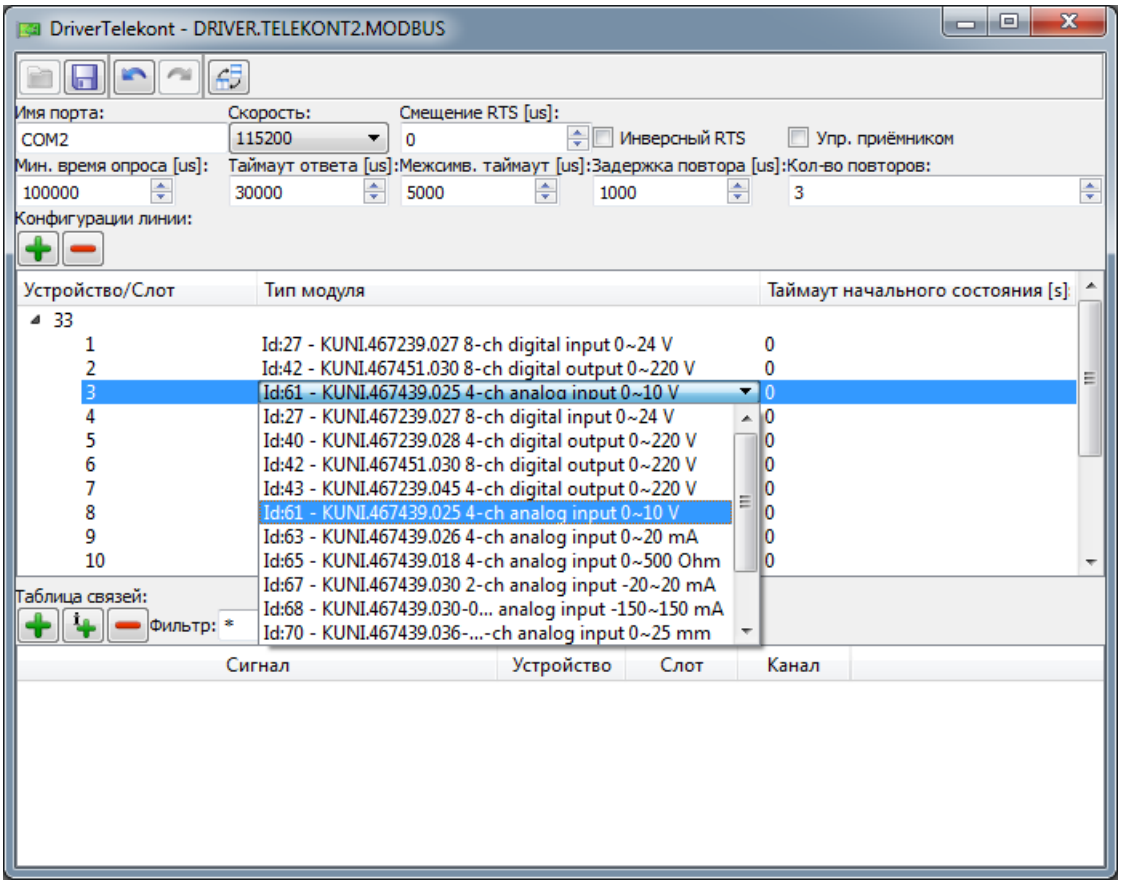


Рисунок 3.310 - Выбор модулей ввода/вывода в редакторе приложения-драйвера

На рис. 3.311 настроена корзина с микропроцессорным модулем **ТелеКонт**, имеющем Modbus адрес 33 и 3 модуля ввода/вывода. В первой позиции-слоте находится цифровой модуль на 8 входов, во второй позиции-слоте находится цифровой модуль на 8 выходов, в третьей позиции-слоте находится аналоговый модуль на 4 входа. Каждый модуль оснащён своей маркировкой, в соответствии с которой в столбце **Тип модуля** выбирается нужное значение.

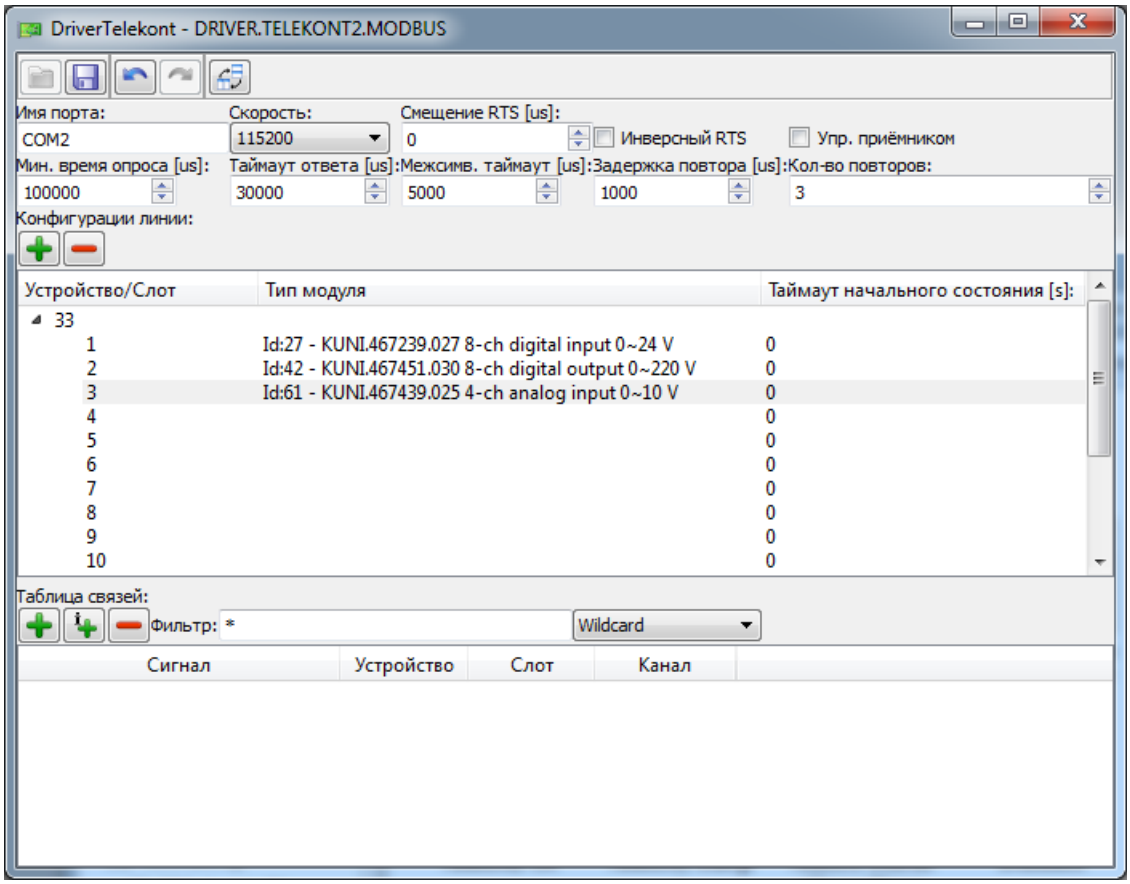


Рисунок 3.311 - Выбранные в редакторе приложения-драйвера модули ввода/вывода

Для привязки сигналов SCADA-системы "Соната" к каналам ввода/вывода служит **Таблица связей**.

В верхней части расположены следующие значки (кнопки) и поле ввода:

 («Добавить элемент») – добавить новую строку в **Таблицу связей**;

 («Импорт из интерфейса») – команда для импорта сразу всех сигналов из интерфейса приложения в схему коммутации;

 («Удалить элемент») – удалить выбранный элемент из таблицы связей;

- поле настройки фильтров отображения сигналов.

Таблица связей состоит из следующих столбцов:

- **Сигнал** - сигнал из интерфейса приложения-драйвера;
- **Устройство** - Modbus адрес устройства;
- **Слот** - разъем для модуля ввода/вывода;
- **Канал** - номер канала на модуле ввода/вывода.

С помощью кнопок  ("Добавить элемент") и  ("Импортировать элементы из интерфейса") в столбец **Сигнал** добавляем нужные сигналы из интерфейса приложения-драйвера. В столбце **Устройство** выбирается нужный Modbus адрес устройства. В столбце **Слот** нужно выбрать соответствующий номер позиции-слота, в которой в соответствии с ранее настроенной таблицей **Конфигурации линии** находится нужный модуль ввода/вывода. Если выбрать в столбце **Слот** значение **info**, то данный сигнал будет сообщать о состоянии микропроцессорного модуля в данной корзине. В столбце **Канал** выбирается номер канала ввода/вывода и также есть дополнительные значения, которые рассмотрим на примере (см. рис. 3.312).

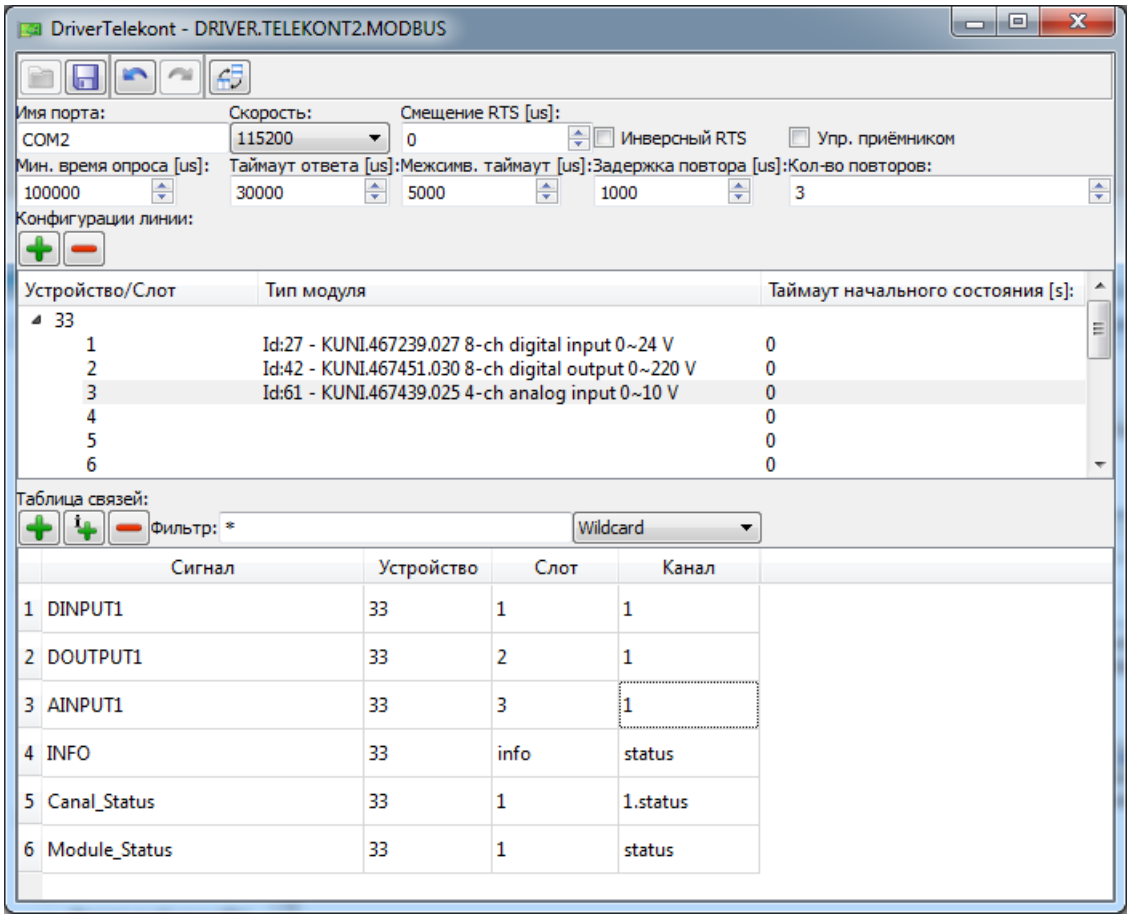


Рисунок 3.312 - Настройка связей сигналов SCADA-системы "Соната" и каналов модулей ввода/вывода

Для сигнала **DINPUT1** выставлены значения **Устройство** = 33 - это микропроцессорный модуль с Modbus адресом 33, **Слот** = 1 - это дискретный модуль на 8 входов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **DOUTPUT1** выставлены значения **Устройство** = 33 - это микропроцессорный модуль с Modbus адресом 33, **Слот** = 2 - это дискретный модуль на 8 выходов и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **AINPUT1** выставлены значения **Устройство** = 33 - это микропроцессорный модуль с Modbus адресом 33, **Слот** = 3 - это аналоговый модуль на 4 входа и **Канал** = 1 - это первый канал данного модуля.

Для сигнала **INFO** выставлены значения **Слот** = **info** - это получение информации о микропроцессорном модуле и **Канал** = **status** - это статус модуля, а как видно по значению **Слот** = **info**, то это статус микропроцессорного модуля. Также значение **Канал** для значения **Слот** = **info** может быть равно **temp** - это значение температуры микропроцессорного модуля.

Для сигнала **Canal_Status** выставлены настройки **Слот** = 1 - это цифровой модуль на 8 входов и **Канал** = **1.status** - это статус первого канала данного модуля. Статусы каналов и модулей описаны в Приложение В.

Для сигнала **Module_Status** выставлены значения **Слот** = 1 - это цифровой модуль на 8 входов и **Канал** = **status** - это статус модуля, соответственно получаем информацию о работоспособности модуля в первом слоте.

После настроек в редакторе необходимо нажать кнопку  ("Сохранить") и закрыть окно редактора. Приложение-драйвер Telekont2.Modbus настроено и готово к работе. Для использования данного приложения в проекте его необходимо добавить на узел проекта, который соединен по

порту, указанному в поле **Имя порта** с микропроцессорным модулем **Telekont** с соответствующей настройкам корзиной модулей ввода/вывода (см. раздел 3.2.2.5.1).

3.3.8.12. Получение конфигурации локальной и удалённой шины для процессорных модулей Sonet

Есть возможность удалённо считать конфигурацию подключённых к контроллеру модулей ввода/вывода.

3.3.8.12.1. Для процессорного модуля СН-МП-ВК

3.3.8.12.1.1. Получение конфигурации модулей ввода/вывода локальной шины

Для получения конфигурации модулей ввода/вывода локальной шины необходимо:

1) Зайти в консоль или подключиться удаленно к процессорному модулю СН-МП-ВК. Удалённое подключение выполняется по протоколу telnet. При правильном подключении появится запрос логина и пароля. Логин по умолчанию root, а пороль пустой. Далее наберите команду mqc - это файловый менеджер в операционной системе QNX.

2) Далее необходимо перейти в папку с дистрибутивом SCADA-системы "Соната". Дистрибутив находится в папке /home/Sonata/Runtime. Это можно сделать командой **cd /home/Sonata/Runtime**.

3) Находясь в папке с дистрибутивом, наберите команду:

./Sonet_LOCALBUS "-forced=1;Drv;127.0.0.1:10000" -continue и нажмите ENTER.

Запустится драйвер Sonet_LOCALBUS, который создаст файл Sonet_LOCALBUS.log и запишет в него информацию о версии драйвера, информацию о процессорном модуле, конфигурацию модулей ввода/вывода на локальной шине и др.

Остановите работу драйвера, нажав сочетание клавиш CTRL+C, и далее ENTER.

Смотрите информацию в файле Sonet_LOCALBUS.log.

3.3.8.12.1.2. Получение конфигурации модулей ввода/вывода удалённой шины

Для получения конфигурации модулей ввода/вывода удалённой шины необходимо:

1) Зайти в консоль или подключиться удаленно к процессорному модулю СН-МП-ВК. Удалённое подключение выполняется по протоколу telnet. При правильном подключении появится запрос логина и пароля. Логин по умолчанию root, а пороль пустой. Далее наберите команду mqc - это файловый менеджер в операционной системе QNX.

2) Далее необходимо перейти в папку с дистрибутивом SCADA-системы "Соната". Дистрибутив находится в папке /home/Sonata/Runtime. Это можно сделать командой **cd /home/Sonata/Runtime**.

3) Находясь в папке с дистрибутивом, наберите команду:

а) **./Sonet_MODBUS "-forced=1;Drv;127.0.0.1:10000" "-config=/dev/ser2,115200,8,2,NONE,TRUE"** и нажмите ENTER. Данная команда опросит удалённую шину,

подключённую к Modbus1 контроллера СН-МП-ВК. Обратите внимание на параметр 115200 - это скорость обмена, которая руками настраивается на удаленном контроллере. Выставляйте данный параметр в соответствии с установленным на контроллере.

б) **./Sonet_MODBUS "-forced=1;Drv;127.0.0.1:10000" "-config=/dev/ser3,115200,8,2,NONE,TRUE"** и нажмите ENTER. Данная команда опросит удалённую шину, подключённую к Modbus2 контроллера СН-МП-ВК. Обратите внимание на параметр 115200 - это скорость обмена, которая руками настраивается на удаленном контроллере. Выставляйте данный параметр в соответствии с установленным на контроллере.

Запустится драйвер Sonet_MODBUS, который создаст файл Sonet_MODBUS.log и запишет в него информацию о версии драйвера, информацию о процессорном модуле, конфигурацию модулей ввода/вывода на удаённой шине и др.

Остановите работу драйвера, нажав сочетание клавиш CTRL+C, и далее ENTER.

Смотрите информацию в файле Sonet_MODBUS.log.

3.4. Другие программы SCADA-системы "Соната"

3.4.1. Описание работы с приложением BRIDGE (межпроектный обмен)

Приложение **BRIDGE** позволяет обмениваться данными между разными проектами, выполненными в SCADA-системе "Соната". Для работы необходимо в обоих проектах настроить приложение для межпроектного обмена **BRIDGE**.

3.4.1.1. Создание приложения BRIDGE

Создание нового типа приложения в SCADA-системе "Соната" и редактирование его интерфейса, т.е. выбор нужного типа приложения, задание удобного имени приложения и добавление сигналов приложению, осуществляется единообразно для всех приложений (см. раздел 3.2.2.4).

На рис. 3.313 в поле ввода **Вид приложения** необходимо изменить вид приложения **APPLICATION** (присвоенный по умолчанию) на **BRIDGE**. Затем в поле ввода **Имя типа** можно заменить имя на более подходящее, например, **Bridge** (допустимыми символами в имени типа являются большие и маленькие буквы латинского алфавита, цифры и знак подчёркивания, причем, имя типа не может начинаться с цифры).

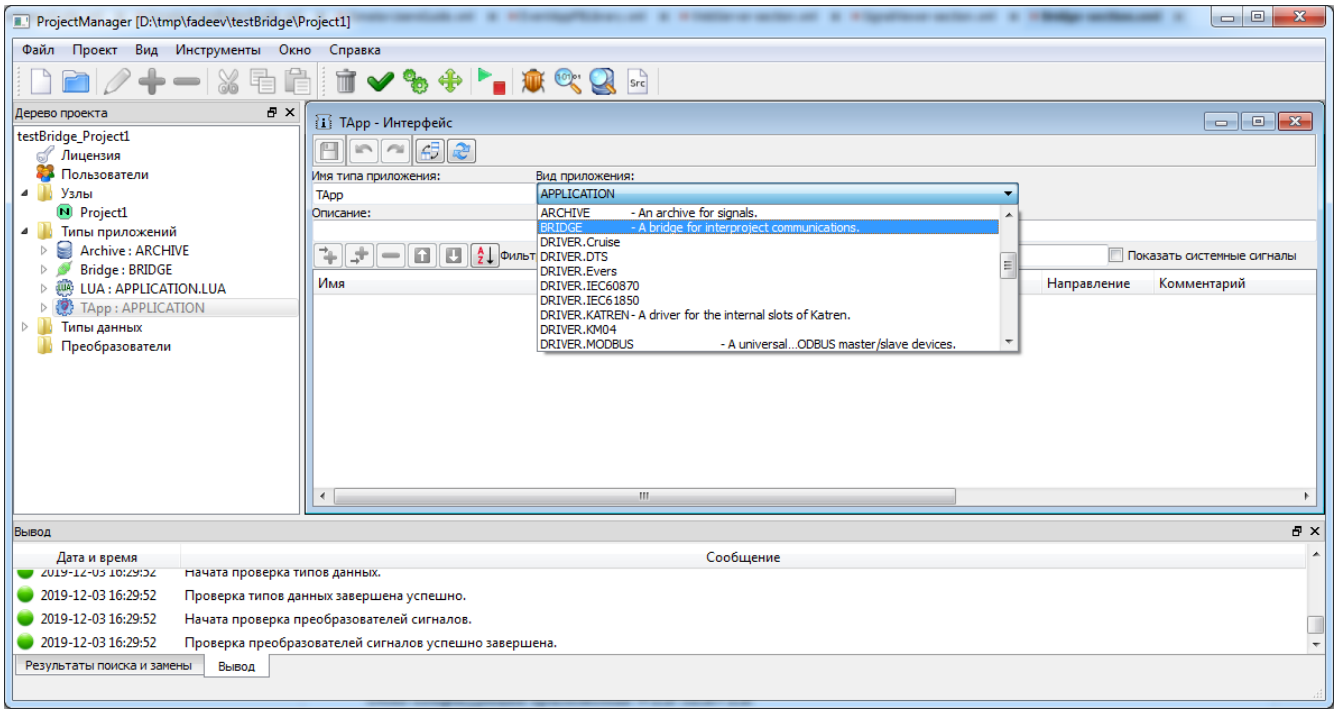


Рисунок 3.313 - Интерфейс приложения BRIDGE

Обратите внимание на параметр **Направление** у сигналов при настройке интерфейса приложения (см. рис. 3.314).

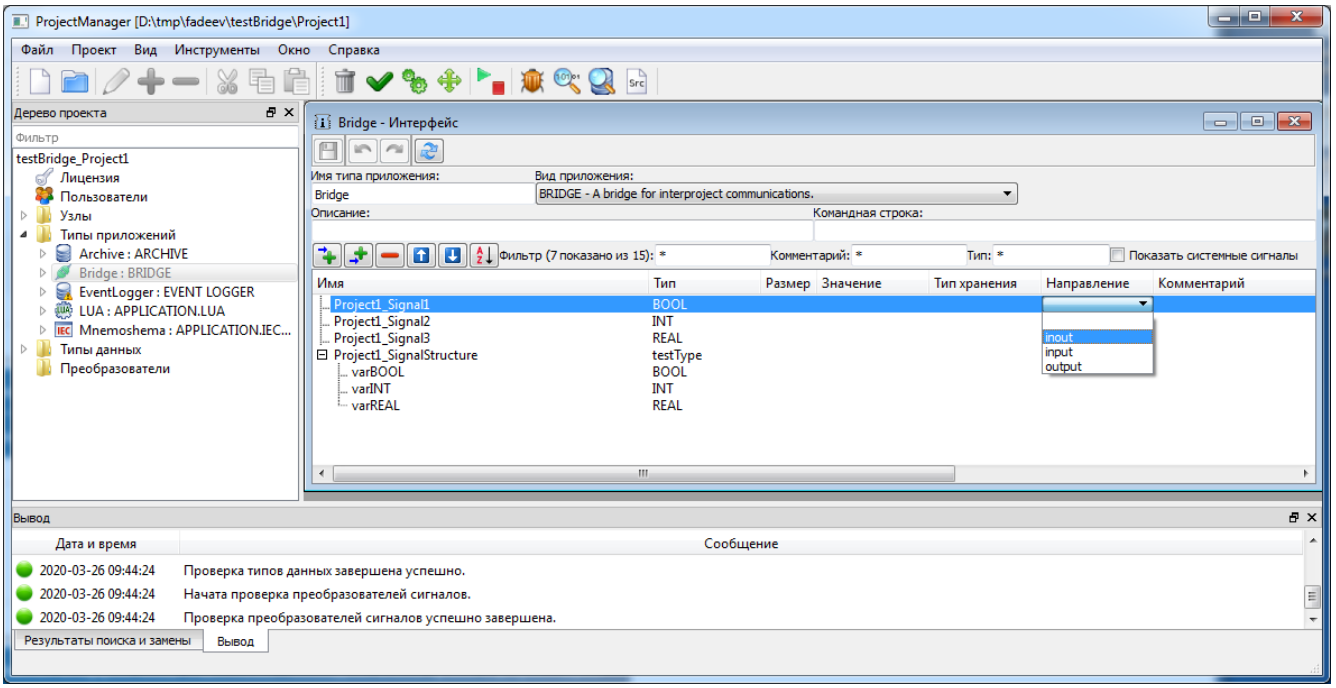


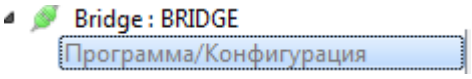
Рисунок 3.314 - Интерфейс приложения BRIDGE с добавленными в него сигналами

Для приложения **BRIDGE** у параметра **Направление** для сигналов используются следующие значения:

- **пустое значение (значение по умолчанию) в обоих проектах** - при такой настройке, связанные между собой проекты будут обмениваться значениями данного сигнала в двухстороннем порядке;

- **input в первом проекте и пустое значение во втором** - при такой настройке, связанные между собой проекты будут обмениваться значениями данного сигнала в одностороннем порядке, т.е первый проект только получает значения, а второй проект только отдает.

3.4.1.2. Описание конфигурации или настройки приложения BRIDGE

Для вызова специализированного редактора приложения **BRIDGE** необходимо раскрыть папку типа приложения в дереве проекта, а затем установить курсор на строку 

, выделить её и нажать кнопку  ("Редактировать элемент") По нажатию на кнопку откроется редактор приложения **BRIDGE** (см. рис. 3.315).

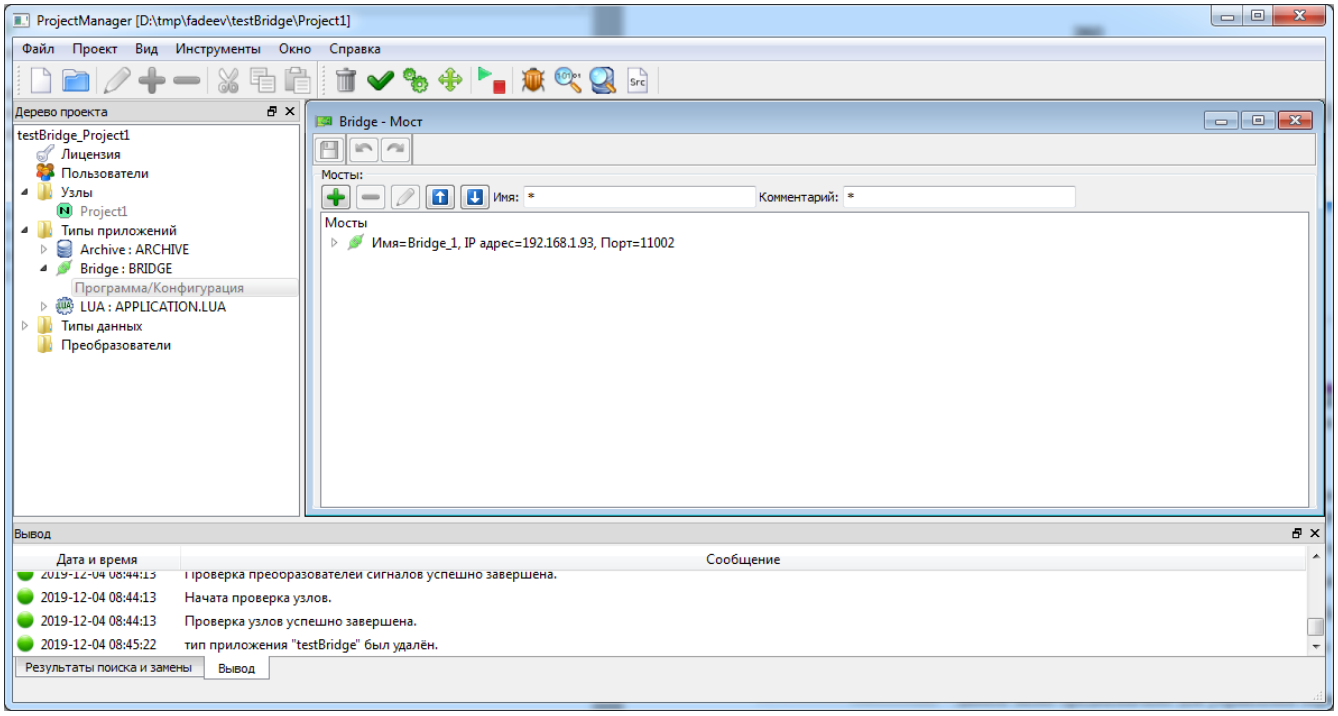
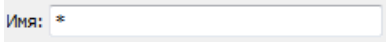


Рисунок 3.315 - Окно редактора настроек приложения BRIDGE

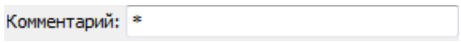
В верхней части окна редактора располагаются кнопки:

-  (Сохранить) - кнопка сохранения внесённых изменений;
-  (Отменить) - кнопка отмены последнего внесенного изменения;
-  (Повторить) - кнопка возврата последнего отмененного изменения.

Ниже располагаются следующие кнопки и фильтры:

-  (Добавить мост) - кнопка добавления записи для настроек межпроектной связи между сигналами;
-  (Удалить мост) - кнопка удаления записи настроек межпроектной связи между сигналами;
-  (Редактировать) - кнопка редактирования свойств межроектной сввязи (моста);
-  (Переместить вверх) - кнопка перемещения вверх строки с настройкой межпроектной связи в общем списке;
-  (Переместить вниз) - кнопка перемещения вниз строки с настройкой межпроектной связи в общем списке;
- 

- фильтр по имени сигнала;

- 

- фильтр по комментарию.

По нажатию на кнопку  (Редактировать) откроется окно настроек свойств межпроектной связи (моста) (см. рис. 3.316).

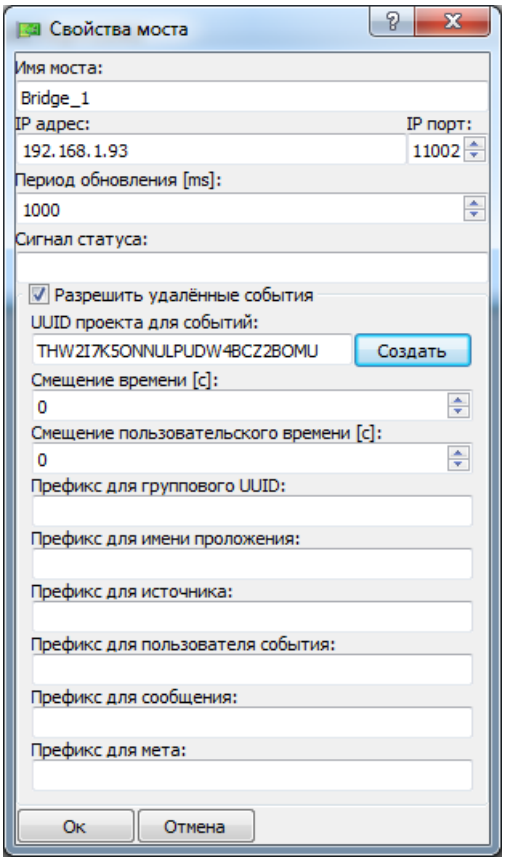


Рисунок 3.316 - Окно настроек межпроектной связи (моста)

В окне настроек свойств моста располагаются следующие поля ввода и кнопки:

- **Имя моста** - имя моста в проекте;

- **IP адрес** - ip адрес моста стороннего проекта, с которым настраивается связь;

- **IP порт** - ip порт приложения BRIDGE в проекте, с которым настраивается связь. Данный порт равен начальному номеру порта, на котором запущен узел стороннего проекта, плюс номер позиции приложения Bridge в списке приложений узла стороннего проекта (см. в настройках узла);

- **Период обновления [ms]** - период принудительного перечитывания значений всех сигналов;

- **Сигнал статуса** - сигнал статуса наличия связи со сторонним мостом;

- **Разрешить удаленные события** - разрешает прохождение событий и тревог из стороннего проекта;

- **UUID проекта для событий** - уникальный иденитификатор событий стороннего проекта;

- **Смещение времени [c]** - смещение времени событий из стороннего проекта при получении в текущий проект;

- **Смещение пользовательского времени [c]** - смещение пользовательского времени событий из стороннего проекта при получении в текущий проект;

- **Префикс для группового UUID** - префикс, добавляемый к групповому UUID событий из стороннего проекта, при получении в текущий проект;

- **Префикс для имени приложения** - префикс, добавляемый к имени приложения, в котором появилось событие из стороннего проекта, при получении в текущий проект;

- **Префикс для источника** - префикс, добавляемый к источнику событий из стороннего проекта, при получении в текущий проект;

- **Префикс для пользователя события** - префикс, добавляемый к имени пользователя, при котором появилось событие из стороннего проекта, при получении в текущий проект;

- **Префикс для сообщения** - префикс, добавляемый к сообщению для события из стороннего проекта, при получении в текущий проект;

- **Префикс для мета** - префикс, добавляемый к мета событий из стороннего проекта, при получении в текущий проект.

Чтобы настроить связь между сигналами разных проектов необходимо (см. рис. 3.317):

1) выделить курсором мыши строку с настройками моста;

2) нажать на кнопку  и добавить нужные для связывания сигналы из интерфейса. Сигналы добавятся в столбец Сигнал;

3) в поле **Подключенный сигнал** прописать сигнал, который будет передаваться из смежного проекта.

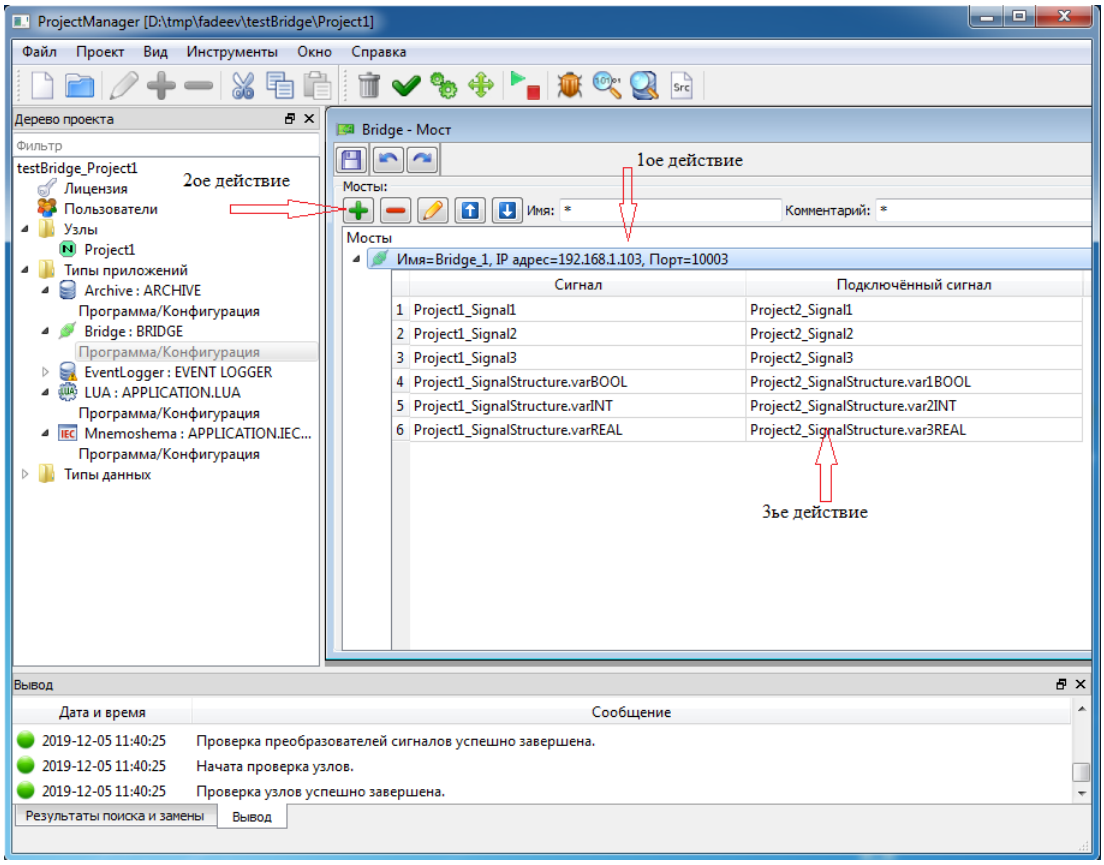


Рисунок 3.317 - Настройки связи сигналов смежных проектов

3.4.1.3. Пример работы с приложением BRIDGE

В данном разделе будет рассмотрен пример настройки межпроектного обмена сигналами и событиями между проектами **testBridge_Project1** и **testBridge_Project2**, которые работают на разных АРМ (АРМ1 с ip адресом 192.168.1.93 и АРМ2 с ip адресом 192.168.1.103).

3.4.1.3.1. Настройка межпроектного обмена сигналами

В проекте **testBridge_Project1** заведены глобальные сигналы **Project1_Signal1:BOOL**, **Project1_Signal2:INT**, **Project1_Signal3:REAL** и сигнал **Project1_SignalStructure** структурного типа **testType** с полями: **varBOOL:BOOL**, **varINT:INT** и **varREAL:REAL** (см. рис. 3.318).

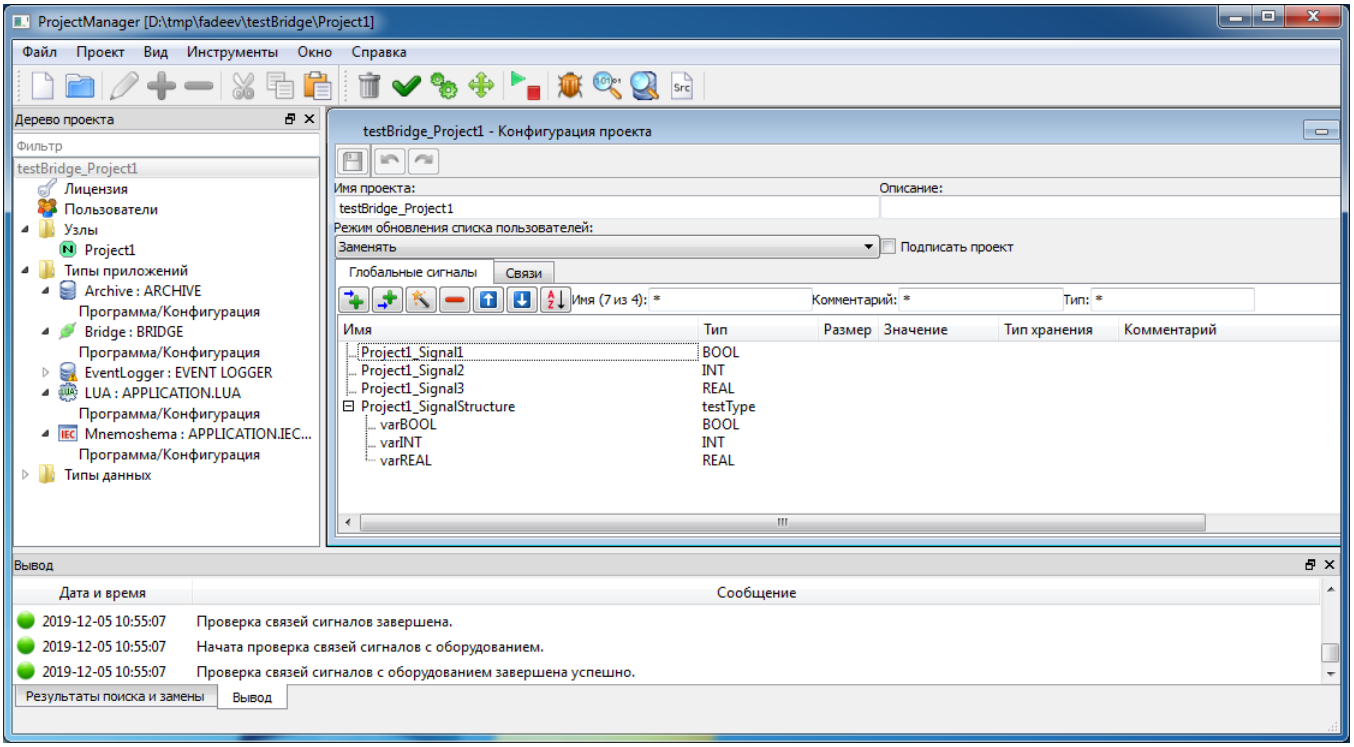


Рисунок 3.318 - Глобальные сигналы проекта testBridge_Project1

В проекте **testBridge_Project2** заведены глобальные сигналы **Project2_Signal1:BOOL**, **Project2_Signal2:INT**, **Project2_Signal3:REAL** и сигнал **Project2_SignalStructure** структурного типа **testType** с полями: **varBOOL:BOOL**, **varINT:INT** и **varREAL:REAL** (см. рис. 3.319).

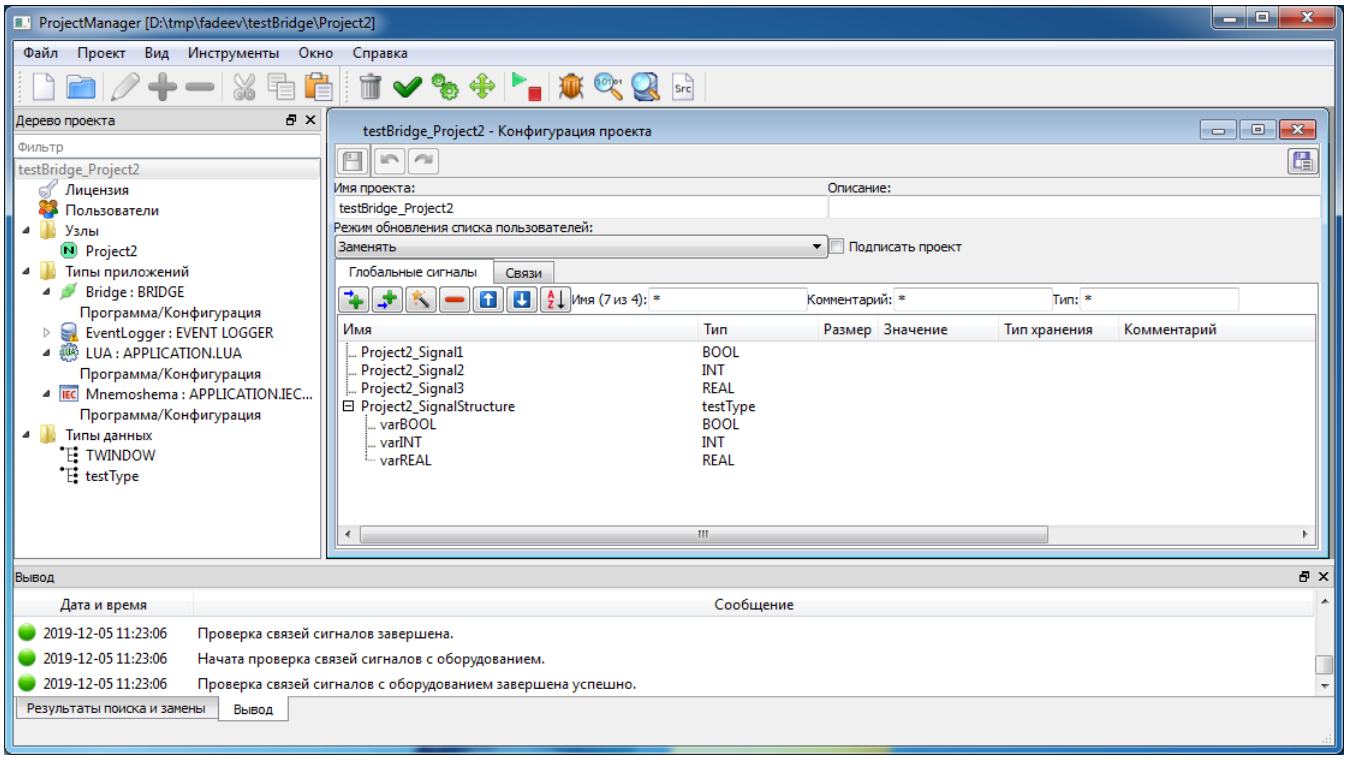


Рисунок 3.319 - Глобальные сигналы проекта testBridge_Project2

Для настройки межпроектного обмена сигналами в обоих проектах необходимо создать и настроить приложение типа **BRIDGE**.

В проекте **testBridge_Project1** создано приложение типа **BRIDGE** с именем приложения **Bridge** и в его интерфейс добавлены глобальные сигналы (см. рис. 3.320).

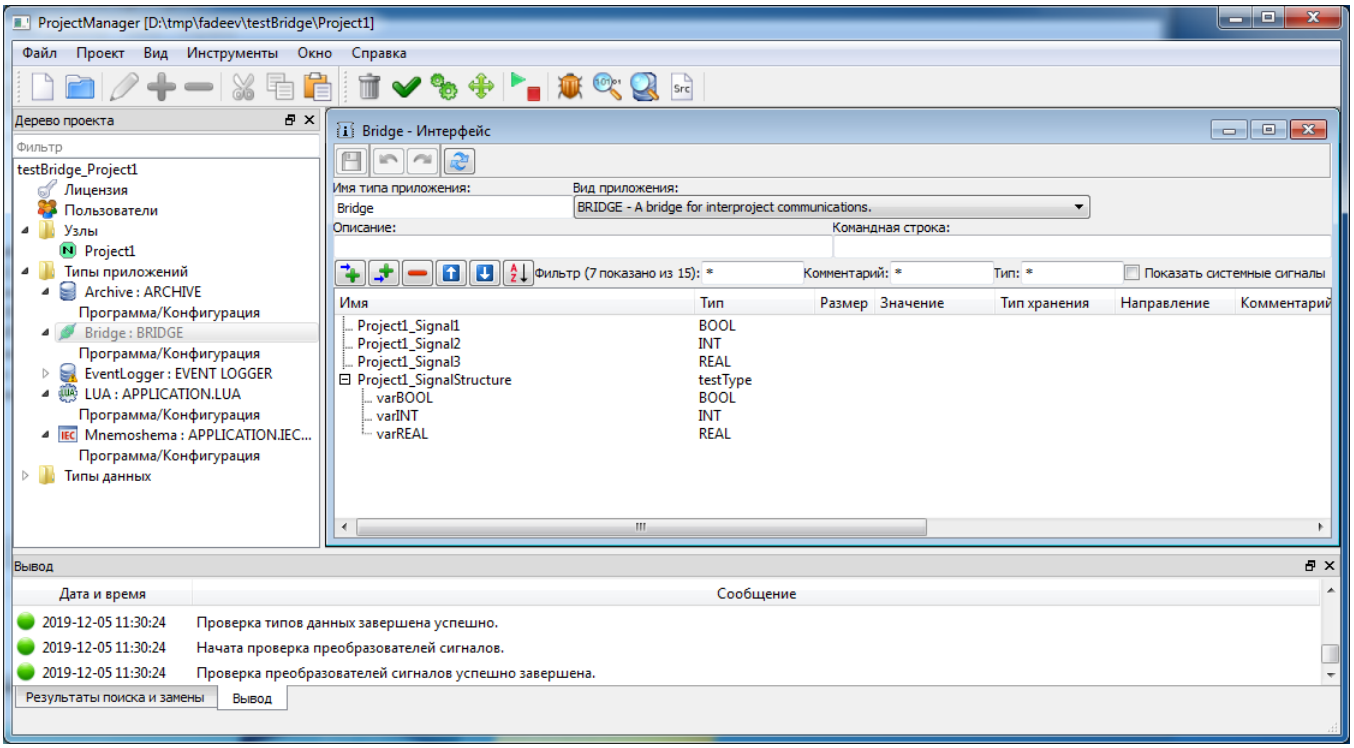


Рисунок 3.320 - Интерфейс приложения BRIDGE проекта testBridge_Project1

В проекте **testBridge_Project2** создано приложение типа **BRIDGE** с именем приложения **Bridge** и в его интерфейс добавлены глобальные сигналы (см. рис. 3.321).

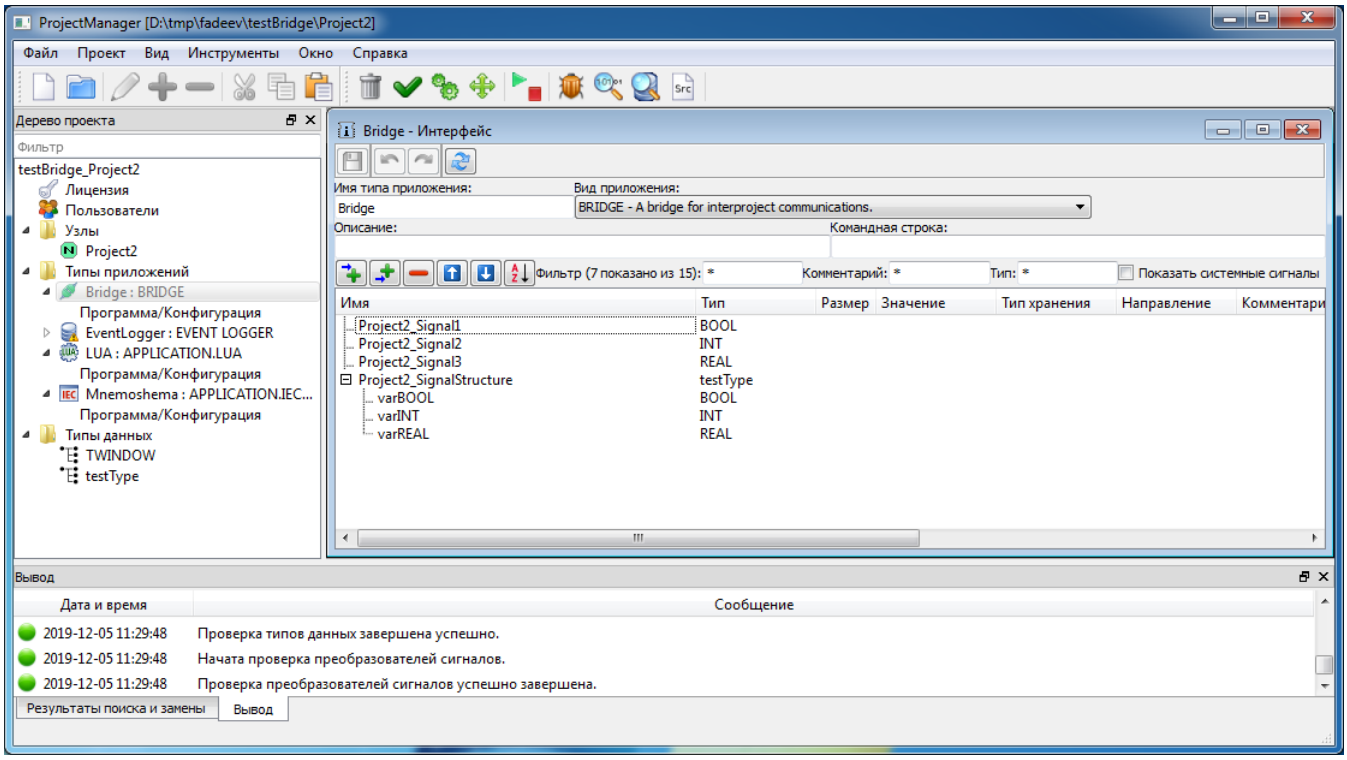


Рисунок 3.321 - Глобальные сигналы проекта testBridge_Project2

В проекте **testBridge_Project1** создан и настроен узел с именем **Project1**, на котором мы будем запускать наши приложения. Необходимо обратить внимание на следующие настройки: начальный ip порт равный 10000, порядковый номер приложения Bridge в списке приложений равный 4 и на имя, которое мы написали приложению Bridge в данном списке, равное Мост (см. рис. 3.322).

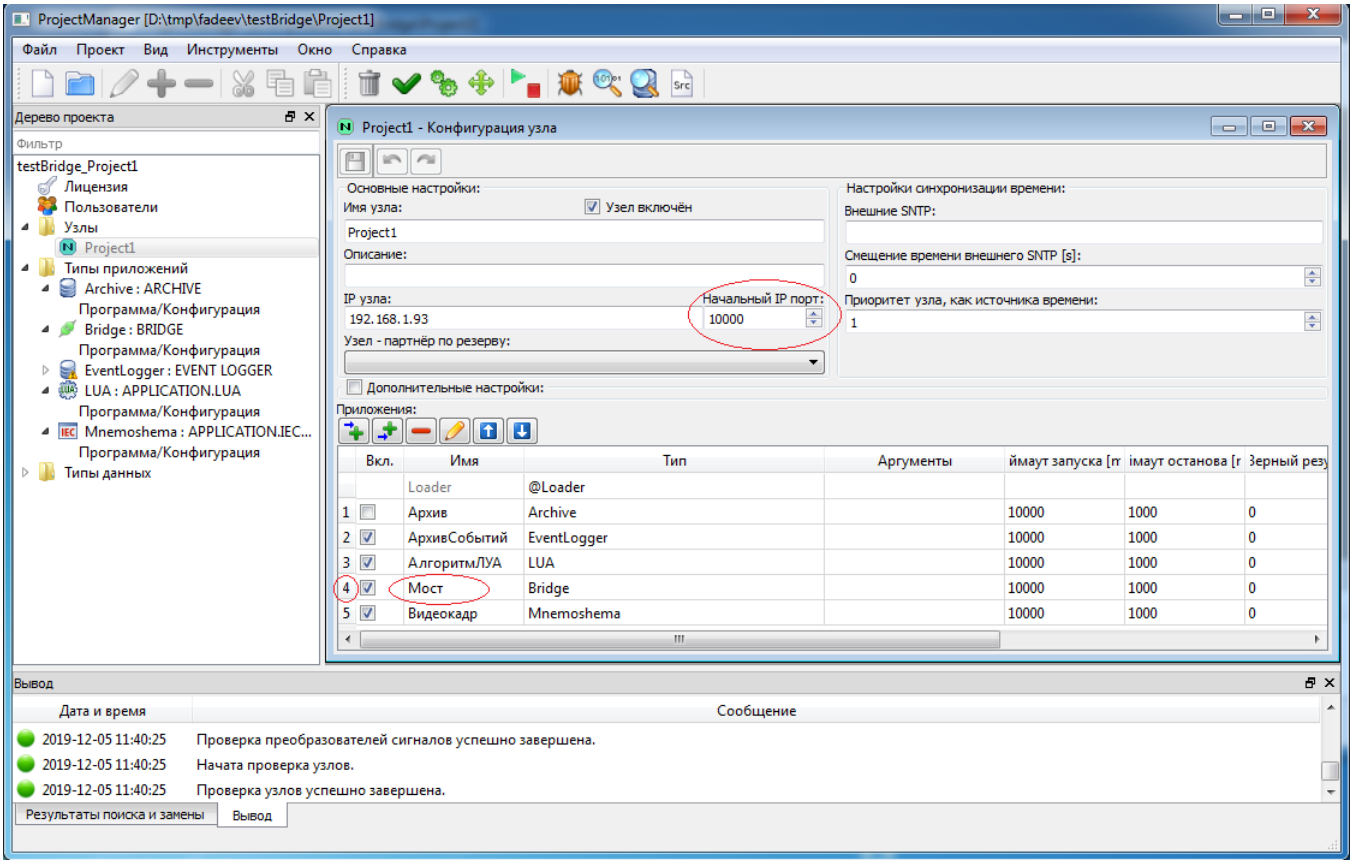


Рисунок 3.322 - Настройки узла в проекте testBridge_Project1

В проекте **testBridge_Project2** создан и настроен узел с именем **Project2**, на котором мы будем запускать наши приложения. Необходимо обратить внимание на следующие настройки: начальный ip порт равный 10000, порядковый номер приложения Bridge в списке приложений равный 3 и на имя, которое мы написали приложению Bridge в данном списке, равное Мост (см. рис. 3.323).

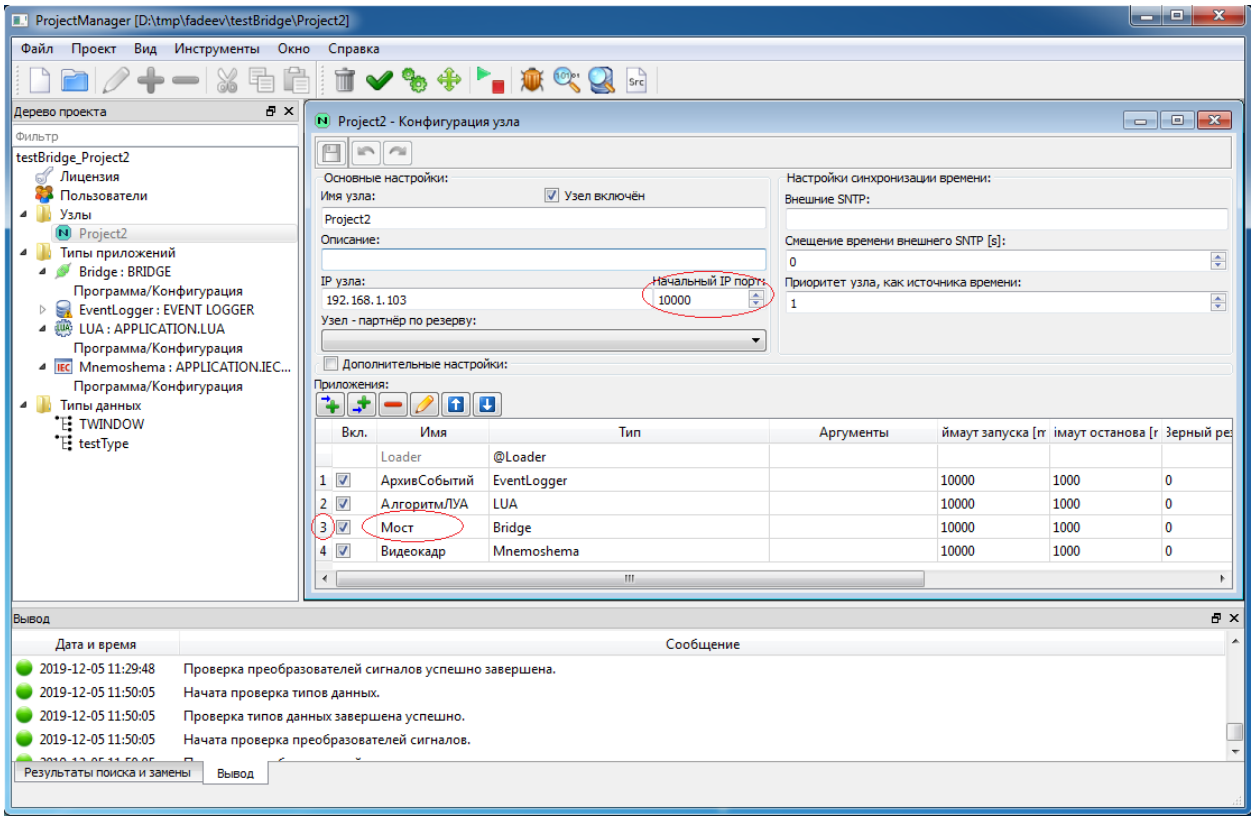
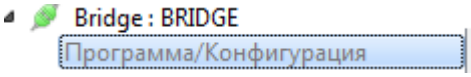


Рисунок 3.323 - Настройки узла в проекте testBridge_Project2

После этого нужно зайти в пункт 

и выпонить настройки приложения **Bridge**.

Для конфигурации межпроектного обмена сигналами в проекте **testBridge_Project1** необходимо выполнить следующие настройки (см. рис. 3.324):

- ip адрес проекта, с которым настраивается межпроектная связь. В нашем случае ip адрес проекта **testBridge_Project2** равный 192.168.1.103;

- ip порт приложения Bridge в проекте, с которым настраивается связь. В нашем случае ip порт проекта **testBridge_Project2** равный 10003.

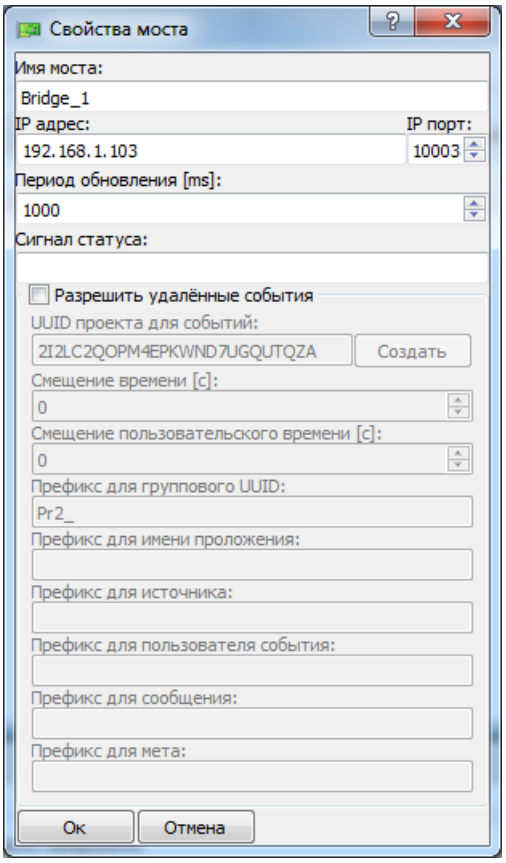


Рисунок 3.324 - Окно конфигурации свойств моста в проекте testBridge_Project1

Для конфигурации межпроектного обмена сигналами в проекте **testBridge_Project2** необходимо выполнить следующие настройки (см. рис. 3.325):

- ip адрес проекта, с которым настраивается межпроектная связь. В нашем случае ip адрес проекта **testBridge_Project1** равный 192.168.1.93;

- ip порт приложения Bridge в проекте, с которым настраивается связь. В нашем случае ip порт проекта **testBridge_Project1** равный 10004.

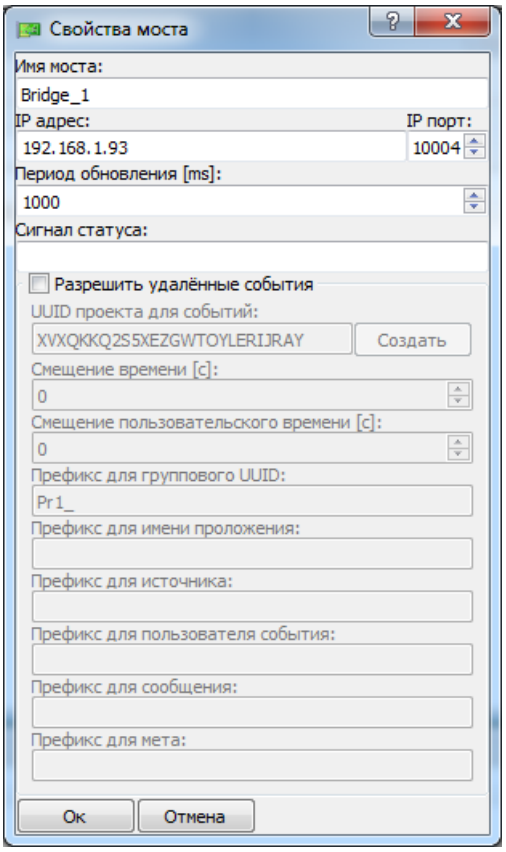


Рисунок 3.325 - Окно конфигурации свойств моста в проекте testBridge_Project2

Далее необходимо настроить связь сигналов проекта **testBridge_Project1** и проекта **testBridge_Project2** между собой. Для этого необходимо в каждом проекте в конфигурации приложения **Bridge** выполнить следующие действия (см. рис. 3.326 и рис. 3.327):

- 1 - выделить курсором мыши строку с настройками моста;

- 2 - нажать на кнопку  и добавить нужные для связывания сигналы из интерфейса;

- 3 - в поле **Подключенный сигнал** прописать сигнал, который будет передаваться из смежного проекта.

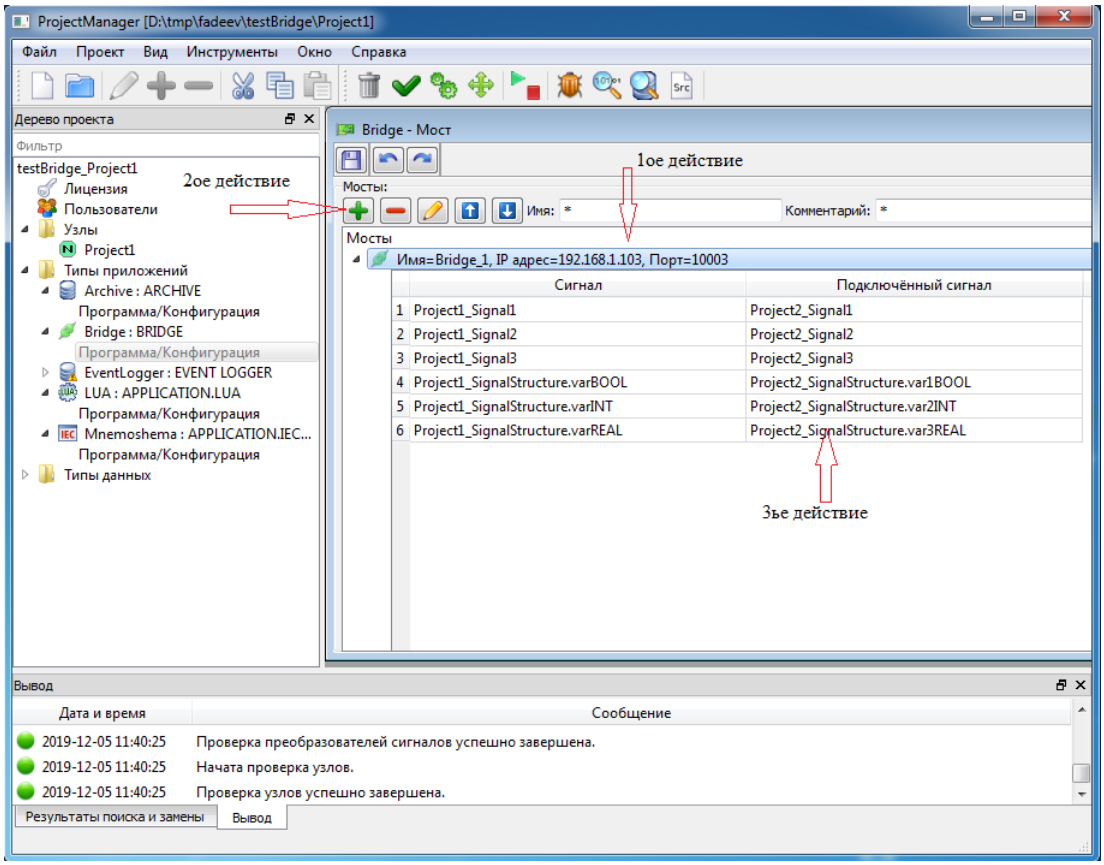


Рисунок 3.326 - Настройки связи сигналов смежных проектов в проекте testBridge_Project1

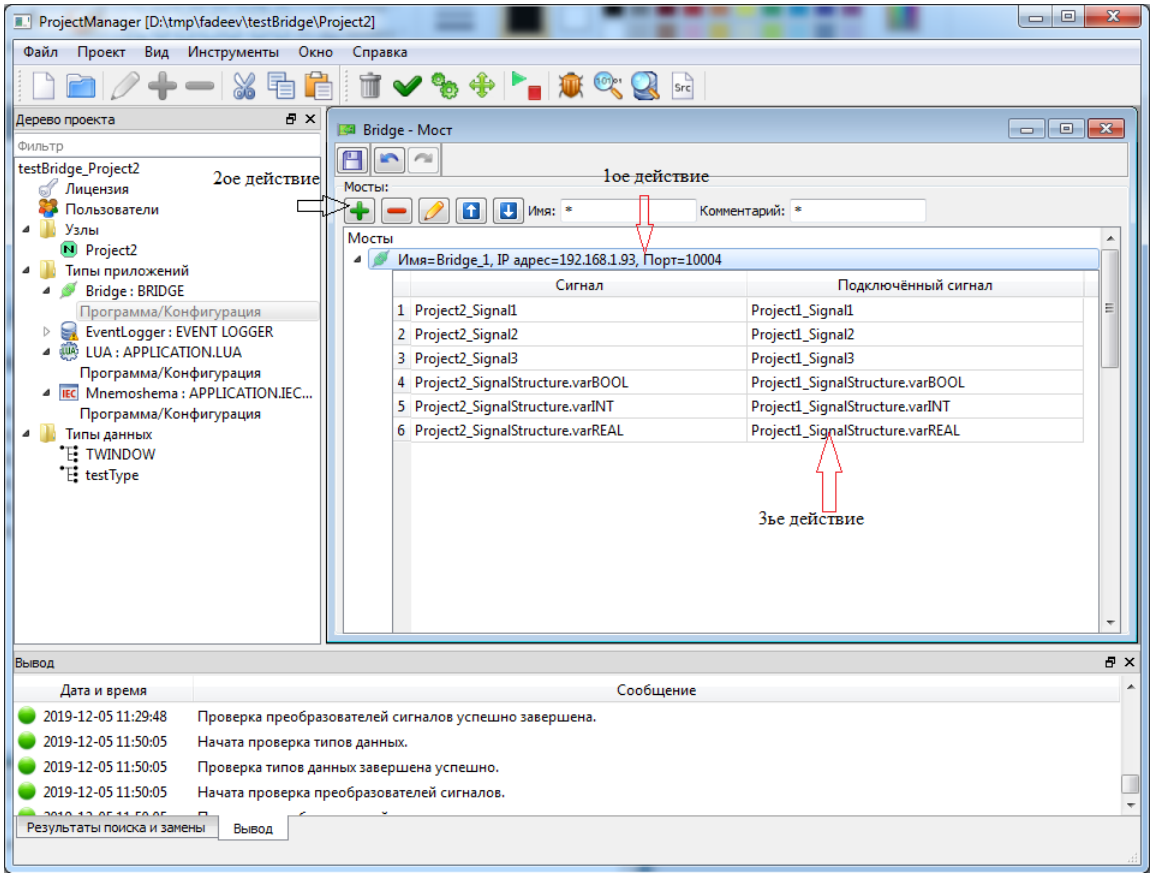


Рисунок 3.327 - Настройки связи сигналов смежных проектов в проекте testBridge_Project2

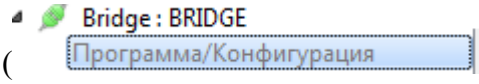
На этом настройка межпроектной связи для сигналов SCADA-системы "Соната" завершена. Связанные сигналы между проектами будут синхронизироваться.

3.4.1.3.2. Настройка межпроектной связи для передачи событий между проектами

Для настройки прохождения событий из одного проекта в другой через приложения **BRIDGE** необходимо настроить:

- прохождение событий в приложениях **BRIDGE**;
- список источников данных в приложениях-мнемосхемах (APPLICATION.IEC.WINDOW);
- указать источники данных в списках событий и тревог (EventViewer и AlarmViewer).

3.4.1.3.2.1. Настройка прохождения событий в приложениях BRIDGE

Для настройки прохождения событий между проектами необходимо войти в редактор приложения BRIDGE (

). Далее выделить строку с настройками моста и нажать кнопку  (Редактировать, см. рис. 3.328). В результате откроется окно конфигурации свойств моста (см. рис. 3.329).

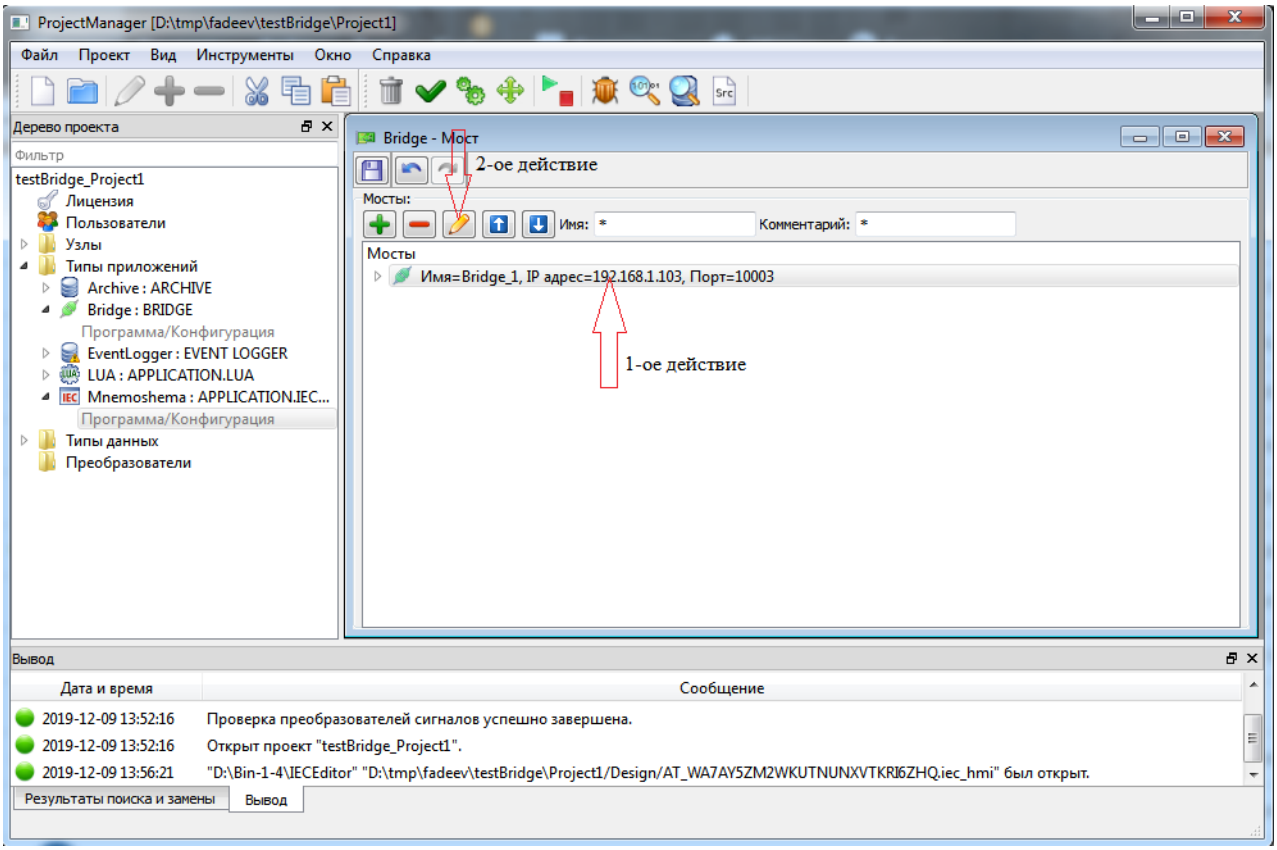


Рисунок 3.328 - Действия по открытию свойств моста в редакторе

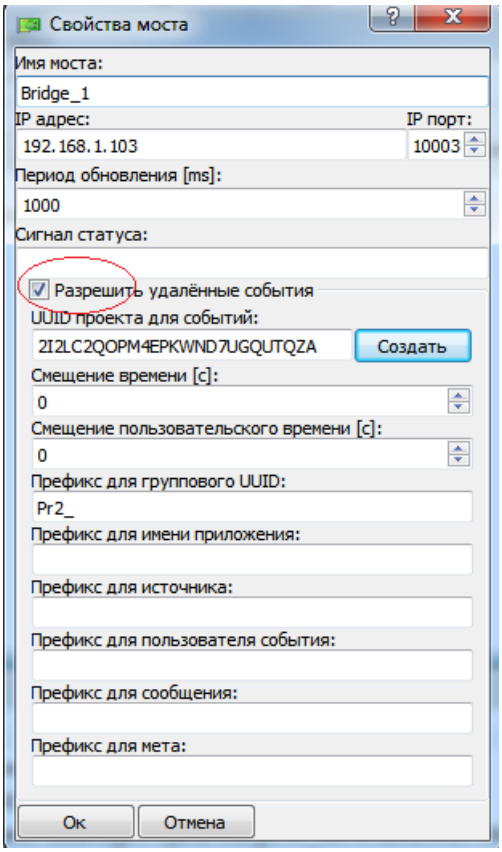


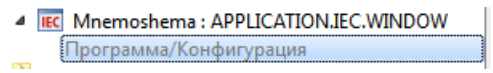
Рисунок 3.329 - Окно редактора свойств моста проекта testBridge_Project1

В окне свойств моста необходимо поставить галочку **Разрешить удаленные события** и настроить соответствующие поля (см. рис. 3.316). Обязательным для заполнения является только поле **UUID проекта для событий** (оно заполнено автоматически).

