Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

А. В. Ковалев, Д. А. Литвинов, О. М. Ростокина

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ

ПРАКТИКУМ
для студентов специальности
1-53 01 07 «Информационные технологии
и управление в технических системах»
дневной формы обучения

Гомель 2022

УДК 004(075.8)
ББК 32.965я73
К56

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 7 от 01.03.2021 г.*)

Рецензент: проф. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
д-р техн. наук, доц. *И. А. Мурашко*

Ковалев, А. В.

К56 Компьютерные технологии проектирования систем автоматического управления : практикум для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах». днев. формы обучения / А. В. Ковалев, Д. А. Литвинов, О. М. Ростокина. – Гомель : ГГТУ им. П.О. Сухого, 2022. – 164 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Практикум посвящен изучению SCADA-системы TRACE MODE для проектирования систем автоматического управления. Дается краткое описание и основные понятия системы TRACE MODE.

Для студентов специальности 1-53 01 07 «Информационные технологии и управление в технических системах» дневной формы обучения.

УДК 004 (075.8)
ББК 32.965я73

СОДЕРЖАНИЕ

Лабораторная работа №1
Изучение среды TRACE MODE 6. Создание простого проекта

Цель работы: Изучить основные элементы проекта, освоить технологию создания операторского интерфейса с использованием механизма автопостроения.

1. Теоретические сведения

1.1 Разработка графического интерфейса оператора

Пакет TRACE MODE 6 реализует графическое представление хода выполнения техпроцесса, а также управление техпроцессом с помощью графических средств. Графический интерфейс оператора реализуется в трех видах:

1. В виде набора **графических экранов**, шаблоны которых разрабатываются в **редакторе представления данных** (РПД) – для узлов, которые исполняются мониторами на аппаратных средствах, имеющих достаточную производительность и другие необходимые характеристики (например, при использовании объемной графики от видеосистемы требуется поддержка OpenGL 1.1). В состав TRACE MODE 6 входит большое количество ресурсов – текстов, изображений, видеоклипов, различных графических объектов – которые могут использоваться при разработке графических экранов.;
2. В виде набора **графических панелей**, шаблоны которых разрабатываются в РПД – для узлов, которые исполняются мониторами на аппаратных средствах, имеющих ограниченную производительность (например, в контроллерах с ОС Windows CE);
3. В виде **мнемосхем** – для узлов, исполняемых мониторами в среде DOS.

Интегрированная среда (ИС) разработки объединяет в единой оболочке навигатор и набор редакторов для создания всех составляющих проекта. В навигаторе структура проекта представлена в виде дерева. Корневые группы этого дерева (слои) предопределены и создаются автоматически при создании нового проекта (слои отображаются в левом окне навигатора). Элементарные структурные составляющие (листья структурного дерева) называются компонентами проекта. Группы компонентов, которые могут быть созданы в структуре проекта, предназначены для структурирования проекта. В пра-

вом окне навигатора отображается содержимое слоя (группы), выделенной в левом окне.

1.2 Классификация компонентов TRACE MODE

По функциональному назначению компоненты проекта можно разделить на виды:

– **каналы** – компоненты, определяющие алгоритм работы проекта;

– **шаблоны** – компоненты, которые при работе в реальном времени могут вызываться каналами с передачей параметров (аналог подпрограмм). Передача параметров настраивается при разработке проекта в ИС посредством привязки аргументов шаблона к каналам или источникам/приемникам;

– **источники/приемники** – шаблоны каналов обмена с различными устройствами и приложениями. Под устройствами здесь понимаются контроллеры, а также внешние и внутренние модули/платы различного назначения, обмен с которыми поддерживается мониторами TRACE MODE (в том числе через драйверы). Системные переменные TRACE MODE и встроенные генераторы также создаются в ИС как источники/приемники;

– **наборы ресурсов** – наборы текстов, изображений и видеоклипов, которые могут быть использованы при разработке шаблонов графических экранов;

– **графические объекты** – компоненты, представляющие собой в общем случае несколько графических элементов (из имеющихся в редакторе представления данных), сгруппированных в один. Графические объекты могут быть использованы при разработке шаблонов графических экранов;

– **последовательные порты** – параметры COM-портов;

– **словари сообщений** – наборы сообщений, генерируемых при возникновении различных событий;

– **клеммы** – эти компоненты, описывающие электрические контакты (например, монтажных шкафов), являются элементами схемы электрических соединений АСУ.

1.2.1 Каналы

По функциональному назначению классы каналов можно сгруппировать следующим образом:

1. каналы для работы с данными (числовые каналы):
   - **HEX16** - для работы с 2-байтовыми целыми числами;
   - **HEX32** - для работы с 4-байтовыми целыми числами;
   - **FLOAT** - для работы с 4-байтовыми вещественными числами;
   - **DOUBLE FLOAT** - для работы с 8-байтовыми вещественными числами;
   - **TIME** - дня работы со значениями времени (дата и время);
2. каналы для мониторинга:
   - **Событие** - для мониторинга объекта с целью фиксирования возникновения/исчезновения на этом объекте некоторого события или ситуации (например, аварии);
3. каналы для задания прав пользователей:
   - **Пользователь** - для задания прав пользователя на разработку и/или запуск проекта;
4. каналы T-FACTORY:
   - **Единица оборудования** - для учета единицы оборудования, планирования и мониторинга ее техобслуживания;
   - **Персонал** - для учета работника, а также планирования и мониторинга его участия в техобслуживании оборудования;
   - **M-ресурс** - для учета складских ресурсов;
   - **D-ресурс** - для мониторинга техобслуживания оборудования и ряда других задач;
5. каналы многофункционального назначения:
   - **CALL** - свойство **вызов** канала этого класса конфигурируется для выполнения различных функций. В ИС можно создать следующие каналы этого класса с предустановленным свойством **вызов** (при создании такого канала в соответствующем слое шаблонов создается шаблон, вызываемый каналом): **Экран** - канал с вызовом шаблона экрана; **Программа** - канал с вызовом шаблона программы; **Документ** - канал с вызовом шаблона документа; **Связь с БД** - канал с вызовом связи с базой данных.

1.2.2 Шаблоны

Шаблон можно рассматривать как функцию, которую вызывает основная программа (монитор) с передачей определенных значений. Шаблоны вызываются каналами класса **CALL** (шаблоны программ

могут быть вызваны каналами других классов с настроенным свойством **вызов)** при их отработке монитором.

Значения в шаблон передаются через его аргументы. Эта передача настраивается в ИС с помощью привязки аргументов шаблона к каналам или источникам/приемникам в редакторе аргументов. Передача аргументов при вызове шаблона обязательна – т.е., шаблон должен иметь хотя бы один аргумент. В соответствующих слоях структуры проекта могут быть созданы следующие шаблоны (компоненты проекта): **шаблон; шаблон экрана, графической панели, мнемосхемы; шаблон документа (отчета); шаблон связи с базой данных** - компонент проекта, в котором хранятся SQL-запросы к определенной базе данных**.**

1.3 Классификация слоев

Предопределенные слои структуры проекта имеют следующее назначение:

- **Ресурсы** – для создания **пользовательских** наборов текстов, изображений и видеоклипов, а также графических объектов;
- **Шаблоны программ** – для создания шаблонов программ;
- **Шаблоны экранов** – для создания шаблонов графических экранов и графических панелей;
- **Шаблоны связей с БД** – для создания шаблонов связей с базами данных;
- **Шаблоны документов** – для создания шаблонов документов (отчетов);
- **База каналов** – этот слой является хранилищем всех каналов проекта. Выполнять операции с каналами можно в различных слоях, однако во всех случаях эти операции на самом деле реализуются в слое **База каналов**;
- **Система** – для конфигурирования **узлов** и их составляющих (узел создается как корневая группа этого слоя);
- **Источники/приемники** – для создания встроенных генераторов, шаблонов каналов обмена с различными устройствами и программными приложениями, а также для конфигурирования системных переменных TRACE MODE 6,
- **Технология** – для разработки проекта от технологии (т.е. с группировкой компонентов по признаку их принадлежности к технологическому объекту). При отладке проекта слой Технология

может играть роль узла – для него определена команда Сохранить узел для МРВ. Кроме того, для этого слоя определены команды взаимодействия с технологической базой данных;

- **Топология** – для разработки проекта от топологии (т.е. с группировкой компонентов по месту расположения);
- **КИПиА** – для описания электрических соединений АСУ;
- **Библиотеки компонентов** – для создания **библиотек объектов** – проектных решений отдельных задач. Этот слой содержит предопределенные группы «**Системные»** и «**Пользовательские»**. В группе «**Системные»** содержатся библиотеки, подключенные к ИС по умолчанию.

1.4 Классификация узлов

Узлы проекта создаются как корневые группы слоя «**Система**». Предопределенное название узла указывает на семейство мониторов, для которых данный узел предназначен. Узел может **содержать только те компоненты, которые поддерживаются мониторами соответствующего семейства.** В общем случае, узлы могут выполняться под управлением различных мониторов. Как правило, узел выполняется на отдельном аппаратном средстве. В случае запуска двух и более узлов на одном аппаратном средстве оно должно быть оборудовано соответствующим количеством сетевых карт. Классификация узлов:

- **RTM –** предназначен для запуска **на компьютере** под управлением исполнительных модулей семейства **RTM** (**MPB**) – мониторов с **поддержкой** отображения **графических экранов** оператора, **поддержкой** обмена по последовательному интерфейсу и сети с различным оборудованием и выполняющего пересчет каналов всех классов, кроме каналов T-FACTORY.
- **T-FACTORY –** предназначен для запуска на компьютере под управлением исполнительных модулей семейства **T-FACTORY** – мониторов для решения задач **АСУП**.
- **MicroRTM –** предназначен для запуска на компьютере или в **контроллере** под управлением исполнительных модулей семейства **Micro RTM.** Основное отличие этих мониторов от МРВ – **отсутствие** поддержки отображения **графических экранов**.

- **Logger –** предназначен для запуска на компьютере под управлением исполнительного модуля **Logger** (регистратор) – монитора, способного вести **архивы по каналам** всех узлов проекта.
- **EmbeddedRTM –** предназначен для запуска на компьютере или в **контроллере** под управлением исполнительных модулей семейства **Embedded RTM** – мониторов **с поддержкой графических панелей**, поддержкой обмена с оборудованием по различным протоколам и выполняющего пересчет каналов.
- **NanoRTM –** предназначен для запуска в **контроллере** под управлением исполнительного модуля **Nano RTM** – монитора, аналогичного **Micro RTM**, но предназначенного для работы с малым числом каналов.
- **Console –** предназначен для запуска на компьютере под управлением исполнительных модулей, которые, в отличие от MPB, **не выполняют пересчет каналов**. Консоли позволяют получать данные от других узлов проекта по сети, отображать их на графических экранах и управлять технологическим процессом из графики. Консоли не могут взаимодействовать с узлами T-FACTORY.
- **TFactory_Console –** предназначен для запуска на компьютере под управлением исполнительных модулей, аналогичных консолям, но, кроме того, способных взаимодействовать с узлами T-FACTORY.
- **TM_OPC_Server –** Этот узел зарезервирован.

1.5 Технология разработки проекта в ИС

Разработка проекта в ИС включает следующие этапы:

1. создание структуры проекта в навигаторе;
2. конфигурирование или разработка структурных составляющих – разработка шаблонов графических экранов оператора, разработка шаблонов программ, описание источников/приемников и т.д.;
3. конфигурирование информационных потоков;
4. выбор аппаратных средств АСУ (компьютеров, контроллеров и т.п.);
5. создание узлов в слое **Система** и их конфигурирование;
6. распределение каналов, созданных в различных слоях структуры, по узлам и конфигурирование интерфейсов взаимодействия компонентов в информационных потоках;

7. экспорт данных в узлы для последующего запуска под управлением мониторов TRACE MODE (по команде **Сохранить для MPB)**.

Перечисленные процедуры (за исключением последней) и входящие в их состав операции могут выполняться в произвольном порядке.

1.6 Интегрированная среда разработки

ИС объединяет в единой оболочке **навигатор** и набор редакторов для создания всех составляющих проекта. ИС имеет многооконный интерфейс.

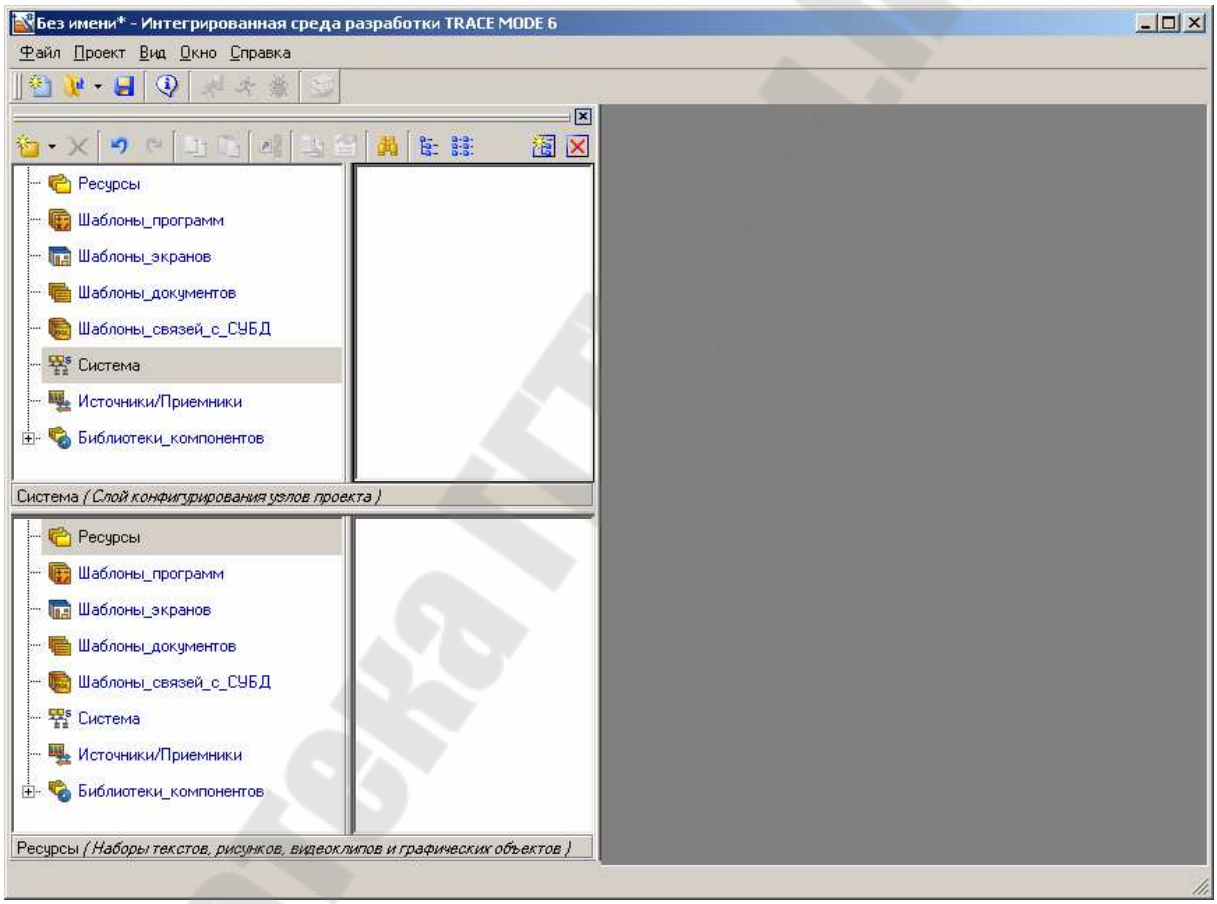


Рисунок 1.1 – Вид окна ИС разработки проекта

В ИС поддерживаются стандартные операции изменения размеров и перемещения окон**.** В навигаторе структура проекта представлена в виде дерева (рисунок 1.2).

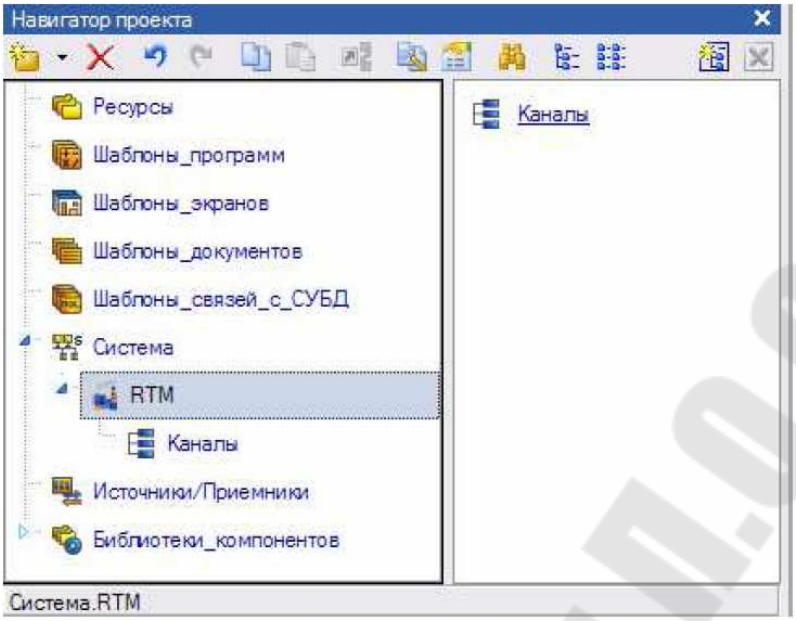


Рисунок 1.2 – Вид окна навигатора структуры проекта

Корневые группы этого дерева **(слои)** предопределены и создаются автоматически при создании нового проекта (слои отображаются в левом окне навигатора). Элементарные структурные составляющие (листья структурного дерева) называются **компонентами** проекта. Например, компонентами проекта являются: канал; канал, вызывающий шаблон; шаблон; источник данных и т.д. **Группы компонентов,** которые могут быть созданы в структуре проекта, предназначены для структурирования проекта. Структурирование в значительной степени облегчает редактирование проекта.

1.7 Редактирование структуры проекта

Навигатор имеет следующие средства для редактирования структуры проекта: меню **Проект;** панели инструментов; контекстное меню. Для конфигурации/разработки объектов структуры в навигаторе предусмотрены команды **Свойства** и **Редактировать**, с помощью которых для каждого объекта структурного дерева могут быть открыты соответствующие **окно свойств** и **редактор.**

1.8 Создание и редактирование свойств объектов структуры

Для создания объектов структуры (компонентов и групп компонентов) используются типовые команды меню **Проект,** контекстного меню и панели инструментов навигатора. Меню **Проект,** контекстное

меню и панель инструментов навигатора содержат команды создания только тех объектов, которые может содержать выделенный слой/группа.

По команде **Свойства** (– нанели инстументов, или в контекстном меню объекта) в нижней части ИС открывается **окно свойств** выделенного объекта структуры проекта.

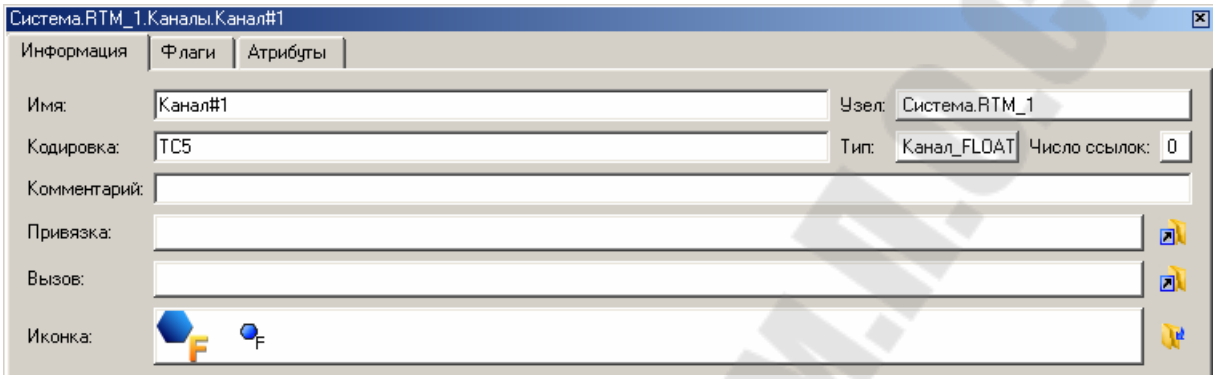


Рисунок 1.3 – Вид окна свойств объекта канал, структуры проекта

Эта вкладка присутствует в окне свойств любого объекта структуры. На ней отображаются/редактируются следующие свойства:

- **Имя** – имя объекта (задается разработчиком);
- **Кодировка** – кодировка типа объекта, задаваемая по умолчанию в TRACE MODE. Кодировку объекта можно изменить с помощью этого поля;
- **Комментарий** – комментарий разработчика;
- **Иконка** – иконка, заданная для объекта по умолчанию. Для смены иконки надо нажать кнопку и выбрать иконку в появившемся меню;
- **Тип** – предопределенный в TRACE MODE тип объекта;
- **Узел** – узел, в который входит объект**;**
- **Счетчик ссылок –** число компонентов, связанных с данным компонентом или вызывающих данный компонент;
- **Привязка** – конфигурация свойства **связь.** Чтобы задать связь, нужно нажать кнопку справа от данного поля и выбрать в диалоге, компонент (канал или источник/приемник) и его атрибут;
- **Вызов** – конфигурация свойства **вызов.** Чтобы задать это свойство, нужно нажать кнопку справа от данного поля и выбрать в диалоге, показанном на рисунке ниже, вызываемый шаблон.

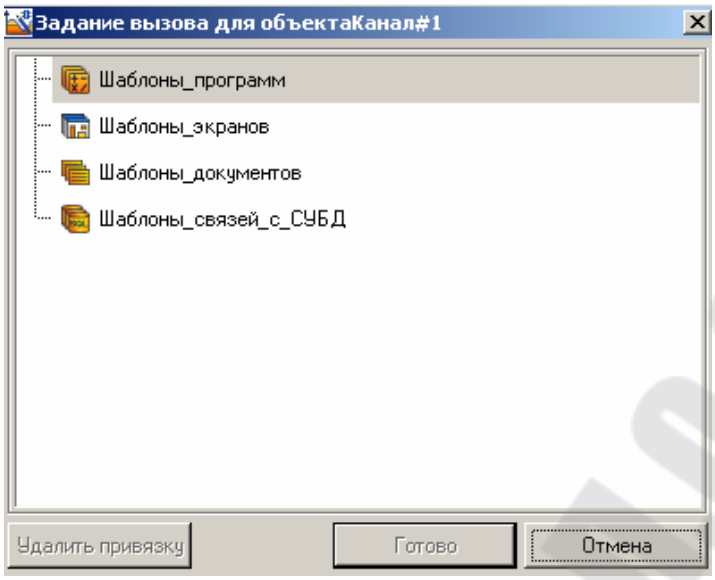


Рисунок 1.4 – Вид окна конфигурации свойства **вызов**

1.8 Аргументы. Табличный редактор аргументов

Аргументы компонента проекта и их привязки к атрибутам/аргументам других компонентов настраиваются в табличном **редакторе аргументов,** в котором параметры каждого аргумента задаются в отдельной строке. Аргументы могут быть созданы только для шаблона и канала класса **CALL** с ненастроенным свойством **вызов,** при этом редакторы аргументов этих компонентов имеют отличающиеся функции.

Существует также третий вид редактора аргументов - для компонента с настроенным свойством **вызов**. В этом случаи при задании свойства **вызов**, таблица аргументов шаблона копируется в вызывающий компонент (в том числе и привязки аргументов, если они были заданы в редакторе аргументов шаблона), при этом в окне свойств вызывающего компонента появляется вкладка **Аргументы.** С помощью этой вкладки привязки аргументов (теперь уже аргументов компонента, вызывающего шаблон) могут быть изменены (компоненты могут вызывать один и тот же шаблон с передачей различных параметров). Такая перепривязка не отражается на таблице аргументов шаблона. Добавление/удаление аргументов в шаблоне воспроизводится во всех компонентах, вызывающих этот шаблон. Кроме того, задание привязки аргумента в шаблоне воспроизводится во всех компонентах, вызывающих шаблон, в которых данный аргумент не был привязан. Основные параметры аргументов - **имя, тип** и **тип данных**

- не могут быть изменены в редакторе аргументов компонента с настроенным свойством **вызов.**

Если компонент имеет аргументы, редактор аргументов доступен в окне свойств и редакторе этого компонента.

1.8 Поля редактора аргументов

Табличные редакторы аргументов имеют следующий набор полей:

• **Имя** - имя аргумента, задается по умолчанию и может быть изменено. Для перехода к редактированию имени аргумента нужно дважды нажать JIK в данном поле. Имена аргументов одного и того же компонента должны быть уникальными;

• **Тип** - тип аргумента; выбирается в списке, который открывается при двойном нажатии JIK в данном поле (**INPUT** - передача по значению (для приема); **OUTPUT** и **INPUT/OUTPUT** - передача по ссылке (для передачи);

Тип данных - тип данных. Тип данных выбирается в списке, который открывается при двойном нажатии ЛК в данном поле. Этот параметр должен учитываться при привязке аргументов;

[ ] - объявление массива, используется только в функциях шаблонов;

Значение по умолчанию - значение, указанное в этом поле, используется монитором в случае отсутствия привязки у аргумента;

Привязка - привязка аргумента. При двойном нажатии ЛК в данном поле на экране появляется диалог выбора компонента/аргумента для привязки**;**

Принадлежность - зарезервировано;

Флаги - флаги, установленные для аргумента. Могут быть установлены следующие флаги: **HW, SL, NP, PO.** От флагов зависит результат автопостроения/привязки каналов**;**

Группа - номер группы, к которой принадлежит аргумент. Для группирования нужно выделить несколько аргументов и нажать кнопку **Группировать выделенные аргументы** - по этой команде выделенной группе аргументов автоматически присваивается номер. От параметра **Группа** зависит результат автопостроения каналов из редактора аргументов**;**

Единицы измерения - единицы измерения (или унифицированный сигнал), выбираются в списке, который открывается при двойном нажатии ЛК в данном поле;

Комментарий - комментарий разработчика к аргументу.

1.9 Разновидности редактора аргументов

1.9.1 Редактор аргументов шаблона

Редактор аргументов шаблона (рисунок 1.5) снабжен типовыми инструментами создания, удаления, работы с буфером обмена и поиска.

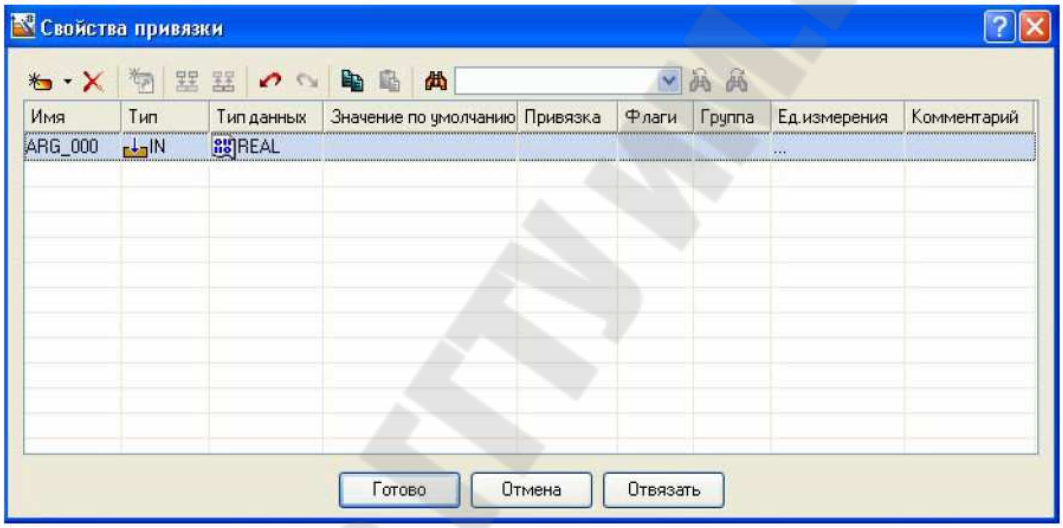


Рисунок 1.5 – Вид окна редактора аргументов шаблона

При задании имен аргументов шаблона нужно учитывать следующее:

- шаблоны экранов и связей с БД используют индексы аргументов - это означает, что при изменении имени аргумента, например, шаблона связи с БД в ИС **автоматически** изменятся: имя соответствующего аргумента компонента, вызывающего этот шаблон, и SQL-запрос, в котором использован аргумент;
- шаблоны отчетов и программ используют имена аргументов – это означает, что после изменения имени аргумента необходимо соответствующим образом отредактировать шаблон программы или документа.

1.9.2 Редактор аргументов канала CALL с ненастроенным свойством ВЫЗОВ

В отличие от редактора аргументов шаблона, в этом редакторе присутствуют инструмент –  **Создать каналы и привязать к аргументам.** По этой команде в той же группе каналов, к которой принадлежит и канал CALL, автоматически создаются каналы и привязываются к аргументам. Алгоритм этой процедуры зависит от параметров аргументов.

1.9.3 Редактор аргументов компонента с настроенным свойством ВЫЗОВ

В этом редакторе отсутствует инструмент создания аргументов и присутствует переключатель –  **Синхронизировать изменения привязок со всеми вызовами шаблона**. Если эта опция включена (иконка имеет «утопленный» вид), то изменение привязок аргументов данного компонента равнозначно изменению привязок всех компонентов, вызывающих тот же шаблон.

2. Ход работы

Для изучения средств разработки и основных элементов проекта, рассмотрим процесс его создания на примере.

2.1 Создание узла АРМ

Разработка проекта начинается с запуска Интегрированной среды разработки (ИСР) – TRACE MODE IDE 6 иконка  .

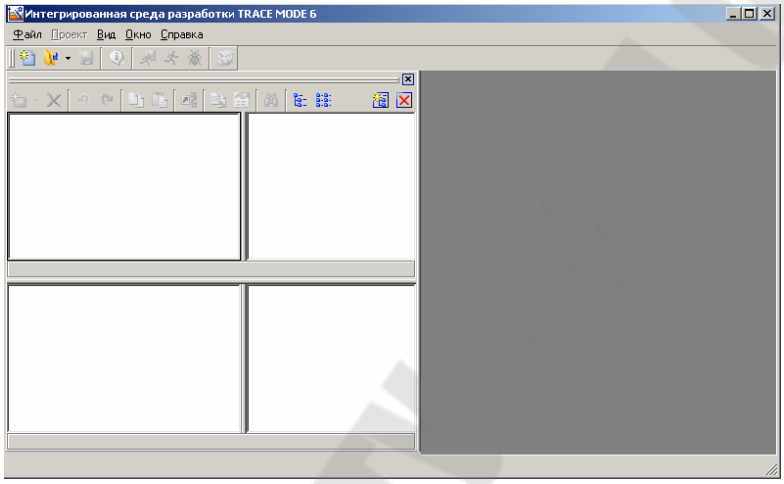


Рисунок 2.1 – Интегрированная среда разработки

После запуска ИСР в меню «**Файл»** выбираем команду «**Настройки ИС».** В появившемся окне настраиваем пункты – «**Уровень сложности»** (рисунок 2.2) и «**Отладка» (**рисунке 2.3).

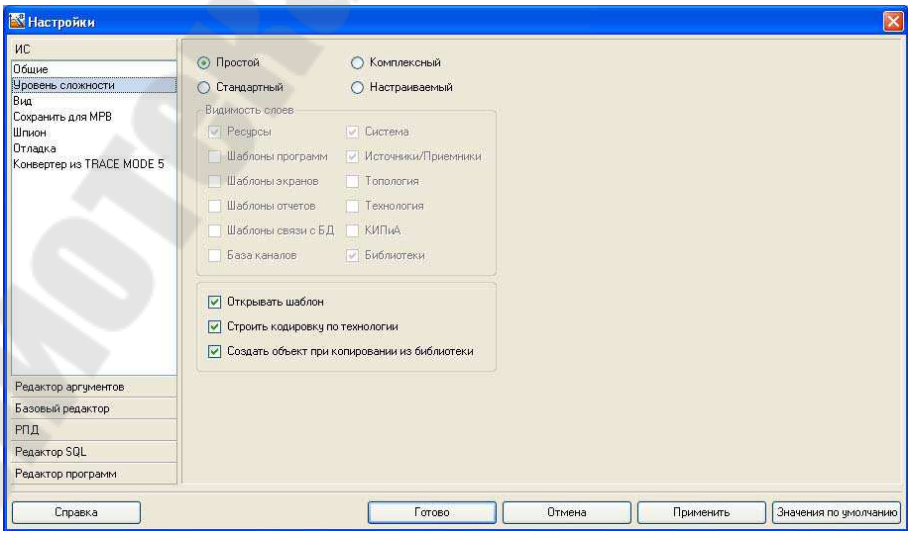


Рисунок 2.2 – Настройки уровня сложности проекта

Уровни сложности:

- **простой** - отображаются слои Ресурсы, Система, Источники/Приемники и Библиотеки компонентов;
- **стандартный** (значение по умолчанию) - отображаются те же слои, что и для простого уровня, и все слои шаблонов (экранов, программ, связей с БД и документов);
- **комплексный** - отображаются все слои, кроме слоя База каналов;
- **настраиваемый** - при выборе этого уровня в диалоге доступны переключатели отображения всех слоев, включая слой База каналов.

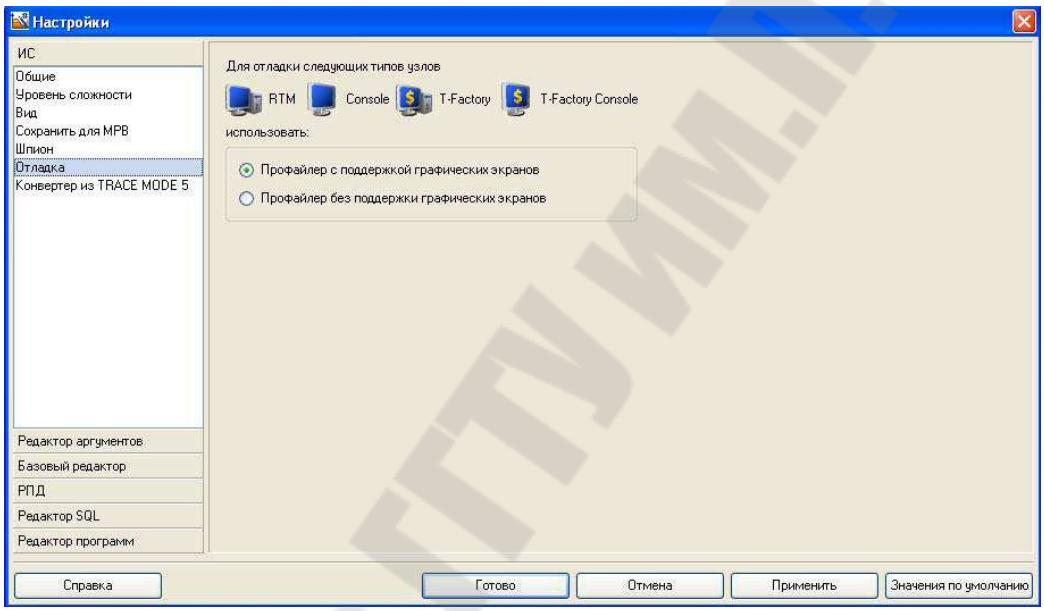


Рисунок 2.3 – Настройки отладки узлов проекта

С помощью иконки инструментальной панели создадим новый проект. В навигаторе проекта (рисунок 2.4) выделить строку «**СИСТЕМА**» и в контекстном меню выбрать «**СОЗДАТЬ УЗЕЛ**» – **RTM**.

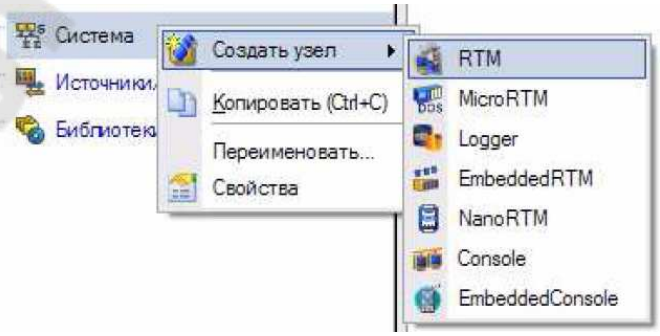


Рисунок 2.4 – Создание нового узла

Для создания шаблона экрана, вызвать контекстное меню к группе «**RTM**» и выбрать «**СОЗДАТЬ КОМПОНЕНТ**» – «**ЭКРАН**» (рисунок 2.5).

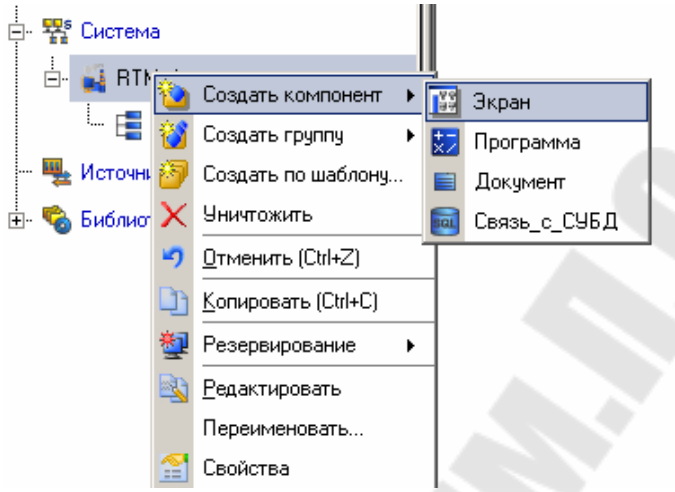


Рисунок 2.5 – Создание шаблона экрана

2.2 Создание графического экрана

Двойным щелчком ЛК на компоненте **Экран#1** откроем окно графического редактора.

2.2.1 Создание статического текста

Разместим в левом верхнем углу экрана статический текст – надпись «**Значение параметра**». Для этого выполним следующие действия:

- на панели инструментов графического редактора ЛК мыши выделим иконку ГЭ «**Текст**» ABC;
- в поле графического редактора установим прямоугольник ГЭ, для чего зафиксируем ЛК *точку привязки* - левый верхний угол;
- развернем прямоугольник движением курсора до необходимого размера;
- зафиксируем ЛК выбранный ГЭ.

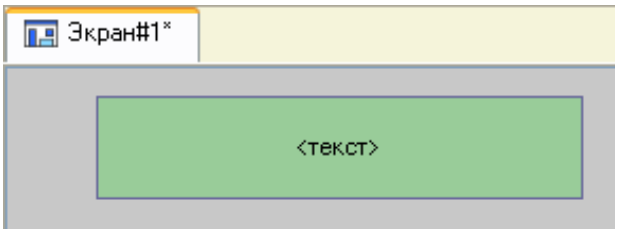


Рисунок 2.6 – Графический элемент «Текст»

Для перехода в режим редактирования атрибутов размещенного ГЭ выделим ЛК иконку на панели инструментов и двойным щелчком ЛК по ГЭ откроем окно его свойств (рисунок 2.7). В правом поле строки «**Текст**» наберем «**Значение параметра**» и нажмем на клавиатуре клавишу **Enter**.

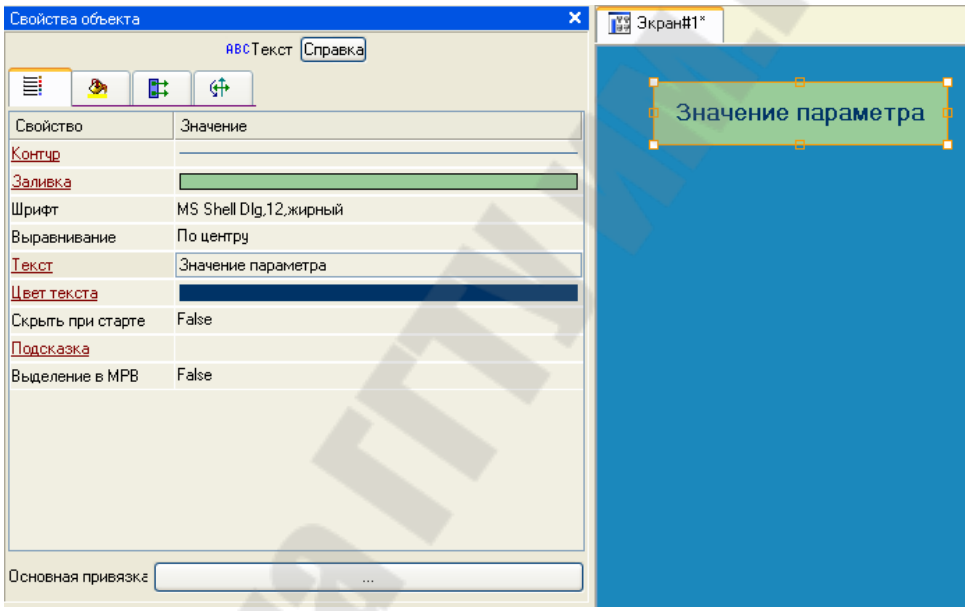


Рисунок 2.7 – Графический элемент «Текст»

Закроем окно свойств щелчком ЛК по иконке , ГЭ примет вид – рисунок 2.8.

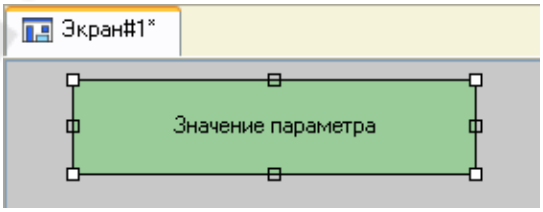


Рисунок 2.8 – Графический элемент «Текст»

мЕсли введенный текст не уместился в прямоугольнике ГЭ, выделите его и растяните до нужного размера с помощью мыши.

Для автоматического вывода окна свойств ГЭ по завершению его размещения можно в настройках интегрированной среды разработки

в разделе «**РПД/Основные свойства**» активировать пункт «**Открывать свойства автоматически**» (рисунок 2.9).