Аналогично можно настроить и свойства моста в проекте **testBridge_Project2**. Для одностороннего обмена достаточно выполнить настройки только для проекта получателя.

3.4.1.3.2.2. Настройка списка источников данных в приложениях-мнемосхемах (APPLICATION.IEC.WINDOW)

Создадим в наших проектах **testBridge_Project1** и **testBridge_Project2** приложения APPLICATION.IEC.WINDOW (приложения для графических кадров). Данные приложения в нашем примере названы **Mnemoshema**. Для межпроектной связи никаких настроек в интерфейсе делать не нужно. Необходимо открыть данное приложение на редактирование



и в верхнем меню выбрать пункт **Редактировать - Приложение - Настройка событий и тревог**. Откроется окно настройки событий и тревог (см. рис. 3.330).

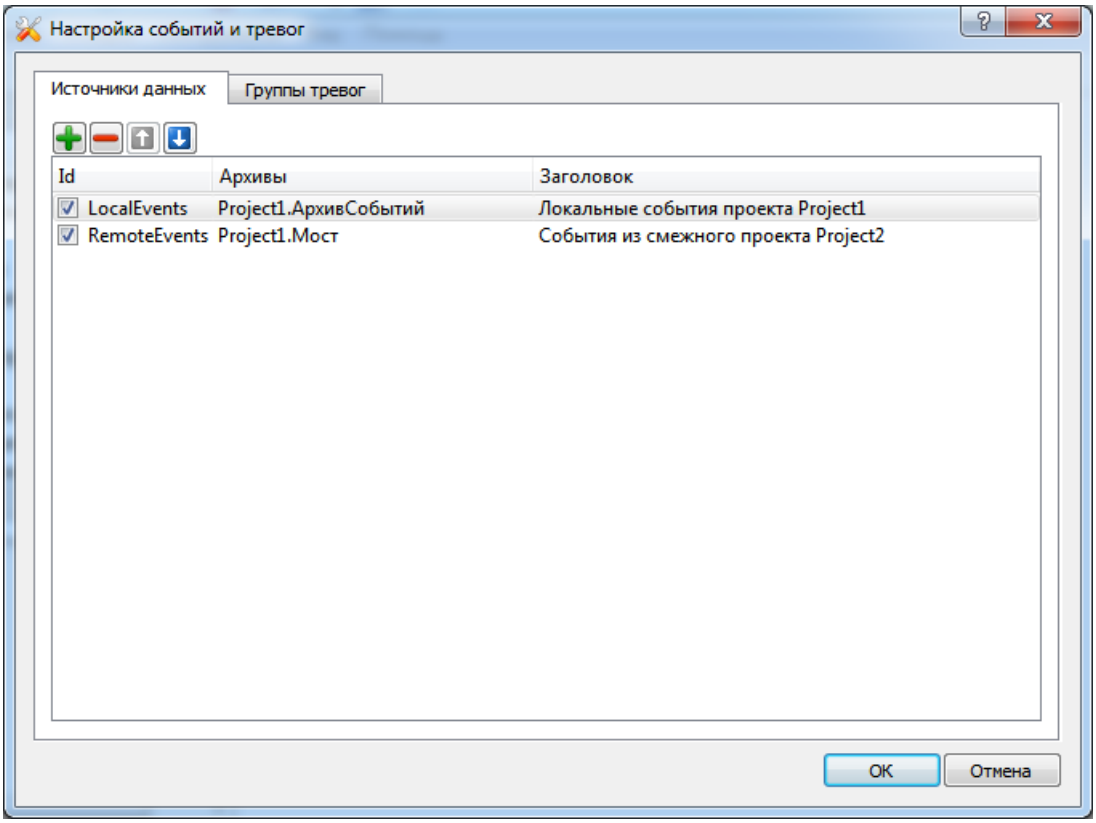


Рисунок 3.330 - Окно настроек событий и тревог проекта testBridge_Project1

Во вкладке **Источники данных** располагаются следующие кнопки:

- - кнопка добавления источника событий и тревог;

- - кнопка удаления источника событий и тревог;

- - кнопка перемещения вверх строки с источником событий и тревог в общем списке;

- - кнопка перемещения вниз строки с источником событий и тревог в общем списке.

Ниже располагается окно со списком источников событий и тревог. В данном списке находятся следующие столбцы:

- **Id** - в данном столбце располагается галочка включения/выключения источника из используемых источников событий и тревог и поле для ввода пользовательского идентификатора (данный идентификатор будет использоваться в дальнейших настройках);

- **Архивы** - в данном столбце необходимо прописать полное имя приложения (Имя_узла.Имя_приложения), которое является архивом для событий и тревог. Архивом событий и тревог смежного проекта является приложение BRIDGE, в котором настроена данная межпроектная связь;

- **Заголовок** - в данном столбце необходимо написать комментарий для определения данного источника событий и тревог.

В нашем примере для проекта **testBridge_Project1** настроены два источника: **LocalEvents**, который берет события из приложения **EventLogger**, и **RemoteEvents**, который берет события из приложения **Bridge** (см. рис. 3.331 и рис. 3.330).

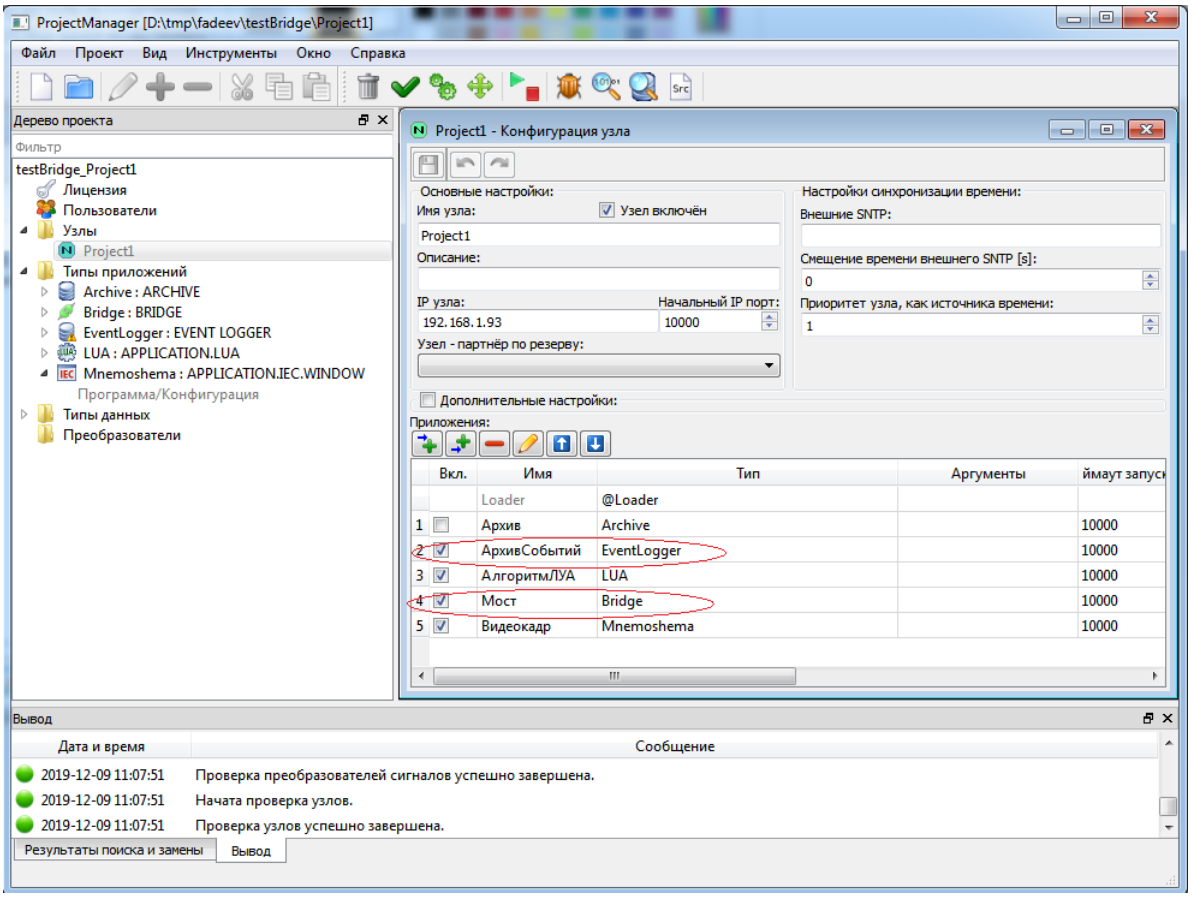


Рисунок 3.331 - Окно настроек узла Project1 в проекте testBridge_Project1

Для проекта **testBridge_Project2** настроены два источника: **LocalEvents**, который берет события из приложения **EventLogger**, и **RemoteEvents**, который берет события из приложения **Bridge** (см. рис. 3.332 и рис. 3.333).

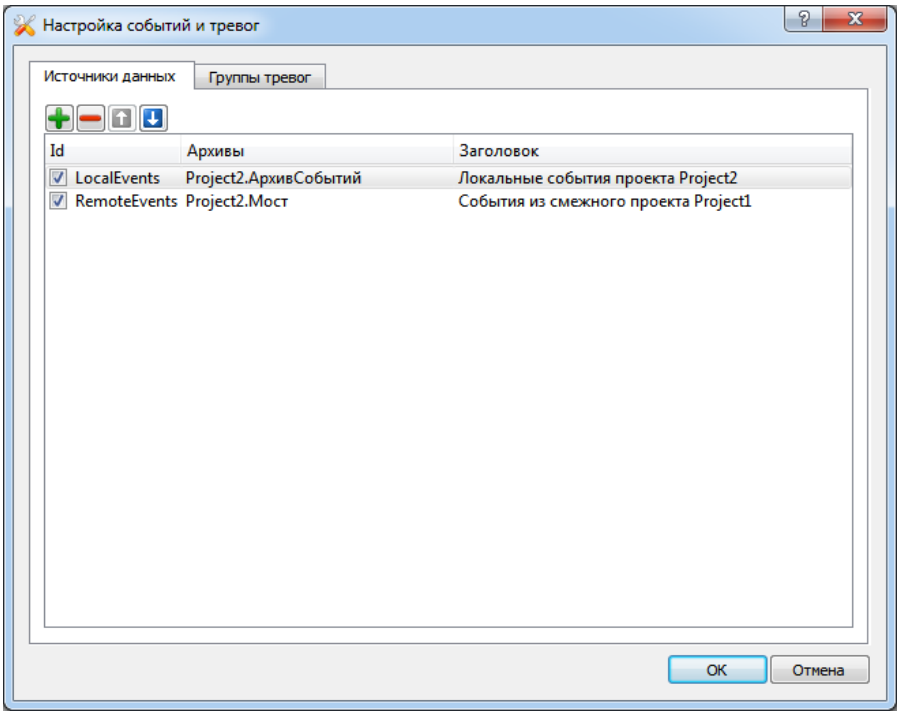


Рисунок 3.332 - Окно настроек событий и тревог проекта testBridge_Project2

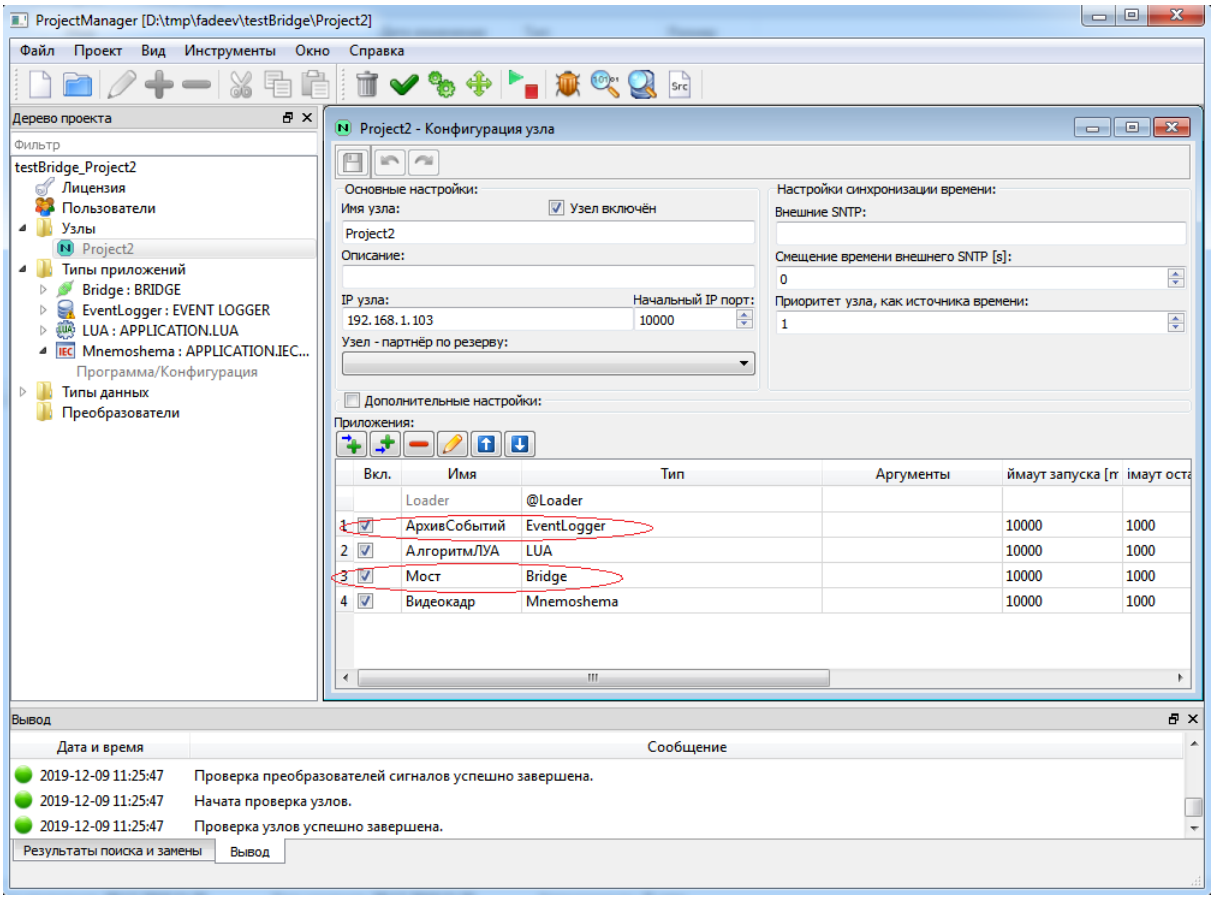


Рисунок 3.333 - Окно настроек узла Project2 в проекте testBridge_Project2

3.4.1.3.2.3. Настройка FBD блоков отображения событий и тревог или как указать источники данных в списках событий и тревог

Для отображения событий и тревог на видеокадрах (мнемосхемах) в SCADA системе "Соната" используются функциональные блоки **EventViewer** и **AlarmViewer** (описание данных функциональных блоков приведено в Руководстве программиста КУНИ.505200.023-01.01 33).

Настроим функциональный блок **EventViewer** в проекте **testBridge_Project1** чтобы он отображал события из проекта **testBridge_Project2**.

Для этого необходимо:

1) Добавить функциональный блок **EventViewer** на видеокадр;

2) Нажать правой кнопкой мыши на функциональный блок **EventViewer** и в появившемся контекстном меню выбрать пункт Настроить объект (см. рис. 3.334);

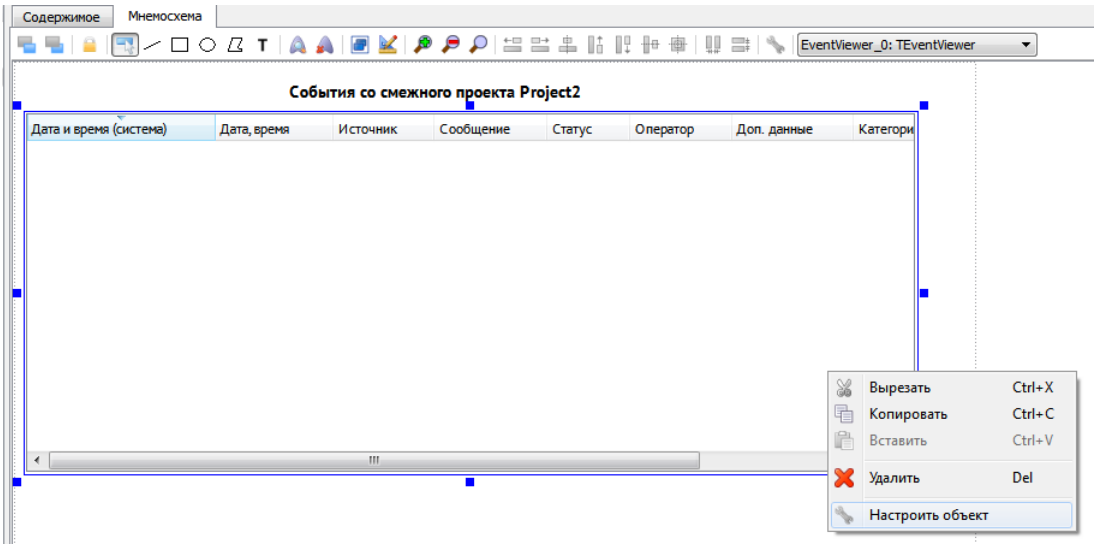


Рисунок 3.334 - Контекстное меню функционального блока EventViewer

3) Откроется окно настроек списка событий, в котором нужно выбрать вкладку Прочие (см. рис. 3.335);

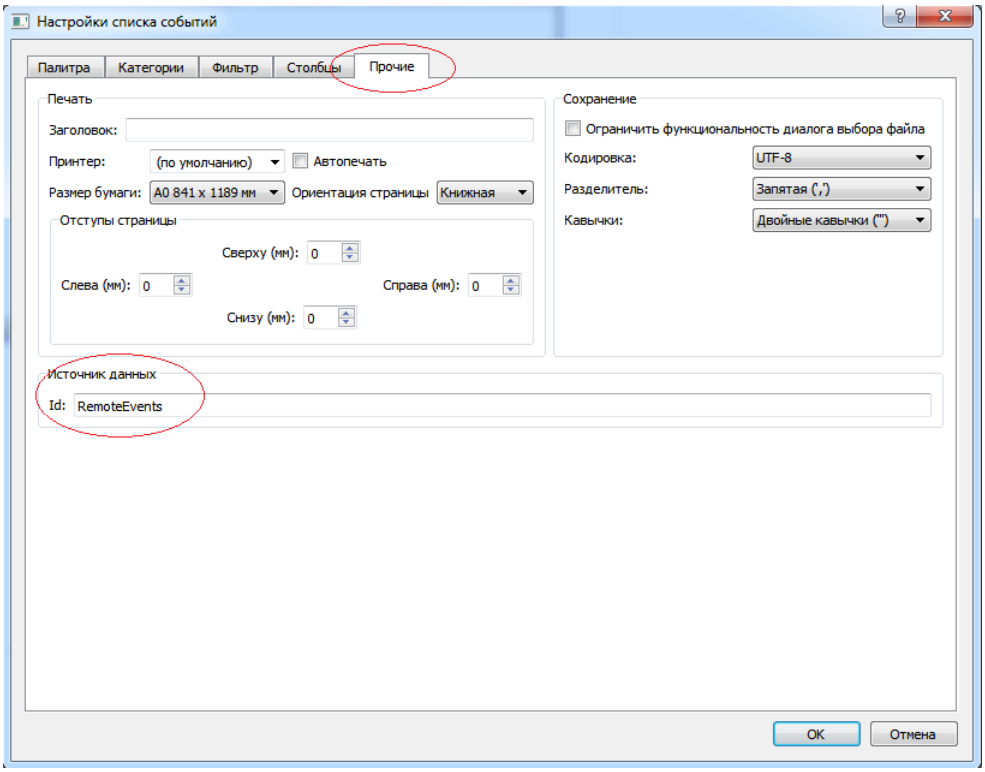


Рисунок 3.335 - Окно настроек списка событий функционального блока EventViewer

4) Во вкладке Прочие в поле ввода **Источник данных** необходимо указать из какого источника будут браться события (источник можно указать только один). В данном случае указан источник **RemoteEvents** (см. рис. 3.335), который связан с приложением **Project2.Мост** (настраивали ранее), а значит в данном функциональном блоке **EventViewer** будут отображаться события из проекта **testBridge_Project2**.

Функциональный блок **AlarmViewer** настраивается аналогично.

Для того чтобы отображать события из разных источников необходимо добавить несколько функциональных блоков **EventViewer** и **AlarmViewer** и указать им разные источник событий.

На этом настройка межпроектной связи для передачи событий и тревог закончена.

3.4.2. Описание программы "Отладчик" (Debugger)

Программа **Debugger** (далее Отладчик) входит в состав SCADA-системы "Соната" и предназначена для отладки консольных приложений, таких как APPLICATION.ST.CONSOLE, APPLICATION.LUA и других.

3.4.2.1. Интерфейс отладчика

Программа Отладчик вызывается из раздела Проект главного меню программы **ProjectManager** (Менеджер проектов) нажатием на пункт Отладчик или нажатием на кнопку ("Отладчик") панели инструментов Менеджера проектов.

Главное окно программы Отладчик представлено на рис. 3.336.

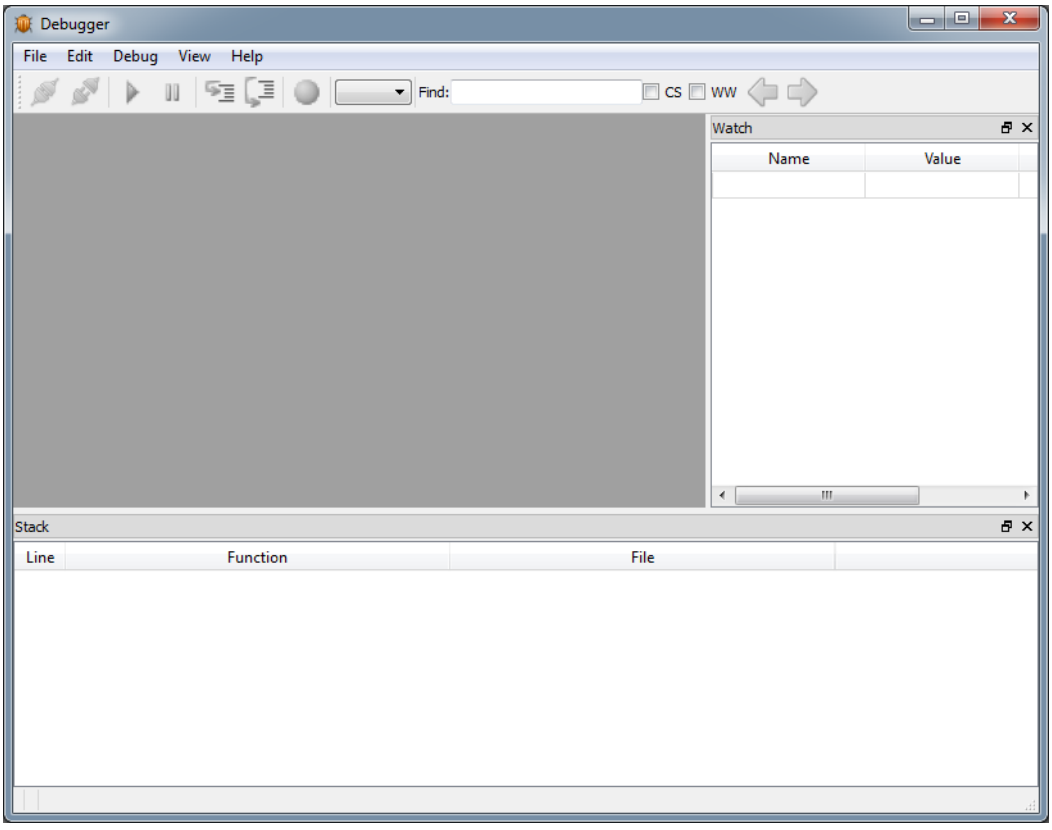
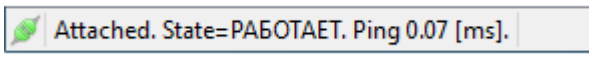


Рисунок 3.336 - Главное окно программы Отладчик

Главное окно программы Отладчик состоит из следующих элементов:

- в верхней левой части располагается название программы Debugger;
- в верхней правой части находятся кнопки управления окном редактора, соответственно свернуть, развернуть и закрыть окно редактора;
- далее располагается главное меню программы (см. раздел 3.4.2.1.1);
- ниже располагается панель инструментов (см. раздел 3.4.2.1.2);
- в средней части слева располагается окно кода, который подлежит отладке (см. раздел 3.4.2.1.3);
- в средней части справа располагается окно **Watch** (окно просмотра переменных, см. раздел 3.4.2.1.4);

- в нижней части располагается окно **Stack** (см. раздел 3.4.2.1.5);
- в самом низу располагается информационная строка, в которой отображается информация о состоянии подключения к процессу (к примеру 

- сообщает, что подключение есть и связь хорошая).

3.4.2.1.1. Описание главного меню

Главное меню программы Отладчик состоит из следующих разделов:
- File - раздел по работе с файлом (см. раздел 3.4.2.1.1.1);
- Edit - раздел редактировать (см. раздел 3.4.2.1.1.2);
- Debug - раздел с командами отладки (см. раздел 3.4.2.1.1.3);
- View - раздел отображения элементов главного окна (см. раздел 3.4.2.1.1.4);
- Help - раздел "Помощь" (см. раздел 3.4.2.1.1.5).

3.4.2.1.1.1. Описание раздела File

На рис. 3.337 представлено развёрнутое меню раздела File.

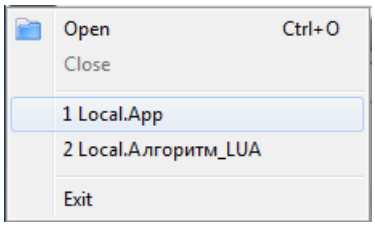
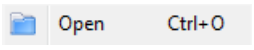


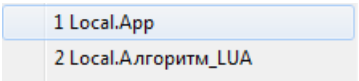
Рисунок 3.337 - Развёрнутое меню раздела File

В раздел File входят следующие команды:



- открыть файл с кодом для отладки;

- закрыть файл с кодом;



- список быстрого вызова процессов, с которыми недавно работали;

- закрыть окно программы Отладчик.

3.4.2.1.1.2. Описание раздела Edit

На рис. 3.338 представлено развёрнутое меню раздела Edit.

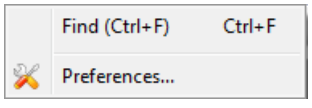
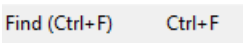
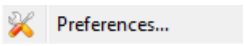


Рисунок 3.338 - Развёрнутое меню раздела Edit

В раздел Edit входят следующие команды:



- поиск слова в тексте;



- настройки (пока не используются).

3.4.2.1.1.3. Описание раздела Debug

На рис. 3.339 представлено развёрнутое меню раздела Debug.

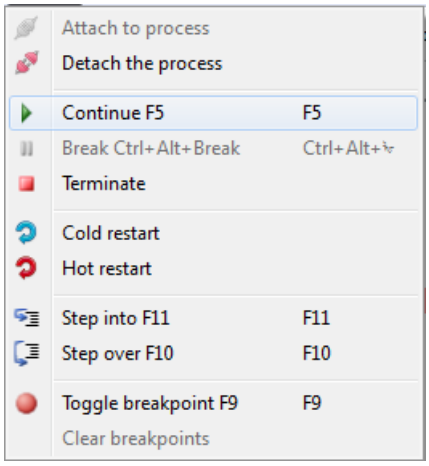
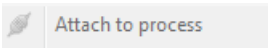
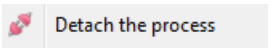


Рисунок 3.339 - Развёрнутое меню раздела Debug

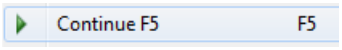
В раздел Debug входят следующие команды:



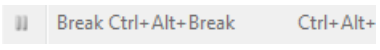
- подключиться к рабочему процессу;



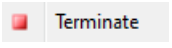
- отключиться от процесса;



- продолжить исполнение кода;



- остановить(поставить на паузу) исполнение кода;



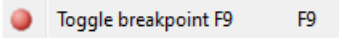
- удаляет процесс, исполняющий код;

- перезапуск процесса, исполняющего код, в режиме "холодного перезапуска";

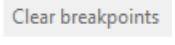
- перезапуск процесса, исполняющего код, в режиме "горячего перезапуска";

- шаг внутрь цикла;

- шаг поверх цикла;



- установить точку останова;



- удалить точки останова.

3.4.2.1.1.4. Описание раздела View

На рис. 3.340 представлено развёрнутое меню раздела View.

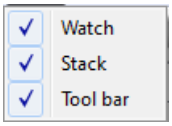


Рисунок 3.340 - Развёрнутое меню раздела View

В раздел View входят следующие команды:

Watch - добавить/удалить окно **Watch**;

Stack - добавить/удалить окно **Stack**;

Tool bar - добавить/удалить панель инструментов.

3.4.2.1.1.5. Описание раздела Help

В раздел Help входит одна команда - About, которая выводит информацию о программе Отладчик.

3.4.2.1.2. Описание панели инструментов

Панель инструментов располагается под главным меню и содержит наиболее часто используемые команды.

В состав панели инструментов входят следующие команды-кнопки:

- подключиться к процессу (из раздела главного меню Debug);
- отключиться от процесса (из раздела главного меню Debug);
- запустить выполнение процесса (из раздела главного меню Debug);
- остановить процесс (из раздела главного меню Debug);
- двигаться с шагами внутри циклов (из раздела главного меню Debug);
- двигаться шагами, не заходя внутрь циклов (из раздела главного меню Debug);
- поставить точку останова на выделенную строку (из раздела главного меню Debug);

Lua - выбрать язык, на котором написан код;

Find: - строка ввода слова для поиска (из раздела главного меню Edit);

CS - проводить поиск с учетом регистра;

WW - искать слово целиком;

- проводить поиск вверх по тексту;
- проводить поиск вниз по тексту.

3.4.2.1.3. Окно отладочного кода

Окно отладочного кода располагается в центре слева на общем окне программы Отладчик. При подключении к рабочему процессу приложения SCADA-системы "Соната" в нём отображается код, по которому работает процесс (см. рис. 3.341).

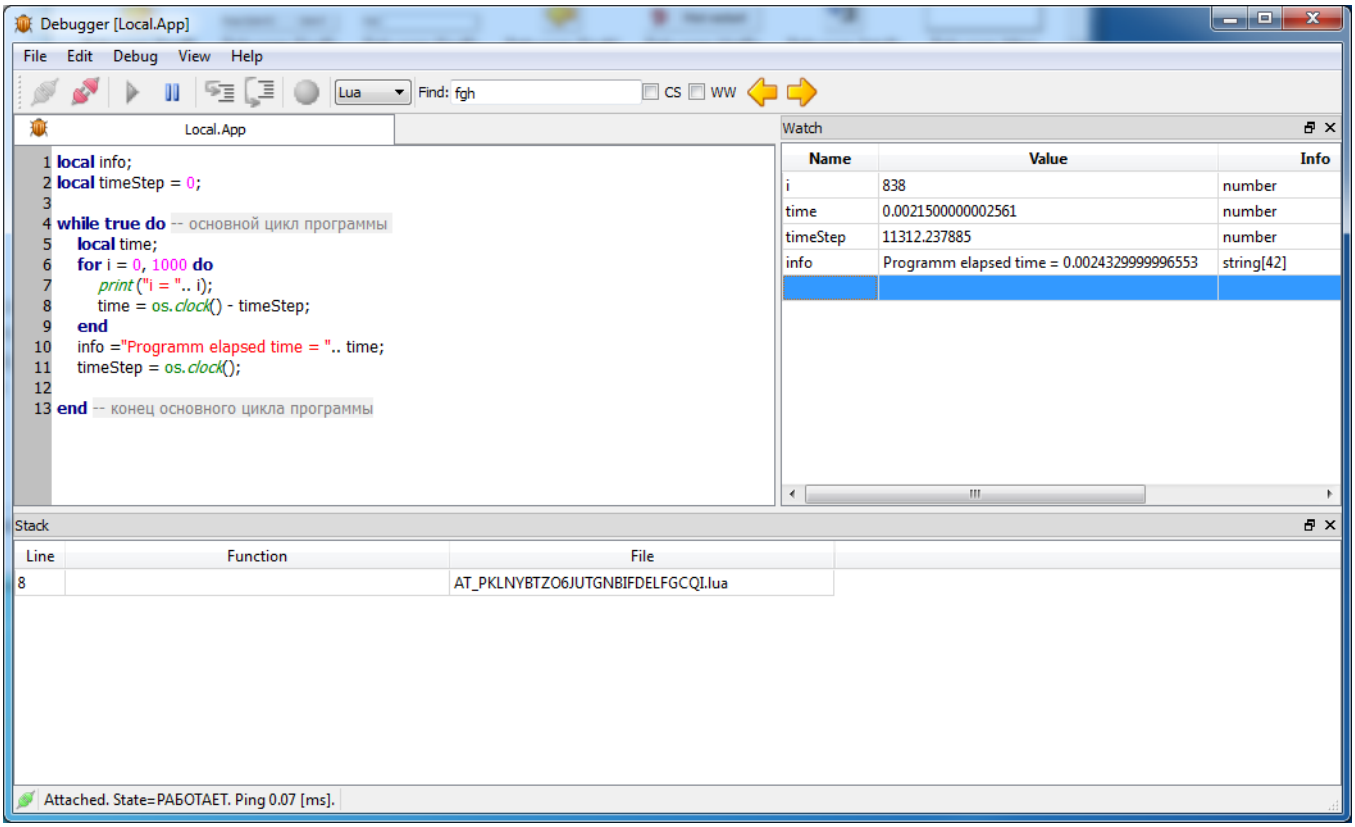


Рисунок 3.341 - Окно программы Отладчик с кодом процесса для отладки

В окне отладочного кода нельзя вносить изменения, а можно только выделять нужные строки и ставить точки останова.

3.4.2.1.4. Описание окна Watch (анализа переменных)

Окно **Watch** располагается в центре справа на общем окне программы Отладчик (см. рис. 3.341). Оно предназначено для просмотра значений переменных запущенного процесса.

Окно **Watch** состоит из следующих столбцов:

- **Name** - имя переменной;
- **Value** - значение переменной;
- **Info** - информация о переменной.

3.4.2.1.5. Описание окна Stack

Окно **Stack** располагается в нижней части на общем окне программы Отладчик (см. рис. 3.341). Данное окно является информационным и состоит из следующих столбцов:

- **Line** - номер строки, на которой остановлена в данный момент программа;
- **Function** - имя функции;
- **File** - имя файла, в котором содержится отладочный код.

3.4.2.2. Описание работы в программе Отладчик

Основной задачей программы Отладчик является отладка и анализ кода алгоритмов консольных приложений SCADA-системы "Соната".

В данном разделе будут описаны основные принципы работы в программе Отладчик и не будут описываться подробно все имеющиеся команды, так как их применение интуитивно понятно из названий и описаний кнопок и команд, приведённых ранее.

3.4.2.2.1. Открытие файла кода приложения

Для того чтобы открыть файл кода приложения, необходимо в разделе File главного меню программы Отладчик выбрать команду Open Ctrl+O. При этом откроетя диалоговое окно выбора файла (см. рис. 3.342).

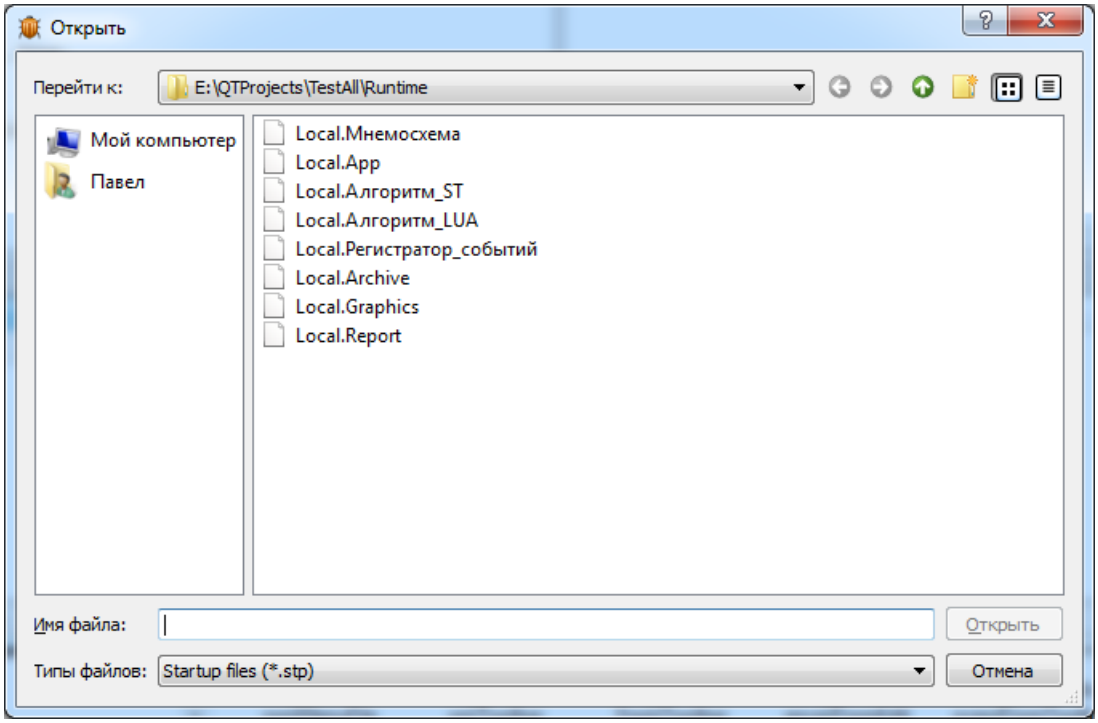
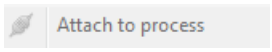


Рисунок 3.342 - Диалоговое окно выбора файла приложения

В диалоговом окне при помощи мыши или клавиатуры необходимо выбрать нужный файл приложения SCADA-системы "Соната" и нажать кнопку Открыть, при этом в окне отладочного кода отобразится код приложения. По нажатию на кнопку Отмена диалоговое окно просто закроется.

3.4.2.2.2. Подключение и отключение от работающего приложения

Для подключения к работающему приложению необходимо воспользоваться командой 

раздела Debug гавного меню или нажать кнопку  на панели инструментов. При этом откроется окно подключения к приложению (см. рис. 3.343).

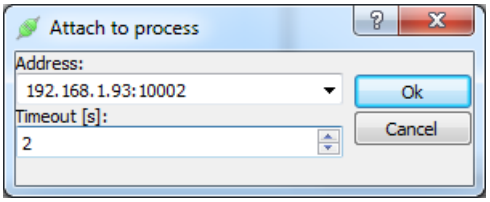
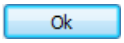


Рисунок 3.343 - Окно подключения к работающему приложению

Окно подключения состоит из следующих полей:

- **Addres** - ip-адрес и порт, по которым можно связаться с приложением (формат ip-адрес:порт);

- **Timeout [s]** - время подключения.

По нажатию на кнопку 

будет произведена попытка подключения к приложению, в случае удачного подключения в информационной строке общего окна программы Отладчик, которая находится снизу слева, будет выведена информация о подключении. После подключения вы получите возможность останавливать приложение, двигаться пошагово по приложению и ставить точки останова.

По нажатию на кнопку 

окно подключения закроется, и попытка подключения к приложению не будет произведена.

3.4.2.2.3. Остановка, пошаговая отладка и получение значений переменных

Для отладки приложения в первую очередь необходимо добавить переменные, значения которых используются в коде и дадут нам информацию о ходе программы. Для этого в окне **Watch** необходимо двойным щелчком «мыши» активировать свободную строку в столбце **Name** и ввести имя нужной переменной (см. рис. 3.344).

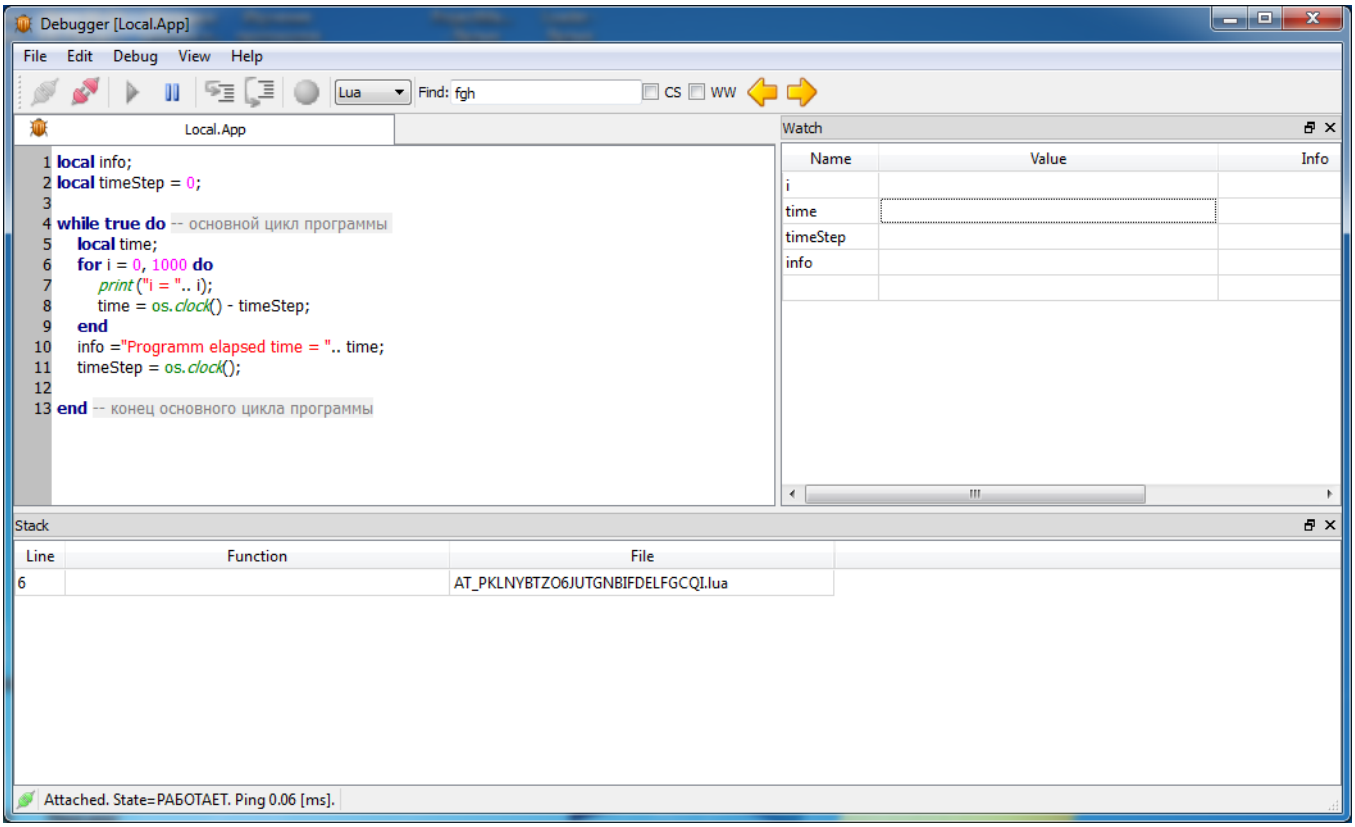


Рисунок 3.344 - Добавление переменных кода в окно просмотра Watch

На рис. 3.344 в окно просмотра **Watch** добавлены переменные **i** - шаг цикла for, **time** - время выполнения цикла программы, **timeStep** - общее время работы программы и **info** - строка.

Для остановки приложения можно воспользоваться кнопкой  панели инструментов. При этом приложение остановится, и в окне отладочного кода строка, где произошла остановка, будет выделена красным фоном (см. рис. 3.345). Также в окне просмотра **Watch** будут выведены значения переменных в момент остановки.

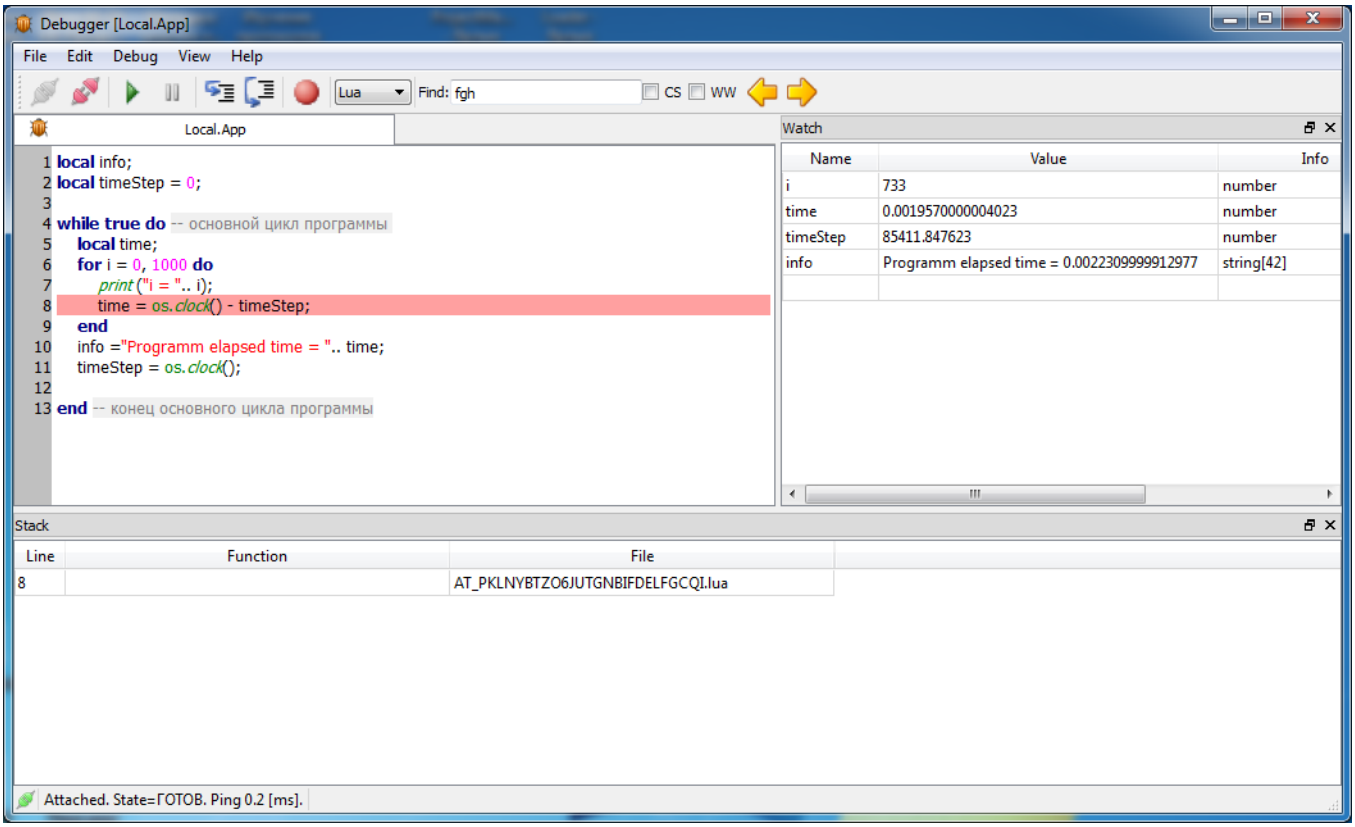


Рисунок 3.345 - Остановка приложения для отладки

Для пошаговой отладки используются кнопки  - шаги с входом внутрь цикла и  - шаги без входа внутрь цикла. По нажатию на данные кнопки будет выполнен шаг, в соответствии с выбранной кнопкой, выделенная строка в окне кода переместится в новое место и будут выведены значения переменных на данный момент.

Для того чтобы поставить точку останова, необходимо выделить нужную строку кода и нажать кнопку  на панели инструментов. При этом слева от строки появится  (см. рис. 3.346). Для удаления точки останова необходимо выбрать строку с уже имеющейся точкой останова и нажать кнопку  .

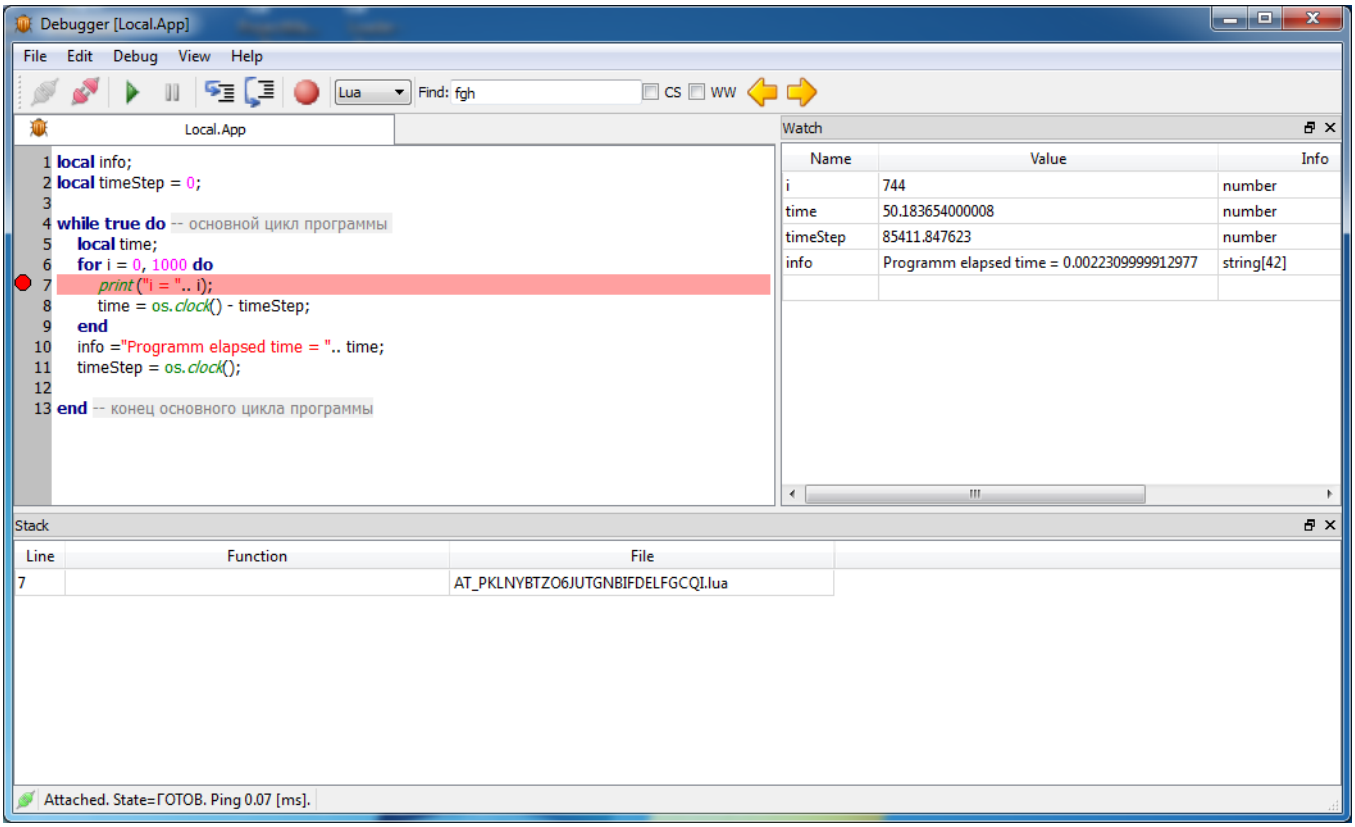


Рисунок 3.346 - Точка останова в окне отладочного кода

Работающее приложение всегда будет автоматически останавливаться на точках останова и, чтобы возобновить работу приложения, необходимо нажать кнопку  на панели инструментов.

3.4.3. Программа-отладчик IEC-приложений (IECDebugger)

Функциональность отладчика IEC-программ комплекса «Соната» (IECDebugger) предполагает два режима отладки IEC-приложений. Режим локальной отладки предназначен для автономной отладки отдельно взятого приложения. Режим удаленной отладки предназначен для отладки IEC-приложения, работающего среди других приложений запущенного проекта.

3.4.3.1. Интерфейс отладчика

3.4.3.1.1. Главное окно приложения

Главное окно содержит следующие элементы:
- главное меню;
- панель инструментов;
- окно дерева объектов;
- окно отладки композитного типа ФБ;
- окно сообщений;

- окно переменных/событий IEC;
- окно событий «Соната»;
- окно точек останова.

Главное окно отладчика приведено на рис. 3.347

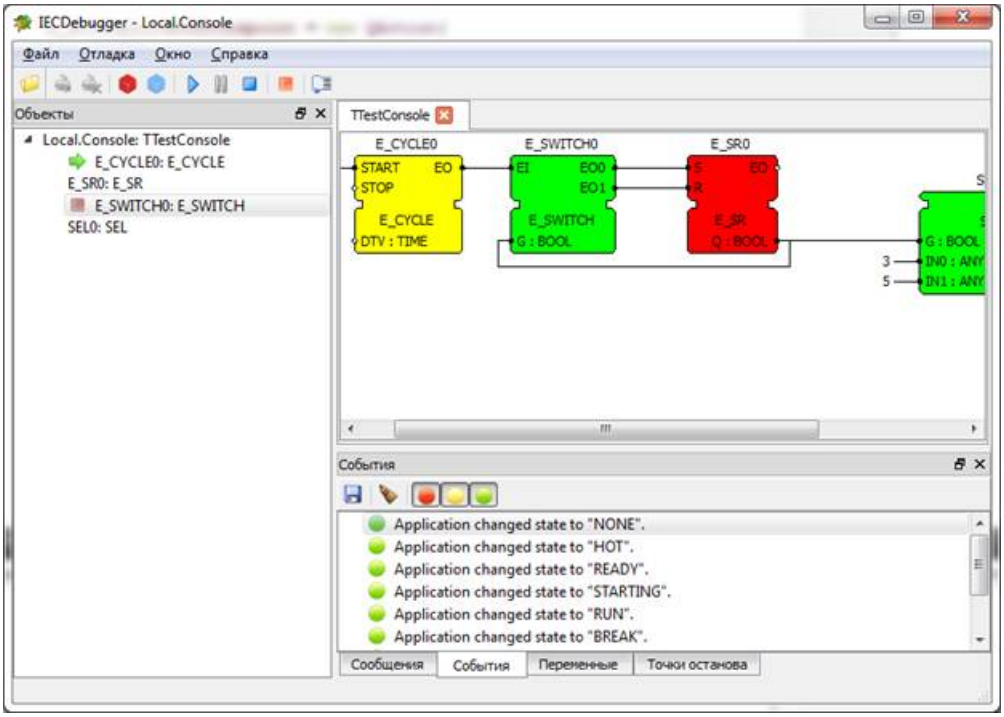


Рисунок 3.347 - Главное окно отладчика

Окна дерева объектов, сообщений, переменных/событий IEC, событий Sonata, точек останова представляют собой dock-окна. Для управления их видимостью предназначена группа главного меню «**Окно**».

Поле, содержащее окна отладки композитных блоков, представляет собой многостраничный элемент управления. Имя закладки совпадает с именем типа функционального блока.

3.4.3.1.2. Главное меню и панель инструментов

Главное меню и панель инструментов расположены в верхней части окна отладчика. Структура данных элементов приведена в представленной далее таблице.

Пункт меню	**Кнопка**	**Описание**
Файл		Группа команд работы с файлами
Открыть		Открыть stp-файл
Закрыть		Закрыть stp-файл
Выход		Завершить работу отладчика
Отладка		Группа команд отладки
Подключиться		Подключиться для отладки уже запущенного приложения

Пункт меню	**Кнопка**	**Описание**
Отключиться		Отключиться от отлаживаемого приложения
Запустить/возобновить		Запустить отладку либо возобновить работу приостановленного приложения
Прервать		Приостановить работу приложения
Установить/Убрать точку останова		Установить/Снять точку останова
Активировать точку останова		Активировать деактивированную точку останова
Деактивировать точку останова		Деактивировать активную точку останова
Открыть родительский тип		Открыть окно отладки типа данных родительского функционального блока
Открыть тип		Открыть окно отладки типа данных функционального блока
Шаг		Следующий шаг
Приложение		Группа команд управления приложением
Приложение/Горячий рестарт		Горячий рестарт приложения
Приложение/Холодный рестарт		Холодный рестарт приложения
Приложение/Остановка		Остановка приложения
Окно		Группа команд работы с окнами
Объекты		Показать/скрыть окно дерева объектов
События		Показать/скрыть окно событий Sonata
Переменные		Показать/скрыть окно переменных
Точки останова		Показать/скрыть окно точек останова
Сообщения		Показать/скрыть окно сообщений

3.4.3.1.3. Окно дерева объектов

Окно дерева объектов содержит иерархически организованный список объектов приложения. Элементом верхнего уровня является узел, соответствующий приложению. Для каждого узла в списке выводится его имя и тип (*имя: тип*). Дерево объектов содержит контекстное меню.

В окне дерева объектов пользователю доступны следующие функции:

- открыть окно отладки типа функционального блока (композитного или базового);
- открыть окно отладки родительского типа функционального блока;
- установить точку останова на конкретный объект.

Контекстное меню дерева объектов имеет следующую структуру:

- точка останова;
- открыть тип;
- открыть родительский тип.

Для открытия типа функционального блока необходимо выделить требуемый узел дерева, после чего нажать на правую кнопку «мыши» и выбрать опцию «**Отладка/Открыть тип**» главного меню либо опцию «**Открыть тип**» контекстного меню. Программа проверит, есть ли среди открытых указанный тип. Если тип есть, то программа переключается на страницу отладки указанного типа. Если такого типа нет среди открытых, то программа создает страницу отладки данного типа и переключает управление на нее.

Для открытия родительского типа функционального блока необходимо выделить требуемый узел дерева, после чего нажать на правую кнопку «мыши» и выбрать опцию «**Отладка/Открыть родительский тип**» главного меню либо опцию «**Открыть родительский тип**». Дальнейшие действия программы аналогичны действиям при открытии своего типа функционального блока.

Для установки точки останова на данном блоке необходимо выделить требуемый узел дерева, после чего нажать на правую кнопку «мыши» и выбрать опцию «**Отладка/Точка останова**» главного меню, либо «**Точка останова**» контекстного меню, либо нажать на кнопку « » панели инструментов. Программа изменит список точек останова следующим образом: если на данном блоке точка останова не была установлена, то программа установит ее; если точка была установлена ранее, то она будет снята.

Если происходит приостановка отладки либо остановка при достижении точки останова, то элемент дерева, на котором произошла остановка, помечается иконкой « ».

3.4.3.1.4. Окно сообщений

Окно сообщений содержит список сообщений и панель инструментов списка. Список содержит следующие колонки:

- иконку (« » для ошибок, « » для предупреждений, « » для простых сообщений);
- текст.

Панель инструментов содержит следующие элементы управления:

- кнопку очистки списка (« »);
- кнопку сохранения списка в файл (« »);
- кнопки фильтрации по типу сообщения (включение/отключение ошибок, предупреждений, простых сообщений).

При копировании, а также сохранении в текстовый файл список сообщений преобразуется в текст следующим образом:

- порядок следования сообщений соответствует порядку их следования в окне;
- формат строки сообщения: *Номер. [ТИП] Текст сообщения.*

3.4.3.1.5. Окно событий «Соната»

Окно событий содержит список событий и панель инструментов списка. Список содержит следующие колонки:

- иконку (« » для ошибок, « » для предупреждений, « » для простых сообщений);
- текст.

Панель инструментов содержит следующие элементы управления:

- кнопку очистки списка (« »);
- кнопку сохранения списка в файл (« »);
- кнопки фильтрации по типу события (включение/отключение ошибок, предупреждений, простых сообщений).

При копировании, а также сохранении в текстовый файл список событий преобразуется в текст следующим образом:

- порядок следования событий соответствует порядку их следования в окне;
- формат строки события: *Номер. [ТИП] Текст события.*

3.4.3.1.6. Окно точек останова

Окно точек останова содержит список точек останова и контекстное меню.

Информация, выводимая в список, не может быть отредактирована и зависит от того, где именно установлена точка останова. Подробная информация приведена далее в таблице .

Место установки	**Информация в поле адреса**
Объект в дереве	*Имя_приложения.объект.объект.....объект*
Объект в типе ФБ	*Имя_типа.объект*

Адрес точки представляет собой имя функционального блока (с именем всех его родительских блоков), либо имя типа и имя блока, если точка останова установлена в типе блока.

Точка останова может быть временно отключена.

Контекстное меню таблицы точек останова имеет следующие опции:

- активировать (ранее отключенную точку останова);
- деактивировать (отключить активную точку останова);
- удалить.

3.4.3.1.7. Окно переменных

Окно переменных содержит таблицу, в которой находятся переменные, значения которых отслеживает пользователь. Окно переменных содержит таблицу переменных с колонками имени и значения, а также панель инструментов, содержащую кнопки добавления («»), удаления («») переменных и изменения порядка их следования («» и «»).

Добавление переменных возможно в любой момент времени работы отладчика, однако просмотр и изменение их значения возможно только во время приостановки приложения путем собственно приостановки, точки останова или в ходе пошаговой отладки.

3.4.3.1.8. Окно отладки композитного блока

Окно отладки композитного блока предназначено для пошаговой отладки функционального блока, написанного на языке FBD. Окно содержит в себе поле отладки диаграммы.

Поле отладки диаграммы содержит набор объектов, связей событий и переменных.

В окне отладки композитного блока пользователь может установить точку останова на объект.

3.4.3.2. Функции отладчика

3.4.3.2.1. Подготовка к отладке приложения

Для подготовки к отладке приложения необходимо запустить отладчик (IECDebugger), после чего открыть stp-файл (см. раздел 3.4.3.2.2).

3.4.3.2.2. Открытие файла

Для открытия файла пользователь должен выбрать пункт главного меню «**Файл/Открыть**», либо нажать на кнопку «  » панели инструментов.

Если файл загружен и запущена отладка, то программа выведет сообщение «Запущена отладка. Прервать?», с кнопками «Да», «Нет». Если пользователь выберет «Да», то процесс отладки будет прерван. Если пользователь ответит «Нет», то программа прекратит процесс остановки отладки и загрузки нового stp-файла.

Затем, если stp-файл загружен, то программа его выгрузит.

Программа выведет диалог выбора файла. Пользователь должен выбрать stp-файл и нажать на кнопку «OK». Отладчик загрузит выбранный stp-файл и выведет окна дерева объектов (в левой части), а также окна сообщений, переменных/событий IEC, событий «Соната», точек останова в нижней части окна отладчика.

3.4.3.2.3. Закрытие файла

Для закрытия файла пользователь должен выбрать пункт главного меню «**Файл/Закрыть**».

Если запущена отладка, то программа выведет сообщение «*Запущена отладка. Прервать?*», с кнопками «*Да*», «*Нет*». Если пользователь выберет «*Да*», то процесс отладки будет прерван. Если пользователь ответит «*Нет*», то программа прекратит процесс остановки отладки и закрытия stp-файла.

Затем программа выгрузит stp-файл и уберет окна отладчика, после чего пользователь будет видеть только главное окно.

3.4.3.2.4. Завершение работы отладчика

Для завершения работы отладчика пользователь должен выбрать пункт главного меню «**Файл/Выход**», либо нажать на кнопку закрытия главного окна отладчика.

Если запущена отладка, то программа выведет сообщение «*Запущена отладка. Прервать?*», с кнопками «*Да*», «*Нет*». Если пользователь выберет «*Да*», то процесс отладки будет прерван. Если пользователь ответит «*Нет*», то программа прекратит процесс завершения работы отладчика.

3.4.3.2.5. Запуск процесса локальной отладки

Для запуска локальной отладки приложения необходимо выбрать пункт главного меню «**Отладка/Приложение/Горячий старт**» или пункт главного меню «**Отладка/Приложение/Холодный старт**», либо нажать на кнопки «  » или «  » панели инструментов соответственно.

Программа проверит возможность запуска приложения, после чего запустит его, если это возможно. Если запуск приложения невозможен, то программа выведет пользователю сообщение об ошибке в диалоговом окне, а список ошибок – в окне сообщений. Перед выводом диалогового окна с сообщением об ошибке программа сделает видимым окно списка сообщений.

Для запуска приложения используется IP-адрес 127.0.0.1:10001.

3.4.3.2.6. Остановка локальной отладки

Остановка отладки возможна только в том случае, когда процесс отладки был запущен локально. Если пользователь не запускал приложение, то остановка процесса отладки запрещена на уровне интерфейса.

Для остановки отладки пользователь должен выбрать пункт главного меню «**Отладка/Приложение/Остановить**» либо нажать на кнопку «  » панели инструментов. Программа остановит запущенную отладку.

3.4.3.2.7. Приостановка работы приложения

Приостановка процесса отладки возможна, если отладка была запущена локально, либо отладчик был подключен к работающему приложению. Если приостановка невозможна, то данная операция запрещена на уровне интерфейса.

Для остановки отладки пользователь должен выбрать пункт главного меню «**Отладка/Пауза**», либо нажать на кнопку «  » панели инструментов. Программа приостановит запущенное приложение.

3.4.3.2.8. Возобновление отладки

Если приложение приостановлено, то для возобновления его работы необходимо выбрать пункт главного меню «**Отладка/Запустить**», либо нажать на кнопку «  » панели инструментов. Данная функция доступна как в режиме локальной, так и в режиме удаленной отладки.

3.4.3.2.9. Подключение к работающему приложению

Подключение возможно, если ранее не было осуществлено подключение к работающему приложению. Если подключение невозможно, то операция запрещена на уровне интерфейса.

Для подключения к работающему приложению пользователь должен выбрать пункт главного меню «**Отладка/Подключиться**», либо нажать на кнопку «  » панели инструментов. Программа выведет диалог подключения, в котором пользователь задает IP-адрес, порт и время ожидания подключения. В качестве IP-адреса и порта используются данные из загруженного stp-файла, а в качестве таймаута предлагается последнее введенное значение. По умолчанию предлагается значение 2 сек. Пользователь задает требуемые значения, после чего нажимает на кнопку «**Подключиться**». Если пользователь нажимает на кнопку «**Отмена**», то попытка подключения не происходит.

Отладчик пытается установить соединение с приложением. Если в течение указанного времени попытка не удается, отладчик выводит сообщение «*Подключение невозможно. Истекло время ожидания*». Если подключение успешно установилось, то отладчик каждую секунду проверяет активность отлаживаемого приложение. Если связь с приложением потеряна, то отладчик автоматически отключается.

3.4.3.2.10. Отключение от работающего приложения

Отключение возможно, если запущенный пользователь ранее подключился для отладки к работающему приложению. Если подключения не было, то данная операция запрещена на уровне интерфейса.

Для отключения от работающего приложения пользователь должен выбрать пункт главного меню «**Отладка/Отключиться**» либо нажать на кнопку «  » панели инструментов. Программа прекратит процесс отладки и отключится от работающего приложения.

3.4.3.2.11. Точки останова

Точки останова служат для приостановки выполнения программы. Точка останова может быть установлена как перед отладкой, так и во время нее. Для работы с точками останова служит окно точек останова.

Точка останова может быть установлена на каком-либо объекте в дереве объектов или на каком-либо объекте в типе. Точка останова может быть безусловной, а также срабатывать при наступлении некоторого условия. Примером условия может быть выражение, факт изменения переменной или факт возникновения события.

3.4.3.2.11.1. Установка точки останова на объекте

Для установки точки останова необходимо выбрать объект в дереве, после чего выбрать пункт главного меню «**Отладка/Точка останова**», либо нажать на кнопку «  » панели инструментов. Программа установит безусловную точку останова и добавит соответствующую строку в список

точек. Выполняемая программа будет приостанавливаться каждый раз непосредственно перед вычислением данного объекта.

Объект, на котором установлена точка останова, будет помечен иконкой «  ».

При срабатывании точки останова программа откроет окно отладки родительского типа функционального блока и покажет блок, на котором произошла остановка.

3.4.3.2.11.2. Установка точки в композитном типе функционального блока

Для установки точки останова в типе функционального блока необходимо открыть диаграмму родительского типа, выделить объект и выбрать пункт главного меню «**Отладка/Точка останова**», либо нажать на кнопку «  » панели инструментов. Программа установит безусловную точку останова, которая будет срабатывать каждый раз, когда будет пересчитываться объект типа, где установлена точка останова. Также программа добавит соответствующую строку в список точек.

Объект, на котором установлена точка, помечается иконкой точки останова «  ». Объект, на котором сработала точка останова, помечается красным цветом фона.

3.4.3.2.12. Деактивирование точки останова

Для деактивирования точки останова необходимо выделить требуемую точку останова в окне точек останова, либо объект в дереве, либо объект в окне отладки композитного функционального блока, после чего выбрать пункт главного меню «**Отладка/Деактивировать точку останова**», либо выбрать пункт «**Деактивировать**» контекстного меню списка точек останова. Программа активирует точку останова и пометит ее иконкой «  ».

3.4.3.2.13. Активирование точки останова

Для активирования точки останова необходимо выделить требуемую точку останова в окне точек останова, либо объект в дереве, либо объект в окне отладки композитного функционального блока, после чего выбрать пункт главного меню «**Отладка/Активировать точку останова**», либо выбрать пункт «**Активировать**» контекстного меню списка точек останова. Программа активирует точку останова и пометит ее иконкой «  ».

3.4.3.2.14. Удаление точки останова

Точка останова может быть удалена из дерева объектов, окон отладки базовых и композитных блоков, а также списка точек останова.

Для удаления точки останова из дерева объектов, окон отладки базовых и композитных блоков необходимо выполнить действия, аналогичные действиям при установке точки. Там, где точка была установлена, она будет снята.

Для удаления точки останова необходимо выделить удаляемую строку в списке точек останова, после чего нажать на кнопку «**Del**» на клавиатуре, либо нажать на правую кнопку мыши и выбрать пункт «**Удалить**» открывшегося контекстного меню. Программа удалит выбранную точку останова.

3.4.3.2.15. Пошаговая отладка приложения

Пошаговая отладка приложения возможна в начале выполнения программы, ее приостановке соответствующей операцией или после ее остановки в точке останова.

Для осуществления пошагового выполнения программы пользователь должен выбрать пункт главного меню «**Шаг**», либо нажать на кнопку «  » панели инструментов. Программа выполнит следующий шаг. Если среди окон диаграмм композитных блоков находится тип, в который входит выполняемый объект, то отладчик активизирует страницу типа и подсветит объект, на котором произошла остановка. Также текущий выполняемый объект будет показан в дереве объектов.

3.4.3.2.16. Добавление переменной

Для добавления переменной необходимо нажать на кнопку «  » панели инструментов окна переменных. Редактор выведет окно добавления переменных, содержащее древовидный список, в который выведены переменные и объекты, к которым принадлежат эти переменные. На верхнем уровне списка помимо объектов верхнего уровня расположены интерфейсные переменные приложения. Пользователь должен выделить одну или несколько переменных, после чего нажать на кнопку «**Выбрать**». Редактор добавит выбранные переменные в список и закроет окно добавления переменных.

Альтернативным способом добавления переменной является ввод имени переменной в строку, следующую за строкой последней переменной в списке.

3.4.3.2.17. Изменение имени переменной

Для редактирования значения необходимо выделить строку переменной, после чего открыть редактор двойным щелчком «мыши» в колонке имени, либо нажатием на кнопку «**F2**». После завершения редактирования значения переменной необходимо нажать на кнопку «**Enter**».

3.4.3.2.18. Изменение значения переменной

Для редактирования значения необходимо выделить строку переменной, после чего открыть редактор двойным щелчком «мыши» в колонке значения, либо нажатием на кнопку «**F2**». После завершения редактирования значения переменной необходимо нажать на кнопку «**Enter**». Программа откроет редактор, соответствующий типу данных переменной.

3.4.3.2.19. Удаление переменной

Для удаления переменной необходимо выделить ее строку в списке, после чего нажать на кнопку «» панели инструментов окна редактирования переменных. Программа удалит выбранную переменную из списка.

3.4.3.2.20. Изменение порядка следования переменных

Для перемещения переменной вверх необходимо выделить строку переменной, после чего нажать на кнопку «  » панели инструментов окна редактирования переменных. Программа переместит выбранную переменную на одну строку вверх.

Для перемещения переменной вниз необходимо выделить строку переменной, после чего нажать на кнопку «  » панели инструментов окна редактирования переменных. Программа переместит выбранную переменную на одну строку вниз.

3.4.3.2.21. Управление приложением

Функции управления приложением позволяют пользователю выполнить холодный и горячий рестарт приложения, а также остановить его выполнение.

3.4.3.2.21.1. Холодный рестарт приложения

Выполнить холодный рестарт возможно, если запущена отладка приложения, либо было произведено подключение к удаленному приложению.

Для выполнения холодного рестарта пользователю необходимо выбрать пункт главного меню «**Отладка/Приложение/Холодный рестарт**» либо нажать на кнопку «  » панели инструментов. Программа выполнит указанную команду.

3.4.3.2.21.2. Горячий рестарт приложения

Выполнить горячий рестарт возможно, если запущена отладка приложения, либо было произведено подключение к удаленному приложению.

Для выполнения горячего рестарта пользователю необходимо выбрать пункт главного меню «**Отладка/Приложение/Горячий рестарт**» либо нажать на кнопку «  » панели инструментов. Программа выполнит указанную команду.

3.4.3.2.21.3. Остановка выполнения приложения

Выполнить остановку возможно, если запущена отладка приложения, либо было произведено подключение к удаленному приложению.

Для выполнения остановки пользователю необходимо выбрать пункт главного меню «**Отладка/Приложение/Остановить**» либо нажать на кнопку «» панели инструментов. Программа выполнит указанную команду. При этом отладка будет прекращена и произойдет отключение отладчика от приложения.

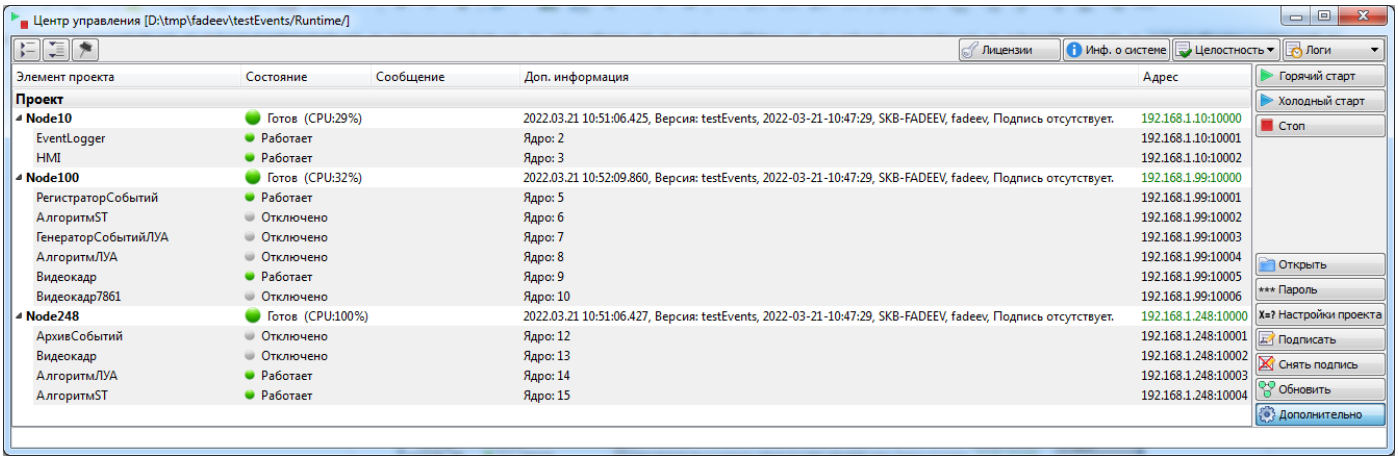
3.4.4. Программа управления "ControlCenter" или Центр управления

По нажатию на кнопку в верхней части главного окна **ProjectManager** (Менеджер проектов) будет запущена программа управления - **ControlCenter** (Центр управления).