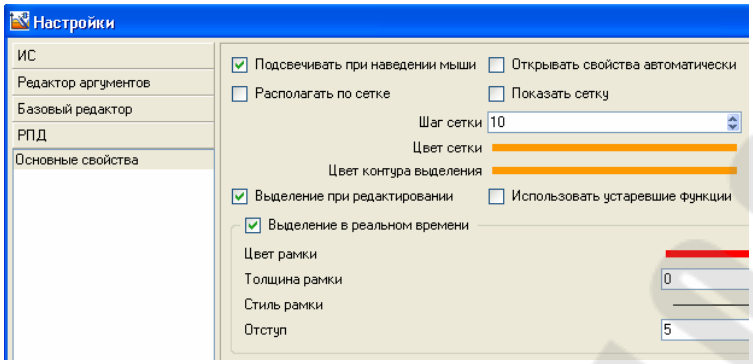


Рисунок 2.9 – Окно настройки автоматического вывода свойств ГЭ

2.2.2 Создание динамического текста, создание аргумента экрана в процессе настройки динамического текста

Создадим элемент для динамического отображения численного значения какого-либо источника сигнала – внешнего или внутреннего - путем задания динамизации атрибута «**Текс**» ГЭ. Для этого необходимо проделать следующие действия:

– создадим и разместим новый ГЭ 

справа от ГЭ с надписью «**Значение параметра**»;
– откроем свойства вновь размещенного ГЭ;
– двойным щелком ЛК на строке «**Текст**» вызовем меню «**Вид индикации**» (рисунок 2.10);

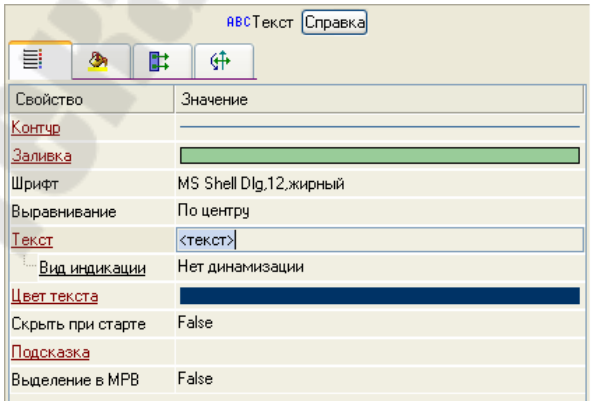


Рисунок 2.10 – Вид окна настройки

– в правом поле строки щелчком ЛК вызовем список доступных типов динамизации атрибута и выберем «**Значение**» (рисунок 2.11);

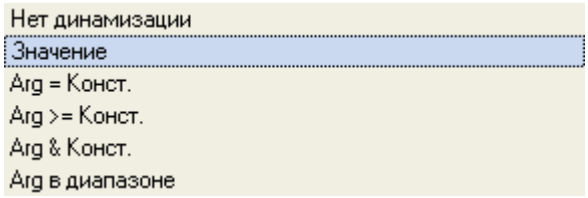


Рисунок 2.11 – Вид окна

– в открывшемся меню настройки параметров динамизации щелкнем ЛК в правом поле строки **Привязка**;

Рисунок 2.12 – Вид окна

– в открывшемся окне «**Свойства привязки**», нажмем ЛК по иконке  на панели инструментов и тем самым создадим **аргумент шаблона экрана**;

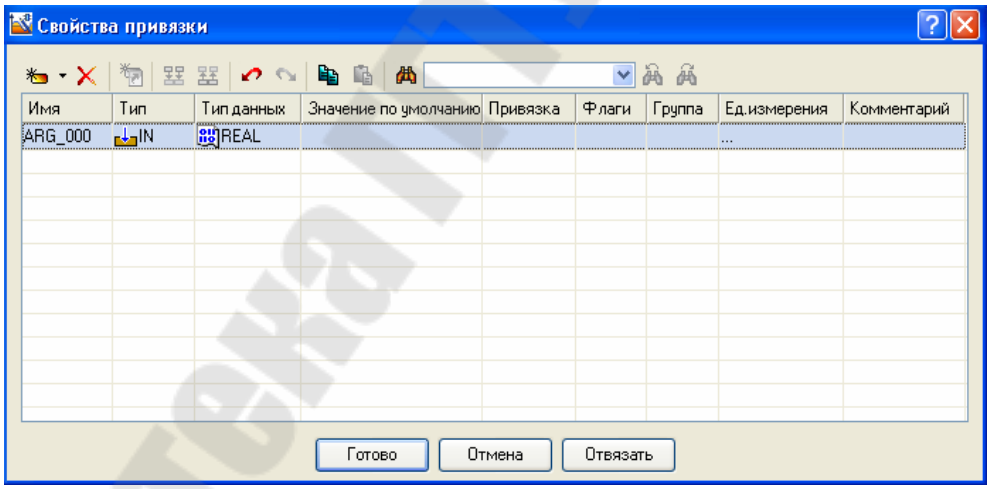


Рисунок 2.13 – Вид окна

– двойным щелчком ЛК выделим имя аргумента и изменим его, введя с клавиатуры «**Параметр**» (здесь и в дальнейшем ввод данных с клавиатуры будем завершать нажатием клавиши **Enter**);
– подтвердим связь атрибута **Текст** ГЭ с данным аргументом щелчком ЛК по экранной кнопке **Готово**;
– закроем окно свойств ГЭ.

Созданный графический экран будет иметь следующий вид:

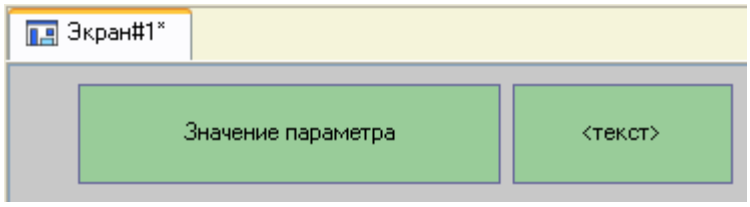


Рисунок 2.14 – Вид экрана

2.2.3 Создание стрелочного прибора, с привязкой показаний к аргументу

Применим для отображения значения новый тип ГЭ – «**Стрелочный прибор**». Для этого необходимо выполнить последовательность действий:

- выделим двойным щелчком ЛК на инструментальной панели графического редактора иконку и выберем из появившегося меню иконку стрелочного прибора;
- установим ГЭ, выбрав его размер таким, чтобы все элементы графики и текста на нем были разборчивы и симметричны (рисунок 2.15);
- перейдем в режим редактирования и откроем окно свойств ГЭ;
- щелчком ЛК на экранной кнопке «**Основная привязка**» откроем окно табличного редактора аргументов шаблона экрана;
- ЛК выберем уже имеющийся аргумент «**Параметр**»;
- подтвердим выбор щелчком ЛК на кнопке «**Готово**»;
- двойным щелчком ЛК откроем атрибут «**Заголовок**» и в строке «**Текст**» введем надпись «**Параметр**»;
- закроем окно свойств ГЭ.

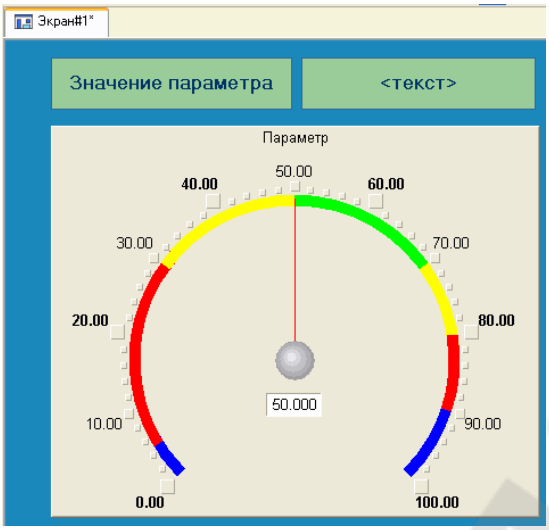


Рисунок 2.15 – Графический элемент «Стрелочный прибор»

Для проверки правильности привязок ГЭ к аргументам экрана воспользуемся режимом эмуляции, переход в который осуществляется с помощью иконки  , панели инструментов. По нажатию, на экран графического редактора выводится окно задания значения аргумента в соответствующем поле (рисунок 2.16).

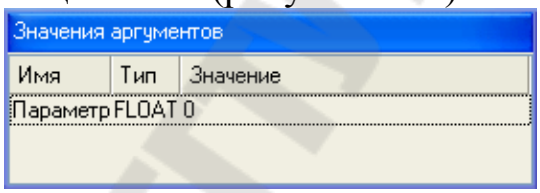


Рисунок 2.16 – Окно задания значения аргумента

Зададим, для примера, значение – **25**. Если оба ГЭ отображают введенное значение, то привязка выполнена верно. Выход из режима эмуляции – повторное нажатие ЛК по иконке  .

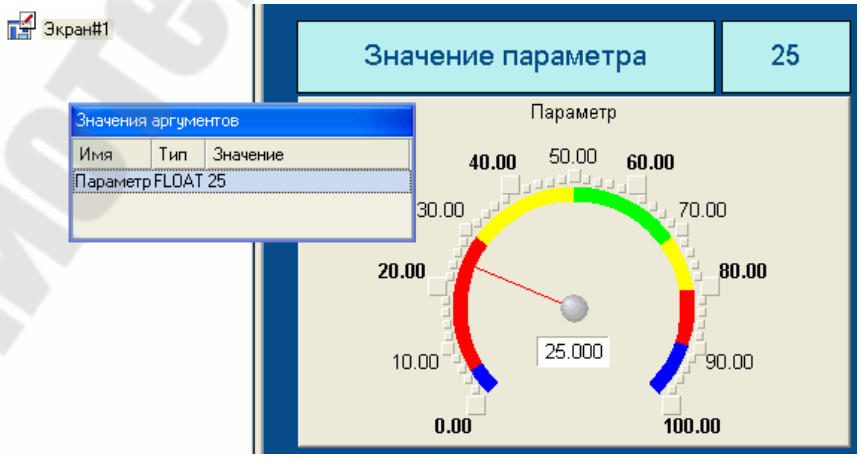


Рисунок 2.17 – Режим эмуляции

2.3. Автопостроение канала

Для создания **канала** в узле проекта по **аргументу шаблона экрана** воспользуемся процедурой *автопостроения*. Для этого:

– в слое «**Система**» выберем ЛК узел **RTM_1**;
– в поле компонентов узла ЛК выберем **Экран#1**;
– щелчком правой кнопки мыши (ПК) вызовем контекстное меню, в котором откроем свойства компонента **Экран#1** и выберем вкладку «**Аргументы»;**
– выделим ЛК аргумент «**Параметр**» и с помощью иконки  создадим канал класса **Float** типа **Input** с именем «**Параметр**» (рисунок 2.18).

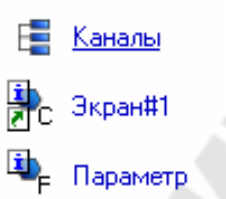


Рисунок 2.18 – Вид окна компонентов проекта

2.3.1 Задание границ и уставок

Двойным щелчком ЛК по каналу «**Параметр**» откроем окно редактирования его атрибутов и заполним раздел «**Границы**» как показано на рисунке 2.19.

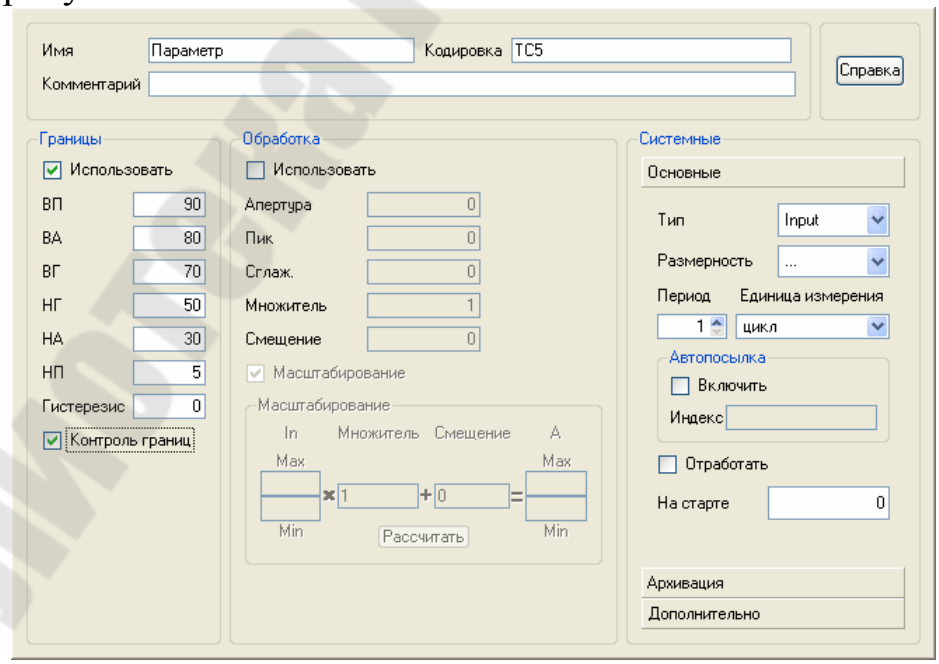


Рисунок 2.19 – Окно редактирования атрибутов канала

2.4. Создание генератора синуса и привязка его к каналу

Добавим в состав проекта источник сигнала – внутренний генератор синусоиды, свяжем его с созданным каналом и опробуем в работе выполненные средства отображения. Для этого проделаем следующие действия:

– откроем слой «**Источники/Приемники**» и через ПК создадим в нем группу компонентов «**Генераторы**» (рисунок 2.20);

–

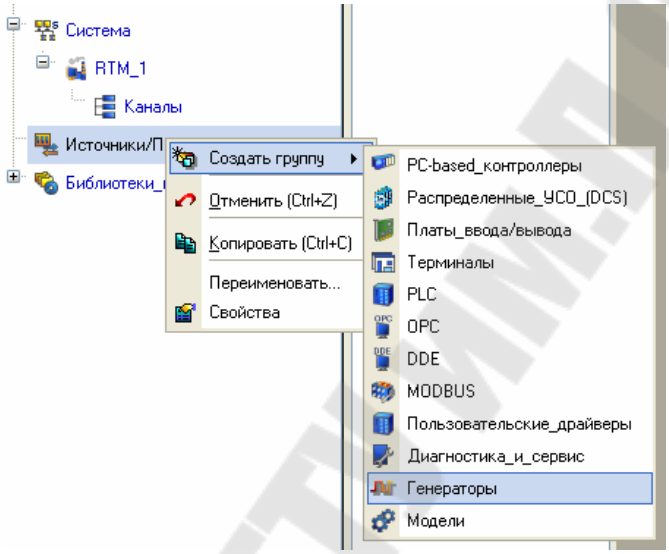


Рисунок 2.20 – Создание в группу компонентов «**Генераторы**»

– двойным щелчком ЛК откроем группу «**Генераторы_1**» и создадим в ней компонент «**Синусоида**»;

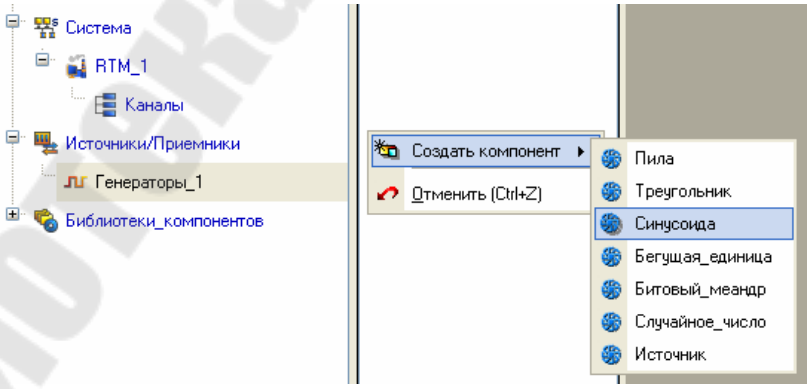


Рисунок 2.21 – Создание компонента «**Синусоида**»

– перетянем созданный источник на узел **RTM_1** в слой «**Система**», а затем, в открывшемся окне компонентов, на канал «**Параметр**» и отпустим ЛК (рисунок 2.22).

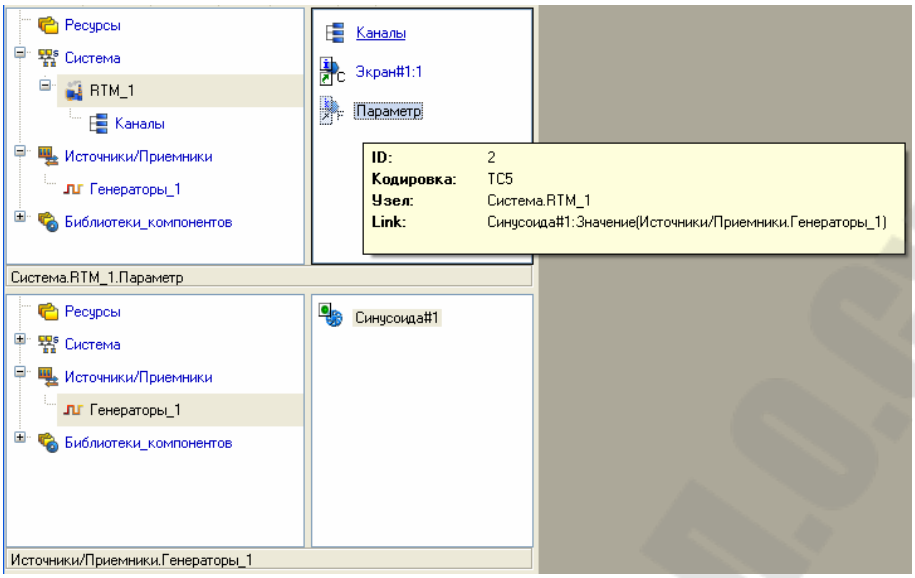


Рисунок 2.22 – Вид окна проекта после привязки генератора к каналу

2.5. Запуск проекта

Сохраним проект с помощью иконки , задав в открывшемся окне имя **Laba_1.prj**. На инструментальной панели выберем иконку и подготовим проект для запуска в реальном времени. ЛК выделим в слое «**Система**» узел **RTM_1**, а после, нажав ЛК иконку на инструментальной панели, запустим профайлер. Запуск/останов профайлера осуществляется с помощью иконки на его инструментальной панели или клавишной комбинации **Ctrl+R**.

В открывшемся окне ГЭ справа от надписи «**Значение параметра**» должно показываться изменение синусоидального сигнала. То же значение должен отображать и стрелочный прибор (рисунок 2.23).

Рисунок 2.23 – Вид окна профайлера

Для остановки работы профайлера кнопка .

2.6. Добавление функции управления

Введем в состав графического экрана ГЭ, позволяющий реализовать ввод числовых значений с клавиатуры. Создадим новый аргумент шаблона экрана для их приема. Для этого:

– вызовем графический экран на редактирование;
– на инструментальной панели графического редактора выберем ЛК иконку ГЭ «**Кнопка**» - ;
– с помощью мыши разместим ГЭ в поле экрана под стрелочным индикатором;

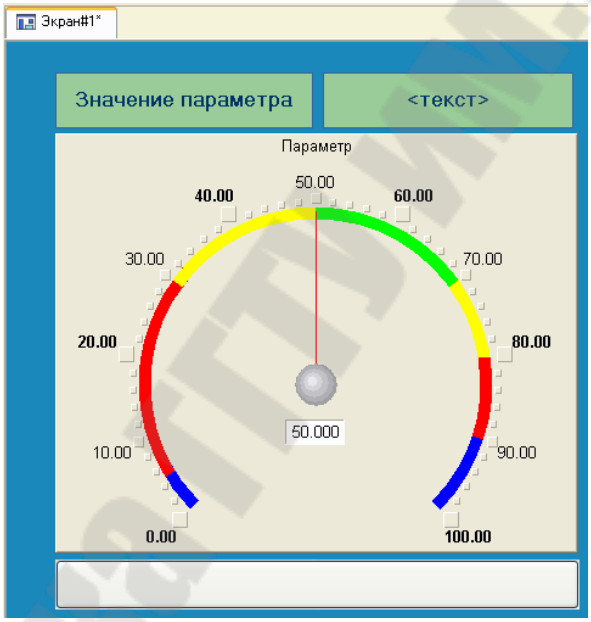


Рисунок 2.24 – Вид графического экрана после добавления элемента кнопка

– перейдем в режим редактирования , выделим ГЭ и вызовем окно его свойств;

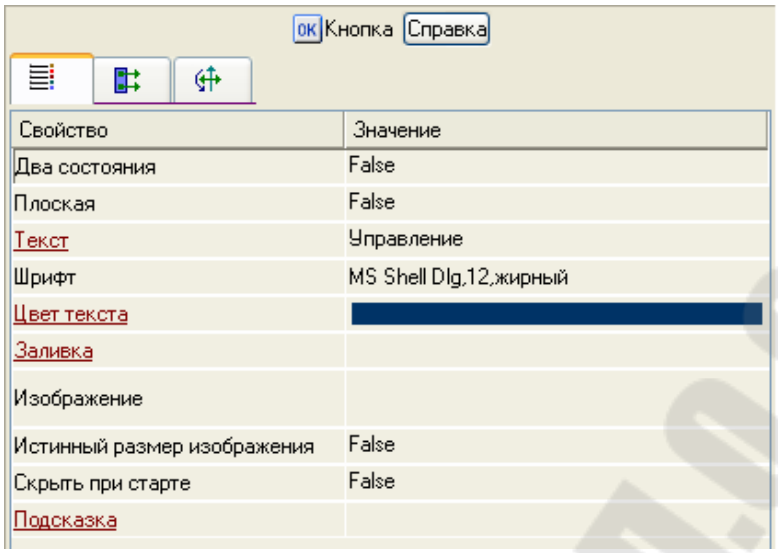


Рисунок 2.25 – Окно свойств элемента кнопка

– в поле «**Текст**» введем надпись «**Управление**»;
– откроем бланк «**События**» и раскроем меню «**По нажатию**» (**mousePressed**). Выберем из списка команду «**Добавить Send Value**»;

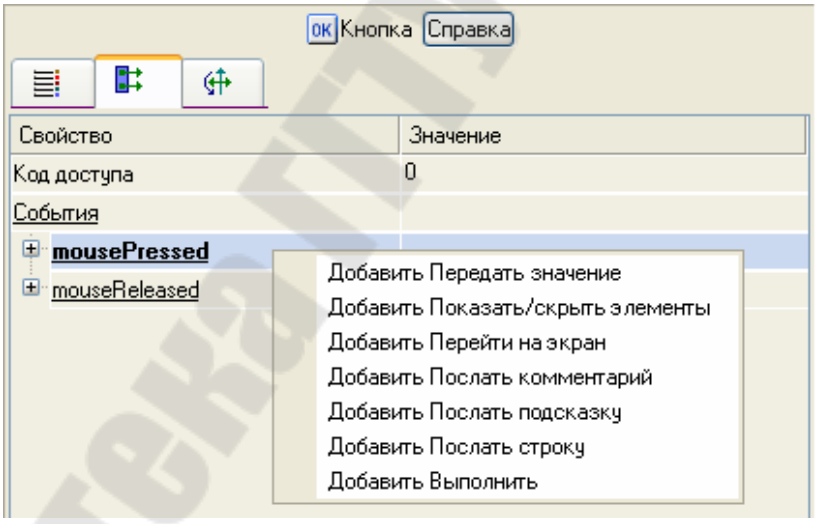


Рисунок 2.26 – Окно настройки событий элемента кнопка

– в раскрывшемся меню настроек выбранной команды в поле «**Тип передачи (Send Type)**» выберем из списка «**Ввести и передать (Enter & Send)**»;

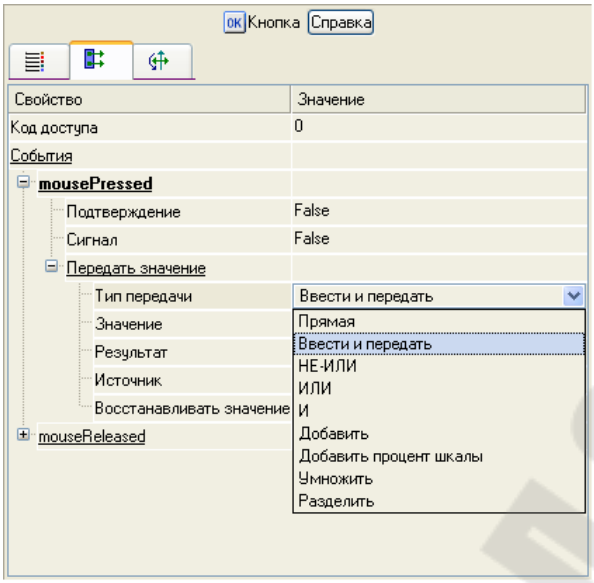


Рисунок 2.27 – Окно настройки событий элемента кнопка

– щелчком ЛК в поле «**Результат (Destination)**» вызовем табличный редактор аргументов;
– создадим еще один аргумент и зададим ему имя «**Управление**»;
– изменим тип аргумента на «**IN/OUT**», кнопкой «**Готово**» подтвердим привязку атрибута ГЭ к этому аргументу;

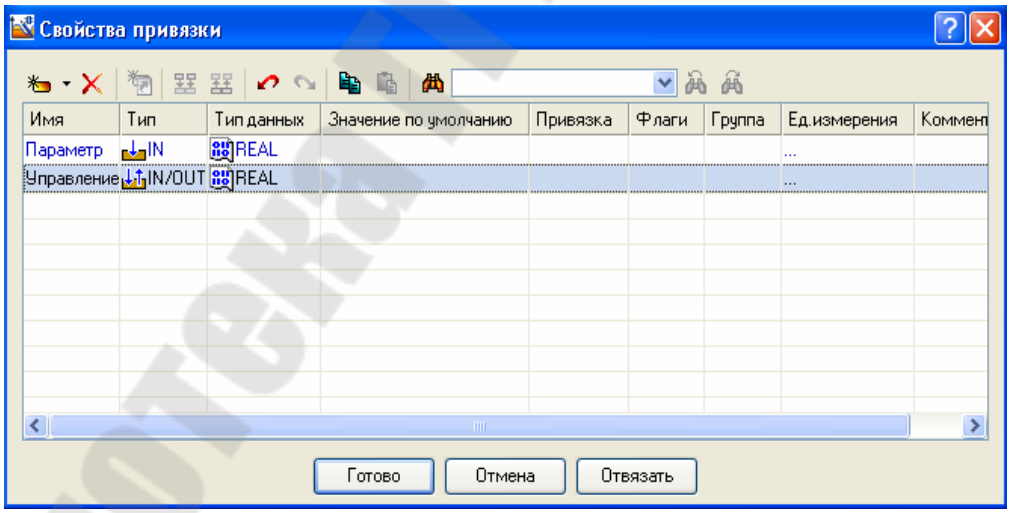


Рисунок 2.28 – Окно создания аргумента

– закроем окно свойств ГЭ с помощью щелчка ЛК по иконке .

Далее выполним размещение ГЭ «**Текст**» для отображения вводимого с клавиатуры значения. Воспользуемся уже имеющимся на графическом экране ГЭ путем его копирования/вставки и перепривязки. Для этого:

– выделим ЛК ГЭ «**Текст**», служащий для отображения аргумента «**Параметр**»;

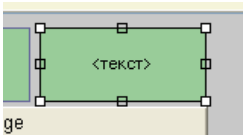


Рисунок 2.29 – Выделенный графический элемент

– с помощью иконки  на панели инструментов или комбинацией клавиш Ctrl+C скопируем выделенный ГЭ Текст в буфер обмена;
– далее с помощью иконки  или комбинацией клавиш Ctrl+V извлечем копию ГЭ из буфера обмена и поместим ее на графический экран;
– переместим, удерживая нажатой ЛК, копию ГЭ «**Текст**» справа от размещенного на экране ГЭ Кнопка;
– двойным щелчком ЛК на перемещенном ГЭ «**Текст**» откроем окно его свойств;
– двойным щелчком ЛК на строке «**Текст**» перейдем к настройке динамизации данного атрибута ГЭ. В правом поле строки «**Привязка**» щелчком ЛК откроем табличный редактор аргументов шаблона экрана;
– выделим ЛК в списке аргумент «**Управление**» и щелчком ЛК по экранной кнопке «**Готово**» подтвердим привязку атрибута ГЭ «**Текст**» к данному аргументу шаблона экрана;
– закроем окно свойств ГЭ «**Текст**».

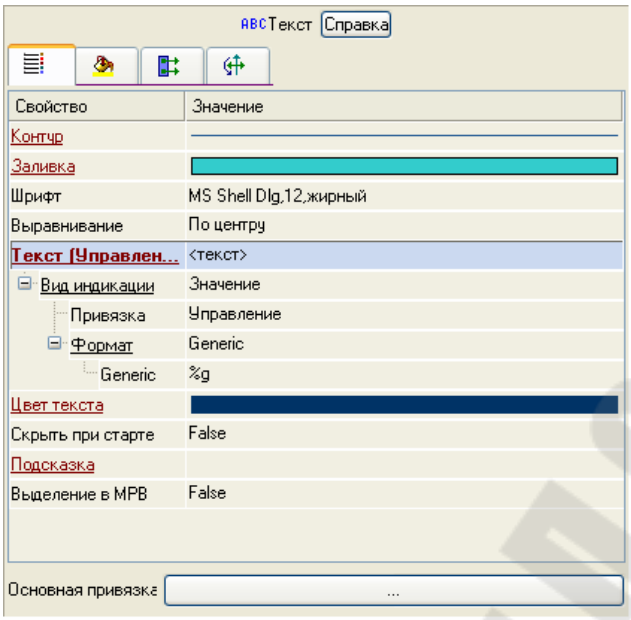


Рисунок 2.30 – Окно настройки привязки атрибута ГЭ «**Текст**»

2.7. Привязка аргумента экрана к каналу

Создадим по аргументу «**Управление**» шаблона экрана, новый канал и отредактируем привязку атрибута канала к аргументу шаблона экрана. Для этого:

– в слое «**Система**» откроем узел **RTM_1**;

– по щелчку ПК вызовем через контекстное меню свойства компонента **Экран#1**;

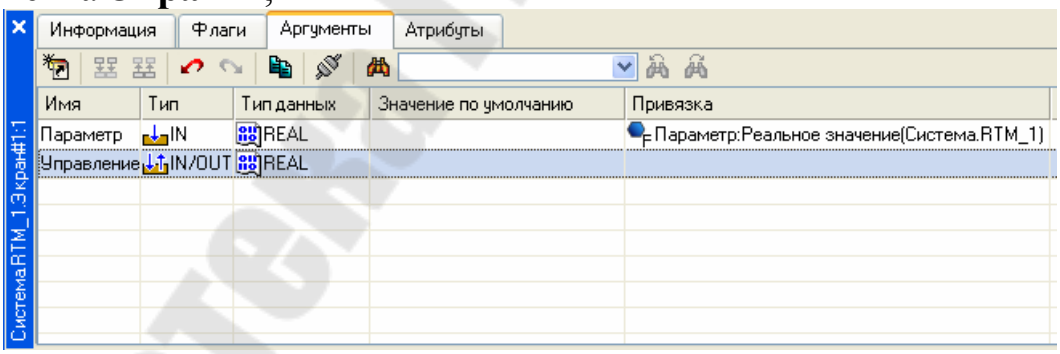


Рисунок 2.31 – Окно свойств компонента **Экран#1**

– выберем вкладку «**Аргументы**», ЛК выделим аргумент «**Управление**» и с помощью иконки  выполним автопостроение канала. В результате, в узле **RTM_1** ,будет создан канал с именем «**Управление**»;

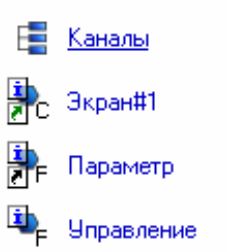


Рисунок 2.32 – Вид окна компонентов проекта

– двойным щелчком ЛК в поле «**Привязка**» аргумента «**Управление**» вызовем окно настройки связи, выберем в нем атрибут «**Входное значение**» канала «**Управление**» и кнопкой «**Привязка**» подтвердим связь аргумента экрана «**Управление**» с атрибутом «**Входное значение**» канала «**Управление**»;
– закроем окно свойств компонента **Экран#2.**

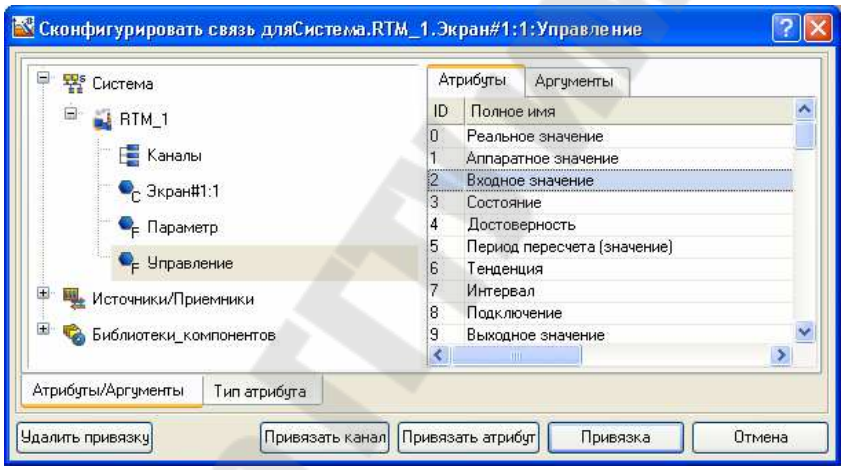


Рисунок 2.33 – Вид окна настройки аргумента с атрибутом канала

2.8. Размещение ГЭ Тренд

Дополним созданный экран новым ГЭ для совместного просмотра изменений значений каналов узла во времени и отслеживании предыстории – трендом.

В правой части графического экрана разместим ГЭ Тренд для вывода значений «**Параметр**» и «**Управление**» в виде графиков. Основные свойства ГЭ оставим заданными по умолчанию. Перейдем во вкладку и, выделив ЛК строку «**Кривые**», с помощью ПК создадим две новых кривых. Настроим для них привязки к существующим аргументам, толщину и цвет линий:

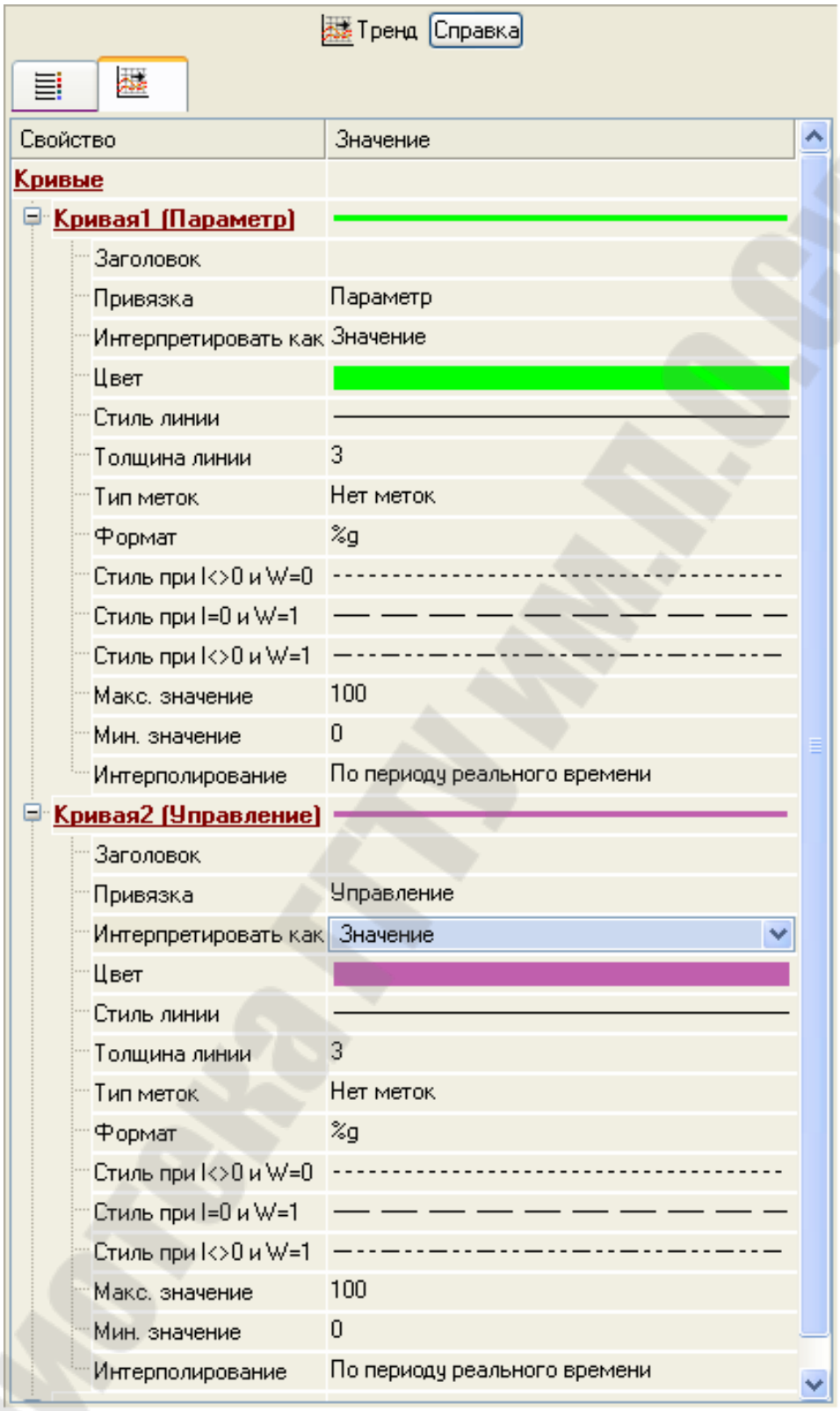


Рисунок 2.34 – Окно настройки ГЭ – тренд

После настройки, ГЭ тренд, примет вид – рисунок 2.35.

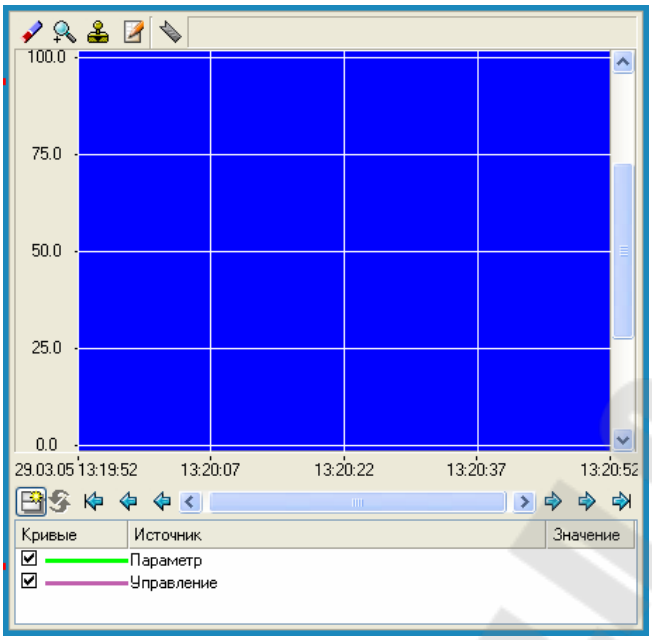


Рисунок 2.35 – Вид ГЭ – тренд

2.9 Запуск проекта

Сохраним проект с помощью иконки . На инструментальной панели выберем ЛК и подготовим тем самым проект для запуска в реальном времени. С помощью иконки на инструментальной панели запустим проект на исполнение.

С помощью кнопки «**Управление**» зададим величину «**управляющего воздействия**» (рисунок 2.36).

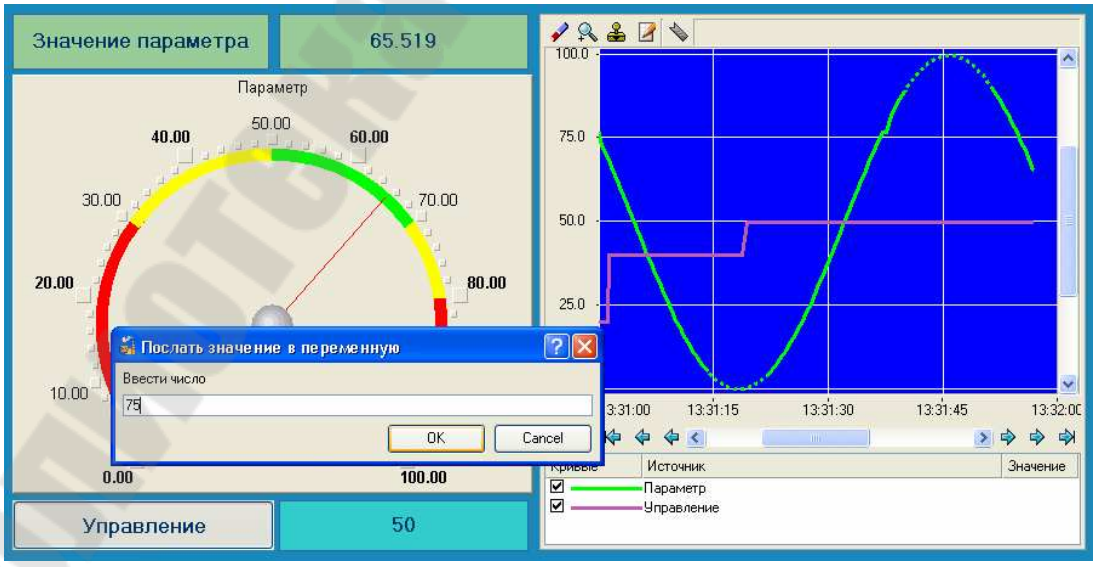


Рисунок 2.36 – Вид окна работы профайлера

2.10 Простейшая обработка данных

С помощью нового компонента проекта – **шаблона программы** свяжем два имеющихся канала операцией сложения. Будем суммировать реальные значения каналов «**Параметр**» и «**Управление**», а результат помещать во вновь созданный аргумент экрана «**Сумма**».

Скопируем два первых ГЭ – «**Значение параметра**», «**Текст**» и разместим их ниже ГЭ «**Кнопка**». Изменим статический текст первого ГЭ на «**Сумма**» (рисунок 2.37).

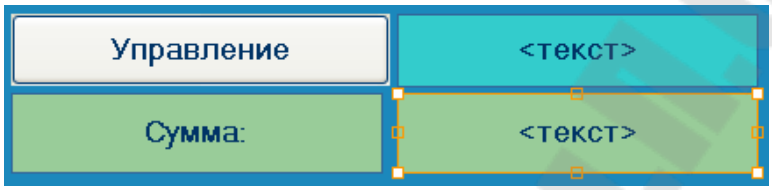


Рисунок 2.37 – Вид окна с новыми элементами

Динамику нового ГЭ привяжем к – третьему аргументу шаблона экрана типа **«IN»** с именем «**Сумма**», который создадим в процессе привязки.

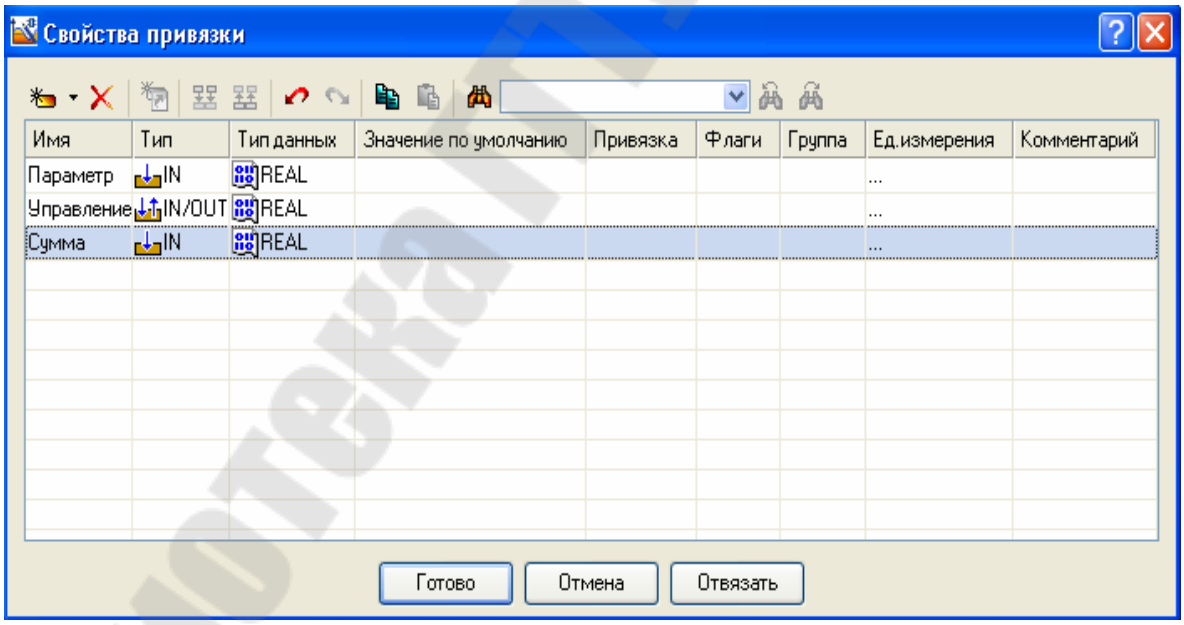


Рисунок 2.38 – Окно создания аргумента

Добавим еще одну кривую на тренд с привязкой к аргументу «**Сумма**».