Также **ControlCenter** (Центр управления) можно запускать из командной строки. При запуске программы из командной строки первым аргументом является абсолютный путь к папке с конфигурационными файлами проекта. К примеру, для Windows команда будет выглядеть так - ControlCenter.exe D:\Bin\, если файлы проекта располагаются в папке D:\Bin\.

Программа **ControlCenter** (Центр управления) считывает конфигурационные файлы и определяет структуру системы. После этого она начинает циклический опрос элементов системы: узлов (компьютеры и главные контроллеры) и выполняемых на них программ (драйвера, технологические алгоритмы, контроль уставок и т.п.)

Диалоговое окно программы представлено на рис. 3.348. В окне отображается древовидный список системы и элементы управления ей.



Элемент проекта	Состояние	Сообщение	Доп. информация	Адрес
Проект				
Node10	Готов (CPU:29%)		2022.03.21 10:51:06.425, Версия: testEvents, 2022-03-21-10:47:29, SKB-FADEEV, fadeev, Подпись отсутствует.	192.168.1.10:10000
EventLogger	Работает		Ядро: 2	192.168.1.10:10001
HMI	Работает		Ядро: 3	192.168.1.10:10002
Node100	Готов (CPU:32%)		2022.03.21 10:52:09.860, Версия: testEvents, 2022-03-21-10:47:29, SKB-FADEEV, fadeev, Подпись отсутствует.	192.168.1.99:10000
РегистраторСобытий	Работает		Ядро: 5	192.168.1.99:10001
АлгоритмST	Отключено		Ядро: 6	192.168.1.99:10002
ГенераторСобытийЛУА	Отключено		Ядро: 7	192.168.1.99:10003
АлгоритмЛУА	Отключено		Ядро: 8	192.168.1.99:10004
Видеокадр	Работает		Ядро: 9	192.168.1.99:10005
Видеокадр7861	Отключено		Ядро: 10	192.168.1.99:10006
Node248	Готов (CPU:100%)		2022.03.21 10:51:06.427, Версия: testEvents, 2022-03-21-10:47:29, SKB-FADEEV, fadeev, Подпись отсутствует.	192.168.1.248:10000
АрхивСобытий	Отключено		Ядро: 12	192.168.1.248:10001
Видеокадр	Отключено		Ядро: 13	192.168.1.248:10002
АлгоритмЛУА	Работает		Ядро: 14	192.168.1.248:10003
АлгоритмST	Работает		Ядро: 15	192.168.1.248:10004

Рисунок 3.348 - Диалоговое окно программы управления SCADA-системой "СОНАТА"

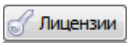
Слева в верхней части окна находится строка с именем приложения, а правее располагаются стандартные для оконных приложений кнопки управления окном , имеющие привычные функции свернуть окно, развернуть на весь экран и закрыть.

Ниже расположены следующие элементы:

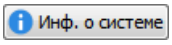
("Свернуть всё") - сворачивает дерево проекта;

("Развернуть всё") - разворачивает дерево проекта;

(""Поверх всех окон) - позволяет разместить или отменить резмещение окна приложения поверх всех других окон.

Для использования кнопки 

необходимо выбрать узел и после этого нажать данную кнопку. Откроется окно с информацией об оборудовании узла и с информацией о состоянии лицензии данного узла. Данная кнопка может помочь разобраться с причиной почему может не подходить лицензия к узлу.

Используя кнопку 

можно получить информацию об отдельном узле (нужно выбрать узел и нажать данную кнопку) или обо всех узлах проекта (для этого нужно выделить корневой узел с именем Проект и нажать данную кнопку). Информация содержит конфигурацию узла и операционной системы на узле (см. рис. 3.349).

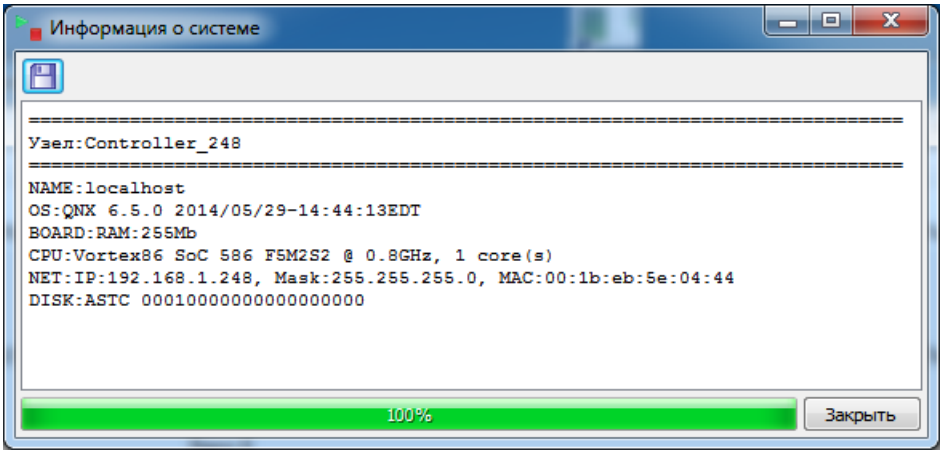
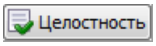
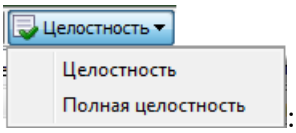


Рисунок 3.349 - Окно вывода информации об узле

По нажатию на кнопку 

она развернётся в меню из команд 

:

- **Целостность** - выведет только информацию о состоянии среды исполнения (ОК или Изменён). Если были изменены какие-либо файлы, то будут перечислены их имена (см. рис. 3.350 и рис. 3.351);

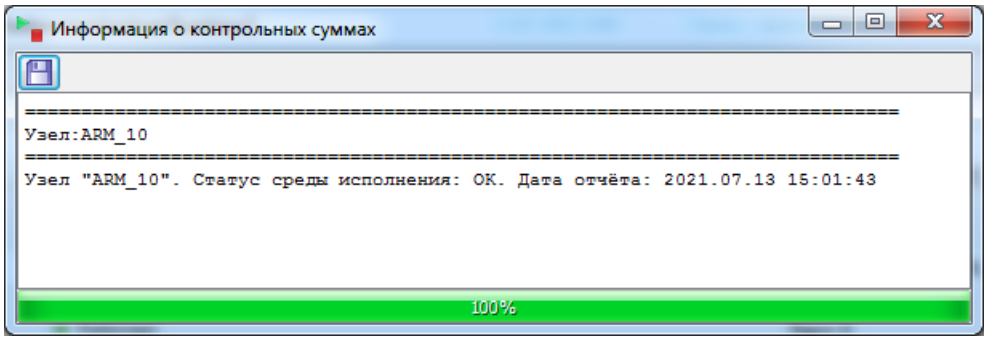


Рисунок 3.350 - Положительная краткая информация о целостности узла

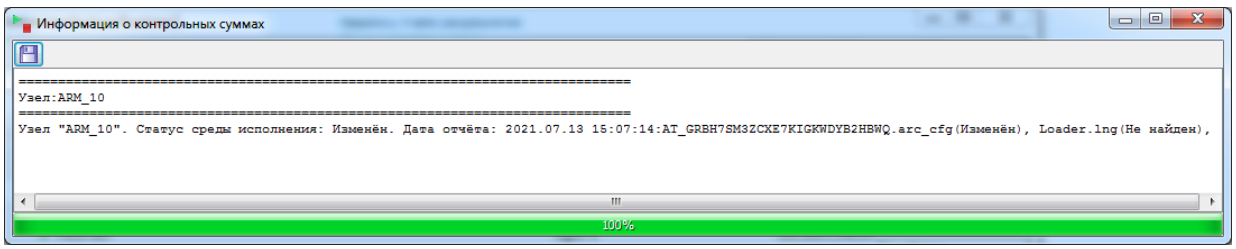


Рисунок 3.351 - Краткая информация о нарушении целостности узла

- **Полная целостность** - выведет информацию о состоянии среды исполнения (ОК или Изменён) и отобразит файлы с их контрольными суммами, которые были подписаны цифровой подписью (есть возможность настроить список файлов, которые подписываются цифровой подписью, описано в Дополнительных настройках узла в приложении Менеджер проета).

Можно получить информацию о целостности всего проекта (для этого нужно выделить корневой узел с именем Проект и выбрать один из пунктов меню) или отдельного узла в проекте (для этого нужно выделить отдельный узел и выбрать один из пунктов меню).

ВНИМАНИЕ! Данный функционал работает при наличии цифровых подписей на файлы проекта (см. ниже описание кнопки Дополнительно - Подписать).

По нажатию на кнопку 

она развернётся в меню из команд :

- Логи - информация о работе проекта после последнего запуска. Можно посмотреть данные по работе узла в целом (для этого нужно выделить узел) или по работе отдельного приложения (для этого нужно выделить отдельное приложение).;

- Полные логи - выводит информацию обо всех узлах. Данный лог отображает информацию о текущем запуске и, если есть, то о предыдущих запусках проекта (глубина хранения 5 предыдущих запусков).

Основная часть рабочего окна состоит из следующих столбцов:

- **Элемент проекта** - элементы, из которых состоит проект (узлы и их приложения);

- **Состояние** - в каком состоянии находится узел или приложение (см. далее);

- **Сообщение** - текущие сообщения о состоянии узлов и приложений;

- **Доп. информация** - дополнительная информация об узлах и приложениях;

- **Адрес** - ip-адрес:порт узлов и приложений.

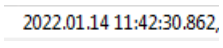
Столбец **Элемент проекта** представляет собой древовидный список, состоящий из узлов и работающих на них приложений.

Столбец **Состояние** отображает состояние узлов системы и выполняемых на них программ. Зеленый значок состояния  сообщает о том, что узел или программа функционируют должным образом. Жёлтый значок состояния  сообщает, что возникла какая-либо проблема или узел и приложение еще в процессе загрузки. Красный значок состояния  сообщает о том, что нет связи с элементом системы. Серый значок состояния  сообщает о том, что приложение добавлено в конфигурацию узла, но ему не выставлена галочка **Вкл.**, т.е. приложение не будет стартовать на узле. Синий значок состояния  сообщает, что приложение находится в резерве. Значок – синяя стрелка  сообщает о том, что происходит обработка узлом команды управления (запуск/останов). Так же в столбце **Состояние** выводится информация о загрузке центрального процессора (CPU) узла.

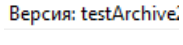
В столбце **Сообщение** выводятся сообщения о текущем состоянии узлов и приложений, в том числе выводятся сообщения об ошибках.

В столбце **Доп. информация** выводится различная дополнительная информация об узлах и приложениях.

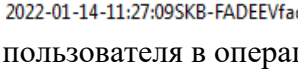
К **Доп. информации** относятся:

- 

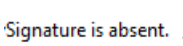
- дата и время на узле;

- 

- название проекта;

- 

- дата и время запуска проекта, имя АРМа в операционной системе и имя пользователя в операционной системе;

- 

- информация об отсутствии подписи на проект на данном узле;

- 

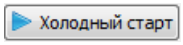
- у каждого приложения указан номер его микроядра.

В столбце **Адрес** отображается ip-адрес оборудования, на котором работает узел с приложениями, и отображаются значения коммуникационных портов каждого узла и приложения.

В правой части окна программы расположены следующие элементы управления:

- кнопка 

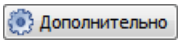
предназначена для запуска всей системы или отдельных её узлов в режиме "горячего старта", когда обычные переменные инициализируются начальными значениями, а переменные **RETAIN** и **PERSISTANCE** сохраняют свои последние значения;

- кнопка 

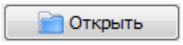
предназначена для запуска всей системы или отдельных её узлов в режиме "холодного старта", когда обычные переменные и переменные **RETAIN** инициализируются начальными значениями, а переменные **PERSISTANCE** сохраняют свои последние значения. Важный момент в том, что, если перезапустить один узел уже работающего проекта в режиме "холодного старта", а другие не перезапускать, то обычные переменные и переменные RETAIN инициализируются начальными значениями и сразу же снова получат свои последние значения до перезапуска, прочитав их с работающих узлов, что приводит к потере смысла перезапуска. Поэтому, чтобы выполнить "Холодный старт", нужно остановить проект полностью и потом запустить его кнопкой "Холодный старт";

- кнопка 

предназначена для останова всей системы или отдельных её узлов;

- по нажатию на кнопку 

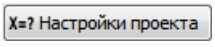
- развернётся меню с кнопками дополнительного функционала:

- кнопка 

("Открыть проект") - предназначена для открытия проекта, за которым ведется наблюдение при помощи программы **ControlCenter** (Центр управления), при этом будет открыт стандартный диалог операционной системы для выбора директории, в котором нужно выбрать директорию, где расположены конфигурационные файлы проекта;

- кнопка 

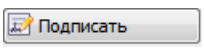
("Изм. пароль узлов") - если вы используете шифрования данных и ваше приложение ControlCenter не смогло подключиться к проекту (узлы и приложения работающего проекта не доступны), то с помощью данной кнопки вы сможете подключиться к узлу в целом (нужно выделить узел инажать кнопку) или к отдельному приложению (нужно выделить отдельное приложение и нажать кнопку) с паролем, который узел или приложение используют для шифрования данных;

- кнопка 

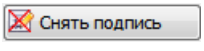
("Получить настройки проекта") - служит для получения основных настроек проекта. Эти настройки представляют собой таблицу со значениями всех переменных. В таблице одним столбцом указываются начальные значения переменных на момент старта. Если переменная имеет тип хранения PERSISTENT или RETAIN, то для них вторым столбцом указывается сохранённое значение, с отметкой времени в скобках;

- кнопка 

("Обновить среду исполнения") - данный функционал позволяет заменить/обновить один или несколько файлов проекта или дистрибутива SCADA "СОНАТА" на выбранных узлах;

- кнопка 

("Подписать узел") - данный функционал позволяет добавить цифровую подпись на файлы проекта. Подписать можно либо проект целиком (необходимо выделить корневой элемент с названием Проект), либо выбранные узлы (необходимо выделить нужные узлы);

- кнопка 

("Снять подпись с узла") - данный функционал позволяет снять цифровую подпись с файлов проекта. Снять цифровую подпись можно либо с проекта целиком (необходимо выделить корневой элемент с названием Проект), либо с выбранных узлов (необходимо выделить нужные узлы);

В нижней части окна выводятся различные диагностические сообщения. Данные сообщения могут быть очень полезными. К примеру сообщение типа "Версии проекта различаются на узлах"

встречается довольно часто в проектах с большим количеством узлов, если один из узлов забыли обновить (см. рис. 3.352).

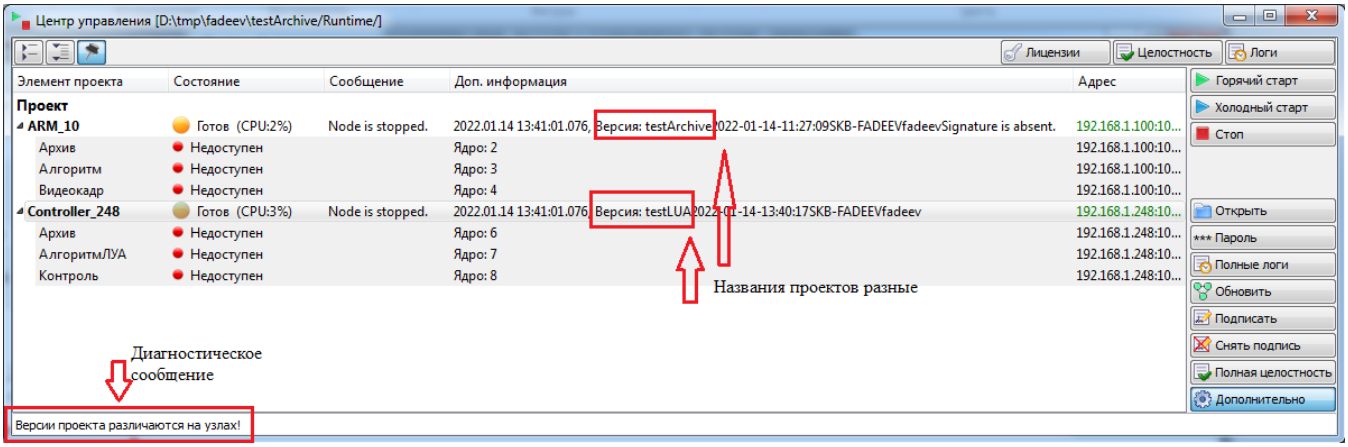


Рисунок 3.352 - Диагностическое сообщение о том, что на узлах разные проекты

Пользователь может подавать команды управления как всем узлам системы одновременно, так и каждому узлу индивидуально. Для подачи команды всем узлам системы необходимо в древовидном списке выбрать корневой элемент **Проект** и подать команду. Для подачи индивидуальных команд узлам необходимо, удерживая клавишу "Ctrl", выбрать в древовидном списке нужные позиции и подать команду. Отдельным приложениям нельзя подавать управляющие команды, можно только в целом узлу, на котором работает приложение.

ВНИМАНИЕ! Распространение проекта запрещено при стартующем или останавливающимся узле. При попытке распространения Distributor (приложение, распространяющее проект) выдаст ошибку "Доступ запрещён (403)". Возможна ситуация, когда из-за какой-либо ошибки в приложениях, в ходе старта узла, он не запустился до конца. Это состояние также считается состоянием старта узла, при котором распространение проекта запрещено. Для выхода из этой ситуации нужно из центра управления подать команду на останов проекта.

ВНИМАНИЕ! При изменении в проекте: структурных типов, интерфейсов приложений, порядка приложений на узлах, перечня узлов и перечня глобальных сигналов требуется перезапуск всего проекта. Недопустимо перезапускать только часть узлов. Часть узлов допустимо перезапускать при работающих остальных узлах, если были изменены только: тексты программ, конфигурации драйверов, конфигурация архива, конфигурация мнемосхемы.

3.4.5. Диагностическая программа для просмотра значений сигналов "SignalViewer" - Просмотрщик сигналов

Данная диагностическая программа используется при пуско-наладочных работах. Она позволяет посмотреть и изменить значения переменных (сигналов) любой программы системы.

Вызов осуществляется нажатием на кнопку верхней панели окна "**Менеджера проектов**"  - "**Просмотрщик сигналов**".

3.4.5.1. Описание главного окна программы "Просмотрщик сигналов"

Главное окно программы "**Просмотрщик сигналов**" представлено на рис. 3.353.

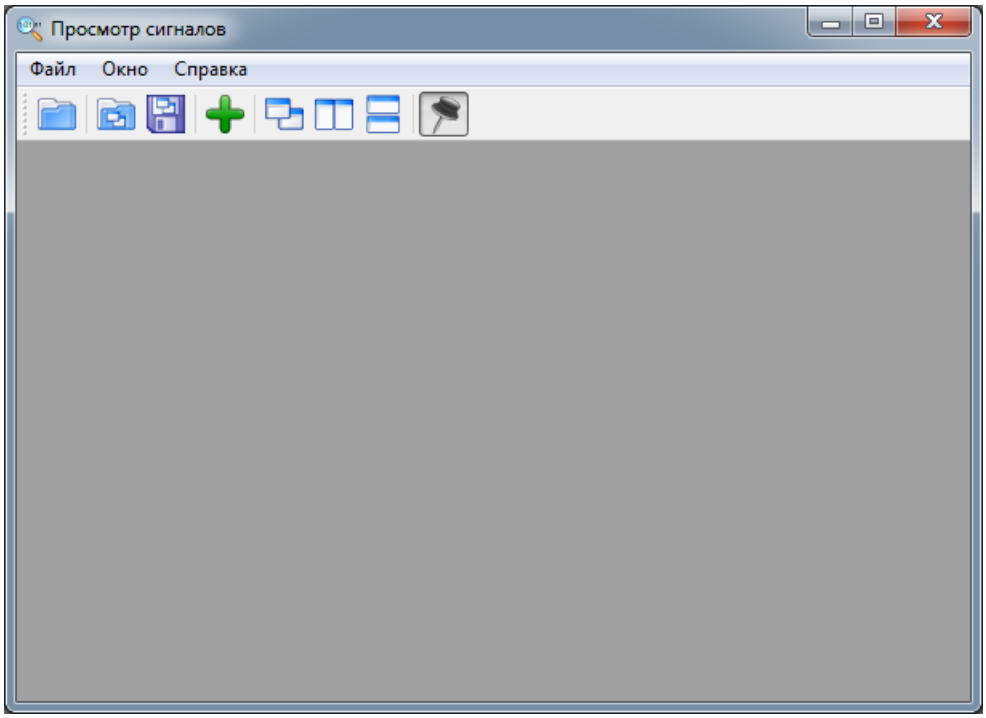


Рисунок 3.353 - Главное окно программы "Просмотрщик сигналов"

В верхней части окна расположена панель инструментов с кнопками.

Кнопка  панели инструментов главного окна предназначена для открытия проекта и считывания информации о приложениях.

Кнопка  панели инструментов главного окна предназначена для открытия файла, сохраненного ранее списка сигналов, окна "**Просмотрщика сигналов**".

Внимание! Сохранённый список сигналов можно будет корректно открыть, если с момента сохранения в проекте изменялись только алгоритмы или настройки драйверов. Сохранённый список сигналов не получится корректно открыть, если с момента сохранения в проекте были следующие изменения:

- был изменён список глобальных сигналов проекта;
- был изменён интерфейс приложений проекта;
- был изменён порядок запуска приложений в настройках узла.

Кнопка  панели инструментов главного окна предназначена для сохранения списка сигналов окна в файл.

Кнопка  панели инструментов главного окна предназначена для создания нового (дочернего) окна для просмотра сигналов (таких окон можно создавать много).

Кнопка панели инструментов главного окна предназначена для закрепления главного окна программы **"Просмотрщик сигналов"** поверх остальных окон (не работает в операционной системе QNX).

Кнопка  панели инструментов главного окна предназначена для расположения дочерних MDI окон каскадом.

Кнопка  панели инструментов главного окна предназначена для расположения дочерних MDI рядом друг с другом горизонтально.

Кнопка  панели инструментов главного окна предназначена для расположения дочерних MDI рядом друг с другом вертикально.

Кнопка  панели инструментов главного окна предназначена для закрепления главного окна программы **"Просмотрщик сигналов"** поверх остальных окон (не работает в операционной системе QNX).

3.4.5.2. Описание дочернего окна программы "Просмотрщик сигналов"

По нажатию на кнопку  главного окна программы "Просмотрщик сигналов" создается дочернее окно (см. рис. 3.354). Данное окно предназначено для отображения значения сигналов.

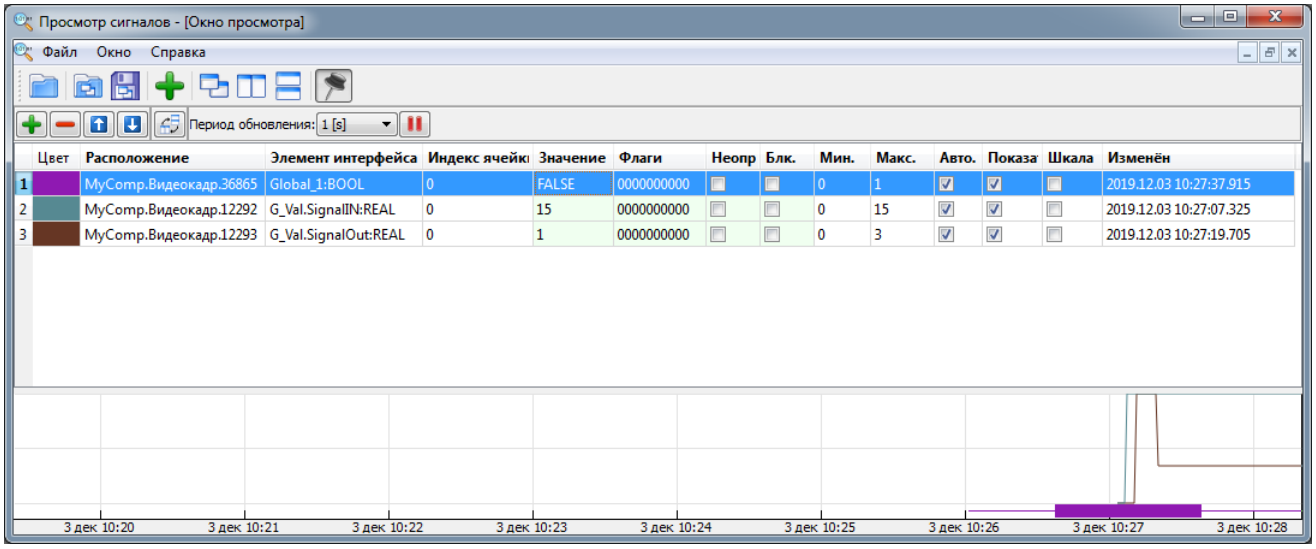


Рисунок 3.354 - Дочернее MDI окно для просмотра значений сигналов

В верхней части окна расположена панель инструментов с кнопками.

Кнопка  предназначена для добавления в таблицу нового сигнала для просмотра. При этом будет открыт диалог выбора сигнала из списка доступных сигналов сторонней программы.

Кнопка  предназначена для удаления выделенной позиции из таблицы просмотра.

Кнопка  предназначена для перемещения вверх выделенной позиции таблицы просмотра.

Кнопка  предназначена для перемещения вниз выделенной позиции таблицы просмотра.

Кнопка  предназначена для переключения размещения таблицы и графиков из горизонтального положения в вертикальное и обратно.

- данное меню предназначено для управления периодом обновления значений сигналов и выводом их на графиках (нижняя дочернея часть окна).

Под панелью инструментов располагается таблица со списком сигналов. У таблицы следующие столбцы:

- **Цвет** – цвет графика сигнала;
- **Расположение** – место нахождения сигнала;
- **Элемент интерфейса** – имя сигнала;
- **Индекс ячейки** – индекс ячейки сигнала (если сигнал – массив);
- **Значение** – значение сигнала;

- **Флаги** – пользовательские флаги, которые могут назначаться и использоваться по его усмотрению (на флаги выделено 16 бит, а для пользователя доступны первые 10 бит; смотри документ SCADA-система "СОНАТА" Описание применения КУНИ.505200.023-01.01 31);
- **Неопр.** (неопределенное или недостоверное значение) – данная галочка указывает, что значение сигнала достоверно или нет;
- **Блк.** - запрет на изменение сигнала другими приложениями;
- **Мин.** – минимальное значение сигнала;
- **Макс.** – максимальное значение сигнала;
- **Авто** – если данный флаг выставлен, то программа автоматически определяет минимальное и максимальное значение сигнала;
- **Показать** – если данный флаг выставлен, то значение данного сигнала также отображается в виде графика;
- **Шкала** - добавляет шкалу со значениями в область графического представления сигналов;
- **Изменён** - дата и время последнего изменения значения сигнала.

В нижней части дочернего окна отображается графическое представление сигналов.

3.4.5.3. Работа с дочерним окном программы "Просмотрщик сигналов"

Чтобы добавить нужный сигнал для просмотра и управления, нужно нажать на кнопку  верхней панели дочернего окна. При этом откроется окно "**Сигналы ядра**" (см. рис. 3.355).

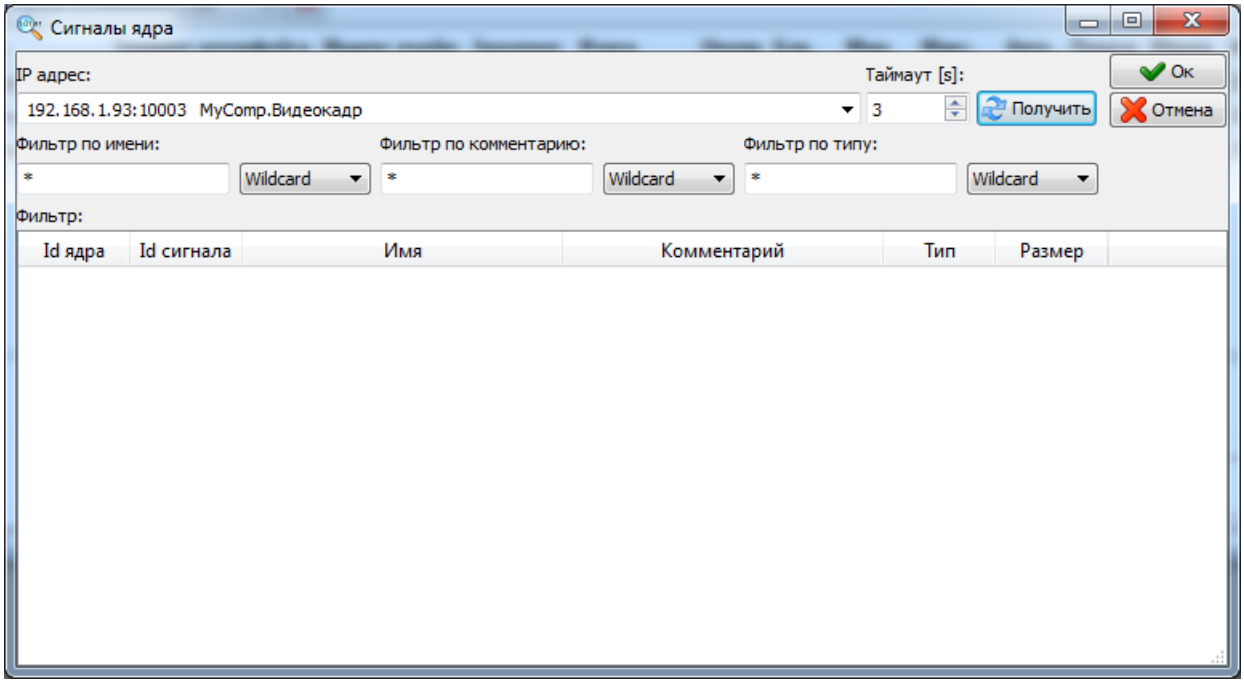
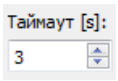


Рисунок 3.355 - Окно "Сигналы ядра"

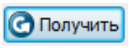
В верхней части окна на рис. 3.355 расположены следующие элементы управления.

Поле 

предназначено для выбора приложения, сигналы которого нужно смотреть.

Поле 

предназначено для указания тайм-аута запроса.

По нажатию на кнопку 

происходит подключение к указанному в поле **IP адрес** приложению и, в случае удачного подключения, ниже будут выведены все сигналы данного приложения (см. рис. 3.356).

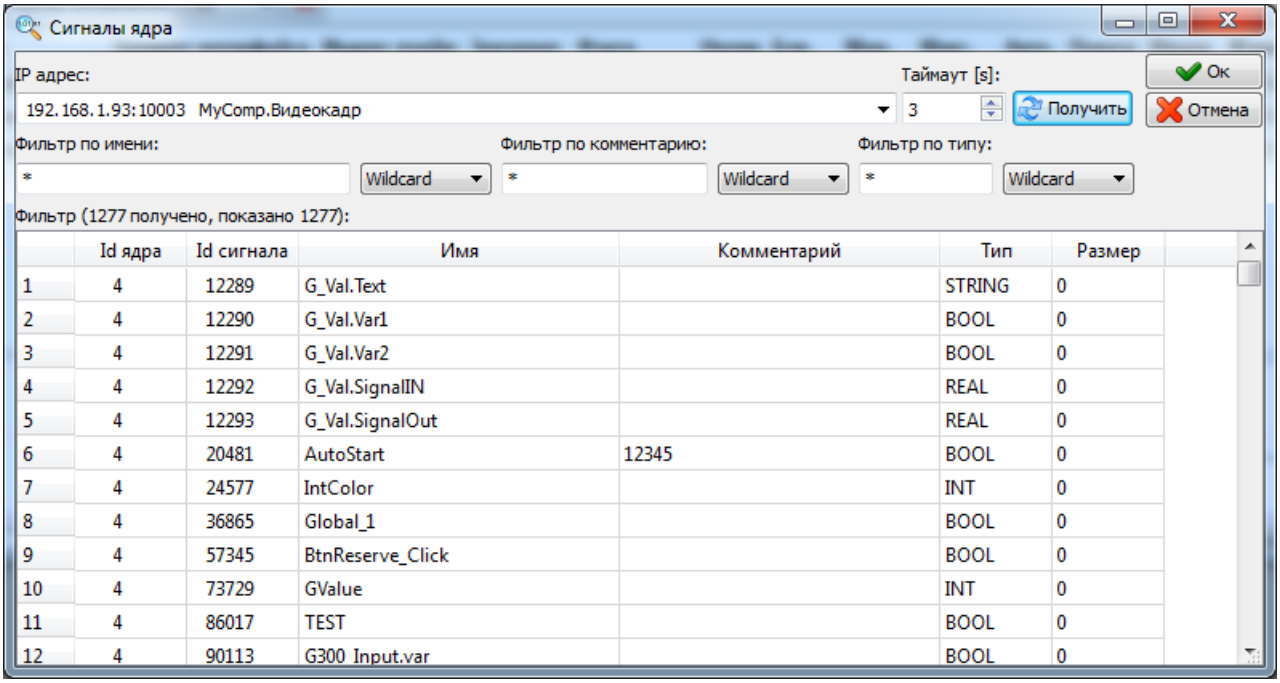
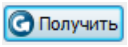
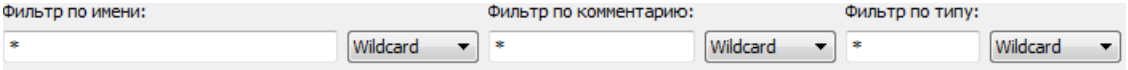
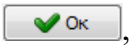


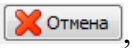
Рисунок 3.356 - Окно "Сигналы ядра", после нажатия на кнопку 

Поля 

предназначены для выборки нужных сигналов из общего списка по разным критериям фильтрации: по имени, по комментарию и по типу.

Нижняя часть окна "**Сигналы ядра**" представляет собой таблицу под названием **Фильтр**, именно в ней и отображаются сигналы приложения, к которому произведено подключение (см. рис. 3.356).

После выбора нужных сигналов необходимо нажать на кнопку 

, после чего сигналы отобразятся в дочернем окне программы "**Просмотрщик сигналов**" (см. рис. 3.357). Если нажать на кнопку 

, то окно "**Сигналы ядра**" закроется и все действия будут отменены.

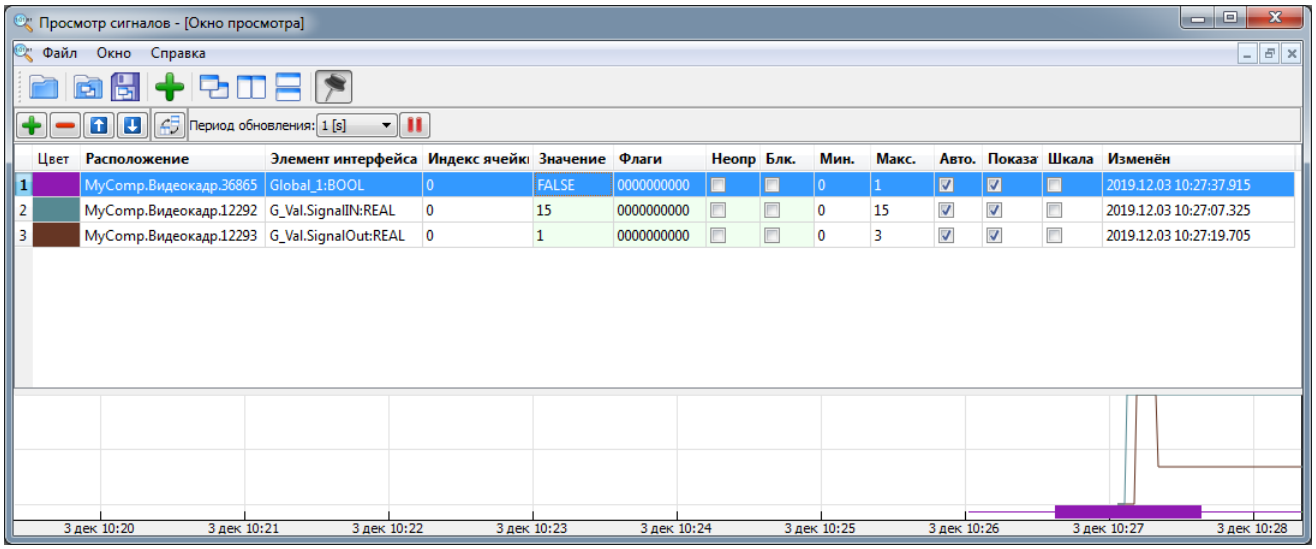


Рисунок 3.357 - Окно с добавленными сигналами

В окне на рис. 3.357 можно менять значения сигналов (в колонке **Значение**), следить за графическим отображением изменений, следить за временем изменений в колонке **Изменён** и производить много других действий, необходимых в процессе пуско-наладочных работ.

ВНИМАНИЕ!!! Если запустить приложение "**Просмотрщик сигналов**" с ключом -restricted, то будет заблокирована возможность управлять сигналами (т.е. изменять их значения и другие настройки). Можно будет только просматривать их значения.

3.4.6. Программа "ArchiveViewer" для просмотра архивных значений сигналов и событий SCADA-системы "Соната"

"ArchiveViewer" - диагностическая программа для синхронного просмотра архивных значений сигналов и событий системы (далее приложение **Просмотр архивов**).

Приложение **Просмотр архивов** можно запустить из главного окна программы **Менеджер проектов**. Для этого необходимо нажать правой кнопкой мыши на иконку  в верхней панели инструментов. Так же можно отдельно запустить приложение Просмотр архивов из дистрибутива. Для этого нужно в дистрибутиве запустить исполняемый файл **ArchiveViewer**.

Просмотр архивов может применяться для работы с запущенным проектом, на котором для просмотра архивных значений сигналов должно работать приложение **Archive** (далее **Архив сигналов**), а для просмотра событий в системе должно работать приложение **EventLogger** (далее **Регистратор событий**). Так же можно просматривать архивы сигналов и событий, сохраненные в файлы архивив.

3.4.6.1. Знакомство с главным окном программы

На рис. 3.358 представлено главное окно программы в котором происходит просмотр архивных значений сигналов и событий системы.

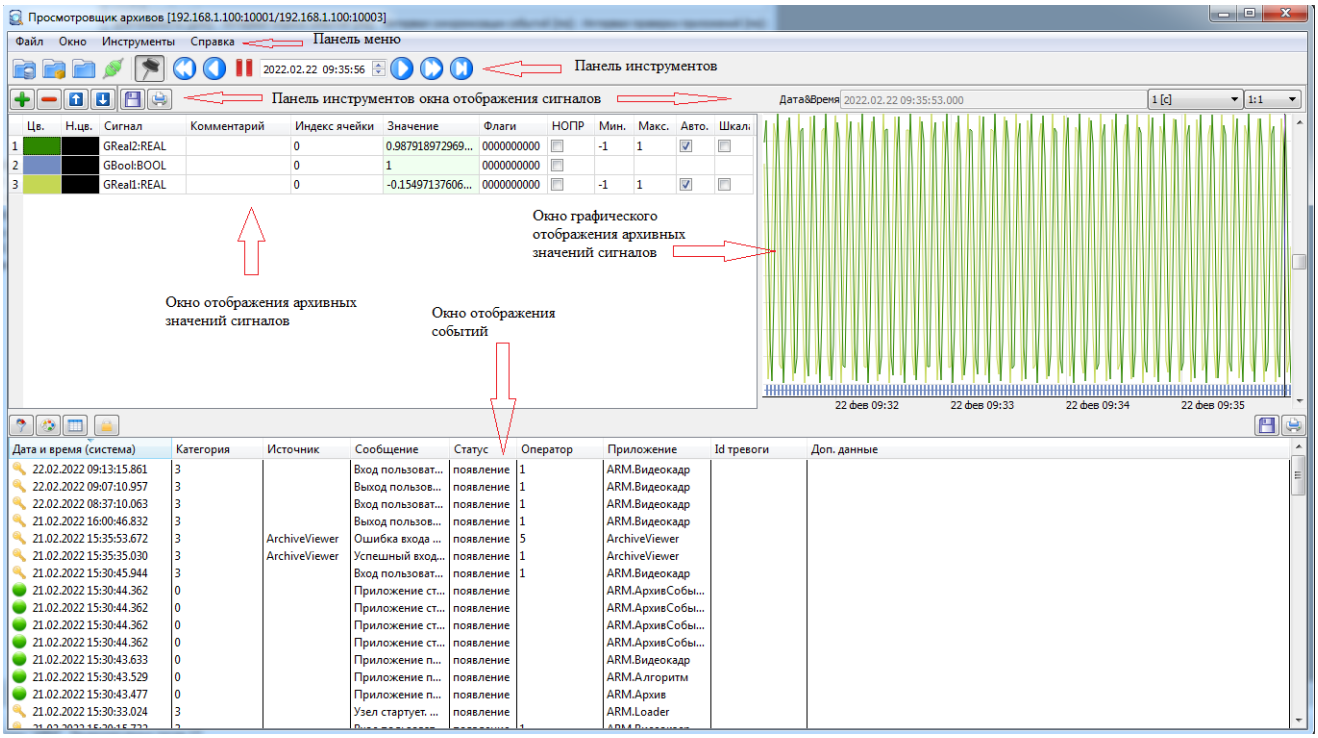
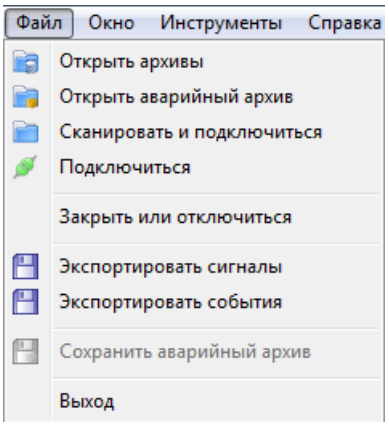


Рисунок 3.358 - Главное окно программы **Просмотр архивов**

Окно программы Просмотр архивов поделено на несколько областей (далее окон, см. рис. 3.358):

- панель меню;
- панель инструментов;
- окно отображения архивных значений сигналов;
- окно графического отображения архивных значений сигналов;
- окно отображения событий.

В верхней части окна расположено меню и главная панель инструментов со следующими визуальными компонентами:



- в меню Файл находятся следующие команды:

- Открыть архивы - по нажатию на данный пункт откроется окно файловой системы, в котором нужно выбрать папку с архивами сигналов или архивами событий и открыть их для просмотра. Можно просматривать архивы сигналов и событий неработающего проекта;

- Открыть аварийный архив - по нажатию на данный пункт откроется окно файловой системы, в котором нужно выбрать папку с аварийными архивами и открыть их для просмотра;

- Сканировать и подключиться - по нажатию на данный пункт откроется окно файловой системы, в котором нужно выбрать папку с работающим проектом, после чего можно будет подключиться для просмотра к архиву сигналов и архиву событий;

- Подключиться - по нажатию на данный пункт откроется окно подключения к архивам сигналов и событий (см. рис. 3.373);

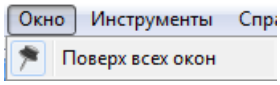
- Закрыть или отключиться - по нажатию на данный пункт приложение **Просмотр архивов** отключиться от архивов и очистит свои окна, отображающие данные;

- Экспортировать сигналы - сохраняет данные о сигналах, выбранных ранее для просмотра, в файл txt;

- Экспортировать события - сохраняет данные о событиях, выбранных ранее для просмотра, в файл csv;

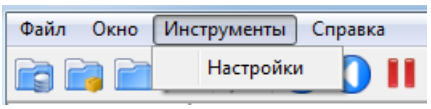
- Сохранить аварийный архив - сохраняет данные аварийного архива, выбранных ранее для просмотра, в файл aad;

- Выход - по нажатию на данный пункт будет закрыто приложение **Просмотр архивов**.



- в меню Окно находятся следующие команды:

- Поверх всех окон - когда данный пункт меню в активном состоянии, то окно приложения **Просмотр архивов** отображается поверх всех окон;



- в меню Инструменты находятся следующие команды:

- Настройки - по нажатию на данный пункт меню откроется окно настроек (см. рис. 3.359).

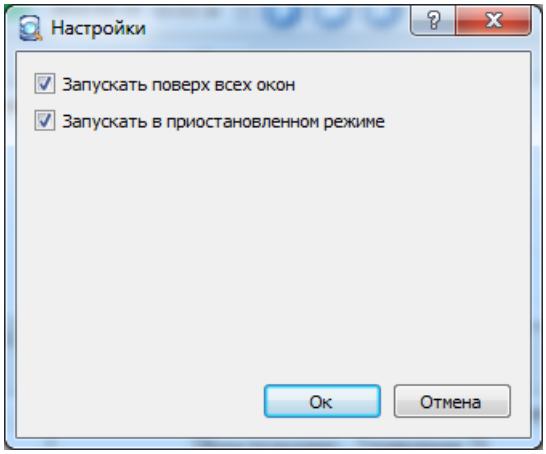
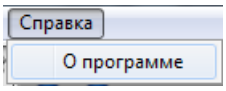


Рисунок 3.359 - Окно настроек приложения Просмотр архивов

В окне настроек располагаются две флаговые кнопки:

- Запускать поверх всех окон - при активном состоянии данного флага окно приложения **Просмотр архивов** запускается со свойством отображения поверх всех окон (значение данного флага хранится в конфигурационном файле, т.е. сохраняется после закрытия приложения);

- Запускать в приостановленном режиме - при активном состоянии данного флага после подключения к архивам, архив сигналов и событий отображаются в приостановленном режиме (значение данного флага хранится в конфигурационном файле, т.е. сохраняется после закрытия приложения).



- меню Справка, в котором находится кнопка "О программе", выводящая на экран информацию о текущей версии программы (см. рис. 3.360).

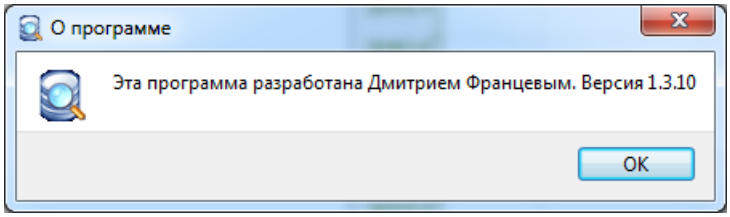


Рисунок 3.360 - Окно информации о версии программы

Ниже располагается панель инструментов, состоящая из следующих элементов:

 - по нажатию на данный пункт откроется окно файловой системы, в котором нужно выбрать папку с архивами сигналов или архивами событий и открыть их для просмотра. Можно просматривать архивы сигналов и событий неработающего проекта;

 - по нажатию на данный пункт откроется окно файловой системы, в котором нужно выбрать папку с аварийными архивами и открыть их для просмотра;

 - кнопка просканировать проект и подключиться к его архивам, позволяет указать Runtime папку работающего проекта и считать его конфигурацию, что далее позволит подключиться к проекту без ввода конфигурационной информации о приложениях **Архив сигналов** и **Регистратор событий**;

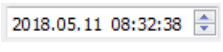
 - кнопка Подключиться вызывает диалог настройки подключения к **Архиву сигналов** и **Регистратору событий**;

 - кнопка **Поверх всех окон**;

 - кнопка **Быстро назад** позволяет перемещаться по оси времени назад с большим шагом (на один экран графического отображения назад);

 - кнопка **Назад** позволяет перемещаться по оси времени назад с шагом в одно деление шкалы времени;

 /  - кнопки позволяют остановить/запустить процесс вывода информации в программе **Просмотр архивов**;



- поле позволяет запрашивать архивные данные за конкретные дату и время;

 - кнопка **Вперёд** позволяет перемещаться по оси времени вперёд с шагом в одно деление шкалы времени;

 - кнопка **Быстро вперёд** позволяет перемещаться по оси времени вперёд с большим шагом (на один экран графического отображения вперёд);

 - кнопка **В начало** перемещает по оси времени в .

Окно отображения архивных значений сигналов (см. рис. 3.361) состоит из следующих элементов:

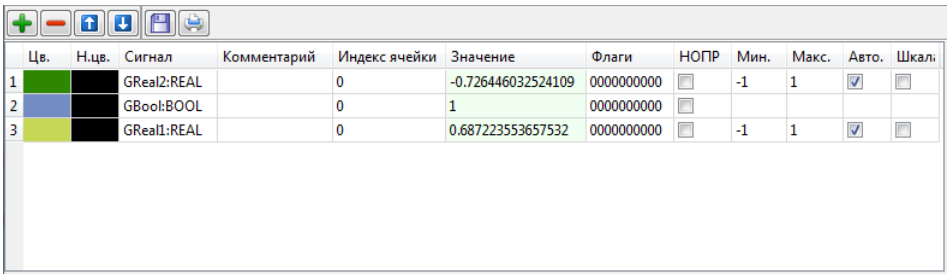


Рисунок 3.361 - Окно отображения архивных значений сигналов

- кнопка **Добавить сигнал** относится к окну отображения информации об архивных сигналах и вызывает меню добавления архивных сигналов проекта;

- кнопка **Удалить сигнал** относится к окну отображения информации об архивных сигналах и удаляет выделенные сигналы;

- перемещает выбранный сигнал вверх в окне отображения архивных значений сигналов;

- перемещает выбранный сигнал вниз в окне отображения архивных значений сигналов;

- сохранить значения выбранных архивных сигналов в файл txt;

- распечатать выбранные значения архивных сигналов.

Окно отображения архивных значений сигналов содержит следующие поля:

- в первом поле порядковый номер сигнала в окне;

- **Цв.** - цвет графика для отображения реальных (валидных) значений сигнала;

- **Н.цв.** - цвет графика для отображения невалидных значений сигнала;

- **Сигнал** - имя и тип сигнала;

- **Комментарий** - текстовый комментарий для сигнала;

- **Индекс ячейки** - индекс ячейки сигнала (если сигнал – массив);

- **Значение** - значение сигнала;

- **Флаги** - флаги сигнала;

- **НОПР** - признак валидности сигнала. Если данный чекбокс активен, то сигнал является невалидным;

- **Мин.** - минимальное значение сигнала;

- **Макс.** - максимальное значение сигнала;

- **Авто** - если данный флаг выставлен, то программа автоматически определяет минимальное и максимальное значение сигнала;

- **Шкала** - если данный флаг выставлен, то в окне графического отображения значение данного сигнала будет отображаться вертикальная шкала со значениями.

Окно графического отображения архивных значений сигналов (см. рис. 3.362) состоит из следующих элементов:

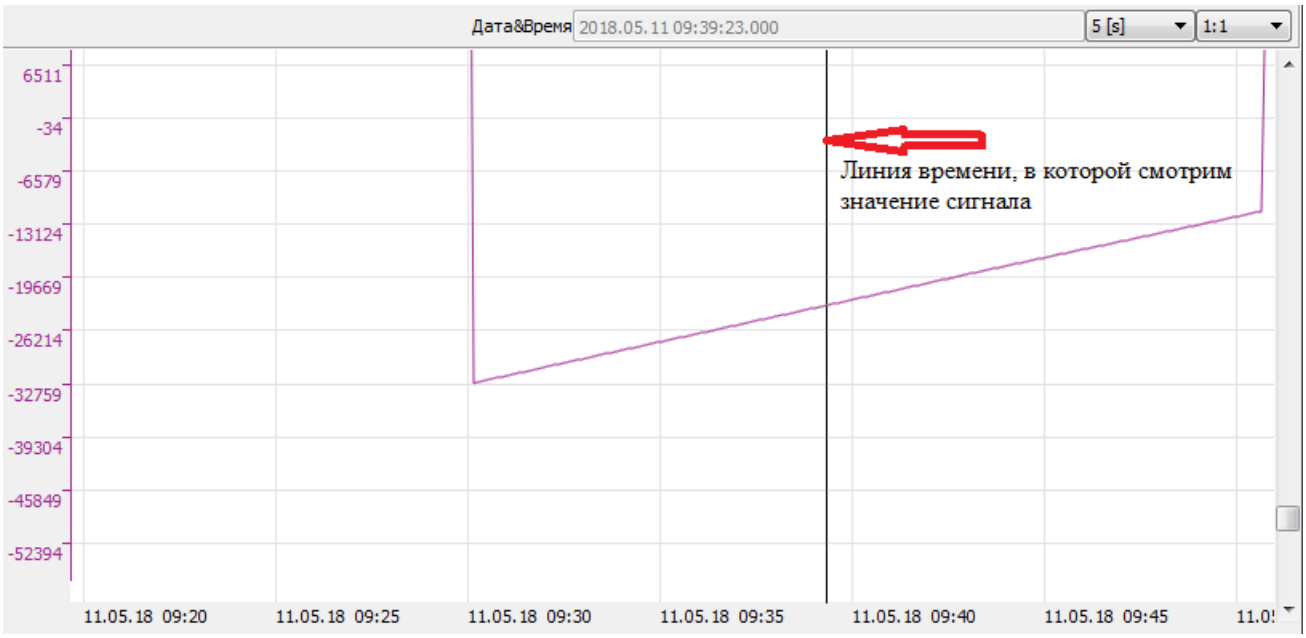
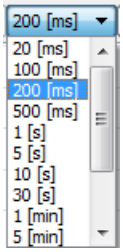
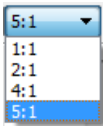


Рисунок 3.362 - Окно графического отображения архивных значений сигналов

- поле даты и времени, запрашиваемого из архива значения сигнала;



- интервал между точками данных;



- масштабирование вертикальной шкалы значений архивных сигналов.

В окне графического отображения значения сигнала можно смотреть значения в одном или двух моментах времени. Для этого с помощью левой клавиши мыши устанавливается точка времени, на графике она отображается как линия (см. рис. 3.362). По умолчанию данная точка времени является текущим моментом времени. Чтобы установить две такие линии, необходимо зажать левую клавишу мыши в перой точке и, не отпуская, переместить во вторую точку (см. рис. 3.363). При этом мы будем видеть значения сигналов в этих двух точках.

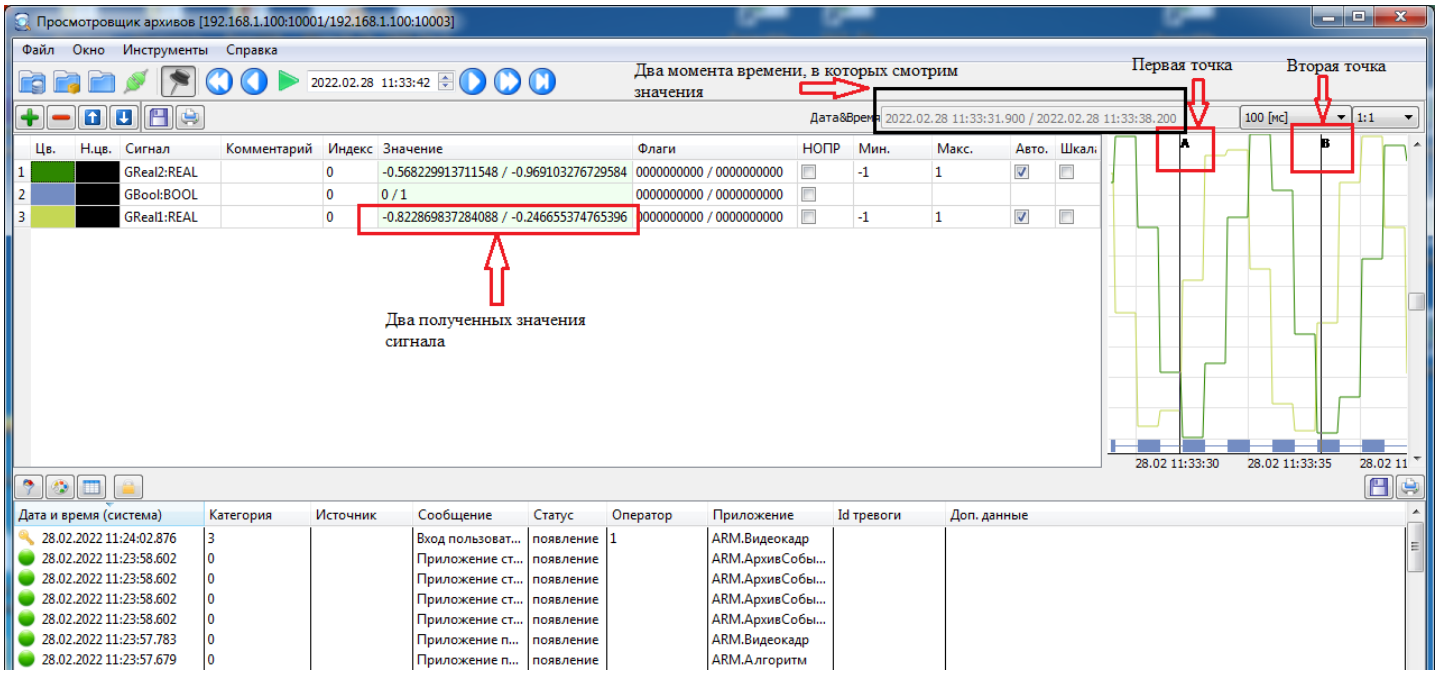


Рисунок 3.363 - Окно приложения Просмотр архивов,
на котором установлены две отсечки времени

Окно отображения событий (см. рис. 3.364) состоит из следующих элементов:

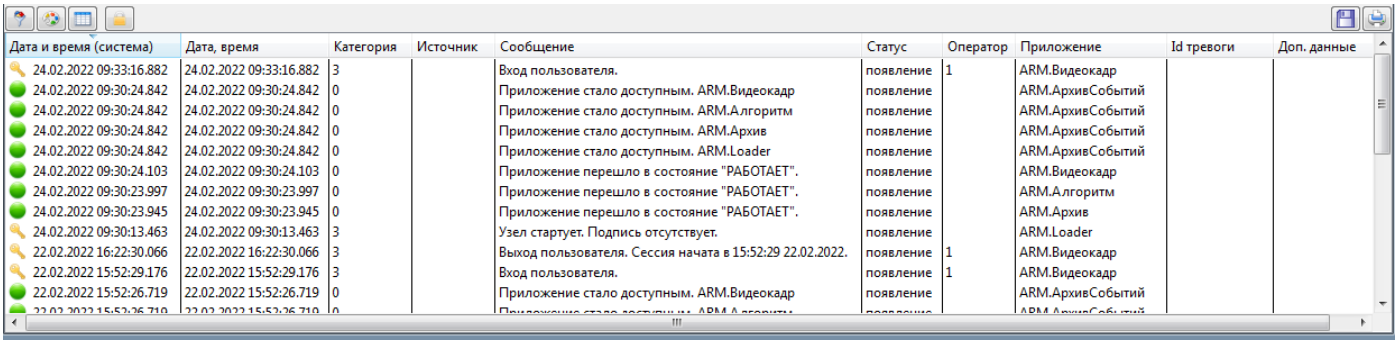


Рисунок 3.364 - Окно событий системы

В верхней части располагается панель инструментов со следующими кнопками:

 - кнопка настройки фильтров отображения событий (см. описание ниже);

 - кнопка настройки цветовой палитры отображения событий;

 - кнопка настройки отображаемых в таблице столбцов;

 - кнопка блокировки прокрутки списка событий;

 - сохранить список событий в файл txt;

 - распечатать список событий.

Окно отображения событий содержит следующие столбцы:

- **Дата и время (система)** - дата и время появления события;
- **Дата, время** - дата и время появления события;
- **Категория** - к какой категории относится событие;
- **Источник** - источник данного события;
- **Сообщение** - сообщение для данного события;

- **Статус** - статус состояния события. Может быть четыре варианта: появление, исчезновение, квитирование (подтверждение опреатора), пропадание (условие появления тревоги пропало, но тревога еще не квитирована);

- **Оператор** - оператор (пользователь), при котором возникло данное событие;

- **Приложение** - приложение, которое создало данное событие;

- **Id тревоги** - уникальный идентификатор пользовательского события или тревоги;

- **Доп. данные** - дополнительные данные, описывающие событие.

В крайней левой колонке для лучшего визуального восприятия все события по категориям отображаются с помощью следующих графических компонентов:

 - события информационной безопасности;

 - информационные события;

 - информационные события и предупреждения:

 - предупреждения;

 - предупреждения и критические события;

 - критические события.

3.4.6.2. Описание настроек фильтра отображаемых событий

По нажатию на кнопку  ("Настроить фильтр") откроется окно настроек фильтра (см. рис. 3.365).

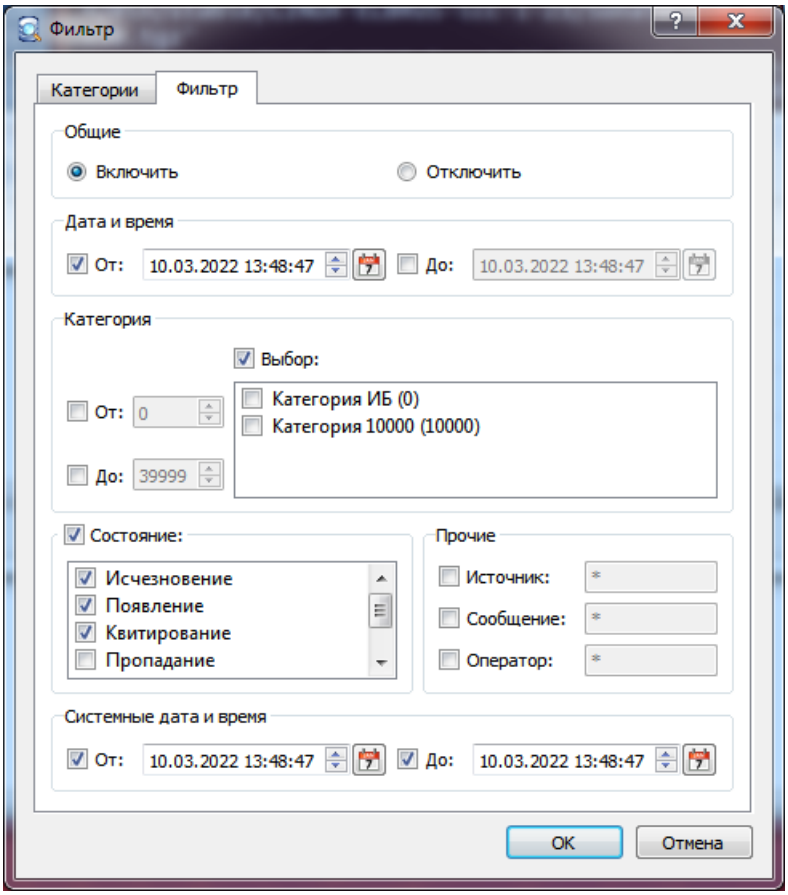


Рисунок 3.365 - Окно настроек фильтра событий

Окно настроек фильтра событий состоит из двух вкладок:

- вкладка **Фильтр** - в данной вкладке располагаются все настойки для фильтра;

- вкладка **Категории** - в данной вкладке создаются категории, по которым далее можно настроить фильтрацию во вкладке **Фильтр**.

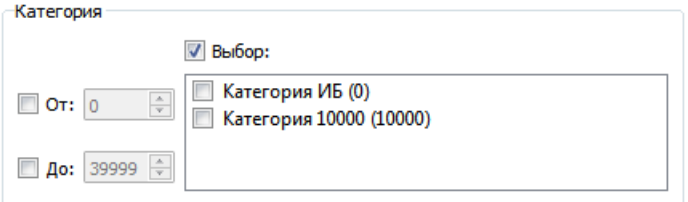
Во вкладке **Фильтр** располагаются следующие настройки:

- 

- включить/отключить применение всех выполненных настроек;

- 

- фильтрация по дате и времени;

- 

- фильтрация событий по категориям. Можно настроить фильтрацию в пределах интервала категорий. Это настройки от какой и до какой категории выводить события.

По настройке 

- можно указать конкретные категории, которые будут отображаться. Категории, которые можно выбрать в данном окне, создаются во вкладке **Категории** (см. рис. 3.366).

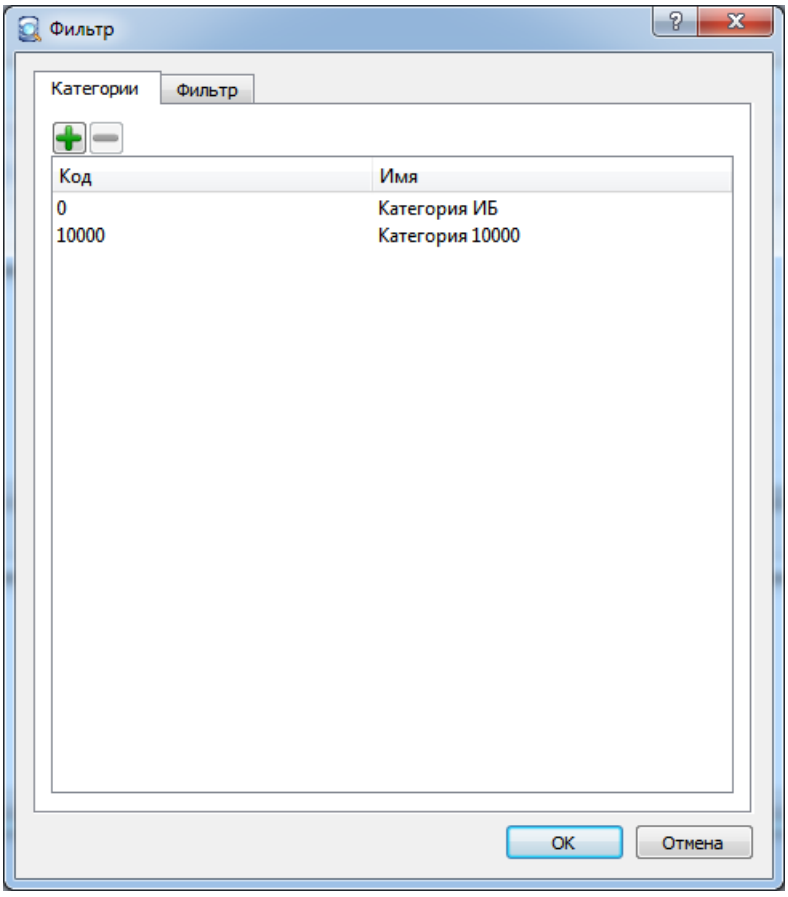
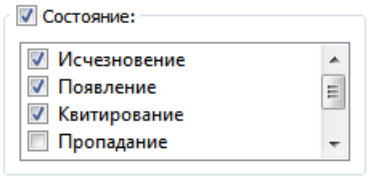
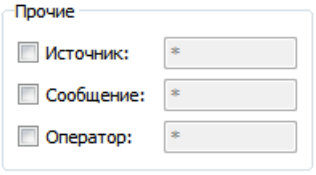


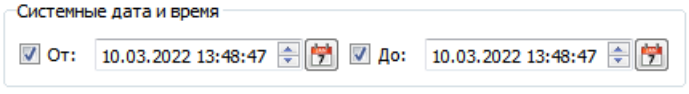
Рисунок 3.366 - Вкладка Категории

- 

- фильтрация событий по состояниям;

- 

- фильтрация событий по содержимому в колонках Источник, Сообщение и Оператор;

- 

- фильтрация по системной дате и времени.

3.4.6.3. Настройка цветовой палитры отображения событий

По нажатию на кнопку  ("Настроить палитру") откроется окно настроек цветовой палитры отображения событий (см. рис. 3.367).

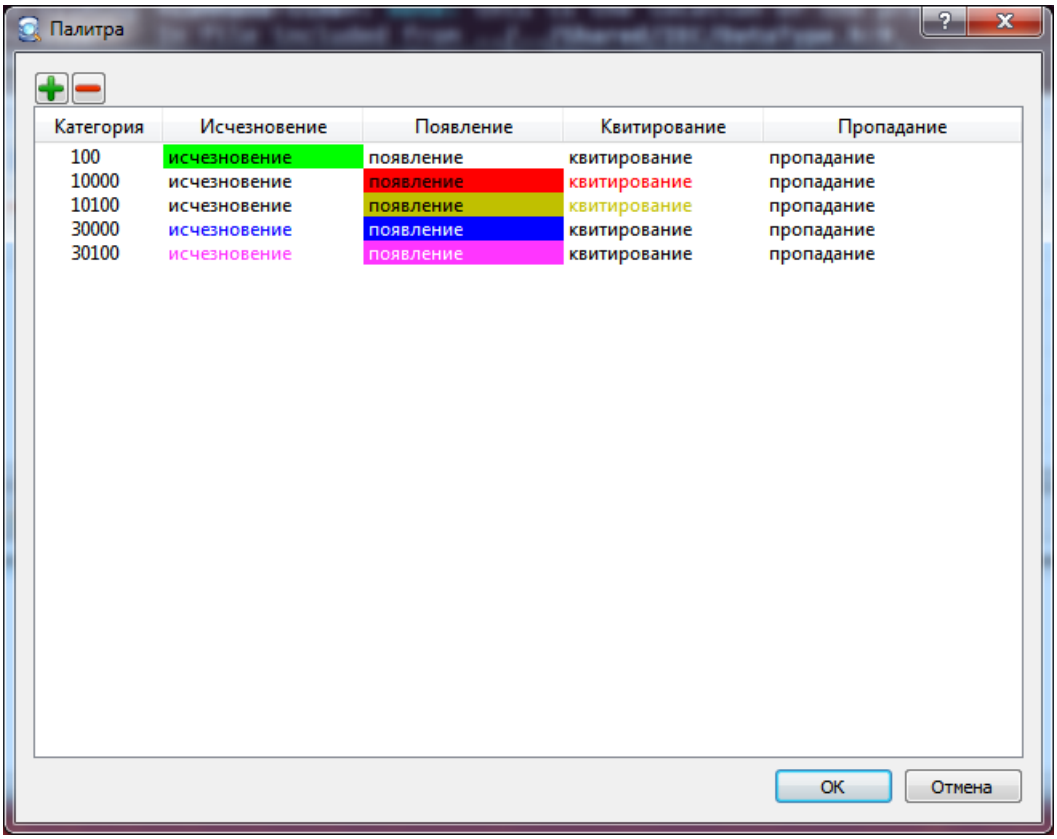


Рисунок 3.367 - Окно настроек цветовой палитры отображения событий

В верхней части окна располагаются следующие кнопки:

-  - добавить категорию событий для настроек палитры;

-  - удалить категорию событий.

Основную часть окна занимает таблица, состоящая из следующих колонок:

- **Категория** - настраиваемая категория событий;
- **Исчезновение** - настройки палитры для событий в состоянии исчезла;
- **Появление** - настройки палитры для событий в состоянии появилась;
- **Квитирование** - настройки палитры для событий в состоянии квитирована;
- **Пропадание** - настройки палитры для событий в состоянии пропала.

Для выполнения настроек палитры необходимо выполнить двойной клик левой клавишей мыши в нужном поле. Данное поле изменит свой вид и в нём появится кнопка, на которую нужно нажать (см. рис. 3.368).

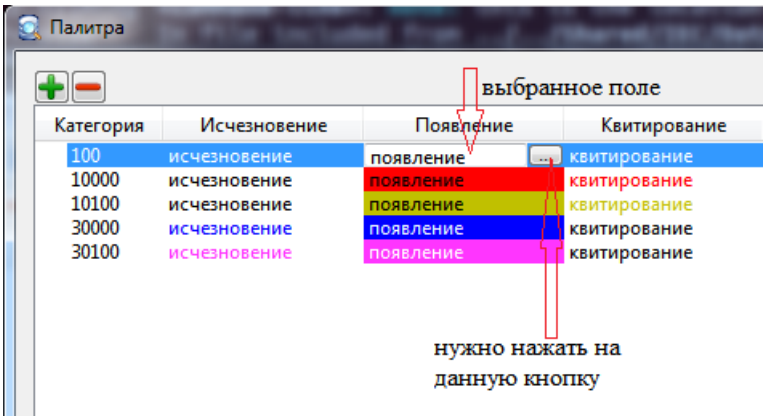


Рисунок 3.368 - Вид поля после двойного клика мышью

По нажатию на кнопку  откроется окно для настроек цветов отображения (см. рис. 3.369).

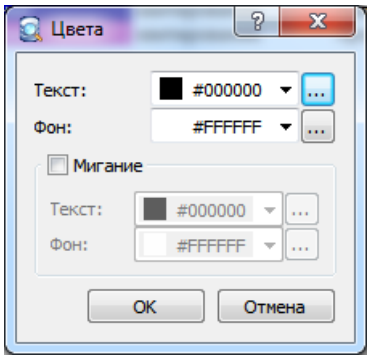


Рисунок 3.369 - Окно настроек цветов отображения

В окне настроек цветов отображения можно настроить цвет текста и цвет фона. Так же, можно настроить периодическую (каждые 250 мсек) смену цвета текста и цвета фона (мигание). Для этого нужно активировать галочку 

и настроить второй вариант цвета текста и цвета фона.

Цвет текста и фона можно выбрать из вариантов, предложенных в выпадающем меню (см. рис. 3.370) или нажать на кнопку справа от выпадающего меню и выбрать цвет в дополнительном открывшемся окне (см. рис. 3.371).

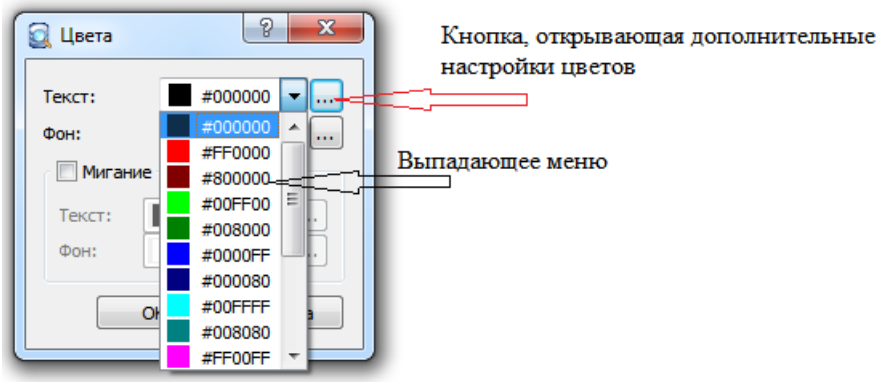


Рисунок 3.370 - Выбор цвета из выпадающего меню

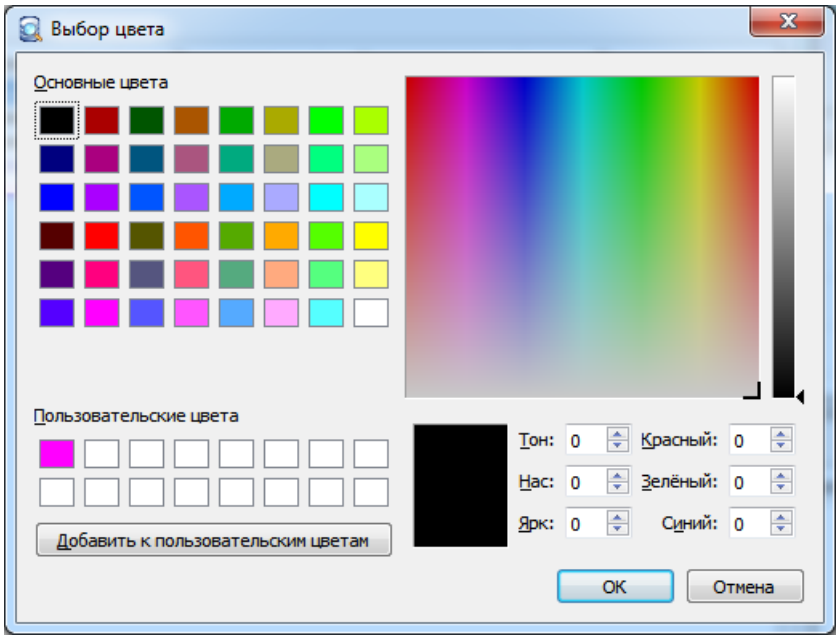


Рисунок 3.371 - Выбор цвета из дополнительного окна

Таким образом можно настроить различное отображение цвета текста и фона для разных категорий событий, что улучшит их визуальное восприятие для пользователя.

3.4.6.4. Настройки колонок в окне отображения событий

По нажатию на кнопку  ("Настроить колонки") откроется окно настроек для колонок окна отображения событий (см. рис. 3.372).

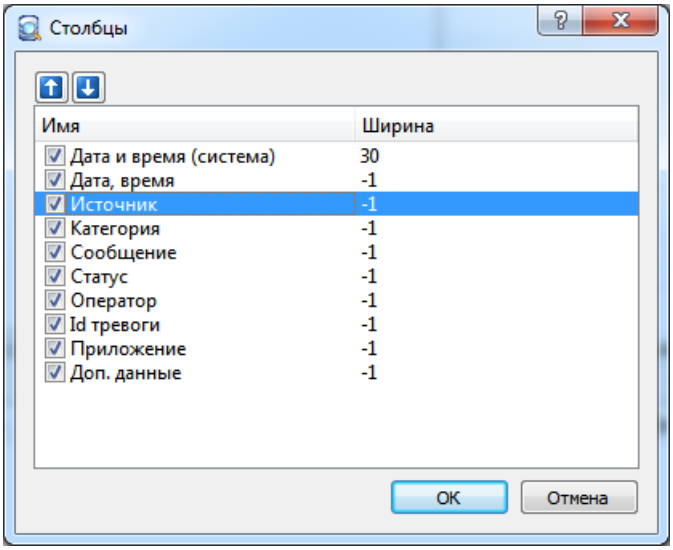
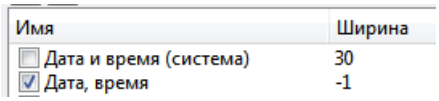


Рисунок 3.372 - Окно настроек колонок

В данном окне с помощью кнопок  можно менять порядок расположения колонок в окне отображения событий.

Так же помощью флаговых кнопок в столбце **Имя**  можно включать и исключать колонки из окна отображения событий.

В столбце **Ширина** задаётся ширина каждой колонки. Если значение равно -1, то устанавливается ширина по умолчанию. Если ввести положительное значение, то будет установлена ширина столбца, где ваше значение будет умножено на среднюю ширину символа установленного шрифта.

3.4.6.5. Подключение к архиву сигналов и регистратору событий.

Для подключения к архиву сигналов и событий необходимо нажать на кнопку  (Подключиться) в панели инструментов. По нажатию на данную кнопку появится диалоговое окно (см. рис. 3.373).

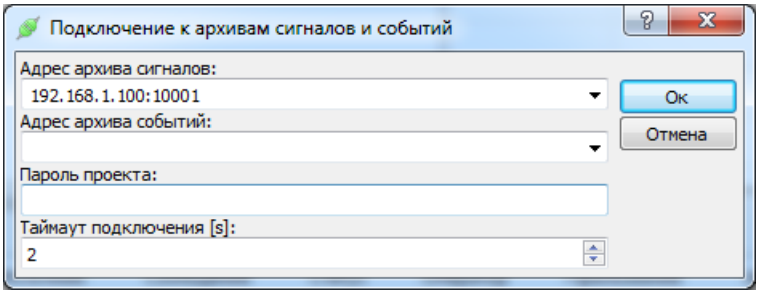


Рисунок 3.373 - Диалоговое окно подключения к архиву сигналов и событий

В данном диалоговом окне вводятся **Адрес архива сигналов** - это адрес приложения архив в формате ip-адрес узла: порт на котором запущено данное приложение и в таком же формате вводится адрес архива событий- поле ввода **Адрес архива событий**.

Поле **Пароль проекта** - если в проекте используется шифрование данных, то для подключения к архивам нужно указать пароль, который установлен в проекте для шифрования данных.

Поле **Таймаут подключения [s]** - это таймаут между попытками подключения к архивам.

При подключении к **Архиву событий** дополнительно будет выведено окно авторизации пользователя (см. рис. 3.374). Данное окно авторизации необходимо для разграничения прав на просмотр различных категорий событий. К примеру, в редакторе пользователей есть настройка прав на систему, и, если пользователь имеет права на просмотр архива ИБ (информационной безопасности), то он увидит события с категориями, которые относятся к информационной безопасности, иначе, не увидит.

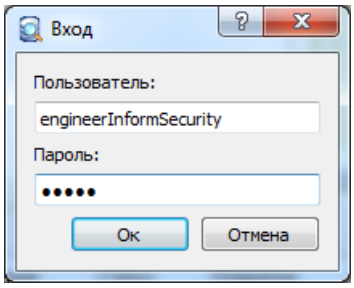


Рисунок 3.374 - Окно авторизации при подключении к архиву событий

Если у вас на компьютере есть папка с проектом, то можно воспользоваться кнопкой (Открыть проект). по нажатию на данную кнопку откроется стандартное для операционной системы диалоговое окно выбора папки, в котором необходимо выбрать папку Runtime проекта. В результате программа **Просмотр архивов** считает конфигурацию проекта и по нажатию на кнопку (Подключиться) автоматически задаст информацию в поля **Адрес архива сигналов** и **Адрес архива событий**.

3.4.6.6. Работа с архивом сигналов.

При работе с архивом сигналов используются окно отображения архивных значений сигналов и окно графического отображения.

Для работы с архивом сигналов необходимо чтобы в проекте было настроено приложение **Архив сигналов** и данный проект был запущен. Для подключения к приложению **Архив сигналов** используется кнопка (Подключиться) на панели инструментов (см. раздел 3.4.6.5).

После подключения необходимо добавить в окно отображения архивных значений сигналов нужные нам для анализа сигналы. Для этого используется кнопка (Добавить сигнал). По нажатию на данную кнопку открывается диалоговое окно добавления сигналов (см. рис. 3.375).

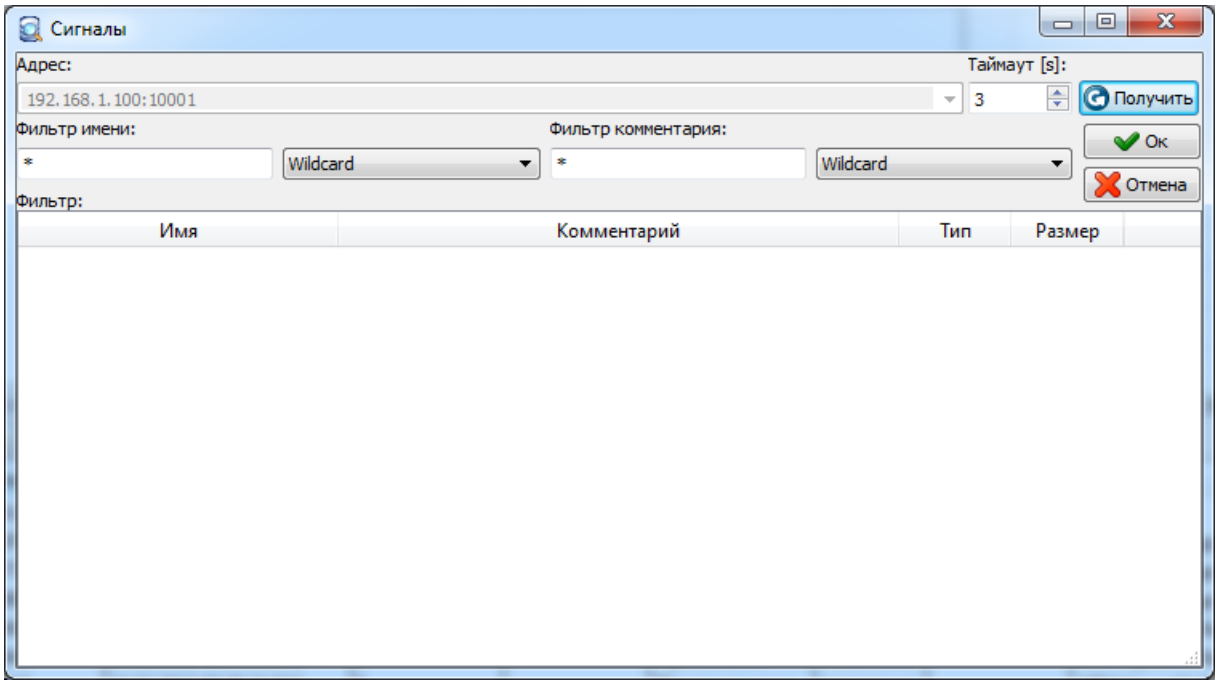
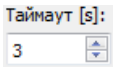
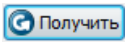


Рисунок 3.375 - Диалог выбора сигналов для последующего просмотра

В верхней части диалога размещены следующие элементы управления.

В поле **Адрес:** указаны IP адрес и порт программы, к которой будет произведено подключение для запроса списка сигналов. Поле 

предназначено для указания таймаута запроса. По нажатию кнопки 

будет произведён запрос списка сигналов. Результат запроса списка сигналов будет представлен в таблице.

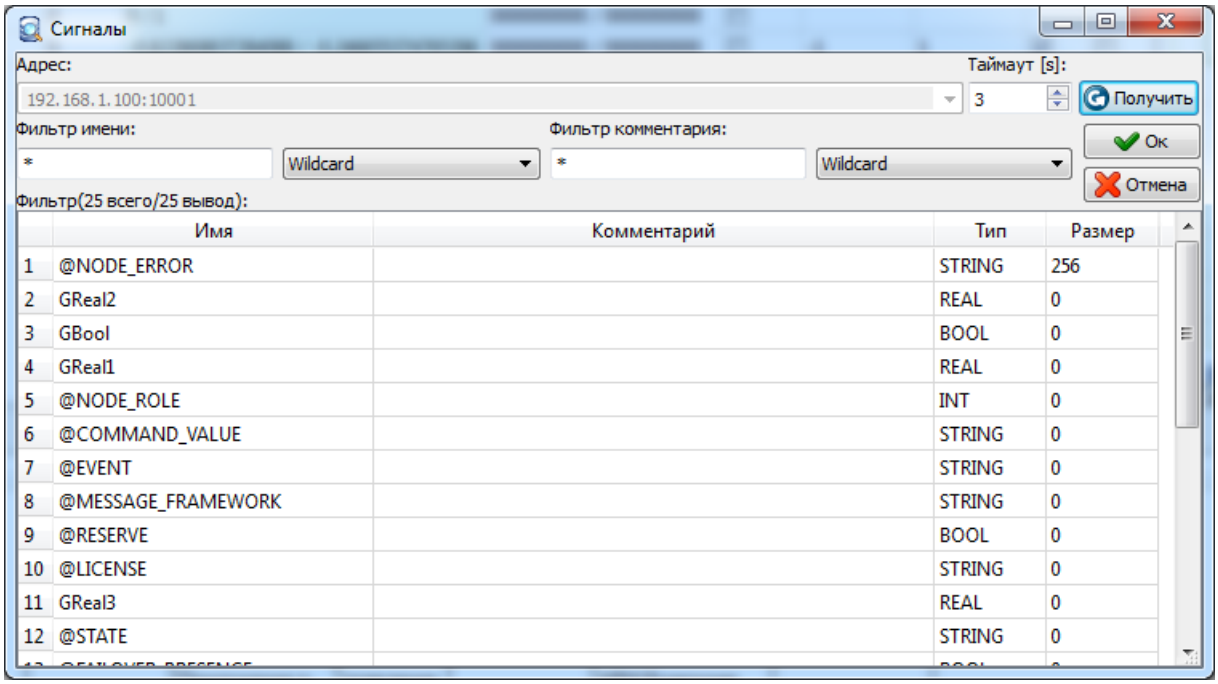


Рисунок 3.376 - Результат запроса сигналов

Для фильтрации списка синалов можно воспользоваться имеющимися фильтрами:

- **Фильтр имени** - фильтр по имени сигнала (поле Имя);
- **Фильтр комментария** - фильтр по комментариям к сигналу (поле Комментарий).

Пользователь может выбрать одну или несколько позиций в таблице и нажать кнопку Ok. Выбранные сигналы появятся в окне отображения архивных сигналов (см. рис. 3.377).

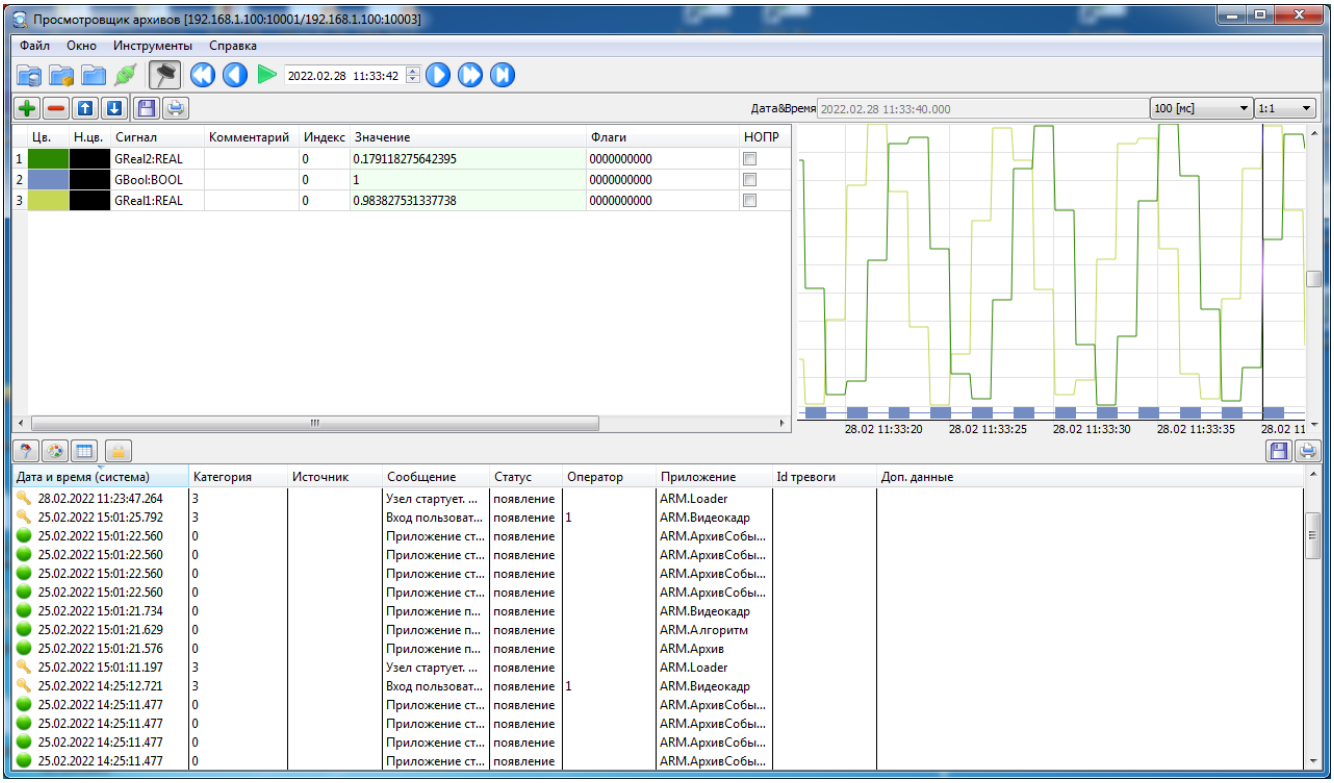


Рисунок 3.377 - Настроенное окно отображения архивных значений сигналов

Также в окне графического отображения сигналов появятся графики изменения выбранных сигналов.

Значения сигналов выводятся в соответствии с вертикальной линией (линия замера значения), которая находится в окне графического отображения сигналов. Также можно смотреть изменение сигнала в пределах диапазона времени, для этого линию замера значения нужно разделить на две с помощью левой клавишы мыши и настроить нужный диапазон (см. рис. 3.378).

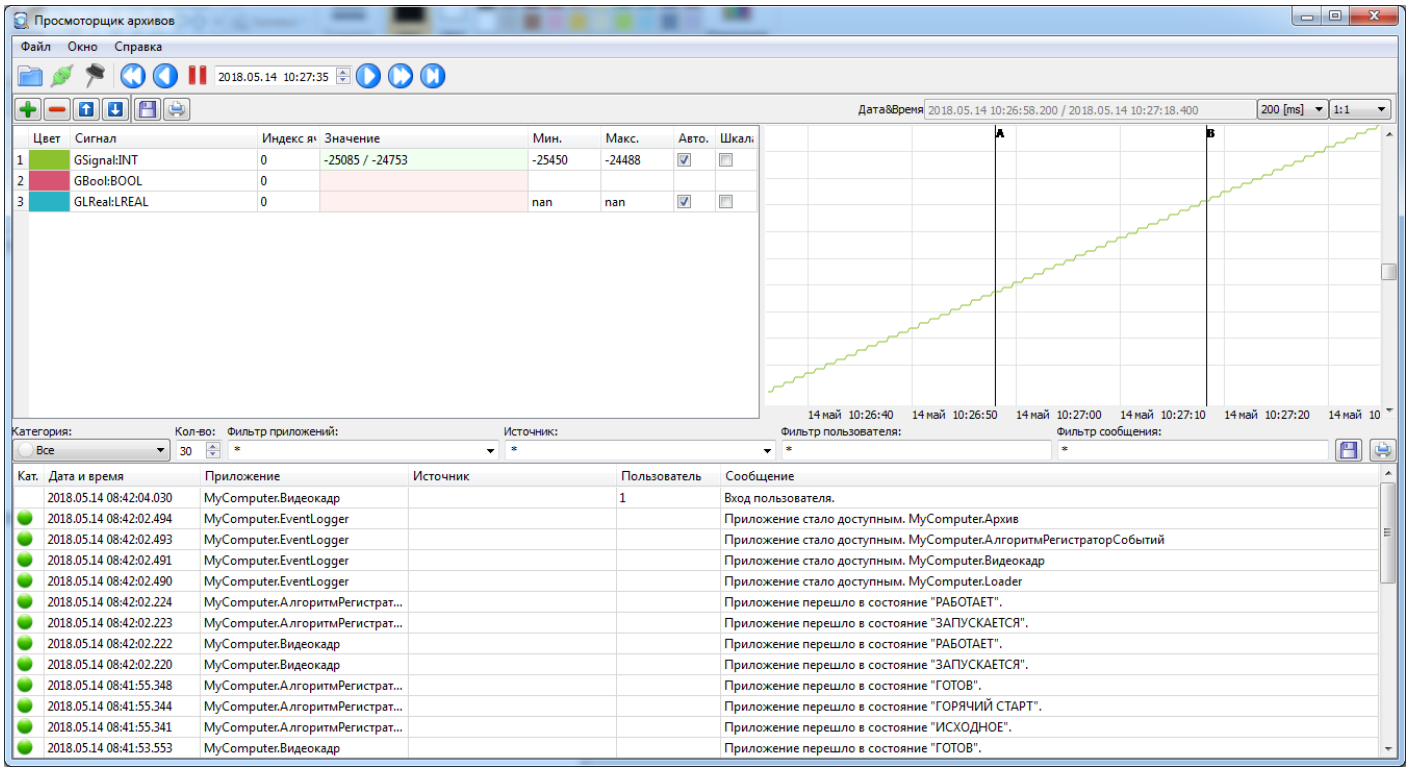


Рисунок 3.378 - Измерение значения сигнала в пределах диапазона времени

3.4.6.7. Работа с архивом событий

При работе с архивом событий используется окно отображения событий (см. рис. 3.379).

Для работы с архивом событий необходимо чтобы в проекте было добавлено приложение **Регистратор событий** и данный проект был запущен. Для подключения к приложению **Регистратор событий** используется кнопка  (Подключиться) на панели инструментов (см. раздел 3.4.6.5). После подключения к **Регистратору событий** в окне отображения событий появятся все события системы (см. рис. 3.379).

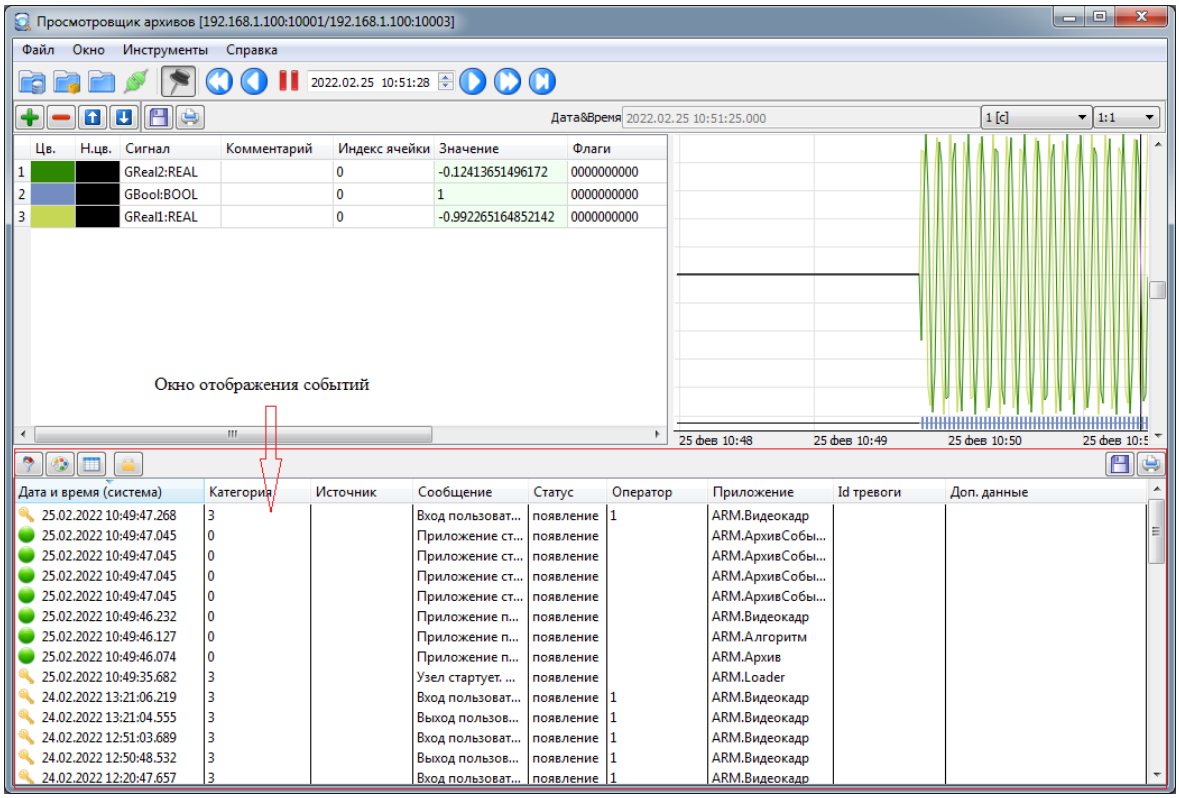


Рисунок 3.379 - Нижняя часть - Окно с событиями системы

В окне отображения событий можно настроить фильтрацию событий по различным критериям - кнопка , можно настроить отображение событий в различных состояниях разным цветом - кнопка , можно настроить отображение и ширину столбцов в окне - кнопка и можно заблокировать прокрутку списка событий - кнопка . Также можно сохранить список событий в текстовый файл, нажав на кнопку , и распечатать список событий, нажав кнопку . Данные кнопки находятся справа сверху от окна отображения событий.

ВНИМАНИЕ! На данный момент в окне событий максимально может быть отображено 100000 событий.

3.4.6.8. Просмотр событий младше указанной даты и времени

На данный момент в окне отображения событий программы **Просмотр архивов** максимально может отображаться 100000 событий. Для того чтобы посмотреть более старые события необходимо выполнить следующие действия.

1. Подключиться к архиву событий с помощью кнопки (Подключиться) на панели инструментов.

2. После подключения будут подгружаться события от текущей даты и времени (см. рис. 3.380). Максимальное количество отображаемых событий 100000.

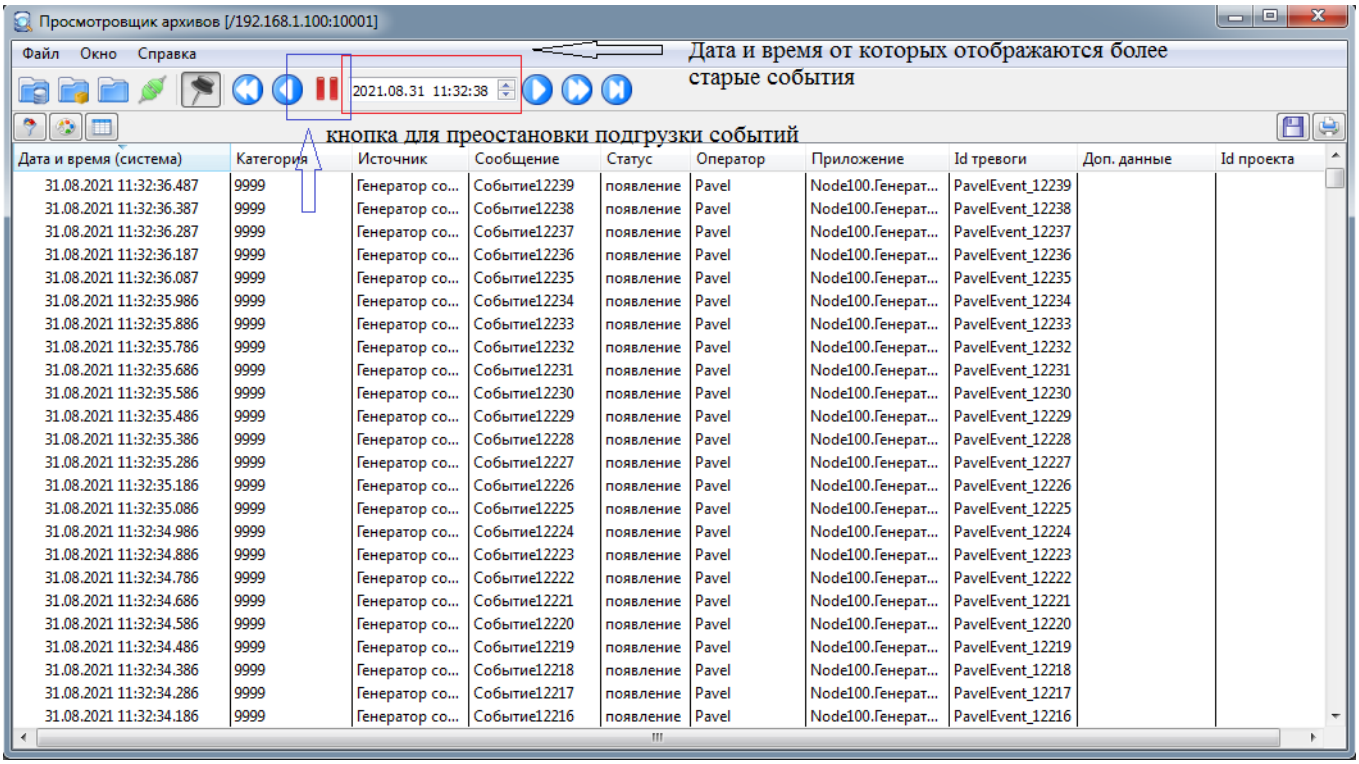


Рисунок 3.380 - Окно просмотра событий

3. Необходимо нажать на кнопку  и остановить подгрузку текущих событий.

4. В поле ввода даты и времени (см. рис. 3.380) необходимо ввести нужные дату и время и нажать Enter. После этого в окне событий отобразятся события, которые есть в архиве событий до этой даты и времени. Максимальное количество событий, которое подгрузится 100000.

3.4.6.9. Дополнительные возможности приложения Просмотр архивов

1. Приложение **Просмотр архивов** может быть запущено в режиме для отображения только событий информационной безопасности. Для запуска в данном режиме необходимо запустить приложение **ArchiveViewer** с ключом **-ISViewer**.

В режиме информационной безопасности будет видно только окно отображения событий и, если у пользователя есть права на просмотр событий информационной безопасности (категории 3 и 5), то выводится будут только они (см. рис. 3.381).

Так же в данном режиме формируются события об успешной и неуспешной авторизации пользователя, который подключается к приложению Просмотр архивов в режиме информационной безопасности.

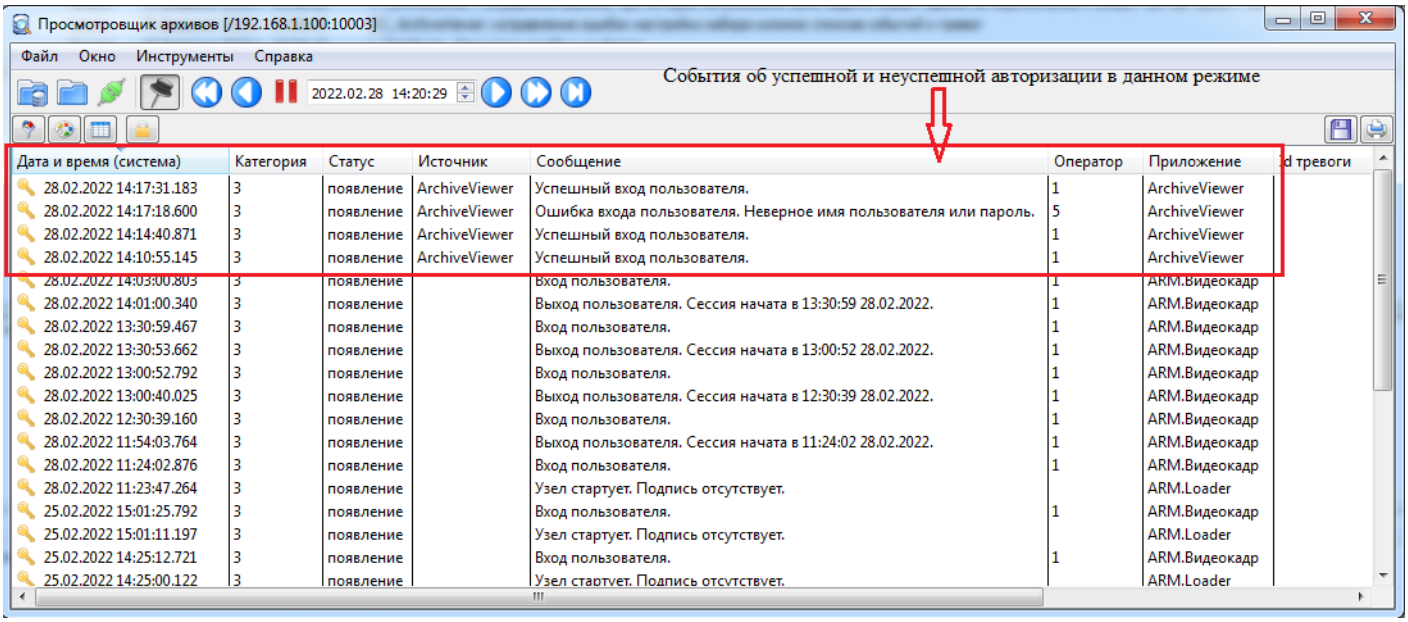


Рисунок 3.381 - Приложение Просмотр архивов,
открытое в режиме информационной безопасности

2. В приложении **Просмотр архивов** есть настройки для окна просмотра событий (см. рис. 3.382). Данные настройки автоматически сохраняются в файл ArchiveViewer.cfg.

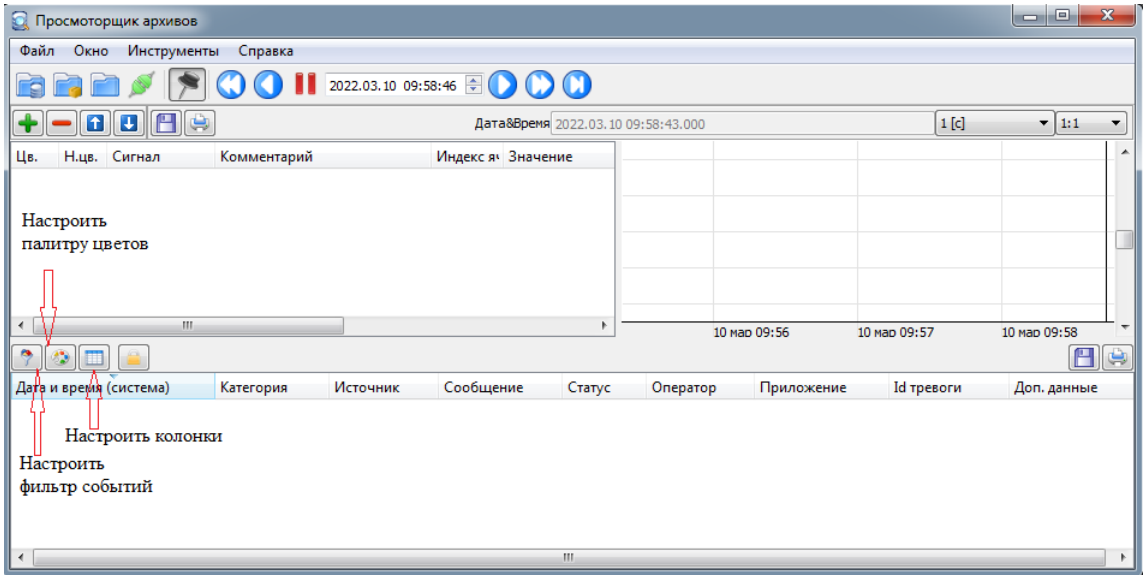


Рисунок 3.382 - Настройки в окне просмотра событий

Приложение **Просмотр архивов** можно запускать с предварительно настроенным файлом конфигурации. Для этого приложение нужно запустить с ключом:

-config=путь_к_файлу_конфигурации.

3.4.7. Программа "Loader" для загрузки проекта на узлах системы "Соната"

Данная программа предназначена для запуска и останова SCADA-системы "Соната". Программа может работать в двух режимах: в режиме демона (сервиса) и режиме командной строки. В первом случае программа **Loader** загружается в память и остается там резидентно, принимая

управляющие команды по UDP порту. Во втором случае программа производит запуск или останов SCADA системы и завершает свою работу. Запуск и останов SCADA системы загрузчик **Loader** осуществляет на основе конфигурационных файлов запуска приложений, которые отыскивает в папке проекта. Данные файлы имеют расширение "*.stp".

3.4.7.1. Командная строка

Формат командной строки для запуска программы **"Loader"** следующий: **Loader -hot|-cold|-stop [-daemon] [-sntp=address] [-sntp_shock] [-sntp_ver=3|4] [-sntp_offset=] [-node=prefix] [path]**. Так же есть и другие ключи, которые описаны ниже.

Ключи **-hot**, **-cold**, **-stop** предназначены для указания команды, которую нужно выполнить загрузчику:

- **-hot** – данная команда осуществляет запуск SCADA системы в режиме "горячего старта". При "горячем старте" системы обычные переменные инициализируются начальными значениями, переменные **RETAIN** и **PERSISTANCE** сохраняют свои последние значения;
- **-cold** – данная команда осуществляет запуск SCADA системы в режиме "холодного старта". При "холодном старте" системы обычные переменные и переменные **RETAIN** инициализируются начальными значениями, переменные **PERSISTANCE** сохраняют свои последние значения. Важный момент в том, что, если перезапустить один узел уже работающего проекта в режиме "холодного старта", а другие не перезапускать, то обычные переменные и переменные RETAIN инициализируются начальными значениями и сразу же снова получат свои последние значения до перезапуска, прочитав их с работающих узлов, что приводит к потере смысла перезапуска. Поэтому чтобы выполнить "Холодный старт", нужно остановить проект полностью и потом запустить его кнопкой "Холодный старт";
- **-stop** – данная команда осуществляет останов SCADA системы.

Описание других ключей:

- **-daemon** – ключ предназначен для указания программе Loader, что нужно запускаться в режиме демона (сервиса);
- **-sntp=address (сейчас не используется)** – опциональный ключ предназначен для активации синхронизации времени по SNTP протоколу с указанным серевером. ВНИМАНИЕ! При перезапуске узла происходит ударная синхронизация времени. В ходе работы проекта происходит безударная синхронизация времени (не более 100 мс за 1 секунду);
- **-sntp_ver** - данный ключ предназначен для установки версии протокола sntp клиента, если -sntp_ver=3, то будет работать по протоколу sntp (v3), если -sntp_ver=4, то будет работать по протоколу sntp (v4);
- **-sntp_shock** – опциональный ключ предназначен для включения постоянной ударной синхронизации, даже при работе проекта;
- **-sntp_offset** – количество секунд, которое прибавляется ко времени, которое получено от sntp - сервера, данный параметр может принимать отрицательные и положительные значения. Данный параметр признан устаревшим и крайне не рекомендуется к использованию. Вместо него используется настройка смещения времени внешнего SNTP в свойствах узла (см. раздел 3.2.2.5.3);
- Опциональный ключ **-node=prefix** предназначен для указания префикса имён конфигурационных файлов, специфичных для узла, на котором запускается **Loader**. В исходном состоянии папка проекта содержит все необходимые файлы для работы SCADA системы, в том числе конфигурационные файлы запуска приложений для всех (!) узлов. Для каждого узла начало имени (префикс) конфигурационных файлов уникально. Чтобы указать программе **Loader** группу конфигурационных файлов, которые необходимо обрабатывать, служит ключ **-node=prefix**;
- **-sntp_client_disable** - запрещение работы встроенного SNTP клиента;

- **-sntp_server_disable** - запрещение работы встроенного SNTP сервера;

- **-password**=NNNNNNNNNNNNNNNNNNNNNNNN, где NNNNNNNNNNNNNNNNNNNNNNNN - хэш пароля проекта (смотрите описание приложения Менеджер проекта - Настройки проекта);

- **-clock_period**=NNN, где NNN - период системного тика [us]. По умолчанию равен 100 [us]. Допустимый диапазон [100..1000] [us]. Данный параметр работает для операционной системы QNX;

- **-verification_interval**=NNN, где NNN - интервал (в секундах) периодической проверки Runtime на предмет изменения. Периодическая проверка целостности Runtime производится в фоновом режиме только при запущенном проекте. О результатах проверки Loader сообщает в свой лог и в события системы. Минимальный интервал времени проверки - 600 секунд;

- **-settings_update_interval=NNN** - задаёт интервал обновления списка пользователей узлов в секундах. По умолчанию период синхронизации пользователей между узлами 60 секунд. Можно указать меньший интервал синхронизации. Минимально возможный интервал 5 секунд;

- **-checksum_full[=file]** - выполняет проверку цифровой подписи проекта и выводит полный отчёт о всех проверенных файлах. Программа Loader при успешном завершении возвращает код 0, иначе -1. Дополнительно можно указать имя файла [=file], куда будет выведена информация, если имя файла не указано, то информация будет печататься в консоль;

- **-checksum[=file]** - выполняет проверку цифровой подписи проекта и выводит краткий отчёт. Программа Loader при успешном завершении возвращает код 0, иначе -1. Дополнительно можно указать имя файла [=file], куда будет выведена информация, если имя файла не указано, то информация будет печататься в консоль;

- **-checksum_show** - включает отображение контрольных сумм в логах приложений, в отчетах целостности файлов и в событиях информационной безопасности. По умолчанию показ контрольных сумм отключен;

- **-license[=file]** - выводит информацию о состоянии лицензии проекта. Дополнительно можно указать имя файла [=file], куда будет выведена информация, если имя файла не указано, то информация будет печататься в консоль;

- **-system_info[=file]** - выводится системная информация обо всех узлах проекта. Информация содержит конфигурацию оборудования и данные об операционных системах на узлах. Дополнительно можно указать имя файла [=file], куда будет выведена информация, если имя файла не указано, то информация будет печататься в консоль;

- **-gui** - опциональный ключ, который предназначен для включения диалога программы Центр управления (ControlCenter) с усеченным функционалом для отображения информации о данном узле (см. рис. 3.383)

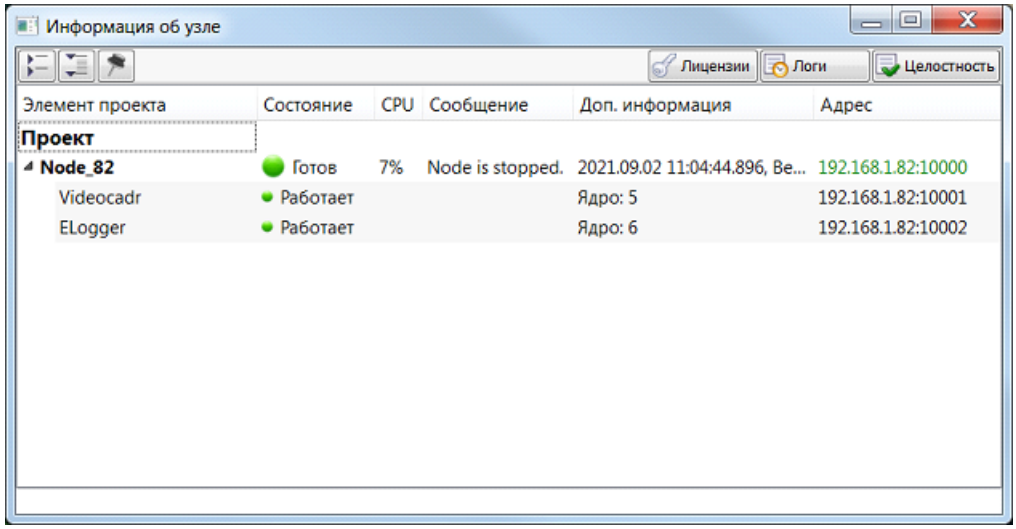


Рисунок 3.383 - Окно "Информация об узле, запускаемое при помощи параметра -gui"

На протяжении работы приложения Loader данное окно будет появляться при возникновения следующих событий:

- Запуск восстановления Loader'a после закрытия;
- Горячий старт проекта;
- Холодный старт проекта;
- Остановки проекта;
- Завершение работы Loader'a (с остановкой проекта).

В случае, если данные процедуры завершились успешно, окно «Информация об узле» скрывается. Также в системном трее будет отображена иконка для открытия окна «Информация об узле». На кнопке будет нарисован значок, соответствующий значку состояния узла. При завершении вышеописанных процедур появляется сообщение о результате процедуры (см.).

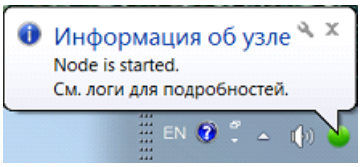


Рисунок 3.384 - Сообщение с результатом процедуры запуска проекта

При попытке закрытия данного окна оно сворачивается в трей. При завершении работы приложения Loader окно автоматически закрывается.

Примечание: в ОС QNX системный трей отсутствует, поэтому видимость окна «Информация об узле» управляется при помощи соответствующего значка на панели задач.

Последним аргументом командной строки является путь к папке проекта **-path**, содержащей конфигурационные файлы для запуска приложений.

ВНИМАНИЕ! Распространение проекта запрещено при стартующем или останавливающимся узле. При попытке распространения Distributor (приложение, распространяющее проект) выдаст ошибку "Доступ запрещён (403)". Возможна ситуация, когда из-за какой-либо ошибки в приложениях, в ходе старта узла, он не запустился до конца. Это состояние также считается состоянием старта узла, при котором распространение проекта запрещено. Для выхода из этой ситуации нужно из центра управления подать команду на останов проекта.

3.4.7.1.1. Принцип работы приложения Loader при включённом шифровании данных

При включённом шифровании данных в SCADA "СОНАТА" (описание смотри раздел 3.2.2.8) приложение Loader работает следующим образом:

- По умолчанию, когда приложение Loader только стартует, шифрования нет, так как он не считал проект и не знает, есть ли пароль или нет;
- Когда проект загружается, приложение Loader получает информацию и, если включено шифрование данных, переключается на шифрование, с использованием пароля проекта;
- Если остановить работающий проект с шифрованием данных, то приложение Loader снова выключает шифрование;
- Чтобы шифрование было включено всегда, нужно добавить в аргументы командной строки Loader параметр -password=NNNNNNNN, где NNNNNNNN – это хеш от пароля проекта. Данный хеш отображается в диалоге свойств проекта в ProjectManager (описание смотри раздел 3.2.2.8).

3.4.8. Редактор графических и консольных приложений IECEditor

Программный модуль **IECEditor** работает в составе SCADA-системы «Соната» и предназначен для редактирования графических и консольных типов приложений, основанных на стандартах IEC 61131 и IEC 61499. Данными приложениями являются графическое приложение **APPLICATION.IEC.WINDOW** и консольное приложение **APPLICATION.IEC.CONSOLE**.

Работа в данном редакторе заключается в формировании библиотеки типов функциональных блоков приложения и редактировании типа приложения. Подобно приложениям, которые, выполняясь на узле, являются экземплярами того или иного типа приложения, функциональные блоки, входящие в те или иные приложения, являются экземплярами того или иного типа функционального блока. Редактирование типа приложения в программном модуле **IECEditor** осуществляется так же, как и редактирование любого другого типа функционального блока.

Вызов данного редактора связан с созданием и настройкой интерфейса приложения **APPLICATION.IEC.WINDOW** (см. раздел 3.3.4.2) или **APPLICATION.IEC.CONSOLE** (см. раздел 3.3.3.1). Данные приложения отличаются лишь тем, что для **APPLICATION.IEC.CONSOLE** отсутствует возможность создавать графические типы функциональных блоков, что есть у приложения **APPLICATION.IEC.WINDOW**. Сами принципы работы в редакторе **IECEditor** идентичны для обоих приложений.

На рис. 3.385 изображено основное окно редактора IEC-приложений **IECEditor**.

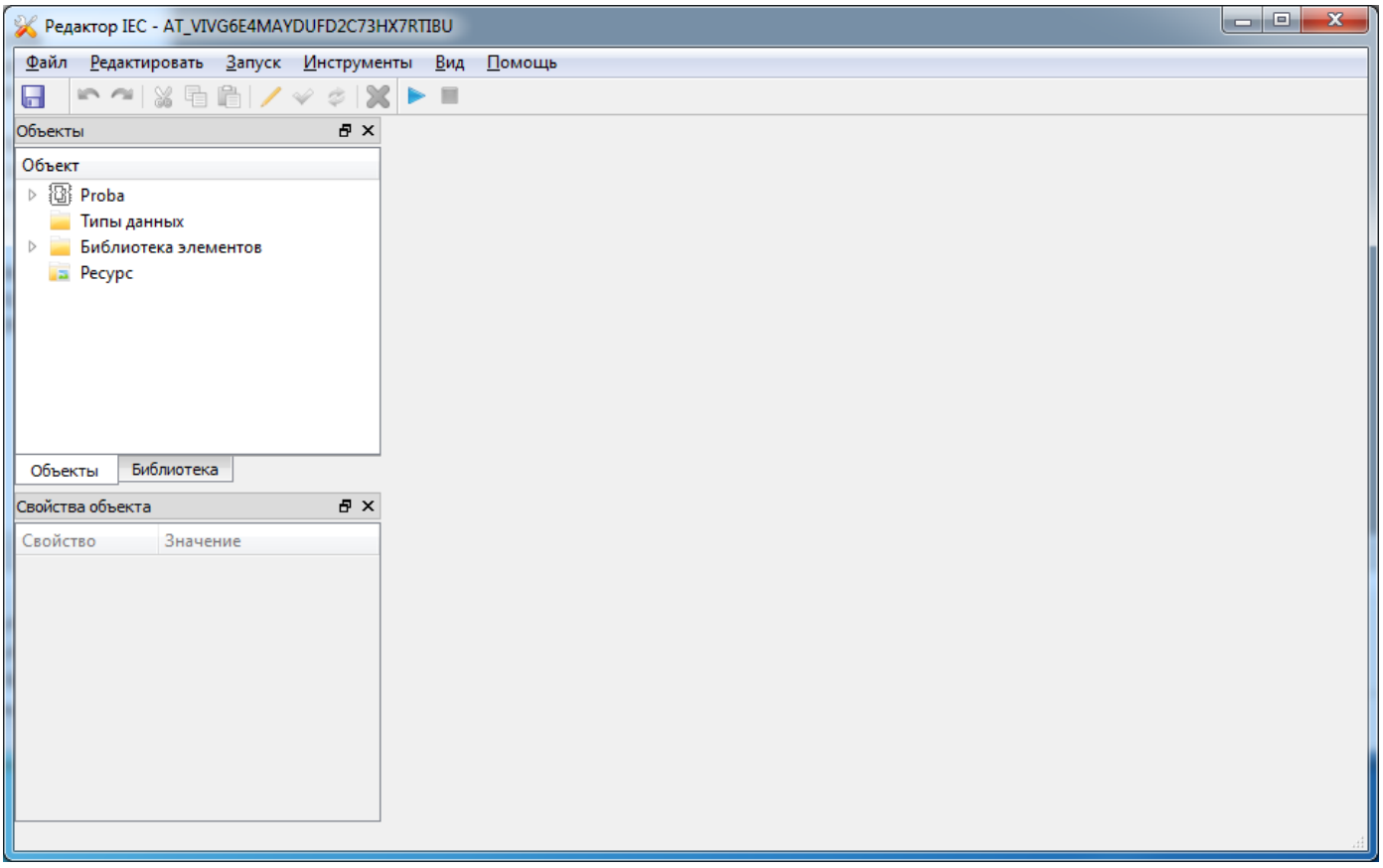


Рисунок 3.385 - Основное окно редактора IEC-приложений IECEditor

3.4.8.1. Описание интерфейса программы IECEditor

Программа **IECEditor** открывается в отдельном окне. Главное окно программы содержит элементы, необходимые для формирования структуры библиотеки типов функциональных блоков приложения, редактирования типов функциональных блоков и отладки приложения. Внешний вид главного окна приведен на рис. 3.386.

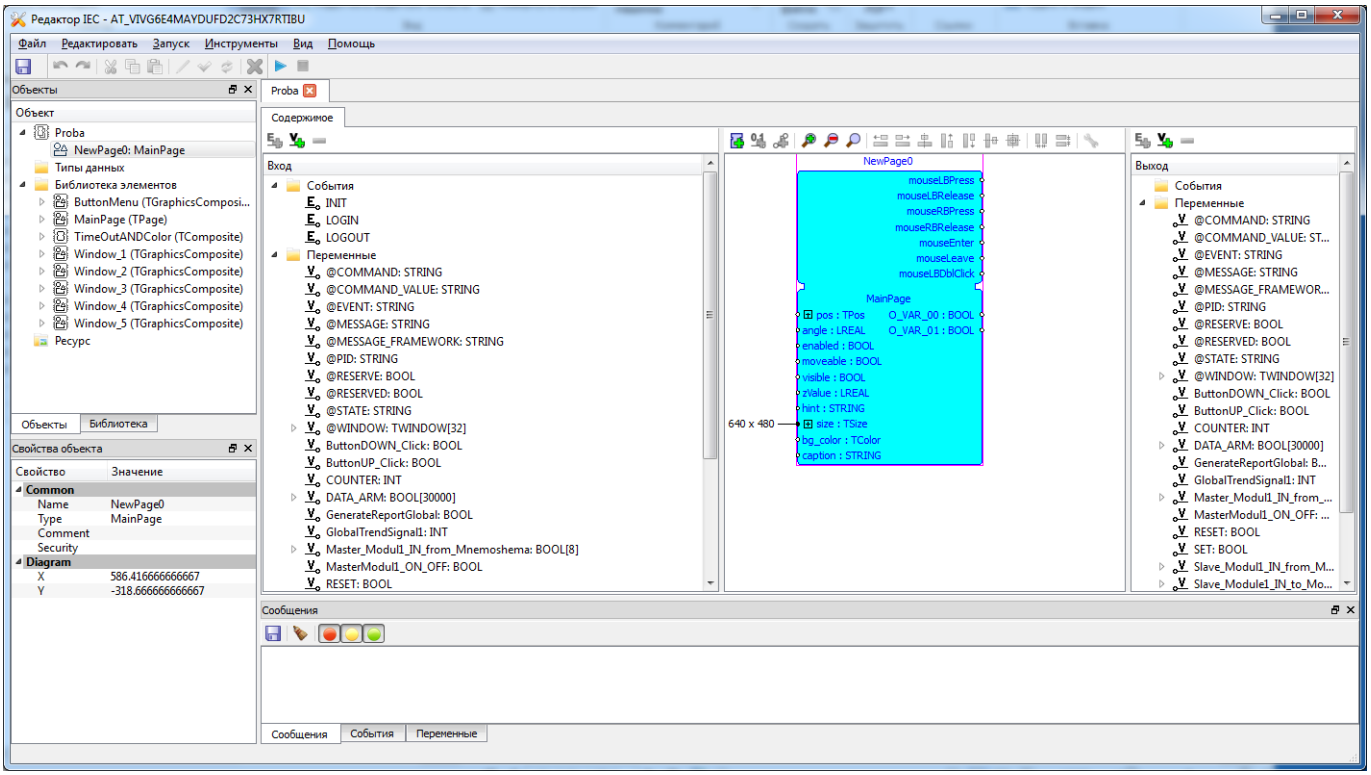


Рисунок 3.386 - Главное окно программы IECEditor

Главное окно редактора содержит следующие элементы:

- в верхней левой части имя редактора **Редактор IEC** и уникальное имя **AT_******;
- в верхней правой части находятся кнопки управления окном редактора  , соответственно свернуть, развернуть и закрыть окно редактора;
- главное меню (см. раздел 3.4.8.1.1);
- панель инструментов (см. раздел 3.4.8.1.1);
- окно дерева объектов (см. раздел 3.4.8.1.2);
- окно библиотеки типов функциональных блоков (см. раздел 3.4.8.1.3);
- окно редактора свойств (см. раздел 3.4.8.1.4);
- окно редакторов типов функциональных блоков (см. раздел 3.4.8.1.8);
- окно списка сообщений (см. раздел 3.4.8.1.5);
- окно списка событий (см. раздел 3.4.8.1.6);
- окно переменных отладчика (см. раздел 3.4.8.1.7).

3.4.8.1.1. Главное меню и панель инструментов

Главное меню содержит полный список операций редактора.

Главное меню делится на следующие разделы:

Файл – группа команд работы с файлами (см. раздел 3.4.8.1.1.1);

Редактировать – группа команд редактирования типа приложения (см. раздел 3.4.8.1.1.2);

Запуск – группа команд запуска и отладки приложения (см. раздел 3.4.8.1.1.3);

Инструменты – группа команд различного назначения (см. раздел 3.4.8.1.1.4);

Вид – группа команд, управляющих видимостью отдельных окон редактора (см. раздел 3.4.8.1.1.5);

Помощь – группа команд доступа к справочной информации (см. раздел 3.4.8.1.1.6).

Панель инструментов расположена под главным меню – на этой панели расположены значки (кнопки), которые вызывают наиболее часто используемые команды.

Это следующие кнопки:

 («Сохранить») – команда для сохранения типа приложения (из раздела **Файл**);

 («Назад») – отменить предыдущее действие по редактированию (из раздела **Редактирование**);

 («Вперед») – повторить ранее отменённое действие (из раздела **Редактирование**);

 («Вырезать») – вырезать выделенный объект (объекты) и сохранить его в буфере памяти (из раздела **Редактирование**);

 («Копировать») – копировать выделенный объект (объекты) в буфер памяти (из раздела **Редактирование**);

 («Вставить») – вставить вырезанный или скопированный объект (объекты) из буфера памяти (из раздела **Редактирование**);

 («Редактировать») – открыть тип функционального блока для просмотра или редактирования (из раздела **Редактирование**);

 («Применить) – сохранить сразу все сделанные при редактировании функционального блока изменения (из раздела **Редактирование**);

 («Отмена) – отменить сразу все сделанные при редактировании функционального блока изменения (из раздела **Редактирование**);

 («Удалить) – удалить выделенный объект (объекты) (из раздела **Редактирование**);

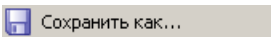
 («Запустить») – запустить приложение для отладки (из раздела **Запуск**);

 («Остановить») – остановить отладку приложения (из раздела **Запуск**).

3.4.8.1.1.1. Файл – группа команд работы с файлами

Раздел Файл (группа команд работы с файлами) содержит следующие команды и подразделы:

– сохранить тип приложения;

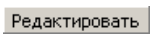


– сохранить тип приложения под другим именем;

– закрыть тип приложения;

– завершить работу с редактором **IECEditor**.

3.4.8.1.1.2. Редактировать – группа команд редактирования типа приложения

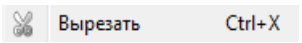
Раздел 

(группа команд редактирования **типа приложения**) содержит следующие команды и подразделы:

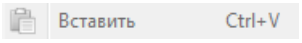
– отменить предыдущее действие по редактированию;

– повторить ранее отменённое действие;

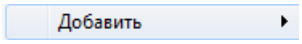
– копировать выделенный объект (объекты) в буфер памяти;



– вырезать выделенный объект (объекты) и сохранить его в буфере памяти;



– вставить вырезанный или скопированный объект (объекты) из буфера памяти;



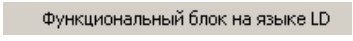
подраздел для создания новых типов функциональных блоков или папок библиотеки, содержит следующие команды:

- 

– создать папку в дереве объектов;
- 

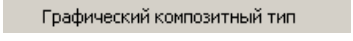
– создать папку в дереве объектов;
- 

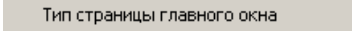
– создать папку в дереве объектов;
- 

– создать папку в дереве объектов;
- 

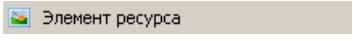
– создать папку в дереве объектов;
- 

– создать базовый тип функционального блока;
- 

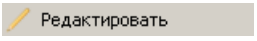
– создать композитный не графический тип функционального блока;
- 

– создать графический композитный тип функционального блока;
- 

– создать тип функционального блока – страницу главного окна;
- 

– создать тип функционального блока – окно;
- 

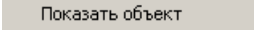
– добавить элемент ресурса;



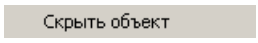
– открыть тип функционального блока для просмотра или редактирования;

– сохранить сразу все сделанные при редактировании функционального блока изменения;

– отменить сразу все сделанные при редактировании функционального блока изменения;



– показать скрытый графический объект в редакторе мнемосхем;



– скрыть видимый графический объект в редакторе мнемосхем;

– настроить объект;

– удалить выделенный объект (объекты);

– редактировать свойства объекта (папки или типа функционального блока).

3.4.8.1.1.3. Запуск – группа команд запуска и отладки приложения

Раздел 

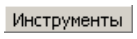
(группа команд запуска и отладки приложения) содержит следующие команды:

– запустить приложение для отладки;

– запустить приложение без отладки;

– остановить отладку приложения.

3.4.8.1.1.4. Инструменты – группа команд различного назначения

Раздел 

(группа команд различного назначения) содержит следующие команды:

- проводит проверку вашей IEC-программы;

- формирует данные для вынесения в отдельный модуль;

- экспортировать данные в текстовый файл;

- импортировать данные из ранее сохранённых;

- запуск редактора настроек **IECEditor** (см. раздел 3.4.8.3).

3.4.8.1.1.5. Вид – группа команд, управляющих видимостью отдельных окон редактора

Раздел 

(группа команд, управляющих видимостью отдельных окон редактора **IECEditor**) содержит следующие команды:

– показать/скрыть окно свойств объекта;

– показать/скрыть окно дерева объектов;

– показать/скрыть окно дерева библиотеки функциональных блоков;

– показать/скрыть окно сообщений;

– показать/скрыть окно событий;

– показать/скрыть окно переменных отладчика.

По умолчанию при открытии главного окна программы **IECEditor** видны окна **Свойства объекта**, **Объекты** и **Библиотека**.

3.4.8.1.1.6. Помощь – группа команд доступа к справочной информации

Раздел Помощь (группа команд доступа к справочной информации) содержит следующие команды:

Справка – вызвать справку (не реализовано);

О программе... – вывести информацию о редакторе **IECEditor**.

3.4.8.1.2. Окно дерева объектов

Окно дерева объектов расположено в левой части главного окна редактора и открывается при нажатии на вкладку Объекты. Данное окно служит для отображения и формирования структуры библиотеки типов функциональных блоков типа приложения, а также редактирования списка ресурсов типа приложения. Окно дерева объектов приведено на рис. 3.387.

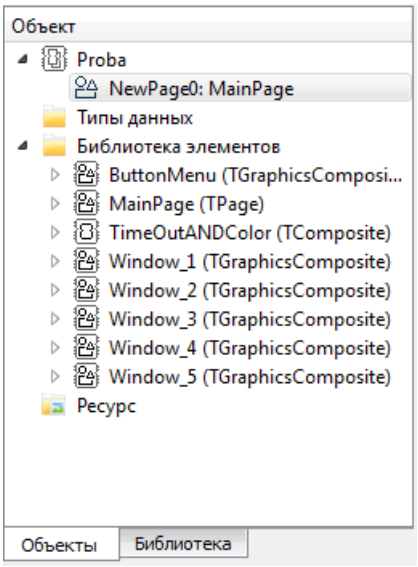


Рисунок 3.387 - Окно дерева объектов

Окно дерева объектов содержит узел редактируемого типа приложения. На рис. 3.387 узлом редактируемого типа приложения является узел **Proba**, где **Proba** - это имя IEC-приложения, заданное ранее в интерфейсе приложения. Именно этот узел запускается на исполнение, так что конечный вариант вашего IEC-приложения, собранный из различных функциональных элементов, должен быть добавлен на данный узел.

Далее располагается узел пользовательских типов данных (узел **Типы данных**), множество узлов типов функциональных блоков, входящих в библиотеку типов редактируемого типа приложения (узел **Библиотека элементов**), а также узлы ресурса типа приложения (узел **Ресурс**).

Узел типа приложения всегда расположен в верхней части дерева объектов. Вложенными узлами типа приложения являются узлы объектов, входящих в тип приложения. Для каждого узла объекта в дерево выводится имя объекта и его тип.

Узлы, входящие в папку "**Типы данных**", представляют собой:

- пользовательские типы данных (см. раздел 3.4.8.1.9).

Узлы, входящие в папку «**Библиотека элементов**», представляют собой:

- узлы типов функциональных блоков;
- узлы объектов, входящих в состав композитных функциональных блоков;
- узлы папок, служащих для организации библиотеки типов.

Узлы, входящие в папку «**Ресурс**», представляют собой:

- узлы файлов ресурса;
- узлы папок, служащих для организации ресурса типа приложения.

При помощи дерева объектов оператор может выполнить следующие действия:

- создать новый тип функционального блока;
- удалить тип функционального блока;
- изменить тип функционального блока;
- открыть для редактирования или просмотра любой тип функционального блока;
- изменить иерархию типов функциональных блоков;
- добавить новый элемент ресурса;
- изменить элемент ресурса;
- изменить иерархию элементов ресурса.

Операции в дереве объектов могут осуществляться при помощи главного меню (см. раздел 3.4.8.1.1), панели инструментов (см. раздел 3.4.8.1.1), а также контекстного меню дерева объектов. Контекстное меню приведено на рис. 3.388.

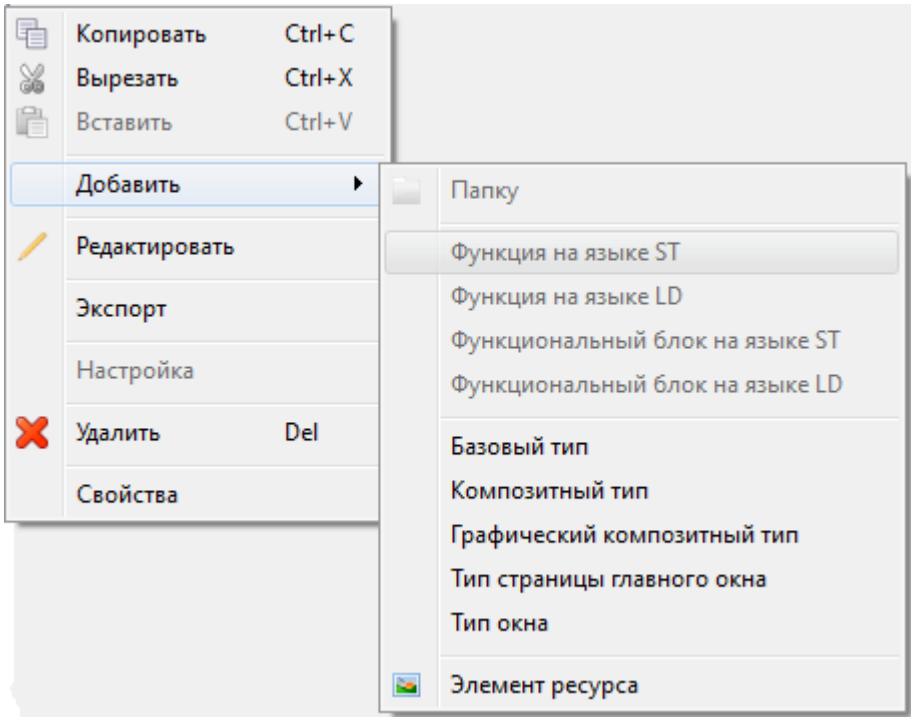
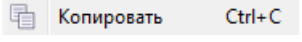
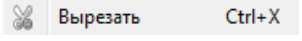


Рисунок 3.388 - Контекстное меню дерева объектов

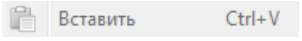
Вызов контекстного меню происходит при нажатии на правую кнопку «мыши» в дереве объектов. Контекстное меню содержит следующие операции:



– копировать выделенный тип функционального блока в буфер памяти;



– вырезать выделенный тип функционального блока в буфер памяти;



– вставить вырезанный или скопированный тип функционального блока из буфера памяти;

– подраздел для добавления в проект новых типов функциональных блоков или папок библиотеки, содержит следующие команды:

– добавить папку в дереве объектов;

– не используется;

– не используется;

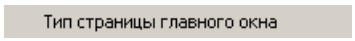
– не используется;

– не используется;

– добавить базовый тип функционального блока;

– добавить композитный не графический тип функционального блока;

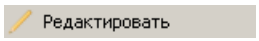
– добавить графический композитный тип функционального блока;



– добавить тип функционального блока – страницу главного окна;

– добавить тип функционального блока – диалоговое окно;

– добавить элемент ресурса;



– открыть тип функционального блока для просмотра или редактирования;

– экспорт данных о выделенном функциональном блоке в текстовый файл;

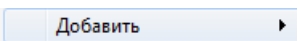
– настроить тип функционального блока;

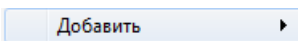
– удалить выделенный тип функционального блока;

– редактировать свойства объекта (папки или типа функционального блока).

3.4.8.1.2.1. Добавление папки в дерево объектов

Для добавления папки в дерево объектов необходимо выделить узел на дереве, куда будет добавлена новая папка, после чего необходимо в разделе 

главного меню выбрать команду 

либо выбрать команду 

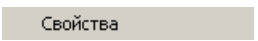
контекстного меню дерева объектов. Затем в выпадающем меню следует выбрать команду 

. Редактор создаст папку и добавит ее в дерево объектов.

3.4.8.1.2.2. Изменение свойств типа функционального блока

Для изменения имени, группы безопасности или описания необходимо выделить в дереве узел, соответствующий изменяемому типу функционального блока, после чего в разделе 

главного меню выбрать команду 

либо выбрать команду 

контекстного меню дерева объектов (см. рис. 3.389). Программа выведет диалог редактирования свойств типа функционального блока (см. рис. 3.390).

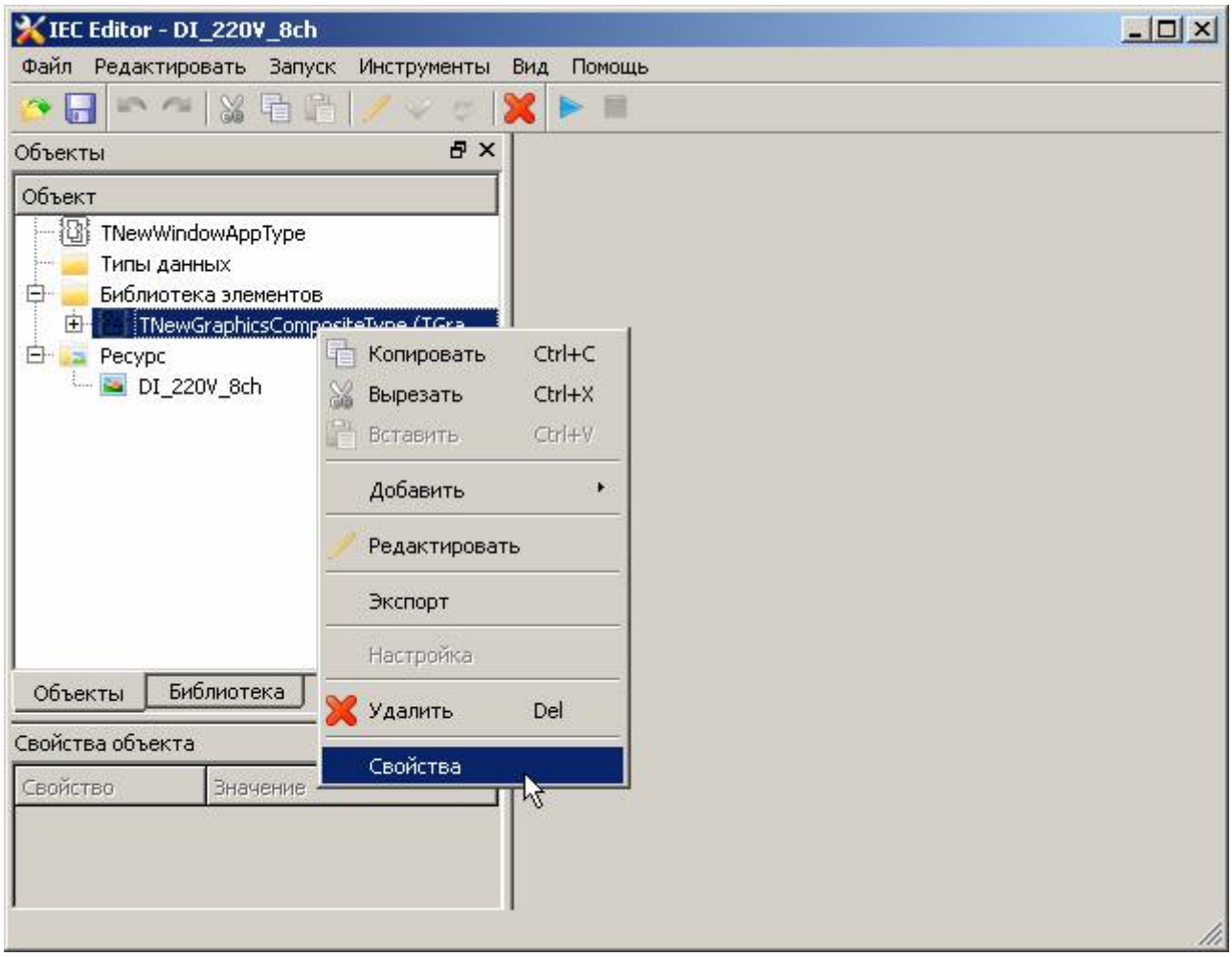


Рисунок 3.389 - Контекстное меню вызова диалога редактирования свойств

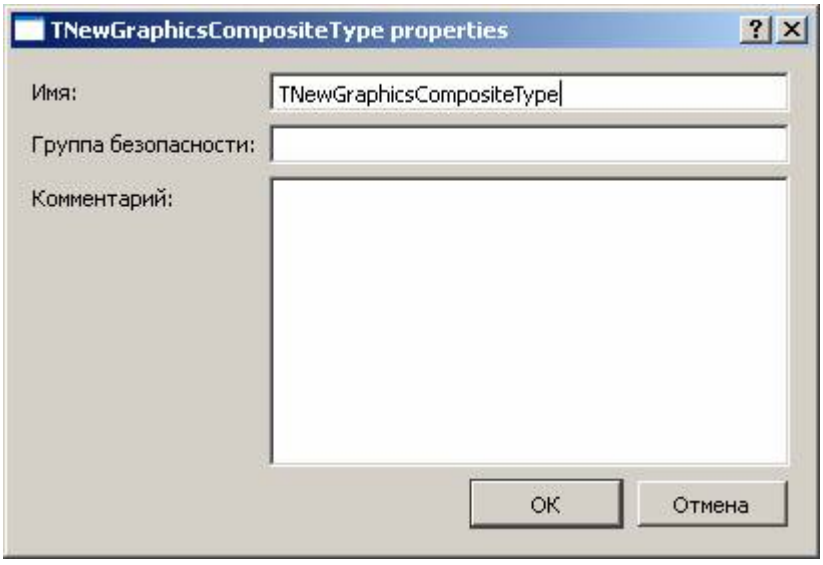


Рисунок 3.390 - Диалог редактирования свойств типа функционального блока

Диалог редактирования состоит из следующих полей ввода:

- **Имя** - имя типа функционального блока;

- **Группа безопасности** - название группы безопасности для объекта, которая используется при настройках прав у пользователя, если данную группу указать, то пользователь сможет управлять данным объектом во время работы проекта, если не указать, то не сможет (см. ???);

- **Комментарий** - описание данного объекта.

Оператор должен изменить требуемые характеристики и, затем, нажать на кнопку OK. Программа проверит уникальность имени. Если имя уникально, то данные будут сохранены и диалог будет закрыт. Если имя не уникально или же введенное имя некорректно, то будет выведено сообщение об ошибке.

При нажатии на кнопку Отмена изменения внесены не будут.

3.4.8.1.2.3. Изменение свойств папки

Для изменения имени или описания папки необходимо выделить узел в дереве, соответствующий изменяемой папке, после чего в разделе Редактировать главного меню выбрать команду Свойства либо выбрать команду Свойства контекстного меню дерева объектов (см. рис. 3.391). Программа выведет диалог редактирования свойств папки (см. рис. 3.392).

Оператор должен изменить требуемые характеристики и, затем, нажать на кнопку OK. Программа проверит уникальность папки. Если имя уникально, то данные будут сохранены и диалог будет закрыт. Если имя не уникально или же введенное имя некорректно, то будет выведено сообщение об ошибке.

При нажатии на кнопку Отмена изменения внесены не будут.