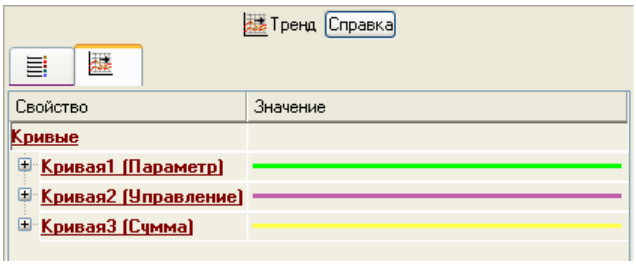


Рисунок 2.39 – Окно настройки ГЭ – тренд

2.11 Создание программы на языке Техно ST

Создадим программу, в которой сумма двух аргументов, связанных с атрибутами «**Реальное значение**» каналов «**Параметр**» и «**Управление**», будет помещается в третий аргумент – «**Сумма**». Затем свяжем аргументы шаблонов для вывода на экран результата работы программы без создания дополнительного канала.

Двойным щелчком ЛК откроем узел **RTM_1** и создадим в нем компонент «**Программа**».

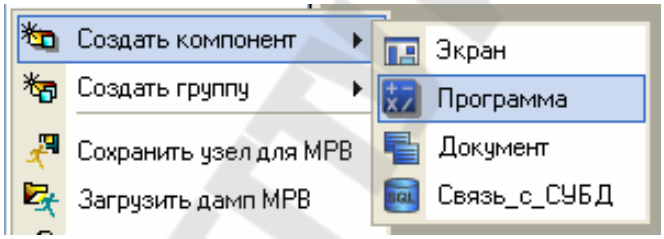


Рисунок 2.40 – Создание компонента – «Программа»

Двойным щелчком ЛК по компоненту **Программа#1** перейдем в режим ее редактирования.

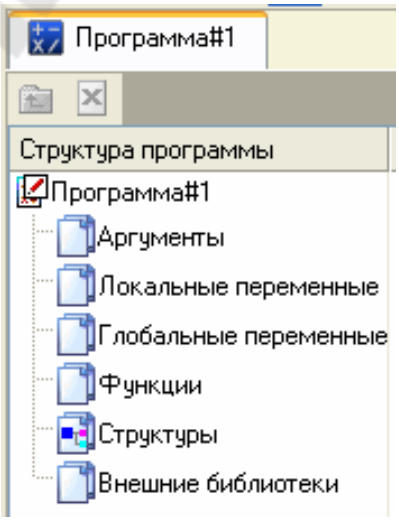


Рисунок 2.41 – Окно компонента – «Программа»

Выделением ЛК в дереве шаблона «**Программа#1**» строки «**Аргументы**» вызовем табличный редактор аргументов, в котором с помощью иконки создадим в редакторе аргументов три аргумента с именами «**Параметр**», «**Управление**» и «**Сумма**». При этом первые два **аргумента** должны быть типа «**IN**», а третий – «**OUT**».

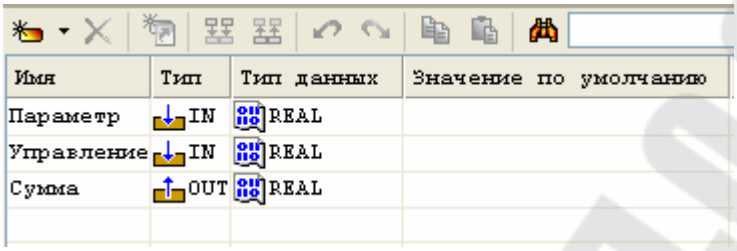


Рисунок 2.42 – Окно аргументов компонента – «Программа»

Выделим ЛК в дереве шаблона строку «**Программа#1**» и в открывшемся диалоге **Выбор языка** выберем язык **ST**.

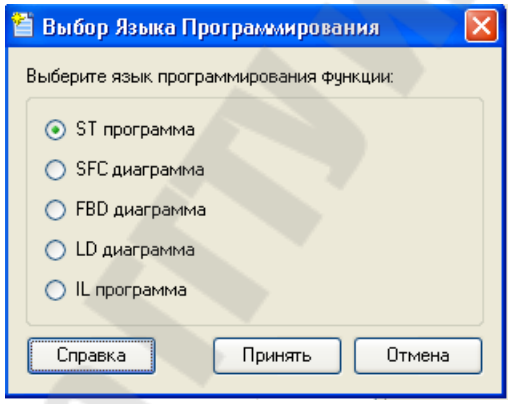


Рисунок 2.43 – Окно выбора языка программы

По нажатию экранной кнопки «**Принять**» откроется окно редактора программы, с объявленными переменными. Добавим в него следующий программный код – рисунок 2.44.

```
PROGRAM
    VAR_INPUT Параметр : REAL; END_VAR
    VAR_INPUT Управление : REAL; END_VAR
    VAR_OUTPUT Сумма : REAL; END_VAR

Сумма=Параметр+Управление;

END_PROGRAM
```

Рисунок 2.44 – Окно редактора программы

С помощью иконки  на инструментальной панели редактора или нажатием **F7** скомпилируем программу и убедимся в успешной компиляции в окне «**Выход** (Output)», вызываемом из инструментальной панели с помощью иконки  .

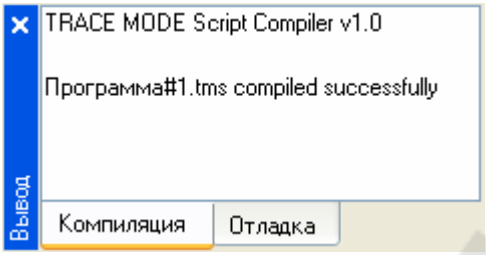


Рисунок 2.45 – Окно результатов компиляции программы

2.12 Привязка аргументов программы

Выполним привязку **аргументов** программы к **атрибутам** каналов**:**

– вызовем свойства компонента «**Программа#1**» через контекстное меню;

– выберем вкладку «**Аргументы**». Двойным нажатием в поле «**Привязка**» свяжем **аргументы программы** с **атрибутами каналов – аргумент** «**Параметр**» к **реальному значению** канала «**Параметр**», аргумент «**Управление**» к реальному значению канала «**Управление**»;

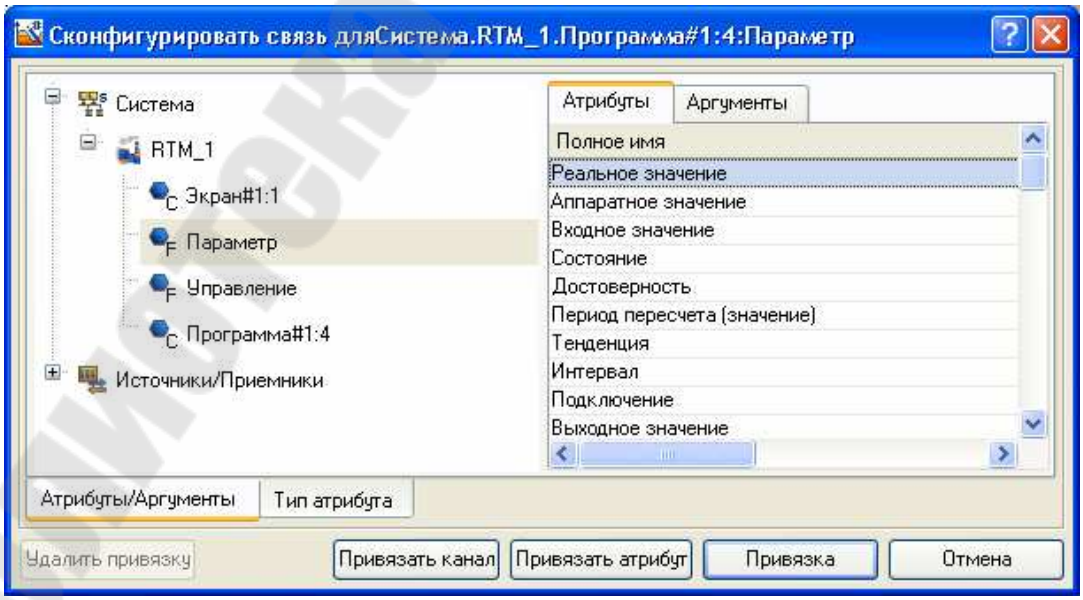


Рисунок 2.46 – Окно настройки привязки аргументы программы с атрибутами каналов

Двойным щелчком в поле «**Привязка**» аргумента программы «**Сумма**» вызовем окно настройки связи, выберем в левом окне канал класса **CALL Экран#1**, а в правом откроем вкладку **Аргументы** и укажем в ней аргумент **Сумма**, затем щелчком ЛК по экранной кнопке **Привязка** подтвердим связь.

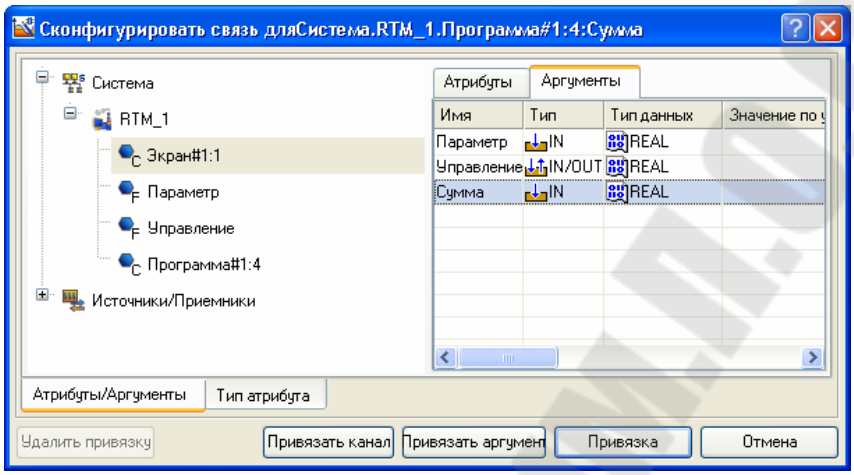


Рисунок 2.47 – Окно конфигурирования связи аргумента программы «**Сумма**»

В результате, будем иметь следующие связи – рисунок 2.48.

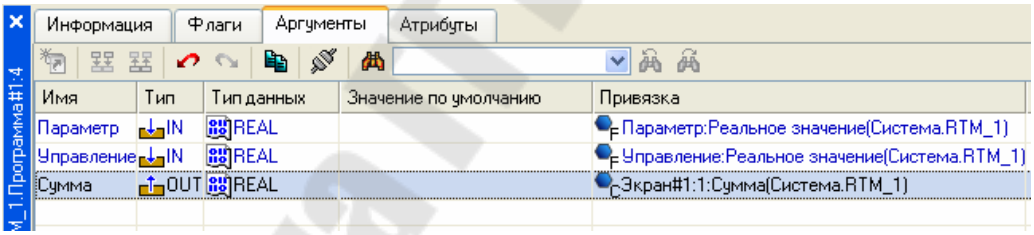


Рисунок 2.48 – Окно привязок компонента «Программа#1»

2.13 Запуск проекта

Сохраним проект с помощью иконки . На инструментальной панели выберем ЛК иконку и подготовим тем самым проект для запуска в реальном времени. С помощью иконки на инструментальной панели запустим режим исполнения. Задавая с помощью кнопки «**Управление**» управляющие воздействия, будем наблюдать изменение реального значения канала «**Управление**».

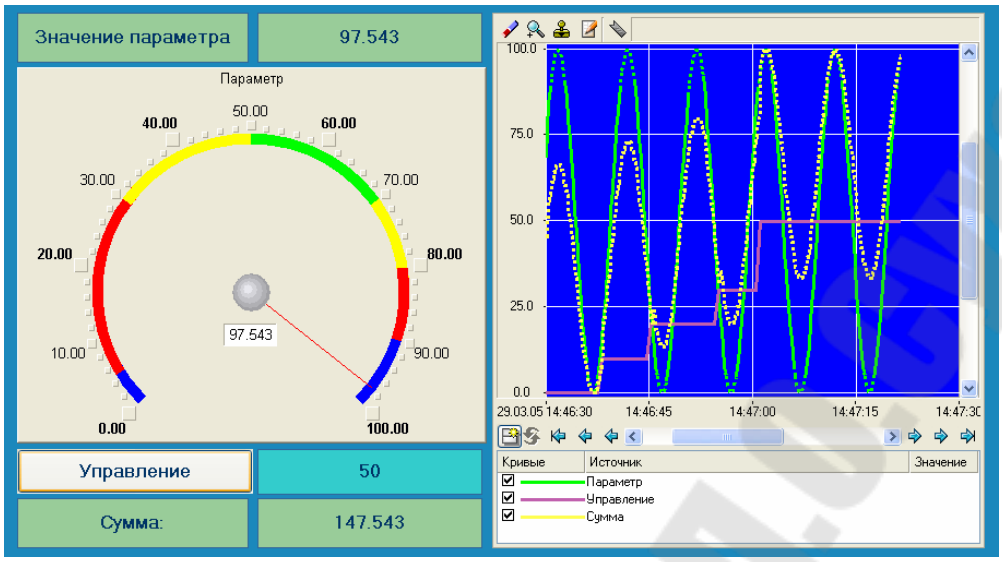


Рисунок 2.49 – Вид окна работы профайлера

2.14 Связь по протоколу DDE с приложением MS Windows на примере Excel MPB как DDE-сервер

Организуем запрос реальных значений каналов узла разработанного проекта приложением MS Windows в качестве, которого выберем книгу MS Excel. Для этого выполним:

– запуск приложения MS Excel;
– запишем в двух ячейках первого столбца запросы на получение данных:
  =RTM0|GET!Параметр
  =RTM0|GET!Управление
  где **0** – индивидуальный номер узла в проекте;
– запустим на исполнение узел АРМ RTM_1;
– в меню таблицы MS Excel «**Правка**» выберем команду «**Связи**», выделим оба параметра и нажмем кнопку «**Обновить**», после чего закроем окно кнопкой ОК.

Убедимся, что значения в ячейках книги Excel изменяются вместе с соответствующими реальными значениями каналов узла (значения канала «**Параметр**» меняется постоянно, а канала «**Управление**» – после введения нового значения с помощью ГЭ Кнопка):

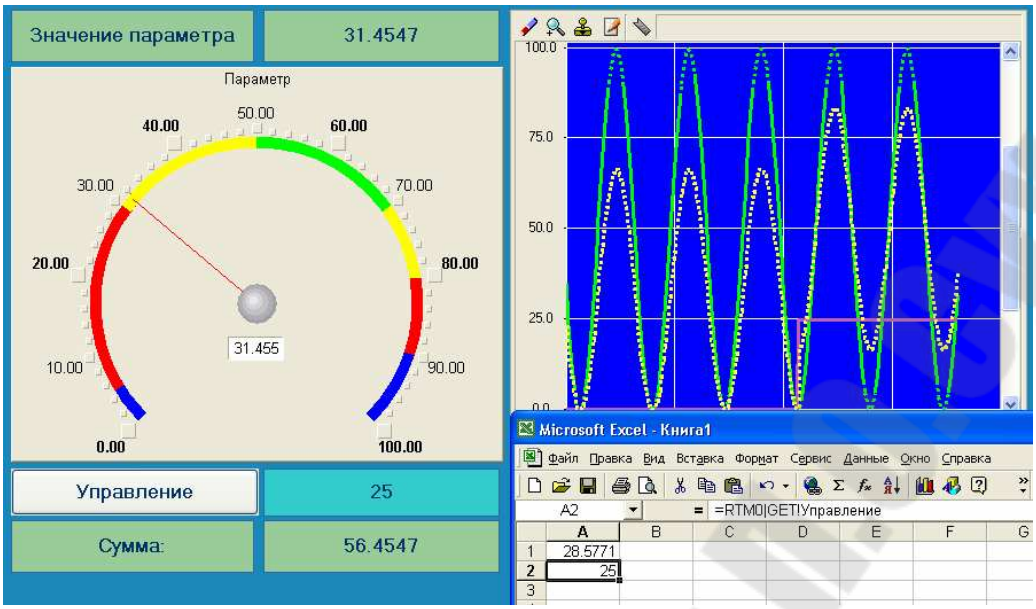


Рисунок 2.50 – Вид окна работы профайлера и MS Excel

3. Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, созданные каналы и привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции.

4. Контрольные вопросы

1. Назначение SCADA систем.
2. Основные элементы проекта TRACE MODE.
3. Виды узлов проекта.
4. Основные элементы операторского интерфейса.
5. Канал. Назначение. Основные понятия.
6. Понятие шаблонов, классификация слоев
7. Классификация узлов
8. Понятие шаблонов, классификация слоев
9. Этапы создания систем управления в TRACE MODE
10. Аргументы, назначение, настройка полей

Лабораторная работа №2
Изучение работы монитора и обработки данных в числовых каналах TRACE MODE.

Цель работы: Изучить алгоритм работы мониторов, виды числовых каналов обработку, данных в каналах, основные атрибуты каналов, пересчет каналов.

1. Теоретические сведения

1.1 Работа монитора TRACE MODE. Канал данных

При старте монитор считывает параметры узла, заданные в ходе разработки проекта в ИС, а также параметры других узлов для корректного взаимодействия с ними.

Алгоритм работы любого монитора TRACE MODE заключается в анализе **каналов** – структур переменных, создаваемых как при разработке проекта в ИС, так и в реальном времени. В зависимости от класса и конфигурации канала, по результатам его анализа монитор выполняет ту или иную операцию – запись значений переменных канала в архив, запрос значения источника данных по указанному интерфейсу и запись этого значения в канал, вызов графического экрана оператора на дисплей и т.п.

Под записью значения в канал в общем случае понимается присвоение значения переменной (атрибуту) **Входное значение** этого канала. Для канала могут быть сконфигурированы два важнейших свойства – **связь** и **вызов**.

Первое свойство означает способность канала принимать данные от источников и передавать данные приемникам – другими словами, с помощью этого свойства можно конфигурировать информационные потоки АСУ.

Второе свойство означает способность канала вызвать (реализовать) шаблон с передачей ему необходимых параметров (для канала класса **CALL** свойство **вызов** имеет расширенные функции). На основе свойства **вызов** реализуется, например, графический интерфейс оператора, обмен с базой данных и т.д.

Совокупность каналов узла называется **базой каналов** этого узла. **Класс канала** определяет его общее назначение. При разработке проекта могут быть созданы каналы только предопределенных классов.

Переменные, входящие в канал, называются его **атрибутами**. Атрибуты канала имеют различное назначение и различный тип данных. Булевы атрибуты и атрибуты, которые могут принимать только два определенных значения, называются **флагами**. Примером флага может служить **тип** канала, который принимает два значения – INPUT (числовые каналы типа INPUT предназначены для приема данных от источников) и OUTPUT (числовые каналы типа OUTPUT предназначены для передачи своего значения приемникам). Атрибуты, которые используются для передачи значений при вызове шаблона, называются **аргументами** канала. Атрибуты снабжены числовыми **индексами** (индексация атрибутов начинается с 0, индексация аргументов – с 1000). Атрибуты имеют **полное имя** и **короткое имя** (мнемоническое обозначение). Идентификаторами атрибута являются его индекс и, в ряде случаев, короткое имя.

Каналы содержат внутри себя предопределенные алгоритмы (часть из них может настраиваться пользователем), в соответствии с которыми некоторые атрибуты канала устанавливаются или вычисляются монитором в зависимости от состояния или значения других атрибутов. Исполнение внутренних алгоритмов канала и анализ его атрибутов монитором называется **пересчетом канала**.

В этих алгоритмах, в общем случае, задействованы 4 атрибута – «**Входное значение**» (2, In), «**Аппаратное значение**» (1, A), «**Реальное значение**» (O, R) и «**Выходное значение**» (9, Q).

При пересчете числовых каналов выполняется также процедура трансляции. **Трансляцией** называется вызов программы числовым каналом (это единственное назначение свойства **вызов** числового канала).

При вызове программы числовым каналом (кроме канала TIME) может быть выполнено следующее преобразование его атрибутов:
аппаратное значение (A)<=>программа<=>реальное значение (R)

Направление преобразования зависит от типа канала:

– INPUT: **A=>программа=>R**
– OUTPUT: **R=>программа=>A**

Направление преобразования следует учитывать для корректной привязки атрибутов числового канала к аргументам программы. По результатам анализа атрибутов монитор выполняет действия, заданные с помощью канала (например, вызов шаблона), – эта процедура называется **отработкой канала**. Отработка канала после его пересчета выполняется при определенных условиях. При пересчете базы ка-

налов пересчет конкретного канала также выполняется при определенных условиях.

Каналы одного класса обладают идентичным набором атрибутов и предопределенных алгоритмов их обработки. Существуют также атрибуты, которыми обладают все каналы вне зависимости от их класса (такие атрибуты имеют одинаковые индексы во всех каналах).

К основным общим атрибутам каналов, настраиваемым с помощью редактора канала, можно отнести следующие:

- **Имя** – (127, **B_NAME**, MPB – **NAME**) – имя канала;
- **Кодировка** – (79, **CODE**) – кодировка канала;
- **Комментарий** – (80, **CMNT**, MPB – **COMMNT**) – произвольный текст, MPB считывает первые 39 символов комментария.

На вкладке «**Основные**» редактора канала редактируются следующие атрибуты канала:

- **Тип** – (81, **TYPE**, MPB – **IO**, в окне свойств – флаг **Тип OUTPUT**) – тип канала: INPUT (0) или OUTPUT (1).;
- **Размерность** – (82, **DIM**) – размерность реального значения канала. Этот параметр выбирается из списка, который хранится в текстовом файле **tmcf/dimension.tmc**. Если требуемая размерность в списке отсутствует, то ее можно добавить, отредактировав указанный файл в редакторе;
- **Период** – (5, **FRQ**) – значение периода пересчета канала;
- **Единица измерения** – (38, **FRQ_D**) – единицы измерения периода пересчета канала, выбирается из списка;
- **Включить** – (60, **T_NET**, MPB – **ToNet**, в окне свойств – флаг **Разрешить**) – при установке этого флага монитор будет передавать в сеть реальное значение канала при каждом его изменении в виде широковещательного сообщения. На других узлах такое сообщение принимается каналами, которые связаны с данным;
- **Отработать** – (39, **EXEC**) – установка этого флага является признаком необходимости отработки канала. Установка флага **Отработать** в редакторе задает пересчет и отработку канала при старте монитора;
- **На старте** – (2, **In**) – значение, указанное в этом поле, присваивается атрибуту (2, **In**) канала при старте монитора.

На вкладке «**Дополнительно**» редактора канала редактируются следующие атрибуты канала:

– **Отключить от источника** – (8, **W**) **Подключение** – при установке этого флага канал отключается от источника/приемника;
– **Выключить** – (3, **C**) **Состояние** – установка этого флага означает остановку пересчета канала;
– **Привязка** (86, **LN_ATTR**, MPB – **nAtrt**) – задание свойства **связь.**

1.2 Канал FLOAT. Обработка данных

В измерительном тракте (в общем случае датчик => УСО => контроллер) происходит преобразование реальной физической величины (температуры, давления и т.п.) в один из следующих "инженерных" видов:

– в число, соответствующее амплитуде некоторого электрического сигнала (в том числе унифицированного – 0-10V, 4-20mA и т.д.);
– в число, соответствующее проценту от диапазона изменения некоторого электрического сигнала;
– в двоичный код (после АЦП).

В управляющем тракте (в общем случае контроллер => УСО => исполнительный механизм) выполняется обратное преобразование.

При обработке данных, поступающих из измерительного тракта или передаваемых в управляющий, необходимо скорректировать различные погрешности трактов. Для отображения поступающих данных требуется переводить "инженерные" данные в реально измеряемые (например, если требуется отображать значение температуры в ее физических единицах – градусах). Управляющий сигнал во многих случаях требуется сглаживать. Для решения подобных задач канал **FLOAT** снабжен встроенными алгоритмами обработки, параметры которых могут быть заданы как в редакторе, так и в реальном времени:

– канал **INPUT**: масштабирование, фильтрация одиночных пиков, фильтрация малых изменений (апертура), экспоненциальное сглаживание.
– канал **OUTPUT**: экспоненциальное сглаживание, линейное сглаживание, фильтрация малых изменений (апертура), клиппирование, масштабирование.

При использовании экспоненциального сглаживания фильтрация малых изменений в канале FLOAT не выполняется.

Если встроенных алгоритмов обработки данных недостаточно, в каналах **FLOAT** может быть использована процедура трансляции – например, для корректировки нелинейности передаточной характеристики измерительного/управляющего тракта.

Атрибуты **Входное значение** (2, **In**), **Аппаратное значение** (1, **A**), **Реальное значение** (0, **R**) и **Выходное значение** (9, **Q**) задействованы во внутренних алгоритмах канала FLOAT следующим образом – рисунок 2.1. В указанных на рисунках формулах масштабирования **KX** и **Z** являются значениями атрибутов **Множитель** (33, **KX**) и **Смещение** (34, **Z Дрейф нуля**, МРВ – **ZERO**).

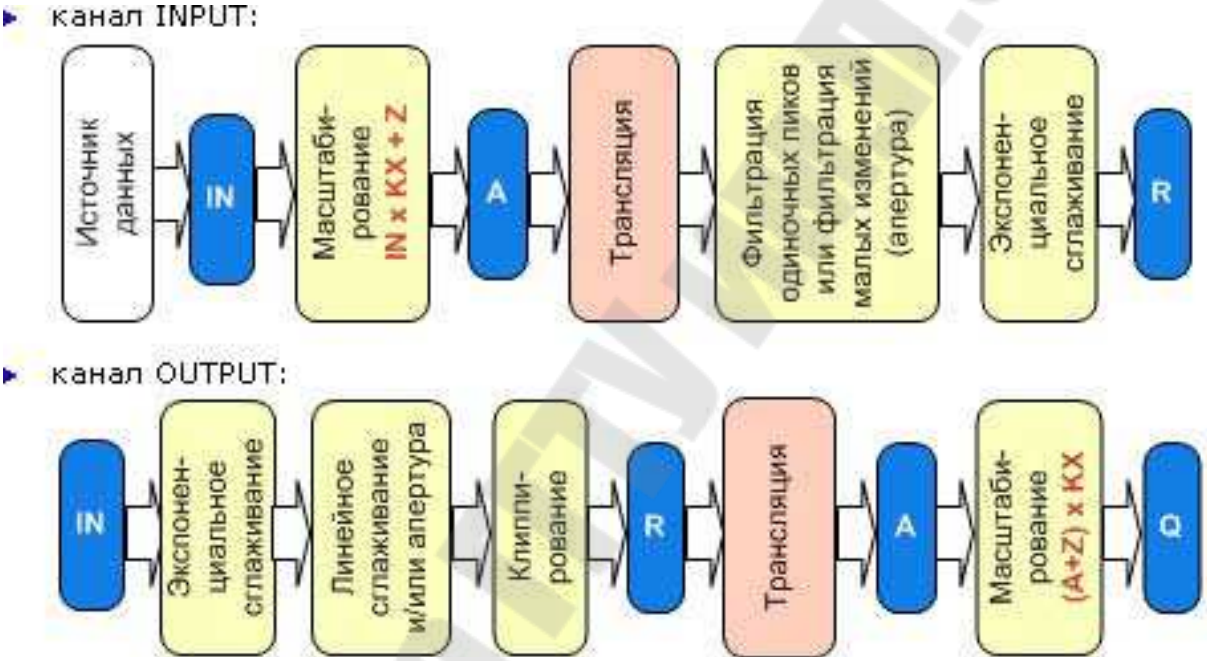


Рисунок 1.1 – Структура канала типа FLOAT

Всё алгоритмы обработки (за исключением клиппирования реального значения в канале OUTPUT) выполняются в том случае, если для канала установлен флаг (50, **PRS Использовать**) – такой канал является **каналом FLOAT с обработкой.**

1.2.1 Масштабирование

Эта процедура используется только в каналах, работающих с аналоговыми переменными. Масштабирование в канале FLOAT выполняется по следующим законам (при **PRS=1**):

– в канале типа INPUT: **A=IN*KX+Z**;
– в канале типа OUTPUT: **Q=(A+Z)*KX;**

1.2.2 Экспоненциальное сглаживание

Этот метод вводит апериодичность изменения реального значения канала по отношению к аппаратному у каналов типа INPUT и по отношению к входному у каналов типа OUTPUT. Используя его, можно задать инерционность канала.

Данный метод фильтрации позволяет уменьшить величину случайных колебаний измеряемых значений параметров. Такие колебания могут быть обусловлены наличием шумов в измерительном тракте.

При **PRS**=1 и **DSmoot**<>0 (**0<DSmoot<=l**) в канале FLOAT выполняется экспоненциальное сглаживание (n - индекс указывает номер такта пересчета):

– в канале типа INPUT:
  $\mathbf{R_n = Result_n(1 - DSmoot_n) + DSmoot_n * R_{n-1}}$
  $\mathbf{Result_n}$ обозначает результат фильтрации пиков;
– в канале типа OUTPUT:
  $\mathbf{RESULT_n = IN_n(1 - DSmoot_n) + DSmoot_n * R_{n-1}}$

Результат экспоненциального сглаживания $\mathbf{RESULT_n}$ далее обрабатывается процедурой линейного сглаживания. При **DSmoot<>0** фильтрация малых изменений в канале FLOAT не выполняется.

1.2.3 Фильтрация пиков или фильтрация малых изменений в канале INPUT

Данная процедура присутствует только у аналоговых каналов. Набор выполняемых ею операций отличается для входных и выходных каналов. У каналов типа INPUT фильтрация выполняется после процедуры трансляции до формирования реального значения и включает в себя следующие операции:

– подавление случайных всплесков в тракте измерения;
– подавление малых колебаний значения канала;
– экспоненциальное сглаживание;
– контроль шкалы – отслеживание выхода реального значения канала за установленные границы шкалы.

При **PRS=1** в канале INPUT выполняется фильтрация пиков или фильтрация малых изменений (апертура). **Фильтрация пиков** - это

алгоритм, позволяющий игнорировать в течение одного такта пересчета изменение значения сверх установленной величины (**DPic**).

Фильтрация малых изменений - это алгоритм, позволяющий игнорировать изменение значения, если это изменение меньше заданной величины (**APert** – зону нечувствительности). Введение этого метода позволяет существенно сократить интенсивность информационных потоков, а также увеличить глубину сохранения и скорость доступа к данным в архивах.

При использовании экспоненциального сглаживания фильтрация малых изменений в канале FLOAT не выполняется. При обработке значения в канале выполняется один из этих алгоритмов фильтрации (n - индекс указывает номер такта пересчета):

- если $|\mathbf{RES_n}-\mathbf{R_{n-i}}| <= \mathbf{DPic_n}$, то выполняется фильтрация малых изменений:
  - если $|\mathbf{RES_n}-\mathbf{R_{n-i}}| < \mathbf{APert_n}$, то $\mathbf{Result_n} = \mathbf{R_{n-1}}$;
  - если $|\mathbf{RES_n}-\mathbf{R_{n-i}}| >= \mathbf{APert_n}$, то $\mathbf{Result_n} = \mathbf{RES_n}$;
- если $|\mathbf{RES_n}-\mathbf{R_{n-i}}| > \mathbf{DPic_n}$, то выполняется фильтрация одиночных пиков:
  - если на такте (**n-1**) фильтрация пиков не использовалась, $\mathbf{Result_n}=\mathbf{R_{n-1}}$ (первый пик фильтруется);
  - если на такте (**n-1**) фильтрация пиков использовалась, $\mathbf{Result_n}=\mathbf{RES_n}$ (второй подряд пик не фильтруется).

где

$\mathbf{RES_n}$ - результат процедуры трансляции или в ее отсутствие, аппаратное значение канала;

$\mathbf{Result_n}$ - результат описываемых алгоритмов (на n-ом такте пересчета).

1.2.4 Линейное сглаживание и фильтрация малых изменений в канале OUTPUT

У каналов типа OUTPUT данная процедура формирует реальное значение по входному значению. При этом выполняются следующие операции:

- ограничение скорости изменения реального значения;
- подавление малых колебаний значения канала;
- экспоненциальное сглаживание;
- контроль шкалы – обрезание величины управляющего воздействия до границ шкалы канала.

При **PRS**=1 в канале OUTPUT выполняется линейное сглаживание и/или фильтрация малых изменений (n - индекс указывает номер такта пересчета):

- если $|\mathbf{RESULT_n - R_{n-i}}| <= \mathbf{DPic_n}$, то выполняется фильтрация малых изменений:
  - если $|\mathbf{RESULT_n - R_{n-i}}| <= \mathbf{APert_n}$, то $\mathbf{Res_n = R_{n-1}}$;
  - если $|\mathbf{RESULT_n - R_{n-i}}| > \mathbf{APert_n}$, то $\mathbf{Res_n = RESULT_n}$;
- если $|\mathbf{RESULT_n - R_{n-i}}| > \mathbf{DPic_n}$, то вычисляется результат линейного сглаживания $\mathbf{LIN_n = Rn\text{-}i + DPic_n}$ (если $\mathbf{RESULT_n > R_{n-1}}$) или $\mathbf{LIN_n = R_{n-1} - DPic_n}$ (если $\mathbf{RESULT_n < R_{n-1}}$) и далее выполняется фильтрация малых изменений:
  - если $|\mathbf{LIN_n - R_{n-1}}| <= \mathbf{APert_n}$, то: $\mathbf{Resn = R_{n-1}}$;
  - если $|\mathbf{LIN_n - R_{n-1}}| > \mathbf{APert_n}$, то: $\mathbf{Resn = LIN_n}$;

где

$\mathbf{RESULT_n}$ - результат предыдущей процедуры;

$\mathbf{Res_n}$ - результат описываемых алгоритмов на n-ом такте пересчета. $\mathbf{Res_n}$ при **BNDR=0** и **LMT=1** обрабатывается далее процедурой клиппирования.

При использовании экспоненциального сглаживания фильтрация малых изменений в канале FLOAT не выполняется.

1.2.5 Клиппирование в канале OUTPUT

Клиппирование – ограничение реального значения в канале FLOAT типа OUTPUT по максимуму и минимуму. Ограничение реального значения в канале FLOAT типа OUTPUT выполняется по следующим законам (при **BNDR**=0 и **LMT=1):**

- если **Res > HL,** то **R = HL;**
- если **Res < LL,** то **R = LL;**
- если **LL <= Res <= HL,** то **R = Res.**

где **Res -** результат предыдущей процедуры обработки (линейного сглаживания и/или фильтрации малых изменений).

1.3 Атрибуты канала FLOAT

Кроме общих для числовых каналов каналы класса **FLOAT имеют** специфические атрибуты, которые могут быть заданы в редакторе **канала FLOAT:**

1. раздел «**Границы**»:

- флаг **Использовать - (85, BNDR) - установка этого флага в редакторе** равнозначна присвоению атрибуту **BNDR** значения 0, что разрешает монитору анализировать значения шести границ канала. При **BNDR**=1 (флаг снят**)** анализ границ запрещен. От этого флага зависит исполнение алгоритма клиппирования в канале **OUTPUT;**
- **ВП** - (26, **HL)** - значение верхнего предела;
- **ВА** - (28, **HA)** - значение верхней аварийной границы;
- **ВГ** - (30, **HW)** - значение верхней предупредительной границы;
- **НГ** - (3**1, LW)** - значение нижней предупредительной границы;
- **НА** - (29, **LA)** - значение нижней аварийной границы;
- **НП** - (27, **LL)** - значение нижнего предела;
- **Гистерезис** - (32, **Hyst)** - от этого параметра зависят условия генерации сообщений при переходе реальным значением канала заданных границ:
  в сторону увеличения номера интервала:
  <LL, <LA, <LW, >HW, >HA, >HL
  в сторону уменьшения номера интервала:
  >(LL+H), >(LA+H), >(LW+H), <(HW-H), <(HA-H), <(HL-H).
- **флаг Контроль границ - (53, SC_F, MPB - LMT)** – установка этого флага равнозначна присвоению атрибуту **LMT** значения 1. Действие флага различно для каналов типов INPUT и OUTPUT. В первом случае наличие флага означает разрешение установки каналу признака программной недостоверности (**FS**=1) в случае выхода **реального значения** канала за пределы диапазона **[LL, HL]** (при **BNDR=0).** При возврате реального значения в диапазон признак программной недостоверности автоматически сбрасывается (**FS**=0). Для типа OUTPUT установка флага «**Контроль границ»** разрешает клиппирование **реального значения** канала (при **BNDR=0).** При **LMT**=0 или **BNDR**=1 описанные алгоритмы не исполняются;

2. раздел «**Обработка**»:
   - флаг «**Использовать»** – (50, **PRS**, недоступен для изменения в реальном времени) - если флаг снят, канал является кана-

лом FLOAT без обработки, если флаг установлен - каналом FLOAT с обработкой;

- **Апертура** - (35, AP, MPB - **APert)** - этот параметр конфигурирует алгоритм фильтрации малых изменений значения. По умолчанию **APert = 0**;
- **Пик** - (36, DP, MPB - **DPic**) - этот параметр конфигурирует алгоритм подавления одиночных пиков в канале INPUT и алгоритм линейного сглаживания - в канале. По умолчанию **DPic=10000**;
- **Сглаживание** - (37, DS Экспоненциальное сглаживание, **MPB - DSmoot**) - коэффициент (**0<=DSmoot<-1**) в стандартном алгоритме экспоненциального сглаживания**.** При **DSmoot=0** (значение по умолчанию) этот алгоритм не выполняется;
- **Множитель (33, KX) и Смещение (34, Z Дрейф нуля,** MPB **- ZERO) –** параметры масштабирования (по умолчанию **KX=1, Z=0**):
  A=In*KX+Z – в канале типа INPUT;
  Q=(A+Z)*KX – в канале типа OUTPUT.

Атрибуты **Множитель** и **Смещение** могут быть также рассчитаны в разделе **Масштабирование** (для активизации раздела нужно установить флаг Масштабирование). Этот раздел, в зависимости от типа канала (INPUT или OUTPUT), имеет вид соответствующей формулы преобразования. Для канала типа INPUT – рисунок 1.2, для канала типа OUTPUT – рисунок 1.3.

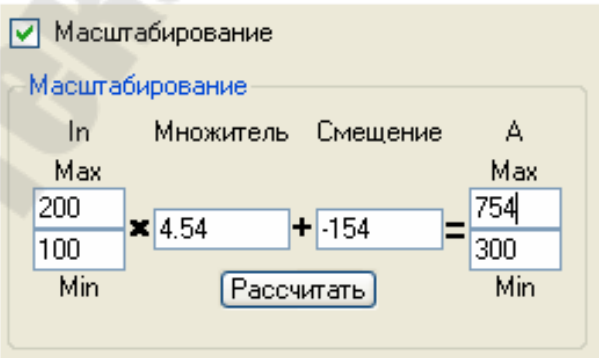


Рисунок 1.2 – Настройка маштабирования канала типа INPUT

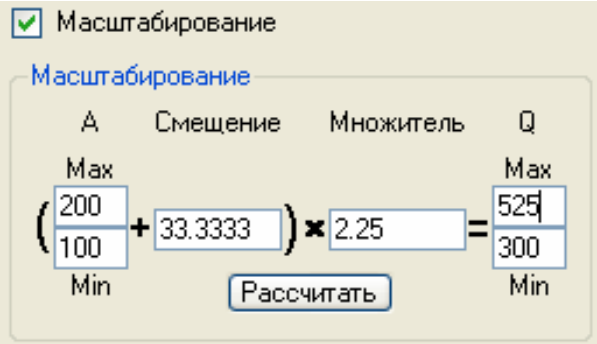


Рисунок 1.3 – Настройка маштабирования канала типа OUTPUT

Для расчета множителя и смещения нужно **ввести** входной диапазон (диапазон изменения атрибута **In** для канала **INPUT или** A для канала OUTPUT), выходной диапазон (диапазон изменения атрибута **A** для канала **INPUT** или Q для канала OUTPUT) и нажать кнопку «**Рассчитать».** Для задания диапазонов используются соответствующие поля **Min** и **Max.**

1.4 Границы и интервалы канала FLOAT

Для мониторинга состояния техпроцесса для каналов FLOAT могут быть заданы 6 **границ** (рисунок 1.4). Границы задают диапазоны (интервалы), в которых может находиться значение отслеживаемого параметра.

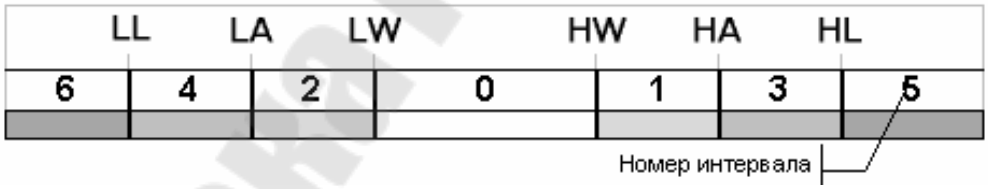


Рисунок 1.4 – Интервалы канала класса FLOAT

Если задано не менее двух ненулевых значений границ и полный перечень заданных значений границ корректен (LL<=LA<=LW<=HW<=HA<=HL), монитор на каждом цикле пересчета определяет номер интервала, в котором находится реальное значение канала, и записывает этот номер в атрибут **Интервал** (7, P).

С помощью флага (53, LMT) «**Контроль границ»** можно задать режим установки каналу типа INPUT признака программной недостоверности в случае выхода его значения за пределы диапазона **[LL, HL].** Этот режим может быть использован в том случае, если в канал записывается некоторая величина, диапазон изменения которой зара-

нее известен, и всякое значение вне этого диапазона, принятое в канал, является следствием какой-либо ошибки или сбоя, поэтому не может быть использовано для анализа.

Если флаг «**Контроль границ»** установлен для канала типа OUTPUT, реальное значение этого канала при выходе из диапазона **[LL, HL]** клиппируется (ограничивается).

1.5 Генерация сообщений. Гистерезис

Условия генерации сообщений при пересечении реальным значением канала заданных границ зависят от значения атрибута «**Гистерезис» (32, Hyst**) (см. раздел 1.3).

Введение гистерезиса позволяет убрать ненужный поток сообщений в отчет тревог при небольших колебаниях контролируемого параметра вблизи значения одной из границ.

1.6 Канал класса DOUBLE FLOAT

Каналы класса **DOUBLE FLOAT** имеют специфические атрибуты, к которым можно отнести раздел «**Границы**»:

- флаг **Использовать** - (85, **BNDR) -** установка этого флага равнозначна присвоению атрибуту **BNDR** значения 0, что разрешает монитору анализировать значения двух границ канала (атрибутов **HL** и **LL).** При **BNDR**=1 границы не анализируются;
- **ВП -** (26, **HL) -** значение верхней границы;
- **НП -** (27, **LL) -** значение нижней границы.

Границы канала задают диапазоны (интервалы), в которых может находиться его реальное значение рисунок 1.5.

Рисунок 1.5 – Интервалы канала класса DOUBLE FLOAT

Монитор на каждом цикле пересчета определяет номер интервала, в котором находится реальное значение канала, и записывает этот номер в атрибут **Интервал** (7, **Р).** Для каналов DOUBLE FLOAT мо-

жет быть определена процедура трансляции. Структура канала DOUBLE FLOAT представлена на рисунке 1.6.