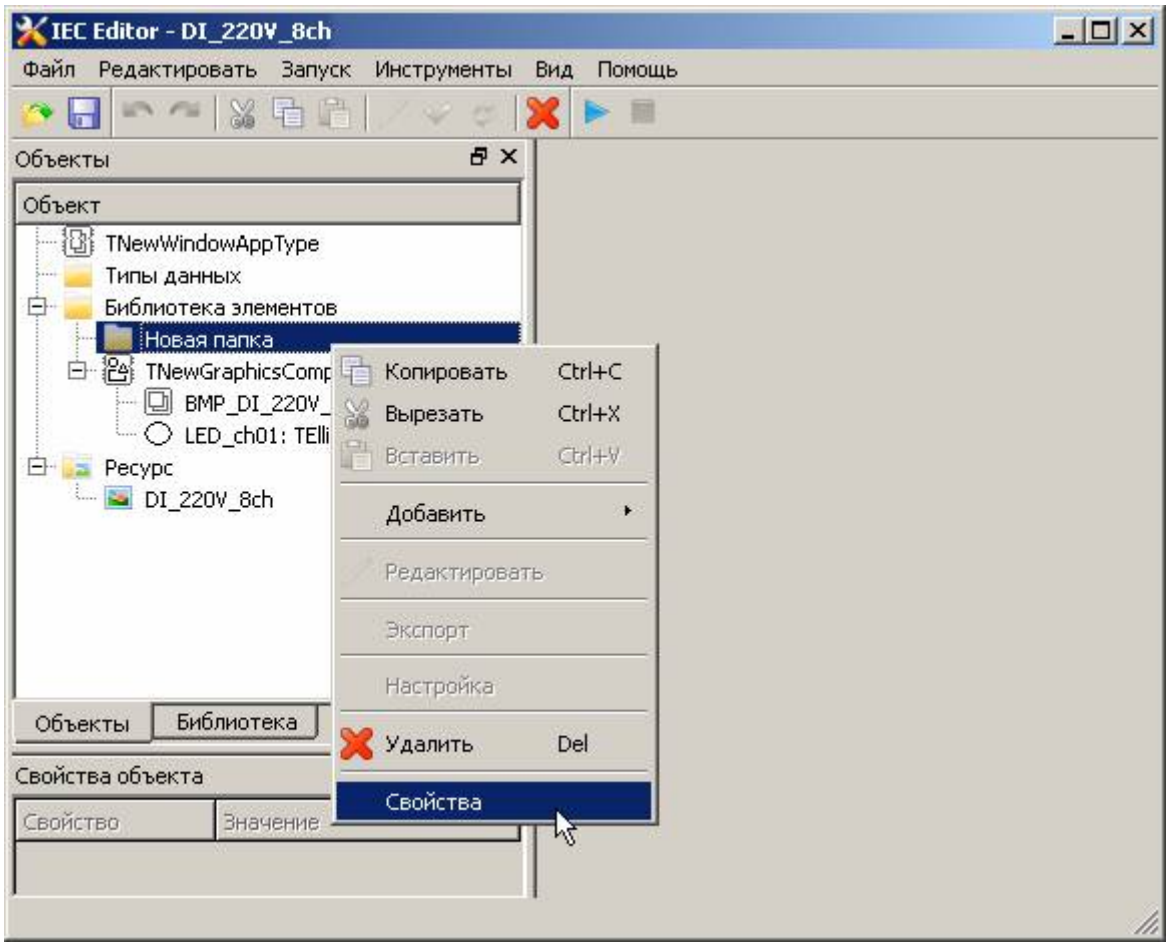


Рисунок 3.391 - Контекстное меню вызова диалога редактирования свойств папки

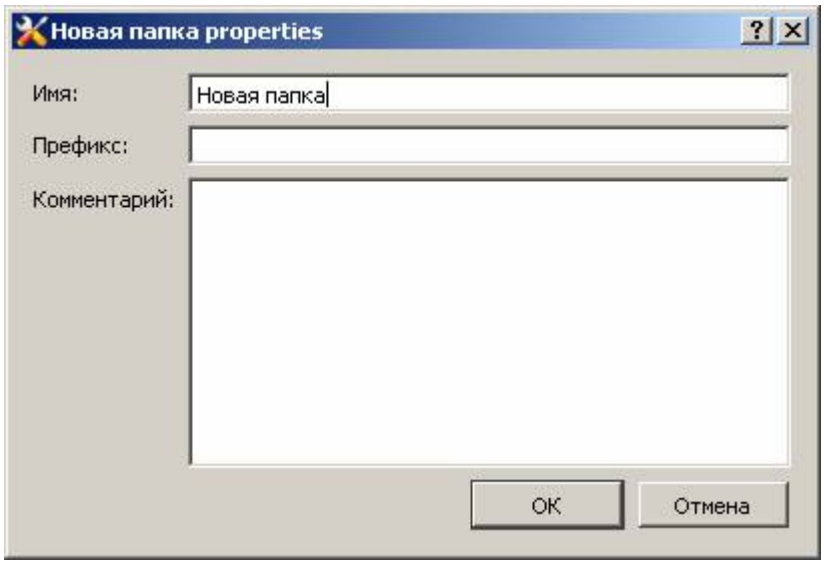


Рисунок 3.392 - Диалог редактирования свойств папки

3.4.8.1.2.4. Использование буфера обмена

Редактор позволяет использовать буфер обмена для копирования/вставки типов функциональных блоков или элементов ресурса. Папки копировать нельзя.

Для копирования типа функционального блока необходимо выделить копируемый тип, после чего в разделе Редактировать главного меню выбрать команду Копировать Ctrl+C, либо выбрать команду Копировать Ctrl+C контекстного меню дерева объектов (см. рис. 3.393), либо нажать на кнопку панели инструментов, либо нажать комбинацию клавиш **<Ctrl>+<C>** на клавиатуре. Редактор скопирует выбранный тип в оперативную память компьютера.

Для вырезания типа функционального блока необходимо выделить требуемый тип, после чего в разделе Редактировать главного меню выбрать команду Вырезать Ctrl+X, либо выбрать команду Вырезать Ctrl+X контекстного меню дерева объектов (см. рис. 3.393), либо нажать на кнопку панели инструментов, либо нажать комбинацию клавиш **<Ctrl>+<X>** на клавиатуре. Редактор скопирует выбранный тип в оперативную память компьютера, после чего удалит его, если это возможно.

Для вставки типа функционального блока необходимо выделить папку, в которую будет вставлен скопированный или вырезанный блок, после чего в разделе Редактировать главного меню выбрать команду Вставить Ctrl+V, либо выбрать команду Вставить Ctrl+V контекстного меню дерева объектов (см. рис. 3.394), либо нажать на кнопку панели инструментов, либо нажать комбинацию клавиш **<Ctrl>+<V>** на клавиатуре. Редактор вставит тип в указанную папку.

Копирование/вырезание/вставка элементов ресурса работает аналогично.

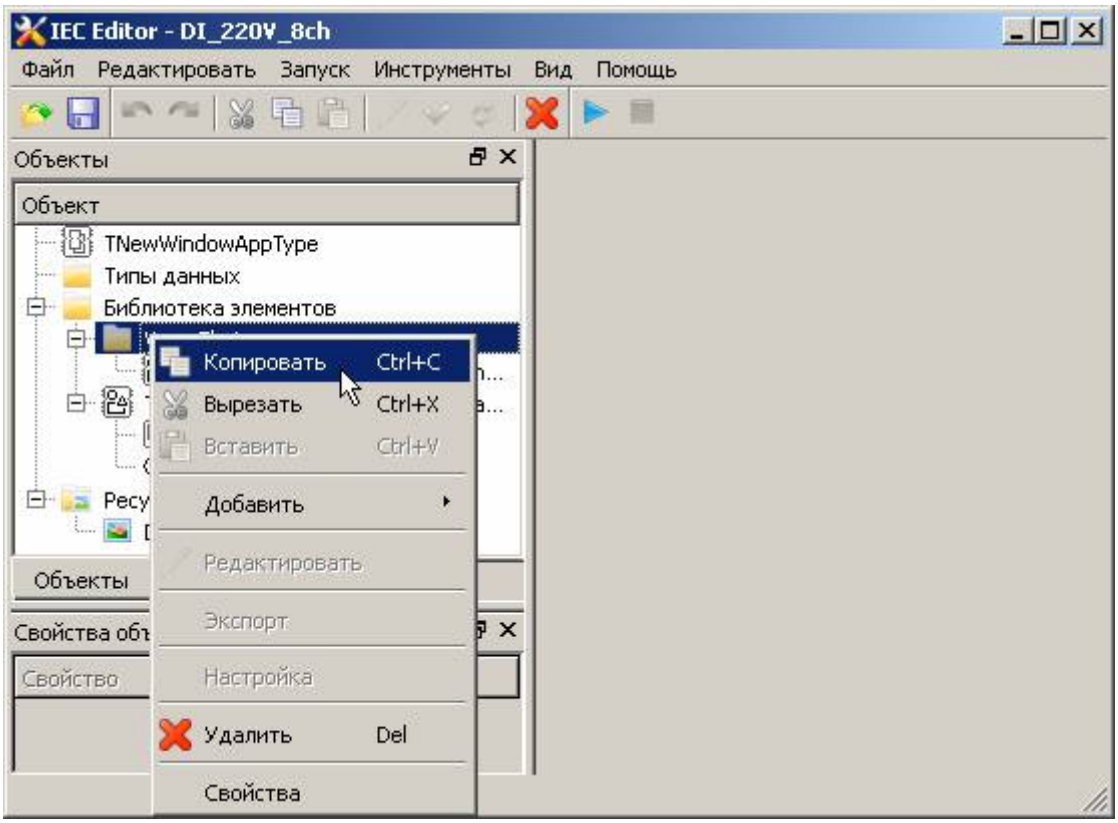


Рисунок 3.393 - Контекстное меню вызова команд Копировать и Вырезать

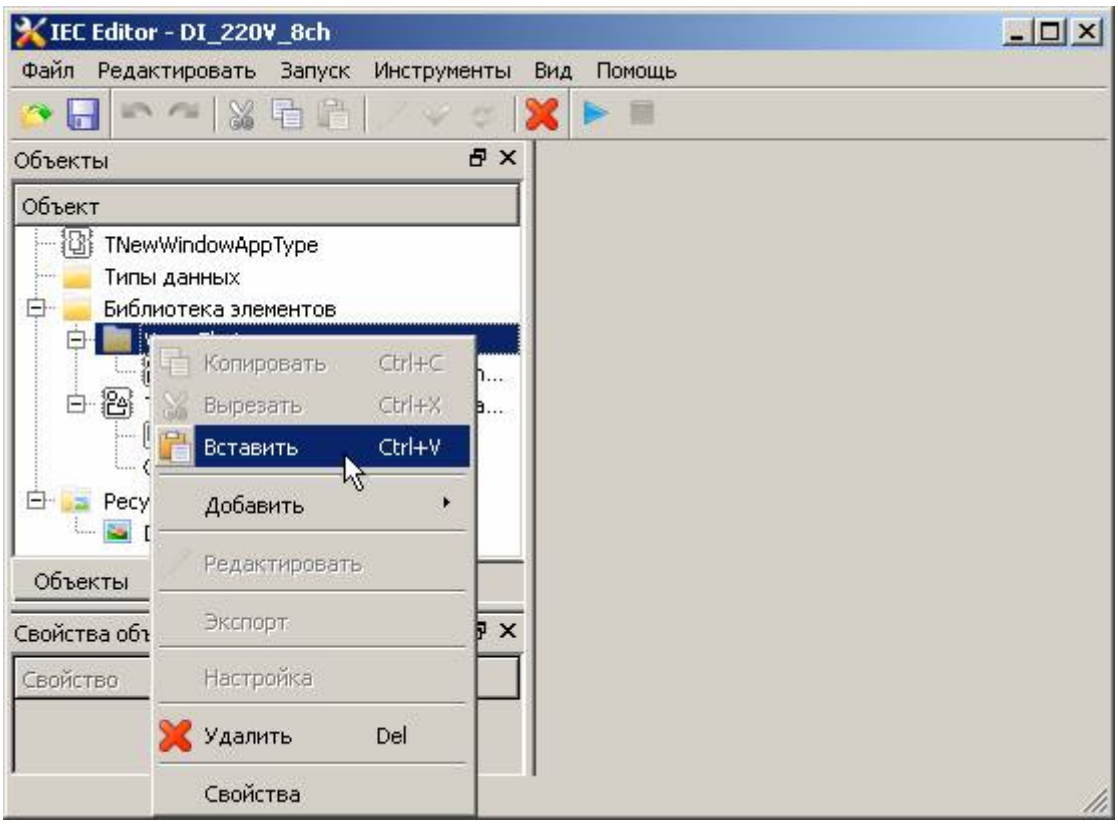


Рисунок 3.394 - Контекстное меню вызова команды Вставить

3.4.8.1.3. Окно библиотеки типов функциональных блоков

Окно дерева библиотеки типов функциональных блоков расположено в левой части главного окна редактора и открывается при нажатии на вкладку Библиотека. Дерево содержит иерархически организованную библиотеку типов встроенных функциональных блоков, а также функциональных блоков, загруженных из внешних подключаемых модулей (см. рис. 3.395). Окно библиотеки типов функциональных блоков формируется автоматически.

Узлы дерева представляют собой узлы типов функциональных блоков и узлы папок, служащих для организации библиотеки типов.

Блоки из библиотеки добавляются в типы композитных (как графических, так и нет) функциональных блоков при помощи «перетаскивания» «мышью» узлов дерева библиотеки, соответствующих типам функциональных блоков, в окно диаграммы композитного функционального блока, либо в окно мнемосхемы.

Описание типов, входящих в библиотеку типов функциональных блоков, приведено в документе SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33.

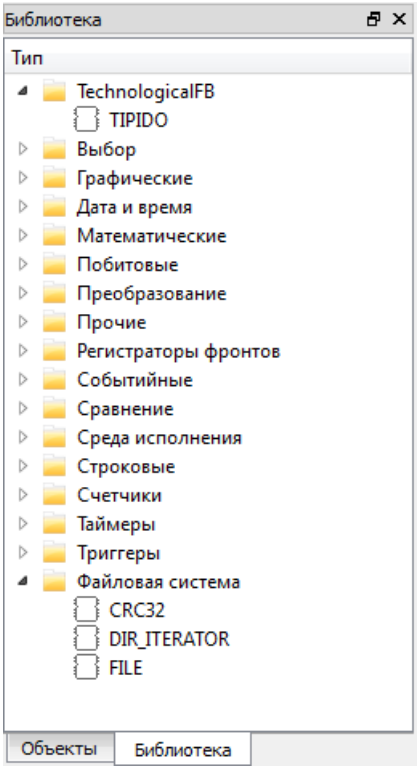


Рисунок 3.395 - Окно библиотеки типов функциональных блоков

3.4.8.1.4. Окно редактора свойств

Окно редактора свойств расположено в левой нижней части главного окна редактора. Оно служит для просмотра и изменения свойств объектов композитных типов функциональных блоков. Окно редактора свойств показано на рис. 3.396.

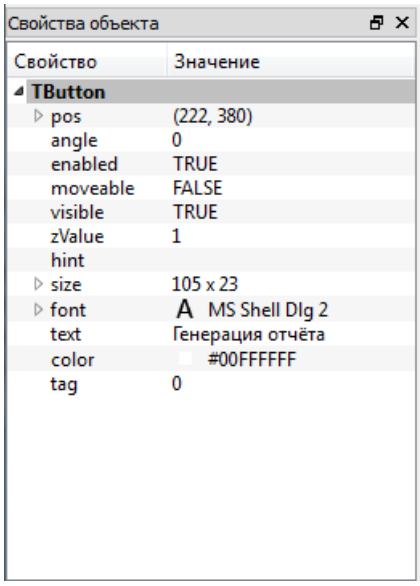


Рисунок 3.396 - Окно редактора свойств

Окно редактора свойств включает в себя строки следующих типов:

- строки, определяющие группу свойств (**TButton** на рис. 3.396);
- строки, определяющие свойства структурного типа данных (**pos**, **size**, **font** на рис. 3.396);
- переменные, значения которых могут быть изменены в редакторе свойств.

Для редактирования свойства необходимо выделить «мышью» строку переменной, значение которой необходимо изменить, после чего открыть редактор двойным щелчком «мыши» в колонке **Значение** либо нажатием на клавишу **< F2>**. После завершения редактирования значения переменной необходимо нажать на клавишу **<Enter >**.

3.4.8.1.5. Окно списка сообщений

Окно списка сообщений расположено в нижней части главного окна редактора и выводится при проверке типа функционального блока, а также при открытии типа, если в ходе загрузки данных были обнаружены какие-либо ошибки. Окно списка сообщений приведено на рис. 3.397.

Рисунок 3.397 - Окно списка сообщений

На панели инструментов окна **Сообщения** располагаются следующие кнопки:

 ("Сохранить") - сохранить сообщения в файл;

 ("Очистить") - очистить все сообщения в окне.

Строки окна списка сообщений помечаются иконками разного цвета. Цвет зависит от категории сообщения:

(зеленый) – информационное сообщение, не влияет на работоспособность типа приложения;

(желтый) – предупреждение, на сообщения данного типа необходимо обращать внимание, поскольку они могут содержать сообщения о некритических ошибках – тип функционального блока с такими ошибками работоспособен;

(красный) – критические ошибки – тип функционального блока не работоспособен.

На панели инструментов находятся кнопки, с помощью которых можно фильтровать сообщения в окне:

- оставить информационные сообщения;
- оставить предупреждения;
- оставить критические ошибки.

3.4.8.1.6. Окно списка событий

Окно списка событий расположено в нижней части главного окна редактора и выводится при запуске отладки типа приложения. Окно списка событий приведено на рис. 3.398.

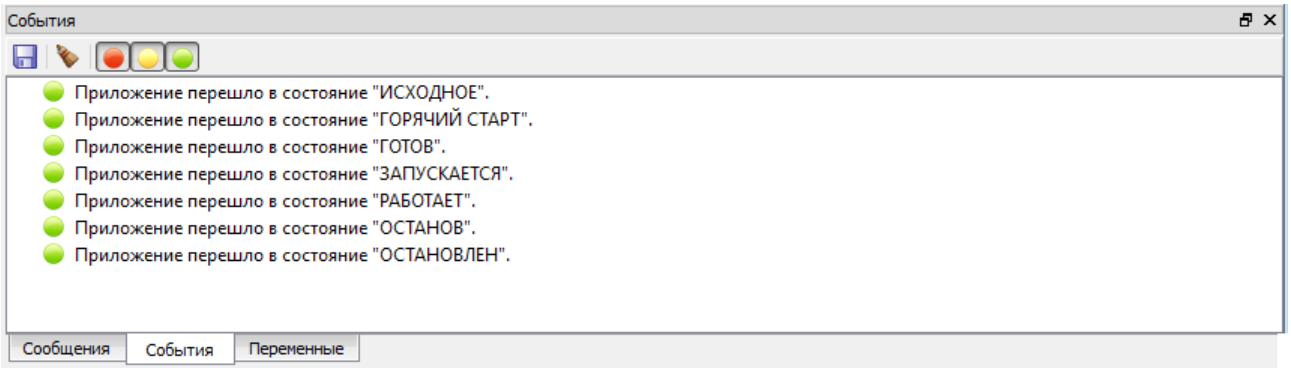


Рисунок 3.398 - Окно списка событий

На панели инструментов окна **События** располагаются следующие кнопки:

("Сохранить") - сохранить события в файл;

("Очистить") - очистить все события в окне.

Строки окна списка событий помечаются иконками разного цвета. Цвет зависит от категории события:

(зеленый) – информационное событие;

(желтый) – предупреждение;

(красный) – фатальное событие.

На панели инструментов находятся кнопки, с помощью которых можно фильтровать события в окне:

- оставить информационные события;
- оставить предупреждения;
- оставить фатальные события.

События можно разделить на две группы:

- события, формируемые самой системой;

- события, формируемые разработанными приложениями.

3.4.8.1.7. Окно переменных отладчика

Окно переменных отладчика расположено в нижней части главного окна редактора и выводится при запуске отладки типа приложения. Окно содержит список переменных, значения которых необходимо отслеживать и изменять в ходе отладки. Список переменных формируется оператором. Окно переменных отладчика приведено на рис. 3.399.

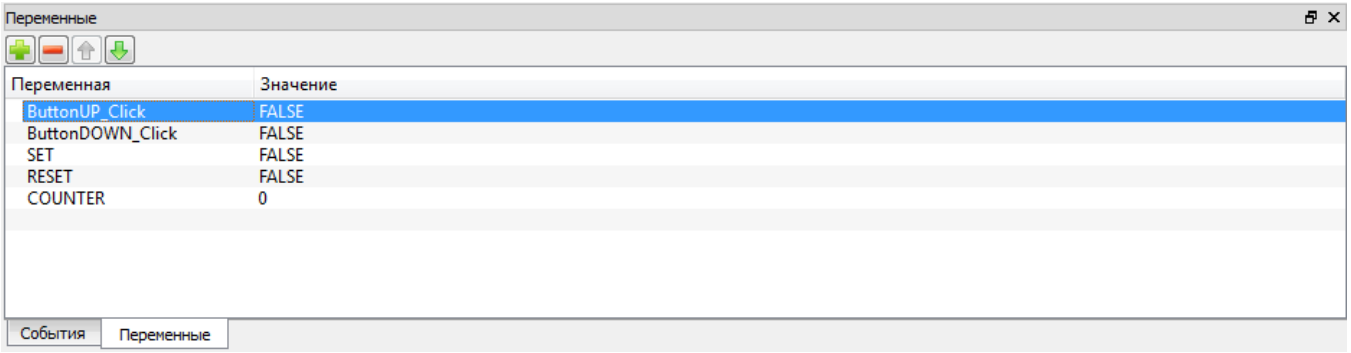


Рисунок 3.399 - Окно переменных отладчика

Для добавления переменной в список необходимо нажать на кнопку  в панели инструментов окна переменных отладчика. Редактор выведет окно добавления переменной (см. рис. 3.400). Диалог содержит древовидный список, в который выведены переменные и объекты, к которым принадлежат эти переменные. На верхнем уровне списка, помимо объектов верхнего уровня, расположены интерфейсные сигналы приложения. Оператор должен выделить одну или несколько переменных, после чего нажать на кнопку  . Редактор добавит выбранные переменные в список и закроет окно добавления переменных.

Для удаления переменной необходимо выделить ее строку в списке, после чего нажать на кнопку  в панели инструментов переменных отладчика. Редактор удалит выбранную строку из списка.

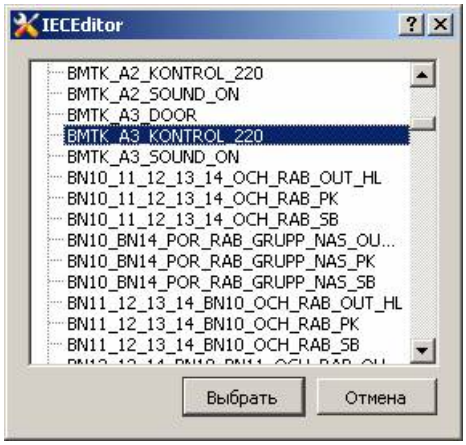


Рисунок 3.400 - Диалог добавления переменных

Порядок следования переменных в списке может быть изменен. Для перемещения переменной вверх необходимо выделить строку переменной, после чего нажать на кнопку  в

панели инструментов переменных отладчика. Аналогично, кнопка  перемещает выделенную переменную на одну строку вниз.

Для редактирования значения необходимо выделить «мышью» строку переменной, после чего открыть редактор двойным щелчком «мыши» в колонке **Значение** либо нажатием на клавишу **<F2>**. После завершения редактирования значения переменной необходимо нажать на клавишу **<Enter>**.

3.4.8.1.8. Окно редакторов типов функциональных блоков

Окно редакторов типов функциональных блоков расположено в центральной части главного окна редактора и представляет собой множество страниц редакторов типов функциональных блоков. Внешний вид страницы редактора типа зависит от того, какой именно тип открыт на редактирование:

- базовый (см. раздел 3.4.8.1.8.1);
- композитный (см. раздел 3.4.8.1.8.2);
- графический композитный (см. раздел 3.4.8.1.8.2);
- тип страницы главного окна ();
- тип окна (служит для построения графических элементов, представляющих собой вторичные видеокадры – небольшие окна, которые не обязательно должны быть видны оператору в течение всего времени его работы).

На рис. 3.401 открыты несколько окон редактирования типов функциональных блоков, но окна **TNewWindow**, **TNewPage** и **ButtonMenu** свёрнуты, а окно **MainPage** открыто.

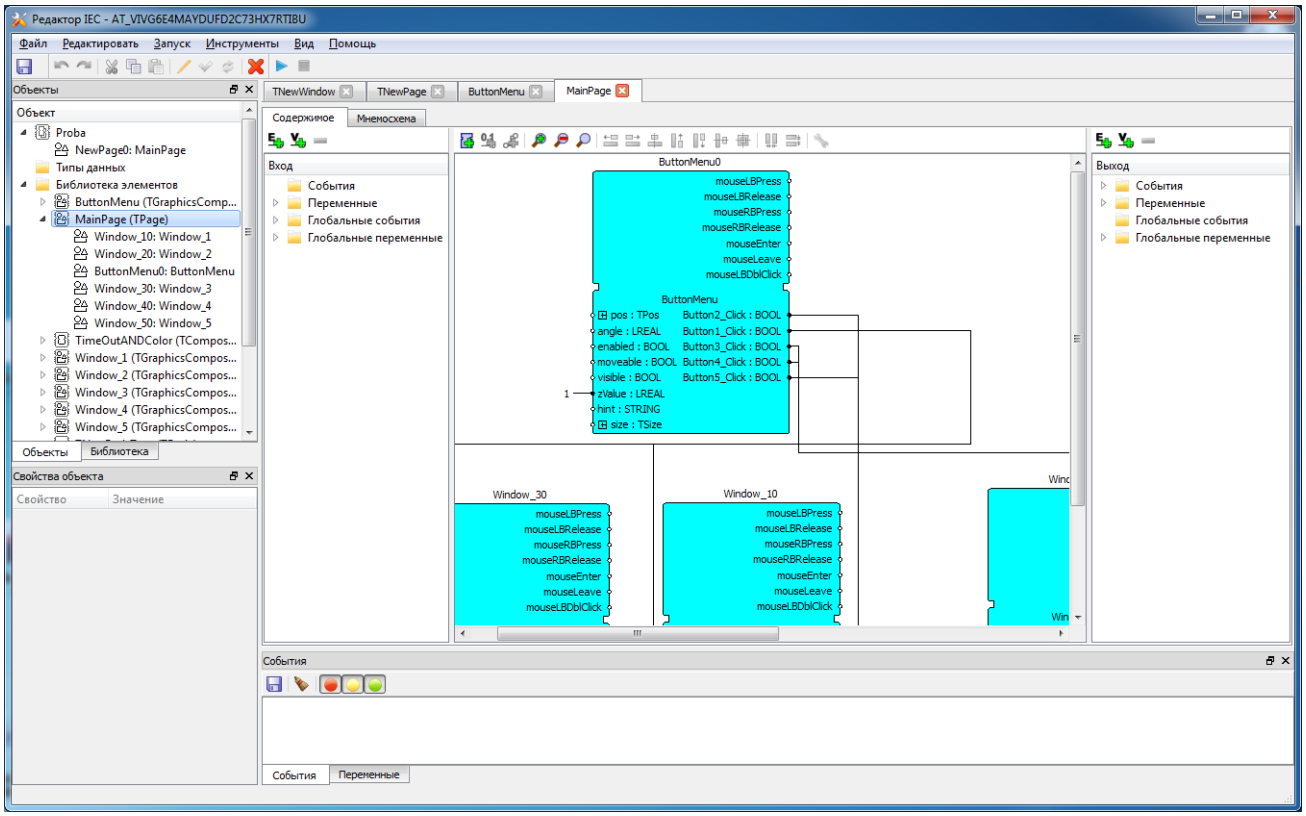


Рисунок 3.401 - Работа в редакторе IECEditor

Для просмотра типа функционального блока необходимо выбрать нужный тип в дереве объектов, после чего выбрать пункт меню **Редактировать** в группе команд **Редактировать**, либо нажать на кнопку  в панели инструментов, либо выбрать пункт контекстного меню

Редактировать в окне дерева объектов, либо выполнив двойной щелчок "мышкой" по элементу. Если данный тип функционального блока был открыт ранее, то программа активирует его редактор, если нет – программа создаст страницы для просмотра и редактирования выбранного типа функционального блока.

После редактирования типа функционального блока можно внести изменения в библиотеку типов типа приложения (нажав кнопку), либо отказаться от внесения изменений (нажав кнопку).

Редактор типа функционального блока закрывается путем нажатия на кнопку , расположенную на закладке с именем типа функционального блока.

3.4.8.1.8.1. Редактор базового типа функционального блока

Базовый тип функционального блока применяется для создания функционального блока, поведение которого можно описать с помощью алгоритмов, написанных на различных языках программирования (в данный момент поддерживаются только алгоритмы на языке ST, в котором нельзя использовать функционал, связанный с задержками по времени, - это блоки таймеров и такие блоки, как PID и др, где используется входной параметр CYCLE, т.е. временной интервал между вызовами функционального блока).

Для создания базового типа функционального блока необходимо вызвать контекстное меню в дереве объектов и выбрать пункты **Добавить** - **Базовый тип** (см. рис. 3.402).

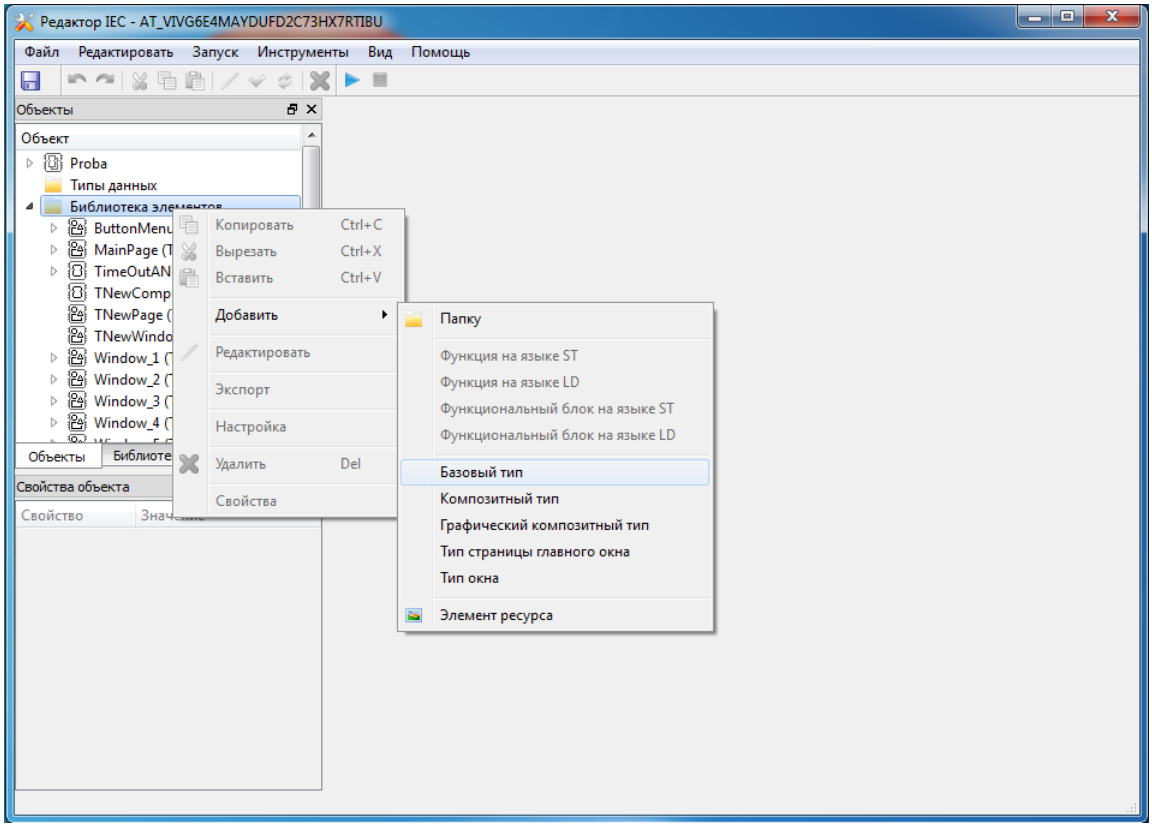


Рисунок 3.402 - Контекстное меню для создания Базового типа

В дереве объектов появится новый объект **TNewBasicTypeN(TBasic)**, где N - число от 0 и далее, **TNewBasicTypeN** - имя, **(TBasic)** - тип (см. рис. 3.403). Об изменении свойств типа функционального блока, таких как имя и другие, смотри раздел 3.4.8.1.2.2.

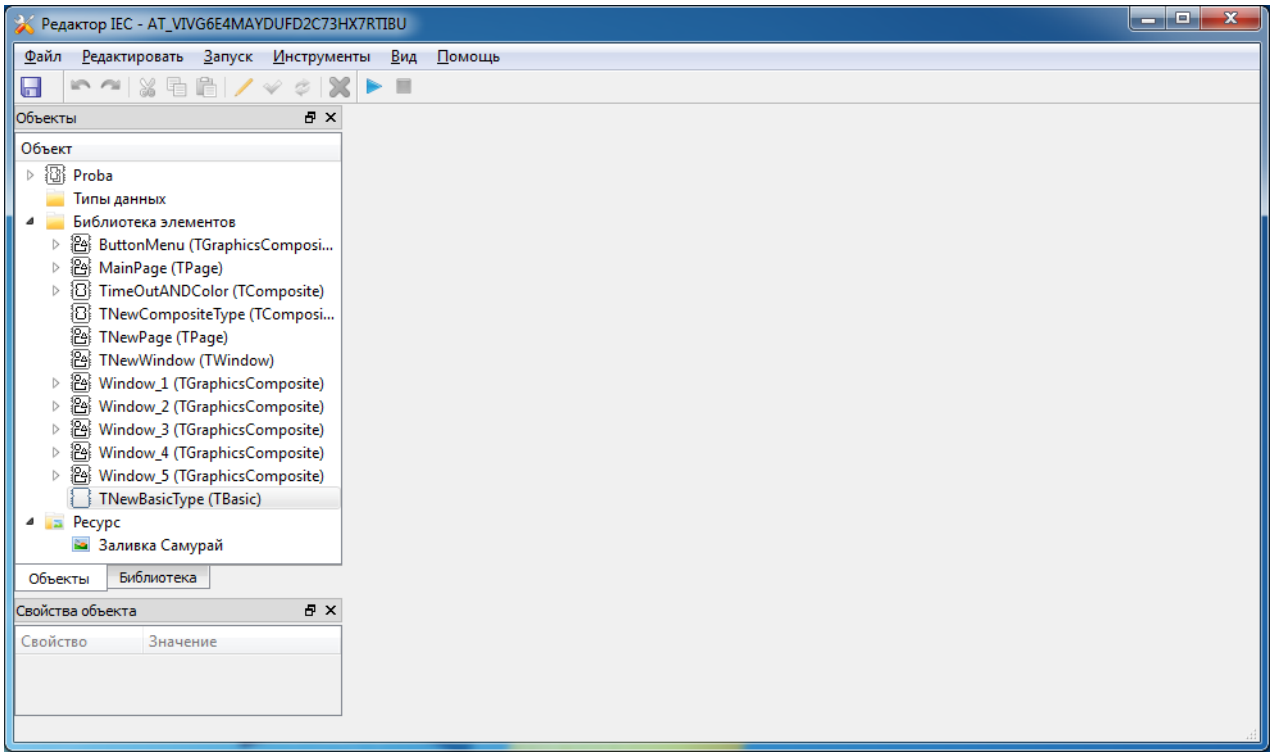


Рисунок 3.403 - Объект Базового типа

Чтобы работать с базовым типом функционального блока, необходимо вызвать его редактор (см. раздел 3.4.8.1.8 [367]).

Редактор базового типа функционального блока состоит из следующих страниц:

- страница редактирования интерфейса (см. раздел 3.4.8.1.8.1.1);
- страница редактирования алгоритмов (см. раздел 3.4.8.1.8.1.2);
- страница редактирования поведения (см. раздел 3.4.8.1.8.1.3).

3.4.8.1.8.1.1. Страница редактирования интерфейса

Страница редактирования интерфейса содержит элементы для редактирования списков событий и переменных функционального блока. Внешний вид страницы приведен на рис. 3.404.

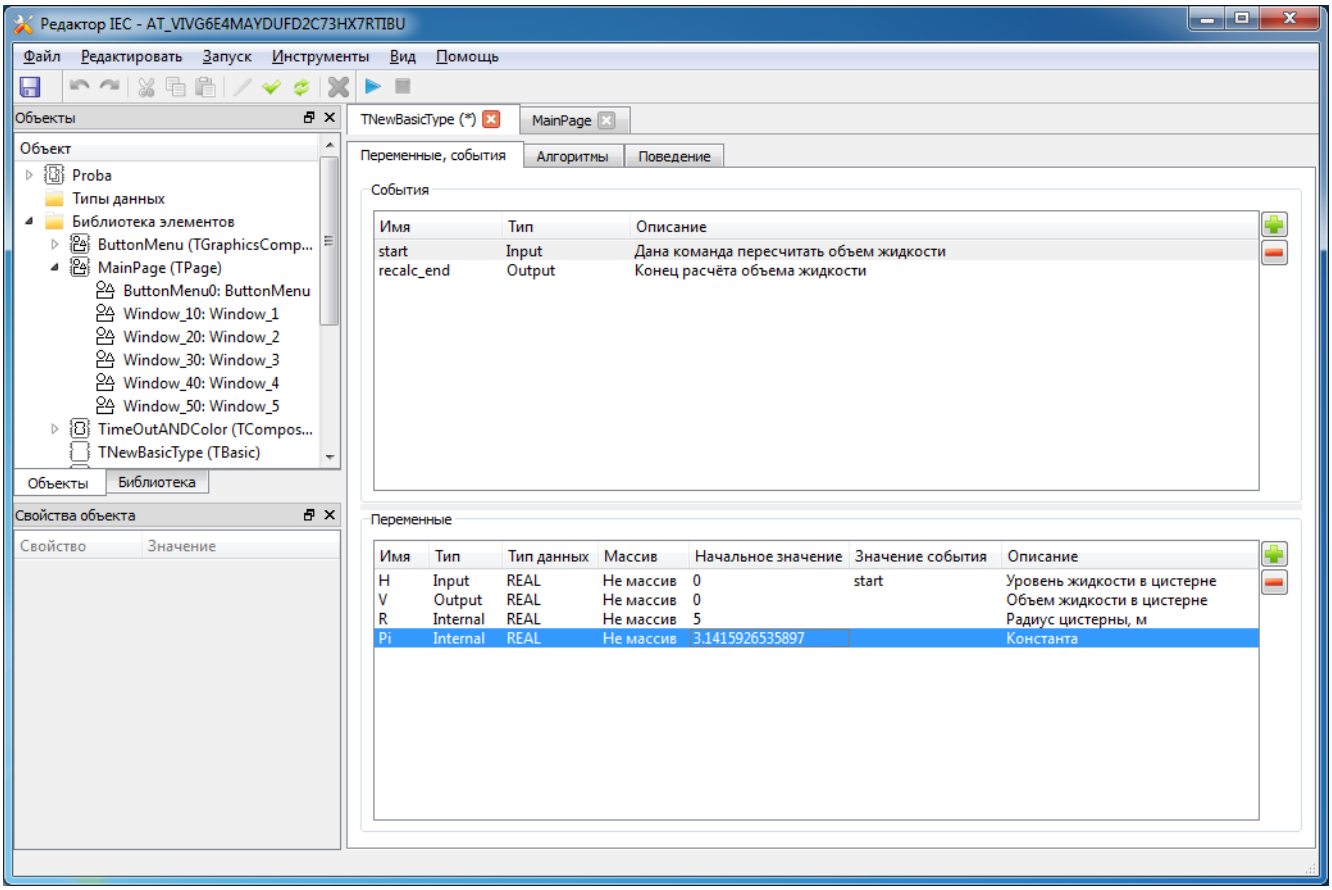


Рисунок 3.404 - Внешний вид страницы редактирования интерфейса Базового типа функционального блока

В верхней части страницы расположен список событий, содержащий следующие колонки:

- **Имя** – имя события, которое должно быть уникальным;
- **Тип** – вход или выход (**Input** - входное событие или **Output** - выходное событие);
- **Описание** – комментарий для описания события.

Для добавления события необходимо нажать на кнопку , расположенную справа от списка событий – редактор добавит в список событий новое входное событие. Имя события будет сгенерировано автоматически.

Для удаления события необходимо выделить нужную строку и нажать на кнопку , расположенную справа от списка событий.

Для редактирования события (изменения **Имени**, **Типа** или **Описания**) необходимо выделить «мышью» строку события в таблице, затем открыть редактор двойным щелчком «мыши» в нужной колонке, либо переместить курсор в нужную колонку и нажать на клавишу **<F2>**. После завершения редактирования события необходимо нажать на клавишу **<Enter>**.

На раздел 3.4.8.1.8.1.1 добавлены два события: **start** - входное событие и **recalc_end** - выходное событие. В столбце **Описание** даны пояснения их функций.

В нижней части страницы расположен список переменных, содержащий следующие колонки:

- **Имя** – имя переменной, которое должно быть уникальным;
- **Тип** – вход, выход или внутренняя переменная (**Input** - входная переменная, **Output** - выходная переменная или **Internal** - внутренняя или локальная переменная);
- **Тип данных** – тип данных переменной (см. Приложение А);
- **Массив** – признак того, является ли переменная массивом; если переменная – массив, то в данной колонке также задается размер массива;
- **Начальное значение** – значение, которым инициализируется переменная;

- **Значение события** – событие, которое возникает при изменении значения входной переменной блока (может быть задано только для входной переменной);

- **Описание** – комментарий для описания переменной.

Для добавления переменной необходимо нажать на кнопку  , расположенную справа от списка переменных. Программа добавит в список переменных новую входную переменную типа **BOOL** с начальным значением FALSE. Имя переменной будет сгенерировано автоматически.

Для удаления переменной необходимо нажать на кнопку  , расположенную справа от списка переменных.

Для редактирования переменной (изменения **Имени**, **Типа** и т.д.) необходимо выделить «мышью» строку переменной в таблице, затем открыть редактор двойным щелчком «мыши» в нужной колонке, либо переместить курсор в нужную колонку и нажать на клавишу **<F2>**. После завершения редактирования переменной необходимо нажать на клавишу **<Enter>**.

На рис. 3.404 в список добавлены следующие переменные:

- **H** - входная переменная типа REAL, означающая уровень жидкости в цистерне, по изменению значения этой переменной генерируется событие **start** (пересчитать объём жидкости);

- **V** - выходная переменная типа REAL, означающая рассчитанный объём жидкости;

- **R** - внутренняя переменная типа REAL, в данном случае означающая радиус цистерны, также задано начальное значение 5;

- **Pi** - внутренняя переменная типа REAL, в данном случае означающая константу с заданным начальным значением 3.1415926535897.

3.4.8.1.8.1.2. Страница редактирования алгоритмов базового функционального блока

На странице редактирования алгоритмов базового функционального блока производится редактирование алгоритмов, которые могут быть вызваны при переходе базового блока в какое-либо состояние. Внешний вид страницы приведен на рис. 3.405.

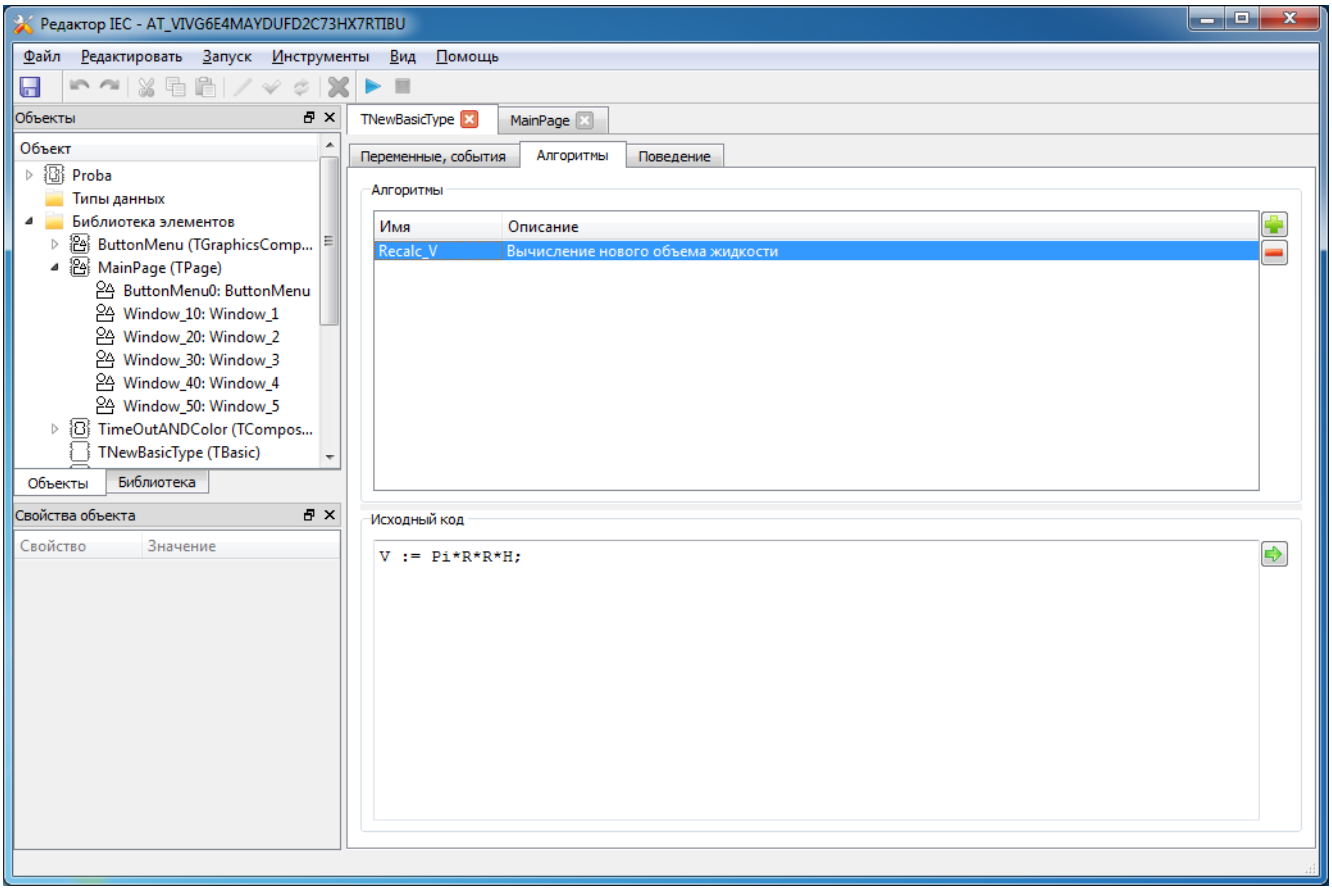


Рисунок 3.405 - Страница редактирования алгоритмов базового функционального блока

В верхней части страницы расположен список алгоритмов, содержащий следующие колонки:
- **Имя** – имя алгоритма (должно быть уникальным);
- **Описание** – комментарий алгоритма.

Для добавления алгоритма необходимо нажать на кнопку , расположенную справа от списка алгоритмов. Программа добавит в список алгоритмов новую строку. Имя алгоритма будет сгенерировано автоматически.

Для удаления алгоритма необходимо нажать на кнопку , расположенную справа от списка алгоритмов.

Для изменения **Имени** или **Описания** алгоритма необходимо выделить «мышью» строку алгоритма в таблице, а затем открыть редактор двойным щелчком «мыши» в нужной колонке, либо переместить курсор в нужную колонку и нажать на клавишу **<F2>**. После завершения редактирования алгоритма необходимо нажать на клавишу **<Enter>**.

На рис. 3.405 в список алгоритмов добавлен алгоритм с именем **Recalc_V** и предназначенным для вычисления объема жидкости, о чём написано в столбце **Описание**.

В нижней части страницы расположено поле редактирования исходного кода алгоритмов. В поле редактирования выводится исходный код алгоритма, выбранного в списке. Исходный код представляет собой текст, написанный на языке ST. Краткое описание языка ST приведено в документе SCADA-система "СОНАТА" Руководство программиста КУНИ.505200.023-01.01 33.

Для введения кода следует выбрать алгоритм в списке и переместить курсор в окно редактирования кода.

На рис. 3.405 в поле редактирования исходного кода добавлена формула вычисления объёма жидкости в цилиндре, написанная по синтаксису языка ST.

Кнопка , расположенная справа от поля редактирования исходного кода, предназначена для проверки синтаксиса алгоритма.

По нажатию на кнопку  будет выведено диалоговое окно с сообщением о правильности синтаксиса языка (см. рис. 3.406) или об ошибке в синтаксисе языка (см. рис. 3.407).

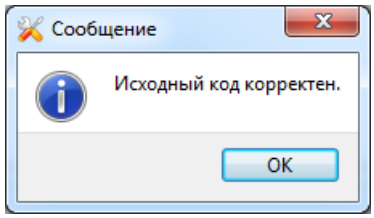


Рисунок 3.406 - Диалоговое окно, сообщающее о верном синтаксисе языка

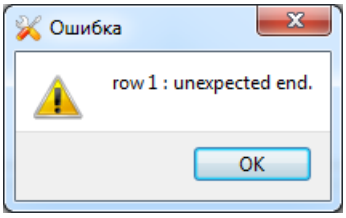


Рисунок 3.407 - Диалоговое окно, сообщающее об ошибке в синтаксисе языка

3.4.8.1.8.1.3. Страница редактирования поведения базового функционального блока

Страница редактирования поведения предназначена для формирования диаграммы состояний базового функционального блока. Внешний вид страницы приведен на рис. 3.408.

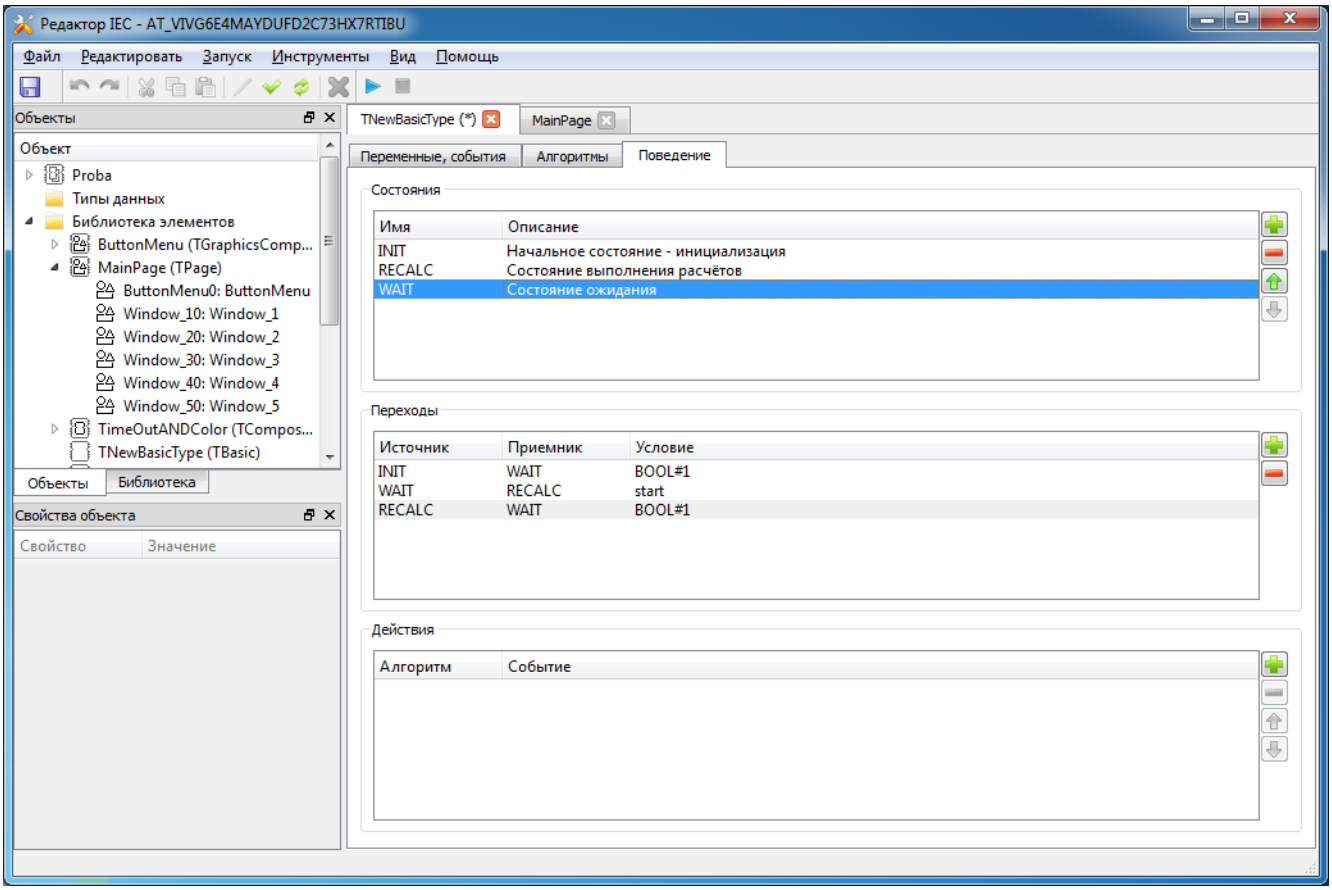


Рисунок 3.408 - Страница редактирования поведения базового функционального блока

В верхней части страницы расположен список состояний, содержащий следующие колонки:

- **Имя** – имя состояния (должно быть уникальным);

- **Описание** – комментарий описания состояния.

Для добавления состояния необходимо нажать на кнопку , расположенную справа от списка состояний. Программа добавит в список состояний новую строку. Имя состояния будет сгенерировано автоматически.

Для удаления состояния необходимо выделить строку и нажать на кнопку , расположенную справа от списка состояний. Выделенная строка будет удалена.

Кнопки и служат для изменения порядка следования состояний. При нажатии на первую из них выбранное состояние сдвигается вверх, при нажатии на вторую – вниз. Первое состояние в списке является начальным. Блок переходит в данное состояние при инициализации приложения. Действия при этом не выполняются.

ВНИМАНИЕ! Базовый блок начнет свою работу с состояния, которое находится первым в списке. А дальнейшую работу вы уже сможете настроить, так что последовательность состояний ниже первого не важна.

Для изменения **Имени** или **Описания** состояния необходимо выделить «мышью» строку алгоритма в таблице, а затем открыть редактор двойным щелчком «мыши» в нужной колонке, либо переместить курсор в нужную колонку и нажать на клавишу **<F2>**. После завершения редактирования состояния необходимо нажать на клавишу **<Enter>**.

На рис. 3.408 в список состояний добавлены три состояния и выполнено редактирование полей **Имя** и **Описание**:

- **INIT** - начальное состояние;

- **RECALC** - состояние выполнения расчётов;

- **WAIT** - состояние ожидания.

В средней части страницы расположен список переходов между состояниями, содержащий следующие колонки:

- **Источник** – исходное состояние;

- **Приемник** – конечное состояние;

- **Условие** – условие перехода, представляющее собой логическое выражение, если значение данного выражения будет равно TRUE, то блок перейдет из исходного состояния в конечное.

Для добавления перехода необходимо нажать на кнопку , расположенную справа от списка переходов. Программа добавит новый переход в список. Затем необходимо указать начальное и конечное состояния и задать условие перехода. При добавлении новой строки в качестве условия задается строка **BOOL#1**, представляющая собой безусловный переход.

Для удаления перехода необходимо выделить строку и нажать на кнопку , расположенную справа от списка переходов. Выделенная строка будет удалена.

На рис. 3.408 в список переходов добавлены следующие строки:

- переход из состояния **INIT** в состояние **WAIT** по безусловному переходу, т.е. всегда;

- переход из состояния **WAIT** в состояние **RECALC** по появлению события **start**;

- переход из состояния **RECALC** в состояние **WAIT** по безусловному переходу, т.е. всегда.

В нижней части страницы расположен список действий, в котором задаются алгоритмы, выполняемые при приходе функционального блока в указанное состояние, а также возникающие при этом выходные события. Действия выполняются в том порядке, в котором они заданы в данном списке. В каждой строке может отсутствовать как алгоритм, так и событие.

Для добавления действия необходимо нажать на кнопку , расположенную справа от списка действий. Программа добавит новое действие в список. Затем необходимо указать алгоритм и событие действия.

Для удаления действия необходимо нажать на кнопку , расположенную справа от списка действий. Выделенная строка будет удалена.

Кнопки и служат для изменения порядка следования действий. При нажатии на первую из них выбранное действие сдвигается вверх, при нажатии на вторую – вниз.

На рис. 3.409 в поле **Состояния** выбрано состояние **RECALC**, для которого в поле действия добавлен алгоритм **Recalc_V**, выполняемый при переходе в это состояние, и добавлено выходное событие **recalc_end**, которое происходит по завершению алгоритма.

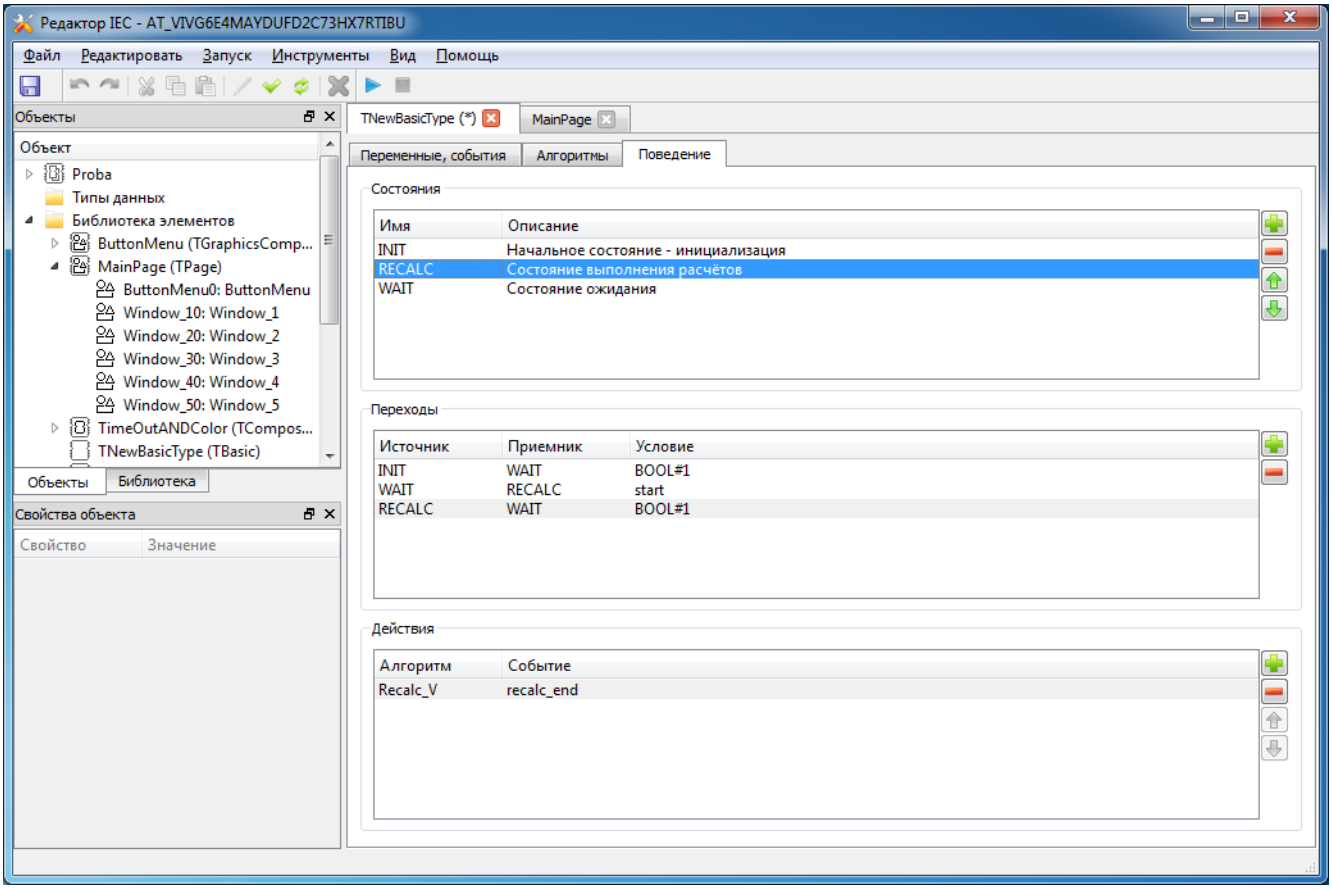


Рисунок 3.409 - Страница поведения базового функционального блока и настроенное поле Действий

На рис. 3.410 изображен внешний вид базового типа функционального блока, который получается, учитывая все наши операции по редактированию. На входе у данного функционального блока событие **start**, по которому будет выполняться алгоритм вычисления объёма, и входная переменная **H** - уровень жидкости, по изменению которой также будет выполняться алгоритм вычисления объёма, но это было настроено дополнительно. На выходе функционального блока событие **recalc_end**, сообщающее, что расчёт объема завершён, и переменная **V** - рассчитанный объём жидкости.

Рисунок 3.410 - Внешний вид описанного выше базового типа функционального блока

3.4.8.1.8.2. Редактирование композитных типов функциональных блоков

С точки зрения функциональности редактирования композитные типы функциональных блоков можно разделить на графические и неграфические. Редактор графического композитного типа функционального блока содержит страницы редактирования диаграммы и мнемосхемы, а редактор неграфического композитного типа – только страницу диаграммы.

Для создания композитного типа функционального блока необходимо вызвать контекстное меню в дереве объектов и выбрать пункты **Добавить** - **Композитный тип** (см. рис. 3.411). Аналогично для создания графического композитного типа функционального блока необходимо в контекстном меню выбрать пункты **Добавить - Графический композитный тип** (см. рис. 3.411).

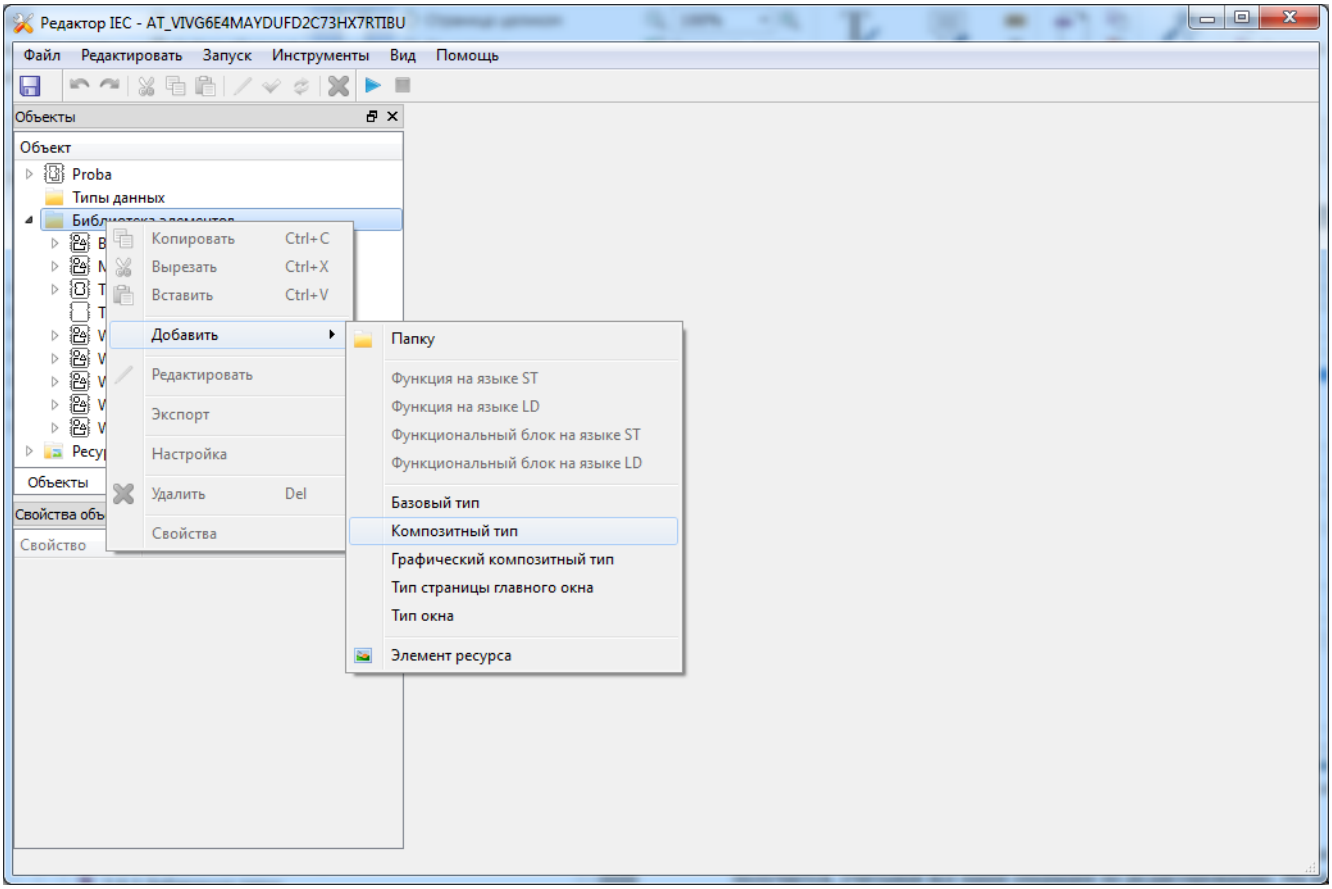


Рисунок 3.411 - Выбор композитного типа функционального блока в контекстном меню

На рис. 3.412 в дереве объектов создан новый композитный тип функционального блока. По умолчанию его имя создаётся по шаблону **TNewCompositeTypeN(TComposite)**, где **N** - число от 0 и далее, **TNewCompositeTypeN** - имя, **(TComposite)** - тип. Об изменении свойств типа функционального блока, таких, как имя и другие, смотри раздел 3.4.8.1.2.2.

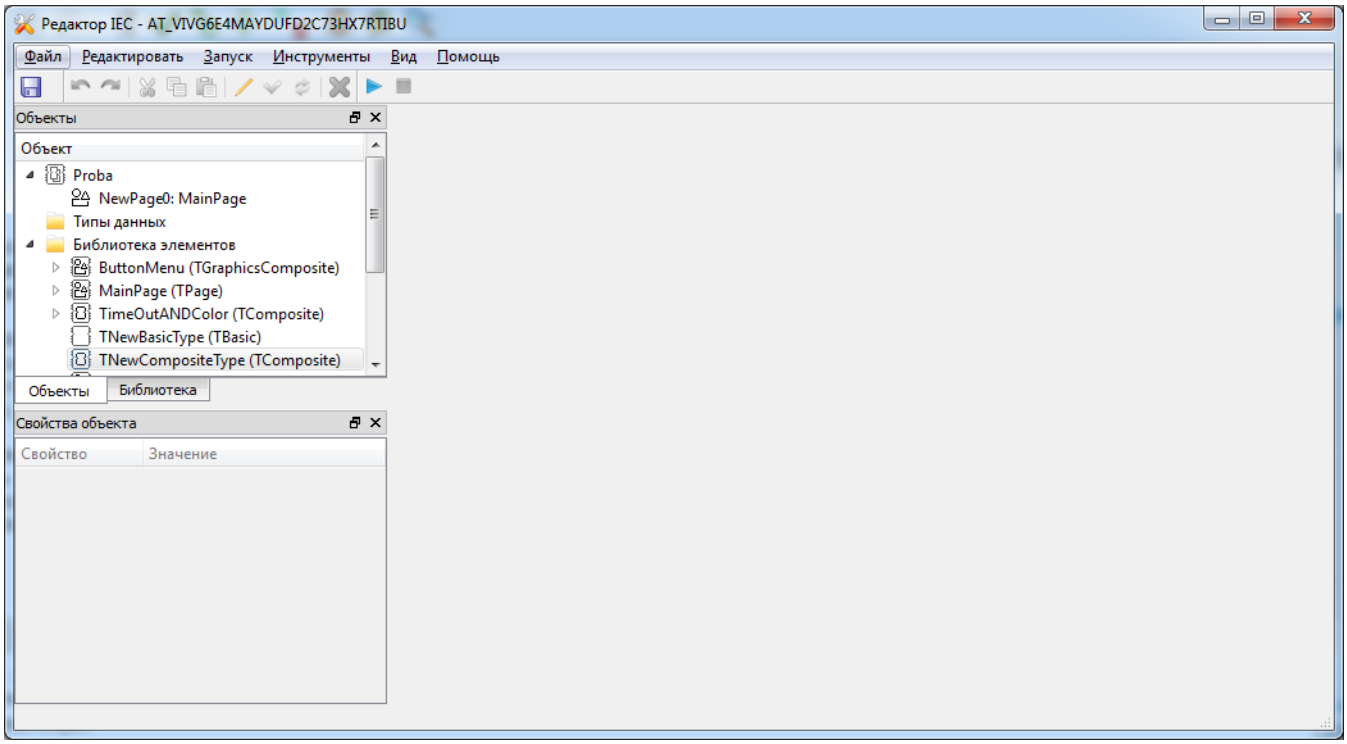


Рисунок 3.412 - Композитный блок в дереве объектов

На рис. 3.413 в дереве объектов создан новый графический композитный тип функционального блока. По умолчанию его имя создаётся по шаблону **TNewGraphicsCompositeTypeN(TGraphicsComposite)**, где **N** - число от 0 и далее, **TNewGraphicsCompositeTypeN** - имя, **(TGraphicsComposite)** - тип. Об изменении свойств типа функционального блока, таких, как имя и другие, смотри раздел 3.4.8.1.2.2.

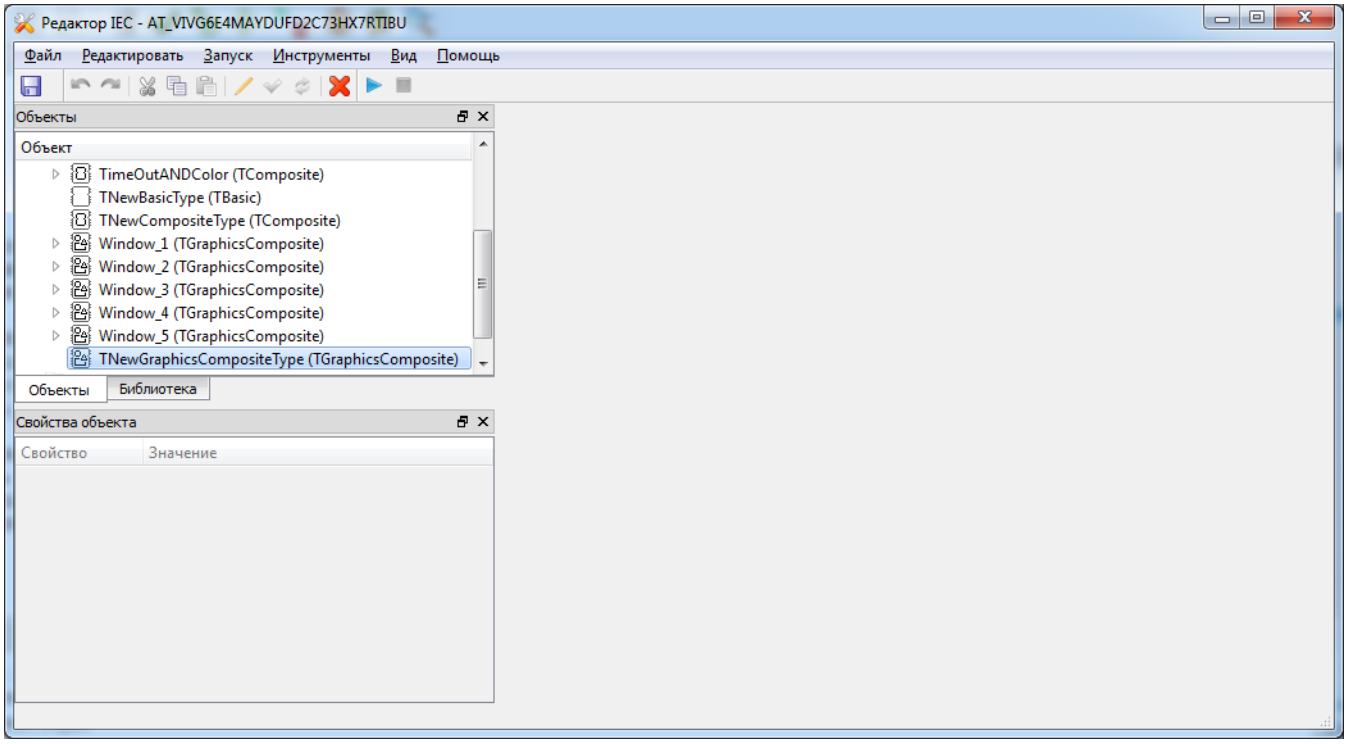


Рисунок 3.413 - Графический композитный блок в дереве объектов

Чтобы работать с композитными типами функциональных блоков, необходимо вызвать их редактор (см. раздел 3.4.8.1.8 [367]).

Композитный тип функционального блока состоит из страницы редактирования диаграммы (см. раздел 3.4.8.1.8.2.1). На рис. 3.414 для редактирования открыт композитный тип, состоящий из нескольких окон под общим названием **Содержимое**.

Названия окон слева направо:

- **Вход** (окно входных данных);

- **Окно диаграммы** (основное);

- **Выход** (окно выходных данных).

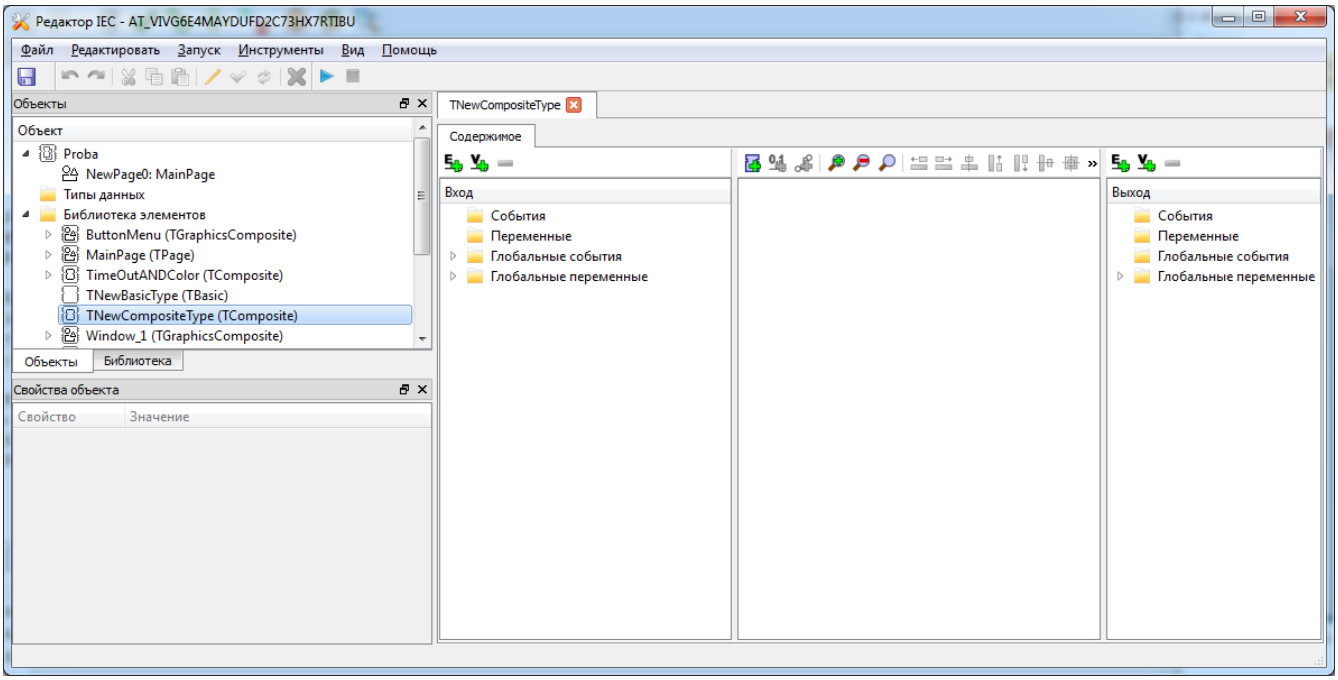


Рисунок 3.414 - Страница редактирования композитного типа

Графический композитный тип функционального блока состоит из страницы редактирования диаграммы (см. раздел 3.4.8.1.8.2.1) и страницы редактирования мнемосхемы. На рис. 3.415 для редактирования открыт графический композитный тип, состоящий из нескольких окон под общим названием **Содержимое** - вкладка **Содержимое**. Данная вкладка аналогична описанной странице **Содержимое** для композитного типа.

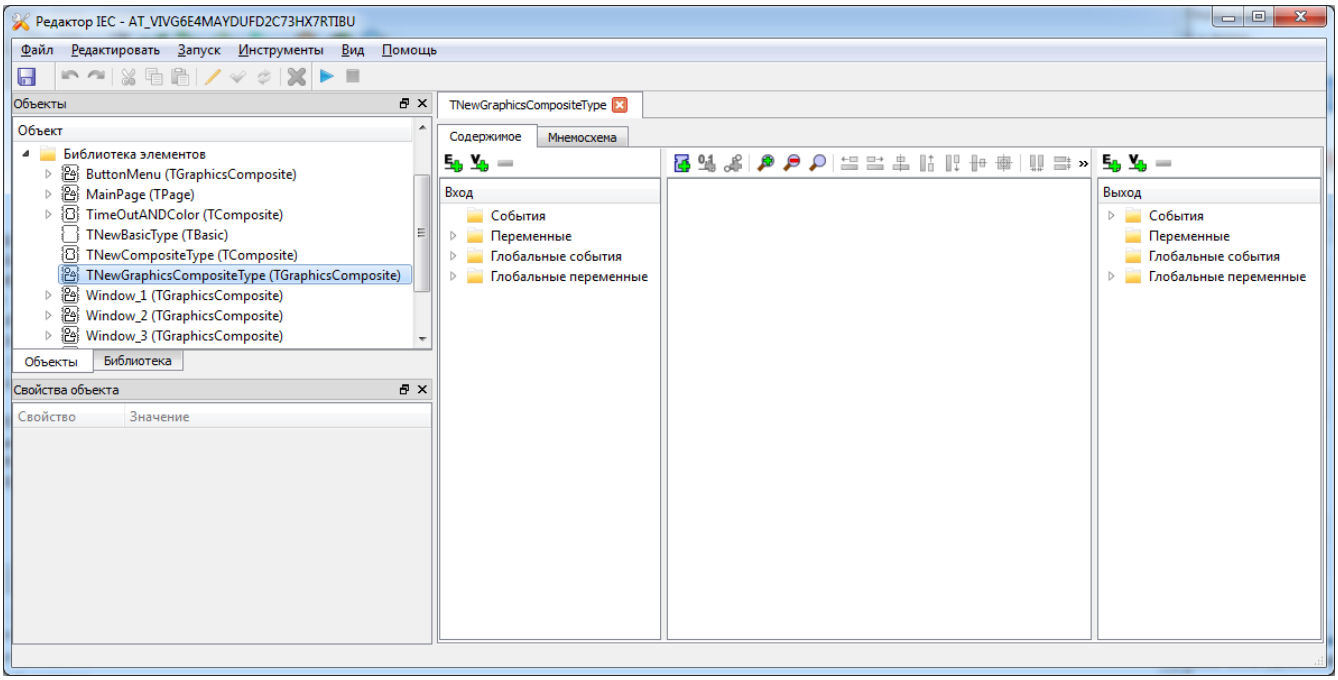


Рисунок 3.415 - Страница редактирования графического композитного типа

На рис. 3.416 изображено окно редактирования мнемосхемы для графического композитного типа.

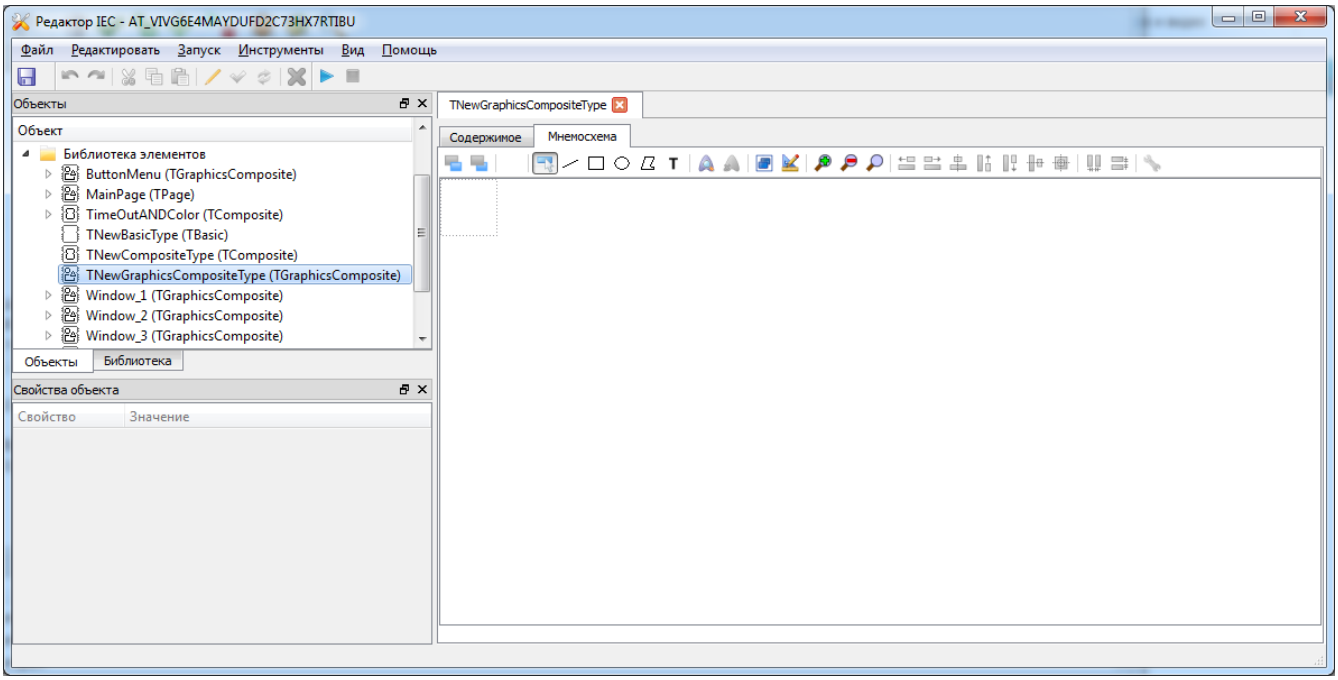


Рисунок 3.416 - Вкладка Мнемосхема графического композитного типа

3.4.8.1.8.2.1. Страница редактирования диаграммы

Страница редактирования диаграммы содержит элементы, необходимые для редактирования интерфейса функционального блока, а также набора входящих в него объектов и связей между ними. Страница диаграммы приведена на рис. 3.417.

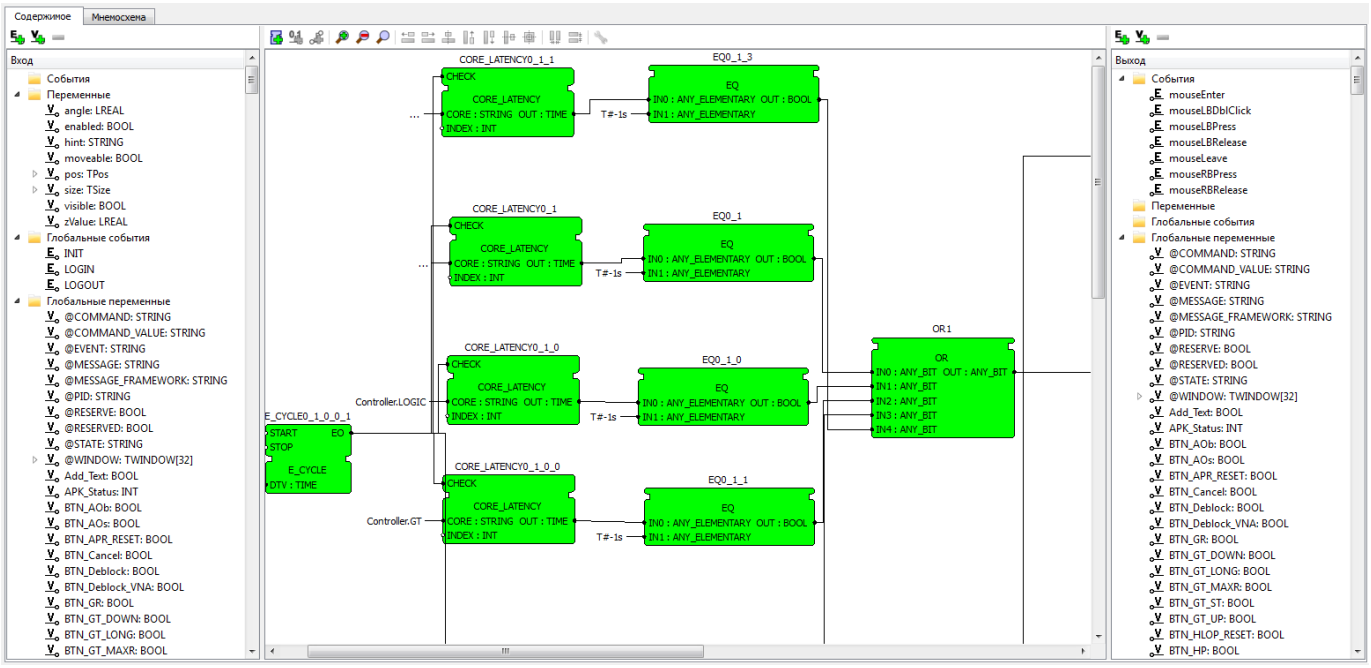


Рисунок 3.417 - Страница редактирования диаграммы функционального блока

В левой и правой частях страницы расположены списки входных и выходных элементов интерфейса:

- события (локальные для функционального блока);

- переменные (локальные для функционального блока);

- глобальные события (задаются в интерфейсе приложения и видны для всех функциональных блоков);

- глобальные переменные (задаются в интерфейсе приложения и видны для всех функциональных блоков).

В средней части страницы расположено окно редактирования диаграммы.

В списке событий и переменных оператор может добавлять, изменять и удалять интерфейсные элементы. События и переменные в списке отсортированы по алфавиту. Имена событий и переменных должны быть уникальными, то есть, не может быть двух событий или переменных с одинаковыми именами, причем, уникальность имен отслеживается среди обоих списков – входных и выходных элементов.

Для добавления события необходимо нажать на кнопку  , расположенную над соответствующим входным либо выходным списком. Программа добавит новый элемент в папку **«События»** выбранного списка. Имя события будет задано автоматически.

Для добавления переменной необходимо нажать на кнопку  , расположенную над соответствующим входным либо выходным списком. Программа добавит новый элемент в папку **«Переменные»** выбранного списка. Имя переменной будет задано автоматически.

Для удаления события либо переменной необходимо выделить ее в списке, после чего нажать на кнопку  , расположенную над списком. Программа удалит выделенный интерфейсный элемент.

Для изменения свойств события необходимо выделить нужный элемент событие в интерфейсе. В редакторе свойств объекта появятся следующие свойства события (см. рис. 3.418):

- **Name** – имя события;

- **Comment** – описание (комментарий) события.

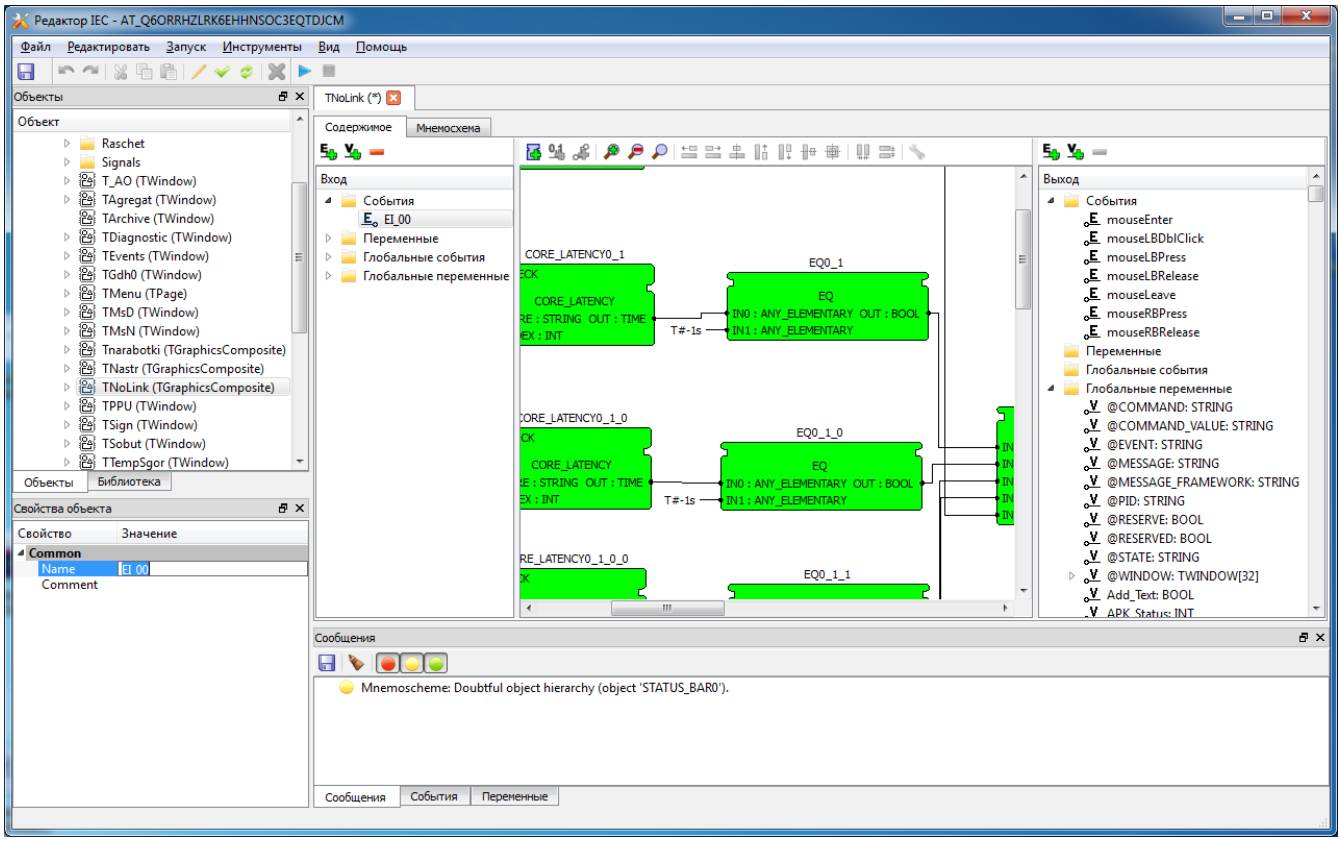


Рисунок 3.418 - Изменение свойств события

Для изменения свойств переменной необходимо выделить нужную переменную в интерфейсе. В редакторе свойств объекта появятся следующие свойства переменной (см. рис. 3.419):

- **Name** – имя события;

- **Type** – тип данных переменной;

- **Comment** – описание (комментарий) события;

- **Array** – признак того, является ли переменная массивом (FALSE - не является массивом, TRUE - является массивом);

- **Size of array** – размер массива, если переменная является массивом;

- **Initial value** – начальное значение переменной.

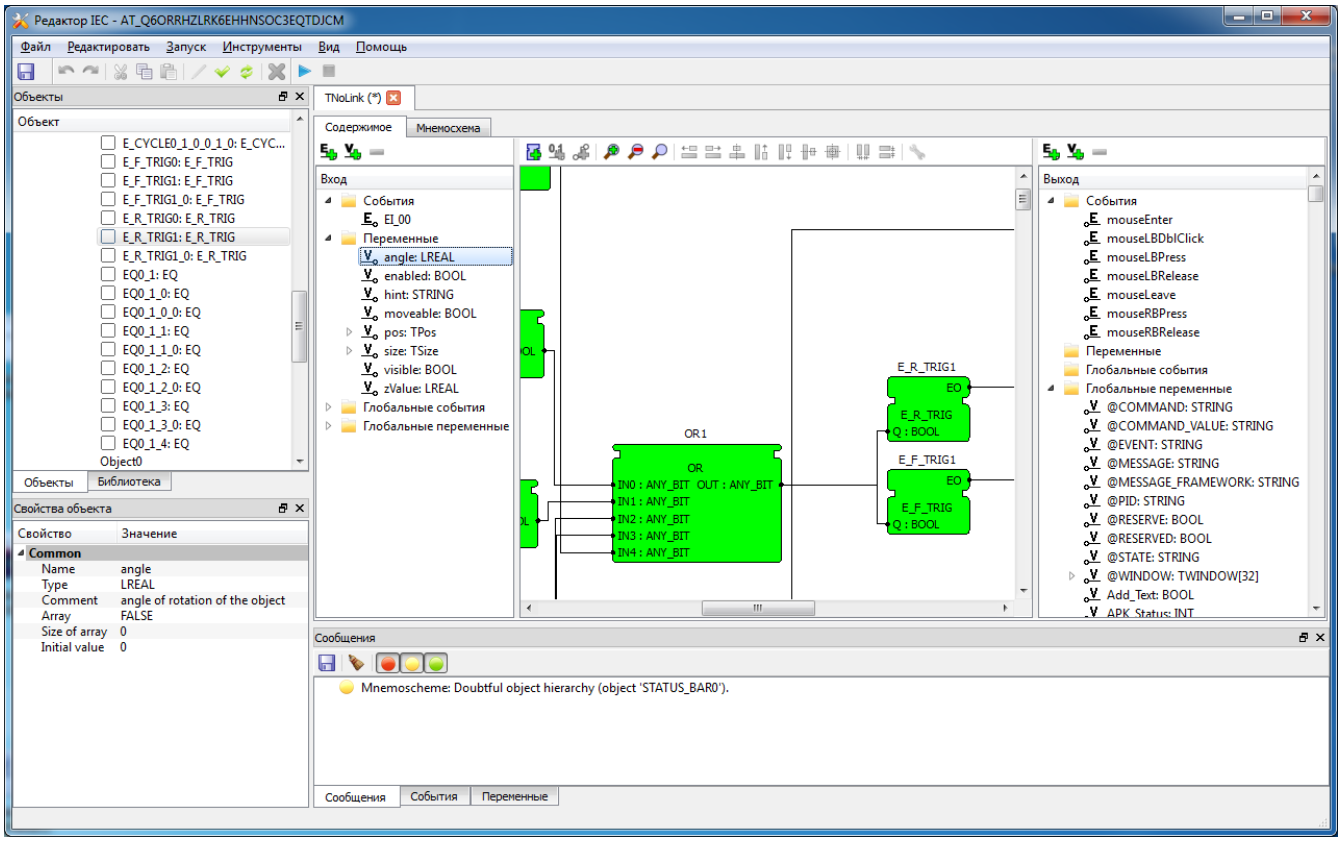


Рисунок 3.419 - Изменение свойств переменной

Редактирование диаграммы заключается в формировании набора функциональных блоков, а также в связывании их между собой и с событиями и переменными, входящими в интерфейс редактируемого типа функционального блока.

Добавление функционального блока на диаграмму осуществляется следующими способами:

- перетаскиванием типа функционального блока из дерева библиотеки или дерева объектов проекта;

- нажатием на кнопку , расположенную над полем редактирования диаграммы;

- добавлением графического элемента на мнемосхему;

- при вставке скопированных объектов.

Для удаления функционального блока или группы функциональных блоков необходимо выделить их на диаграмме, после чего выбрать пункт главного меню «Редактировать/Удалить», либо нажать на кнопку панели инструментов. Программа удалит выбранные блоки и связи между ними с редактируемой диаграммы.

Для изменения свойств функционального блока необходимо выделить его на диаграмме. При этом в редактор свойств будут выведены следующие характеристики (см. рис. 3.420):

- **Name** – имя объекта;

- **Type** – тип объекта;

- **Comment** – описание объекта;

- **Security** – группа безопасности объекта. Если в данной строке не задано значение, то выходные события объекта формируются для всех пользователей. Если в данной строке задано значение и оно совпадает с правами на объект у пользователя (см. Права на проект ???), то выходные события объекта формируются для данного пользователя. Если не совпадает, то данный объект не формирует выходные события для данного пользователя;

- **Enabled** – флаг разрешения объекта. Используется для исключения из цепочки отдельных блоков диаграммы без их удаления;

- **X** – x-координата объекта на диаграмме;

- **Y** – y-координата объекта на диаграмме.

Далее следуют свойства графических объектов

- **pos** – позиция объекта на мнемосхеме;

- **angle** – угол поворота объекта на мнемосхеме;

- **enabled** – доступность обекта. Если enabled = False, то объект недоступен для взаимодействия;

- **moveable** – подвижность обекта. Возможность перемещения объекта на мнемосхеме;

- **z-value** – z-индекс объекта на мнемосхеме;

- **hint** – подсказка;

- **size** – размер объекта на мнемосхеме;

- **checkable** – залипание. Доступно для кнопок;

- **checkable** – залипание. Доступно для кнопок;

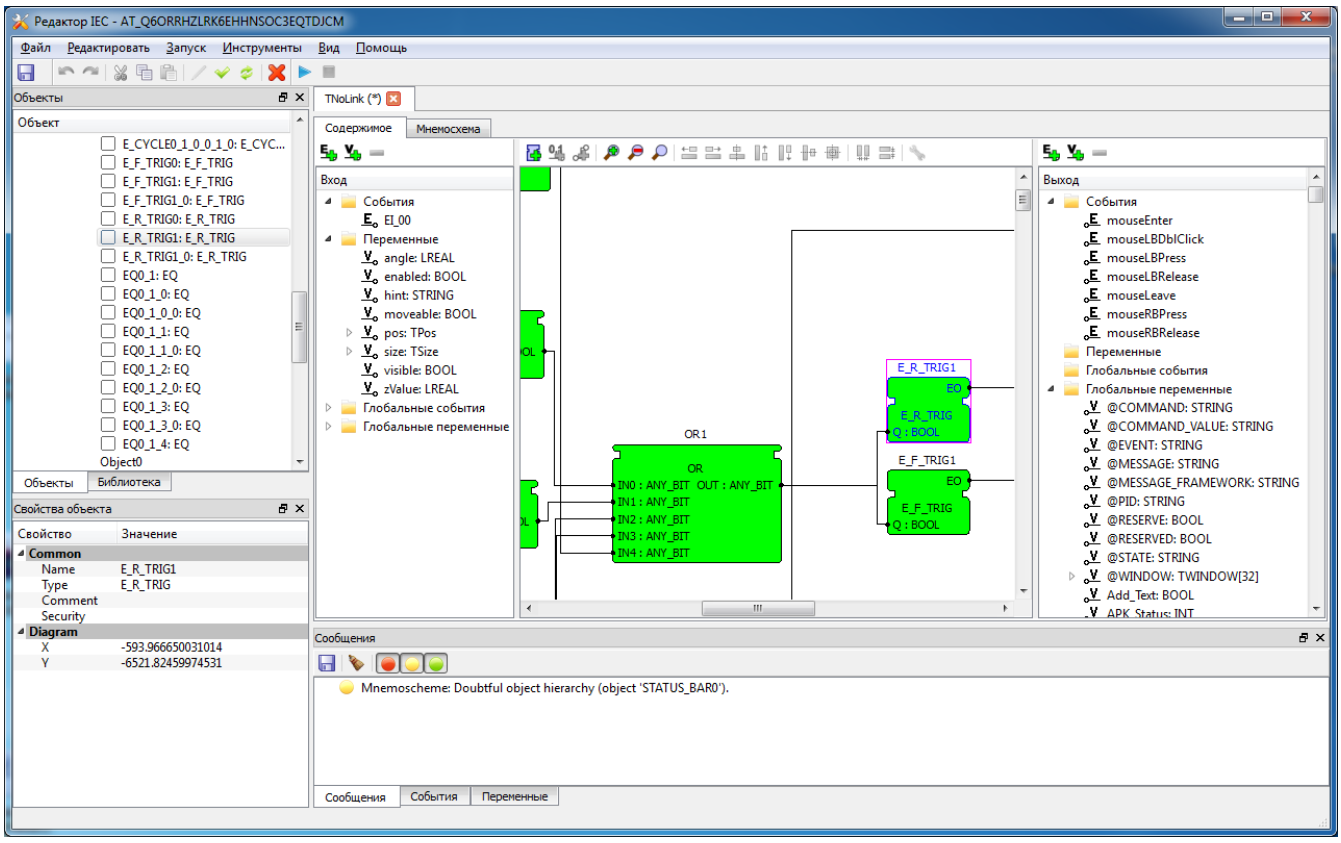


Рисунок 3.420 - Изменение свойств функционального блока

Для настройки состояния и взаимодействия объектов к интерфейсным событиям объектов могут быть привязаны события, входящие в интерфейс функционального блока, а также события других блоков. К переменным могут быть привязаны интерфейсные переменные редактируемого типа, переменные других блоков, а также константные значения.

Для привязки константного значения к переменной объекта необходимо выделить «мышью» узел на объекте , к которому производится подключение, после чего нажать на кнопку панели

инструментов диаграммы. Программа добавит константное значение . Для изменения данного значения или его типа необходимо выделить его на диаграмме, после чего изменить тип или значение в редакторе свойств объекта.

Любые связи между событиями или переменными объектов на диаграмме можно добавить, выделив узел у события или переменной объекта и нажав на кнопку . Программа выведет диалог добавления события или переменной (см. рис. 3.421). Необходимо выбрать интерфейсный элемент, после чего нажать на кнопку OK. Программа создаст требуемую связь.

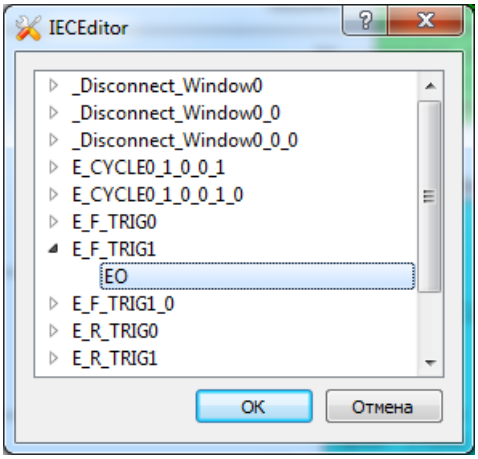


Рисунок 3.421 - Диалог добавления связи между переменными или событиями

Также можно создать связь с переменной или событием типа функционального блока путем «перетаскивания» требуемой переменной или события из интерфейса на поле диаграммы при помощи «мыши». Левую кнопку «мыши» необходимо отпустить тогда, когда станет возможным создание связи. Программа показывает данную возможность, окрашивая конец связи красным цветом (см. рис. 3.422).

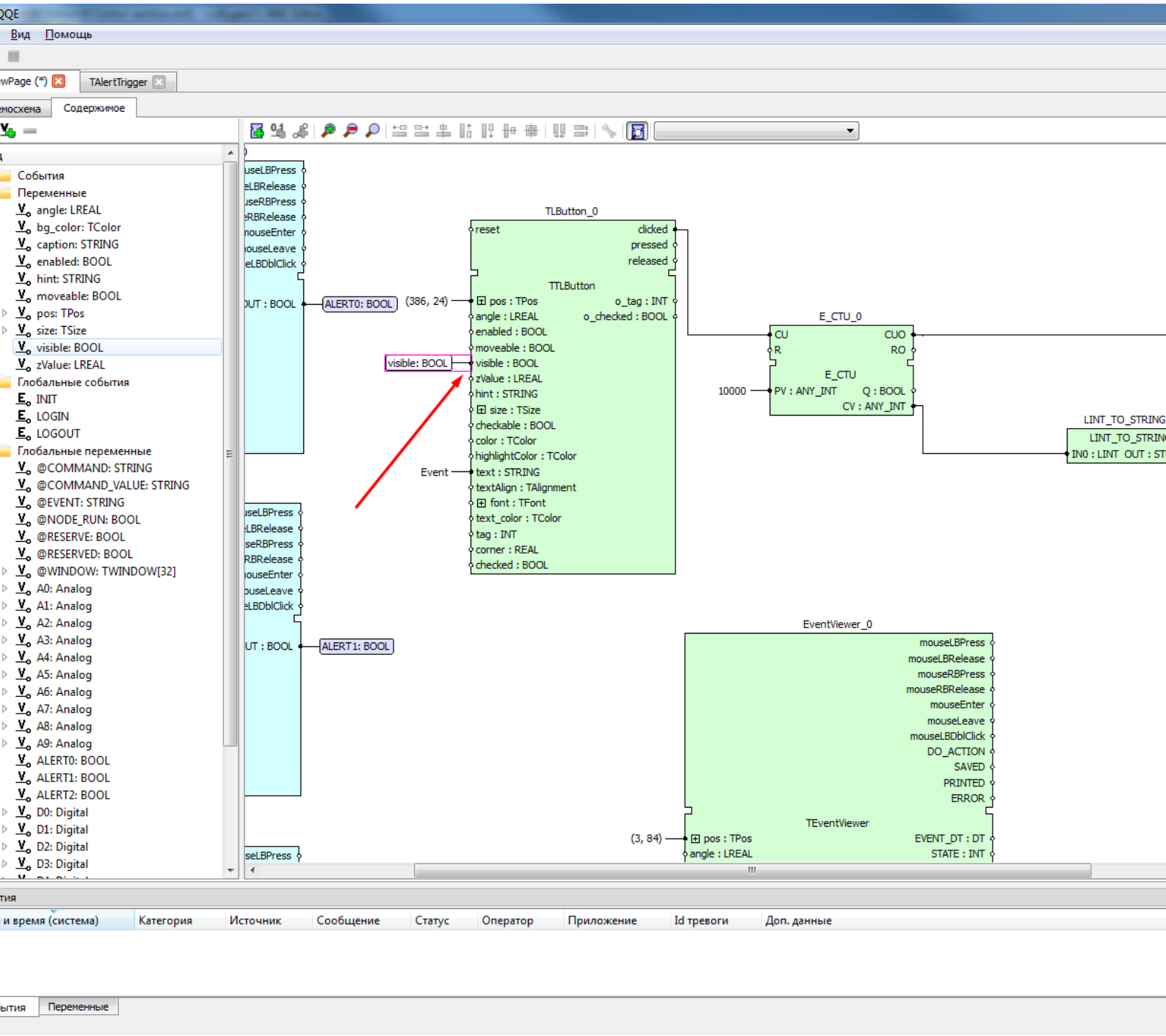


Рисунок 3.422 - Создание связи переменной из интерфейса с узлом блока на диаграмме

Связь между объектами также может быть создана при помощи мыши». Для этого необходимо выделить «мышью» один из концов будущей связи, после чего, удерживая нажатой левую кнопку «мыши», довести указатель «мыши» до второго конца связи. Затем левую кнопку «мыши» можно отпустить (см. рис. 3.423).

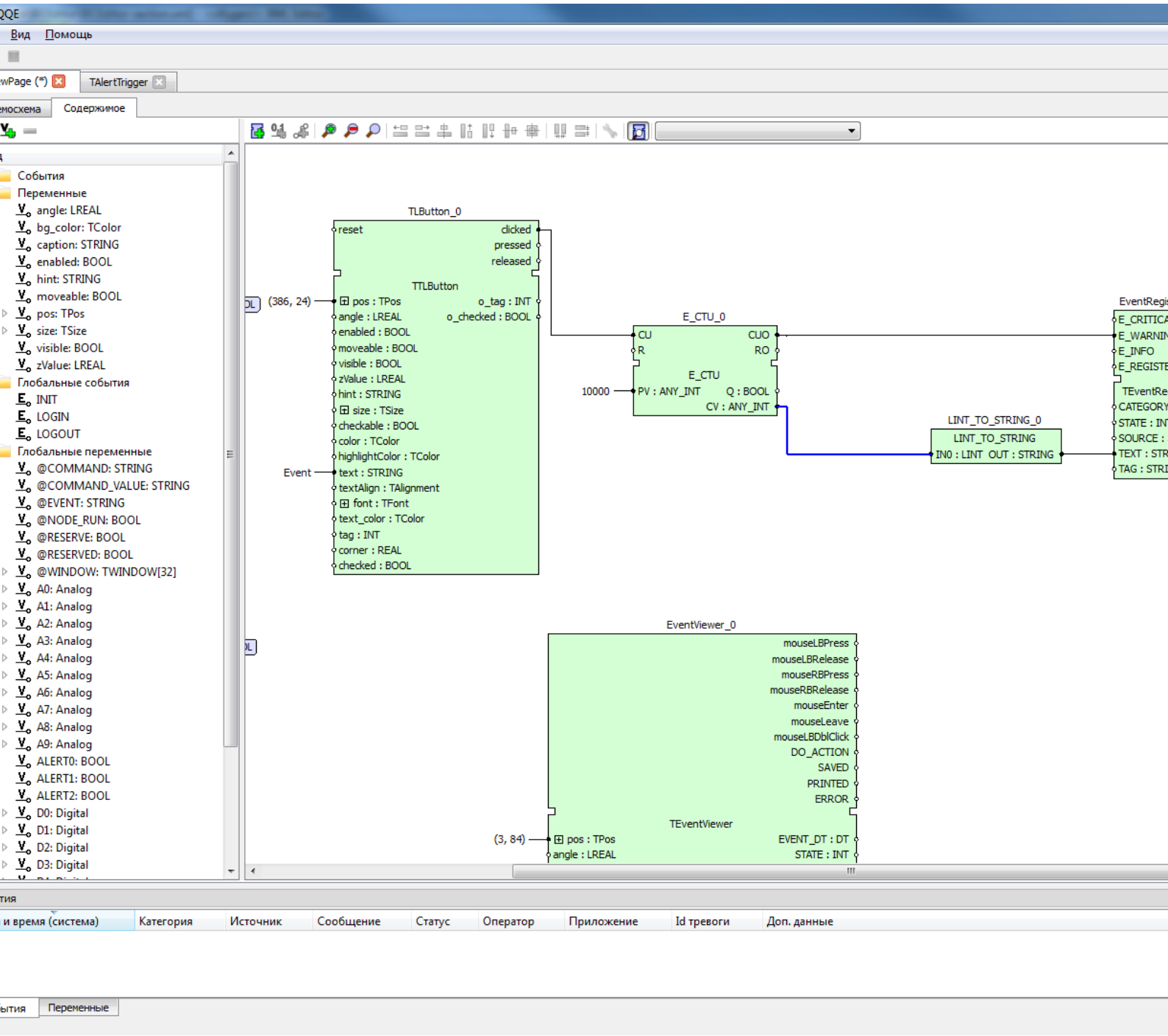


Рисунок 3.423 - Создание связей между переменными
и событиями функциональных блоков на диаграмме

Следует отметить, что у переменной или события может быть сколько угодно исходящих связей, но только одна входящая связь.

Другие элементы управления окном диаграммы:

("Увеличить") - увеличить масштаб в окне диаграммы;

("Уменьшить") - уменьшить масштаб в окне диаграммы;

("Нормальный размер") - масштаб возвращается к исходному для окна диаграммы;

("Выравнивание по левому краю") - выравнивание объектов в окне диаграммы по левому краю;

 ("Выравнивание по правому краю") - выравнивание объектов в окне диаграммы по правому краю;

 ("Выравнивание по центру по горизонтали") - выравнивание объектов в окне диаграммы по центру и расположение их по горизонтали;

 ("Выравнивание по верхнему краю") - выравнивание объектов в окне диаграммы по верхнему краю;

 ("Выравнивание по нижнему краю") - выравнивание объектов в окне диаграммы по нижнему краю;

 ("Выравнивание по центру по вертикали") - выравнивание объектов в окне диаграммы по центру и расположение их по вертикали;

 ("Выравнивание по центру") - выравнивание объектов в окне диаграммы по центру;

 ("Расставить объекты по горизонтали") - расположить объекты в окне диаграммы по горизонтали;

 ("Расставить объекты по вертикали") - расположить объекты в окне диаграммы по вертикали;

 ("Настроить объект") - доступные настройки для объекта.

 ("Показать типы переменных") - Влючить или отключить отображение типов переменных для всех объектов.

Выпадающий список всех объектов текущей диаграммы. Используется для переключения фокуса между объектами.

3.4.8.1.8.2.2. Страница редактирования мнемосхемы

Страница редактирования мнемосхемы предназначена для редактирования набора графических элементов, составляющих мнемосхемы, выводимые оператору. Страница редактирования мнемосхем приведена на рис. 3.424.

В верхней части страницы расположена панель инструментов мнемосхемы. В центральной части расположено поле редактирования мнемосхемы. Процесс редактирования мнемосхемы заключается в формировании набора графических элементов и изменении их свойств.

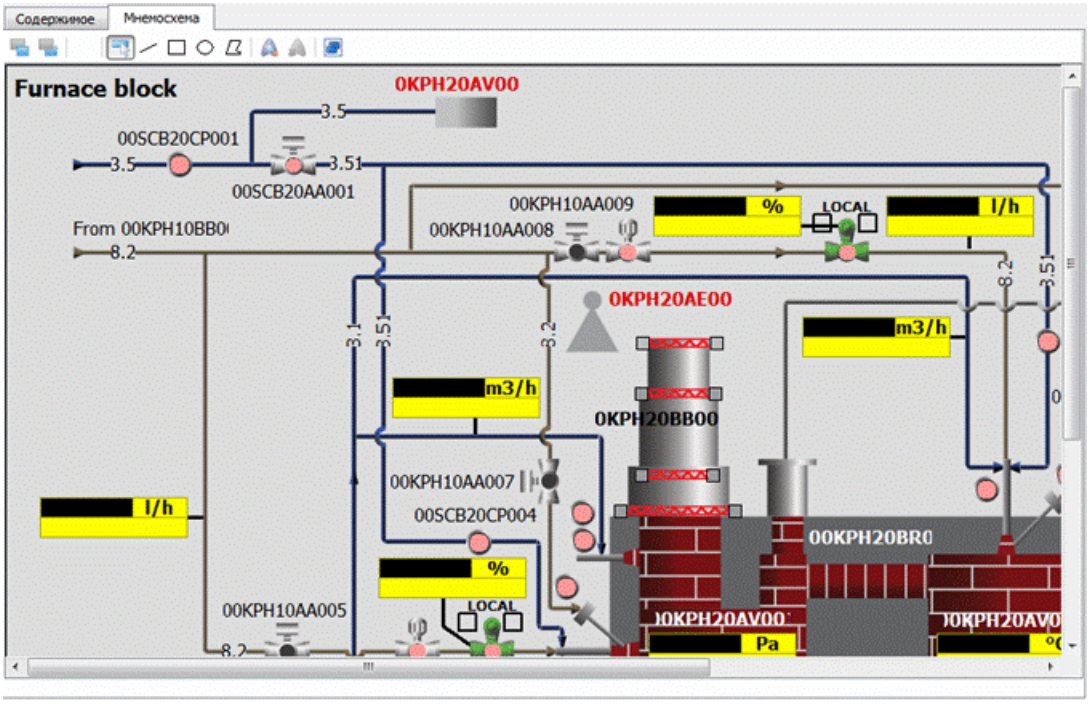


Рисунок 3.424 - Страница редактирования мнемосхемы

Добавление функционального блока на мнемосхему производится следующими способами:

- «перетаскиванием» типа функционального блока из дерева библиотеки или дерева объектов проекта;

- добавлением объекта на диаграмму, если объект представляет собой графический элемент;

- рисованием графического примитива;

- при вставке скопированных объектов.

Для удаления выделенных объектов необходимо нажать на кнопку на панели инструментов либо на клавишу **<Delete>** на клавиатуре.

На панели инструментов страницы редактирования мнемосхемы доступны следующие кнопки:

("Вперед") – переместить выделенные объекты вверх (значение zValue увеличивается на 1);

("Назад") – переместить выделенные объекты вниз (значение zValue уменьшается на 1);

("Выделение") – завершить рисование и перейти к выделению объектов;

("Нарисовать объект-линию") – перейти в режим рисования линии;

("Нарисовать объект-прямоугольник") – перейти в режим рисования прямоугольника;

("Нарисовать объект-эллипс") – перейти в режим рисования эллипса;

("Нарисовать объект-многоугольник") – перейти в режим рисования многоугольника;

("Добавить текстовой объект") – добавляет на мнемосхему надпись;

("Показать объекты") – показать спрятанные объекты;

("Скрыть объекты") – скрыть выделенные объекты;

("Выровнять границы") – откорректировать границы элемента;

 ("Выровнять границы") – откорректировать границы элемента;

 ("Увеличить") – увеличить изображение в окне редактирования мнемосхемы;

 ("Уменьшить") – уменьшить изображение в окне редактирования мнемосхемы;

 ("Нормальный размер") – показать изображение в окне редактирования мнемосхемы в масштабе 1:1;

 ("Выравнивание по левому краю") – все выделенные объекты выравниваются по левому краю объекта, расположенного левее других выделенных объектов;

 ("Выравнивание по правому краю") – все выделенные объекты выравниваются по правому краю объекта, расположенного правее других выделенных объектов;

 ("Выравнивание по центру по горизонтали") – центры всех выделенных объектов помещаются на линию, находящуюся ровно посередине между левым краем объекта, расположенного левее других выделенных объектов, и правым краем объекта, расположенного правее других выделенных объектов;

 ("Выравнивание по верхнему краю") – все выделенные объекты выравниваются по верхнему краю объекта, расположенного выше других выделенных объектов;

 ("Выравнивание по нижнему краю") – все выделенные объекты выравниваются по нижнему краю объекта, расположенного ниже других выделенных объектов;

 ("Выравнивание по центру по вертикали") – центры всех выделенных объектов помещаются на линию, находящуюся ровно посередине между верхним краем объекта, расположенного выше других выделенных объектов, и нижним краем объекта, расположенного ниже других выделенных объектов;

 ("Выравнивание по центру") – центры всех выделенных объектов помещаются в точку на пересечении линии, находящейся ровно посередине между левым краем объекта, расположенного левее других выделенных объектов, и правым краем объекта, расположенного правее других выделенных объектов, и линии, находящейся ровно посередине между верхним краем объекта, расположенного выше других выделенных объектов, и нижним краем объекта, расположенного ниже других выделенных объектов;

 ("Расставить объекты по горизонтали") – выделенные объекты располагаются через равные промежутки по горизонтали, при повторном нажатии эти промежутки уменьшаются, но левая граница объекта, расположенного левее других выделенных объектов, остается на месте;

 ("Расставить объекты по вертикали") – выделенные объекты располагаются через равные промежутки по вертикали, при повторном нажатии эти промежутки уменьшаются, но верхняя граница объекта, расположенного выше других выделенных объектов, остается на месте;

 ("Настроить объект") - доступные настройки для объекта.

Для рисования нового объекта в виде линии необходимо нажать на кнопку  на панели инструментов страницы, подвести «мышь» к первой точке линии, нажать на левую кнопку «мыши» и, удерживая ее в нажатом состоянии, довести «мышь» до конечной точки, после чего отпустить левую кнопку «мыши».

Для рисования новых объектов в виде прямоугольника или эллипса необходимо нажать на кнопку  или  , соответственно, на панели инструментов страницы, подвести «мышь» к одному из углов ограничивающего объект прямоугольника, нажать на левую кнопку «мыши» и, удерживая ее в нажатом состоянии, довести «мышь» до конечной точки, после чего отпустить левую кнопку «мыши».

Для рисования нового объекта в виде многоугольника необходимо нажать на кнопку на панели инструментов страницы, подвести «мышь» к первой точке многоугольника и выделить ее щелчком левой кнопки «мыши». Затем подвести указатель «мыши» к второй точке и т.д. Рисование завершается, когда очередной точкой многоугольника станет его начальная точка.

Рисование любого объекта может быть прервано путем нажатия на кнопку на панели инструментов либо путем нажатия на клавишу **<Esc>** на клавиатуре.

При нажатии на кнопку на панели инструментов страница редактора мнемосхем изменит начальное значение переменной «size» (размер мнемосхемы) в соответствии с положением и размерами объектов мнемосхемы.

3.4.8.1.9. Пользовательские типы данных

Пользовательские типы данных позволяют создавать из имеющихся в SCADA-системе "Соната" типов данных (см. Приложение А) свои структурные типы данных, состоящие из нескольких полей, что позволяет более удобно описывать объекты, имеющие много информации.

Пользовательские типы данных располагаются в папке **Типы данных**, которая находится в дереве объектов (см. рис. 3.425).

На данном этапе нельзя создавать пользовательские типы данных. В папке **Типы данных** программы **IECEditor** располагаются пользовательские типы данных, созданные ранее в программе **ProjectManager** (Менеджер проектов) (см. раздел 3.2.2.7).

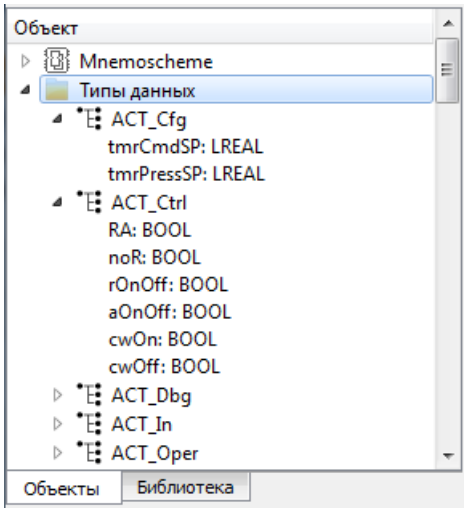


Рисунок 3.425 - Папка Типы данных в дереве объектов

3.4.8.1.10. Работа с папкой Ресурс (добавление элементов и редактирование свойств)

Для добавления нового элемента ресурса необходимо выделить в дереве объектов папку, в которую будет добавлен новый элемент, после чего в разделе Редактировать главного меню выбрать команду Добавить либо выбрать команду Добавить контекстного меню дерева объектов (см. рис. 3.426). Затем в выпадающем меню следует выбрать команду «Элемент ресурса».

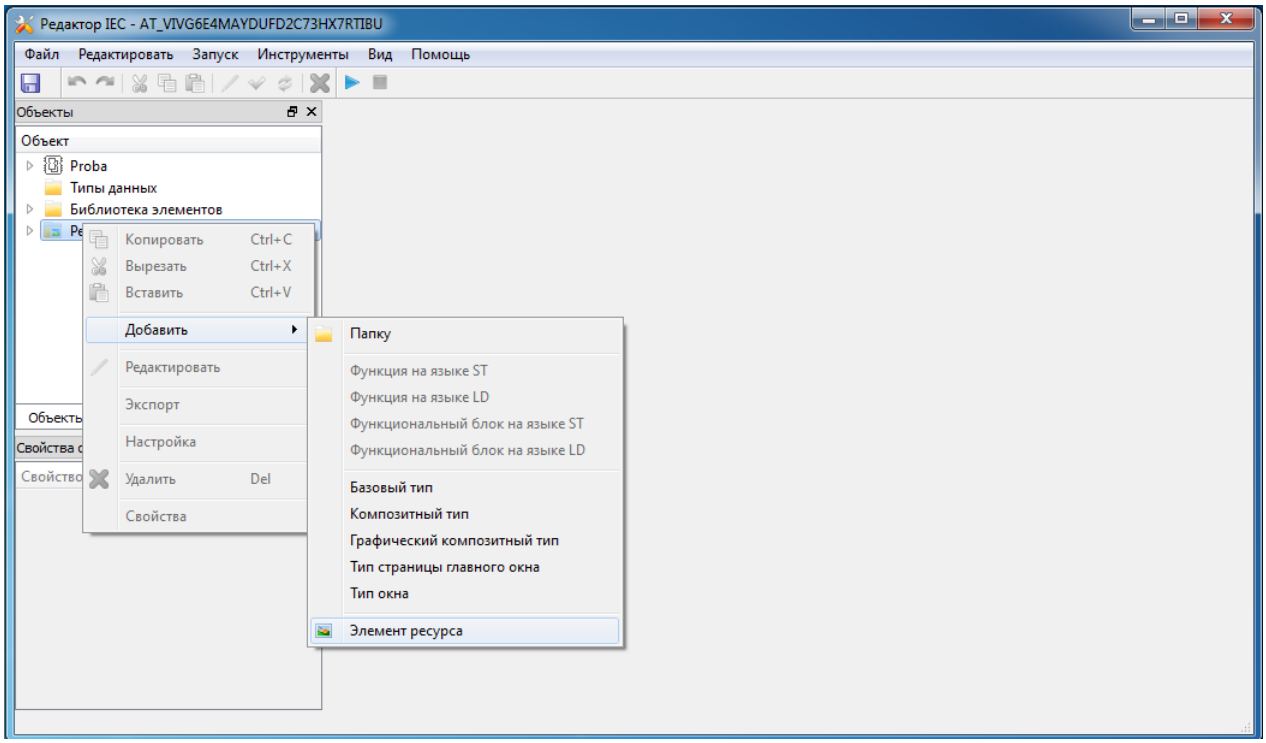


Рисунок 3.426 - Контекстное меню добавления элемента ресурса

Редактор выведет диалог выбора файлов ресурса (см. рис. 3.427). Оператор должен выбрать один или несколько файлов, после чего нажать на кнопку «Открыть». Редактор добавит выбранные файлы в папку **Ресурс** дерева объектов. Если оператор нажмет на кнопку Отмена, то новые файлы добавлены не будут.

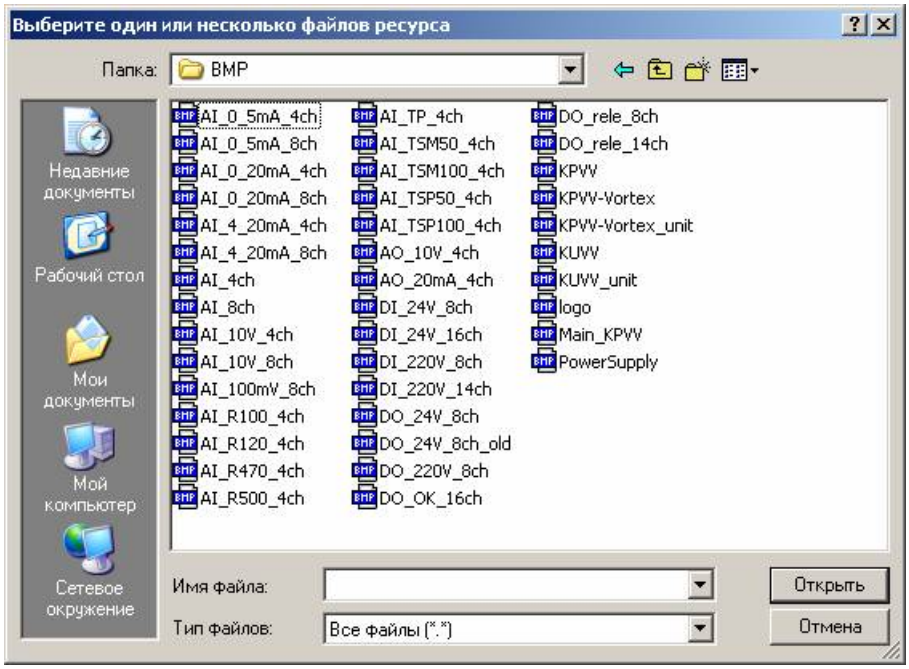


Рисунок 3.427 - Диалог выбора файла ресурсов

Для изменения имени или описания элемента ресурса необходимо выделить узел в дереве, соответствующий изменяемому элементу, после чего выбрать пункт главного меню «Редактировать/ Свойства», либо пункт «Свойства» контекстного меню дерева объектов. Программа выведет диалог свойств типа функционального блока (см. рис. 3.428).

Оператор должен изменить требуемые характеристики, после чего нажать на кнопку «OK». Редактор внесет сделанные оператором изменения. При нажатии на кнопку «Отмена» изменения внесены не будут.

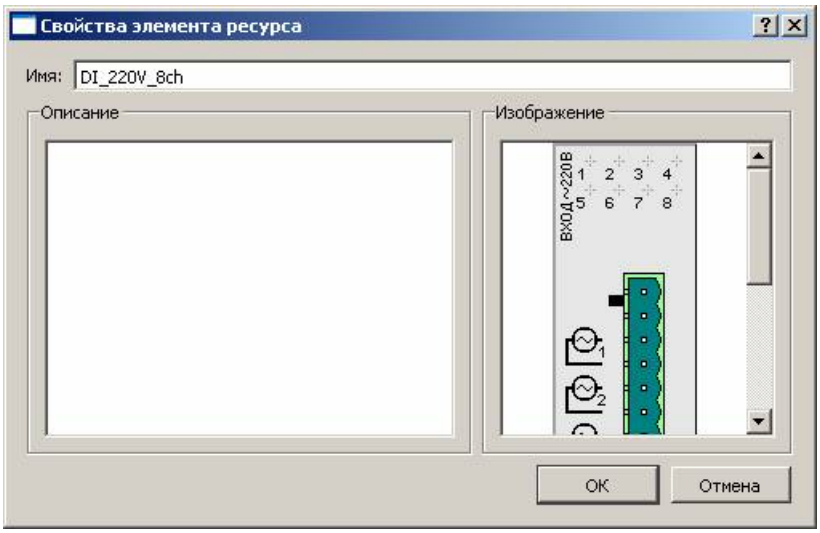


Рисунок 3.428 - Диалог редактирования свойств элемента ресурса

3.4.8.1.11. Удаление папки, типа функционального блока или элемента ресурса

Для удаления папки, типа функционального блока или элемента ресурса из дерева объектов необходимо выделить удаляемый элемент, после чего в разделе Редактировать главного меню выбрать команду Удалить Del, либо выбрать команду Удалить Del контекстного меню дерева объектов (см. рис. 3.429), либо нажать на кнопку панели инструментов, либо нажать клавишу **<Del>** на клавиатуре.

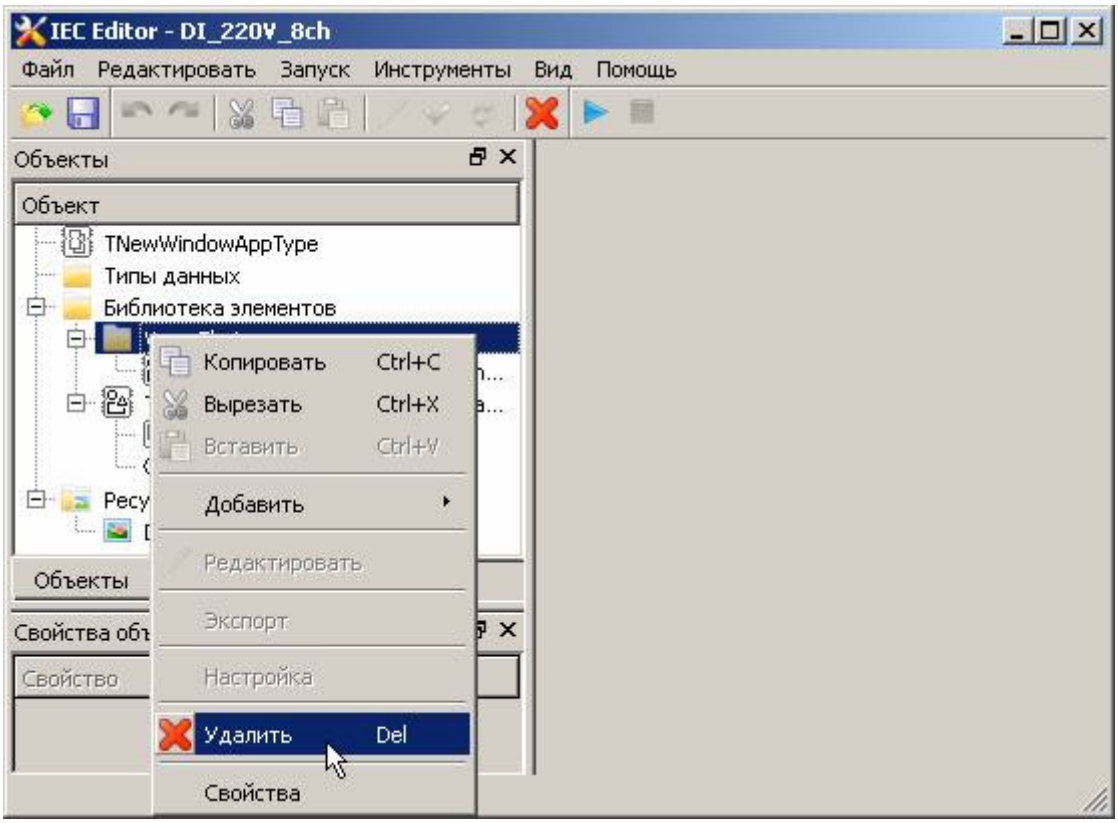


Рисунок 3.429 - Контекстное меню вызова команды Удалить

Редактор проверит возможность удаления элемента. Тип функционального блока может быть удален, если он не используется при создании объектов, то есть, ни в каком из типов функциональных блоков нет объектов удаляемого типа. Папка может быть удалена, если она и ее подпапки не содержат типов функциональных блоков. Элемент ресурса может быть удален, если он не используется ни в каком из типов приложений.

Если удаление возможно, то программа удалит выбранные элементы. Если удаление невозможно, то будет выведено сообщение об ошибке (см. рис. 3.430). Для закрытия сообщения и продолжения работы необходимо нажать на кнопку OK.

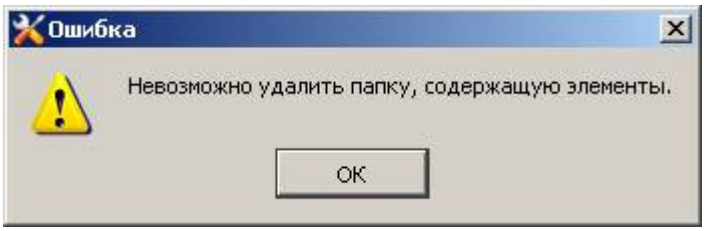


Рисунок 3.430 - Сообщение об ошибке в процессе удаления

3.4.8.1.12. Экспортирование и импортирование элементов

В редакторе **IECEditor** есть возможность переноса готовых элементов одного проекта в другой проект. Если у данного элемента есть вложенные элементы, то они так же могут быть перенесены. Это реализуется с помощью функций Экспорт и Импорт, которые располагаются на панели инструментов в меню Инструменты (см. рис. 3.431). Функция Экспорт так же может быть вызвана из контестного меню у элемента (см. рис. 3.432).

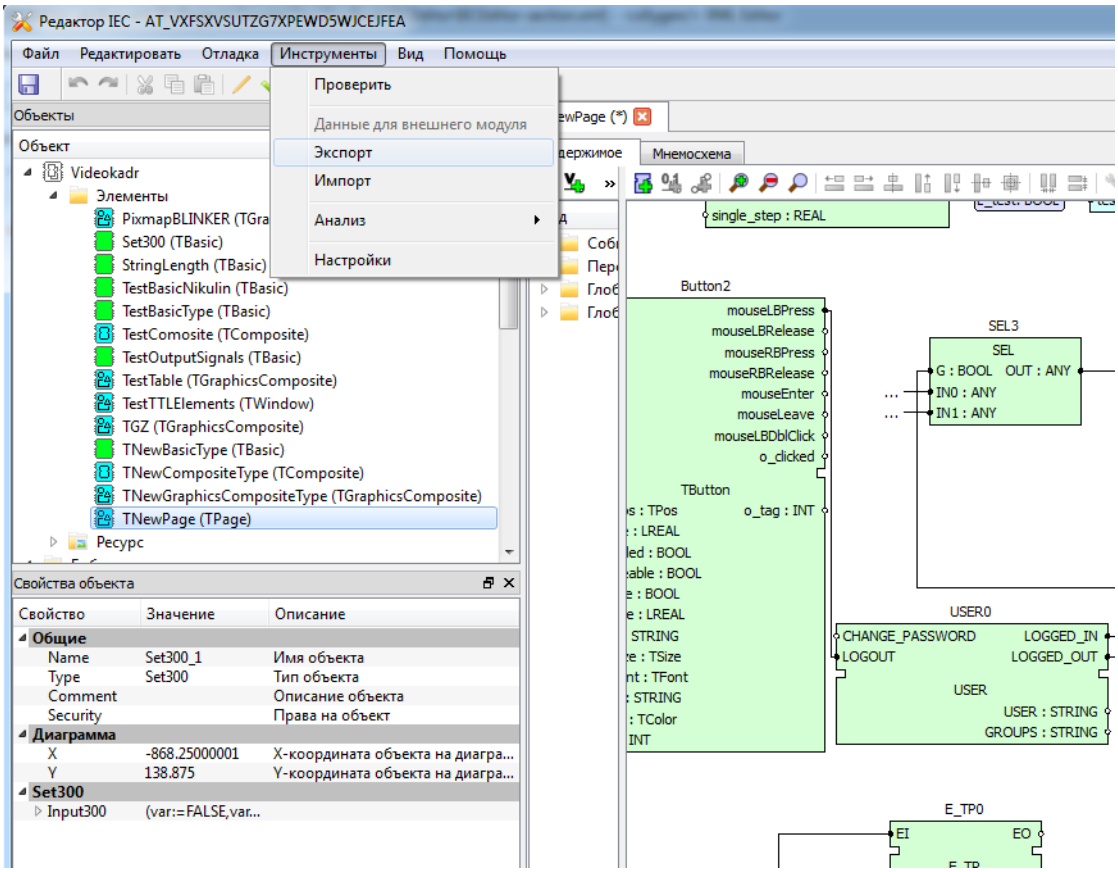


Рисунок 3.431 - Функция Экспорт на панели задач

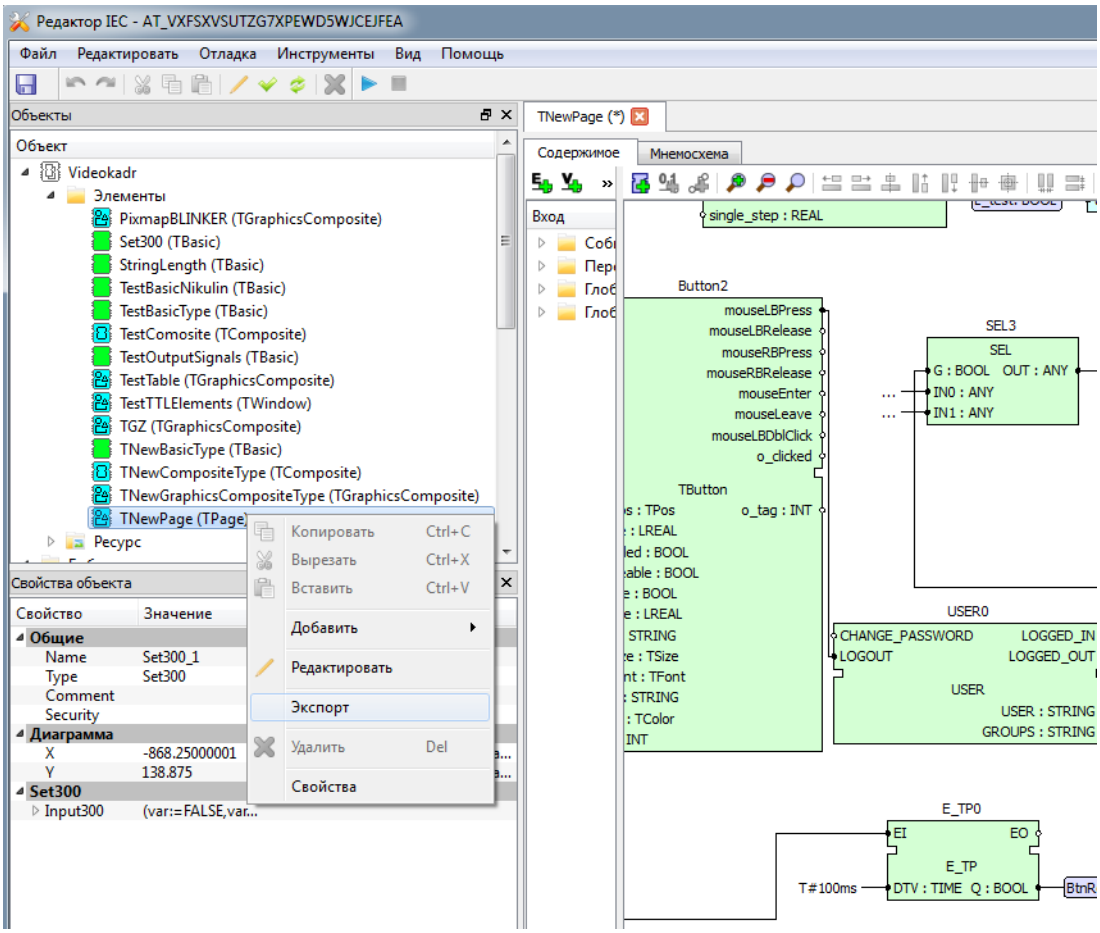


Рисунок 3.432 - Функция Экспорт в контекстном меню элемента

Для экспорта нужного элемента необходимо выделить его в дереве обьектов и нажать на функцию Экспорт. После этого откроется окно, в котором необходимо выбрать нужные элементы для экспортирования (см. рис. 3.433). Выбранные элементы будут экспортированы вместе со своими шаблонами.

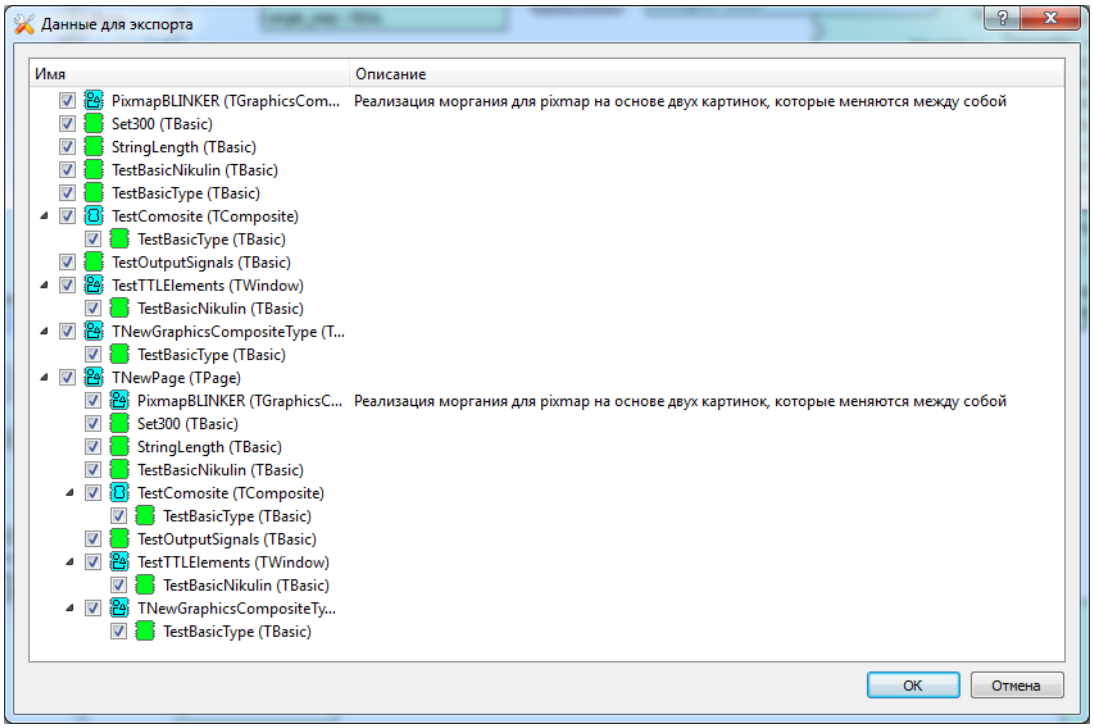


Рисунок 3.433 - Окно выбора вложенных элементов

По нажатию на кнопку OK откроется стандартное окно для выбора места сохранения и указания имени файла. Экспортированный элемент сохраняется в текстовом файле формата txt.

Для добавления данного элемента в новый проект необходимо воспользоваться функцией Импорт в меню Инструменты на панели задач. При использовании данной функции откроется стандартное окно для выбора файла, который содержит необходимую информацию. В новый проект будет добавлен сохранённый элемент и все вложенные, если они были указаны при сохранении.

3.4.8.2. Отладка приложений

3.4.8.2.1. Запуск и остановка отладки

Для отладки приложения необходимо выбрать в разделе Запуск главного меню команду Запустить либо нажать кнопку на панели инструментов. Редактор проверит приложение и, если нет критических ошибок, запустит приложение для отладки.

Для прекращения отладки приложения необходимо выбрать в разделе Запуск главного меню команду Остановить либо нажать кнопку на панели инструментов. Отладка приложения будет прекращена.

3.4.8.3. Настройка параметров приложения

Для вызова меню настроек параметров приложения необходимо в главном меню выбрать пункт **Инструменты - Настройки**.

3.4.8.3.1. Настройка автосохранения

Для автоматического сохранения редактируемого типа приложения используются следующие настройки (см. рис. 3.434)

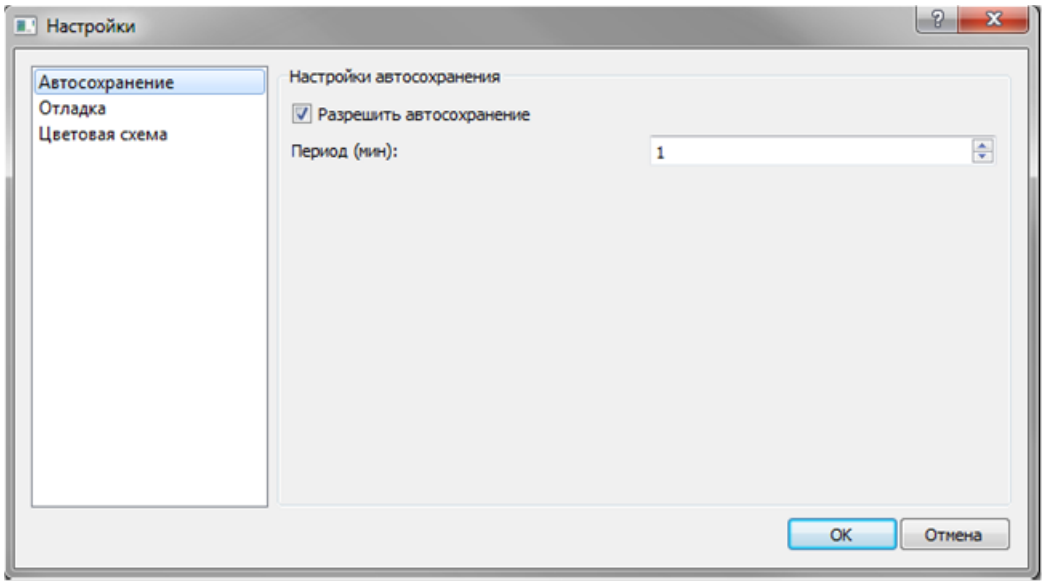


Рисунок 3.434 - Диалог параметров автосохранения

В данном диалоговом окне можно включить или отключить автосохранение и задать интервал в минутах.

3.4.8.3.2. Настройка параметров отладки приложения

Во время отладки графического приложения производится автоматический вход пользователя, что избавляет разработчика от необходимости многократного ввода имени пользователя и пароля. Имя и права пользователя задаются в диалоге настройки параметров отладки приложения (см. рис. 3.435).

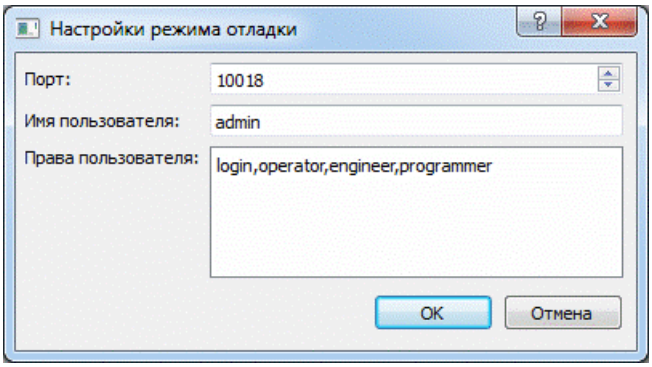


Рисунок 3.435 - Диалог настройки параметров отладки приложения

Поле ввода порта предназначено для редактирования номера IP-порта, который будет использоваться запущенным приложением для взаимодействия с другими модулями системы.

Имя пользователя и его права не используются при отладке консольных приложений.

3.4.8.3.3. Настройка цветовой схемы

Настройки цветовой схемы используются в текстовых редакторах для подсвечивания текста в программах на языке ST. Параметры подсветки задаются в диалоговом окне **Настройки редактора исходного кода** (см. рис. 3.436).

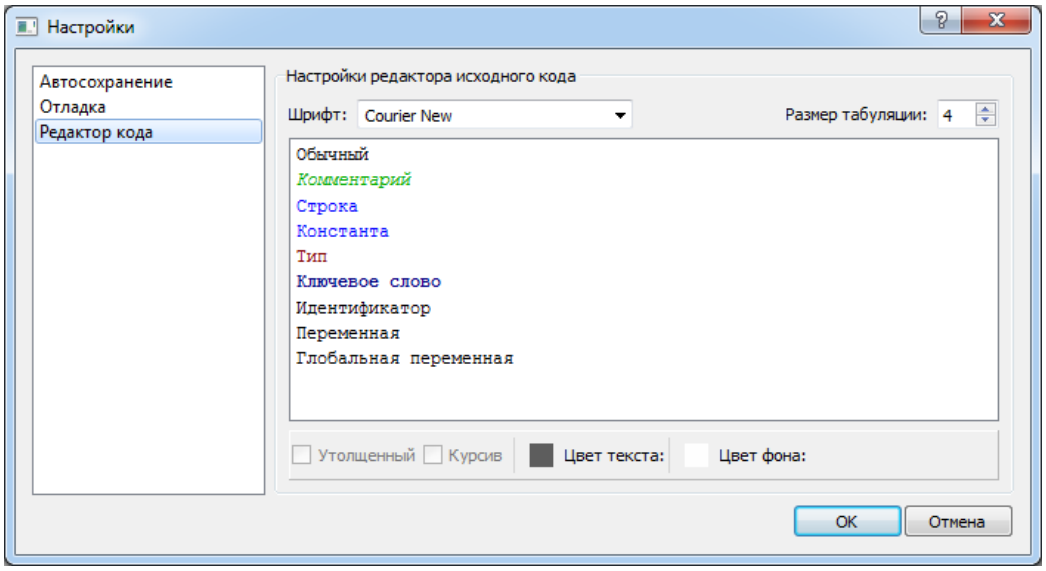


Рисунок 3.436 - Диалог настройки цветовой схемы.

Вверху окна задается шрифт. Этот параметр общий для всех типов литералов. Ниже задаются утолщение, наклон, цвет шрифта и цвет фона для каждого типа литералов в отдельности.

3.4.8.4. Получение справочной информации

3.4.8.4.1. Получение информации о программе

Для получения информации о программе необходимо выбрать в разделе Помощь главного меню команду О программе... . Редактор выведет диалог информации о программе (см. рис. 3.437).