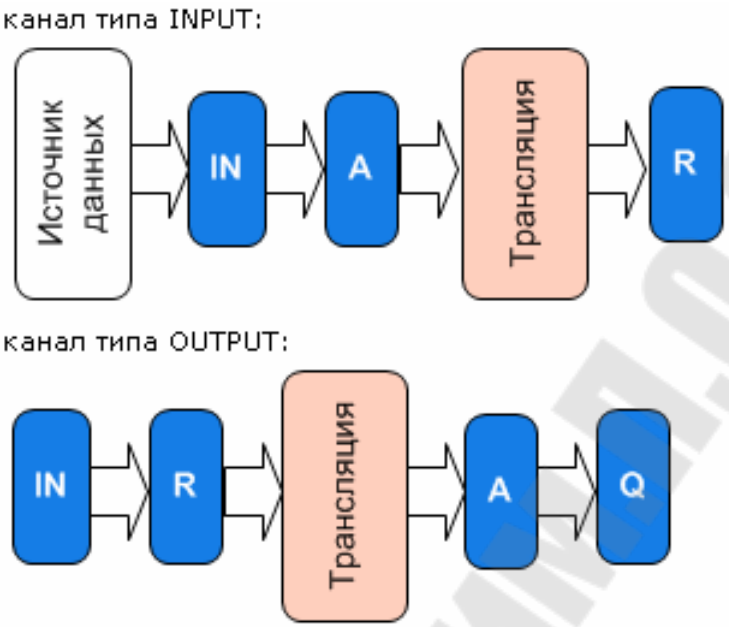


Рисунок 1.6 – Структура канала типа DOUBLE FLOAT

1.7 Канал класса HEX16 и HEX32

Канал класса **HEX16** предназначен для работы с 2-байтовыми целыми числами (**HEX32** – 4 байта). Кроме атрибутов, которые имеют каналы всех классов и атрибутов, общих для числовых каналов, каналы класса HEX имеют специфические атрибуты. К специфическим атрибутам, которые могут быть заданы в редакторе канала, относятся следующие атрибуты раздела «**Параметры**»:

- **Размерность в битах** – (56, **nBits**) **Number Bits** – данный атрибут задает число байт, участвующих в процедуре инверсии:
  <= 8 – 1 байт (<=24-3 байта для HEX32);
  > 8 – 2 байта (>24-4 байта для HEX32);
- флаг **Инверсия** (40, **NM**) – если этот флаг установлен, инвертирование в канале разрешено;
- флаг **DEC** (84, **HD**) – если этот флаг установлен (**HD**=1), значение канала отображается в профайлере в десятичном виде; если флаг не установлен (**HD**=0) – в шестнадцатеричном. От этого флага зависит также алгоритм записи сообщений в отчет тревог.

Атрибуты **Входное значение** (2, **In**), **Аппаратное значение** (1, **A**), **Реальное значение** (0, **R**) и **Выходное значение** (9, **Q**) канала

HEX16 задействованы в его алгоритмах обработки следующим образом – рисунок 2.7.

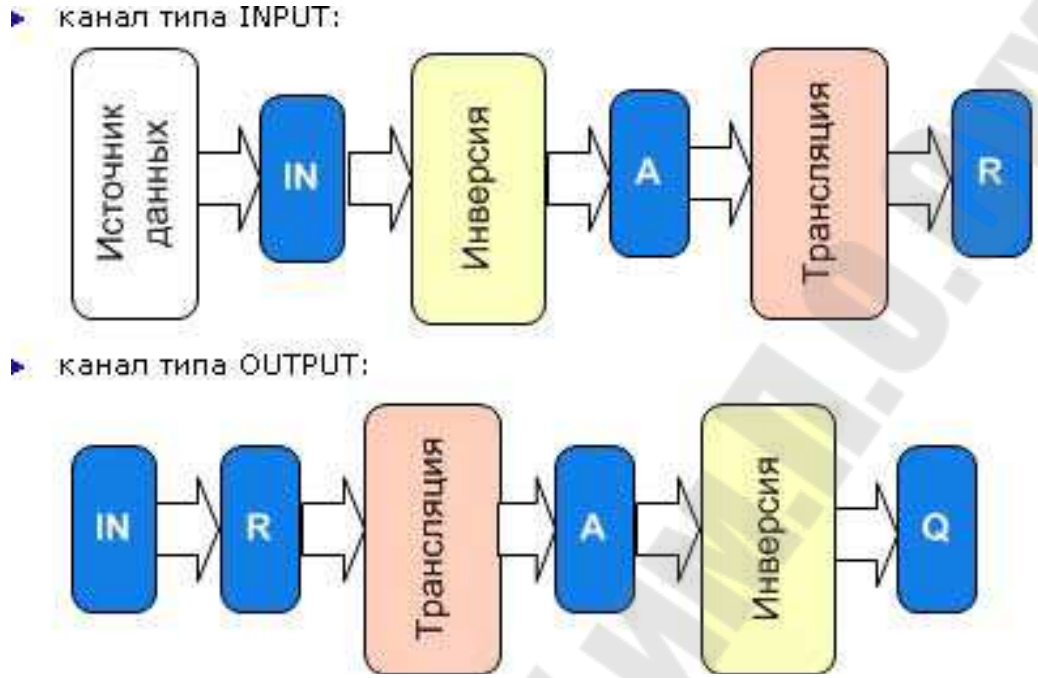


Рисунок 2.7 – Структура канала типа HEX

В отсутствие процедуры трансляции, в каналах HEX не используется атрибут **Аппаратное значение** (1, **A**).

1.8 Пересчет базы каналов монитором

Монитор - многопоточный процесс. Приоритеты потоков заданы по умолчанию, но при необходимости их можно изменить. Приоритет выбирается из следующего списка: -1 - Default; 0 - IDLE; 2 - LOWEST; 4 - BELOWNORMAL; 6 - NORMAL; 10 - ABOVENORMAL; 12 - HIGHEST; 14 - TIMECRITICAL; 77 - STOP.

Рассмотрим алгоритм пересчета базы каналов основным потоком (0 - CALC) монитора TRACE MODE. Один цикл включает следующие последовательно выполняемые этапы:

1. последовательный анализ всех включенных каналов узла и установка флага **SV** (недоступен для пользователя) каналам, требующим пересчета;
2. пересчет всех каналов (кроме каналов CALL) типа INPUT, которые должны пересчитываться в основном потоке;
3. пересчет и отработка каналов класса CALL основного потока;

4. пересчет каналов типа OUTPUT, которые должны пересчитываться в основном потоке, и анализ их выходного значения.

Если используются приоритеты потоков по умолчанию, и для цикла установлено время**,** недостаточное для выполнения всех его задач, система будет работоспособной, однако заданные временные характеристики пересчета/отработки каналов будут нарушены.

1.9 Время цикла монитора

Мониторы реального времени TRACE MODE работают как интерпретаторы базы каналов. Интерпретация базы каналов осуществляется один раз за цикл системы. Условием очередного пересчета базы каналов является начало нового цикла системы. Время цикла CALC (время, отводимое на однократное выполнение задач основного потока) настраивается с помощью двух параметров, которые задаются в разделе «**Пересчет»** вкладки «**Основные»** редактора узла (пункт «**Редактировать**» контекстного меню узла). Параметр «**Разрешение»** задает разрешение таймера в секундах (величина **tick),** параметр «**Период» –** период пересчета в единицах **tick.** Произведение этих параметров определяет время цикла CALC в секундах. Разрешение таймера **(tick)** может варьироваться в следующих пределах:

- в MS Windows - не менее 0.01 с;
- в MS Windows CE - не менее 0.001с;
- в MS DOS - в диапазоне 0.001с - 0.055с;
- в MiniOS7 и ROM-DOS - не менее 0.055с.

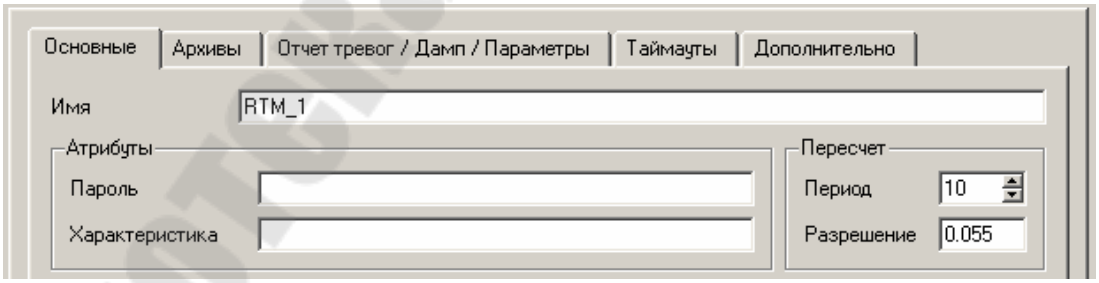


Рисунок 1.8 – Окно настройки времени цикла монитора

1.10 Период пересчета канала

Значение периода пересчета канала может устанавливаться либо в единицах времени (при этом оно должно быть кратно времени цик-

ла CALC), либо в циклах соответствующего потока**.** Если, например, период пересчета канала задан равным 2с, а время цикла CALC при этом равно 5с, то канал будет пересчитываться не чаще, чем 1 раз в 5с. При редактировании проекта в ИС начальное значение и единицы измерения периода задаются на вкладке «**Основные»** раздела «**Системные»** редактора канала.

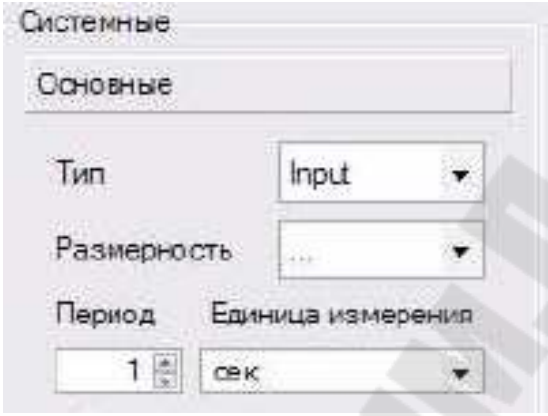


Рисунок 1.9 – Окно настройки пересчета канала

При редактировании в ИС единицы измерения периода канала выбираются из следующего списка (основные)**:**

- (0) **цикл CALC** - период в циклах CALC;
- **(**1**) сек** - период в секундах (1 - 60);
- (2) **мин** - период в минутах (1 - 60);
- (3) **час** - период в часах (1 - 24);
- (4) **флаги** - период в циклах CALC с блокировкой пересчета с помощью канала типа OUTPUT;
- (5) **F1** - период в циклах CALC с отставанием на 1 цикл;
- (6) **F2** - период в циклах CALC с отставанием на 2 цикла;
- (7) **F3** - период в циклах CALC с отставанием на 3 цикла;
- (8) **F4** - период в циклах CALC с отставанием на 4 цикла;
- (10) **на старте** - канал пересчитывается один раз при старте монитора;
- (11) **в нач. часа** - один раз в сутки в начале часа, заданного атрибутом **Период;**
- (12) **в нач. дня** - один раз в месяц в начале суток, номер которых задается атрибутом **Период;**
- (13) **по времени** - в указанное время;
- (14) **однократно** - канал пересчитывается один раз и выключается;

Условием пересчета канала, период которого задан в циклах системы, является равенство 0 остатка от деления счетчика циклов пересчета базы на его период. То есть при значении периода 1 пересчет будет осуществляться на каждом цикле, при 2 – через цикл, при 3 – через два цикла и так далее. Минимальное значение периода канала задает максимальную частоту пересчета его значений.

Если отсчет периода канала осуществляется в циклах системы, то для него можно установить **фазу** пересчета. **Фаза** определяет количество тактов отставания пересчета канала от момента выполнения условия пересчета. Введение фазы позволяет распределить нагрузку по пересчету значений каналов с одинаковым значением периода на разные такты системы. Значение фазы может лежать в диапазоне от 1 до 4. Для задания каналу соответствующей фазы надо в поле выбора размерности периода установить значения **F1**, **F2**, **F3** или **F4**.

Пример

Допустим, что в базе присутствуют 10000 каналов. Для всех период установлен равным 4. Если этим каналам не задать фазу, то они будут пересчитываться один раз в четыре цикла пересчета на одном и том же цикле. То есть в течение трех циклов нагрузка по пересчету базы каналов будет нулевая, а на одном – максимальная. В таком случае можно перегрузить систему вычислениями на четвертом цикле, что приведет к превышению реального времени пересчета базы каналов периода пересчета, установленного для узла.

Если же разбить каналы на четыре группы, по 2500 в каждой, и установить для них различные значения параметра Фаза (0, 1, 2, 3 соответственно), то нагрузка по пересчету базы каналов будет равномерно распределена во времени. В этом случае на каждом цикле системы будут пересчитываться только 2500 каналов. В такой ситуации, как и в первом случае, обновление данных во всех каналах будет осуществляться с одинаковой частотой. Однако это обновление будет сдвинуто по фазе для выделенных групп каналов.

2. Ход работы

Задание 1: создать канал, генератор заданного сигнала (синусоидального, пилообразного, треугольного – назначается преподавателем); произвести привязку генератора к созданному каналу; задать масштабирование входного сигнала, обеспечив выходной диапазон [-10, 10]; на экране вывести отмасштабированный сигнал с помощью стрелочного прибора, тренда, текста.

2.1. Создание проекта Trace Mode

Запустить программу TRACE MODE IDE 6 иконка  . Для создания нового проекта щелкните ЛК мыши на иконке  или выбрать в меню **Файл** пункт **Новый**. При отсутствии окна навигатора проекта в меню **Вид** выбрать **Навигатор проекта**. Сохранить созданный проект, щелкнув левой клавишей мыши на иконке  или на пункте **Сохранить** или **Сохранить как** в меню **Файл**.

2.2 Создание узла

В навигаторе проекта (рисунок 2.1) выделить строку «**СИСТЕМА**» и в контекстном меню выбрать «**СОЗДАТЬ УЗЕЛ**». Среди предложенных типов узлов выбрать – **RTM**.

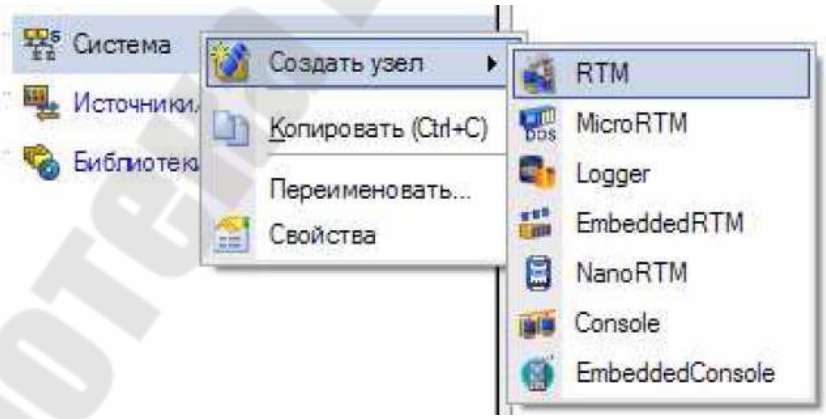


Рисунок 2.1 – Окно навигатора проекта

В разделе «**СИСТЕМА**» навигатор проекта, отобразит созданный узел **RTM** (рисунок 2.2).

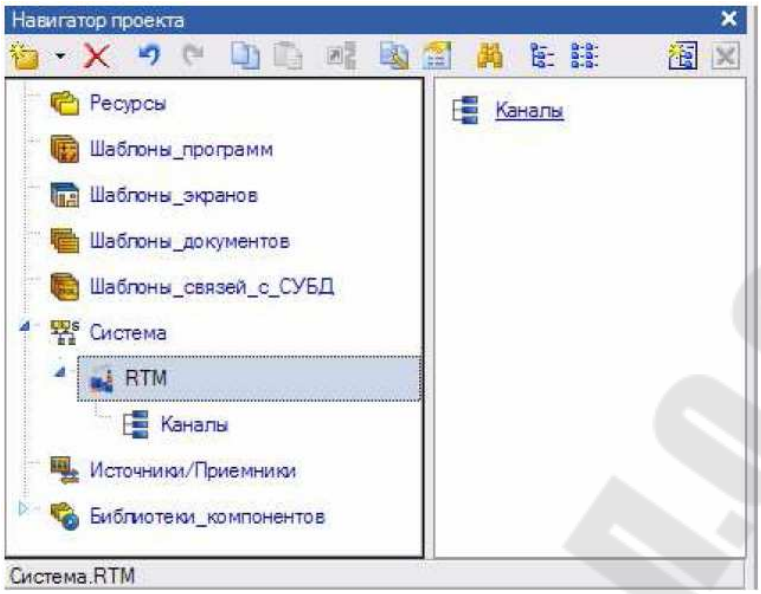


Рисунок 2.2 – Окно навигатора проекта с созданным узлом

2.3 Создание канала

В созданном узле выделить группу «**КАНАЛЫ**» и вызвать контекстное меню. В появившемся меню выбрать «**СОЗДАТЬ КОМПОНЕНТ**», а из предложенных выберите «**КАНАЛ_FLOAT**» (рисунок 2.3).

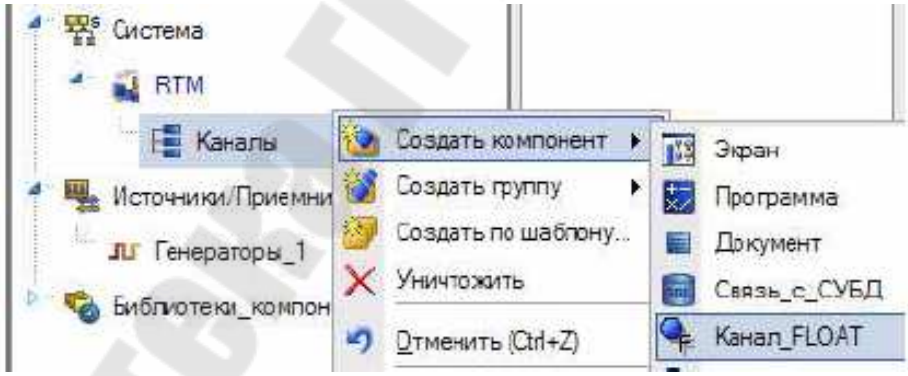


Рисунок 2.3 – Создание канала

2.4 Создание источника сигнала

Для добавления группу генераторов в проект, вызвать контекстное меню к строке «**ИСТОЧНИКИ/ПРИЕМНИКИ**» навигатора проекта. В появившемся меню выбрать строку «**СОЗДАТЬ ГРУППУ**». Среди предложенных групп выбрать – «**ГЕНЕРАТОРЫ**» (рисунок 2.4).

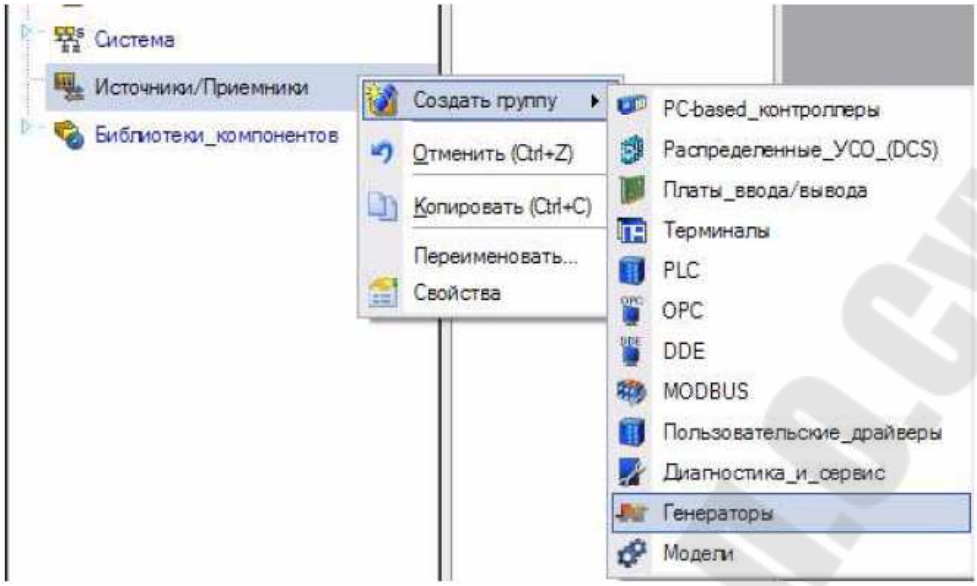


Рисунок 2.4 – Создание группы – Генераторы

Выделить созданную группу «**ГЕНЕРАТОРЫ**» и вызвать контекстное меню. Выбрать пункт «**СОЗДАТЬ КОМПОНЕНТ**». Среди предложенных генераторов выберите требуемый тип генератора (задается преподавателем) к примеру – «**Пила**» (рисунок 2.5).

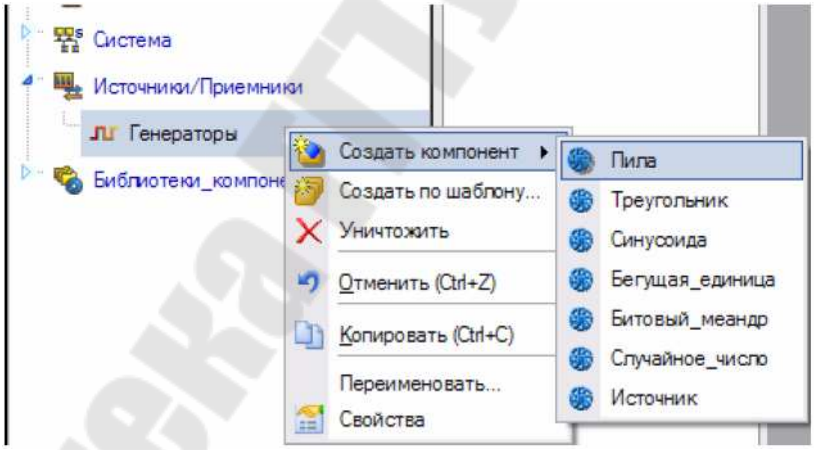


Рисунок 2.5 – Создание группы генератора пилы

2.5 Настройка масштабирования канала

Выполним расчет аргументов масштабирования канала – множителя **K** и смещения **Z** (см. разделы 1.2.1, 1.3). Так для входного значения канала изменяющегося в диапазоне **[0; 100]**, значение множителя *K* и смещения *Z*, которые позволят получить требуемый диапазон изменения выходного сигнала составит **K = 0.2, Z= -10**.

В поле «**ИМЯ**» введите новое имя канала - X. Установите флажок «**ИСПОЛЬЗОВАТЬ**» на панели «**ОБРАБОТКА**». Установите «**АПЕРТУРУ**» равной 0, **пик** равным 1, «**СГЛАЖ**.» равным 0, вычисленные значения множителя *K* и смещения *Z*. Убедитесь, что тип канала – «**INPUT**» (рисунок 2.6).

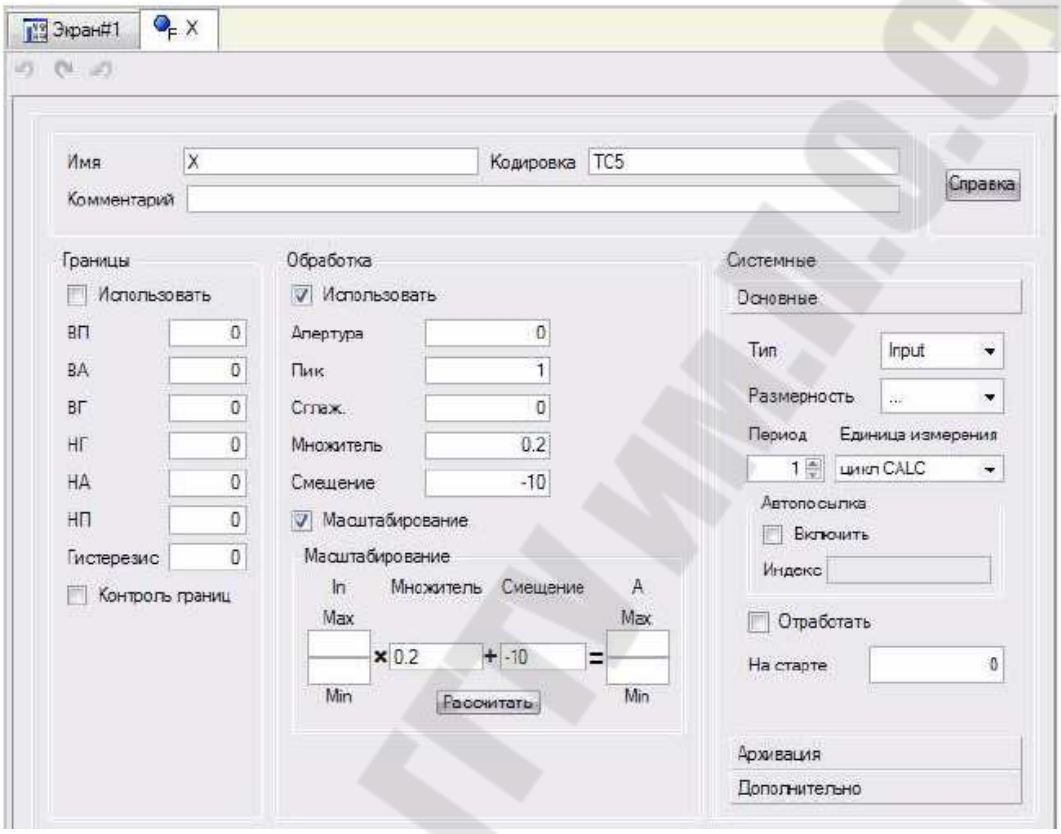


Рисунок 2.6 – Окно настроек атрибутов канала

Можно воспользоваться режимом автоматического расчета масштабных коэффициентов, задав границы **MAX**, **MIN** для входного и выходного сигналов и нажать «**РАСЧИТАТЬ**».

2.6 Привязка генератора к каналу

Создать дополнительное окно навигатора проекта, щелкнув ЛК мыши по иконке  . В верхнем навигатора проекта выберите группу «**КАНАЛЫ**» **RTM** узла, в нижнем группу «**ГЕНЕРАТОРЫ**» (рисунок 2.7)**.**

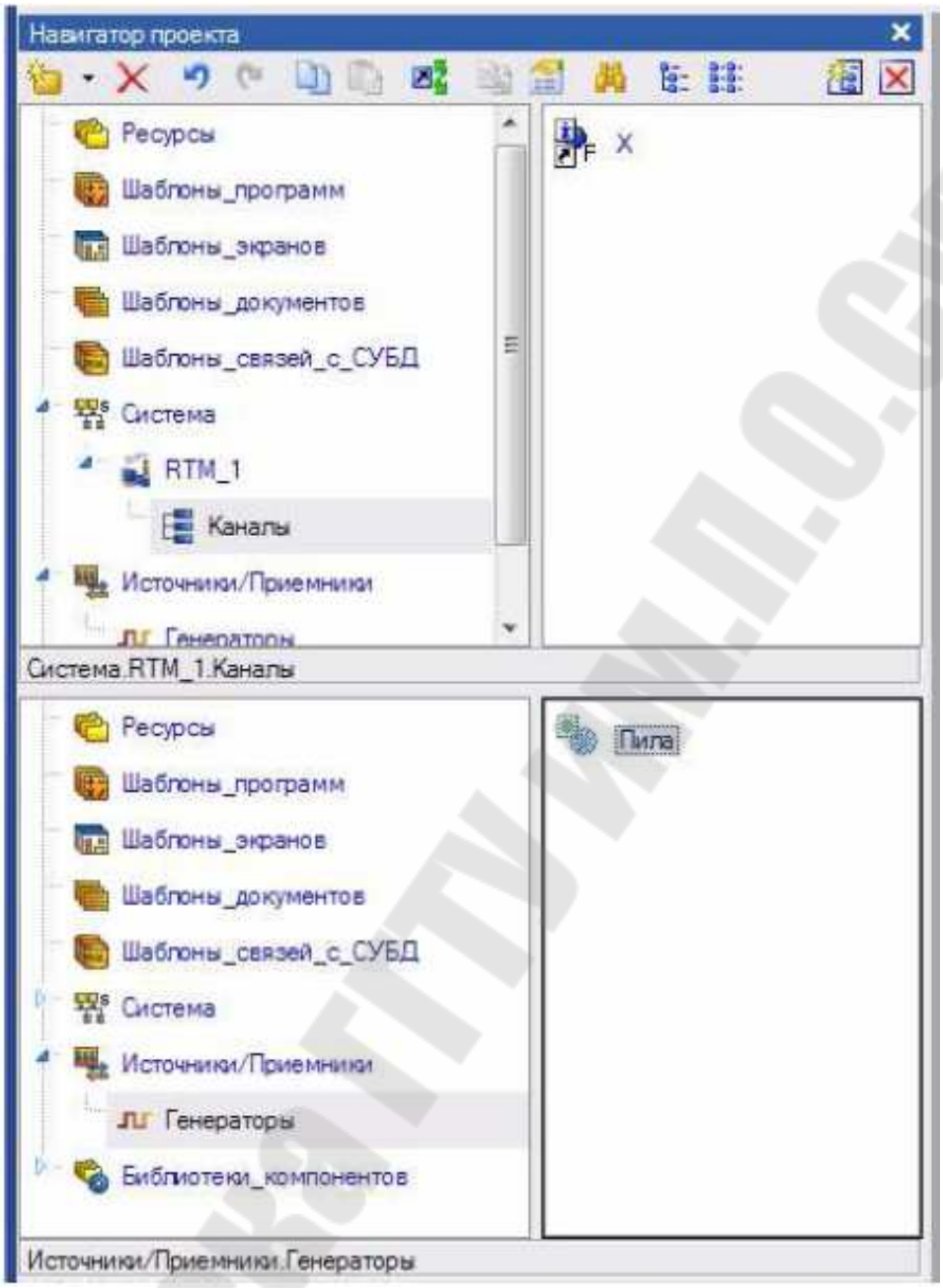


Рисунок 2.7 – Окно настройки привязки генератора к каналу

Для привязки генератора к каналу перетяните ЛК мыши генератор «**ПИЛА**» на канал X. Иконка канала должна на  .

2.7 Создание шаблона экрана

Для создания шаблона экрана, вызвать контекстное меню к группе «**КАНАЛЫ**» и выбрать «**СОЗДАТЬ КОМПОНЕНТ**» – «**ЭКРАН**» (рисунок 2.8).

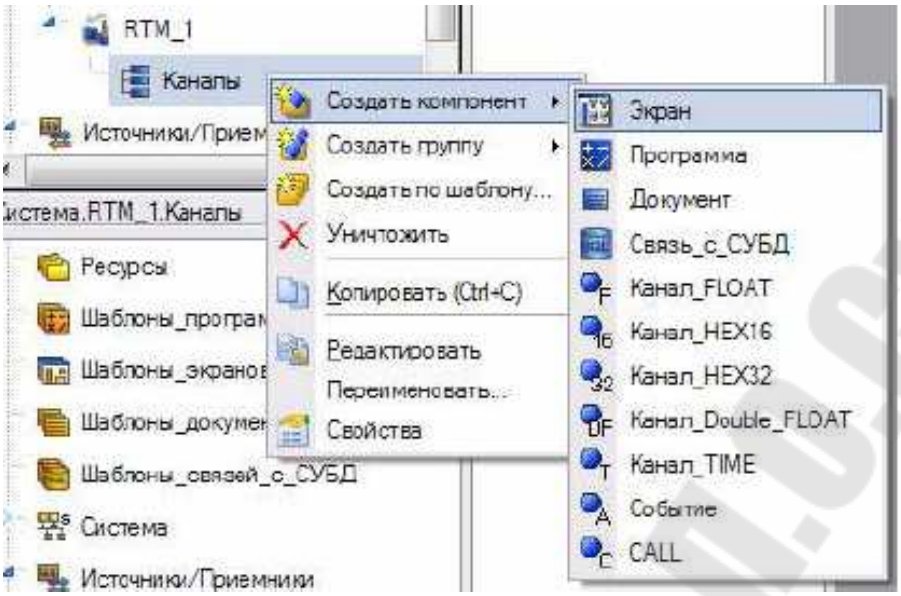


Рисунок 2.8 – Окно создания шаблона экрана

Для переходя в режим редактирования шаблона дважды щелкните ЛК мыши по созданному компоненту в навигаторе проекта.

2.8 Создание и настройка стрелочного индикатора

В панели инструментов ИС щелкнуть ПК мыши по иконке  и появившемся списке объектов, выбрать стрелочный прибор  . Рзместить на экране шаблон индикатора (рисунок 2.9).

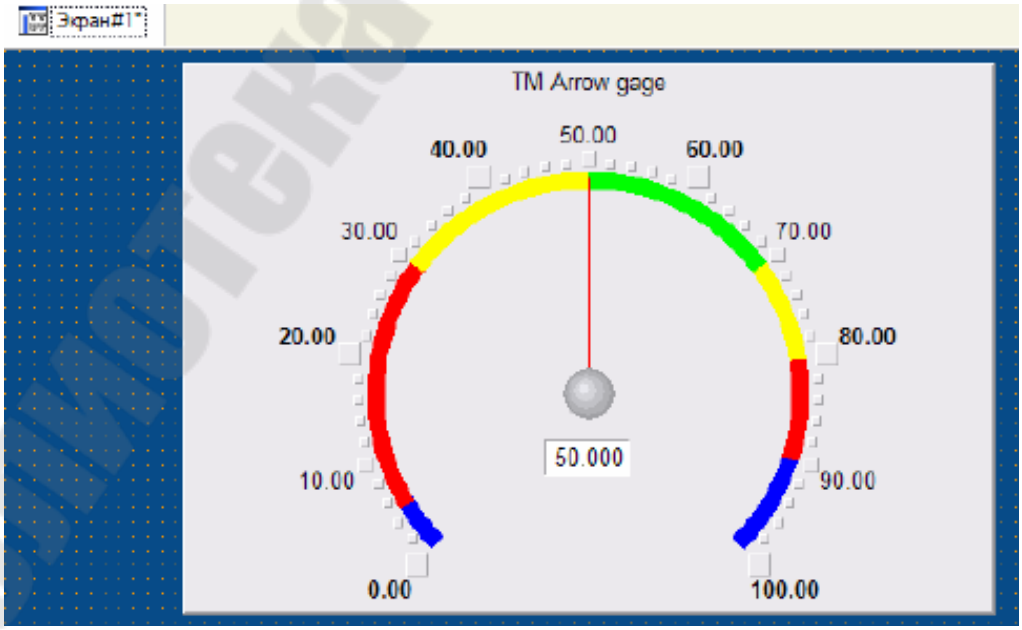


Рисунок 2.9 – Окно экрана с шаблоном индикатора

Для задания свойств объекта дважды щелкните левой клавишей мыши по созданному стрелочному индикатору (рисунок 2.10).

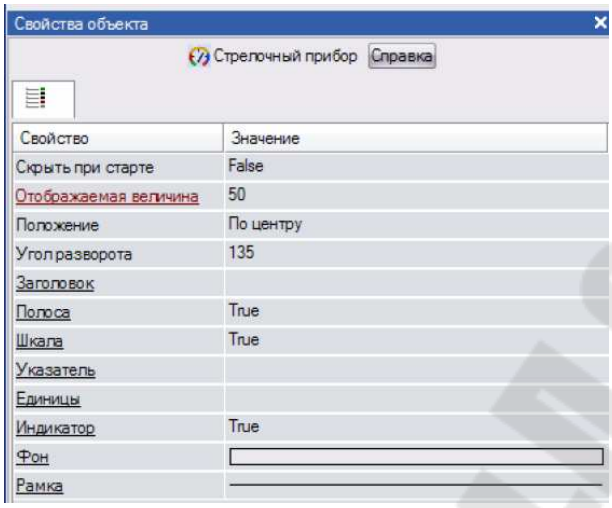


Рисунок 2.10 – Окно свойств индикатора

В разделе «**ЗАГОЛОВОК**» задайте подпись индикатора. Раскройте раздел «**ПОЛОСА**» и заполните поля **HL**, **HA**, **HW**, **LL**, **LA**, **LW** любыми значениями, удовлетворяющими условию: Нижний предел шкалы < LL < LA < LW < HW < HA < HL < Верхний предел шкалы (с учетом всей шкалы [-10, 10]). Пример заполнения полей свойств стрелочного индикатора приведен на рисунке 2.11.

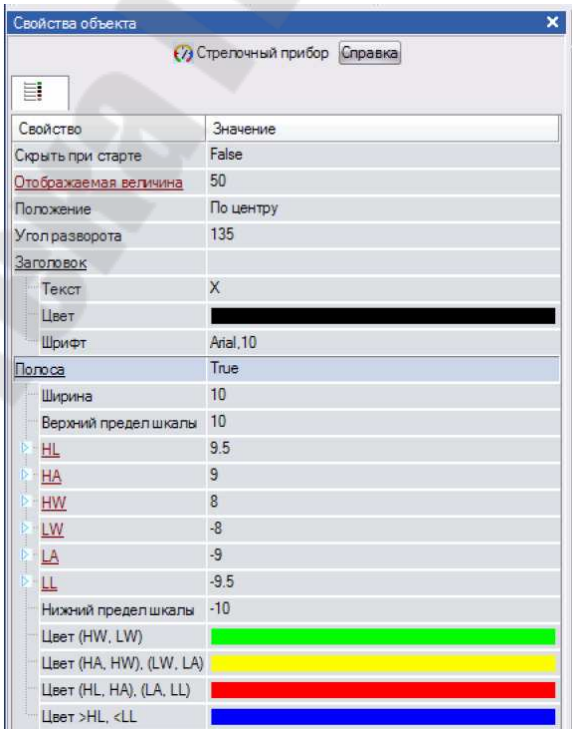


Рисунок 2.11 – Окно свойств индикатора

2.9 Привязка индикатора к каналу

Для привязки показаний индикатора к каналу необходимо:

1. Раскройте раздел «**ОТОБРАЖАЕМАЯ ВЕЛИЧИНА**» и щелкните ЛК мыши в поле «**ПРИВЯЗКА**», для открытия окна настройки привязки.
2. Для создания аргумента щелкните ЛК мыши на иконке  . Установите тип – **IN**, тип данных – **REAL**.
3. Для открытия окна настройки привязки, дважды щелкнуть ЛК мыши в столбце «**ПРИВЯЗКА**» таблицы. В левой части открытого окна выделить канал RTM узла - **X**, созданный ранее, в правой части аргумент «**Реальное значение**» (рисунок 2.12). Для создания привязки нажать кнопку «**Привязка**».
4. В поле ***имя*** задать имя канала аргумента – **X_R.**

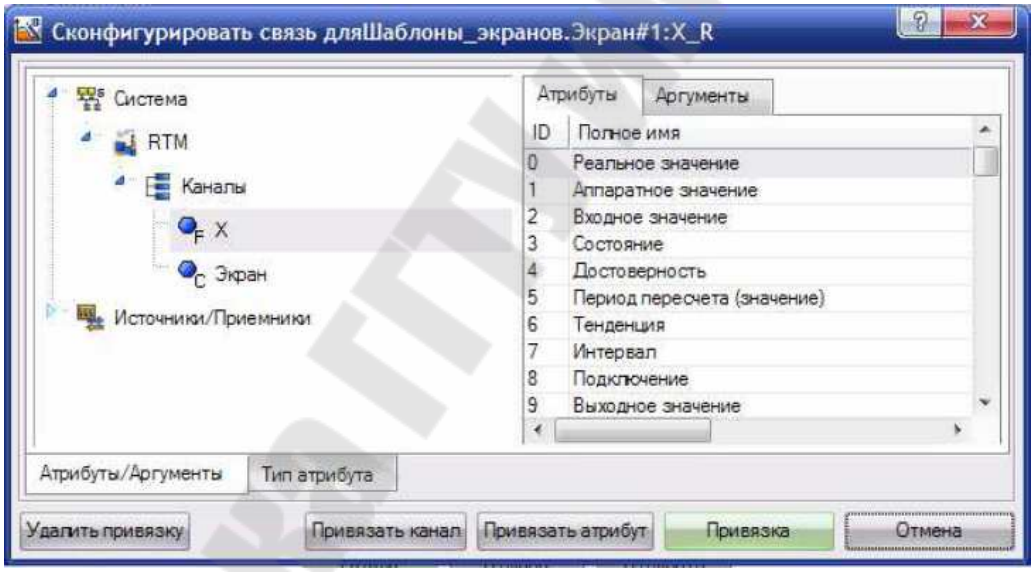


Рисунок 2.12 – Окно конфигурирования связи

Окончательно окно свойств привязки примет вид рисунок 2.13.

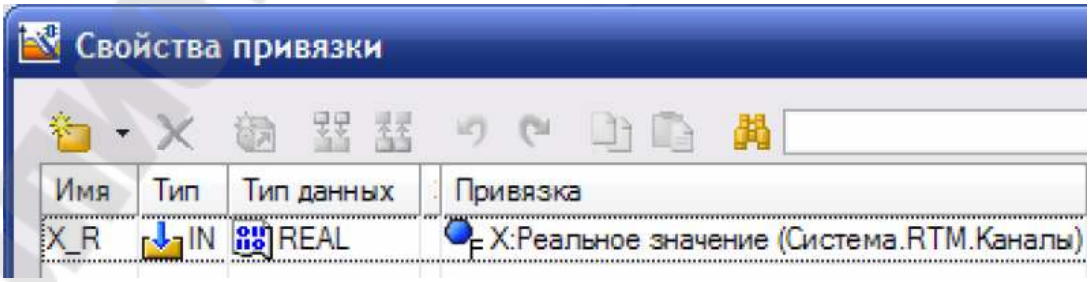


Рисунок 2.13 – Окно свойств привязки

2.10 Создание и настройка тренда

Для создания тренда щелкните ЛК мыши по иконке  . Если вместо нее располагается на панели инструментов иконка архивный тренд  , тренд XY  или архивная гистограмма  , то необходимо щелкнуть правой клавишей мыши на соответствующей иконке и выбрать тренд  . Разместите на экране объект (рисунок 2.14).

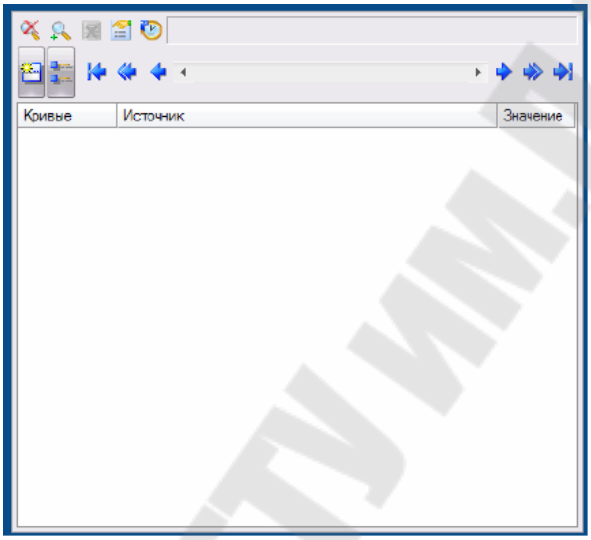


Рисунок 2.14 – Вид объекта – тренд

Откройте окно свойств тренда и настройте следующие параметры:

1. В закладке «**Основные свойства**»  , в поле **заголовок** задать подпись;
2. В закладке «**Кривые**»  ,раскрыть поле **кривые** и настроить следующие параметры:
   - **имя –** имя созданной кривой;
   - **Макс. значение** – верхнюю границу диапазона;
   - **Мин. Значение –** нижней границы диапазона;
   - **Привязка** – привязать к ранее созданному аргументу **X_R**.

При необходимости скорректируйте размеры окна тренда (рисунок 2.15).

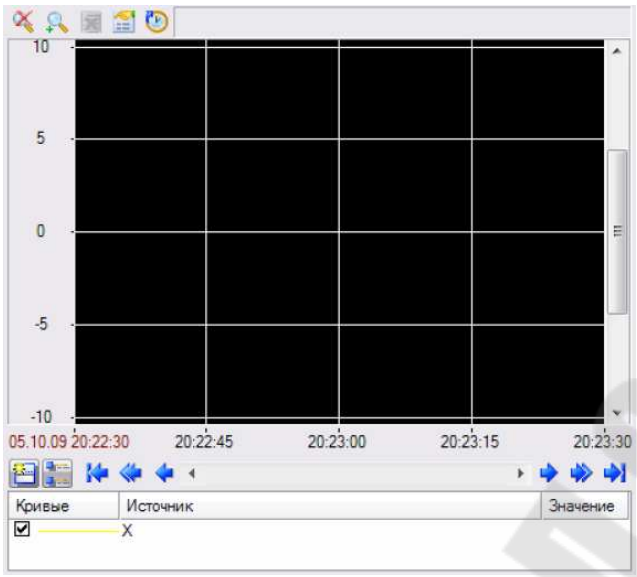


Рисунок 2.15 – Вид объекта – тренд

2.11 Запуск проекта.