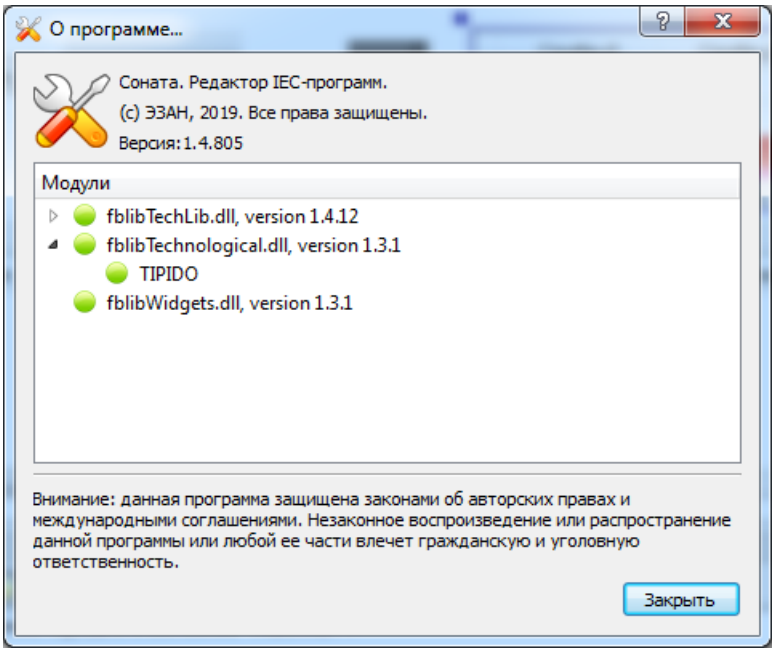


Рисунок 3.437 - Диалог информации о программе

Помимо общей информации, диалог содержит список загруженных библиотек типов функциональных блоков с указанием имен типов, загруженных из данной библиотеки. Если при загрузке какого-либо типа из библиотеки возникла ошибка, то такая строка будет помечена красной или желтой иконкой, а справа от имени типа будет выведено сообщение об ошибке.

3.4.9. Описание утилиты TestTimeSync

Утилита TestTimeSync предназначена для проверки точности синхронизации часов между узлами без какой-либо привязки к системе "Соната". Она осуществляет опрос узлов по протоколу SNTP и отображает времена узлов и расхождение между ними в сводной таблице. На рис. 3.438 представлено окно утилиты TestTimeSync.

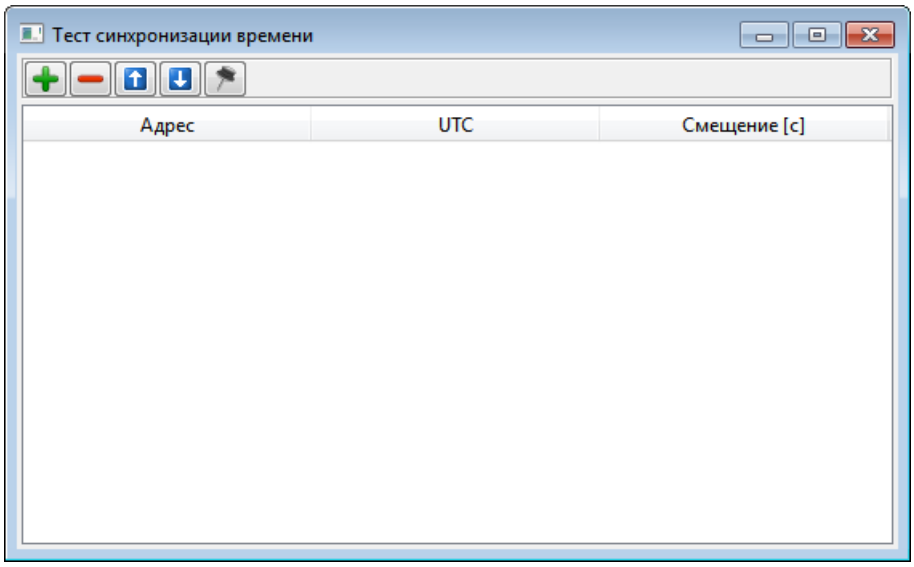


Рисунок 3.438 - Главное окно утилиты TestTimeSync

В верхней части главного окна утилиты TestTimeSync располагаются следующие кнопки:

- ("Добавить") - по нажатию на данную кнопку откроется окно ввода адрес узла (см. рис. 3.439). В данном окне необходимо ввести ip-адрес узла, время которого будет отображаться. Первым необходимо добавить узел с точным временем. Далее остальные узлы;

- ("Удалить") - по нажатию на данную кнопку из списка будет удалена выделенная запись (узел);

- ("Переместить вверх") - по нажатию на данную кнопку выделенная запись (узел) переместиться на одну позицию вверх;

- ("Переместить вниз") - по нажатию на данную кнопку выделенная запись (узел) переместиться на одну позицию вниз;

- ("Поверх других окон") - по нажатию на данную кнопку можно сделать чтобы главное окно утилиты отображалось поверх всех других окон.

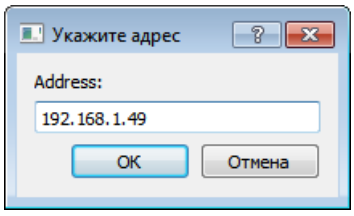


Рисунок 3.439 - Окно ввода ip-адреса добавляемого в тест узла

На рис. 3.440 представлено главное окно утилиты с добавленными для тестирования узлами.

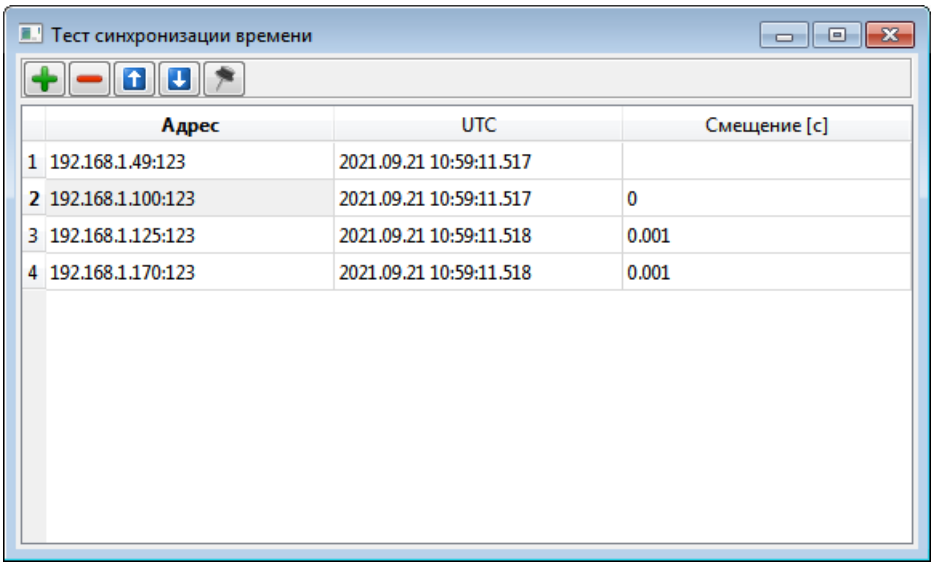


Рисунок 3.440 - Главное окно утилиты TestTimeSync с добавленными для тестирования узлами

Таблица с записями (узлами) для тестирования состоит из следующих столбцов:
- **Адрес** - указан ip-адрес и SNTP порт узла;
- **UTC** - текущее время на узле в UTC;
- **Смещени [с]** - смещение времени узла в секундах относительно первого узла в таблице.

3.5. Полезные примеры

3.5.1. Настройка нескольких узлов проекта на одном АРМ

В данном примере показана настройка трех узлов проекта, которые будут работать на одном АРМ.

1. На АРМ необходимо установить три одинаковых дистрибутива SCADA системы "Соната". Описание установки дистрибутива смотрите в документе SCADA система "Соната" Руководство системного программиста КУНИ.505200.023-01.01 32.

2. В каждом из дистрибутивов необходимо запустить приложение Loader, которое будет управлять данным узлом.

Запуск первого приложения Loader выполняется командой:

./Loader -daemon или **./Loader -daemon=10000**, где значение 10000 - это порт на котором будет запущен Loader. Значение 10000 можно не указывать, т.к. оно является значением по умолчанию.

Запуск второго приложения Loader выполняется командой:

./Loader -daemon=11000.

Запуск третьего приложения Loader выполняется командой:

./Loader -daemon=12000.

Важно! В версии SCADA "Соната" 1-4, начиная с номера дистрибутива 10416, не нужно устанавливать 3 дистрибутива, как написано в п.1. Теперь можно из одного дистрибутива запустить 3 приложения Loader, как написано в п.2 и настраивать далее 3 узла, которые будут работать на одном АРМ.

Таким образом у нас будут запущены на разных портах три приложения Loader (см. рис. 3.441). Значения портов могут быть любые, но лучше начинать с 10000, чтобы не занять

системные порты и делать между значениями отступ, который необходим для набора портов, которые будут занимать приложения на каждом узле проекта. Приложения на узле проекта стартуют на портах, значения которых увеличиваются на 1 относительно порта приложения Loader данного узла.

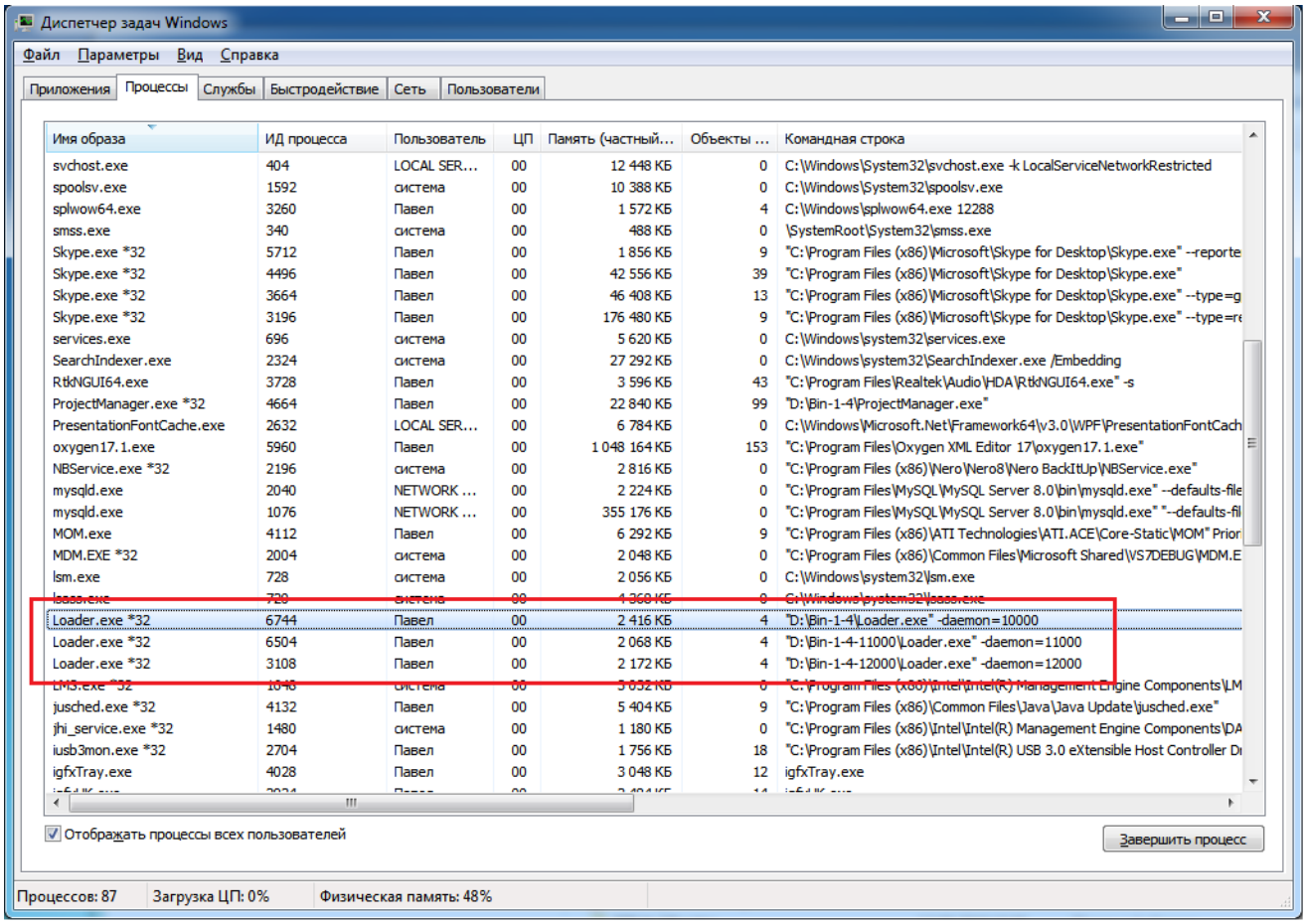


Рисунок 3.441 - Три приложения Loader на одном АРМ

3. Далее можно создавать проект в программе Менеджер проектов(ProjectManager).

В проекте создадим три узла и назовем их: **Node10000**, **Node11000** и **Node12000** (см. рис. 3.442). Далее создадим несколько типов приложений, которые будем запускать на узлах (см. рис. 3.442).

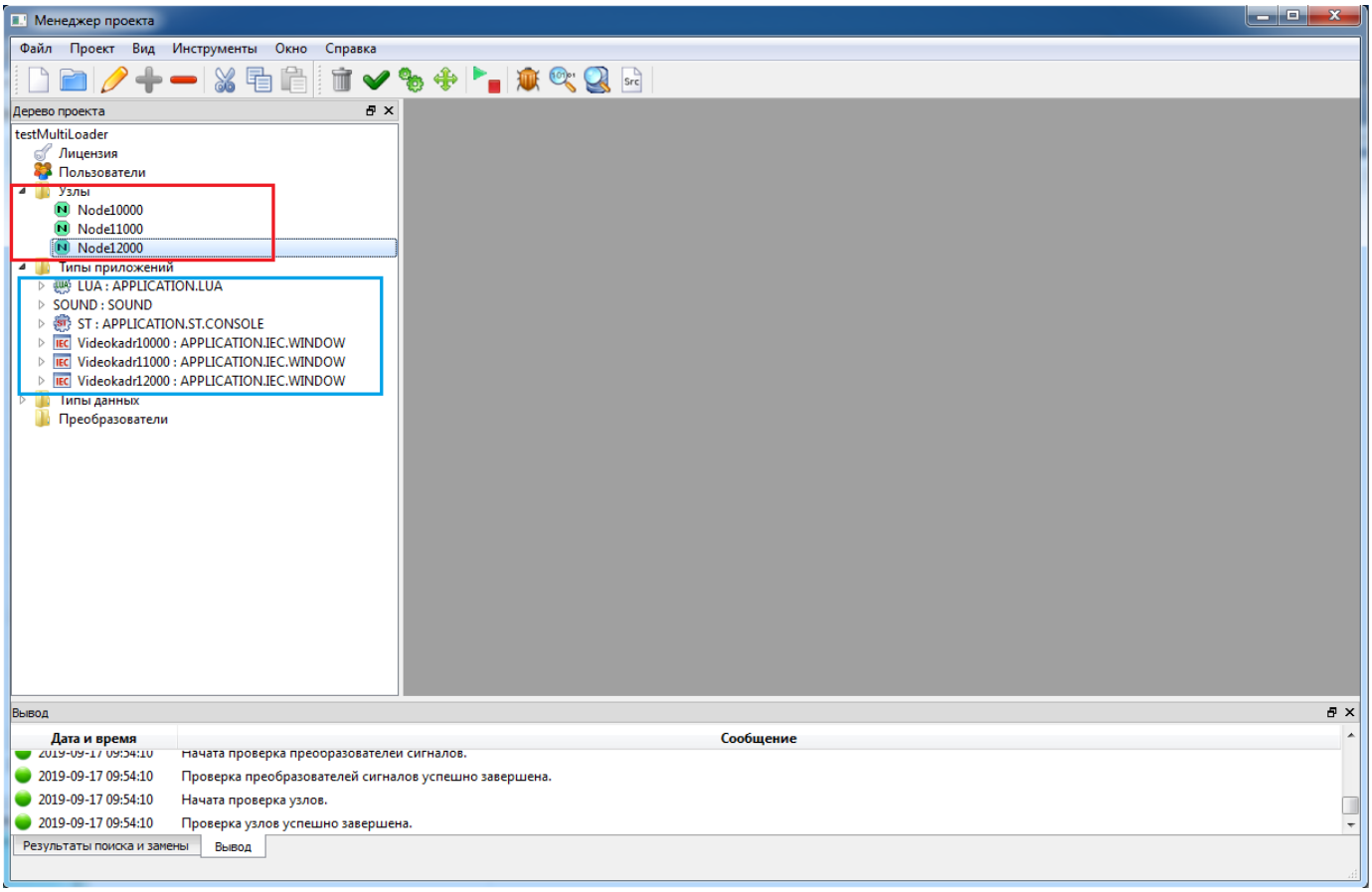


Рисунок 3.442 - Дерево нашего проекта

Далее выполним настройки каждого узла. Обратите внимание на следующие моменты:

- ip-адрес у данных узлов одинаковый;
- начальные ip-порты разные и соответствуют портам, которые использовались при запуске приложений Loader в п.2;
- приоритеты узла, как источника времени должны быть разные.

Настройки узлов смотрите на рис. 3.443, рис. 3.444 и рис. 3.445.

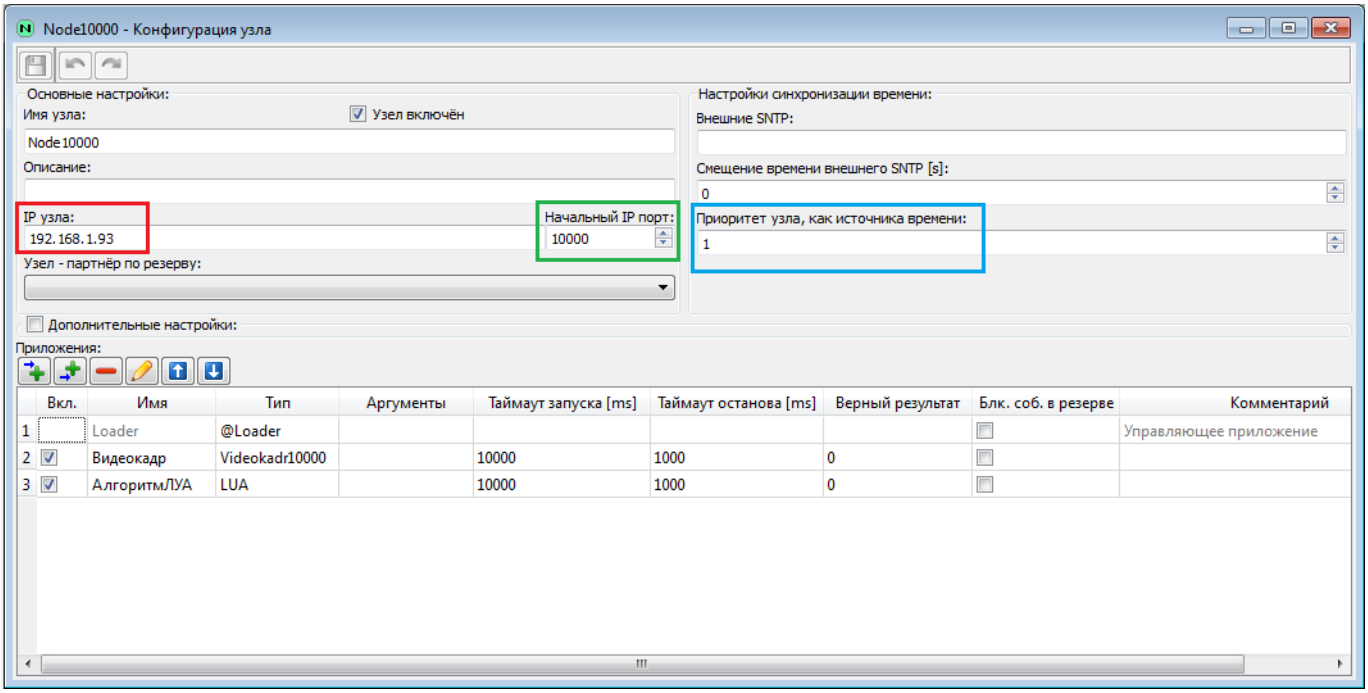


Рисунок 3.443 - Настройки узла Node10000

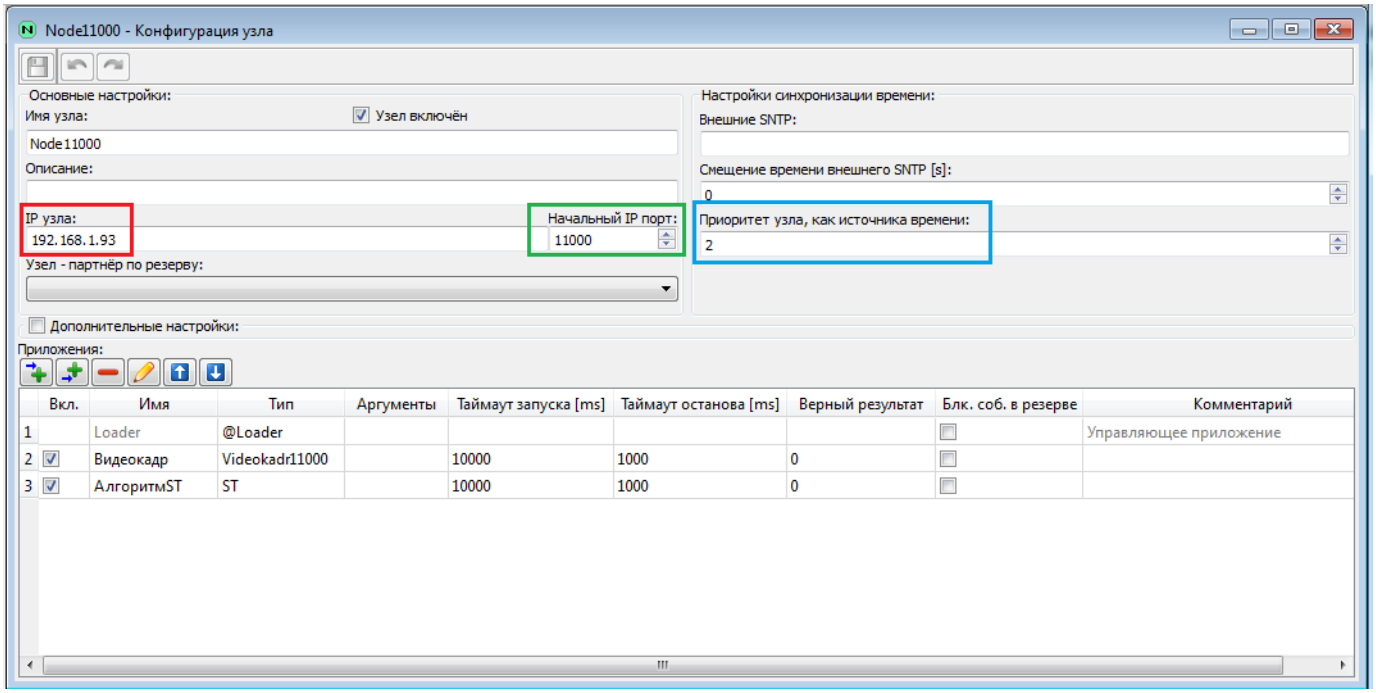


Рисунок 3.444 - Настройки узла Node11000

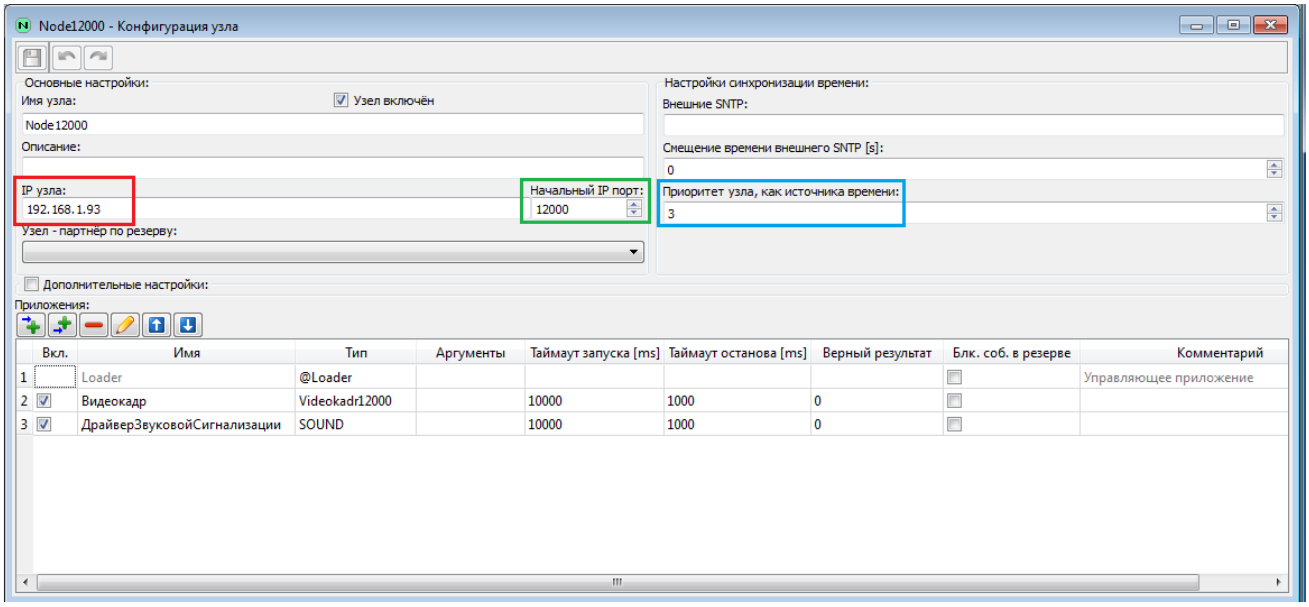


Рисунок 3.445 - Настройки узла Node12000

4. Далее компилируем проект (кнопка ), распространяем проект (кнопка ) и выполняем запуск проекта (кнопка ).

На рис. 3.446 в Центре управления можем видеть три узла, которые работают на одном АРМ.

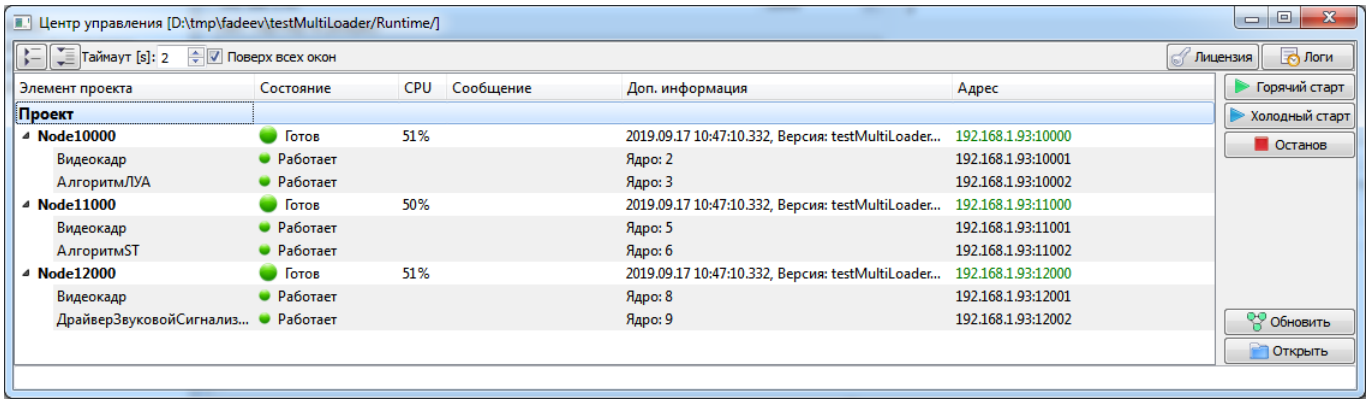


Рисунок 3.446 - Отображение в Центре управления трех запущенных узлов на одном АРМ

3.5.2. Реализация многооконного режима с применением средств СКАДА системы "Соната"

В данном примере будет рассмотрен вариант когда на узле используется одно графическое приложение APPLICATION.IEC.WINDOW, но мы можем получить несколько одинаковых окон на разных мониторах. Все окна будут работать независимо друг от друга.

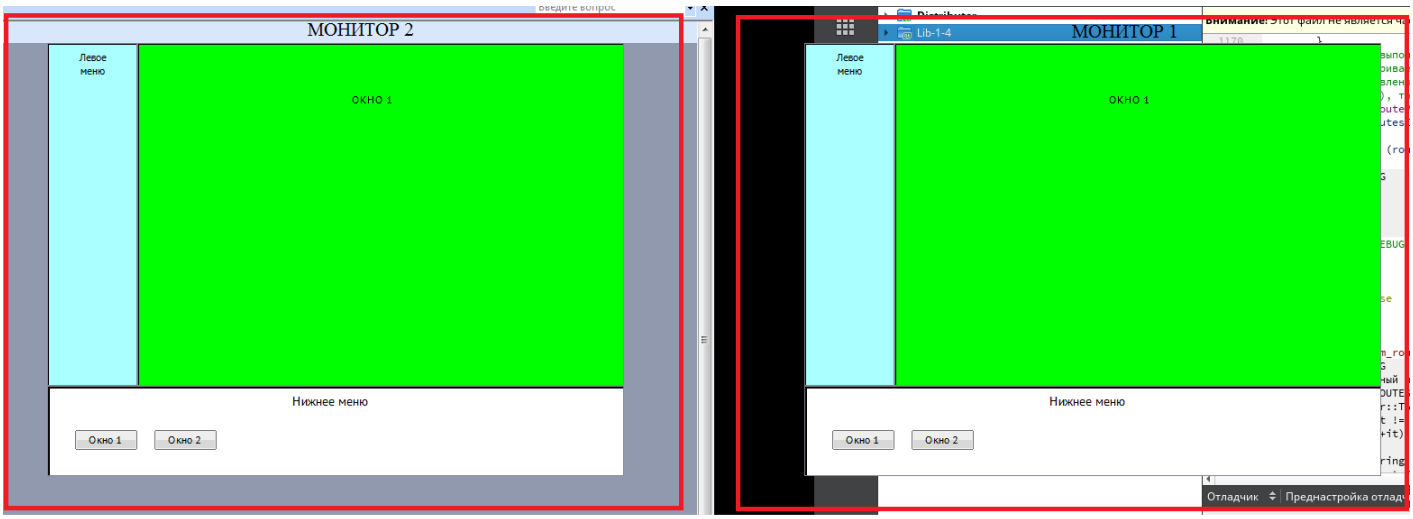


Рисунок 3.447 - Два независимых окна, располагающихся на разных мониторах

В нашем примере мы создадим рабочее окно, у которого есть нижнее меню с кнопками, левое боковое меню (не будем ничем наполнять) и центральная область для различных рабочих окон (создадим два окна). С помощью кнопок на нижнем меню мы сможем переключаться между центральными рабочими окнами.

Создадим наши кадры размером 640*480.

Для размещения наших двух кадров важно понимать как считается размерная сетка при подключении двух мониторов к одному АРМу. В моем варианте Монитор 1 (справа) является основным, а Монитор 2 (слева) дополнительным (это настраивается на АРМе). Разрешение на моих мониторах 1920*1080.

! Вариант, который разбираем в примере.

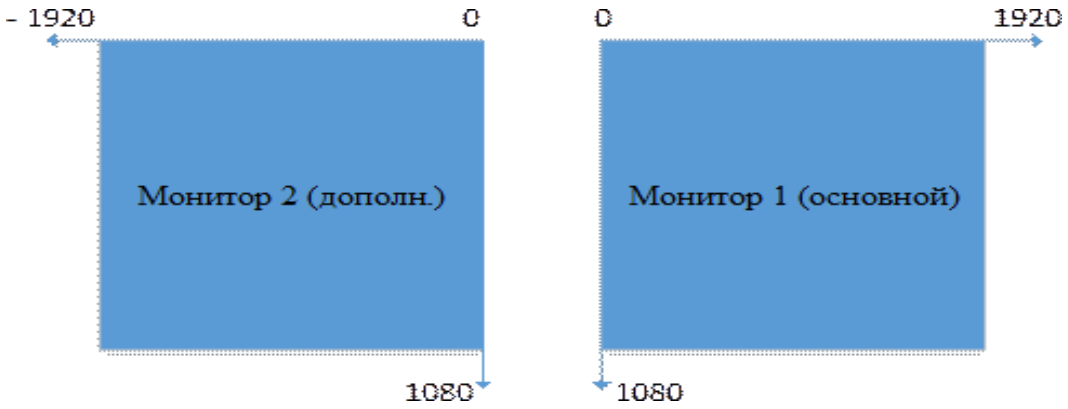


Рисунок 3.448 - Расположение двух мониторов - Вариант 1

! Возможен другой вариант.

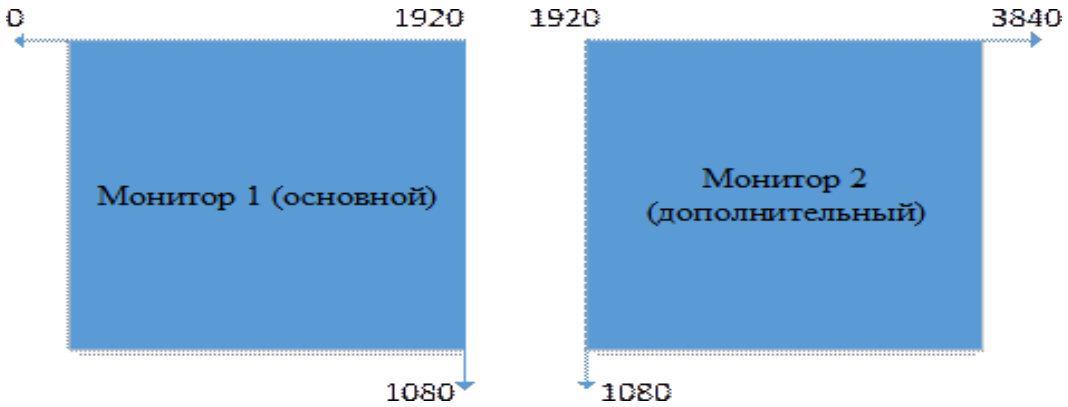


Рисунок 3.449 - Расположение двух мониторов - Вариант 2

1. Необходимо создать Тип приложения APPLICATION.IEC.WINDOW. Назовем его HMI_test_MultiWindow.

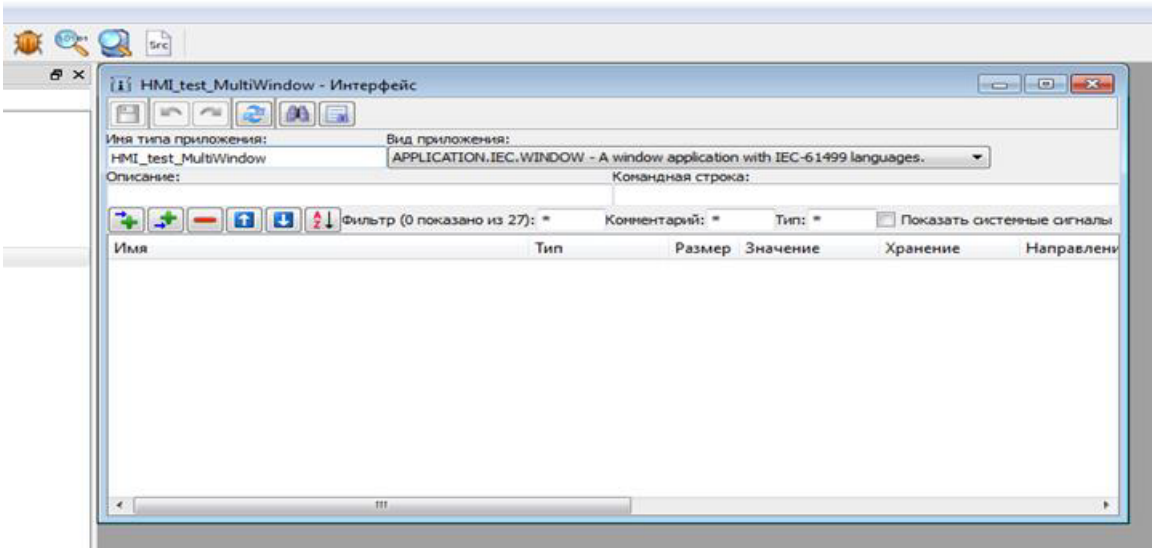


Рисунок 3.450 - Создание приложения APPLICATION.IEC.WINDOW

2. Открываем наше графическое приложение на редактирование.

3. В версии дистрибутива 1-4, в сборках 12494 и далее, добавлены новые компоненты в графический редактор IECEditor - Окно приложения. Данный компонент помогает создавать

графические приложения удобней, чем это было раньше (старые компоненты тоже остаются чтобы поддерживать совместимость проектов).

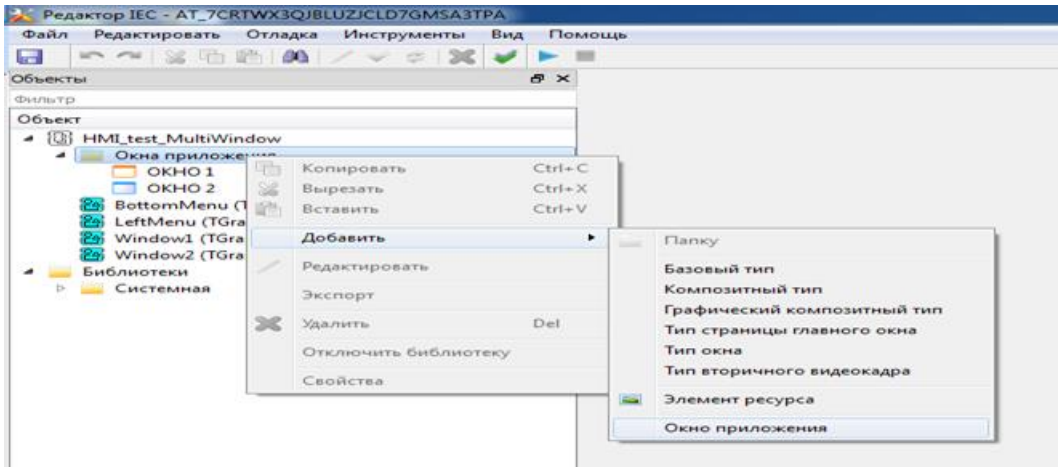


Рисунок 3.451 - Добавление компонента Окно приложения

4. Выбираем окно приложения и настраиваем свойства окна - после нажимаем кнопку Далее

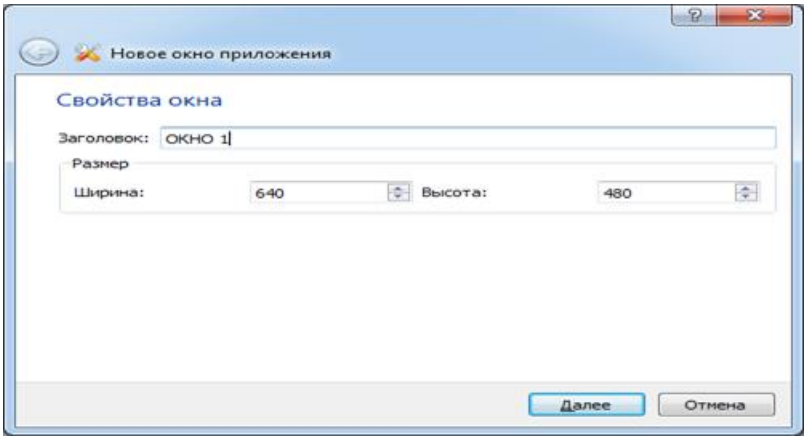


Рисунок 3.452 - Свойства Окна приложения

5. Выбираем компоновку объектов на нашем окне

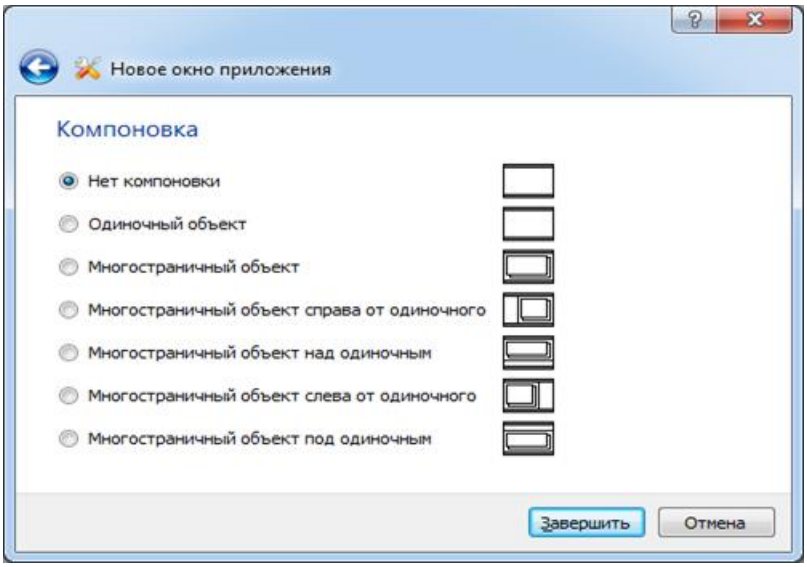


Рисунок 3.453 - Выбор компоновки объектов в окне

Есть два варианта объектов:

- одиночный (одиночное графическое наполнение);

- многостраничный (может находиться несколько кадров, которые сменяют друг друга по нажатию кнопок и др.)

В компоновках представлены несколько наиболее часто используемых вариантов. В нашем примере выберем Нет компоновки и создадим на его основе наш кадр.

6. Получим окно данного вида. Мы сейчас будем создавать нижнее меню с кнопками и левое меню (пустое). Для этого выполните действия как на картинке.

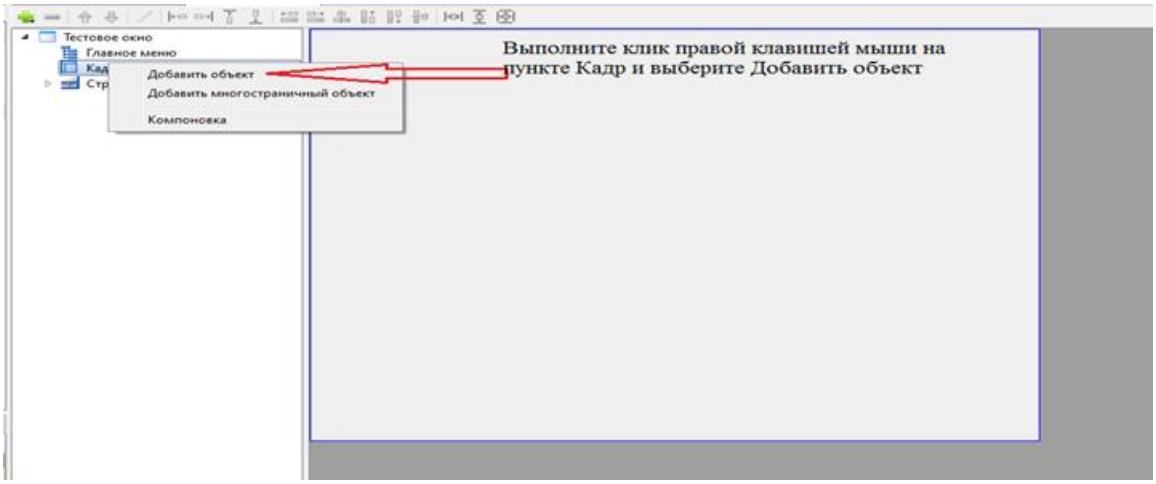


Рисунок 3.454 - Начинаем настраивать Кадр

7. Далее откроется меню создания объекта:

- можно создать новый тип (шаблон) и объект (экземпляр);

- можно создать новый объект на базе имеющегося типа;

- можно использовать существующий объект.

Мы создаем новый тип и объект (см. картинку и назовите так же). После нажмите кнопку Сохранить или Принять изменения.

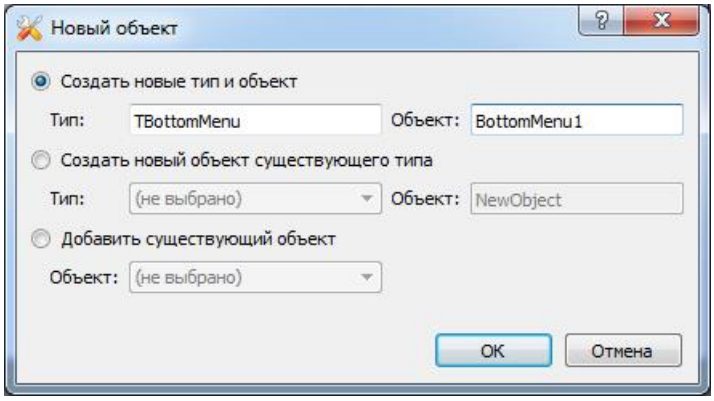


Рисунок 3.455 - Создание нового объекта

Появится тип TBottomMenu.

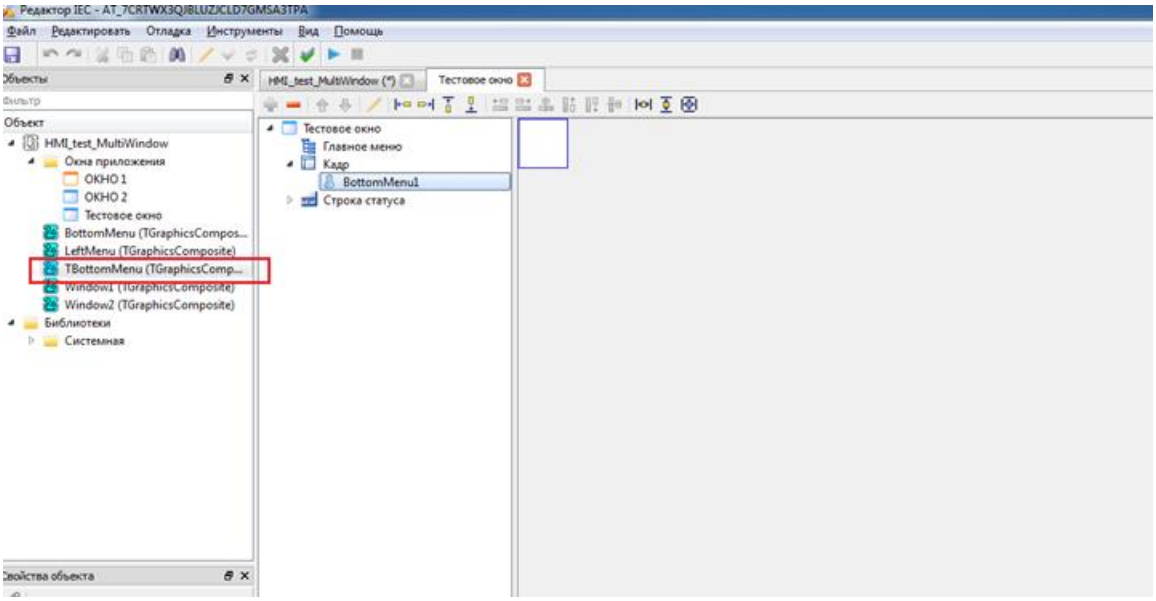


Рисунок 3.456 - Созданный нами новый тип объекта

И на главную страницу автоматически будет добавлен объект BottomMenu1.

>

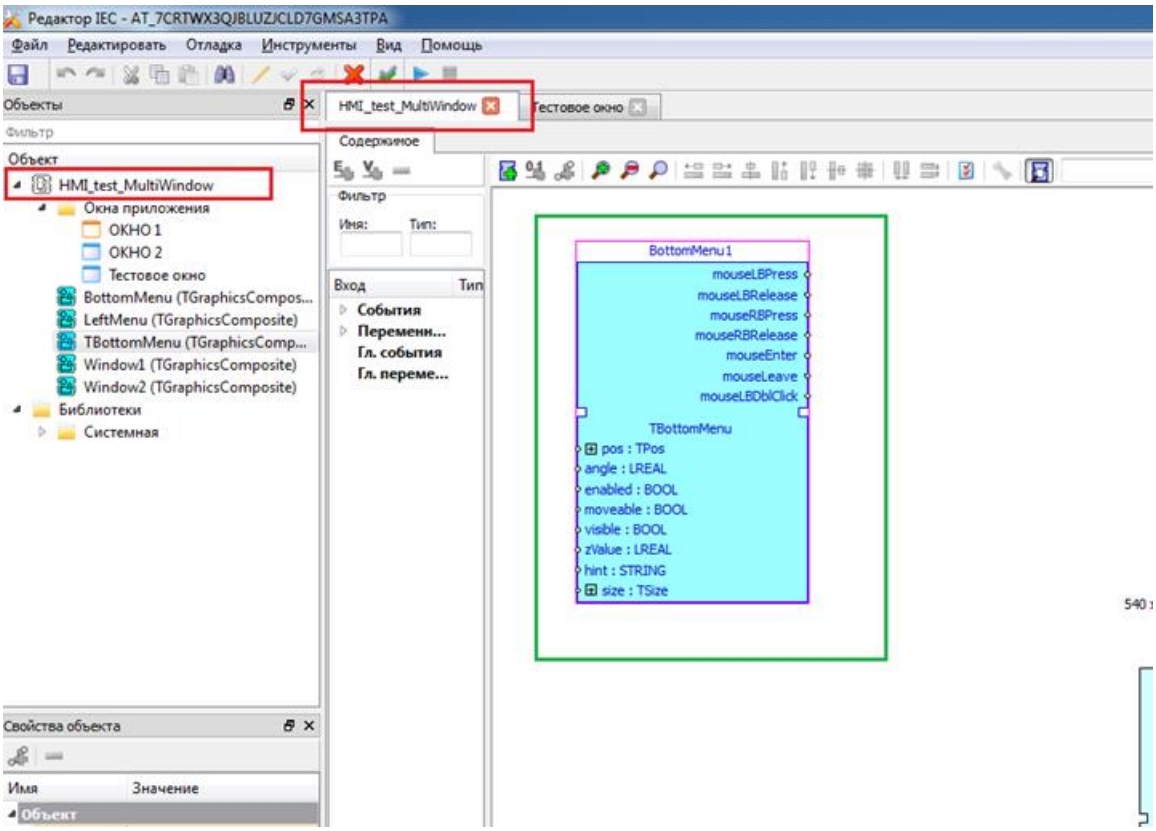


Рисунок 3.457 - Новый объект на основной странице

8. Зададим размер BottomMenu1 равный 640*100 и переместим его вниз (координата левого верхнего угла данного объекта пусть будет равна 380)

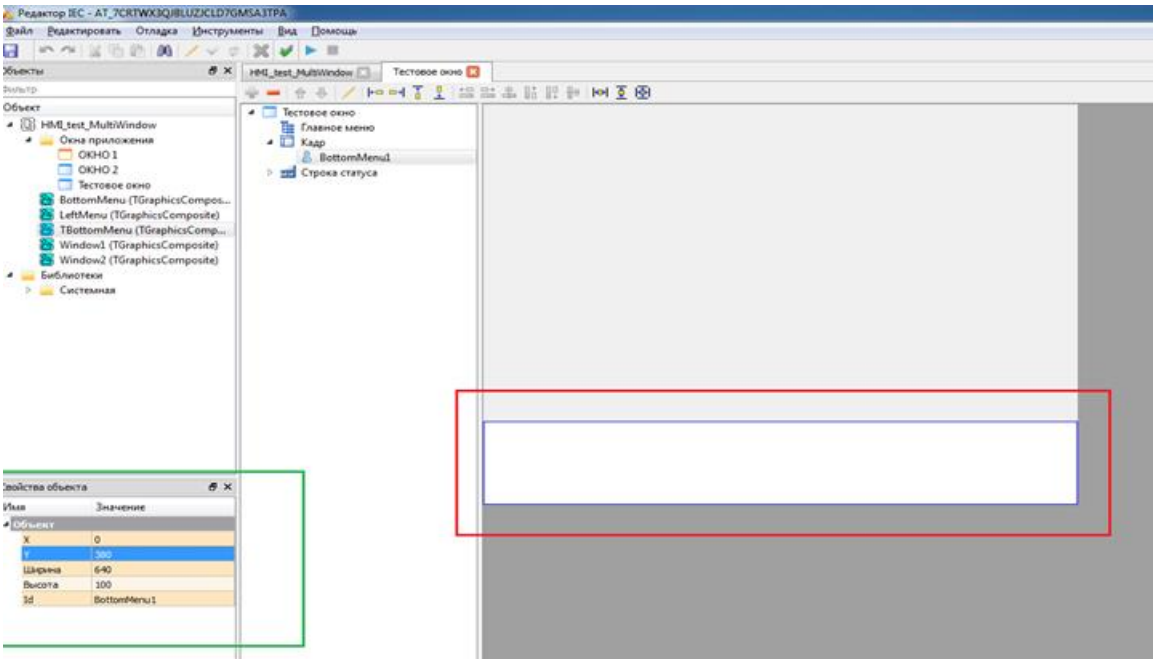


Рисунок 3.458 - Свойства объекта

9. Это у нас нижнее меню с кнопками, так что двойным кликом левой клавиши мыши заходим в его редактор и редактируем

а) задаем ему размер

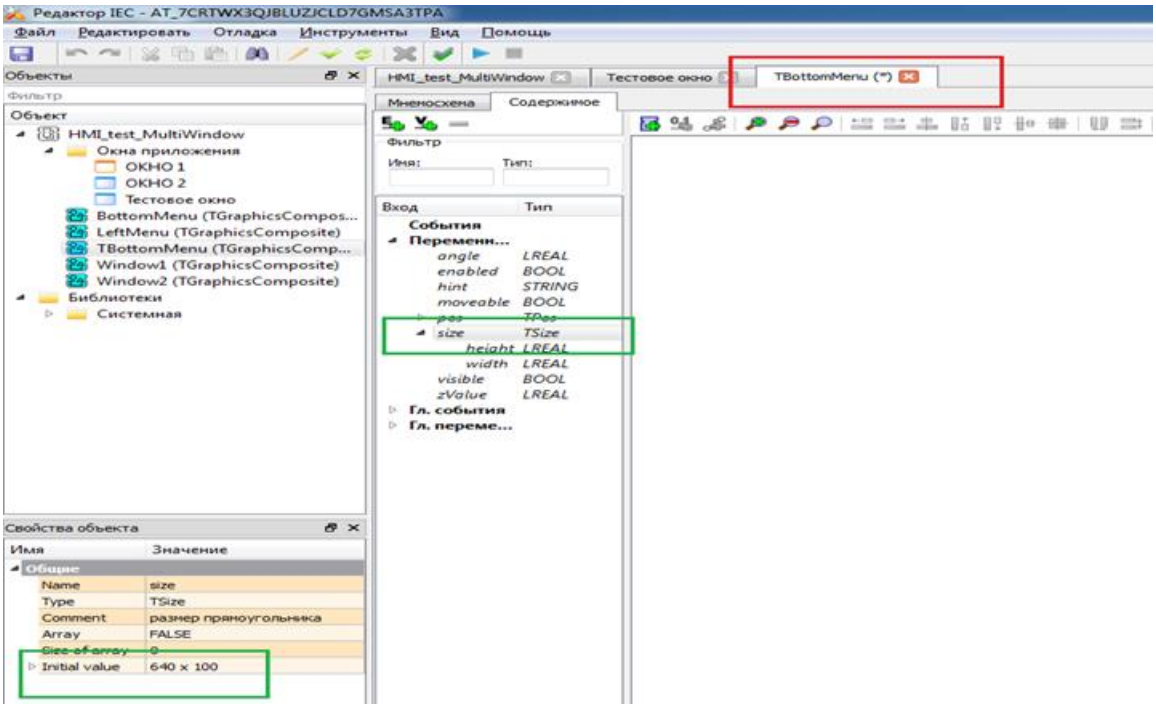


Рисунок 3.459 - Редактирование размера объекта

б) создадим прямоугольник, который будет отрисовывать рамку для этого меню

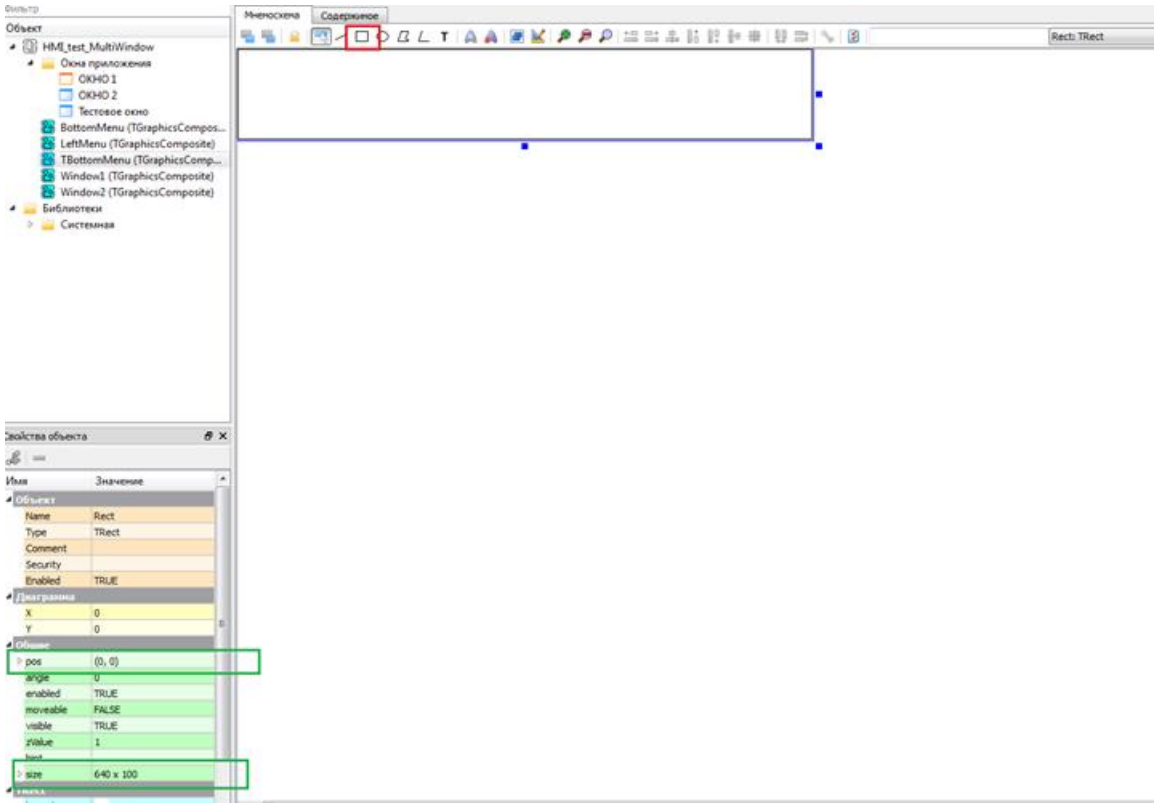


Рисунок 3.460 - Добавление прямоугольника в объект

в) добавим две кнопки TTLButton и назовем их ОКНО 1 и ОКНО 2

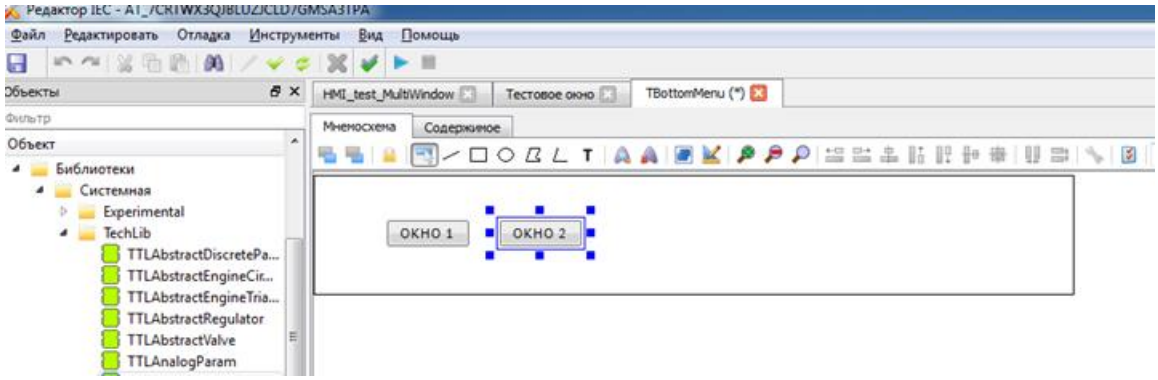


Рисунок 3.461 - Добавление кнопок в объект

10. Сохраним изменения и перейдем в наше ОКНО 1. Оно должно выглядеть следующим образом.

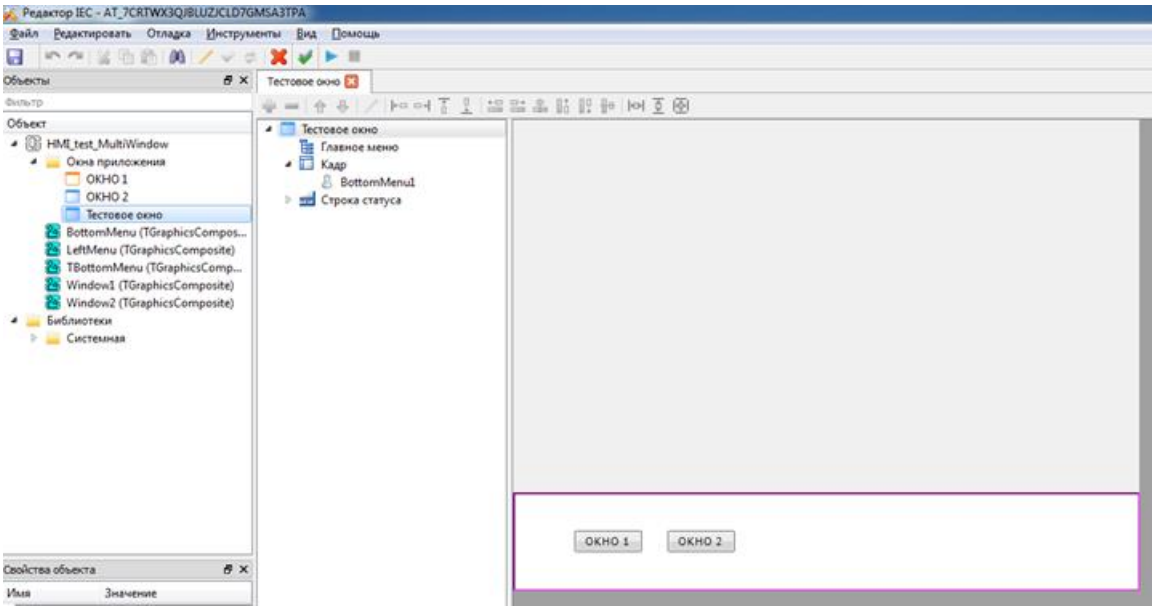


Рисунок 3.462 - Нижнее меню с кнопками

11. Аналогично создайте левое меню

а) создайте новый тип и объект с названием TLeftMenu и LeftMenu1

б) ширина 100, высота 380

в) в нем создайте прямоугольник чтобы была рамка

г) в данном примере не будем больше ничего добавлять

12. Далее будем создавать центральный многостраничный объект. Для этого на меню Кадр нажимаем правую клавишу мыши и выбираем Добавить многостраничный объект.

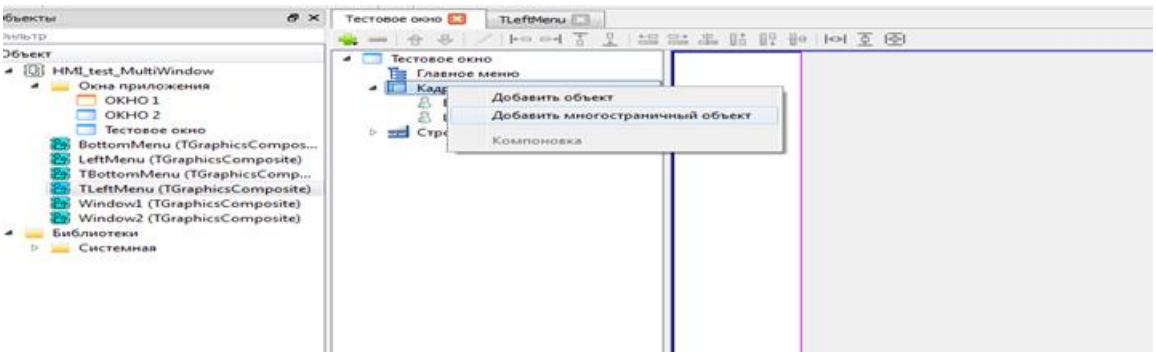


Рисунок 3.463 - Добавление центрального многостраничного объекта

13. Будет добавлено следующее меню (см. рис. 3.464).

Разместим наш объект в свободной центральной области (см. настройки на рис. 3.464)

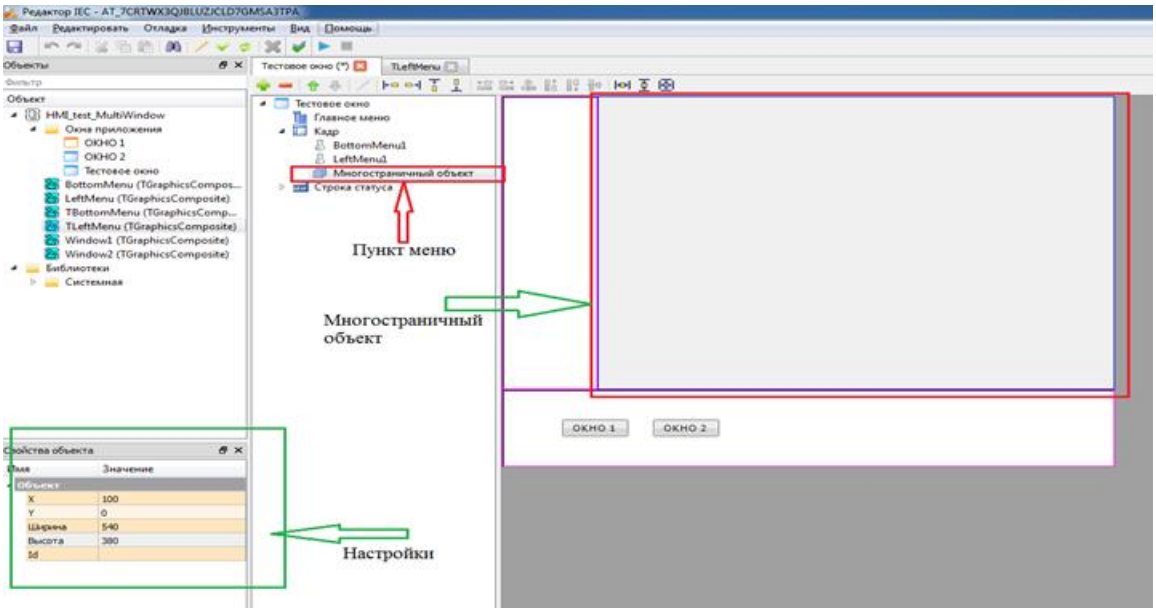


Рисунок 3.464 - Настройки свойств многостраничного объекта

14. В многостраничном объекте у нас будут два окна, которые мы планируем переключать между собой с помощью кнопок нижнего меню. Для создания окон выберите пункт Многостраничный объект и нажмите правую клавишу мыши. Дале вы выберите Добавить кадр (см. картинку рис. 3.465).

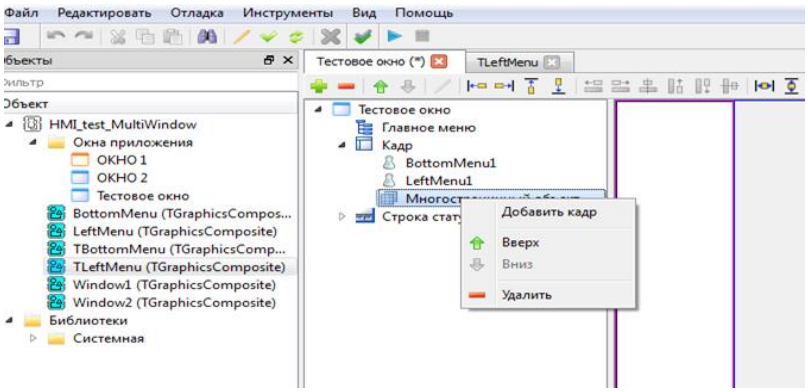


Рисунок 3.465 - Добавление кадра

Каждый кадр также может быть многостраничным объектом, но сейчас мы выбираем, что каждый кадр будет одиночным объектом (см. картинку рис. 3.466).

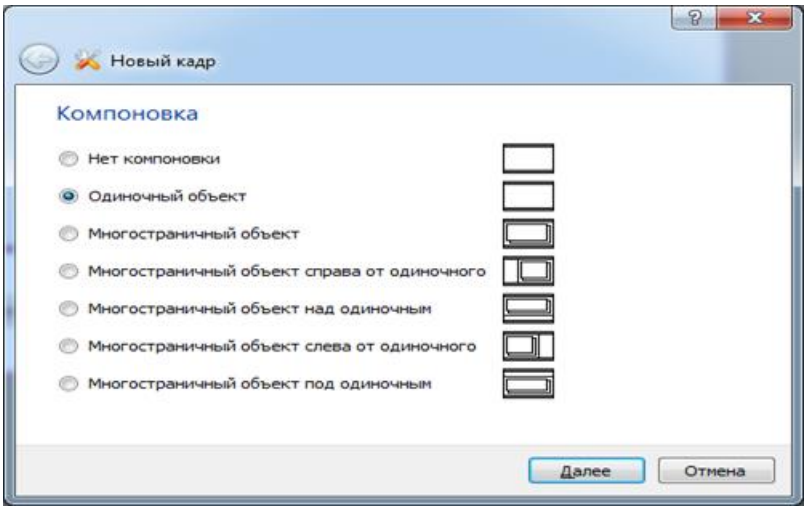


Рисунок 3.466 - Настройка кадра

Далее создаем новый тип (TWindow1) и объект (Window1).

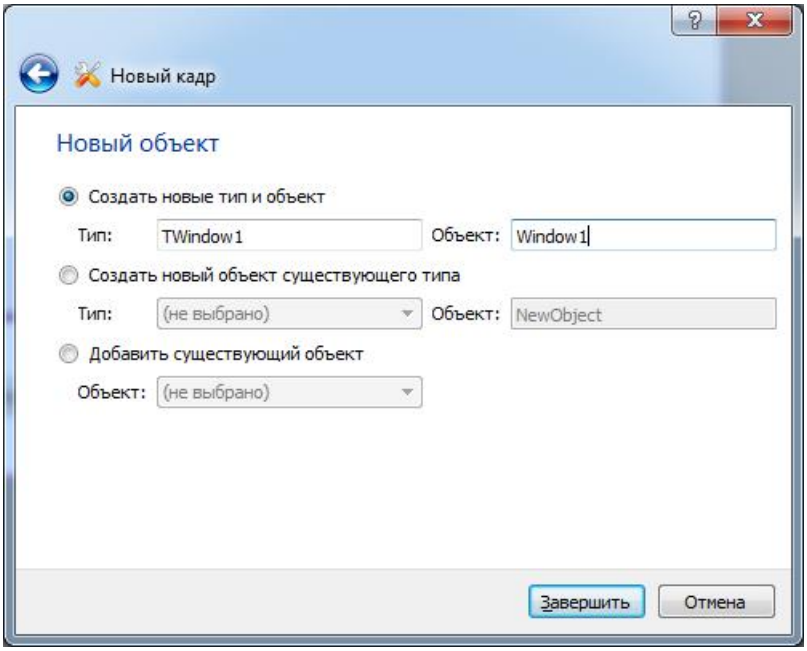


Рисунок 3.467 - Создание нового типа и объекта

Появится Кадр0 с именем Window1.

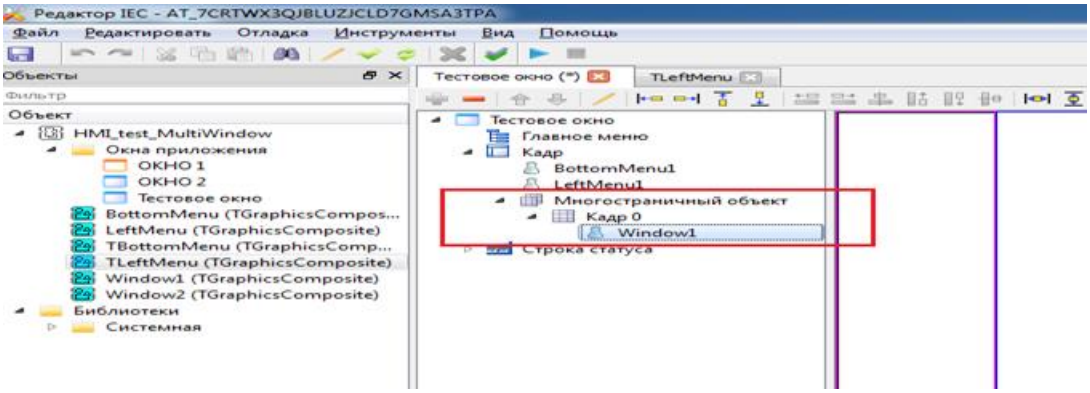


Рисунок 3.468 - Новый кадр 0

Сделайте аналогично еще один кадр (тип TWindow2, объект Window2).

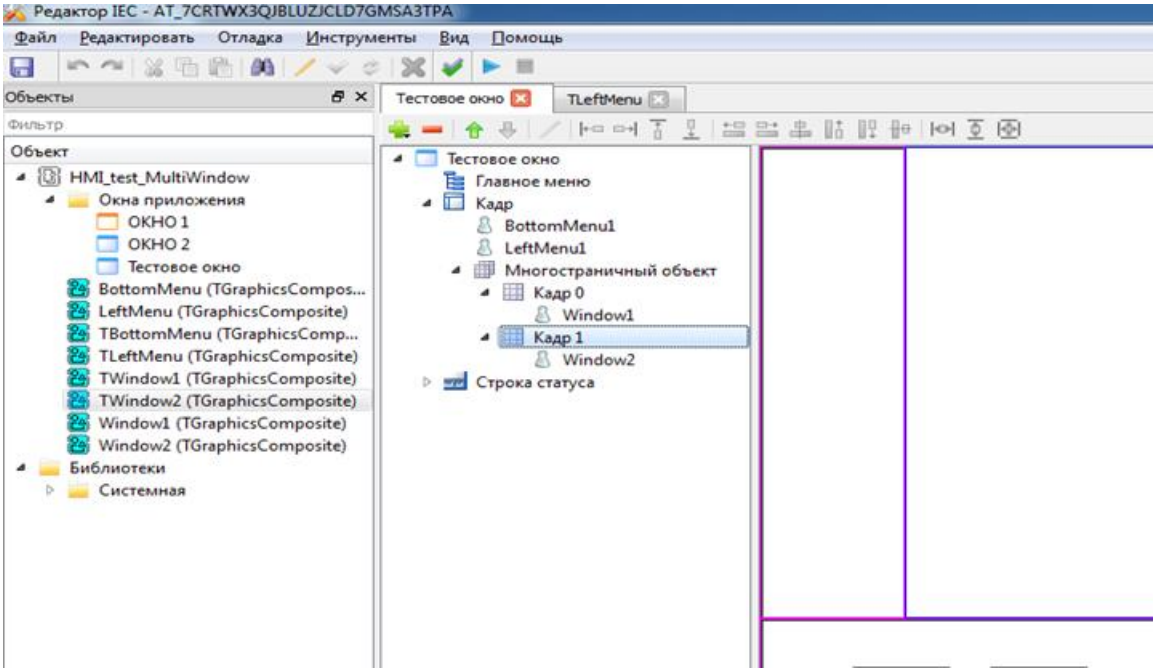


Рисунок 3.469 - Новый кадр 1

15. Чтобы кадры отличались мы добавим на них разный фон и надпись Кадр 1 и Кадр 2.

Для этого редактируем TWindow1. Для создания фона необходимо добавить на кадр прямоугольник размером с кадр (540*380) и у прямоугольника настроить свойство bg_color. Так же добавьте текст КАДР 1. Должно получиться как на картинке.

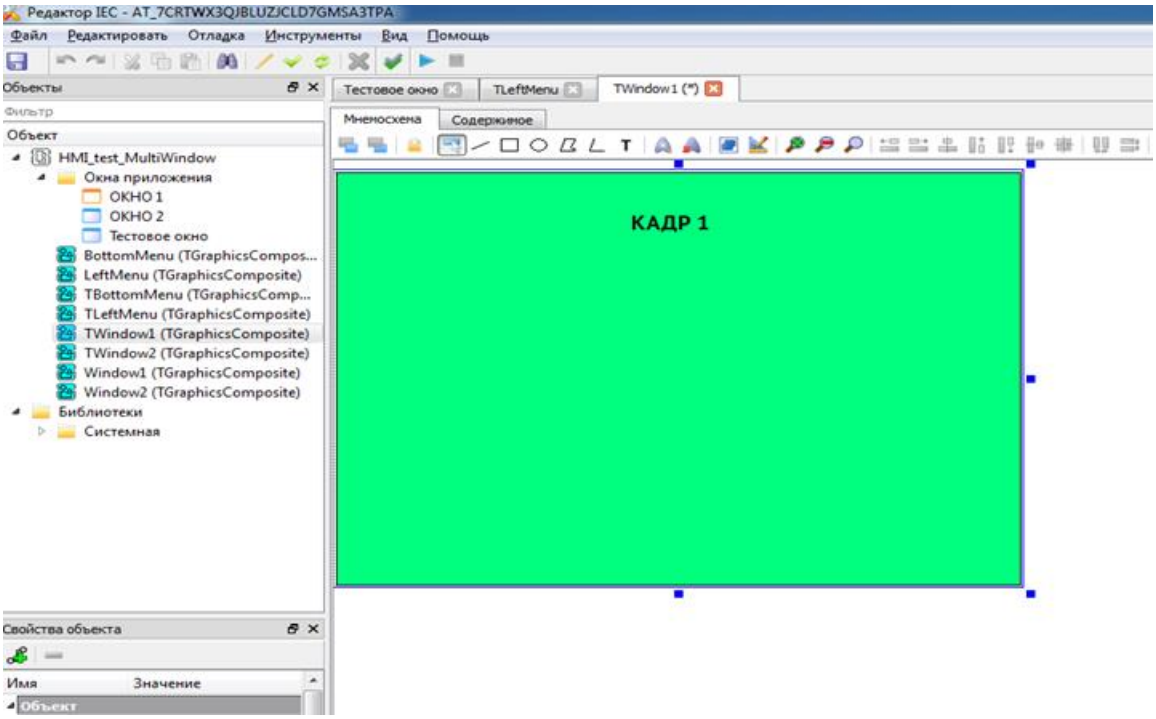


Рисунок 3.470 - Настраиваем первый кадр

Аналогичные действия сделайте с TWindow2.

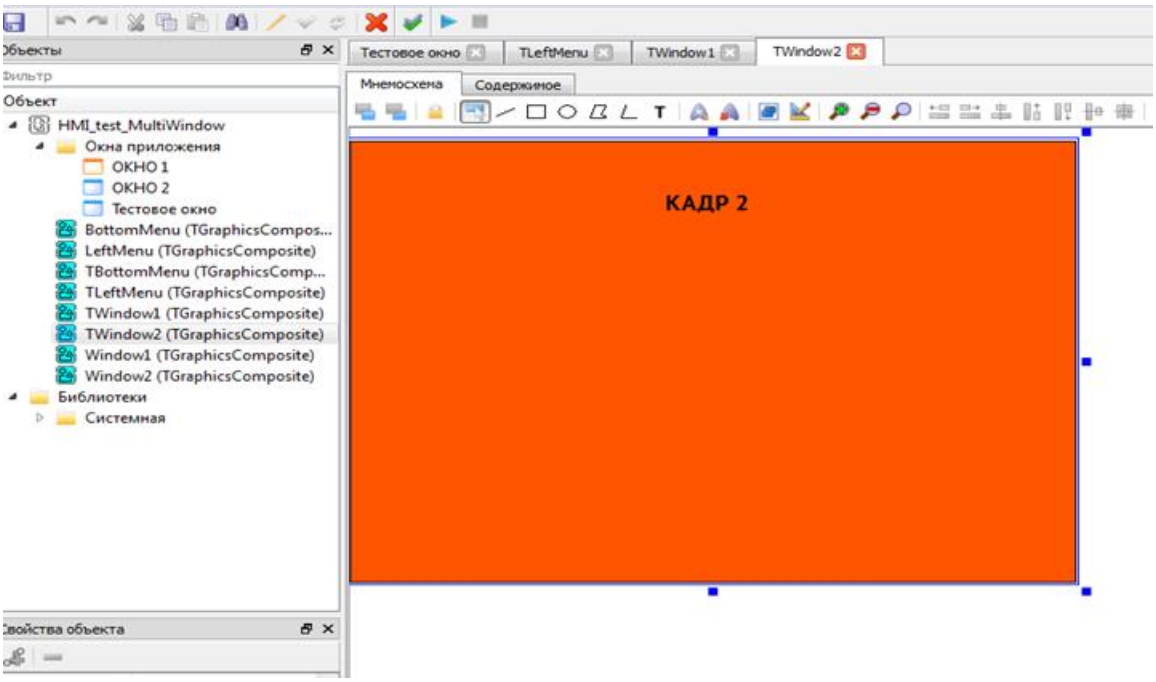


Рисунок 3.471 - Настраиваем второй кадр

16. Вернемся к нашему TBottomMenu. Нам нужно сделать чтобы кнопки могли переключать кадры между собой. Для этого нужно использовать блок Navigator (получает на вход событие и делает видимым кадр по входному значению ID).

Создадим в TBottomMenu два входных сигнала: ID_WINDOW1:STRING и ID_WINDOW2:STRING и поместим их на вход ID блока NAVIGATOR.

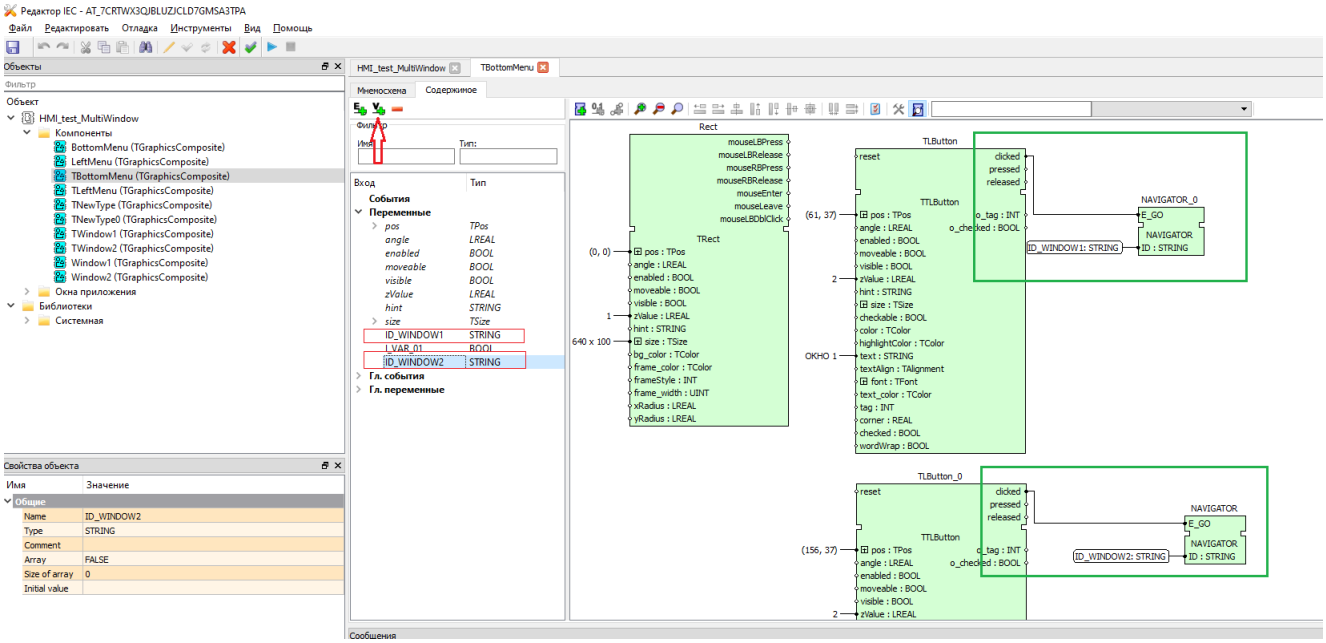


Рисунок 3.472 - Использования функционального блока Navigator для переключения между кадрами

17. У каждого многостраничного кадра (объекта) есть свой ID (см. картинку). В нашем случае у объекта Window1 ID равен Window1, а у объекта Window2 ID равен Window2.

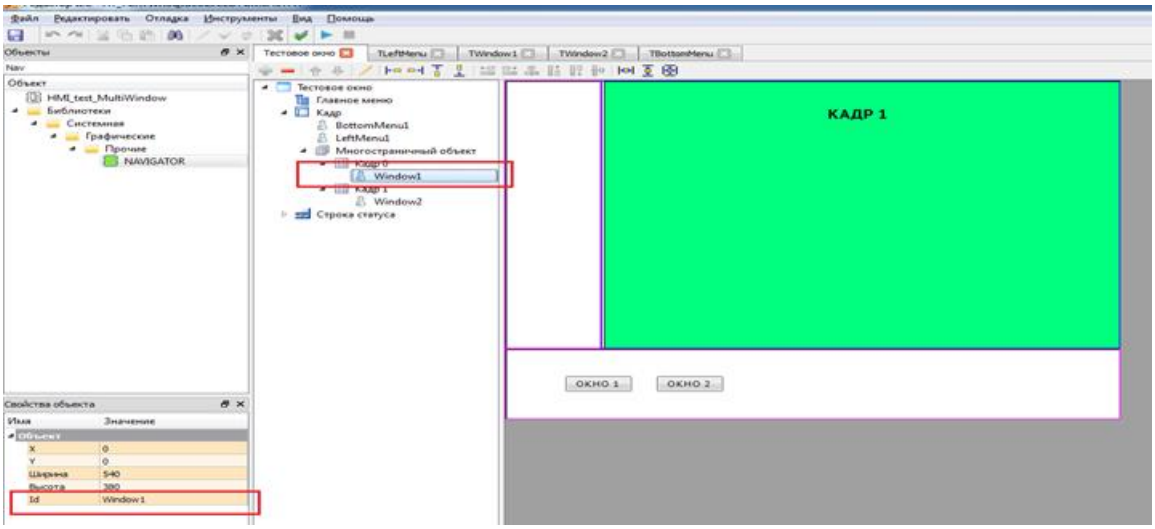


Рисунок 3.473 - ID первого кадра

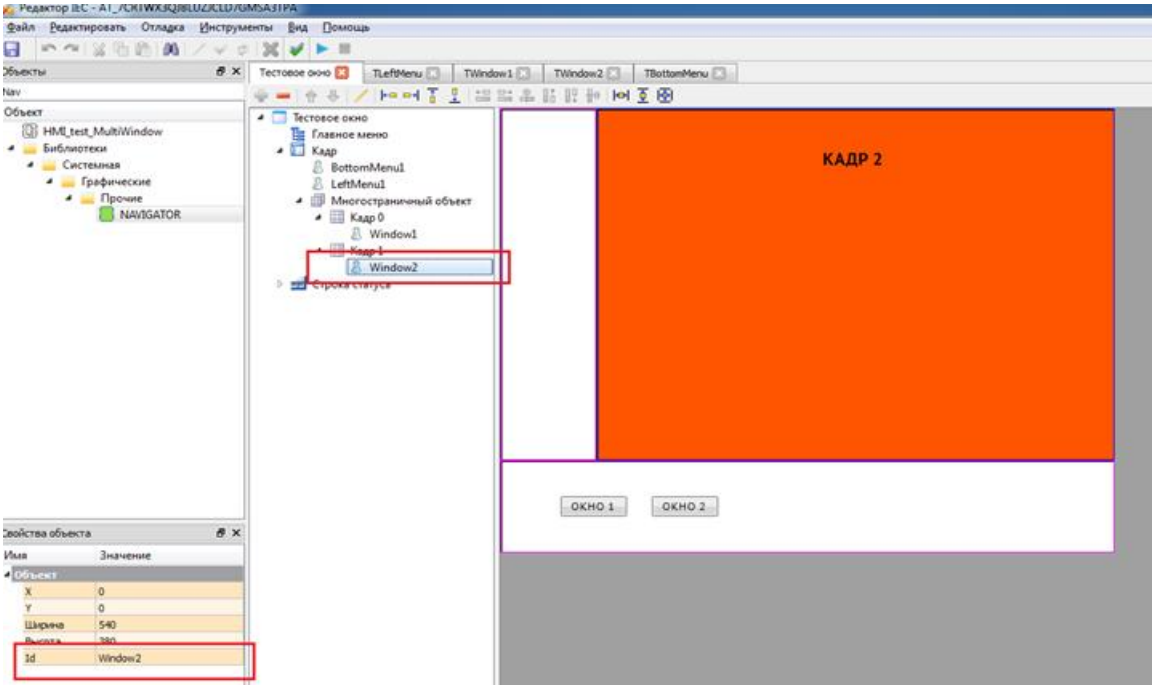


Рисунок 3.474 - ID второго кадра

Перейдем на главное окно, где располагаются все наши объекты (при работе с данными окнами они туда добавляются автоматически). В данном окне привяжем значения ID окон к входным сигналам объекта BottomMenu.

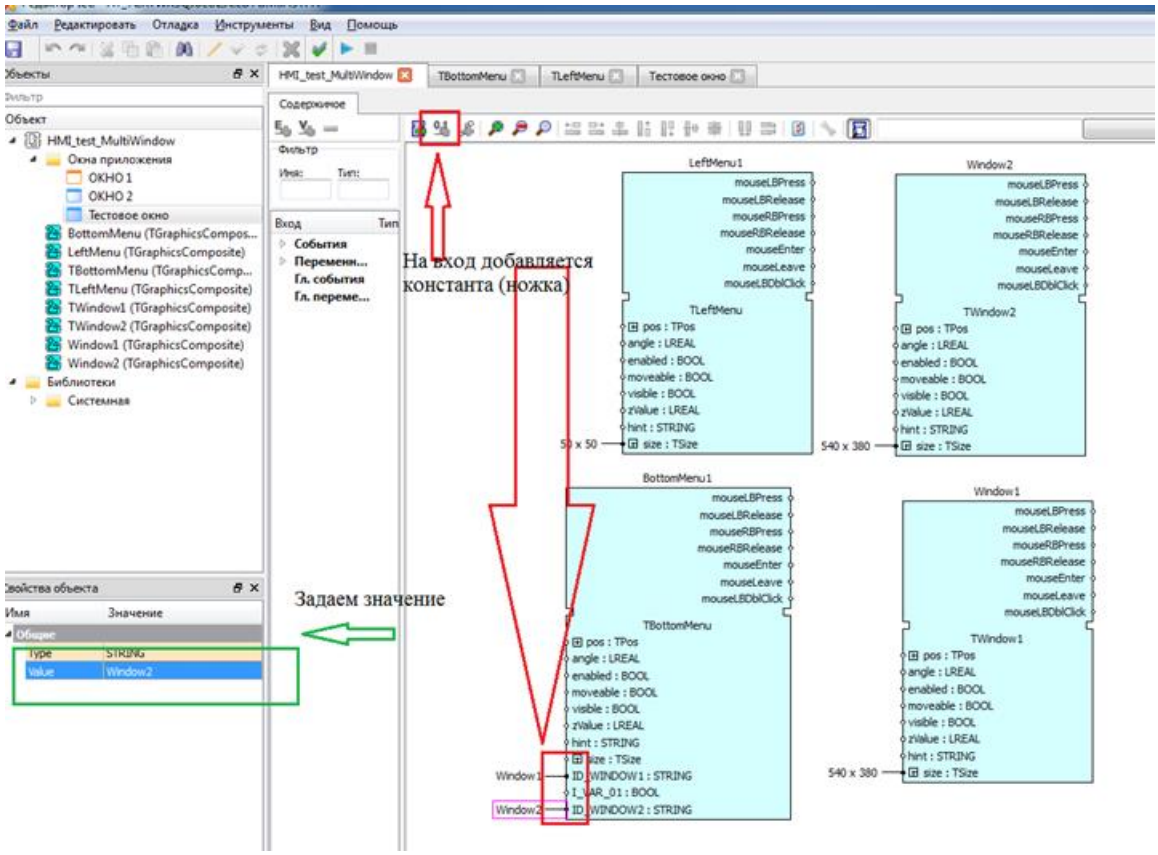


Рисунок 3.475 - Использование ID кадров

Сейчас у нас готово наше одно графическое окно. Нам нужно сделать второе, но используя уже имеющиеся типы меню и др.

18. Выбираем снова Добавить окно приложения.

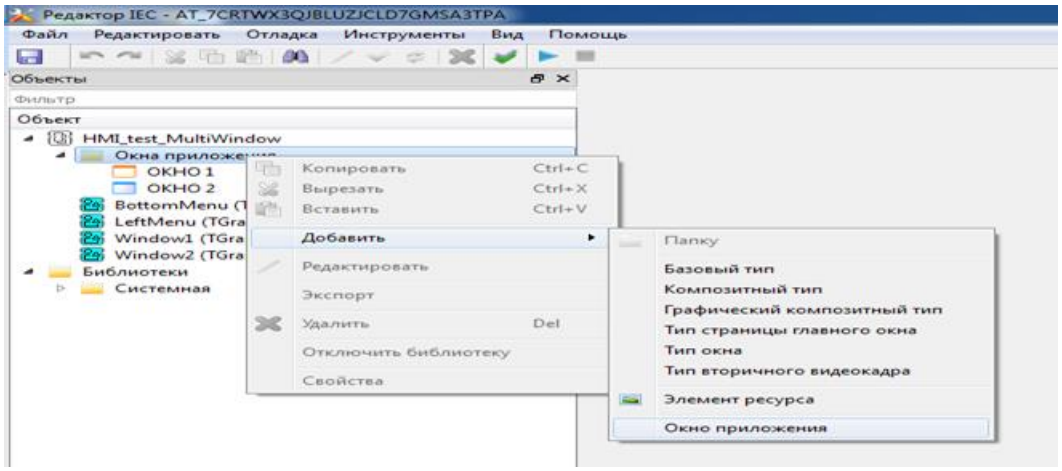


Рисунок 3.476 - Добавление нового окна приложения

Далее настраиваем как на картинках.

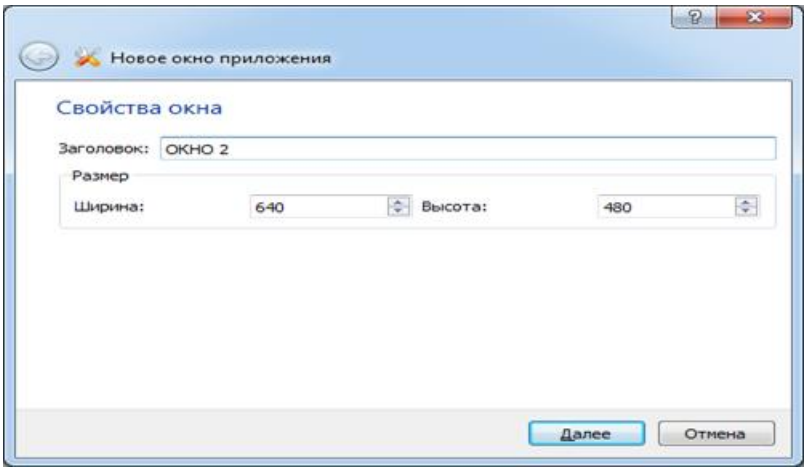


Рисунок 3.477 - Настройки свойств нового окна

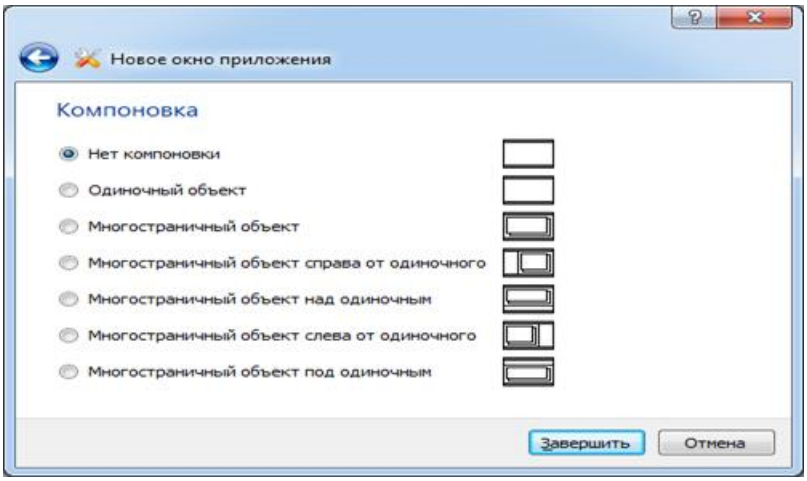


Рисунок 3.478 - Настройки компоновки нового окна

19. Далее мы добавим на второе окно нижнее меню, левое меню и многостраничный объект с двумя окнами.

Делаем это так же как раньше, но используем пункт Создать новый объект существующего типа.

а) нижнее меню

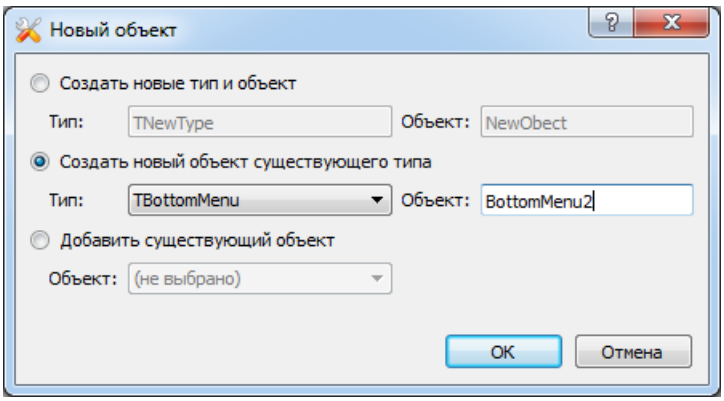


Рисунок 3.479 - Создание нижнего меню на базе существующего типа

После создания переместите его вниз.

б) левое меню

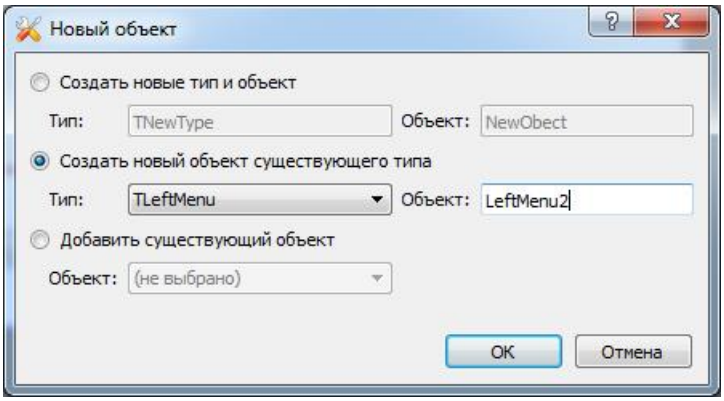


Рисунок 3.480 - Создание нлевого меню на базе существующего типа

в) создайте многостраничный объект и задайте ему размер 540*380, расположение 100*0.
г) Добавьте в многостраничный объект два кадра

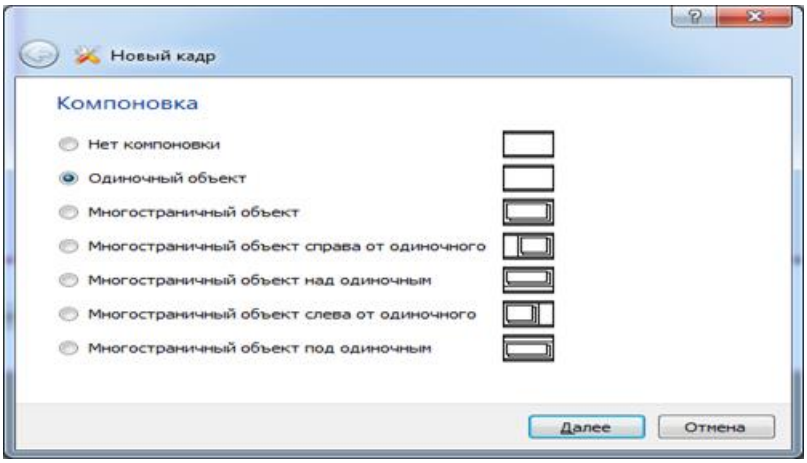


Рисунок 3.481 - Создание нового кадра

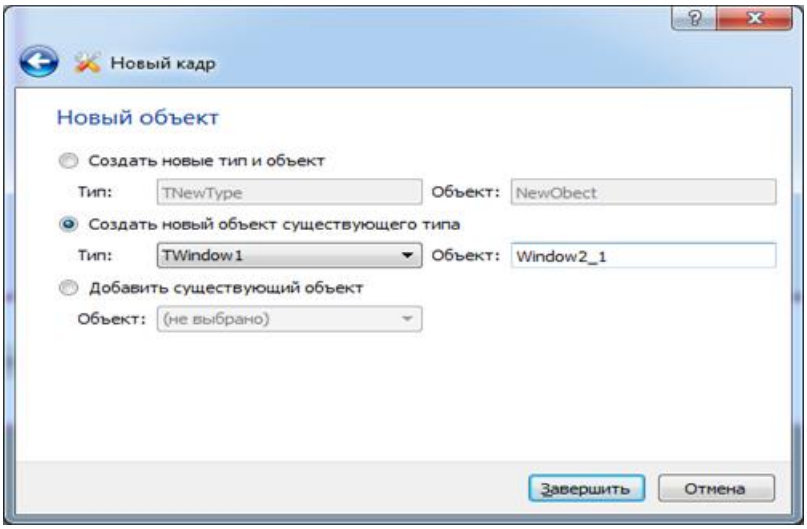


Рисунок 3.482 - Настройка нового кадра

Аналогично создайте второе окно и назовите объект Window2_2.
20. У наших новых кадров ID равны Window2_1 и Window2_2.

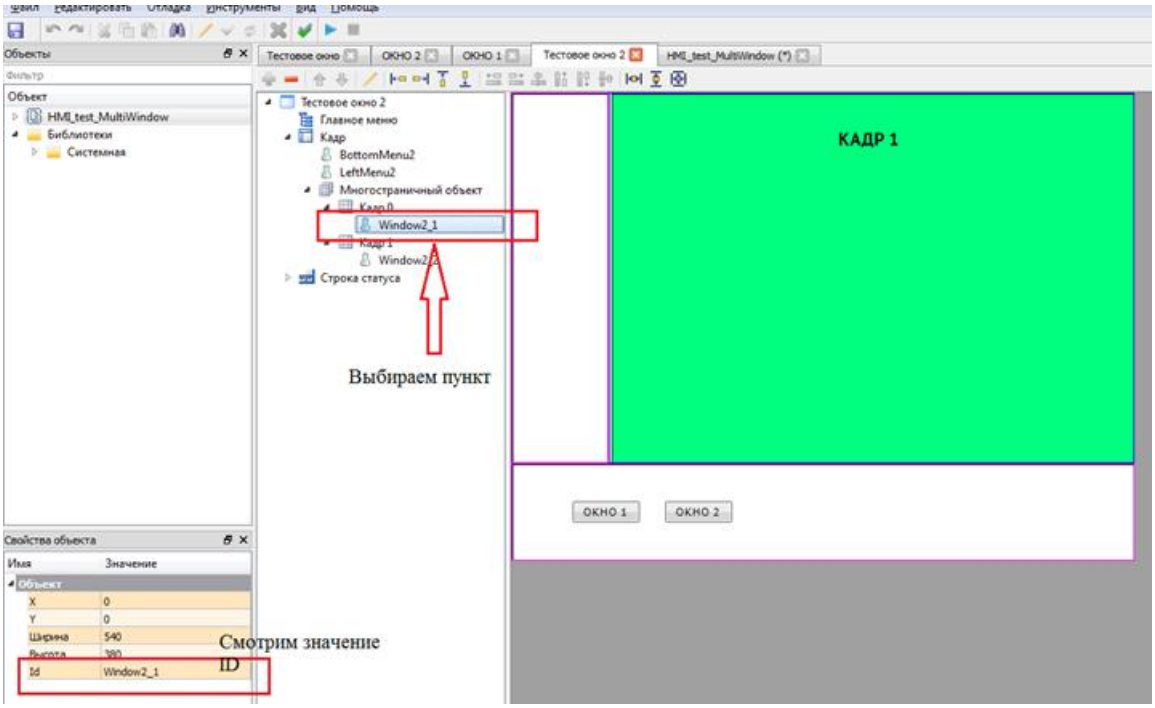


Рисунок 3.483 - Значение ID нового кадра

21. Переходим в главное окно и находим объект BottomMenu2. К его ножкам ID_WINDOW1 и ID_WINDOW2 подключаем значения ID окон (см. рис. 3.484 и рис. 3.485).

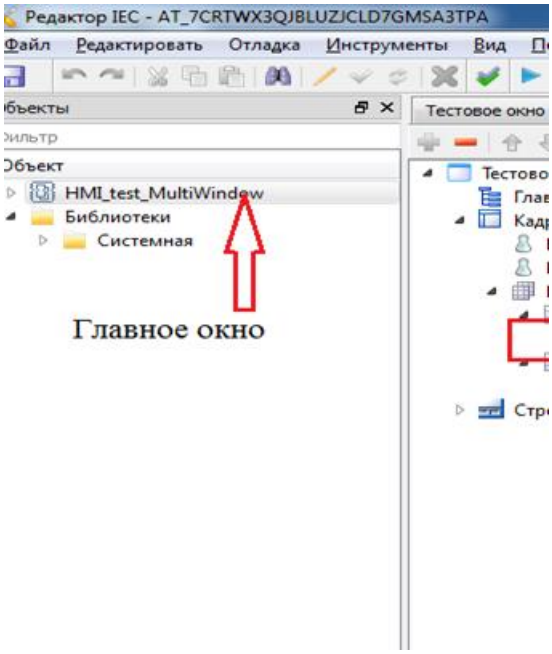


Рисунок 3.484 - Расположение меню Главного окна

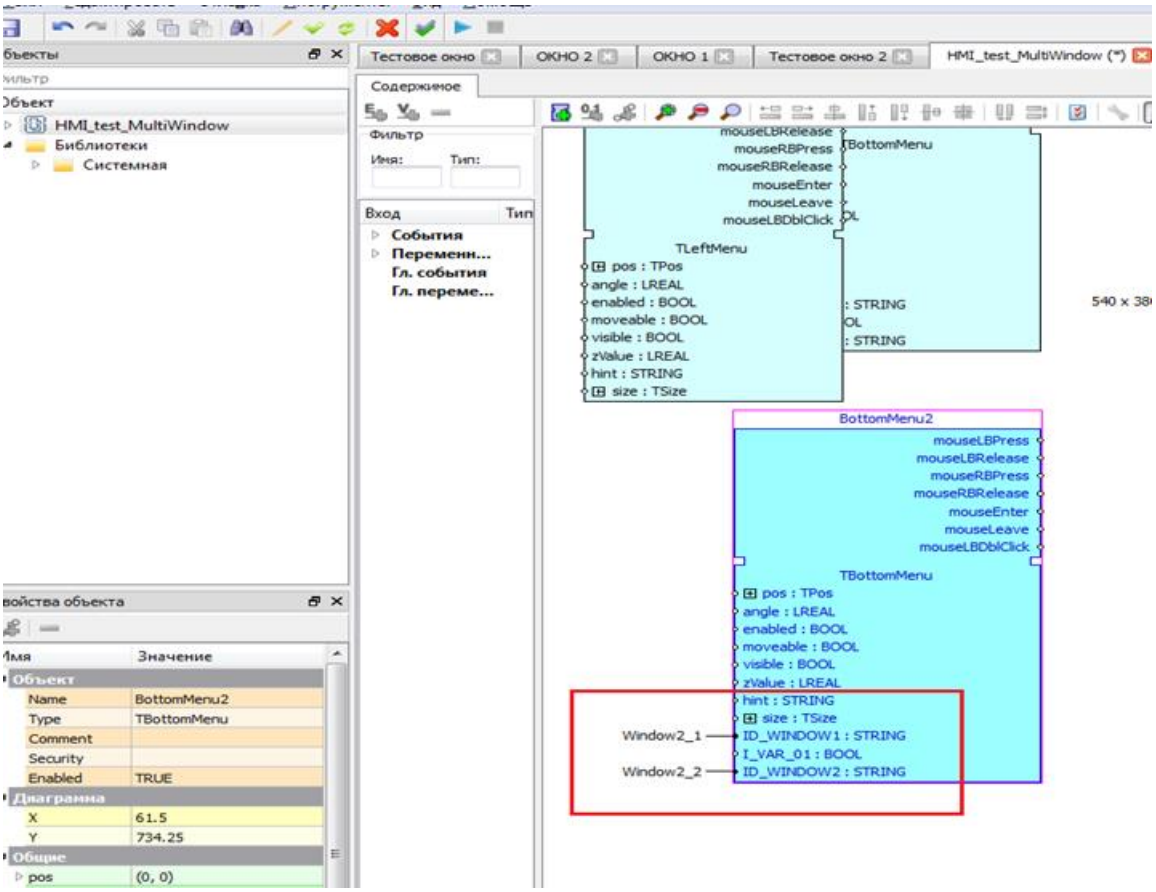


Рисунок 3.485 - Натройки ID для Window2

Сохраняем. У нас готово второе окно и оно такое же как и первое. Внесение изменений в меню или кадры будет передаваться на оба окна.

22. Теперь настроим отображение наших окон на разных мониторах. Для этого закрываем редактор IECEditor. Отрываем редактор интерфейса нашего приложения HMI_test_MultiWindow.

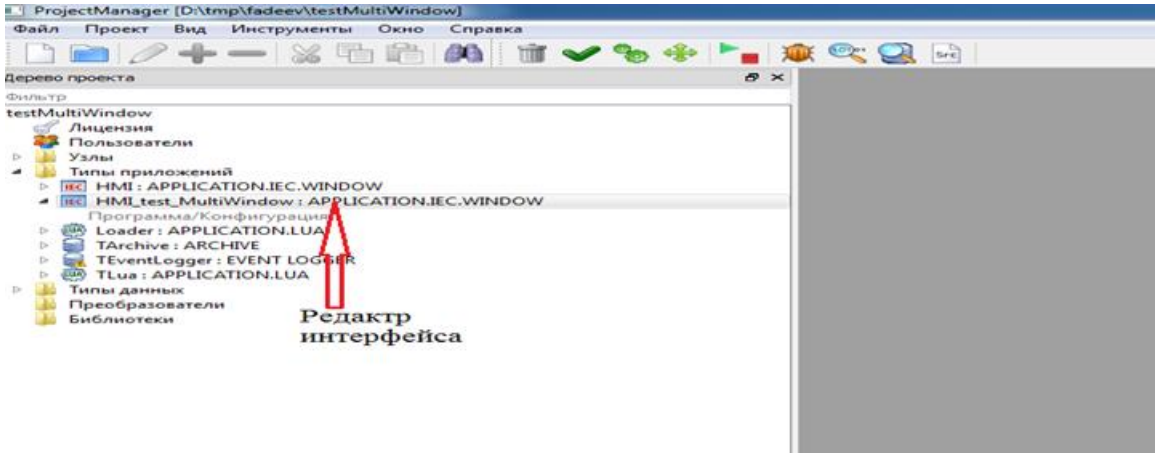


Рисунок 3.486 - Расположение редактора интерфейса графического приложения

Нам нужен системный сигнал @WINDOW.

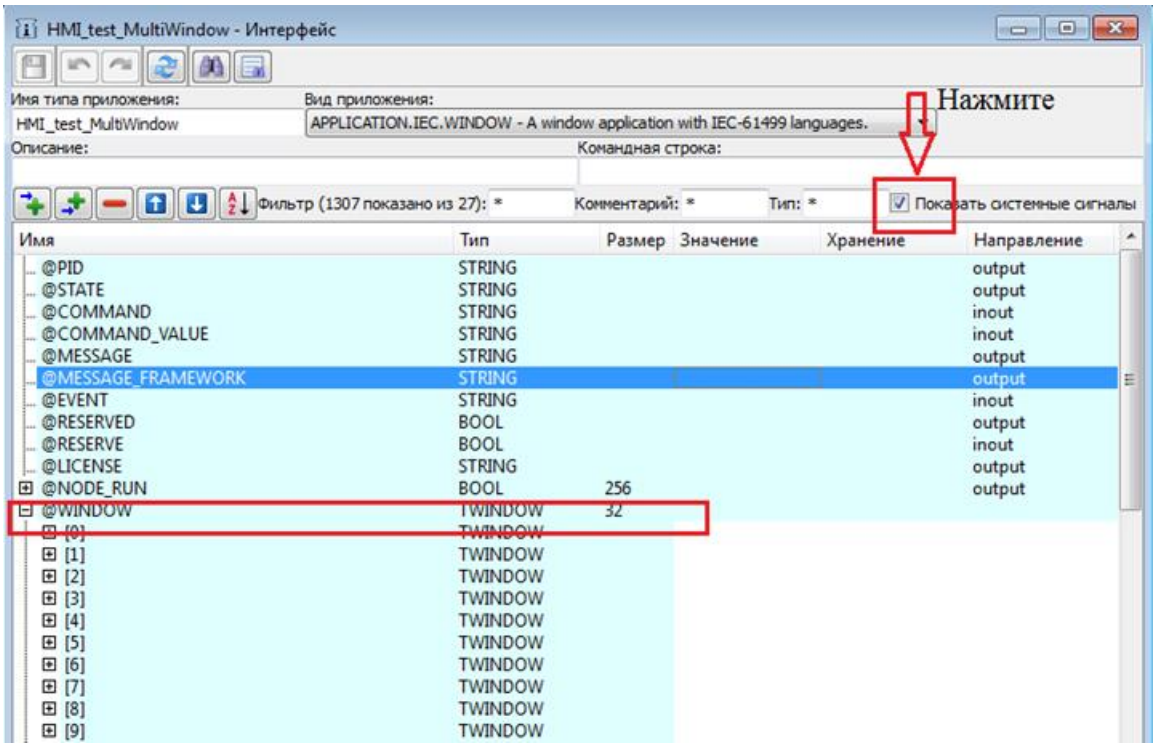


Рисунок 3.487 - Настройка интерфейса графического приложения - выбор системного сигнала @WINDOW

Системный сигнал @WINDOW - это массив, где каждая ячейка массива, это одно из наших окон. Развернем ячейки 0 и 1.

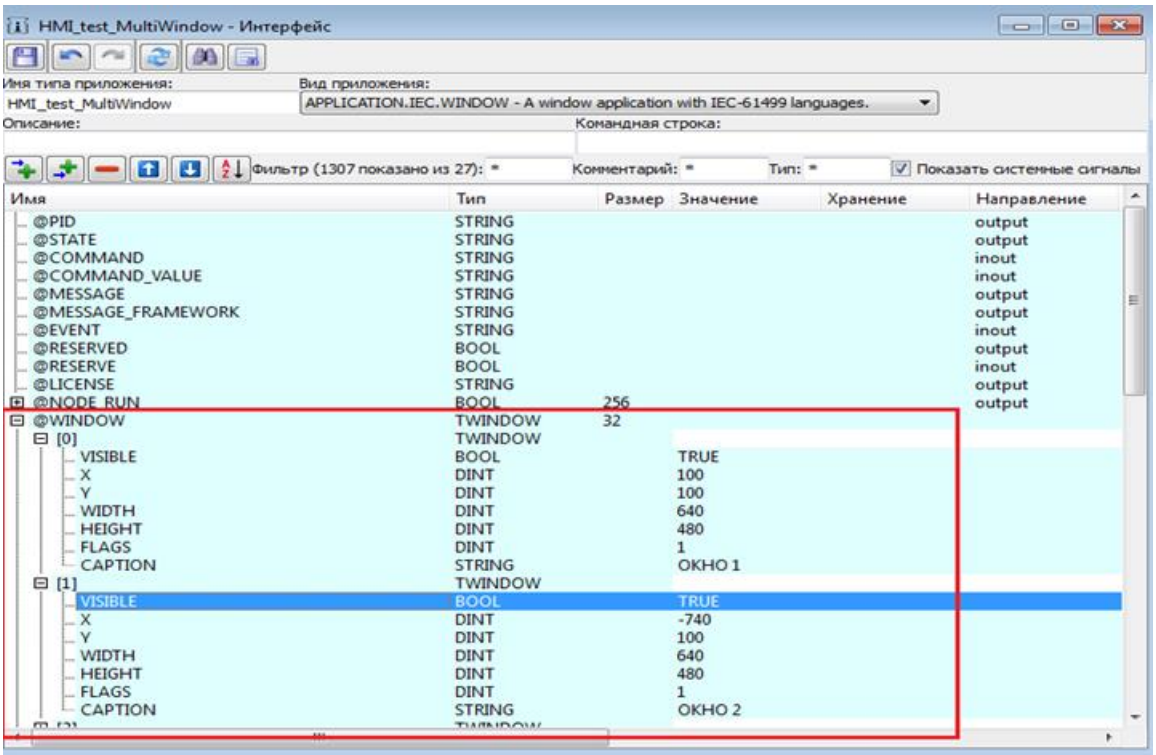


Рисунок 3.488 - Настройка интерфейса графического приложения - настройки системного сигнала @WINDOW

В каждой ячейке есть свойства окна:
- VISIBLE - видимость;

- X - расположение левого верхнего угла окна по оси x;
- Y - расположение левого верхнего угла окна по оси y;
- WIDTH - ширина окна;
- HEIGTH - высота окна;
- FLAGS - флаги окна (описаны в Описании применения);
- CAPTION - название окна.

Первое окно мы разместим на основном экране и сместим его немного x:100*y:100. Второе окно в нашем примере нужно разместить на экране, который у меня настроен слева (смотрите описание в начале), поэтому сместим его по координатам x:-740*y:100.

Видеокадры готовы.

23. В соответствии с принципами работы в СКАДЕ Соната, далее создаем узел и добавляем на него нашего графическое приложение HMI_test_MultiWindow. После запуска проекта мы получаем многооконный режим.

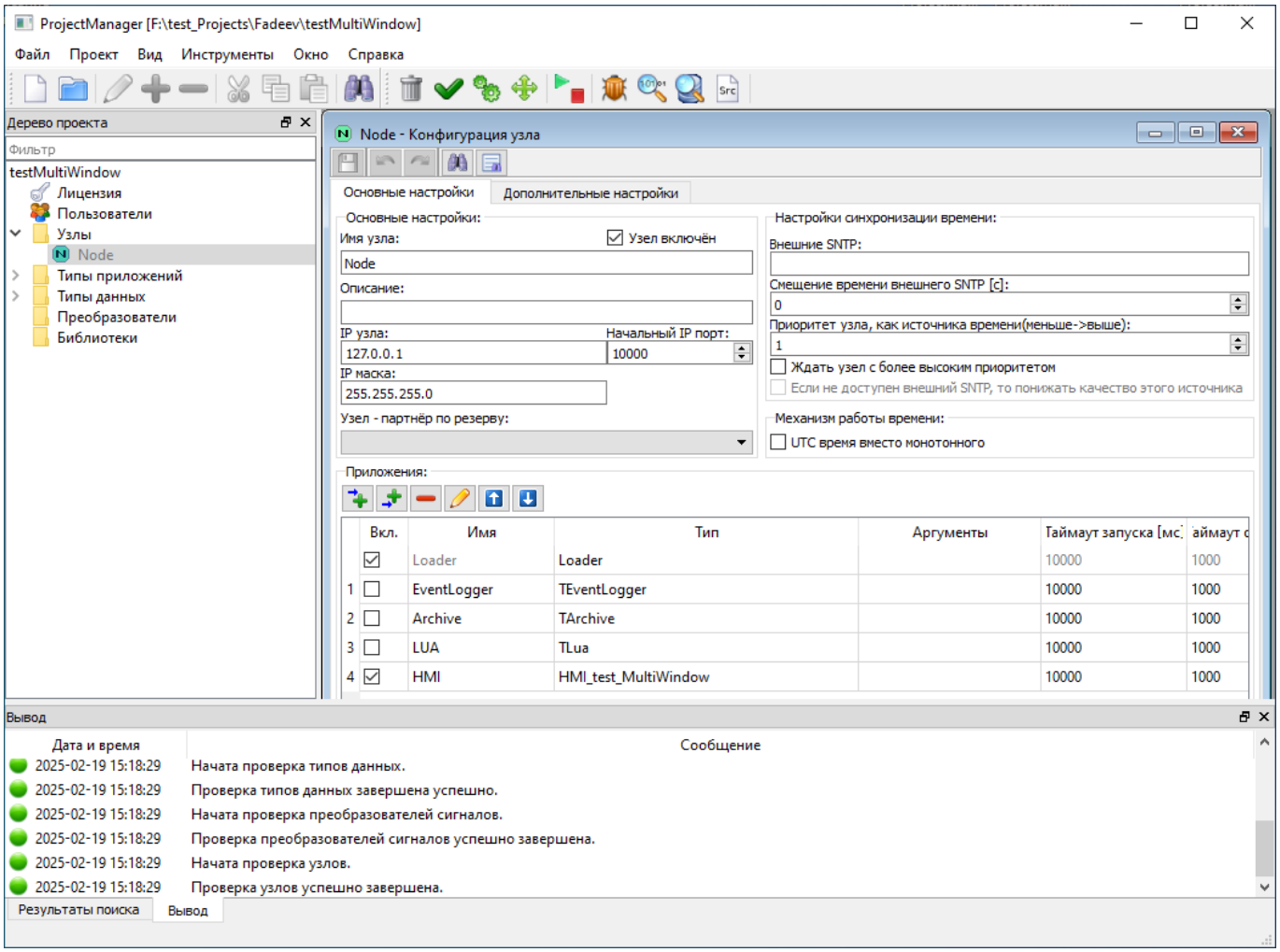


Рисунок 3.489 - Настройки узла для запуска нашего проекта с многооконным режимом

ПРИЛОЖЕНИЕ А

Таблица А.1 - Элементарные типы данных

Тип данных	**Описание**
Битовые типы данных	
BOOL	Логический тип сигналов (0 или 1, FALSE или TRUE: за истину полагается единица, за ложь – ноль)
BYTE	Байт, размер – 8 бит, значения от 0 до 255
WORD	Слово, размер – 16 бит, значения от 0 до 65 535
DWORD	Двойное слово, размер – 32 бита, значения от 0 до 4 294 967 295
Целочисленные типы данных	
USINT	Беззнаковый, размер – 1 байт, значения от 0 до 255
SINT	Знаковый, размер – 1 байт, значения от -128 до 127
UINT	Беззнаковый, размер – 2 байта, значения от 0 до 65 535
INT	Знаковый, размер – 2 байта, значения от -32 768 до 32 767
UDINT	Беззнаковый, размер – 4 байта, значения от 0 до 4 294 967 295
DINT	Знаковый, размер – 4 байта, значения от -2 147 483 648 до 2 147 483 647
ULINT	Беззнаковый, размер – 8 байт, значения от 0 до 18 446 744 073 709 551 615
LINT	Знаковый, размер – 8 байт, значения от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Вещественные типы данных	
REAL	Размер – 4 байта, значения от 1,4×10^−45 (1,2×10^−38) до 3,4×10^+38
LREAL	Размер – 8 байт, значения от 5.0×10^−324 (2,3×10^−308) до 1,7×10^+308
Типы данных, содержащие значение даты и времени	
DT	Содержит значения даты и времени, размер 8 байт
DATE	Содержит значения даты, размер 8 байт
TOD	Содержит значения времени суток, размер 8 байт

Тип данных	**Описание**
TIME	Содержит значение длительности – продолжительности промежутка времени, размер 8 байт
Строковые типы данных	
STRING	Строка длиной до 511 байт; строка хранится в формате UTF8, поэтому каждый кириллический символ занимает 2 байта

ПРИЛОЖЕНИЕ В

КОДЫ И СТАТУСЫ КАНАЛОВ, СЛОТОВ И УСТРОЙСТВ У ДРАЙВЕРОВ

0x00 (0) - ошибок нет, всё в порядке;
0x01 (1) - устройство (канала, слот) частично неисправно;
0x02 (2) - устройство (канала, слот) полностью неисправно или не отвечает;
0x04 (4) - обрыв;
0x08 (8) - короткое замыкание;
0x80 (128) - состояние устройства (канала, слот) неизвестно.

Важно

Если будет несколько неисправностей, соответствующих статусам, то их значения складываются.

ПРИЛОЖЕНИЕ С

КОДЫ ОШИБОК, ИСПОЛЬЗУЕМЫЕ ДЛЯ ДИАГНОСТИКИ В СКАДА-СИСТЕМЕ "СОНАТА"

Данные коды ошибок и сообщения используются при анализе логов работающих или неработающих приложений и в функциях языка **LUA**:

0 - ошибок нет;
1 - неизвестная ошибка;
2 - неверный аргумент функции;
3 - не удалось запустить или остановить служебный поток;
4 - не удалось инициализировать сокет;
5 - ядро уже запущено. Данная операция не может быть выполнена;
6 - тайм-аут ожидания;
7 - не найден преобразователь сигнала;
8 - такой преобразователь сигнала уже существует;
9 - преобразователь сигнала используется и не может быть изменён или удалён;
10 - запрашиваемое ядро-партнёр не найдено;
11 - описание ядра-партнёр уже существует;
12 - ядро-партнёр используется и не может быть изменено или удалено;
13 - потеряна связь с ядром-партнёром;
14 - восстановлена связь с ядром-партнёром;
15 - синхронизированы значения сигналов с ядром-партнёром;
16 - передача значения сигнала с узла наружу заблокирована;
17 - запрашиваемый сигнал не найден;
18 - выход за границы массива сигнала;
19 - такой сигнал уже существует;
20 - невозможно преобразовать значение сигнала. Ошибка в DSP;
22 - отсутствие значения сигнала;
23 - несовпадение типа сигнала с типом запрашиваемого значения;
24 - выход за допустимый диапазон дат.

ПРИЛОЖЕНИЕ D

Таблица D.1 - Системные сигналы приложений

Имя сигнала	**Тип сигнала**	**Описание**
@PID	STRING	Идентификатор процесса приложения
@STATE	STRING	Состояние приложения
@COMMAND	STRING	Команда приложению
@COMMAND_VALUE	STRING	Данные команды приложению, если таковые требуются
@MESSAGE	STRING	Сообщение приложения
@MESSAGE_FRAMEWORK	STRING	Сообщения каркаса приложения
@EVENT	STRING	События приложения. Через данный сигнал архив событий извлекает события от приложения. Также через данный сигнал происходит запрос событий из архива
@RESERVED	BOOL	Состояние резерва приложения. Если значение данной переменной равно TRUE, то приложение находится в резерве
@RESERVE	BOOL	Команды вывода приложения в резерв. Если установить значение данной переменной в TRUE, то приложение начнет процесс вывода в резерв. По завершении процесса приложение выставит значение сигнала @RESERVED в TRUE. Если установить значение данной переменной в FALSE, то приложение начнет процесс вывода из резерва. По завершении процесса приложение выставит значение сигнала @RESERVED в FALSE
@LICENSE	STRING	Уникальный номер лицензии данного приложения

Имя сигнала	**Тип сигнала**	**Описание**
@ALARM	STRING	Через данный сигнал происходит запрос тревог из архива событий
@WINDOW	TWindow[32]	Для графических приложений значение данной переменной определяет положение, размер, флаги и видимость окон приложения
@NODE_ROLE	INT	Диагностический сигнал, формируемый драйвером БПР, указывающий на роль данного узла при резервировании. 0 - роль не определена, 1 - активный по умолчанию, 2 - резервный по умолчанию
@NODE_PEER_ROLE	INT	Диагностический сигнал, формируемый драйвером БПР, указывающий на роль узла партнера по резерву: 0 - роль не определена, 1 - активный по умолчанию, 2 - резервный по умолчанию
@NODE_ERROR	BOOL	Диагностический сигнал, указывающий на ошибки на узле
@NODE_PEER_ERROR	BOOL	Диагностический сигнал, указывающий на ошибки не узле партнёре по резерву
@NODE_RUN	BOOL	Диагностический сигнал, указывающий, что все приложения на узле запустились
@NODE_PEER_RUN	BOOL	Диагностический сигнал, указывающий, что все приложения на узле партнере по резерву запустились
@FAILOVER_PRESENCE	BOOL	Данный сигнал формирует драйвер БПР и сообщает о присутствии в схеме БПР: TRUE - в схеме есть БПР, FALSE - в схеме нет БПР
@FAILOVER_ERROR	BOOL	Данный сигнал формирует драйвер БПР и сообщает об ошибках БПР: TRUE - есть ошибки БПР, FALSE - нет ошибок БПР

Имя сигнала	**Тип сигнала**	**Описание**
@FAILOVER_RESERVE	BOOL	Данный сигнал отключает меандр от контроллера к БПР, что для БПР значит, что контроллер не отвечает и БПР произведет переключение на другой контроллер, если другой контроллер в порядке: TRUE - отключить меандр к БПР, FALSE - включить меандр к БПР
@FAILOVER_RESERVED	BOOL	Данный сигнал формирует драйвер БПР и сообщает о состоянии активный/резервный на БПР для данного контроллера: TRUE - состояние резервный, FALSE - состояние активный

ПРИЛОЖЕНИЕ Е

КЛАССИФИКАЦИЯ МОДУЛЕЙ СОНЕТ

ID	Min/Max, ед.изм.	Кол-во каналов	Обозначение
0	Входной аналоговый 0 mA – 0 5 mA – 65535	4	СН-АВВ-4-5мА (изоляция канал-канал)
1	Входной аналоговый 0 mA – 0 4 mA – 13107 20mA – 65535	4	СН-АВВ-4-20мА (изоляция канал-канал)
2	Входной аналоговый -5 mV – 0 45 mV – 65535	4	СН-АВВ-4-ТП (изоляция канал-канал)
3	Входной аналоговый 8.85 mV – 0 141.95 mV – 65535	4	СН-АВВ-4-ТСП50 (изоляция канал-канал)
5	Входной аналоговый 17.70 mV – 0 283.90 mV – 65535	4	СН-АВВ-4-ТСП100 (изоляция канал-канал)
6	Входной аналоговый 39.35 mV – 0 88.35 mV – 65535	4	СН-АВВ-4-ТСМ50 (изоляция канал-канал)
7	Входной аналоговый 78.70 mV – 0 176.70 mV – 65535	4	СН-АВВ-4-ТСМ100 (изоляция канал-канал)
8	Входной аналоговый 0 Ohm – 0 470 Ohm – 65535	4	СН-АВВ-4-Р470 (изоляция канал-канал)
9	Входной аналоговый 0 V – 0 10 V – 65535	4	СН-АВВ-4-10В (изоляция канал-канал)
10	Входной аналоговый -5 mV – 0 45 mV – 65535	4	СН-АВВ-4-ТП3 (изоляция канал-канал)
11	Входной аналоговый 0 Ohm – 0 100 Ohm – 65535	4	СН-АВВ-4-Р100 (изоляция канал-канал)
12	Входной аналоговый 0 Ohm – 0 120 Ohm – 65535	4	СН-АВВ-4-Р120 (изоляция канал-канал)
21	Выходной дискретный	16	СН-ДВ-16-ОК-24В (ОК, общий провод) КУНИ.467451.010

ID	Min/Max, ед.изм.	Кол-во каналов	Обозначение
22	Выходной дискретный	8	СН-ДВ-8-Р (релейной коммутации) КУНИ.467451.009
23	Выходной дискретный	14	СН-ДВ-14-Р (релейной коммутации) КУНИ.467451.008
24	Выходной дискретный	8	СН-ДВ-8-ТР-220В (бесконтактной коммутации)
25	Выходной дискретный	8	СН-ДВ-8-ОС-24В КУНИ.467451.017
26	Выходной дискретный	16	СН-ДВ-16-ЭП-24В (эмиттерный повторитель) КУНИ.467451.018
27	Выходной дискретный	8	СН-ДВ-8-Р-НЗ КУНИ.467451.009-02
32	Входной дискретный	8	СН-ДВВ-8-24В КУНИ.467451.016
33	Входной дискретный	16	СН-ДВВ-16-24В (общий провод) КУНИ.467451.005
34	Входной дискретный	8	СН-ДВВ-8-220В КУНИ.467451.007
35	Входной дискретный	14	СН-ДВВ-14-220В (общий провод) КУНИ.467451.006
40	Выходной аналоговый 0 mA – 0 4 mA – 13107 20 mA –65535	4	СН-АВ-4-20мА (групповая изоляция) КУНИ.467439.008
41	Выходной аналоговый 0 V – 0 10 V – 65535	4	СН-АВ-4-10В (групповая изоляция)
51	Входной аналоговый 0 mA – 0 5mA – 65535	8	СН-АВВ-8-5мА (групповая изоляция)
52	Входной аналоговый 0 mA – 0 4 mA – 13107 20mA – 65535	8	СН-АВВ-8-20мА (групповая изоляция)
53	Входной аналоговый 0 V – 0 10V – 65535	8	СН-АВВ-8-10В (групповая изоляция)
54	Входной аналоговый 0 mV – 0 100mV – 65535	8	СН-АВВ-8-100мВ (групповая изоляция)
60	Входной аналоговый 0 V – 0 10V – 65535	4	СН-АВВ-4-10В КУНИ.467439.022-02
61	Входной аналоговый -5mV – 0 45mV – 65535	4	СН-АВВ-4-ТП КУНИ.467439.022-01

ID	Min/Max, ед.изм.	Кол-во каналов	Обозначение
62	Входной аналоговый 0mA – 0 4mA – 11915 20 mA – 59578 22 mA - 65535	4	CH-ABB-4-20мА КУНИ.467439.022
63	Входной аналоговый 0Ohm – 0 500Ohm – 65535	4	CH-ABB-4-P500 КУНИ.467439.023
64	Входной аналоговый 0mA – 0 20 mA – 65535	8	CH-ABB-8-20мА КУНИ.467439.024
65	Входной аналоговый 0 V – 0 10 V – 65535	8	CH-ABB-8-10В КУНИ.467439.024-01
66	Входной аналоговый 0nA – 0 100nA – 65535	7	CH-ABB-7-DPZ
67	Входной аналоговый 0mA – 0 4mA – 11915 20 mA – 59578 22 mA - 65535	4	CH-ABB-4-ТП-2 КУНИ.467439.022-06 (изоляция канал-канал)
68	Входной аналоговый 0mA – 0 4mA – 11915 20 mA – 59578 22 mA - 65535	4	CH-ABB-4-ТП-2 КУНИ.467439.022-07 (изоляция канал-канал)

ПРИЛОЖЕНИЕ F

ПЕРЕЧЕНЬ СООБЩЕНИЙ СИСТЕМЫ

Драйвер Sonet_LOCALBUS и **Драйвер Katren_LOCALBUS**	
Сообщение в логах и сообщения о событиях	**Описание**
Не удалось инициализировать оборудование	Драйвер при старте обнаружил, что оборудование контроллера неисправно. Требуется замена процессорного модуля или корзины
Устройство %1 стало недоступным	При работе драйвера оборудование контроллера стало неисправным. Требуется замена процессорного модуля или корзины. %1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера
Устройство %1 стало доступным	Драйвер инициализировал оборудование и оно полностью исправно. %1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера
Устройство %1. Неудачный вызов функции "%2". Код:%3	Данное сообщение свидетельствует о неисправности модуля ВВ. %1 – указывает на неисправный модуль. %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 – содержит название функции; %3 – содержит код аппаратной неисправности. Требуется замена модуля ВВ
Модуль %1 стал неисправным. %2	Данное сообщение свидетельствует о неисправности модуля ВВ. %1 – указывает на неисправный модуль. %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 - указывает на причину выхода модуля из строя. Требуется замена модуля ВВ
Модуль %1 стал частично неисправным. %2	Данное сообщение свидетельствует о неисправности одного или нескольких каналов модуля ВВ. Или выхода одного или нескольких каналов за пределы рабочего диапазона. %1 – указывает на неисправный модуль. %1 = D.M, где D - адрес контроллера, M -

	номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 - указывает на причину неисправности модуля. Требуется проверка корректности значений каналов модуля ВВ
Модуль %1 стал исправным	Данное сообщение свидетельствует о том, что модуль находится в исправном состоянии и значения всех подключённых каналов модуля находятся в допустимом диапазоне. %1 – указывает на исправный модуль. %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Элемент кроссконнекта %1. Не удалось найти соответствующий элемент интерфейса	Сообщение драйвера при старте, если в его конфигурации указан несуществующий сигнал. %1 – имя несуществующего сигнала
Элемент кроссконнекта %1 подключён к неизвестному каналу %2	Сообщение драйвера при старте, если в его конфигурации сигнал подключён к несуществующему каналу ВВ. %1 – имя сигнала; %2 – имя несуществующего канала ВВ. Формат: D.M.C, где D - адрес контроллера, M - номер слота, C - номер канала. D=0 – локальная корзина управляющего контроллера
Элемент кроссконнекта %1 имеет неверный тип %2	Сообщение драйвера при старте, если в его конфигурации сигнал имеет недопустимый для драйвера тип, например, STRING или DT. %1 – имя сигнала; %2 –тип сигнала
Неверный слот: %1	Сообщение драйвера при старте, если в его конфигурации оборудования указан недопустимый слот в корзине. %1 – номер неверного слота. %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Неизвестный тип модуля в слоте %1(%2)	Сообщение драйвера при старте, если в его конфигурации оборудования указан неизвестный тип модуля в слоте корзины. %1 – номер слота. %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 – тип модуля
Устройство %1 имеет неисправные модули: %2	Сообщение драйвера при изменении состояния любого из модулей ВВ на неисправное.

	%1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера; %2 – перечень неисправных модулей ВВ. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Драйвер Sonet_MODBUS и **Драйвер Katren_MODBUS**	
Сообщение в логах и сообщения о событиях	**Описание**
Не удалось открыть COM порт:%1	Драйвер при старте обнаружил, что коммуникационный порт недоступен или неисправен. Возможные причины. - неисправность оборудования; - неправильная конфигурация драйвера; - порт занят другой программой или копией драйвера
Устройство %1 стало недоступным	При работе драйвера оборудование контроллера стало неисправным. Требуется замена процессорного модуля или корзины. %1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера
Устройство %1 стало доступным	Драйвер инициализировал оборудование и оно полностью исправно. %1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера
Устройство %1. Неудачный вызов функции "%2". Код:%3	Данное сообщение свидетельствует о неисправности модуля ВВ. %1 – указывает на неисправный модуль. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 – содержит название функции; %3 – содержит код аппаратной неисправности. Требуется замена модуля ВВ
Модуль %1 стал неисправным. %2	Данное сообщение свидетельствует о неисправности модуля ВВ. %1 – указывает на неисправный модуль. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 - указывает на причину выхода модуля из строя. Требуется замена модуля ВВ

Модуль %1 стал частично неисправным. %2	Данное сообщение свидетельствует о неисправности одного или нескольких каналов модуля ВВ. Или выхода одного или нескольких каналов за пределы рабочего диапазона. %1 – указывает на неисправный модуль. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 - указывает на причину неисправности модуля. Требуется проверка корректности значений каналов модуля ВВ
Модуль %1 стал исправным	Данное сообщение свидетельствует о том, что модуль находится в исправном состоянии и значения всех подключённых каналов модуля находятся в допустимом диапазоне. %1 – указывает на исправный модуль. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Элемент кроссконнекта %1. Не удалось найти соответствующий элемент интерфейса	Сообщение драйвера при старте, если в его конфигурации указан несуществующий сигнал. %1 – имя несуществующего сигнала
Элемент кроссконнекта %1 подключён к неизвестному каналу %2	Сообщение драйвера при старте, если в его конфигурации сигнал подключён к несуществующему каналу ВВ. %1 – имя сигнала; %2 – имя несуществующего канала ВВ. Формат: D.M.C, где D - адрес контроллера, M - номер слота, C - номер канала. D=0 – локальная корзина управляющего контроллера
Элемент кроссконнекта %1 имеет неверный тип %2	Сообщение драйвера при старте, если в его конфигурации сигнал имеет недопустимый для драйвера тип, например, STRING или DT. %1 – имя сигнала; %2 –тип сигнала
Неверный слот: %1	Сообщение драйвера при старте, если в его конфигурации оборудования указан недопустимый слот в корзине. %1 – номер слота. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Устройство %1 имеет неисправные модули: %2	Сообщение драйвера при изменении состояния любого из модулей ВВ на неисправное.

	%1= D - адрес контролллера. Если D=0, то это локальная корзина управляющего контроллера; %2 – перечень неисправных модулей ВВ. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Неизвестный тип модуля в слоте %1(%2)	Сообщение драйвера при старте, если в его конфигурации оборудования указан неизвестный тип модуля в слоте корзины. %1 – номер слота. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера; %2 – тип модуля
Пробуем переключиться с основного контроллера %1 на резервный %2	Драйвер обнаружил, что основной контроллер имеет неисправные каналы или модули, в то время как резервный контроллер исправен. Драйвер посылает основному контроллеру команду на переключение на резервный. %1 – адрес основного контроллера; %2 – адрес резервного контроллера
Пробуем переключиться с резервного контроллера %d на основной %d	Драйвер обнаружил, что текущий резервный контроллер имеет неисправные каналы или модули, в то время как основной контроллер исправен. Драйвер посылает текущему резервному контроллеру команду на переключение на основной. %1 – адрес резервного контроллера; %2 – адрес основного контроллера
Устройство %1 стало активным	При успешном переключении контроллеров драйвер сообщает о том, какой контроллер стал активным. %1 – адрес активного контроллера
Устройство %1 стало резервным	При успешном переключении контроллеров драйвер сообщает о том, какой контроллер стал резервным. %1 – адрес резервного контроллера
Устройство %1. Неверное состояние БПР	Драйвер обнаружил недопустимое состояние Блока Переключения Резерва. Например, оба контроллера сообщают, что они активные или резервные. Такое состояние кратковременно допустимо, если опрос контроллеров производился непосредственно в момент переключения
Сообщения в Центре управления	**Описание**

Недоступные устройства: %1	Перечень устройств, которые полностью неисправны и требуют замены. %1 – перечень устройств. Формат %1 = D, где D - адрес контроллера. Если D=0, то это локальная корзина управляющего контроллера
Неисправные модули: %1	Перечень модулей ВВ, которые полностью неисправны и требуют замены. %1 – перечень модулей. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Частично неисправные модули:%1	Перечень модулей ВВ, имеющих один или несколько неисправных каналов.Требуется поверка всех каналов данного модуля ВВ. %1 – перечень модулей. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Отсутствие питания на модулях:%1	Перечень выходных дискретных модулей ВВ, на которые не поступает внешнее питание. %1 – перечень модулей. Формат %1 = D.M, где D - адрес контроллера, M - номер слота. Если D=0, то это локальная корзина управляющего контроллера
Неисправные каналы:%1	Перечень неисправных каналов. %1 – перечень каналов. Формат: D.M.C, где D - адрес контроллера, M - номер слота, C - номер канала. D=0 – локальная корзина управляющего контроллера
Сообщения о событиях в системе	
Сообщение	**Описание**
Вход пользователя	Был произведен вход пользователя в систему
Выход пользователя. Сессия начата в чч:мм:сс дд.мм.гггг	Был произведён выход пользователя из системы
Приложение перешло в состояние "РАБОТАЕТ"	Запустился основной цикл приложения
Приложение стало доступным. %1	Приложение стало принимать и отправлять изменения сигналов
Приложение стало недоступным. %1	Приложение перестало принимать и отправлять изменения сигналов
Сообщения о событиях, связанных с администрированием пользователей в системе	
Сообщение	**Описание**
Удалена учетная запись пользователя %1	Учетная запись пользователя была удалена.

	%1 - имя пользователя
Добавлена учетная запись пользователя %1	Была добавлена новая учетная запись пользователя. %1 - имя пользователя
Учетная запись пользователя %1 была изменена (заблокирована)	Учетная запись пользователя была заблокирована. %1 - имя пользователя
Учетная запись пользователя %1 была изменена (разблокирована)	Учетная запись пользователя была разблокирована. %1 - имя пользователя
Изменена парольная политика	Изменена парольная политика для всех пользователей
Добавлена группа %1	Группа была добавлена. %1 - имя группы
Удалена группа %1	Группа была удалена. %1 - имя группы
Изменена группа %1 (изменены права)	Изменены настройки прав у группы. %1 - имя группы
Учетная запись пользователя %1 была изменена (изменены права)	У пользователя изменены права. %1 - имя пользователя
Учетная запись пользователя %1 была изменена (изменён набор групп)	У пользователя изменен набор групп в которые он входит. %1 - имя пользователя
Учетная запись пользователя %1 была изменена	У пользователя изменены настройки блокировки после определенного количества неправильных вводов пароля. %1 - имя пользователя
Сообщения в приложении Distributer (приложение, распространяющее проект на узлы)	
Доступ запрещён (403)	Распространение проекта запрещено при стартующем или останавливающимся узле. Возможна ситуация, когда из-за какой-либо ошибки в приложениях, в ходе старта узла, он не запустился до конца. Это состояние также считается состоянием старта узла, при котором распространение проекта запрещено. Для выхода из этой ситуации нужно из центра управления подать команду на останов проекта

ПРИЛОЖЕНИЕ G

ОСОБЕННОСТИ РАБОТЫ СО SCADA СИСТЕМОЙ "СОНАТА"

1. В Qt есть проблема с отображением многострочного текста в SVG. При переносе SVG файла в ресурсы SCADA системы "Соната" многострочный текст отображается одной строкой. Так ведёт себя библиотека QtSVG и это не связано с особенностями SCADA системы "Соната".

При необходимости отображать текст в SVG как многострочный, его необходимо изначально разбивать на несколько отдельных элементов текста и размещать их один под другим.

2. Если планируется работа Видеокадра на ОС QNX, то при разработке кадров не нужно пользоваться типом Тип окна (TWindow). Данное окно реализовано на библиотеках Qt, которые на ОС QNX вызывают видимые задержки в работе.

Для решения данной проблемы на библиотеках Си++ было написано окно Тип вторичного видеокадра (TSubWindow).

3. Если планируется работа Видеокадра в Астра Линукс, то не нужно пользоваться флагом окна = 4 (нельзя изменять размер окна). При использовании данного флага проявляются зависания видеокадра при перемещении мыши.

ПРИЛОЖЕНИЕ Н

Особенности работы контроллеров СН-МП-РС104-2, СН-МП-РС104-3 и контроллеров "КАТРЕН" со SCADA системой "СОНАТА"

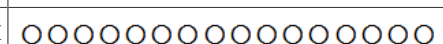
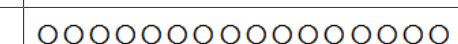
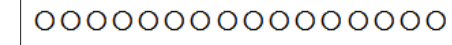
Контроллеры "Катрен" и "Сонет" СН-МП-РС104-2, СН-МП-РС104-3 оснащены статусными светодиодами, которые предназначены для отображения текущего состояния узла.

Контроллер "Катрен" имеет три светодиода:

- "Работа" (двухцветный);
- "Резерв";
- "Питание".

Контроллеры "Сонет" СН-МП-РС104-2, СН-МП-РС104-3 имеют один светодиод: "Работа" (одноцветный).

Светодиод "Работа" может отображать следующие состояния узла:

Состояние узла	**Катрен**	**Сонет на Arbor** (СН-МП-РС104-2, СН-МП-РС104-3)
Операционная система или загрузчик узла не стартовали	 Светодиод выключен	 Светодиод выключен
Загрузчик узла запущен, узел не стартовал	 Светодиод постоянно горит красным	 Светодиод выключен
Загрузчик запускает узел	 Светодиод быстро мигает зелёным	 Светодиод быстро мигает зелёным
Узел исправно работает	 Светодиод горит долго зелёным и периодически выключается на короткое время	 Светодиод горит долго зелёным и периодически выключается на короткое время
Узел работает, но некоторые программы имеют ошибки или предупреждения	 Светодиод горит долго зелёным, затем идут 4 короткие вспышки красного цвета	 Светодиод горит долго зелёным, затем идут 4 коротких выключения

ПРИЛОЖЕНИЕ I

Конфигурирование процессорного модуля МЦПВ контроллера "КАТРЕН"

Назначение DIP-переключателей

Положение переключателей по умолчанию

1	2	3	4	5	6	7	8
ON	**OFF**	**OFF**	**ON**	**ON**	**ON**	**OFF**	**OFF**

Режимы загрузки

Положение переключателя		Назначение
7	**8**	
OFF	**OFF**	Штатный режим. Загрузка операционной системы и автоматический запуск SCADA системы «Соната».
OFF	**ON**	Загрузка только операционной системы.
ON	**OFF**	Загрузка операционной системы с IP адресами по умолчанию. ETH1: 192.168.1.248 , ETH2: 192.168.2.248
ON	**ON**	Зарезервировано

Выбор загрузочного устройства

Положение переключателя			Устройство загрузки
1	**2**	**3**	
ON	**OFF**	**OFF**	SD Card. Штатный режим.
OFF	**ON**	**ON**	eMMC Flash на модуле SMARC
ON	**ON**	**ON**	SPI на модуле SMARC

Переключатели **4, 5, 6** предназначены для служебного использования и в рабочем режиме должны быть установлены в положение **ON**.

Заводские настройки по умолчанию

IP адреса по умолчанию

ETH1: 192.168.1.248
ETH2: 192.168.2.248

Пользователи по умолчанию

Пользователь “sonata”
Login: **sonata**, Password: **sonata**
Пользователь “root”
Login: **root**, Password: **superuser**

Удаленное подключение
Только по SSH для пользователя **sonata**

Установка IP адресов
Для установки IP адресов процессорного модуля необходимо подлючиться к нему по SSH с именем пользователя **sonata**.

Отредактировать файлы конфигурации IP адресов, указав в них требуемый адрес и маску локальной сети.

Для интерфейса ETH1:
$ **sudo mcedit /etc/ip_addres1**
Для интерфейса ETH2:
$ **sudo mcedit /etc/ip_addres2**

Для применения изменений выполнить перезагрузку:
$ **sync**
$ **sudo reboot**

ПРИЛОЖЕНИЕ J

Аргументы, которые можно использовать при запуске приложений СКАДА-системы "СОНАТА"

Аргумент	Описание
-noauth	Данный аргумент позволяет запускать приложения, которые требуют авторизации, без данной опции, т.е. приложение запустится без ввода логина и пароля. Данный аргумент работает только при наличии лицензии на SCADA-систему "СОНАТА"
-user=имя_пользователя -password=пароль	Данные аргументы позволяют запускать приложения, которые требуют авторизации, с указанными именем пользователя и паролем. Данный аргумент работает только при наличии лицензии на SCADA-систему "СОНАТА"
-log_level=уровень	Данный аргумент позволяет указать режим логирования информации для приложений. Используются следующие значения: - 0 - вывод общей диагностической информации о работе приложения. Данное значение настроено по умолчанию и можно не использовать аргумент; - -1 - отключение всех логов у приложения. Данное значение иногда требуется чтобы снизить нагрузку на операцию записи на жесткий диск или др. устройство, т.к. логирование - это запись в файл.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопр. документа и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					
1		все			314	35001-19			02.19