Щелкните левой клавишей мыши по иконке  или по строчке ***сохранить для MPB*** в меню *файл.* В навигаторе проекта выделите созданный **RTM** узел и щелчком ЛК мыши по иконке  запустите профайлер. Для запуска проекта щелкнуть по  . Результат работы представлен на рисунке 2.16.

Рисунок 2.15 – Вид окна профайлера

Задание 2: Создать проект:

1. Создать генератор – **случайное число**;
2. Создать канал (FLOAT) и выполнить привязку генератора к каналу;
3. Настроить масштабирование канала так чтобы выходной сигнал изменялся в интервале [-10; 10];

4. Создать шаблон экрана и разместить на нем объект – тренд;
5. Привязать 3 линии тренда к следующим атрибутам канала;
   - **Входное значение** (2, **In**);
   - **Аппаратное значение** (1, **A**);
   - **Реальное значение** (0, **R**).
6. Настроить для канала экспоненциальное сглаживание. Сравнить полученные линии тренда.
7. Создать объект – стрелочный индикатор и связать его показания с входным значением канала;
8. Создать объект – линейный индикатор и связать его показания с реальным значением канала;

3. Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, созданные каналы и привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции.

4. Контрольные вопросы

1. Работа монитора TRACE MODE
2. Каналы данных
3. Общие атрибуты каналов
4. Канал FLOAT. Обработка данных
5. Канал FLOAT. Масштабирование
6. Канал FLOAT. Экспоненциальное сглаживание
7. Канал FLOAT. Фильтрация пиков или фильтрация малых изменений в канале INPUT
8. Канал FLOAT. Линейное сглаживание и фильтрация малых изменений в канале OUTPUT
9. Канал FLOAT. Клиппирование в канале OUTPUT
10. Атрибуты канала FLOAT
11. Границы и интервалы канала FLOAT
12. Генерация сообщений. Гистерезис
13. Канал класса DOUBLE FLOAT
14. Канал класса HEX16 и HEX32
15. Пересчет базы каналов монитором
16. Время цикла монитора
17. Период пересчета канала

Лабораторная работа №3
Разработка графического интерфейса оператора в SKADA TRACE MODE6

Цель работы: познакомиться со стандартными объектами, предназначенными для создания статических и динамических изображений.

1. Теоретические сведения

1.1 Создание и настройка объектов экрана

Для создания интерактивных информационных систем необходимы динамические объекты, такие как текст, стрелочный прибор, ползунок, кнопка, выключатель, тренд и так далее. Для размещения объектов на экране необходимо щелкнуть левой клавишей мыши по соответствующей иконке инструмента (таблица 1.1) панели инструментов. Для раскрытия группы объектов, следует щелкнуть правой клавишей мыши по иконке и в раскрывшемся списке инструментов выбрать необходимый. После выбора инструмента задают его расположение на шаблоне экрана.

Для перехода в режим редактирования необходимо щелкнуть ЛК мыши по иконке  и двойным щелчком ЛК по соответствующему объекту открыть окно его свойств. Настраивая свойства объекта можно задать внешний его вид и логику его работы.

Таблица 1.1 – Объекты Trace Mode

Группа объектов	Иконка объекта	Наименование объекта
Приборы		Стрелочный прибор
		Ползунок
Тренды		Тренд
		Архивный тренд
		Тренд X-Y
		Архивная гистограмма
Текст		Текст
Кнопки		Кнопка

		Группа кнопок
		Картинки-кнопки
Выключатели		Выключатель 0
		Выключатель 1
		Выключатель 2
		Выключатель 3
		Выключатель 4
		Выключатель 5
		Выключатель 6
		Выключатель 7

Для привязки объекта к необходимому значению (атрибуту) канала, необходимо щелкнуть ЛК мыши в поле **привязка (или другом, зависит от объекта)**, окна свойств выбранного объекта. В открывшимся окне (рисунок 1.1), необходимо создать аргумент и произвести его привязку к требуемому значению (атрибуту) канала. Если требуемый аргумент уже существует, то выбираем его из списка.

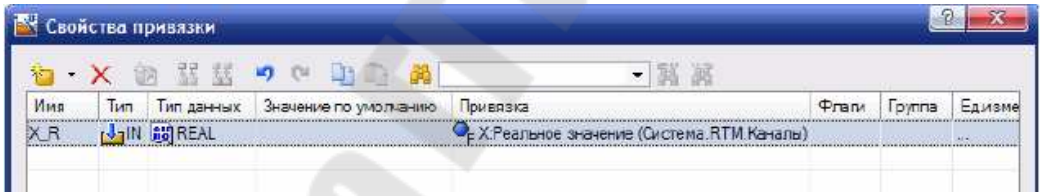


Рисунок 1.1 – Вид окна создания аргумента

Ряд объектов не содержат закладок. Другие объекты содержат несколько закладок. Так тренды содержат закладки: **основные свойства** и **кривые** (секторы, столбцы). На закладке основные свойства задается внешний вид, легенда, оси. На закладке **кривые** или (**секторы**, **столбцы**) создаются и настраиваются все **кривые** (**секторы**, **столбцы**), выводимые объектом. Для создания **кривой** (**сектор**, **столбец**) следует выделить строку **кривые** (**сектор**, **столбец**) и вызвать контекстное меню, в котором настроить параметры отображения и привязку.

Ряд объектов содержат закладки **динамический контур**  , **динамическая заливка**  , **динамическая трансформация**  . Данные закладки позволяют обеспечить динамичность изображения.

Объекты могут содержать закладку **действия**  , которая позволяет настроить действия, выполняемые при нажатии(**mousePressed**) или отпускании (**mouseReleased**) левой клавиши мыши.

Объект **выключатель** позволяет ввести и отобразить значение переменной, принимающие два значения: true (истина) и false (ложь). Для **выключателя** необходимо выбрать в поле **привязка** имя аргумента, привязанного к необходимому значению параметра.

Поля вид индикации, константа, код доступа задают логику работы. Для индикации **Arg & Конст** (**&** – побитовое логическое И) – выключатель переводится в положение «**вкл**», когда **аргумент & константа** = **true**, в противном случае переключатель переводится в положение «**выкл**». При щелчке ЛК мыши по выключателю аргументу присваивается результат вычисления по формуле **аргумент ^ Значение** (^ – побитовое исключающее ИЛИ).

При индикации **Arg >= Конст** выключатель находится в положении «**выл**», когда **аргумент** не меньше чем **константа**, если **аргумент** меньше чем **константа,** выключатель переводится в положение «**выкл**». При индикации **Arg == Константа** выключатель находится в положении «**вкл**», если **аргумент** равен значению **константа**, в противном случае выключатель переводится в положение «**выкл**». При видах индикации **Arg >= Конст**, **Arg == Константа** и при щелчке левой клавишей мыши по выключателю, **аргументу** присваивается значение, заданное атрибутом **значение**.

1.2 Статическое изображение

Для создания статического изображения можно использовать инструменты: линия  , ломанные и кривые (таблица 1.2), прямоугольники (таблица 1.3), плоские фигуры (таблица 1.4), объемные фигуры (таблица 1.5).

Создание ломанных и кривых заключается в фиксации точек излома или перегиба щелчком левой клавиши мыши, когда курсор расположен в точке, где должен быть излом или перегиб. Положение последней точки изгиба или излома следует фиксировать щелчком правой клавиши мыши.

Таблица 1.2 – Ломанные и кривые

Иконка	Название объекта
	Ломаная линия
	Многоугольник
	Ломаная с заливкой
	Разомкнутая кривая
	Замкнутая кривая

Таблица 1.3 – Прямоугольники

Иконка	Название объекта
	Контур
	Прямоугольник
	Панель
	Рамка

Таблица 1.4 – Плоские фигуры

Иконка	Название объекта
	Плоский клапан
	Треугольник
	Стрелка
	Овал
	Эллипс сектор

У плоских фигур, прямоугольников, ломаных и кривых можно настроить статический контур и заливку на закладке **основные свойства** свойств объетов. Для статического контура можно задать его стиль, цвет, толщину контура. Для заливки можно задать тип заливки (цвет или изображение). Если выбрано заполнение цветом, то можно настроить цвет, стиль заполнения.

Инструмент объемные фигуры позволяет создать различные объекты, указанные в таблице 1.5. У объемных фигур можно задать материал, степень прозрачности (прозрачность), текстуру, качество изображения, определяющее степень прорисовки текстуры, толщину стенок. Выбор края объемной фигуры позволяет создавать изображения с различными краями: закругленные, скошенные и т.д.

Таблица 1.5 – Объемные фигуры

Иконка	Название объекта
	Цилиндр
	Сфера
	Конус
	Тор
	Пирамида
	Емкость
	Клапан
	Насос
	Труба
	Рельефный конус
	Криволинейный конус
	Градиент

Рассмотрим настройку свойств объекта на примере задания толщины фигуры. Если **толщина стенок** равна 0, то на экране изображается, как выглядит объект внешне. Когда задана **толщина стенок** больше 0 изображается разрез объекта (рисунок 1.2).

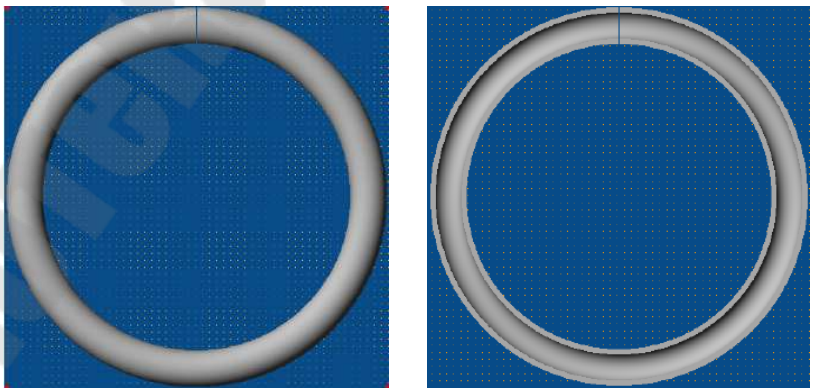


Рисунок 1.2 – Настройка толщины стенки фигуры

Для настройки вида материала следует дважды щелкнуть ЛК мыши по строчке **материал**, и настроить пункт **выбрать из списка.** Если установить его равным **False**, то материал задается как цвет ана-

логично другим объектам, если **True**, то можно выбрать материала, из которого выполнено изделие (хром, бронза, медь, текстуры шлифовка или гравировка).

В качестве текстуры объекта можно выбрать изображение. Для этого необходимо импортировать изображения:

1. создать группу **картинки** в разделе **ресурсы** навигатора проекта (рисунок 1.3);
2. создать **библиотеку изображений** в группе **картинки** (рисунок 1.4);
3. открыть **библиотеку изображений** и импортировать необходимые изображения из папки «**%TRACE MODE%\\Lib\Texture»** (рисунок 1.5);

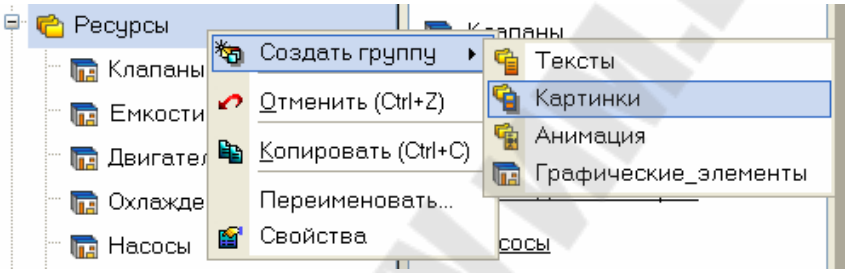


Рисунок 1.3 – Создание группы изображений

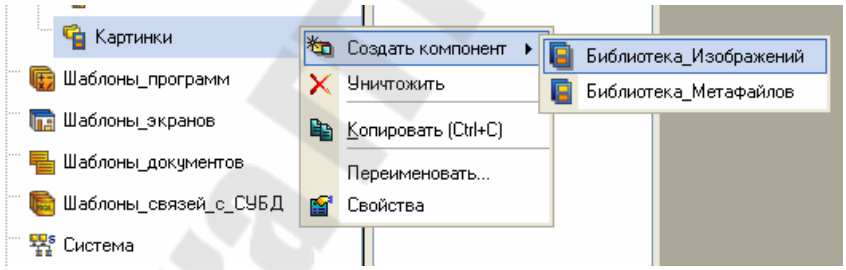


Рисунок 1.4 – Создание библиотеки изображений

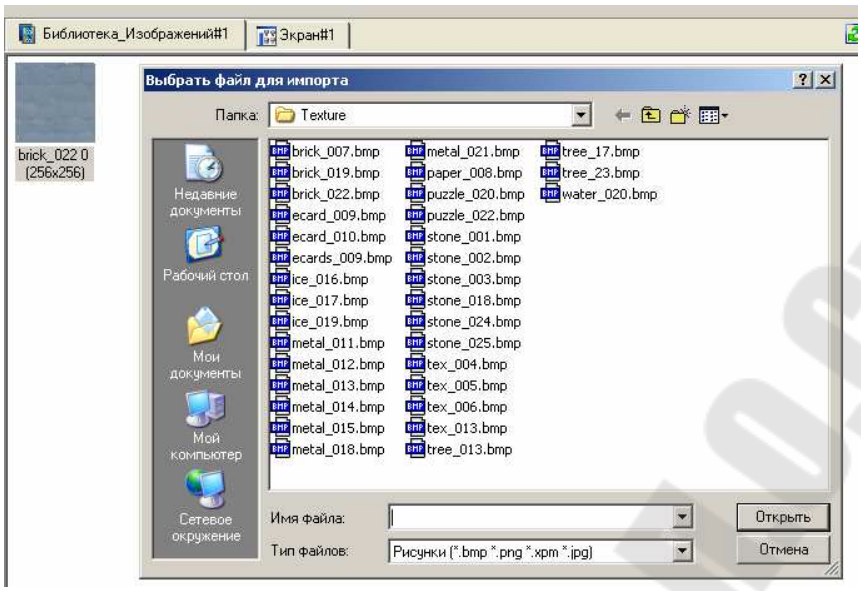


Рисунок 1.5 – Импорт изображений

После импортирования изображений необходимо выбрать изображение в поле **текстура**.

У объемного прямоугольника, емкости, трубы можно менять вид верхнего или нижнего края, изменяя его внешний вид. У конуса можно задавать соотношение оснований в процентах, у тора – его толщину, у клапана – форму, а также форму и цвет привода. Насос позволяет выбрать форму, в виде которой он будет изображен.

Используя указанные объекты, располагая их определенным образом, можно создать статическое изображение.

1.3 Динамическое изображение

Динамическое изображение определяется контролируемым процессом. Изображение может перемешаться, изменять размеры, поворачиваться, может происходить перемещение пунктиров по его контору, объект может быть заполнен до определенного уровня.

Для получения динамического изображения всегда необходима привязка к аргументу, значение которого отображается тем или иным способом.

При динамическом контуре (закладка динамический контур) задаются (рисунок 1.6) два цвета: цвет штрихов и промежутка между ними, длина штриха, которая также определяет шаг перемещения штрихов. При динамизации, происходит перемещение штрихов по контору. Скорость перемещения определяется привязанным аргументом. Если аргумент равен 0. то перемещение отсутствует. Когда аргумент равен 1 происходит перемещение штрихов при каждом такте на

один шаг. Если аргумент равен двум, к примеру, то штрихи перемещаются на один шаг один раз за 2 такта.

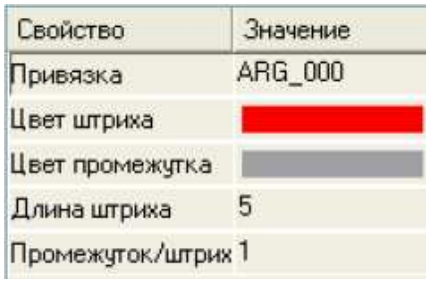


Рисунок 1.6 – Настройка динамического контура

Рассмотрим динамическую трансформацию (закладка динамическая трансформация). Можно выделить следующие виды – динамическое перемещение, масштабирование, вращение. При динамическом перемещении задается ломаная линия, вдоль которой происходит перемещение. Для ряда точек (узлов) задаются соответствующие им значения (рисунок 1.7). Текущее положение объекта зависит от значения привязанного аргумента и значений, соответствующих узлам, флага – перемещать плавно.

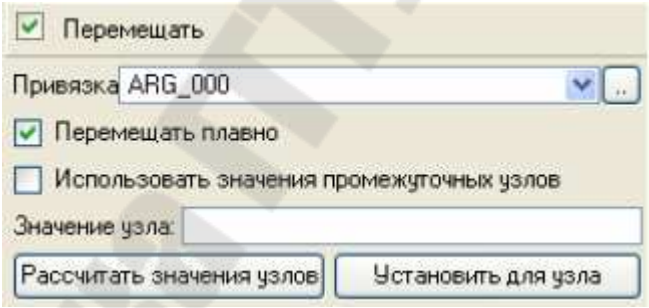


Рисунок 1.7 – Настройка динамического перемещения

При динамизации масштабирования задаются начальный и конечный размер объекта и значения привязанного аргумента, соответствующие заданным значениям (рисунок 1.8) и центр относительно которого будет происходить масштабирование. Текущий размер зависит от значения аргумента.

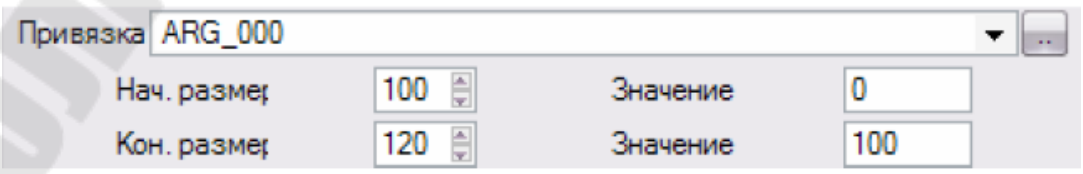


Рисунок 1.8 – Настройка динамического масштабирования

При динамическом вращении задается начальный и конечный угол, значения аргумента, соответствующие начальному и конечному углу, центр относительно которого происходит вращение. Угловое положение в данном случае зависит от значения аргумента. Настройка вращения происходит аналогично настройке перемещения.

Динамическая заливка заключается в том, что задается максимальное и минимальное значение привязываемого аргумента (рисунок 1.9). Происходит заливка объекта до уровня, который определяется привязанным аргументом. Заливка может однослойной (отображается значение одного аргумента) и многослойной (отображение значений нескольких аргументов).

Можно настроить изменение цвета динамической заливки в зависимости от состояния технологического процесса (предупреждение, авария, вне границ). Для этого необходимо выбрать цвета заполнения и выбрать значение **true** в поле **цвета для диапазонов**.

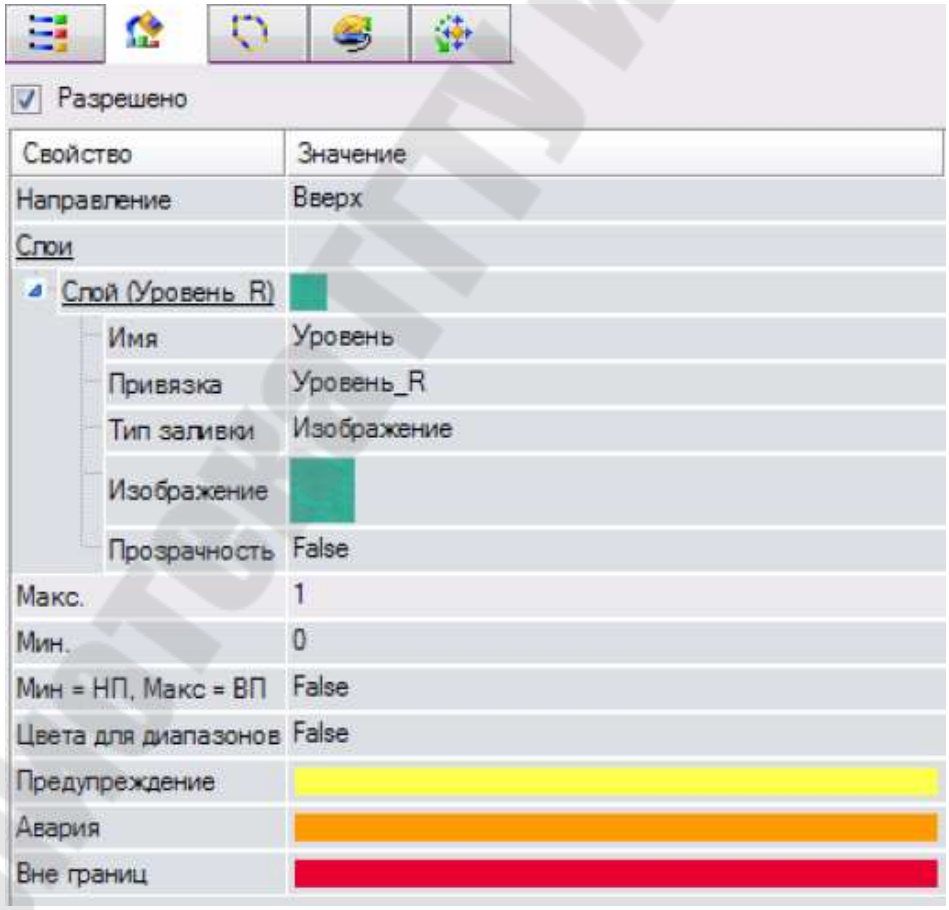


Рисунок 1.9 – Настройка динамической заливки

Для настройки зависимости цвета заливки объекта (объект без динамической заливки) от состояния процесса на закладке ***основные свойства*** в разделе ***заливка*** следует раскрыть раздел ***цвет заливки***, где можно выбрать вид индикации, выбрать цвета заливки (рисунок 1.10), задать диапазон.

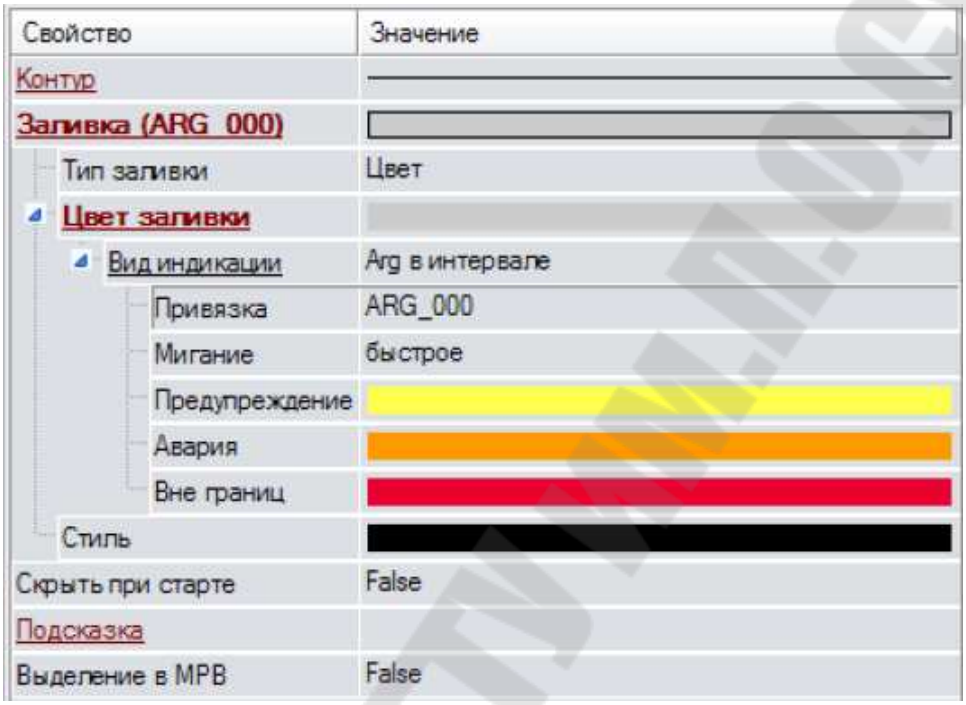


Рисунок 1.10 – Настройка динамического изменения цвета

2. Ход работы

Задание 1

1. создать новый проект. В данном проекте создать генератор, привязанный к каналу, значение которого определяется уровнем продукта в емкости;
2. импортировать изображения текстур;
3. создать экран, расположить на нем тренд, строящий зависимость уровня продукта во времени.
4. создать статическое изображение емкости в разрезе, насоса, трех труб, по одной трубе продукт поступает в емкость, по другой вытекает из нее. Вторая труба соединена с третьей через насос.
5. создать динамический объект, имитирующий заполнение емкости, используя графический файл.

2.1 Создание объектов

Запустить программу **TRACE MODE IDE 6** иконка  , создать и сохранить новый проект. В навигаторе проекта выделить строку «**СИСТЕМА»** и в контекстном меню выбрать «**СОЗДАТЬ УЗЕЛ»**. Среди предложенных типов узлов выбрать – **RTM**.

Создать следующие объекты проекта:

1. Канал FLOAT – «**УРОВЕНЬ**», реальное значение которого пропорционально уровню продукта в емкости. Настроить масштабирование в канале обеспечив выходной диапазон [0, 1];
2. Генератор – **синусоидального сигнала**;
3. Выполнить привязку созданного генератора к каналу;
4. Экран для размещения информационных объектов (если он не был создан автоматически при создании проекта);
5. Тренд. Настроить параметры графика и выполнить привязку к созданному каналу (имя аргумента **УРОВЕНЬ_R**).

Результат созданного при выполнении пунктов 1 – 5 экрана, представлен на рисунке 2.1.

2.2 Создание статического изображения

2.2.1 Создание рамки

Для создания рамки необходимо щелкнуть ЛК мыши по иконке . Если указанной иконки на панели инструментов нет, то ПК мыши щелкаем по одной из следующих иконок: , , или и среди предложенных объектов выберите рамку . Размещаем рамку на экране под трендом (рисунок 2.2). Для перехода в режим редактирования кликнуть ЛК мыши по иконке .

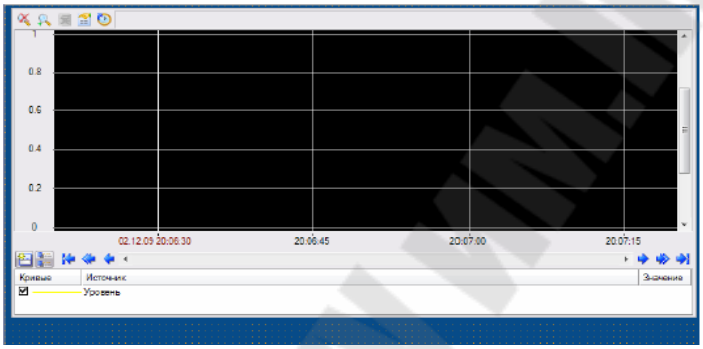


Рисунок 2.1 – Вид экрана информационной системы

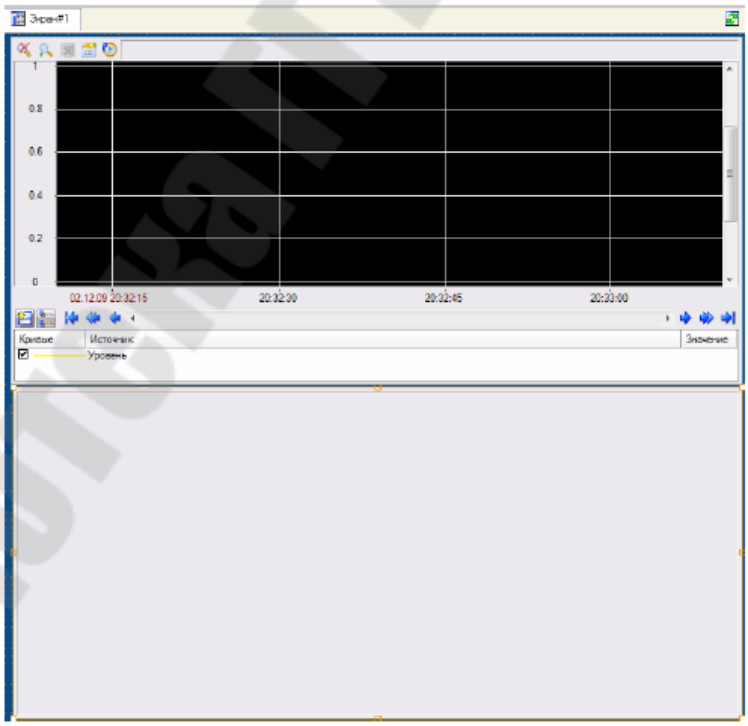
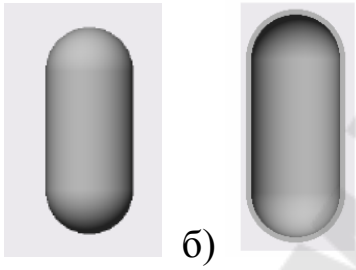


Рисунок 2.2 – Вид экрана информационной системы

2.2.2 Создание емкости

Для выбора объемной фигуры – емкость, щелкнуть ЛК мыши по иконке . Если данной иконки нет на панели инструментов, то щелкните ПК мыши по одной из иконок: , , , , , , , , , , . Среди предложенных объектов выберите емкость и разместить объект на экране, задав противоположные углы (рисунок 2.3а). Перейти в режим редактирования – ЛК мыши по иконке .



а) б)

Рисунок 2.3 – Вид объекта «Емкость»
(а) – до редактирования свойств, (б) –

Открыть окно свойств объекта (выделить созданную емкость или двойное нажатие ЛК мыши по ней). В поле **толщина стенок** задать толщину больше 0 (подобрать значение для лучшего визуального восприятия)(рисунок 2.3б).

В поле **верхний** и **нижний край** выбрать необходимый вид края. Например, верхний край задать – , а нижний – . Для задания **материала емкости,** раскрыть вкладку **материал** (дважды щелкнуть ЛК мыши по подчеркнутой строчке **материал**). В поле **выбрать из списка** выбрать значение **true**. В поле **материал** выбрать необходимый материал, например – **хром**. В поле **стандартная текстура** выбрать необходимую текстуру – **гравировку**.

При необходимости можно добавить другие объемные фигуры. Для размещения конуса щелкнуть ПК мыши по иконке и выбрать . Разместить на экране конус над объектом емкость (рисунок 2.4). Вызвать свойства объекта и в поле **толщина стенок** задать ту же толщину, что и у емкости. В разделе **материал** настроить следующие поля:

– **выбрать из списка – true**;
– **материал – олово**;
– **стандартная гравировка – шлифовка**.

Рисунок 2.4 – Вид составного объекта «Емкость», «Конус»

2.2.3 Создание насоса

Для размещения конуса щелкнуть ПК мыши по иконке и выбрать фигуру . Разместить на экране насос справа, под объектом емкость (рисунок 2.5). Вызвать свойства объекта и разделе **материал,** настроить следующие поля:

- **выбрать из списка – true**;
- **материал – пластик черный**;
- **форма насоса (подобрать самостоятельно)**.

Рисунок 2.5 – Вид объекта «Насос»

2.2.4 Создание труб

Для размещения конуса щелкнуть ПК мыши по иконке и выбрать фигуру . Создадим трубу по которой продукт поступает в емкость и по которой из емкости течет в насос. Для этого щелчком ЛК мыши отметим местоположение начала трубы. Переместить курсор мыши в точку изгиба трубы и снова нажать ЛК мыши. Таким обра-

зом, отметить все точки изгиба трубы. Для указания конца трубы, щелкнуть ПК мыши, тем самым завершим создание текущей трубы. Аналогично создадим трубу, по которой продукт поступает в насос и вытекает из него (рисунок 2.6). Вызвать свойства объекта – «**Труба**» и настроить следующие поля:

– **толщина** – подобрать подходящую толщину каждой трубы;
– **базовый цвет**.

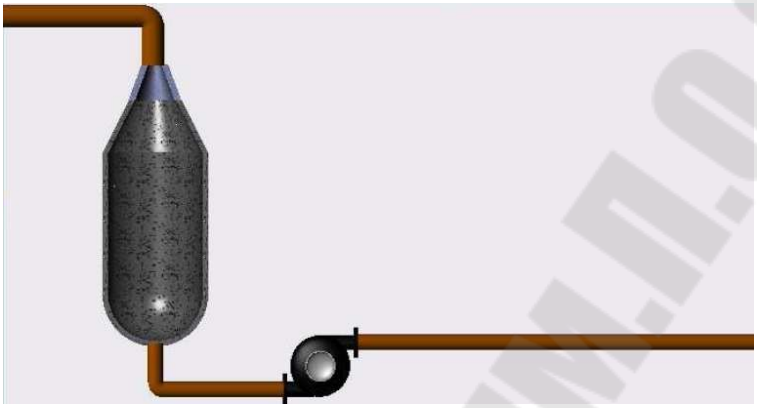


Рисунок 2.6 – Вид окна с размещенными статическими объектами

2.3 Создание динамического изображения

2.3.1 Импорт изображения

Выделить строку «**Ресурсы**» в навигаторе проекта. В контекстном меню выбрать строку «**Создать группу**» – «**Картинки**» (рисунок 2.7).

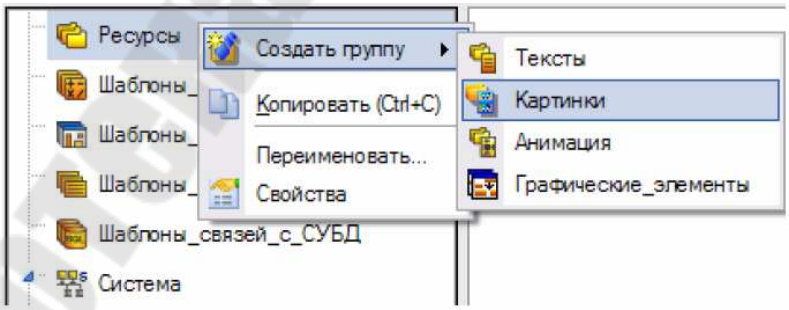


Рисунок 2.7 – Создание группы – «**Картинки**»

В контекстном меню к созданной группе «**Картинки**», вызвать контекстное меню и выбрать – «**Создать компонент**» – «**Библиотека изображений**» (рисунок 2.8).

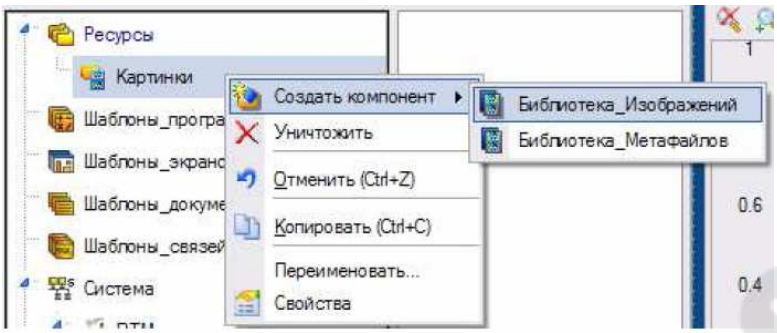


Рисунок 2.8 – Создание библиотеки изображений

Для импортирования изображения дважды щелкнуть ЛК мыши по созданной **библиотеке изображений**. В открывшимся окне, вызвать контекстное меню и выбрать – «**Импортировать**» . В появившемся диалоговом окне, открыть графический файл изображения («**%TRACE MODE%\Lib\Texture**» в папке где установлен пакет TRACE MODE), которое будет использоваться в дальнейшем.

2.3.2 Создание динамической заливки

Для создания динамической заливки создадим объект **многоугольник** и разместим его внутри объекта **емкость**. Для этого кликнем ЛК мыши по иконке . Если на инструментальной панели нет иконки , щёлкнем ПК мыши по одной из иконок группы объектов: , , , . Для размещения многоугольника, ЛК мыши зададим углы многоугольника. Последний угол определяется щелчком ПК мыши (рисунок 2.9). Также для создания заливки можно воспользоваться объектом .

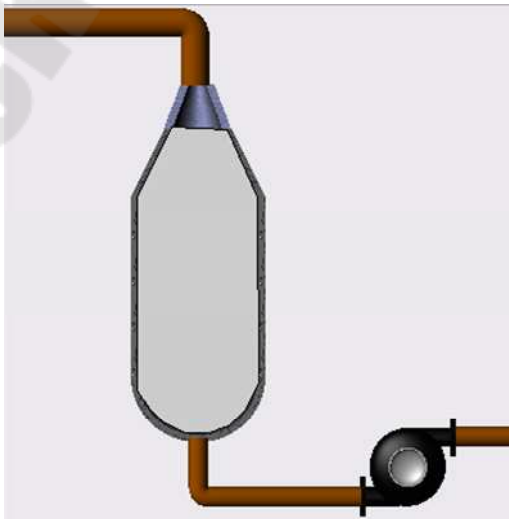


Рисунок 2.9 – Создание многоугольники для заливки

Вызвать окно настройки свойств созданного объекта и в разделе – **заливка** в поле **стиль** выбрать – **без заливки**. В результате изображение примет вид, приведенный на рисунке 2.6.

Для настройки параметров заливки, перейдем на закладку – **динамическая заливка** и поставим флажок – **разрешено**. Двойным нажатием ЛК мыши, раскрыть раздел – «**Слой**» и настроить следующие поля (рисунок 2.10):

- **имя** – имя слоя «**Уровень**».
- **привязка** – настроить привязку слоя к аргументу экрана **УРОВЕНЬ_R**, хранящему значение уровня продукта в емкости (создан при привязке тренда);
- **тип заливки** – в поле **изображение**, выбрать картинку, из библиотеки изображений, созданной ранее (рисунок 2.11);
- **Макс.** – значение верхней границы диапазона значений, хранимых в канале (**1**);
- **Мин.** – значение нижней границы (**0**);

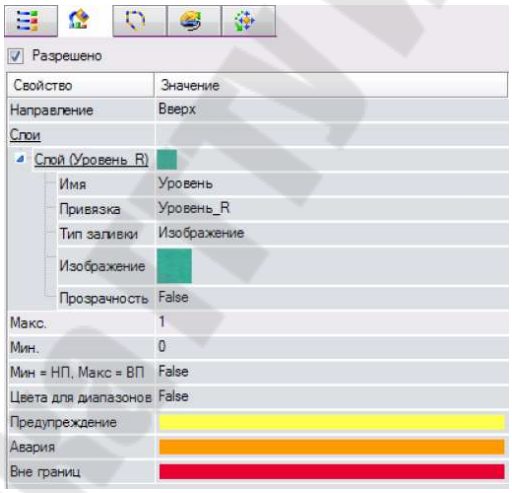


Рисунок 2.10 – Окно настройки параметров заливки

Рисунок 2.11 – Выбор изображения для заливки

2.4 Запуск проекта

Щелкните левой клавишей мыши по иконке  или по строчке **сохранить для MPB** в меню **файл**. В навигаторе проекта выделите созданный **RTM** узел и щелчком ЛК мыши по иконке  запустите профайлер. Для запуска проекта щелкнуть по  . Результат работы представлен на рисунке 2.12.

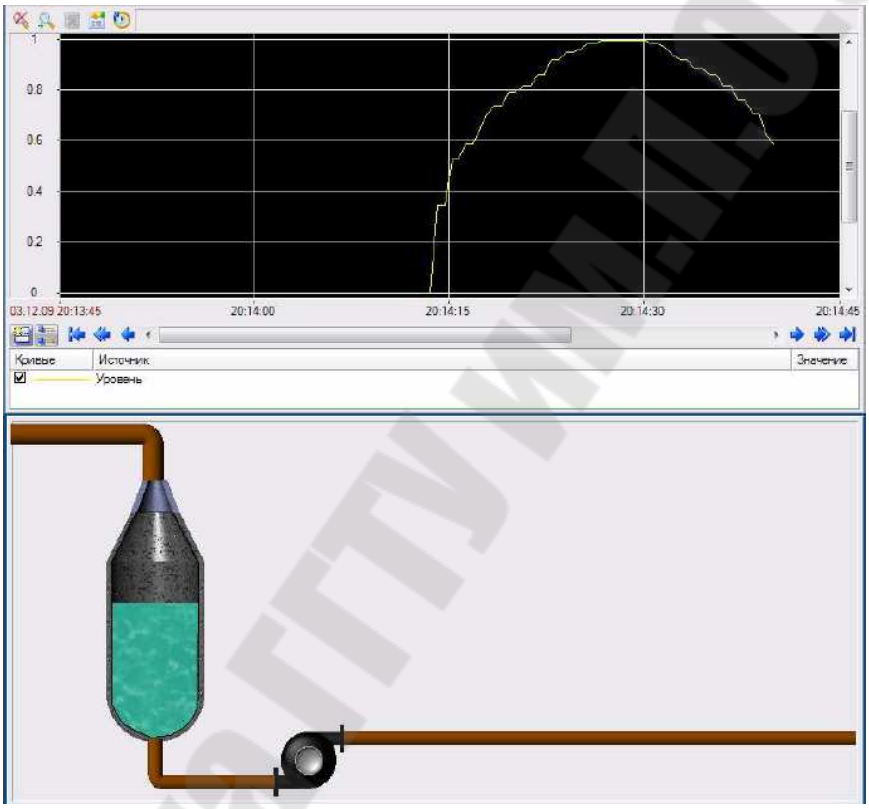


Рисунок 2.11 – Окно профайлера

3. Создание технологических объектов

3.1 Создание проекта

Запустить программу **TRACE MODE IDE 6** иконка  , создать и сохранить новый проект. В навигаторе проекта выделить строку «**СИСТЕМА**» и в контекстном меню выбрать «**СОЗДАТЬ УЗЕЛ**». Среди предложенных типов узлов выбрать – **RTM**.

Для создания технологических объектов воспользуемся встроенной библиотекой компонентов пакета. Для этого скопируем файл **tmdevenv.tmul** из поддиректории **%TRACE MODE%\Lib** в директорию **%TRACE MODE%**. Перейти в слой **Библиотеки_компонентов**, где в разделе **Пользовательская** открыть библиотеку **Библиотека_1**. Сохраненный в данной библиотеке объект **Объект_1** содержит в своем слое **Ресурсы** необходимый для дальнейшей разработки набор графических объектов – изображения клапанов, емкостей, двигателей и т.д (рисунок 3.1).

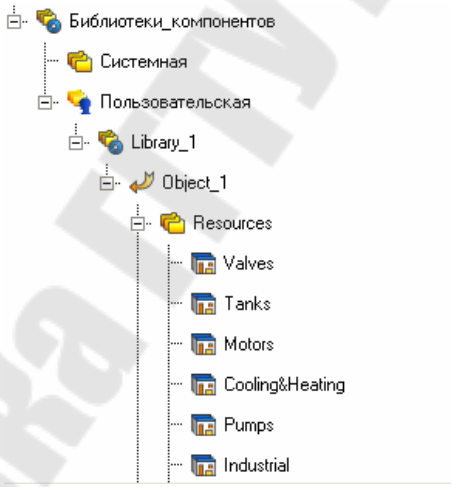


Рисунок 3.1 – Подключение библиотеки пользователя

Перенесем группы: **Клапаны** (Valves), **Насосы** (Pump), **Емкости** (Tanks), в слой **Ресурсы** нашего проекта с помощью механизма **drag-and-drop** и переименуем их.

3.2 Создание экрана оператора

Перейдя в слой **Шаблоны_экранов**, создадим в нем компонент **Экран#1** (рисунок 3.2). На созданном экране будут отображаться технологические параметры.

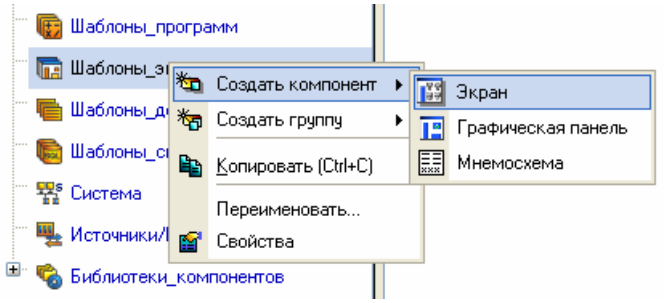


Рисунок 3.2 – Создание шаблона экрана

Дважды щелкнув ЛК мыши по имени шаблона, перейдем в режим создания и редактирования содержимого экрана. В свойствах экрана зададим основной цвет фона окна оператора. С помощью графических объектов (ГО), сохраненных в ресурсных библиотеках и вызываемых с помощью иконки панели инструментов, а также графических элементов (ГЭ) объемных труб и текста ABC, создадим статическую часть экрана (рисунок 3.3).

Рисунок 3.3 – Окно панели оператора

При настройке свойств надписей, откажемся от использования рамки и заливки (рисунок 3.4).

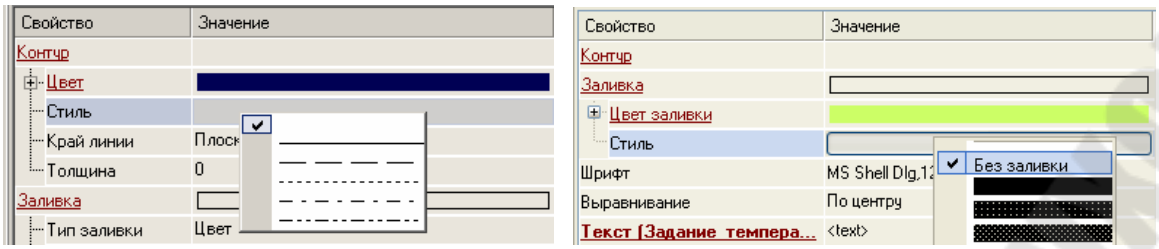


Рисунок 3.4 – Настройка свойств надписей

3.3 Создание аргументов экрана, каналов

Зададим аргументы шаблону экрана – щелчок ПК на созданном шаблоне экрана и выбор из выпадающего списка пункта **Свойства**, далее переход во вкладку **Аргументы**. С помощью иконки создадим необходимые аргументы, укажем имена, тип, тип данных, значения по умолчанию, и т.д (рисунок 3.5).

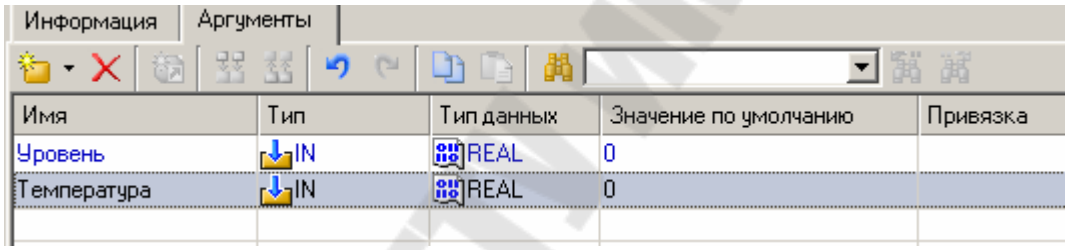


Рисунок 3.5 – Создание аргументов шаблона экрана

Определим с использованием ГЭ вывод значений параметров хранения (рисунок 3.6).

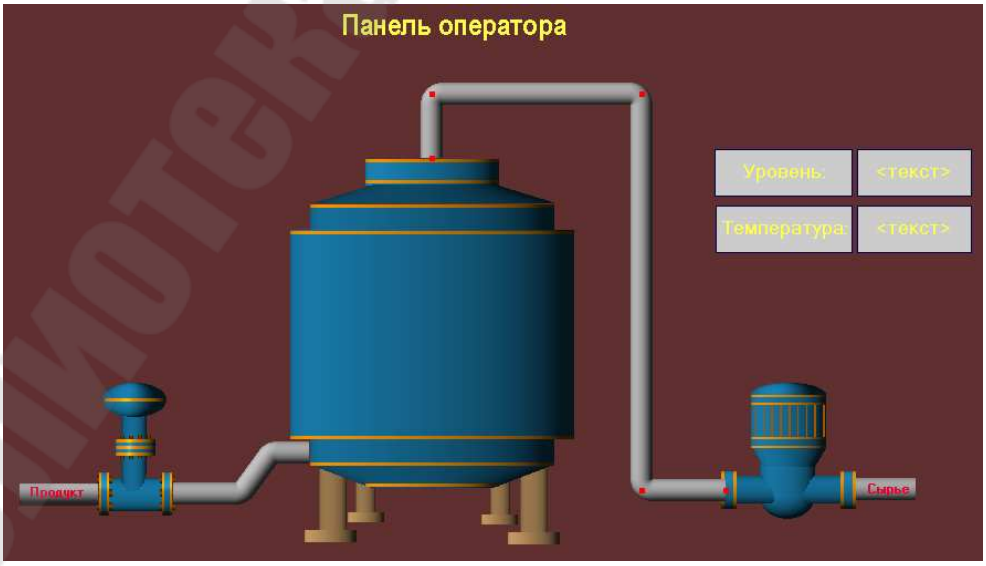


Рисунок 3.6 – Окно панели оператора

Выполним привязку ГЭ к аргументам шаблона экрана, установим формат вывода значений один знак после запятой (рисунок 3.7).

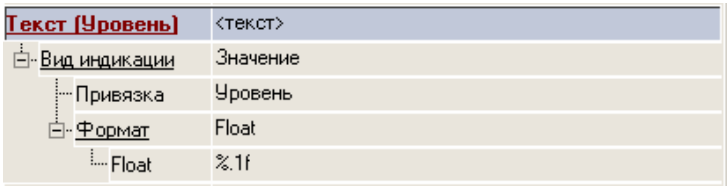


Рисунок 3.7 – Настройка свойств ГЭ

Для визуализации уровня в емкости воспользоваться объектом . Разместим объект внутри бака и настроим параметры заливки (рисунок 3.8).

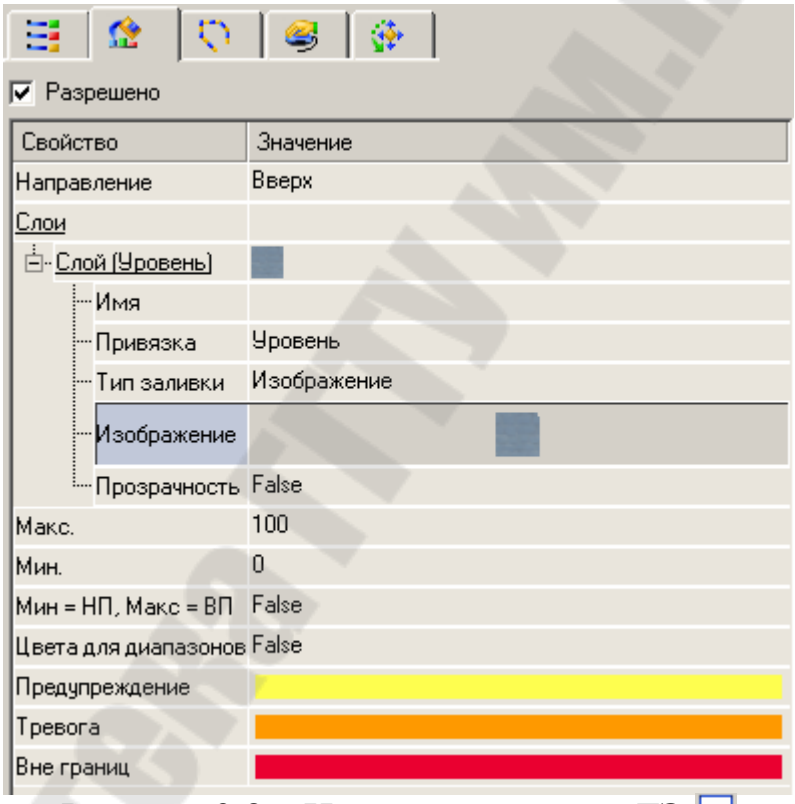


Рисунок 3.8 – Настройка свойств ГЭ

Выделим ЛК шаблон экрана и удерживая ЛК, перетащим его в группу узла **RTM_1**.

Следующим шагом создадим каналы по аргументам разработанных шаблонов экранов. Для этого войдем в группу каналов узла **RTM_1** и вызовем свойства канала класса **CALL Экран#1:2**. Перейдем во вкладку **Аргументы**, ЛК выделим первый аргумент и с помощью щелчка ЛК мыши на иконке создадим каналы в выбранной

группе и автоматически свяжем их атрибуты с аргументами шаблона экрана. Выполнима аналогичную операцию и для второго аргумента (рисунок 3.9).

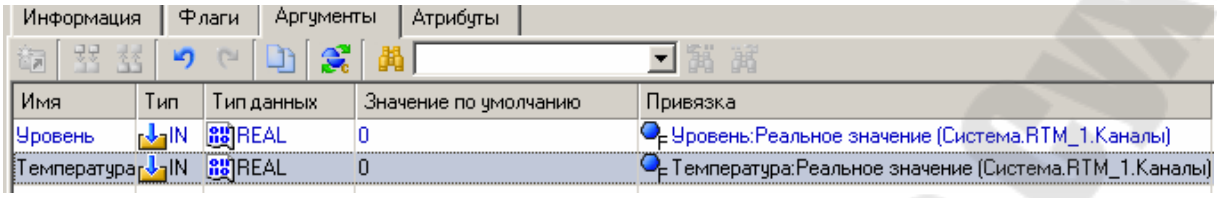


Рисунок 3.9 – Создание каналов по аргументам

3.4 Создание источников сигналов

В группе «**ИСТОЧНИКИ/ПРИЕМНИКИ**» навигатора проекта. создадим – «**ГЕНЕРАТОРЫ**» и два источника «**Треугольник**», «**Синусоида**» (рисунок 3.10).

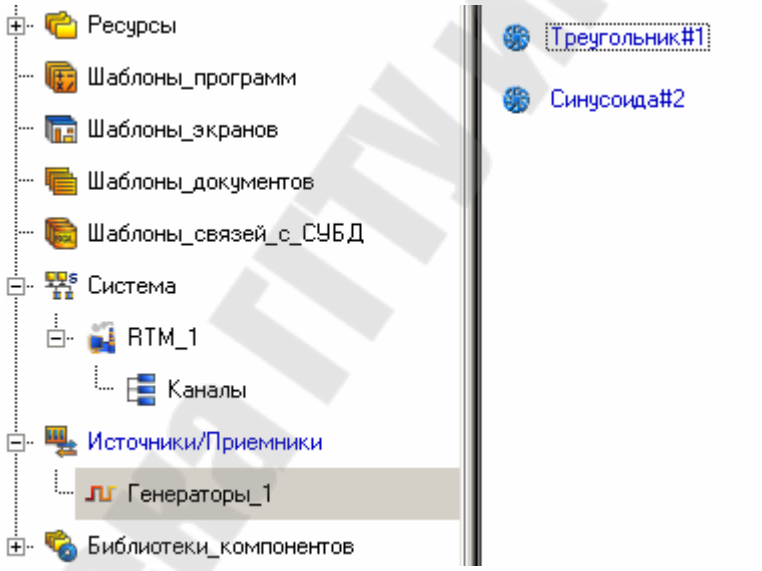


Рисунок 3.10 – Создание источников сигналов

С помощью механизма **drag-and-drop** выполним, привязку источников к каналам «**Треугольник**» – Уровень, «**Синусоида**» – температура.

3.5 Запуск проекта

Щелкните левой клавишей мыши по иконке  или по строчке **сохранить для MPB** в меню **файл**. В навигаторе проекта выделите созданный **RTM** узел и щелчком ЛК мыши по иконке  запустите

профайлер. Для запуска проекта щелкнуть по . Результат работы представлен на рисунке 3.11.

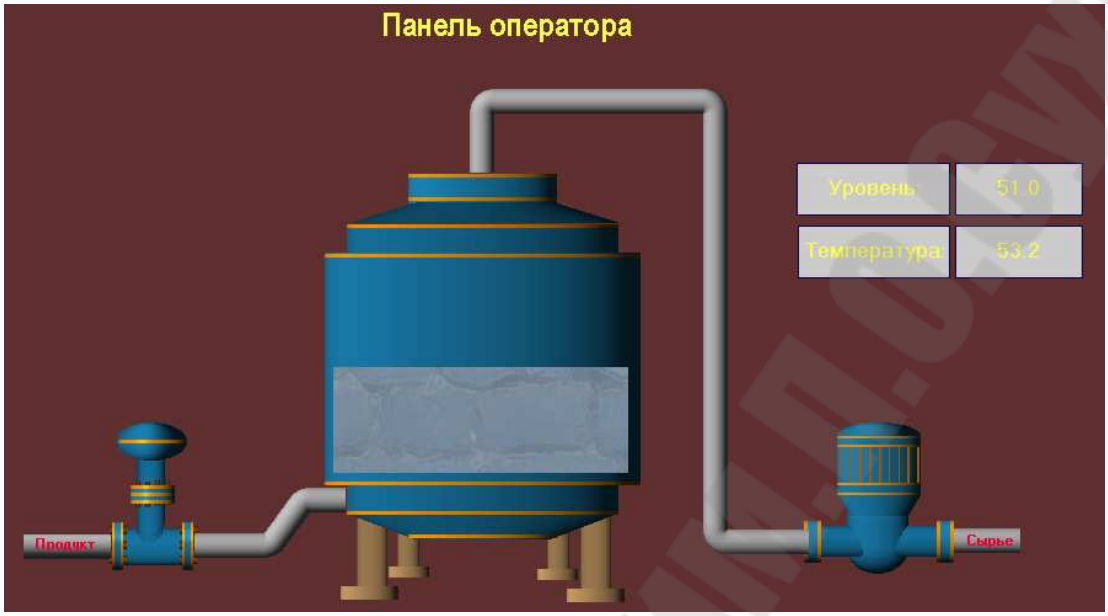


Рисунок 3.11 – Создание источников сигналов

4. Контрольные задания

1. создать новый проект,
2. создать канал, генератор – «**Треугольник**» и связать их;
3. создать экран, разместить на нем графический элемент – прямоугольник . Задать динамическое изменения контура – свойство «**динамический контур** »;
4. создать экран, разместить на нем графический элемент – эллипс . Задать динамическое изменения контура – свойство «**динамическая трансформация** » – «**Перемещение**»;
5. создать экран, разместить на нем графический элемент – эллипс . Задать динамическое изменения контура – свойство «**динамическая трансформация** » – «**Масштабировать**»;
6. создать экран, разместить на нем графический элемент – стрелка . Задать динамическое изменения контура – свойство «**динамическая трансформация** » – «**Вращение**».

5. Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, созданные каналы, динамические объ-

екты, привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции.

6. Контрольные вопросы

1. Статические графические объекты Trace Mode
2. Динамические графические объекты Trace Mode
3. Динамические эффекты – динамический контур
4. Динамические эффекты – заливка
5. Динамические эффекты – трансформация

Лабораторная работа №4
Микропроцессорная реализация коррекции

Цель работы: Изучить технологию создания распределенных систем мониторинга на примере УСО I-7033D, с датчиком температуры типа - термометр сопротивления.

1. Теоретические сведения

1.1 Идеология распределенных комплексов

TRACE MODE имеет мощные средства для создания распре деленных АСУТП, включающих в себя до трех уровней иерархии: уровень контроллеров – нижний уровень; уровень операторских станций – верхний уровень; административный уровень. Деление на уровни иногда может быть весьма условным. В малых системах функции всех уровней часто реализуются на одной операторской станции. В крупных же на каждом уровне может быть выделена своя иерархия. При разработке крупных сетевых систем, включающих в себя десятки узлов, лимитирующим элементом становятся не характеристики пакета по количеству одновременно работающих в сети узлов, а пропускная способность линий связи. Исполнительная система TRACE MODE включает в себя мониторы, предназначенные для работы на всех уровнях систем управления.

1.1.1. Уровень контроллеров

На этом уровне реализуется сбор данных от датчиков и УСО. Для создания этого уровня предусмотрены мониторы: Микро MPB, Микро MPB Модем+, Микро MPB GSM+. Первый из них предназначен для запуска в контроллерах, связанных с верхним уровнем по локальной сети или последовательному интерфейсу, второй при связи по коммутируемым линиям, а третий - по GSM-сети. При использовании выделенных телефонных линий или радиоканалов следует применять первый монитор. Эти мониторы не имеют графического интерфейса. Однако по математическим функциям они идентичны мониторам верхнего уровня, а также имеют ряд функций, необходимых для работы в контроллерах (например, поддержка сторожевого таймера).

1.1.2. Оперативный уровень

Для верхнего уровня АСУТП предусмотрены такие мониторы, как **MPB. NetLiuk MPB. NetLiuk Light.** Они позволяют создавать рабочие станции оперативного управляющего персонала.

MPB может обмениваться данными с другими мониторами TRACE MODE, а также с любыми контроллерами через встроенные протоколы или драйвер. Он запрашивает данные у нижнего уровня и передает ему команды управления. Полученные данные могут отображаться, архивироваться и передаваться другим приложениям WINDOWS по протоколам ODBC, OPC и DDE.

NetLink MPB - это сетевая рабочая станция. Этот монитор может обмениваться данными с операторскими станциями (по последовательному интерфейсу или локальной сети), а также с **Микро MPB,** работающими в PC-based контроллерах. По функциям визуализации, архивирования, связи с базами данных и документирования **NetLink MPB** аналогичен **MPB**. В отличие от MPB, в нем блокированы поддержка плат УСО, обмен с драйвером, обмен по встроенным протоколам MODBUS и DCS, а также клиентские функции OPC и DDE.

NetLink Light - это сетевой графический терминал. Он не имеет своего сервера матобработки, а связывается с сервером MPB или NetLink MPB, запушенным на другом компьютере. **NetLink Light** позволяет создавать дополнительные рабочие места оператора.

1.1.3. Административный уровень

Задачей данного уровня управления является контроль текущего состояния производственных процессов и анализ **функционирования** производства по архивным данным.

Для решения задач данного уровня предусмотрен монитор **SUPERVISOR**. Он является специализированной графической консолью, которая может подключаться к серверу матобработки **MPB. NetLink MPB** или **глобальный регистратор.** В первых двух случаях просматривается локальный СПАД архив, а в последнем - глобальный архив. Кроме того. **SUPERVISOR** можно переключить в режим реального времени. В этом случае он работает как консоль **NetLink Light** и может использоваться для управления процессом.

При работе с архивами SUPERVISOR реализует следующие функции: отображение последних изменений значений каналов: про-

смотр архивов в режиме **PLAYBACK:** просмотр на заданное архивное время с пошаговым переходом по времени.

До тех пор, пока речь идет о связи между компонентами одного узла, не возникает вопрос об аппаратно/программном интерфейсе, который должен быть задействован для обеспечения связи. В этом случае достаточно выполнить конфигурирование свойств связь вызов компонентов. Если взаимодействующие компоненты относятся к разным узлам, необходимо сконфигурировать интерфейс связи.

Мониторы реального времени TRACE MODE могут обмениваться данными по следующим линиям:

- локальная сеть: последовательный интерфейс RS-232, RS-485, RS-422;
- радиоканал: выделенная телефонная линия: коммутируемые телефонные линии:
- сети GSM.

По этим носителям необходимо организовать информационные потоки всех уровней системы управления. При этом могут реализоваться как вертикальные связи (между уровнями), так и горизонтальные (между узлами одного уровня).

Например, при задании связи двух каналов разных узлов по **RS**–протоколу необходимо создать в узлах компоненты **COM-порт,** задать для них необходимые параметры и указать для канала-приемника используемый интерфейс связи.

1.2 Обмен данными в SCADA-системе TRACE MODE

Мониторы реального времени TRACE MODE могут обмениваться данными по следующим линиям: локальная сеть; последовательный интерфейс RS-232, RS-485, RS-422; радиоканал; выделенная телефонная линия; коммутируемые телефонные линии; сети GSM. По этим носителям необходимо организовать информационные потоки всех уровней системы управления. При этом могут реализоваться как вертикальные связи (между уровнями), так и горизонтальные (между узлами одного уровня).

1.2.1. Последовательный интерфейс обмена данными

Обмен по всем линиям кроме локальной сети, реализуется через последовательный порт по протоколу **M-Link.** Узлы в сети **M-Link** неравноправны: один имеет статус Master, а остальные - Slave. Такие сети следует использовать для связи между операторскими станциями и контроллерами. Монитор со статусом Master является активным. Он посыпает команды управления и запросы на передачу информации. Монитор со статусом Slave принимает посланные ему команды и передает запрошенные данные. Команды управления содержат указания на изменение значений атрибутов каналов удаленного узла.

Таким образом, запросы, посылаемые монитором со статусом Master, могут быть двух типов:

1) запрос данных (используется для получения значений каналов или другой информации от монитора со статусом Slave):

2) запрос на изменение (используется для изменения значений атрибутов каналов на удаленном мониторе). В запросах на изменение передаются новые значения корректируемых атрибутов удаленной базы.

В одной сети M-Link не может быть двух мониторов, для которых установлен статус Master. Чтобы один монитор выступал и как Master, и как Slave, надо создать **параллельные** сети, используя при этом по два последовательных порта на каждом узле. Тогда два монитора смогут работать в режиме Master.

1.2.2. Обмен по протоколу M- Link

Для обмена данными между мониторами TRACE MODE по последовательному интерфейсу используется протокол M-Link. Он применяется для обмена по интерфейсам RS-232, RS-485, RS-422, радиоканалу, коммутируемым телефонным линиям и GSM сети.

Используя протокол M-Link, в рамках TRACE MODE можно создавать сетевые комплексы на базе последовательного интерфейса RS-485. Такие комплексы могут включать в себя до 128 узлов (контроллеров и операторских станций). При этом связь может осуществляться по нескольким последовательным портам.

Для связи двух мониторов можно использовать интерфейс RS-232. Для организации взаимодействия с несколькими удаленными узлами по этому интерфейсу - необходимо иметь соответствующее количество последовательных портов. Это позволяет организовать связь типа "звезда". Такая конфигурация может потребовать дополнитель-

ных затрат на многоканальные платы. Однако она позволяет быстрее передавать данные за счет распараллеливания обмена с разными удаленными узлами. TRACE MODE поддерживает обмен одновременно по 32 последовательным портам.

Для связи сильно разнесенных в пространстве мониторов можно использовать радиоканал, выделенные или коммутируемые телефонные линии. В этих случаях нужны дополнительные устройства - модемы. Они согласуют электрические характеристики последовательных портов и используемой среды передачи.

1.2.3. Организация ввода/вывода данных. Настройка каналов

Для обмена данными по последовательному интерфейсу между мониторами Trace Mode применяются каналы подтипа **СВЯЗЬ.** В зависимости от направления передачи информации используются разные дополнения к подтипу этих каналов. **Для запроса данных** по протоколу M-Link предназначены каналы подтипа **СВЯЗЬ** с дополнением **In_M_Link** и дополнением **In_M_Link(T)**. Для второго вместе со значением канала передается время его последнего изменения. При этом отображаемое время изменения значения канала соответствует времени того МРВ, из которого считывается канал. Оно копируется в соответствующий атрибут запрашивающего канала, а также заносится в архивы. Для передачи данных следует использовать каналы с дополнением **OUT_M_Link** и дополнением **OUT_M_Link(T)**. При считывании значения канала по **M-Link(T)** из МикроМРВ в МРВ отображаемое время изменения канала соответствует времени МРВ.

1.2.4. Настройка МРВ для обмена по M-Link

Для обмена данными по протоколу M-Link необходимо настроить соответствующие параметры запуска узла. К ним относятся статус узла, а также физические параметры связи.

Параметры обмена по протоколу M_Link настраиваются в бланках «**Основные**» и «**Параметры**» последовательных портов диалога «**Параметры узла**». Для входа в этот диалог необходимо нажать ПК на изображении настраиваемого узла в навигаторе проекта. Статус узла при обмене по протоколу M_Link задается в бланке «**Основные**» диалога «**Параметры узла**». Чтобы узел поддерживал статус Master, необходимо установить флаг M_Link в разделе **Host Mode** данного

бланка, а для поддержки режима SLAVE - тот же флаг в разделе **Slave Mode**.

Кроме статуса, при обмене по M_Link необходимо настроить физические параметры порта, через который будут передаваться данные. Для обмена данными с контроллерами по последовательным интерфейсам надо настроить используемые порты. Это реализуется в бланке «**Параметры последовательных портов**» диалога «**Параметры узла»**.

Этот бланк содержит список последовательных портов (COM1 - порт 0. COM32 - порт 31) и семь полей настройки параметров выбранного в списке порта. Такими параметрами являются:

- назначение порта;
- базовый адрес порта;
- скорость обмена;
- параметры связи;
- таймаут на ожидание ответа;
- номер используемого прерывания;
- режим управления передатчиком.

Значение параметра «Назначение порта» формируется из списка, содержащего четыре следующих пункта:

- Связь с контроллерами;
- Slave M_Link;
- Modem;
- GSM_SMS.

Связь с контроллерами. Это означает, что порт используется для обмена с контроллерами через внешний драйвер или по встроенным протоколам со статусом Master. Для обмена по протоколу M_Link со статусом Slave, в данном поле следует установить назначение – Slave M_Link. Режим связи Modem нужно установить для порта при его использовании для обмена по коммутируемым линиям, а GSM_SMS – при обмене по GSM сети.

Два поля бланка Параметры портов такие, как «Базовый адрес порта» и «Номер используемого прерывания» предназначены для задания базового адреса и номера прерывания порта. Они имеют смысл при настройке узла, запускаемого под управлением МикроМРВ. В остальных случаях эти параметры портов настраиваются средствами WINDOWS из Панели управления.

Следующее поле «Скорость обмена» заполняется из списка: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19.2k, 38.4k, 57.6k, 115.2k, 144k,

192k, 288k, 576k. Причем скорость обмена по протоколу M-LINK не должна быть ниже 600 бит/с. Её величина при обмене по последовательным портам ограничивается расстоянием и наличием помех в линии.

В поле «Параметры связи» задаются такие параметры обмена, как: количество информационных бит в посылке; количество стоповых бит; наличие проверки на четность. Эти строки имеют следующий формат: k-m-x ,

где k – количество информационных бит; m – количество стоповых бит; x – наличие проверки на четность (n – отсутствие проверки, e – проверка на четность, o – проверка на нечетность).

Значение поля «Таймаут на ожидание ответа» вводится непосредственно с клавиатуры. Оно задает время ожидания ответа от устройства, которому был послан запрос по данному порту. Если величина таймаута не задана, то она принимается равной 100 мс. Если в течение времени таймаута ответ на запрос от устройства или МРВ не пришел, то каналу, запрашивающему эти данные, взводится флаг аппаратной недостоверности.

Кроме того, для задания времени задержки на включение передатчика после завершения приема в каналах на базе RS-485 и RS-232 используется таймаут «RS-передача». Его значение в миллисекундах задается в бланке Таймауты того же диалога. В поле «Режим управления передатчиком» вносится «Нет», если не требуется управлять передатчиком. Остальные пункты, кроме первого, задают различные режимы управления.

2. Ход работы

2.1 Создание проекта

Запустить программу **TRACE MODE IDE 6** иконка  , создать и сохранить новый проект. В навигаторе проекта создать следующие объекты:

1. узел – **RTM**;
2. шаблон экрана и его аргумент – **Температура**;
3. создать объекты для отображения температуры – Тренд, цифровое значение и связать их с аргумент – **Температура**;
4. с помощью механизма автопостроения по аргументу, создать канал FLOAT.

2.2 Подключение модуля удаленного ввода сигналов

Введем в созданный проект модуль удаленного ввода **I-7033D** с подключенным к его входам датчиком – термометр сопротивления (платиновый) с международной градуировочной характеристикой **Pt100**. Предварительно настроим модуль с помощью конфигурационной утилиты **DCON_UTILITY**, поставляемой с модулем на указанную градуировочную характеристику (рисунок 2.1). Зададим «**инженерный**» формат вывода данных, присвоим ему номер в сети **RS-485** равный **1** и установим формат обмена данными **9600,n,8,1** без формирования контрольной суммы (рисунок 2.2). Подключим модуль к порту **COM1** компьютера через автоматический конвертор интерфейсов **I-7520R**, обеспечим питание обоих модулей.

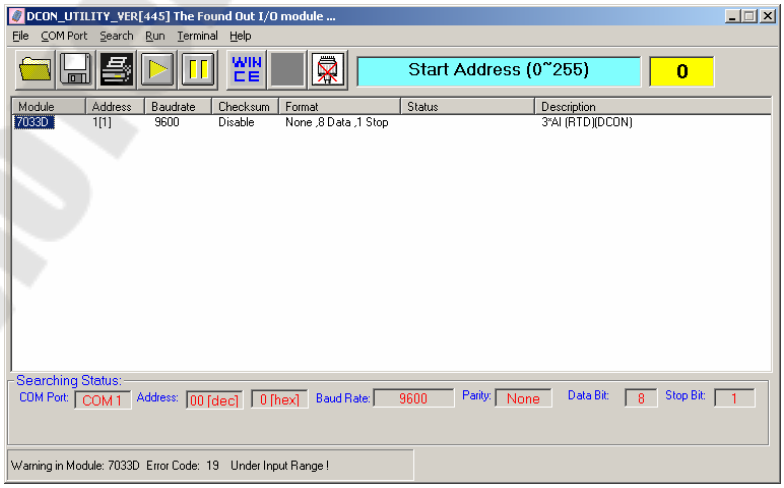


Рисунок 2.1 – Подключение модуля удаленного ввода I-7033D

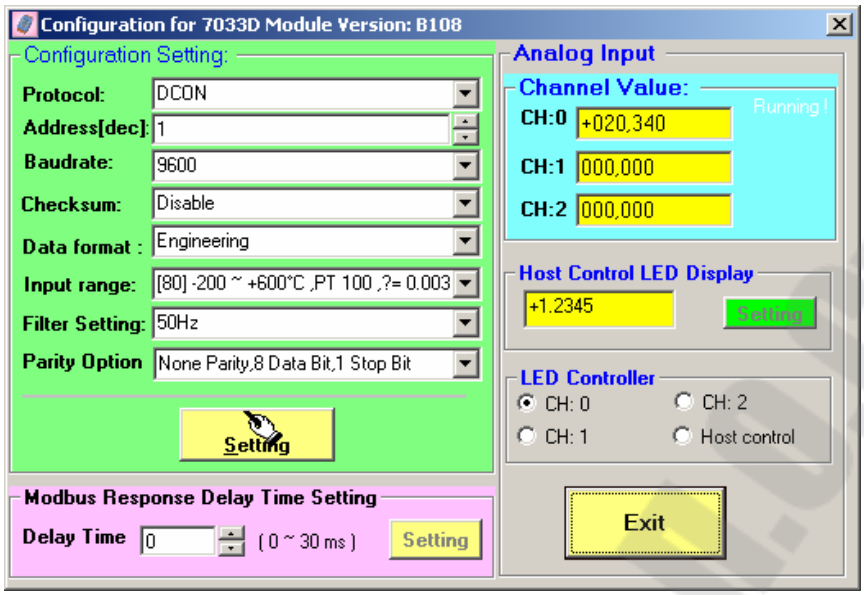


Рисунок 2.2 – Настройка модуля удаленного ввода I-7033D

2.3 Создание компонента-источника для ввода данных от модуля I-7033D

Создадим компоненты-источники, связанные с выбранным типом аппаратуры ввода/вывода, и произведем настройку их атрибутов следующим образом:

– откроем ЛК слой **Источники/Приемники** и через ПК создадим в нем группу **Распределенное УСО (DCS)**;

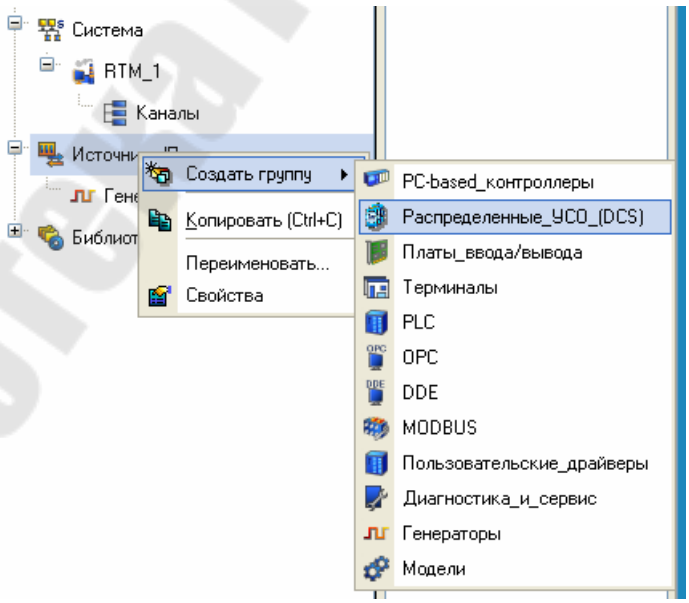


Рисунок 2.3 – Создание группы **Распределенное УСО**

– двойным щелчком ЛК откроем группу компонентов **Распределенное УСО (DCS)_1** и через контекстное меню, вызываемое по щелчку ПК, создадим в ней группу **I-7000**;

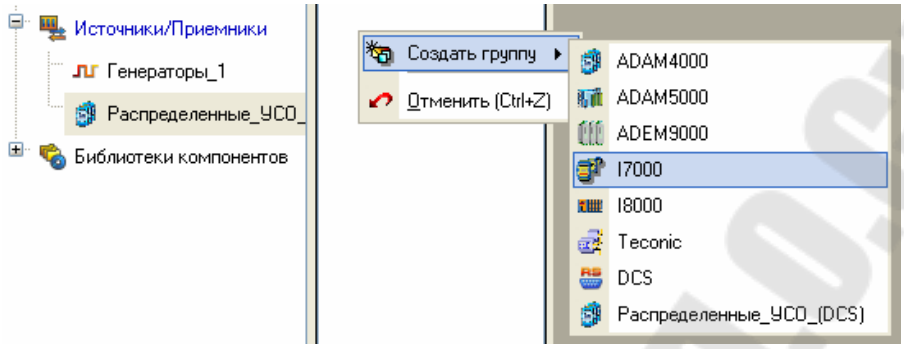


Рисунок 2.4 – Создание группы **I-7000**

– откроем созданную группу **I7000_1** двойным щелчком ЛК и через щелчок ПК создадим в ней подгруппу **I7033D#1**;

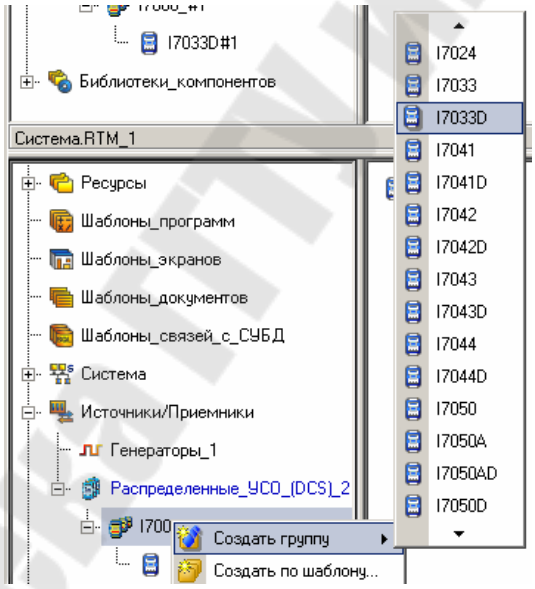


Рисунок 2.5 – Создание подгруппы **I7033D#1**

– откроем созданную подгруппу **I7033D#1** двойным щелчком ЛК и перейдем к редактированию созданных компонентов, описывающих подключение модуля I-7033D к АРМ. Выделим ЛК компонент **AIn#1** и двойным щелчком ЛК перейдем в режим редактирования его атрибутов

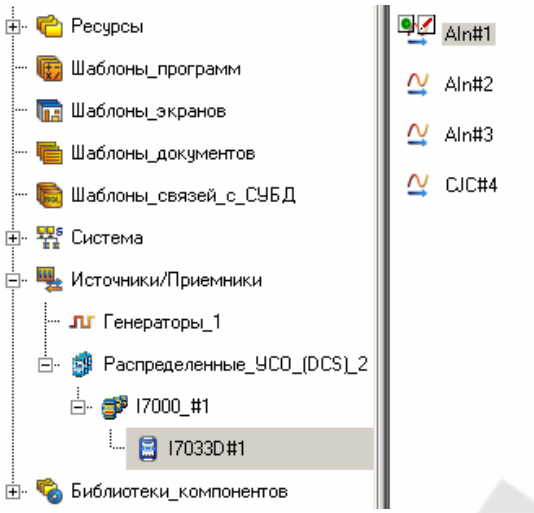


Рисунок 2.6 – Компоненты **УСО I7033D**

– все основные атрибуты, задающие настройки модуля, оставим заданными по умолчанию: **Номер порта** 0 соответствует COM1 компьютера АРМ, **Адрес** модуля в нашем случае 1, атрибуты **Канал** и **Слот** для выбранного модуля не задаются, **Контрольная сумма** – отсутствует, **Направление** – Вход. Изменим из выпадающего меню **Тип сигнала** в соответствии с типом подключенного датчика и введем **Комментарий**.

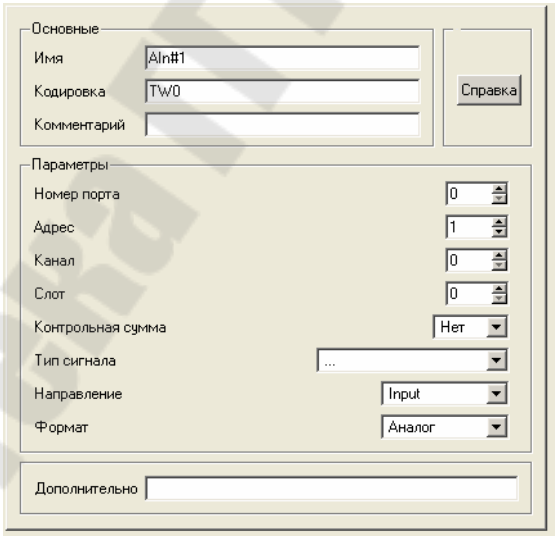


Рисунок 2.7 – Редактирование атрибутов **УСО I7033D**

2.4 Создание и настройка COM-порта

Поскольку модуль I-7033D подключается к узлу по последовательному интерфейсу, создадим и настроим последовательный порт для узла **RTM_1**. Для этого:

– откроем в окне Навигатора проекта слой **Система/RTM_1**, выделим ЛК узел **RTM_1** и через ПК создадим группу **COM-порты**;

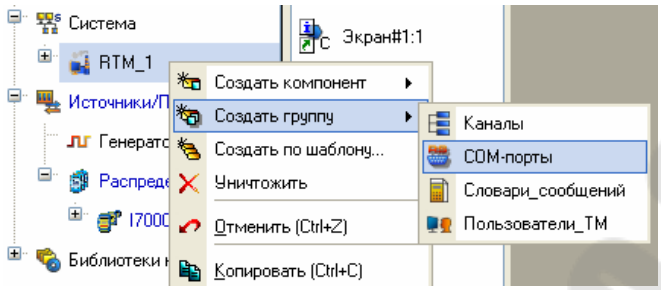


Рисунок 2.8 – Создание группы **COM-порты**

– двойным щелчком ЛК откроем вновь созданную группу и в окне компонентов выделим ЛК **COM-порт#1**;

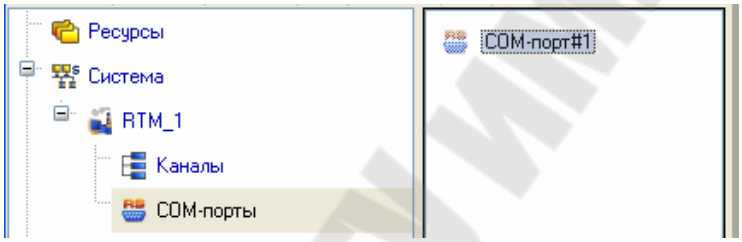


Рисунок 2.9 – Окно навигатора объектов (компонент **COM-порт#1)**

– двойным щелчком ЛК на выделенном объект **COM-порт#1** откроем для редактирования его атрибуты и настроим его свойства.

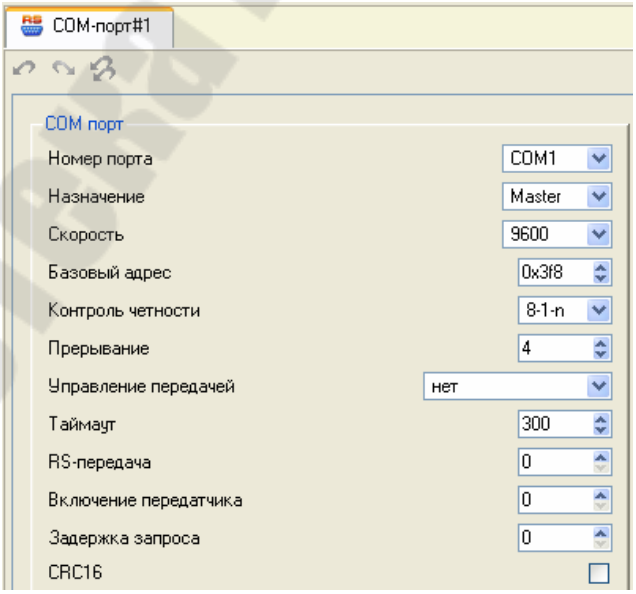


Рисунок 2.10 – Окно атрибутов объекта **COM-порт#1**

2.5 Привязка канала к источнику данных

Выполним привязку канала к источнику данных. Откроем с помощью кнопки  дополнительное окно «**Навигатора проекта».** С помощью ЛК перетянем компонент-источник аналогового входа модуля **AIn#1** на канал, открытый в дополнительном окне.

2.6 Запуск проекта

Щелкните левой клавишей мыши по иконке  или по строчке **сохранить для MPB** в меню **файл**. В навигаторе проекта выделите созданный **RTM** узел и щелчком ЛК мыши по иконке  запустите профайлер. Для запуска проекта щелкнуть по  .

3. Контрольные задания

1. Настроить масштабирование в канале для вывода температуры в Кельвинах или Фаренгейтах;
2. Настроить сглаживание в канале;
3. Привязать 3 линии тренда к следующим атрибутам канала;
   - **Входное значение** (2, **In**);
   - **Аппаратное значение** (1, **A**);
   - **Реальное значение** (0, **R**).

4. Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, созданные каналы, динамические объекты, привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции.

5. Контрольные вопросы

1. Идеология распределенных комплексов. Уровень контроллеров.
2. Идеология распределенных комплексов. Оперативный уровень.
3. Идеология распределенных комплексов. Административный уровень.
4. Обмен данными в SCADA-системе TRACE MODE. Последовательный интерфейс обмена данными.

5. Обмен данными в SCADA-системе TRACE MODE. Обмен по протоколу M- Link.
6. Организация ввода/вывода данных. Настройка каналов.
7. Настройка МРВ для обмена по M-Link.

Лабораторная работа №5
Реализация логического управления в SCADA TRACE MODE 6 с использованием языка Texno FBD

Цель работы: освоить методику программирования логических функций при помощи SCADA–системы TRACE MODE на языке Техно FBD.

1. Теоретические сведения

Язык FBD (Functional Block Diagram, диаграмма функциональных блоков) является графическим языком и наиболее удобен для программирования процессов прохождения сигналов через функциональные блоки. Программа на языке FBD представляет собой совокупность функциональных блоков (functional flocks, FBs), которые соединяются линиями связи (connections). Каждый блок представляет собой математическую операцию (сложение, умножение, триггер, логическое “или” и т.д.). Начальные значения переменных задаются с помощью специальных блоков – входов или констант, выходные цепи могут быть связаны либо с физическими выходами контроллера, либо с глобальными переменными программы.

Типичным применением языка FBD является описание "жесткой логики" и замкнутых контуров систем управления. Язык функциональных блоков является удобным средством при программирования задач промышленной автоматизации. Практика показывает, что FBD является наиболее распространенным языком стандарта IEC. Графическая форма представления алгоритма, простота в применении и повторном использовании, библиотеки функциональных блоков делают язык FBD незаменимым при разработке программного обеспечения ПЛК. К недостаткам можно отнести, неудобство (не очевидность) реализации релейной логики.

Дальнейшим развитием языка FBD, является CFC (Continuous Flow Chart). Язык был специально создан для проектирования систем управления непрерывными технологическими процессами. Проектирование сводится к выбору из библиотек готовых функциональных блоков, установке связей между ними, а также настройке параметров выбранных блоков. В отличие от FBD, функциональные блоки языка CFC выполняют не только простые математические операции, а ориентированы на управление целыми технологическими единицами. Так

в типовой библиотеке CFC блоков находятся комплексные функциональные блоки, реализующие управление клапанами, моторами, насосами; блоки, генерирующие аварийные сигнализации; блоки PID-регулирования и т.д. Вместе с тем доступны и стандартные блоки FBD. Унаследовав от FBD саму концепцию программирования, язык CFC в наибольшей степени ориентирован на сам технологический процесс, позволяя разработчику абстрагироваться от сложного математического аппарата.

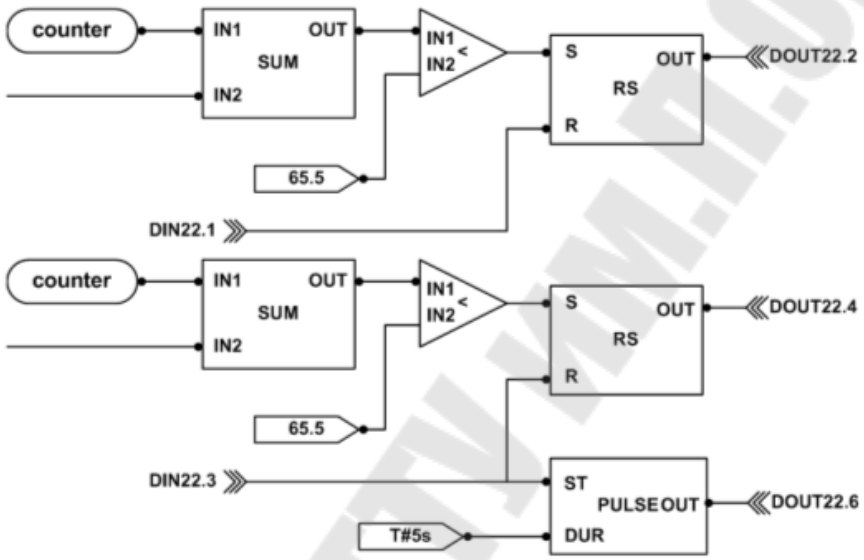


Рисунок 1.1 – Пример программы на языке FBD

Функциональный блок – это графическое изображение вызова встроенной функции Техно FBD (FBD-блока). В качестве примера рассмотрим функциональный блок, производящий сложение. Изображение его приведено на рисунке 1.2.

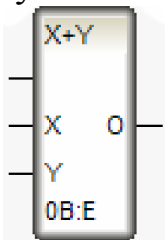


Рисунок 1.2 – Функциональный блок сложения на языке Texno ST

В верхней части блока указывается обозначение блока. Внизу выводится его номер. Номера блоком приписываются автоматически при их размещении в рабочем поле редактора программы. После

двоеточия указывается номер функционального блока, который будет выполняться следующим.

Горизонтальные линии, расположенные слева, выступают в качестве входов, на которые подается та или иная описанная локальная или глобальная переменная, аргумент программы, выходы с других функциональных блоков. На вход можно подать аргументы, тип которых **In** или **In/Out**. У каждого входа указываются его название. В указанном примере названия: X, Y. Безымянный вход, расположенный сверху, управляет выполнением блока: блок выполняет действие (в данном случае сложение) в том случае, если подается 0 или вход не подключен, в противном случае функциональный блок не выполняется.

Горизонтальная линия справа обозначает выход, содержащий результат выполнения функционального блока. Выход можно соединить с входом другого функционального блока. Выход функционального блока можно привязать к описанной глобальной или локальной переменной, аргументу, тип которого **Out** или **In/Out**.

Для размещения функционального блока следует открыть окно – «**FBD блоки**», для чего следует щелкнуть левой клавишей мыши по иконке  или выбрать – «**палитра FBD блоков**» в меню «**Вид**». Появится окно FBD блоки (рисунок 1.3).

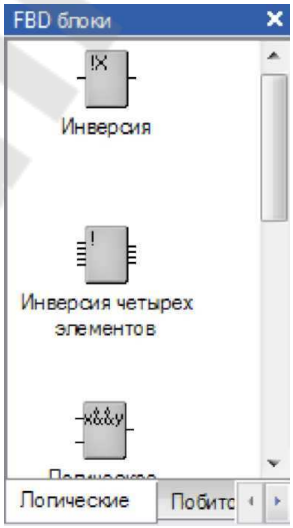


Рисунок 1.3 – Окно **FBD блоки**

В данном окне выбирается нужная закладка (логические, побитовые, арифметические) в соответствии с тематикой нужного блока. В окне отображается ряд блоков, относящихся к данной тематике. Среди этих блоков отыскивается нужный блок и перетаскивается на поле

редактора программы с использованием механизма drag-and-drop. То есть, курсор наводится на необходимый функциональный блок. Нажимается левая клавиша мыши. Блок «перетаскивается» на рабочее поле в нужное место при нажатой левой клавише мыши. Когда курсор переведен в положение, где должен располагаться функциональный блок левая клавиша отпускается.

Для соединения выхода одного функционального блока с входом другого следует навести курсор на выход функционального блока и нажать левую клавишу мыши. Блок станет синим, а имя выделенного выхода будет зеленым. Не отпуская клавиши мыши, наведем курсор на нужный вход функционального блока и отпустим клавишу мыши. Соответствующий вход и выход будут соединены линией, если все было правильно сделано. Аналогично можно нажать левую клавишу мыши, наведя на нужный вход функционального блока, и отпустить, наведя курсор на требуемый выход функционального блока.

Для привязки входа или выхода функционального блока следует выделить вход (выход) щелчком левой клавиши мыши. Появится контекстное меню, в котором следует выбрать – «**привязать**». Появится окно со списком аргументов и переменных, к которым можно привязать вход (выход) функционального блока. В этом списке следует выбрать переменную или аргумент (рисунок 1.4).

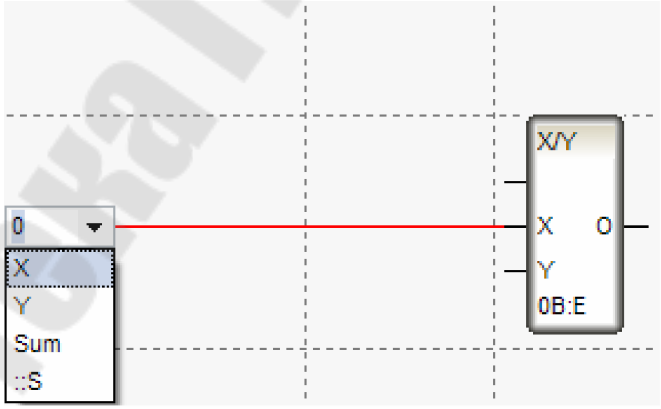


Рисунок 1.4 – Привязка входа (выхода) функционального блока

Создание константы рассмотрим на примере создания константы равной – 1e-9. Для этого следует выделить вход, на который необходимо подать константу, вызвать контекстное меню и выбирать – **привязать**. Будет выведено окно, как указано на рисунке 1.4. В поя-

вившемся окне вместо выбора аргумента или переменной вводится значение константы. В данном случае вводится – «1e-9».

В качестве примера приведена реализация примера программы, написанной на языке Texno ST, используя язык Texno FBD (рисунок 1.5).

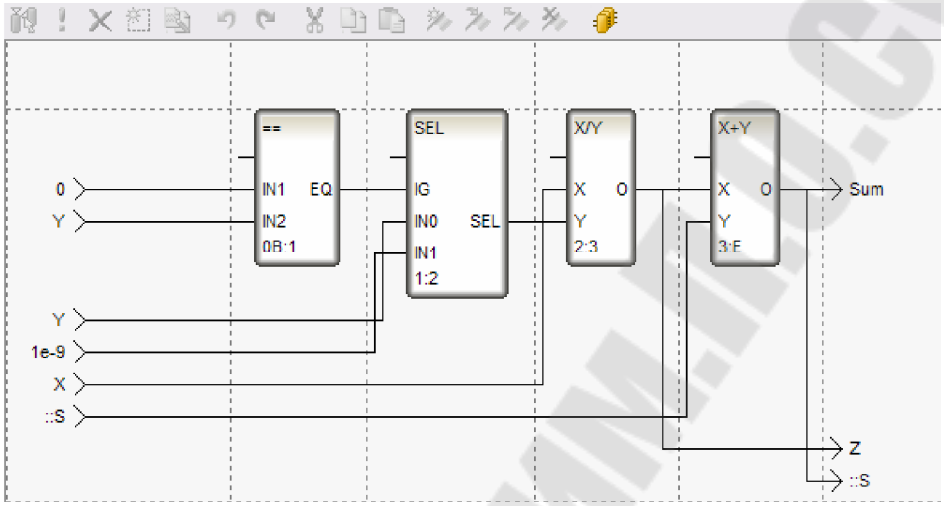


Рисунок 1.5 – Пример FBD программы

2. Ход работы

Для изучения средств разработки и основных элементов проекта, рассмотрим процесс его создания на примере.

2.1 Создание узла АРМ

Рассмотрим реализацию логической функции $f = x_1 x_2 \overline{x_4} \cup \overline{x_1} \overline{x_2} \cup x_1 \overline{x_3} \cup \overline{x_2} x_4$ при помощи SCADA–системы TRACE MODE на языке Texno FBD.

Разработка любого проекта автоматизации всегда начинается с запуска Интегрированной среды разработки (ИСР).

После запуска ИСР в меню «**Файл**» выбрать команду «**Настройки ИС**…». В появившемся окне в закладке «**Уровень сложности**» настроить проект как показано на рисунке 2.1, а затем выбрать закладку «**Отладка**» и настроить проект как показано на рисунке 2.2.

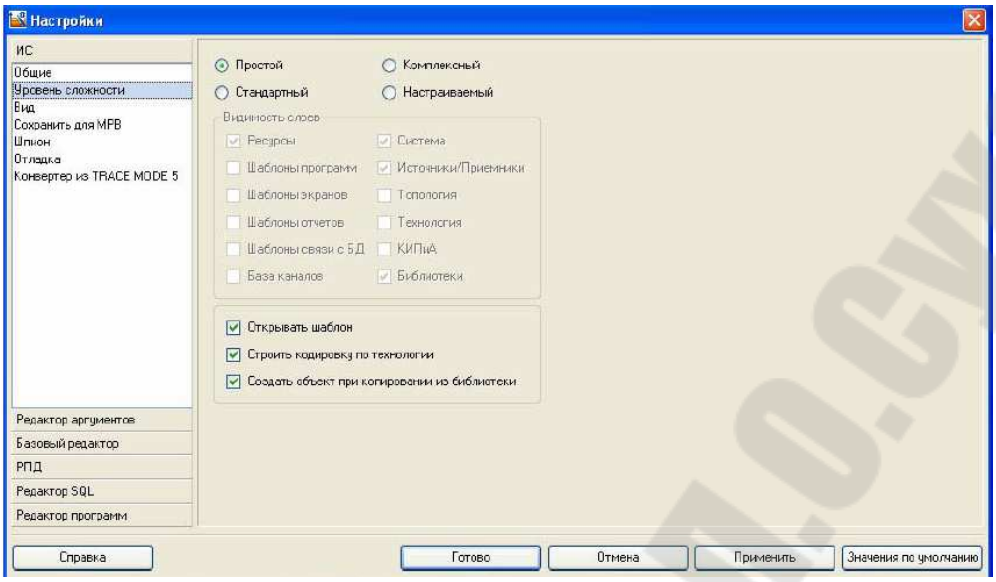


Рисунок 2.1 – Настройки «**Уровень сложности**»

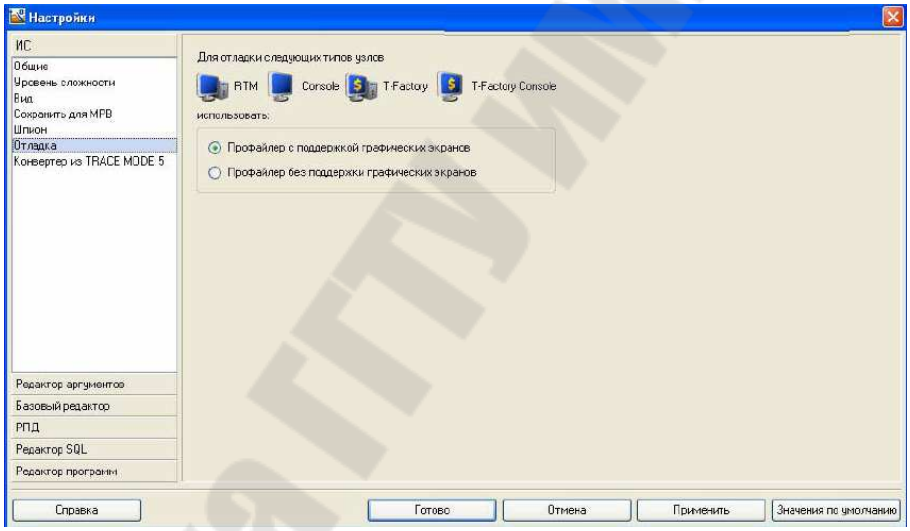


Рисунок 2.2 – Настройки «**Отладка**»

После проведенных настроек ИСР нажать кнопку «**Готово**». С помощью иконки  инструментальной панели создадим новый проект при этом в открывшемся на экране диалоге (рисунок 2.3).

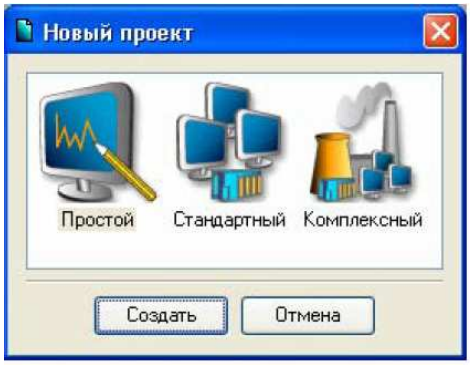


Рисунок 2.3 – Выбор типа проекта

Выберем «**Простой**» стиль разработки. После нажатия левой клавиши мыши (ЛК) на экранной кнопке «**Создать**», в левом окне Навигатора проекта появится дерево проекта с созданным узлом **АРМ RTM_1**. Откроем узел **RTM_1** двойным щелчком ЛК, в правом окне «**Навигатора проекта**» отобразится содержимое узла – пустая группа «**Каналы**» и один канал класса «Вызов» **Экран#1**, предназначенный для отображения на узле **АРМ** графического экрана (рисунок 2.4).

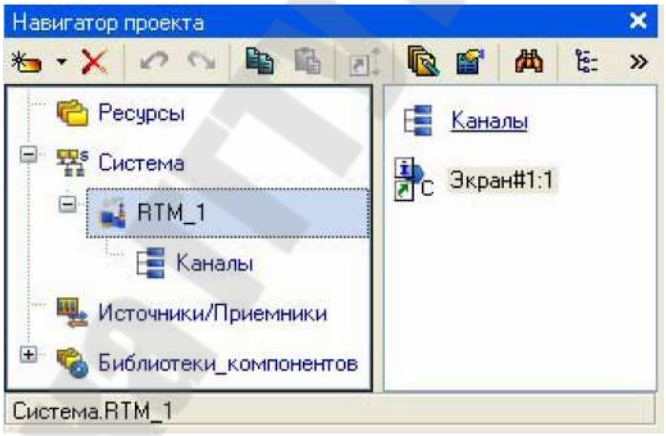


Рисунок 2.4 – Навигатор проекта

2.2 Создание графического интерфейса

Двойным щелчком ЛК по объекту **Экран#1** откроем окно графического редактора. Подготовим на экране вывод динамического текста для отображения численного значения входных переменных X1, X2, X3, X4 и выходного значения Y путем указания динамизации атрибута графического элемента (ГЭ). Определим назначение аргумента шаблона экрана. Создадим и разместим четыре ГЭ «**Текст**» 

как показано на рисунке 2.5.

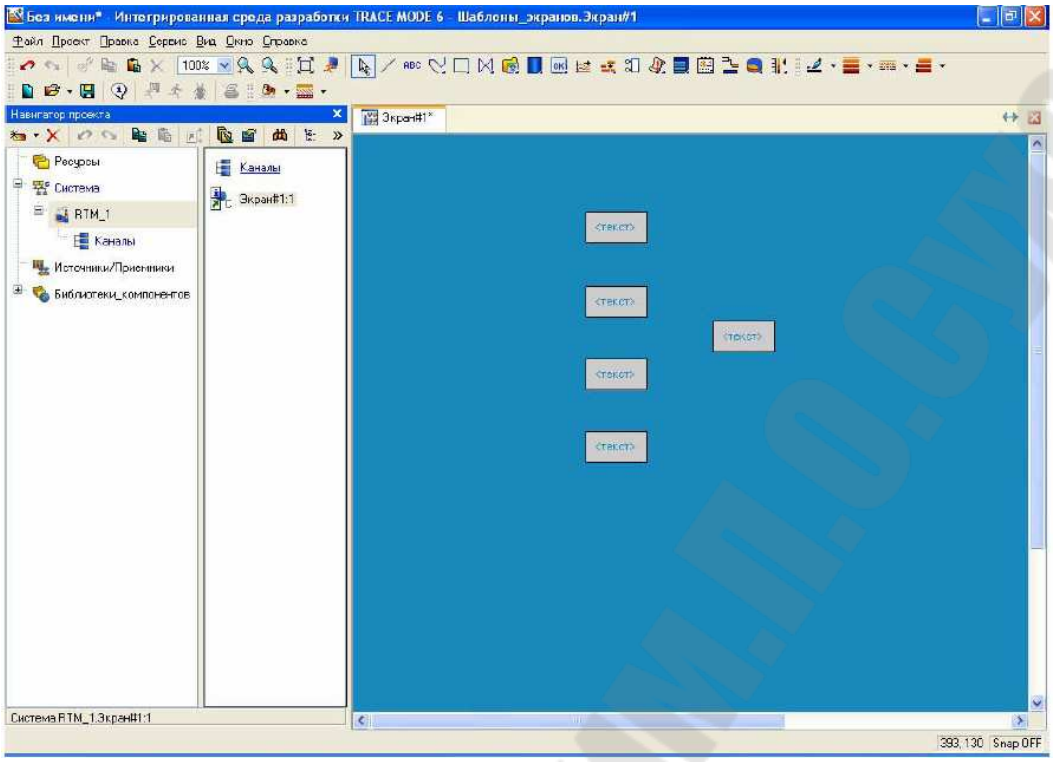


Рисунок 2.5 – Создание графических элементов

Откроем окно свойств ГЭ «**Текст**». Двойным щелком ЛК на строке «**Текст**» вызовем меню «**Вид индикации**» **(**рисунок 2.6).

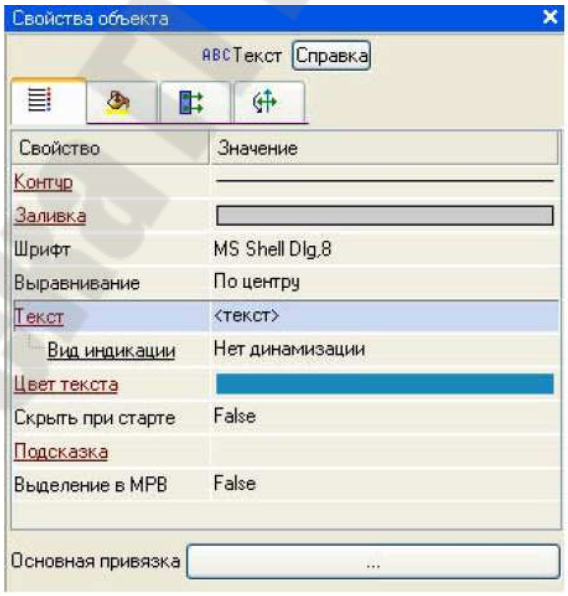


Рисунок 2.6 – Настройка индикации

В правом поле строки щелчком ЛК, вызвать список доступных типов. Выбрать тип **Значение (**рисунок 2.7).

Рисунок 2.7 – Выбор типа динамизации

В открывшемся меню настройки параметров динамизации, выбрать свойство «**Привязка**» (рисунок 2.8).

Рисунок 2.8 – Окно привязки

В открывшемся окне «**Свойства привязки**», нажав кнопку  на его панели инструментов, создать пять аргументов экрана 4 входных и один выходной рисунок 2.9.

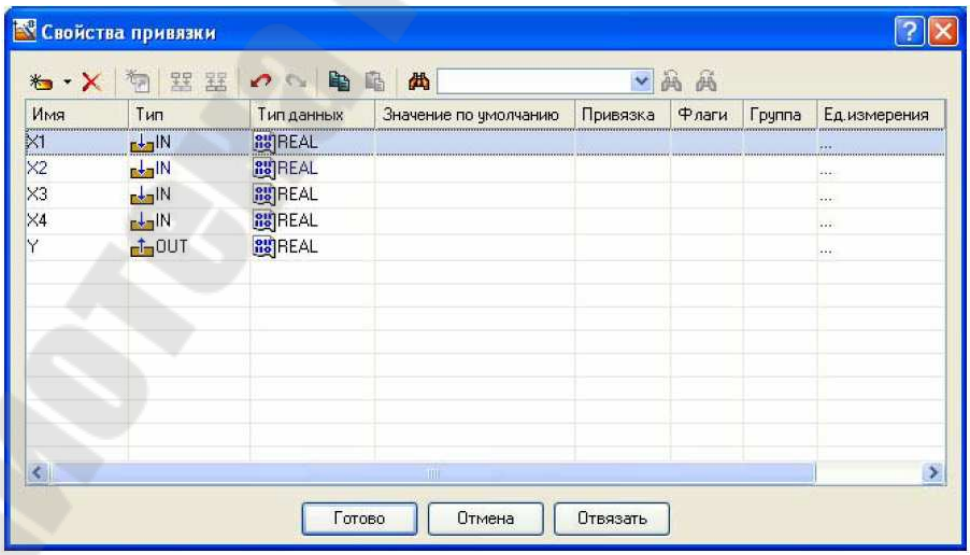


Рисунок 2.9 – Создание аргументов экрана

Двойным щелчком ЛК выделить имя аргумента и изменить его, введя с клавиатуры «**X1**» (завершить ввод нажатием клавиши Enter). Подтвердить связь с этим аргументом нажатием кнопки «**Готово**». Аналогичным образом настроить все **входные** и **выходные** аргументы.

Введем в состав графического экрана средство, позволяющее реализовать ввод числовых значений с клавиатуры. Создадим новый аргумент шаблона экрана для их приема.

Для этого, выбрать на инструментальной панели графического редактора иконку ГЭ Кнопка –  . С помощью мыши разместить его в поле экрана как показано на рисунке 2.10.

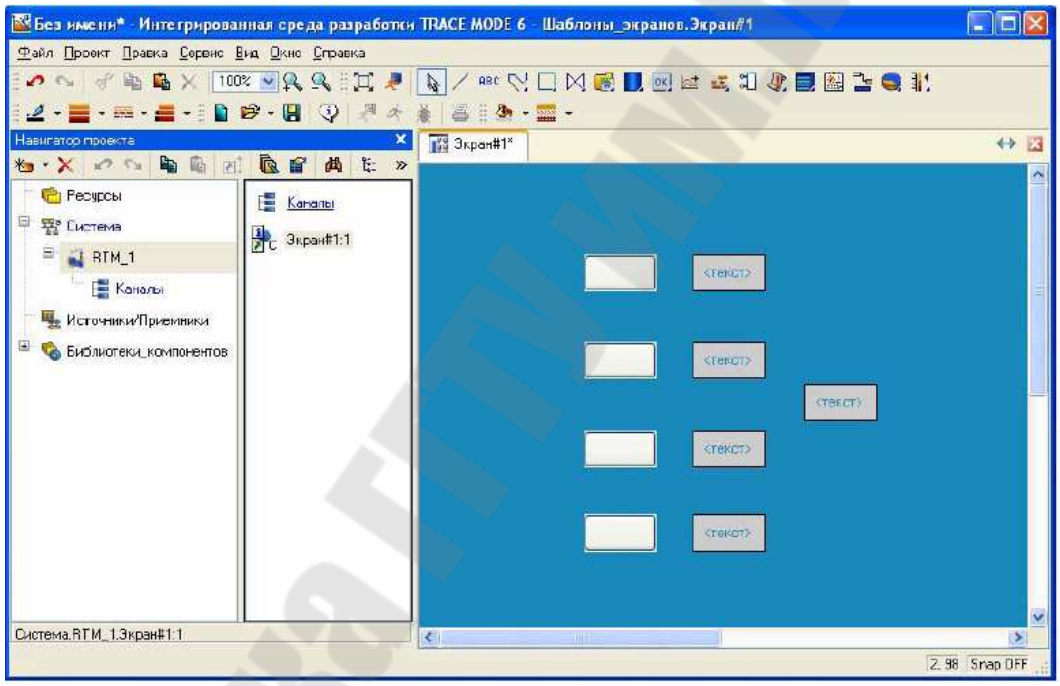


Рисунок 2.10 – Вид экрана

Для перехода в режим редактирования атрибутов размещенного ГЭ выделим ЛК иконку  на панели инструментов и двойным щелчком ЛК по ГЭ откроем окно его свойств (рисунок 2.11).

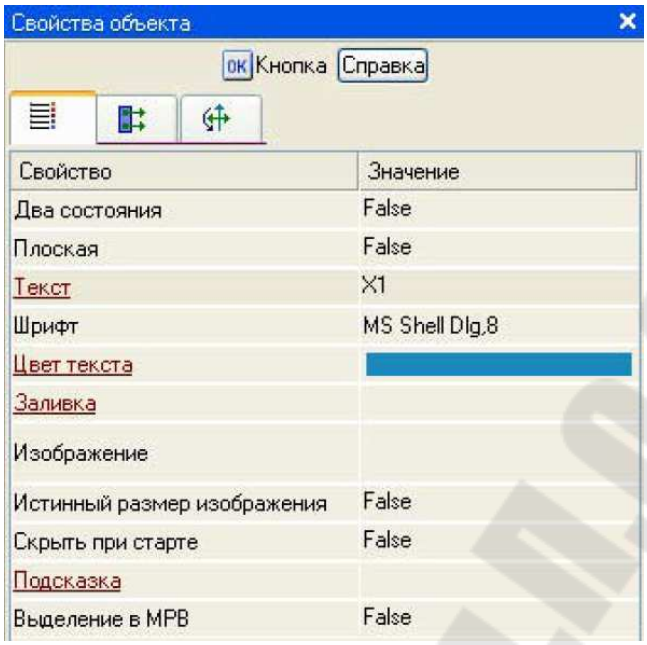


Рисунок 2.11 – Свойства графического элемента – «**Кнопка**»

В поле «**Текст**» ввести «**X1**». Откроем закладку «**События**» и раскроем меню «**По нажатию**» (**mousePressed**). Выберем из списка команду «**Добавить Send Value**» («**Добавить Передать значение**») (рисунок 2.12).

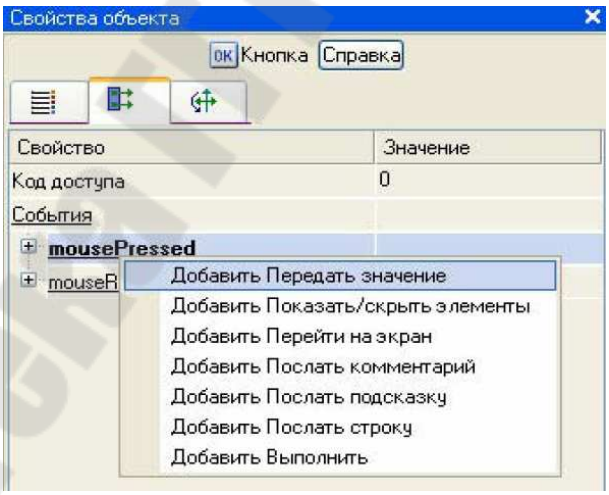


Рисунок 2.12 – Настройка типа передачи

В раскрывшемся меню настроек выбранной команды в поле «**Тип передачи (Send Type)**» выберем из списка «**Ввести и передать (Enter & Send)**» (рисунок 2.13).

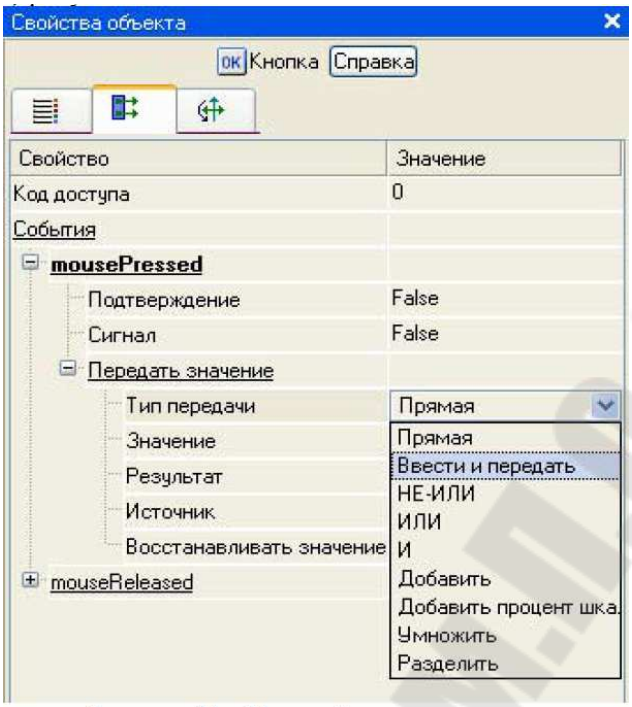


Рисунок 2.13 – Окно настройки событий элемента кнопка

Щелчком ЛК в поле «**Результат (Destination)**» вызовем табличный редактор аргументов и произведем привязку к аргументу **X1**. После привязки окно «**Свойства объекта**» выглядит следующим образом рисунок 2.14.

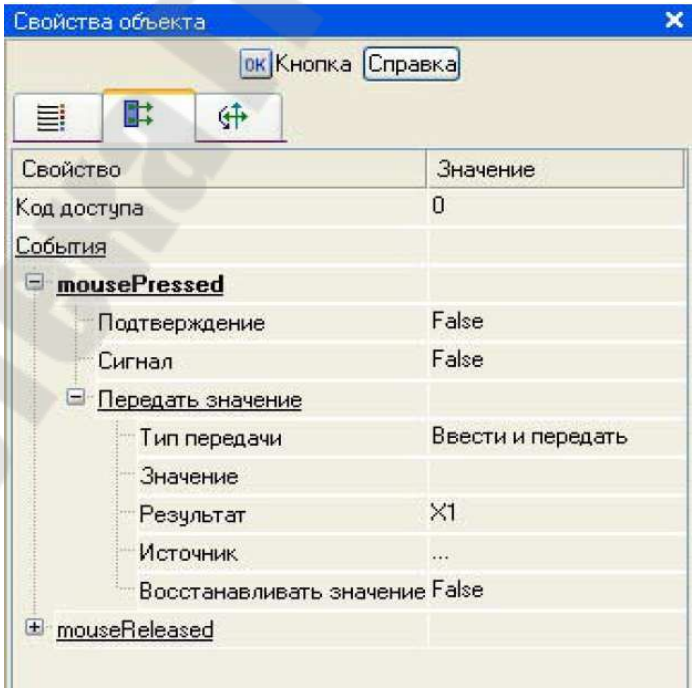


Рисунок 2.14 – Окончательный вид окна свойств графического элемента

Аналогичным образом настроить остальные кнопки **X2**, **X3**, **X4**.

2.3 Привязка аргумента экрана к каналу

Создадим по аргументам **X1, X2, X3, X4** и **Y** шаблона экрана новые каналы и отредактируем их привязку. В слое «**Система»** открыть узел **RTM_1**. С помощью ПК мыши, вызвать через контекстное меню свойства компонента **Экран#1.**

Выбрать вкладку «**Аргументы**», ЛК мыши выделить аргументы **X1**, **X2**, **X3**, **X4** и **Y** и с помощью иконки создать новый канал. В результате, в узле **RTM_1** ,будут автопостроены следующие каналы **X1**, **X2**, **X3**, **X4** и **Y** (рисунок 2.15).

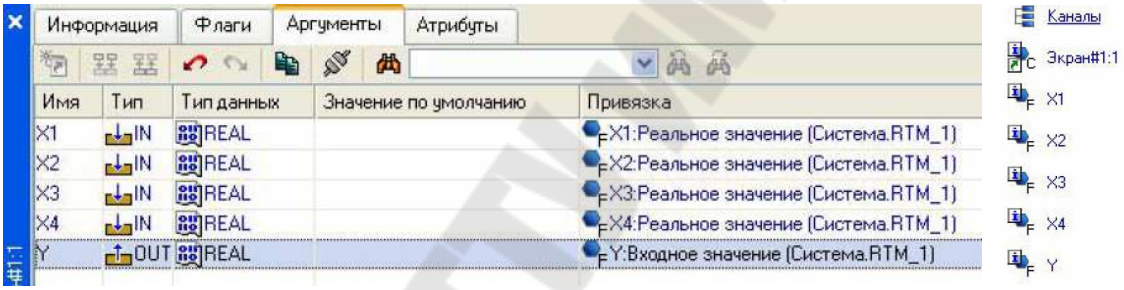


Рисунок 2.15 – Окно свойств экрана и созданные каналы

2.4 Создание программы на языке Техно FBD

Создадим программу, которая будет реализовывать заданную логическую функцию (см раздел «**Контрольное задание»**). Двойным щелчком ЛК открыть узел **RTM_1** и создать в нем компонент «**Программа» (**рисунок 2.16).

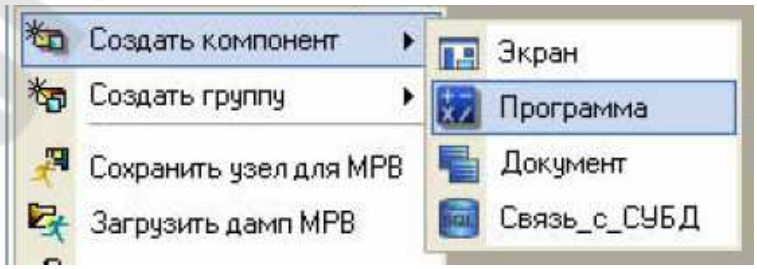


Рисунок 2.16 – Создание компонента Программа

Выделить компонент Программа#1 и ПК мыши вызвать контекстное меню, выбрав в котором ЛК мыши, пункт «**Редактировать шаблон»**, перейти в режим редактирования программы (рисунок 2.17).

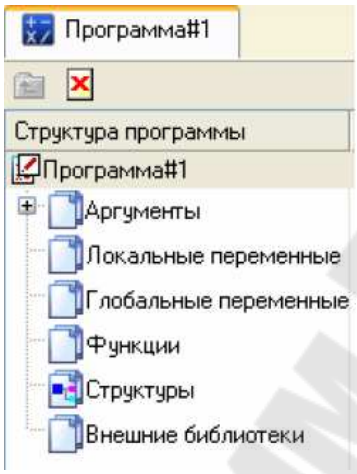


Рисунок 2.17 – Редактирование аргументов «**Программы»**

Выделением ЛК мыши в дереве шаблона **Программа#1** строку «**Аргументы**», вызвать табличный редактор аргументов. Кнопкой создать в редакторе аргументов четыре аргумента X1, X2, X3, X4 и выход Y. При этом первые 4 аргумента должны быть типа **IN**, а последний – **OUT (**рисунок 2.18).

Имя	Тип	Тип данных	Значение по умолчанию
X1	IN	REAL	
X2	IN	REAL	
X3	IN	REAL	
X4	IN	REAL	
Y	OUT	REAL	

Рисунок 2.18 – Аргументы программы

Выделить в дереве шаблона строку **Программа#1** и в открывшемся диалоге «**Выбор языка**» выбрать язык **FBD (**рисунок 2.19).

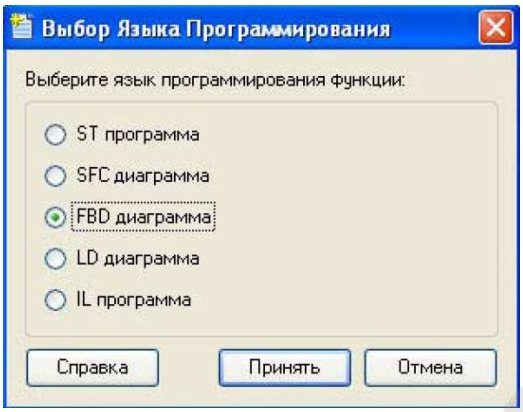


Рисунок 2.19 – Выбор языка программирования

По нажатию кнопки «**Принять**» в открывшемся окне редактора программ с объявленными переменными создать программу в соответствии с заданием (см. раздел «**Контрольное задание**»). Для выбора палитры FBD блоков необходимо ЛК мыши нажать на кнопку  после чего появится окно выбора FBD блоков (рисунок 2.20). При разработки программы верхние входы FBD блоков не используются т.к. они предназначены для изменения порядка пересчета блоков, а информационными входами, являются входы начиная со второго.

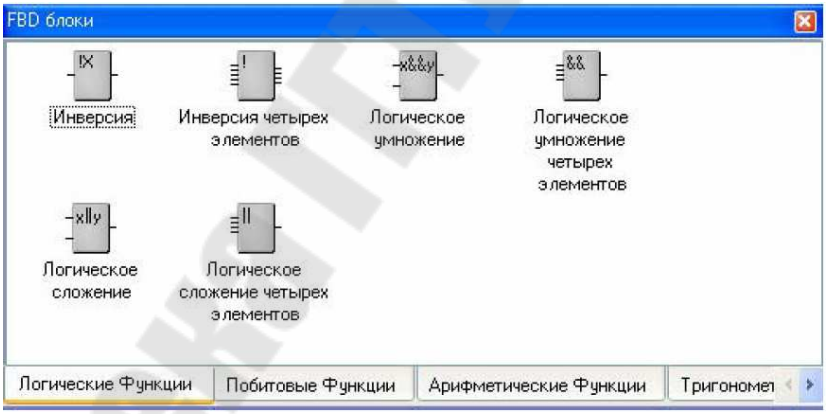


Рисунок 2.20 – Палитра FBD блоков

Для реализации заданной логической функции, выберем следующие блоки: из раздела «**Логические Функции**» **FBD блоки** инверсия (!X), логическое умножение (X&&Y и &&), логическое сложение (||). После размещения всех блоков для рассматриваемого примера программа будет выглядеть следующим образом – рисунок 2.21.

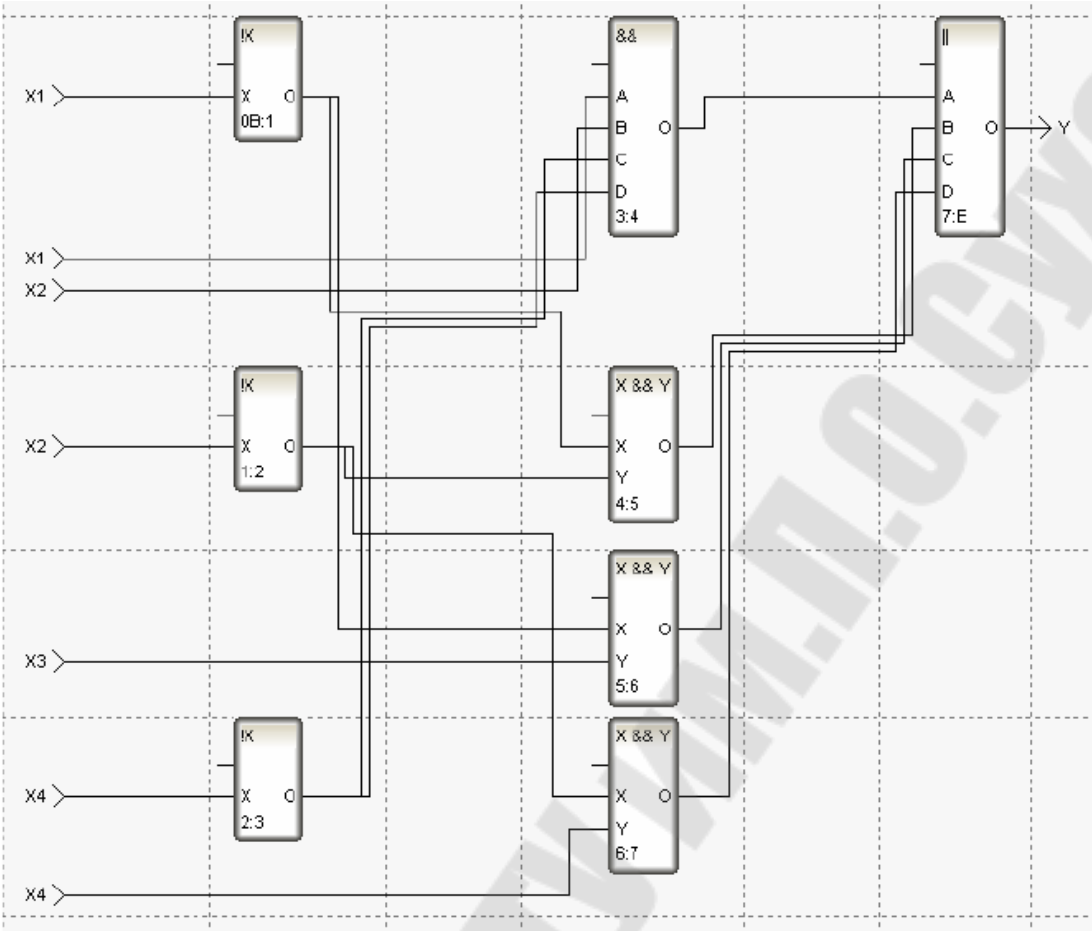


Рисунок 2.21 – Программа на языке **Техно FBD**

С помощью иконки на инструментальной панели редактора или «горячей клавиши» F7, скомпилировать программу и убедиться в успешной компиляции в окне «**Выход**» (Output), вызываемого из инструментальной панели с помощью иконки (рисунок 2.22).

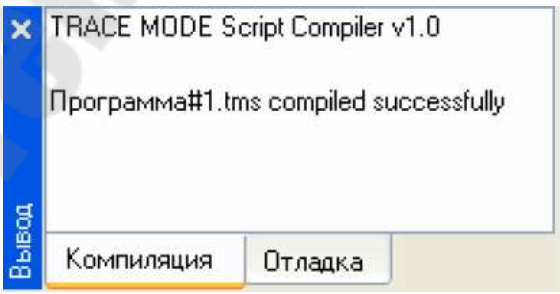


Рисунок 2.22 – Результат успешной компиляции программы

2.5 Привязка аргументов программы

Выполним привязку аргументов программы к атрибутам каналов. Для этого, вызвать свойства компонента **Программа#1** через контекстное меню. Выбрать вкладку «**Аргументы**». Двойным нажатием в поле «**Привязка**» привязать аргументы программы к атрибутам каналов – аргументы X1, X2, X3, X4 к **реальному значению** каналов X1, X2, X3, X4 (рисунок 2.23).

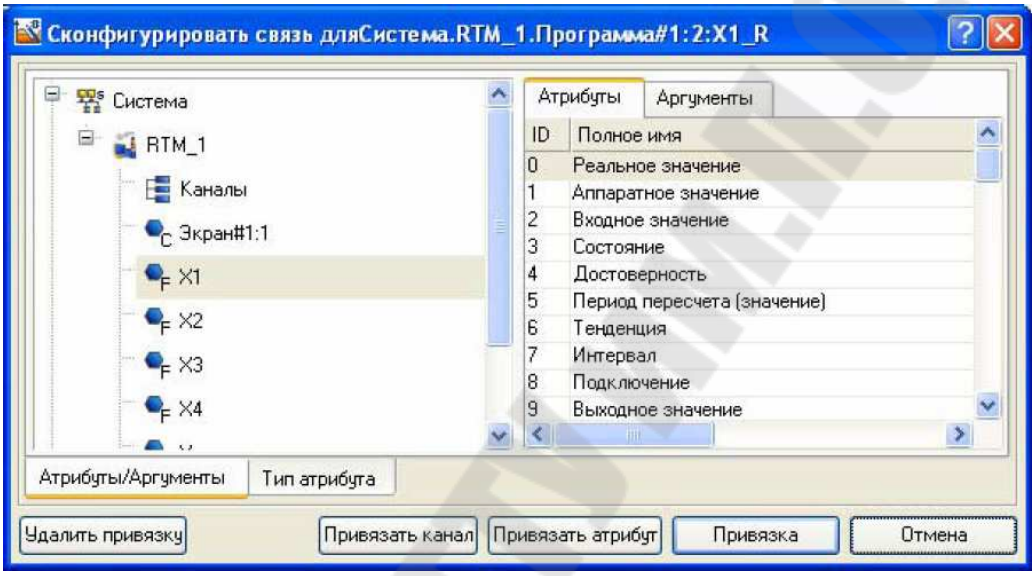


Рисунок 2.23 – Связь аргументов программы с аргументами экрана

Двойным щелчком в поле **Привязка** аргумента программы Y вызвать окно настройки связи, выбрать в левом окне канал класса «**Вызов**» **Экран#1**, а в правом окне выбрать вкладку «**Аргументы**» и указать в ней аргумент Y и кнопкой «**Привязка**» подтвердить связь (рисунок 2.24).

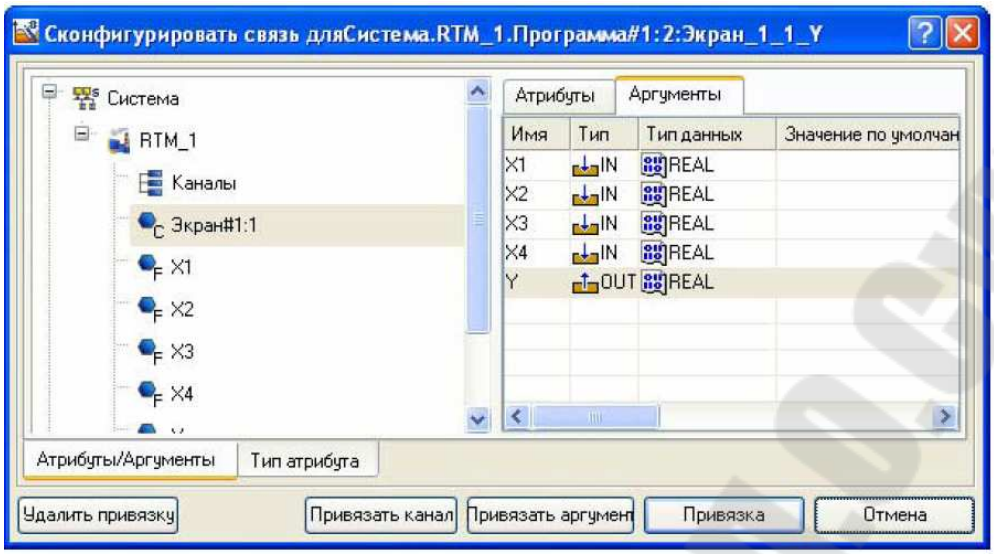


Рисунок 2.24 – Связь выходных аргументов

В результате, получим следующие привязки – рисунок 2.25.

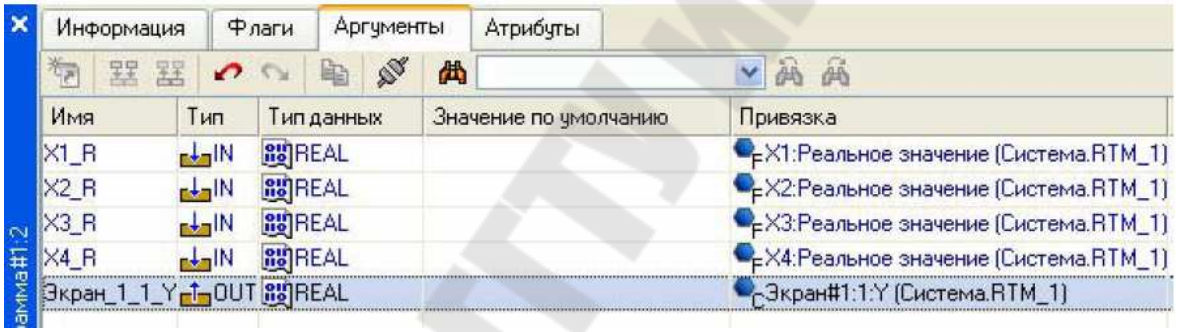


Рисунок 2.25 – Окончательная настройка связи

После закрыть окно свойств компонента **Программа#1.**

2.6 Запуск проекта

Сохранить проект с помощью кнопки . На инструментальной панели выберем иконку и подготовим проект для запуска в реальном времени. ЛК выделим в слое «**Система**» узел **RTM_1** (рисунок 2.26), а после, нажав ЛК иконку на инструментальной панели, запустим профайлер. Запуск/останов профайлера осуществляется с помощью иконки на его инструментальной панели или клавишной комбинации **Ctrl+R**. .

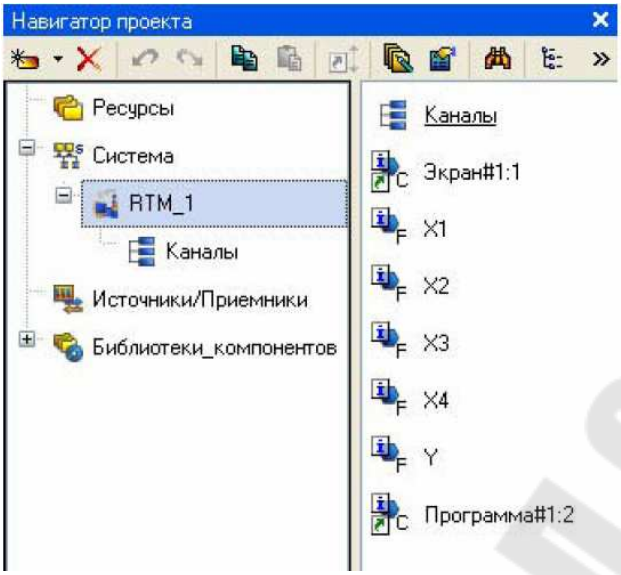


Рисунок 2.26 – Навигатор проекта

Для составления отчета необходимо в меню **Файл** выполнить команду «**Документировать проект»** полученный *.html файл необходимо внести в отчет.

3 Контрольные задания

Варианты заданий выбираются по порядковому номеру в списке группы.

1. $f = x_1x_2 \cup x_1\overline{x_3x_4} \cup \overline{x_1x_2x_3} \cup \overline{x_1}x_4$
2. $f = x_1\overline{x_4} \cup x_1\overline{x_2x_3} \cup \overline{x_1}x_2 \cup \overline{x_1}x_4$
3. $f = x_1\overline{x_2x_3} \cup \overline{x_1x_2}x_3 \cup x_4$
4. $f = \overline{x_1x_3} \cup \overline{x_1}x_4 \cup x_2\overline{x_3} \cup \overline{x_2}x_4$
5. $f = x_1\overline{x_3} \cup \overline{x_1x_2} \cup \overline{x_1x_4} \cup \overline{x_2x_4}$
6. $f = x_1x_4 \cup x_1x_2x_3 \cup \overline{x_1x_2x_4} \cup \overline{x_3}x_4 \cup \overline{x_2x_3}$
7. $f = x_1x_2 \cup x_1\overline{x_3} \cup \overline{x_1x_2}x_3 \cup x_2\overline{x_4} \cup x_3\overline{x_4}$
8. $f = \overline{x_1}x_3 \cup \overline{x_1}x_4 \cup x_2x_3 \cup \overline{x_2x_3x_4}$
9. $f = x_1x_4 \cup x_1x_2x_3 \cup \overline{x_1x_2x_4} \cup \overline{x_3}x_4 \cup \overline{x_2x_3}$
10. $f = x_1\overline{x_2x_3} \cup \overline{x_1}x_2 \cup \overline{x_1}x_3 \cup \overline{x_1}x_4 \cup x_2x_3x_4$

11. $f = x_1 x_2 x_4 \cup x_1 x_2 x_3 \cup x_1 x_3 \cup x_2 x_3 \cup x_1 x_2 x_3 \cup x_3 x_4$
12. $f = x_1 x_3 \cup x_1 x_4 \cup \overline{x_1} x_2 \overline{x_3} \cup \overline{x_1 x_3 x_4} \cup \overline{x_2} x_3 \overline{x_4}$
13. $f = x_1 x_3 \cup x_1 x_2 x_4 \cup \overline{x_1 x_2} x_3 \cup \overline{x_1 x_4}$
14. $f = x_1 \overline{x_2} x_3 \cup \overline{x_1} x_2 x_3 \cup x_1 x_2 \overline{x_3} \cup \overline{x_1 x_2 x_3} \cup x_3 x_4 \cup \overline{x_3 x_4}$
15. $f = \overline{x_1 x_3} \cup x_1 x_3 x_4 \cup \overline{x_1} x_2 \cup \overline{x_2 x_4} \cup \overline{x_3 x_4}$
16. $f = x_1 x_2 \overline{x_3} x_4 \cup x_1 x_3 \overline{x_4} \cup x_1 \overline{x_2} x_3 \cup \overline{x_1 x_2 x_3} \cup \overline{x_1 x_3 x_4} \cup \overline{x_1} x_3 x_4 \cup \overline{x_2 x_3 x_4}$
17. $f = x_1 x_3 \cup \overline{x_1} x_2 \overline{x_4} \cup \overline{x_1 x_2} x_4 \cup x_2 x_3 \cup x_3 x_4$
18. $f = \overline{x_1} x_3 \cup \overline{x_1 x_2} \cup \overline{x_1} x_4 \cup \overline{x_2 x_4} \cup x_2 \overline{x_3} x_4$
19. $f = x_1 x_3 \cup \overline{x_1 x_2 x_3} \cup x_2 \overline{x_3 x_4}$
20. $f = x_1 x_2 x_4 \cup x_1 x_3 x_4 \cup \overline{x_1 x_4} \cup \overline{x_1 x_2 x_3} \cup x_2 x_3 \overline{x_4}$
21. $f = x_1 x_3 \cup \overline{x_1 x_2 x_3} \cup \overline{x_1 x_3 x_4} \cup \overline{x_2 x_4}$
22. $f = x_1 x_2 x_4 \cup x_1 x_3 x_4 \cup \overline{x_1 x_2 x_3} \cup \overline{x_1 x_2 x_4} \cup x_2 x_3 x_4 \cup \overline{x_2 x_3 x_4}$
23. $f = x_1 x_2 x_3 \cup \overline{x_1 x_2} \cup \overline{x_1 x_4} \cup \overline{x_2 x_3 x_4} \cup x_3 \overline{x_4}$
24. $f = x_1 \overline{x_2} x_3 \cup x_1 x_3 \overline{x_4} \cup \overline{x_1} x_4 \cup x_2 \overline{x_3}$
25. $f = x_1 x_2 \overline{x_4} \cup \overline{x_1 x_2} \cup \overline{x_1} x_3 \cup \overline{x_2} x_4$

4 Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, программы, созданные каналы и привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции и **FBD** программы.

5 Контрольные вопросы

1. Языки МЭК 61131-3. Общие характеристики;
2. Принципы построения программ на языках МЭК 61131-3;
3. Язык FBD.
4. Порядок написания FBD-программ.
5. Понятие функционального блока.

Лабораторная работа №6
Реализация логического управления в SCADA TRACE MODE 6 с использованием языка Texno ST

Цель работы: освоить методику программирования логических функций при помощи SCADA–системы TRACE MODE на языке Техно ST.

1. Теоретические сведения

Язык ST (Structured Text, Структурированный Текст) является текстовым языком высокого уровня, имеющий черты языков Pascal и Basic. Язык адаптирован специально для программирования ПЛК. С помощью ST удобно реализовывать арифметические и логические операции (в том числе, побитовые), безусловные и условные переходы, циклические вычисления; возможно использование как библиотечных, так и пользовательских функций.

Этот язык предназначен в основном для выполнения сложных математических вычислений, описания сложных функций, функциональных блоков и программ.

Основным недостатком является сложность использования языка не подготовленным разработчиком и отсутствие наглядного представления о алгоритме программы в целом и происходящих в ней процессах.

1.1 Описание языка Texno ST

В алфавит языка входят:

1. прописные и строчные буквы латинского алфавита;
2. цифры;
3. специальные знаки: + - * / < = > ! : & | ^ ~ % ( ) [ ] , ; #.

Идентификаторы могут состоять из прописных и строчных букв латинского алфавита, знака подчеркивания «_», цифр. Идентификаторы не чувствительны к регистру.

Для данного языка характерны ключевые слова: *and, array, bool, break, by, byte, case, constant, continue, date, date_and_time, dint, do, dt, dword, else, elsif, endcase, endfor, endfunction, end_function_block, end_if, end_program, endrepeat, endstruct, endtype, endvar, endwhile,*

exit, false, for, function, functionblock, goto, handle, if, int, lreal, mod, not, of, or, program, real, repeat, return, rol, ror, shl, shr, sint, string, struct, time, time_of_day, to, tod, true, type, udint, uint, until, usint, var, var_arg, var_global, var_inout, var_input, var_output, while, word, xor.

Строчный комментарий начинается с «//» и продолжается до конца строки. Блочный комментарий начинается с «/*» и продолжается до «*/».

Для описания структуры программы и операторов в приняты следующие терминологические соглашения:

- **выражение** – последовательность операндов, разделителей и символьных операторов, задающая вычисление без присвоения результата;
- **предложение** – последовательность лексем, определяющая выполнение логически законченного промежуточного действия. Таким действием может быть присвоение переменной результата вычислений, вызов функции-блока и т.п. Операторы (кроме символьных) также образуют предложения.

На основании этих соглашений программа или ее компонент на языке Техно ST определяется как последовательность предложений.

Каждое предложение должно **завершаться точкой с запятой**. Исключением из этого правила являются операторы определения переменных, для завершения которых точка с запятой не используется.

Длина строки программы не ограничивается, лексемы разделяются произвольным числом пробелов, знаков табуляции или символов перевода строки.

Основная точка входа в программу определяется следующей конструкцией:

program
    {определение аргументов}
    {список предложений}
end_program

Необязательное выражение {*определение аргументов*} задается аналогично выражению {*определение переменной*} для операторов определения переменной. В дальнейшем конструкция **program…end_program** называется основной программой.

Функции, глобальные переменные и структурные типы не могут быть определены в основной программе.

Переменные можно задать, заполняя таблицу аргументов, локальных и глобальных переменных. Переменные определяются в разделе описания аргументов **автоматически** в соответствии с заполненными таблицами аргументов и переменных.

Язык Texno ST позволяет создавать числовые и строковые константы. Рассмотрим их представление. Целочисленная десятичная константа начинается с цифры, отличной от нуля, после которой располагаются любые цифры. Можно привести следующие примеры целочисленных десятичных констант: 123, 350, 498.

Двоичная целочисленная константа начинается с префикса «2#», после которого приводится двоичное представление целого числа. Примеры:

2#1011, 2#0111,2#1001.

Восьмеричные целочисленные константы начинаются с префикса «8#», после которого записывается восьмеричное представление числа. Примеры:

8#145,8#0277,8#756.

Шестнадцатеричные целочисленные константы начинаются с префикса «16#», после которого приводятся шестнадцатеричные представления чисел. При записи шестнадцатеричного представления числа можно использовать как строчные символы a.. .f, так и прописные A... F. Примеры:

16#149, 16#A145E, 16#a145e.

Вещественная константа состоит из целой и дробной части. Допустимо наличие только целочисленной или дробной части. Примеры: .123, 0.456, 489. . Возможно представление в формате с плавающей точкой (используется префикс e или E с указанием порядка). Примеры:

1.23E-6, 6.7504E4, 6.798e-5.

Частным случаем числовой константы является временной интервал, дата, время дня. Временной интервал записывается в виде:

t#<дни>d<часы>h<минуты>m<секунды>s<миллисекунды>ms

Возможна также запись в виде:

time#<дни>d<часы>h<минуты>m<секунды>s<миллисекунды>ms

Любая составляющая в приведенных представлениях временного интервала может быть опущена.

Пример: Временной интервал равной 2 часам, 31 минута, 25 секундам и 10 миллисекунд может быть записан как

t#2h31m25s10ms или в виде **time#2h31m25s10ms.**

Дата записывается в виде d#**<год>-<месяц>-<день>**, возможна также запись в виде **date#<год>-<месяц>-<день>**.

Пример: 25 сентября 2001 года может быть записано в виде:

d#2001-9-2001 или date#2001-9-2001.

Время дня можно записать в формате:

tod#<час>:<минута>:<секунда> или

time_of_day#<час>:<минута>:<секунда>.

Пример: Время 19 часов 15 минут 42 секунды может быть записано как:

tod#19:15:42, так и time_of_day#19:15:42.

Константа «дата и время» может быть записана как:

dt#<год>-<месяц>-<день>-<час>:<минута>:<секунда>, так и date_and_time#<год>-<месяц>-<день>-<час>:<минута>:<секунда>.

Пример: 12 февраля 1995 года 13 часов 47 минут и 13 секунд можно записать в виде:

dt#1995-2-12-13:47:13 или date_and_time# 1995-2-12-13:47:13.

Помимо числовых констант часто используются строковые константы, которые представляют собой набор символов заключенных в одинарные или двойные кавычки.

Пример: "Первая строка символов", Вторая строка символов'.

В строках не могут присутствовать управляющие символы, кавычки и символ $. Для размещения в строке произвольного символа, включая управляющие, используется механизм эскейп-последовательностей. Данный механизм позволяет разместить в строке следующие последовательности:

– $r— возврат каретки;
– $n— перевод строки;

– $t— табуляция;
– $uXXXX— UNICODE символ, где XXXX— шестнадцатеричный символ;
– $ x— символ x («x»— любой символ).

Рассмотрим символьные операторы. Под символьными операторами понимают знаки операций, выполняемых над операндами. В качестве операндов могут выступать:

1. имена констант;
2. имена переменных;
3. имена массивов с указанием индекса отдельного элемента;
4. вызов пользовательских функций;
5. вызов библиотечных функций;
6. выражения, заключенные в скобки;
7. имена элементов структур.

Арифметические операторы приведены в таблице 1.1, побитовые – в таблице 1.2, операторы сравнения – в таблице 1.3, логические – в таблице 1.4.

Таблица 1.1 – Арифметические операторы

Оператор	**Действие**
Унарный «-»	Смена знака
Унарный «+»	Пустая операция
«+»	Сложение чисел или конкатенация строк
«-»	Вычитание
«*»	Умножение
«/»	Деление
«%»	Получение остатка от деления
«**»	Возведение в степень

Таблица 1.2 – Побитовые операторы

Оператор	**Действие**
«&»	Побитовое «И»
«|»	Побитовое «ИЛИ»
«^» или xor	Побитовое «исключающее ИЛИ»
Унарная «-»	Поразрядная инверсия
«<<» или shl	Сдвиг влево на указанное число разрядов
«>>» или shr	Сдвиг вправо на указанное число разрядов

rol	Циклический сдвиг влево на указанное число разрядов
ror	Циклический сдвиг вправо на указанное число разрядов

Таблица 1.3 – Операторы сравнения

Оператор	**Проверяемое условие**
«==»	Равенство
«!=» или «<>»	Неравенство
«<»	Меньше
«>»	Больше
«<=»	Меньше или равно
«>=»	Больше или равно

Таблица 1.4 – Логические операторы

Оператор	**Действие**
«&&» или and	Логическое «И»
«||» или or	Логическое «ИЛИ»
«!» или not	Логическое отрицание

Оператор присваивания позволяет произвести присваивание значения переменной. Есть два синтаксиса оператора присваивания:

{ операнд} = { выражение}

{ операнд} := { выражение}

В качестве операнда может использоваться имя переменной, массива, уточненное имя переменной объекта. В таблице 1.5 приводится приоритет символьных операторов. Первыми будут выполняться выражения, заключенные в скобках, а затем в порядке возрастания номера строки в таблице.

Таблица 1.5 – Приоритет символьных операторов

Приоритет	**Операция**
1	**
2	!, not, -, унарный -, унарный +
3	<<, >>, shl, shr, rol, ror
4	&, |, A, xor
5	*, /, %, mod

6	+, -
7	== != <> < <= > >=
8	&&, and
9	||, or
10	

В ST – программе можно использовать стандартные функции языка Си: sin, cos, tan, asin, exp, log. Помимо символьных операторов в языке имеются и управляющие:

1. return
2. if
3. case
4. while
5. repeat
6. for
7. break
8. exit
9. continue
10. операторы определения переменных
11. операторы индексирования элементов массива
12. got.

Для разветвления алгоритма используется оператор if. Данный оператор всегда начинается с ключевого слова if и заканчивается ключевым словом end_if Существует три варианта задания данного оператора. Первый вариант:

if {выражение} then {последовательность предложений};
end_if;

В данном варианте последовательность предложений выполняется только в том случае, если выражение истинно. Второй вариант задания оператора if имеет вид:

if { выражение} then { последовательность предложений 1};
else { последовательность предложений 2} ;
end_if;

Во втором варианте задания оператора if проверяется выражение. Если оно истинно, то выполняется последовательность предложений 1, в противном случае— последовательность предложений 2. Третий вариант оператора if имеет вид:

if { выражение 1} then { последовательность предложений 1};
elseif { выражение 2} then { последовательность предложений 2};
…
elseif { выражение N} then { последовательность предложений N};
else { последовательность предложений N+1};
end_if;

В последнем варианте оператора if выполняется i-ая последовательность предложений в том случае, если i-ое выражение истинно. Если все выражения ложны, то выполняется последовательность предложений, которая идет после ключевого слова else.

Язык Texno ST содержит оператор выбора case. Оператор начинается с ключевого слова case и заканчивается ключевым словом end_case. Первый вариант оператора case можно представить следующим образом:

case { выражение} of
{ список значений}:{ последовательность предложений};
…
{ список значений}:{ последовательность предложений};
end_case;

В данном случае вычисляется выражение, производится поиск результата вычисления в списках значений. Выполняется та последовательность предложений, в списках значений которой найден результат вычислений.

Второй вариант оператора case можно представить следующим образом:

case { выражение} of
{ список значений}:{ последовательность предложений};
…
{ список значений}:{ последовательность предложений};
else { последовательность предложений};
end_case;

Отличие второй формы записи оператора case от первой заключается в том, что если результат вычисления выражения не найден ни в одном из списков значений, то выполняется последовательность предложений после ключевого слова else. В первом случае при отсут-

ствии результата вычислений в списках значений приведенные в операторе последовательности предложений не выполняются.

В обоих представлениях оператора case список значений – набор целых чисел или диапазонов целых чисел, разделенных запятой. Диапазон указывается в виде:

{нижняя граница диапазона} .. {верхняя граница диапазона}

Язык Texno ST позволяет создавать циклы используя операторы while, repeat, for. Синтаксис оператора while имеет вид:

while { выражение} do
{ последовательность предложений};
end_while;

В данном операторе последовательность выражений выполняется пока выражение истинно. Каждый раз перед выполнением последовательности предложений производится проверка выражения.

Оператор repeat можно представить в виде:

repeat
{ последовательность предложений};
until { выражение} end_repeat;

В операторе repeat последовательность выполняется, проверяется выражение, если оно истинно, то последовательность выражений повторяется снова, в противном случае происходит выход из цикла.

Цикл for можно представить в виде:

for {имя переменной} := {выражение 1} to {выражение 2} by {выражение 3} do
{последовательность предложений};

end_for;

Данный оператор сначала присваивает переменной цикла с указанным именем результат вычисления выражения 1, выполняет последовательность предложений, если вычисленная переменная цикла не превысит значение выражения 2. Затем к переменной цикла прибавляется выражение 3, выполняется последовательность предложений, если вычисленное значение не превышает выражение 2. Выполнение последовательности предложений и увеличение значения переменной на выражение 3 повторяется до тех пор, пока значение пере-

менной цикла не превышает выражения 2. Для цикла for характерно то, что он не позволяет создавать цикл с отрицательным шагом.

Операторы break и exit позволяют выйти из текущего цикла. Оператор continue служит для перехода в конец цикла. При его вызове все следующие за ним до конца цикла предложения не выполняются.

1.2 Создание программы на языке Texno ST

Для создания программы, необходимо двойным щелчком ЛК открыть узел **RTM_1** и создать в нем компонент «**Программа» (**рисунок 1.1).

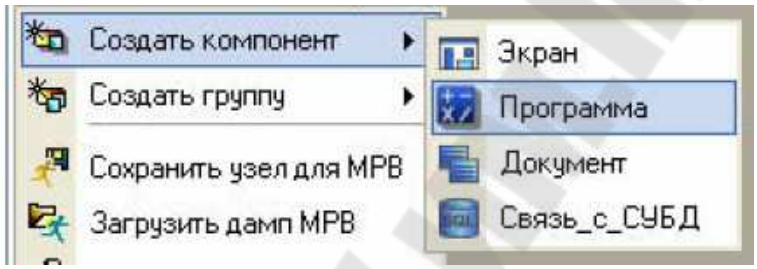


Рисунок 1.1 – Создание компонента Программа

Выделить компонент Программа#1 и ПК мыши вызвать контекстное меню, выбрав в котором ЛК мыши, пункт «**Редактировать шаблон»**, перейти в режим редактирования программы (рисунок 1.2).

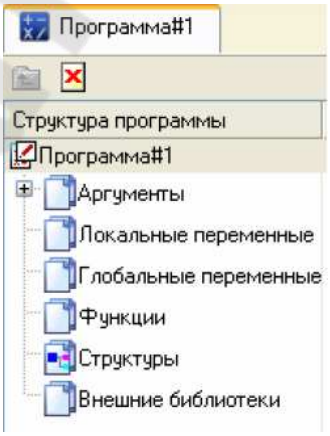
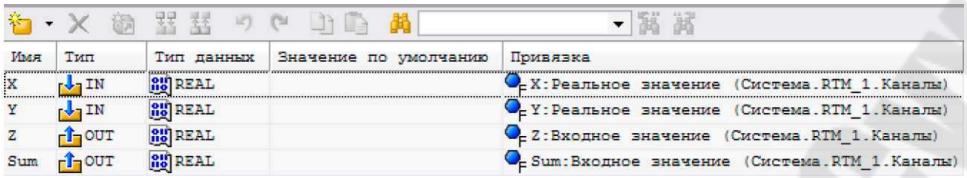


Рисунок 1.2 – Редактирование аргументов «**Программы»**

Выделением ЛК мыши в дереве шаблона **Программа#1** строку «**Аргументы**», вызвать табличный редактор аргументов. Кнопкой создать в редакторе аргументов четыре аргумента X1, X2, X3, X4

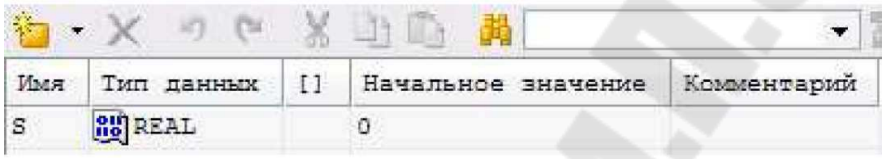
и выход Y. При этом первые 4 аргумента должны быть типа **IN**, а последний – **OUT (**рисунок 1.3).



Имя	Тип	Тип данных	Значение по умолчанию	Привязка
X	IN	REAL		X:Реальное значение (Система.RTM_1.Каналы)
Y	IN	REAL		Y:Реальное значение (Система.RTM_1.Каналы)
Z	OUT	REAL		Z:Входное значение (Система.RTM_1.Каналы)
Sum	OUT	REAL		Sum:Входное значение (Система.RTM_1.Каналы)

Рисунок 1.3 – Аргументы программы

Аналогично задаются и глобальные переменные/



Имя	Тип данных	[]	Начальное значение	Комментарий
S	REAL		0	

Рисунок 1.4 – Глобальные переменные программы

Выделить в дереве шаблона строку **Программа#1** и в открывшемся диалоге «**Выбор языка**» выбрать язык **ST (**рисунок 1.5).

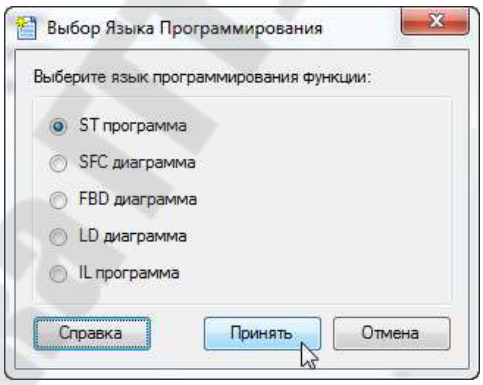


Рисунок 1.5 – Выбор языка программирования

После выбора языка откроется окно редактора, изображенное на рис. 1.6. Все созданные аргументы, и локальные переменные будут описаны в начале программы. **Глобальные переменные** в отличие от локальных, в самой программе не описываются, но они могут использоваться как операнды.

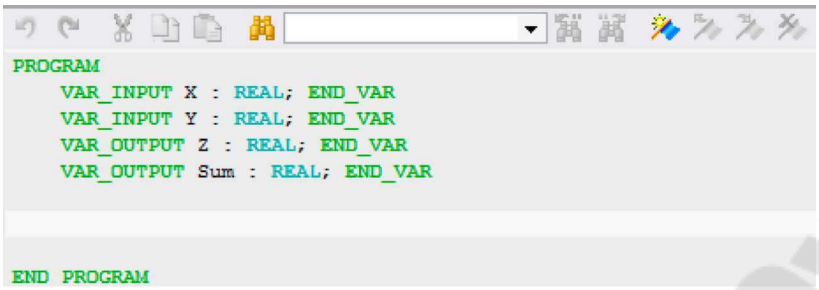


Рисунок 1.6 – Окно редактора для создания программы

Составим программу вычисления деления аргумента X на аргумент Y, произведя проверку на равенство Y нулю, результат присвоим аргументу Z. Аргументу Sum присваивается сумма всех результатов деления. Листинг программы:

```
PROGRAM
    VARINPUT X : REAL; ENDVAR
    VARINPUT Y : REAL; ENDVAR
    VAR_OUTPUT Z : REAL; END_VAR
    VARINOUT Sum : REAL; ENDVAR
    if Y == 0 then Z = X / 1e-9;
    else Z = X / Y;
    end_if;
    Sum = S+ Z;      // накапливаем сумму
    S = Sum;   //сохраняем ее в глобальной переменной до следующего вызова программы
END_PROGRAM
```

После того, как написали текст программы необходимо проверить ее. С помощью иконки  на инструментальной панели редактора или «горячей клавиши» F7, скомпилировать программу и убедиться в успешной компиляции в окне «**Выход**» (Output), вызываемого из инструментальной панели с помощью иконки  .

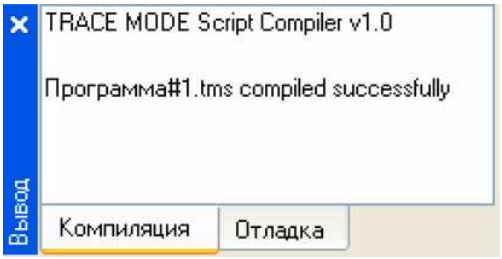


Рисунок 1.7 – Результат успешной компиляции программы

2 Ход работы

2.1 Задание

На основе проекта, реализации логической функции на языке Техно FBD, выполнить реализацию той же функции на языке Техно ST, следующими способами.

1. Арифметическими операторами;
2. Побитовыми операторами;

2.2 Варианты заданий

Варианты заданий выбираются по номеру в списке группы.

1. $f = x_1x_2 \cup x_1\overline{x_3x_4} \cup \overline{x_1x_2x_3} \cup \overline{x_1}x_4$
2. $f = x_1\overline{x_4} \cup x_1\overline{x_2x_3} \cup \overline{x_1}x_2 \cup \overline{x_1}x_4$
3. $f = x_1\overline{x_2x_3} \cup \overline{x_1x_2}x_3 \cup x_4$
4. $f = \overline{x_1x_3} \cup \overline{x_1}x_4 \cup x_2\overline{x_3} \cup \overline{x_2}x_4$
5. $f = x_1\overline{x_3} \cup \overline{x_1x_2} \cup \overline{x_1x_4} \cup \overline{x_2x_4}$
6. $f = x_1x_4 \cup x_1x_2x_3 \cup \overline{x_1x_2x_4} \cup \overline{x_3}x_4 \cup \overline{x_2x_3}$
7. $f = x_1x_2 \cup x_1\overline{x_3} \cup \overline{x_1x_2}x_3 \cup x_2\overline{x_4} \cup x_3\overline{x_4}$
8. $f = \overline{x_1}x_3 \cup \overline{x_1}x_4 \cup x_2x_3 \cup \overline{x_2x_3x_4}$
9. $f = x_1x_4 \cup x_1x_2x_3 \cup \overline{x_1x_2x_4} \cup \overline{x_3}x_4 \cup \overline{x_2x_3}$
10. $f = x_1\overline{x_2x_3} \cup \overline{x_1}x_2 \cup \overline{x_1}x_3 \cup \overline{x_1}x_4 \cup x_2x_3x_4$
11. $f = x_1x_2x_4 \cup x_1x_2x_3 \cup x_1x_3 \cup x_2x_3 \cup x_1x_2x_3 \cup x_3x_4$
12. $f = x_1x_3 \cup x_1x_4 \cup \overline{x_1}x_2\overline{x_3} \cup \overline{x_1x_3x_4} \cup \overline{x_2}x_3\overline{x_4}$
13. $f = x_1x_3 \cup x_1x_2x_4 \cup \overline{x_1x_2}x_3 \cup \overline{x_1x_4}$
14. $f = x_1\overline{x_2}x_3 \cup \overline{x_1}x_2x_3 \cup x_1x_2\overline{x_3} \cup \overline{x_1x_2x_3} \cup x_3x_4 \cup \overline{x_3x_4}$
15. $f = \overline{x_1x_3} \cup x_1x_3x_4 \cup \overline{x_1}x_2 \cup \overline{x_2x_4} \cup \overline{x_3x_4}$
16. $f = x_1x_2\overline{x_3}x_4 \cup x_1x_3\overline{x_4} \cup x_1\overline{x_2}x_3 \cup \overline{x_1x_2x_3} \cup \overline{x_1x_3x_4} \cup \overline{x_1}x_3x_4 \cup \overline{x_2x_3x_4}$
17. $f = x_1x_3 \cup \overline{x_1}x_2\overline{x_4} \cup \overline{x_1x_2}x_4 \cup x_2x_3 \cup x_3x_4$

18. $f = \overline{x_1}x_3 \cup \overline{x_1x_2} \cup \overline{x_1}x_4 \cup \overline{x_2x_4} \cup x_2\overline{x_3}x_4$

19. $f = x_1x_3 \cup \overline{x_1x_2x_3} \cup x_2\overline{x_3x_4}$

20. $f = x_1x_2x_4 \cup x_1x_3x_4 \cup \overline{x_1x_4} \cup \overline{x_1x_2x_3} \cup x_2x_3\overline{x_4}$

21. $f = x_1x_3 \cup \overline{x_1x_2x_3} \cup \overline{x_1x_3x_4} \cup \overline{x_2x_4}$

22. $f = x_1x_2x_4 \cup x_1x_3x_4 \cup \overline{x_1x_2x_3} \cup \overline{x_1x_2x_4} \cup x_2x_3x_4 \cup \overline{x_2x_3x_4}$

23. $f = x_1x_2x_3 \cup \overline{x_1x_2} \cup \overline{x_1x_4} \cup \overline{x_2x_3x_4} \cup x_3\overline{x_4}$

24. $f = x_1\overline{x_2}x_3 \cup x_1x_3\overline{x_4} \cup \overline{x_1}x_4 \cup x_2\overline{x_3}$

25. $f = x_1x_2\overline{x_4} \cup \overline{x_1x_2} \cup \overline{x_1}x_3 \cup \overline{x_2}x_4$

3 Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, программы, созданные каналы и привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции и **ST** программы.

4 Контрольные вопросы

1. Языки МЭК 61131-3. Общие характеристики;
2. Принципы построения программ на языках МЭК 61131-3;
3. Язык ST. Описание языка.
4. Язык ST. Структура программы.

Лабораторная работа № 7
Реализация системы автоматического регулирования при помощи в SCADA TRACE MODE 6 с использованием языка Texno FBD

Цель: освоить методику программирования системы автоматического регулирования при помощи SCADA-системы TRACE MODE на языке Техно FBD

1. Теоретические сведения

Для операций моделирования управления объектами необходимо использование блоков задающих (описывающих) объект управления. В TRACE MODE для этого используется блок **OBJ** (рисунок 1.1).

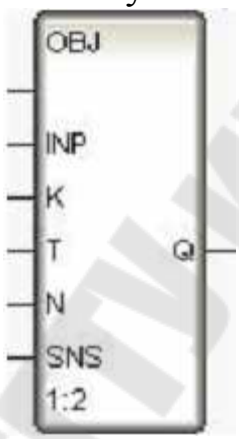


Рисунок 1.1 – FBD блок – модель объекта (OBJ)

Данный блок моделирует объект управления для отладки алгоритмов регулирования или подготовки демонстрационных проектов. Он представляет собой комбинацию периодического (инерционного) звена первого порядка и звена запаздывания. Передаточная функция блока имеет вид:

$$\Phi_a(s) = \frac{k}{Ts+1} e^{-ls},$$

где

- **k** и **T** – соответственно коэффициент усиления и постоянная времени инерционного звена первого порядка;
- **l** > 0 – время запаздывания.

Кроме того, на выходной сигнал блока можно наложить помеху в виде случайной составляющей, синусоидального сигнала или случай-

ных бросков. Здесь же можно задать случайное колебание динамических характеристик объекта. Входным по отношению к моделируемому объекту является вход **INP**. Входы **K**, **T** и **N** используются для задания соответственно коэффициента усиления, постоянной времени и времени запаздывания. Последние два параметра задаются в тактах пересчета, максимальное значение времени запаздывания – 4. Вход **SNS** предназначен для управления случайными помехами, вносимыми в работу объекта. Значение отдельных битов этого входа включает следующие помехи:

- 1 бит – добавление к выходному сигналу случайной величины в диапазоне от 0 до 1%;
- 2 бит – формирование пика величиной 25% от значения выхода с вероятностью 0,01;
- 3 бит – добавление к выходу синусоидального сигнала с амплитудой 2% от значения выхода;
- 5 бит – случайное увеличение коэффициента усиления в диапазоне от 0 до 2%;
- 6 бит – случайное увеличение постоянной времени в диапазоне от 0 до 2%;
- 7 бит – случайное изменение на 1 запаздывание.

Первые три помехи добавляются к выходу блока после формирования его нового значения. Динамические характеристики объекта (последние три помехи) корректируются до пересчета блока.

Для реализации регулирующих функций в TRACE MODE реализованы типовые алгоритмы управления. Наиболее распространенным алгоритмом автоматического управления является ПИД регулирование. В TRACE MODE для этого используется FBD блок – ПИД регулятора **PID** (рисунок 1.2).

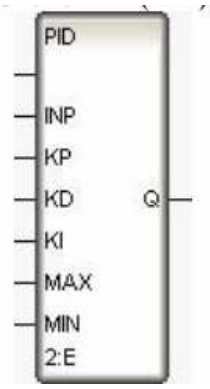


Рисунок 1.2 – FBD блок – ПИД регулятор

Этот блок формирует выходное значение по ПИД – закону от величины, поданной на вход **INP**:

$$Q_i = KP \times INP_i + \frac{KD \times (INP_i - INP_{i-1})}{\Delta t} + KI \times \Delta t \times \sum_{k=1}^{i} INP_k$$

где

i – текущий такт пересчета; **KP, KD** и **KI** – соответственно коэффициенты при пропорциональной, дифференциальной и интегральной составляющих; **Δt** – период пересчета блока в секундах (длительность такта).

Модуль подаваемого на вход **KI** отрицательного значения передается на выход. При подаче на вход **KI** неотрицательного значения регулирование начинается с установленной величины. Для ограничения величины управляющего воздействия используются входы блока **MIN и MAX.** Если величина управления меньше **MIN**, то **Q** = **MIN**, если величина управления больше **MAX**, то **Q** = **MAX**, при этом в обоих случаях накопление интегральной составляющей закона регулирования прекращается. Данный блок вычисляет величину управляющего воздействия по значению рассогласования регулируемой величины и задания, которое предварительно надо вычислять с помощью блока **X-Y**. Введение в алгоритм параметра *Δt* исключает необходимость пересчета настроек регулятора при смене периода пересчета.

2 Ход работы

2.1 Создание проекта Trace Mode

Рассмотрим создание системы автоматического регулирования при помощи SCADA-системы TRACE MODE на языке Техно FBD. Необходимо разработать систему с ПИД регулированием технологическим объектом, с заданной передаточной функцией (параметры по варианту):

$$W(s) = \frac{K \cdot e^{-\tau \cdot s}}{T \cdot s + 1}.$$

где К – коэффициент усиления объекта, Т – постоянная времени; τ – время запаздывания.

Разработка любого проекта автоматизации всегда начинается с запуска Интегрированной среды разработки (ИСР).

После запуска ИСР в меню «**Файл**» выбрать команду «**Настройки ИС**…». В появившемся окне в закладке «**Уровень сложности**» настроить проект как показано на рисунке 2.1, а затем выбрать закладку «**Отладка**» и настроить проект как показано на рисунке 2.2.

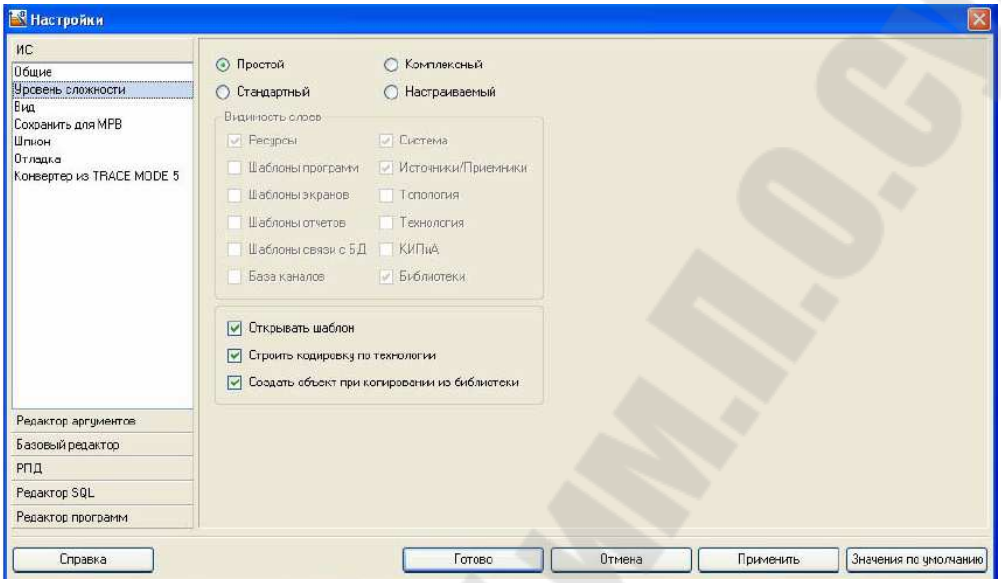


Рисунок 2.1 – Настройки «**Уровень сложности»**

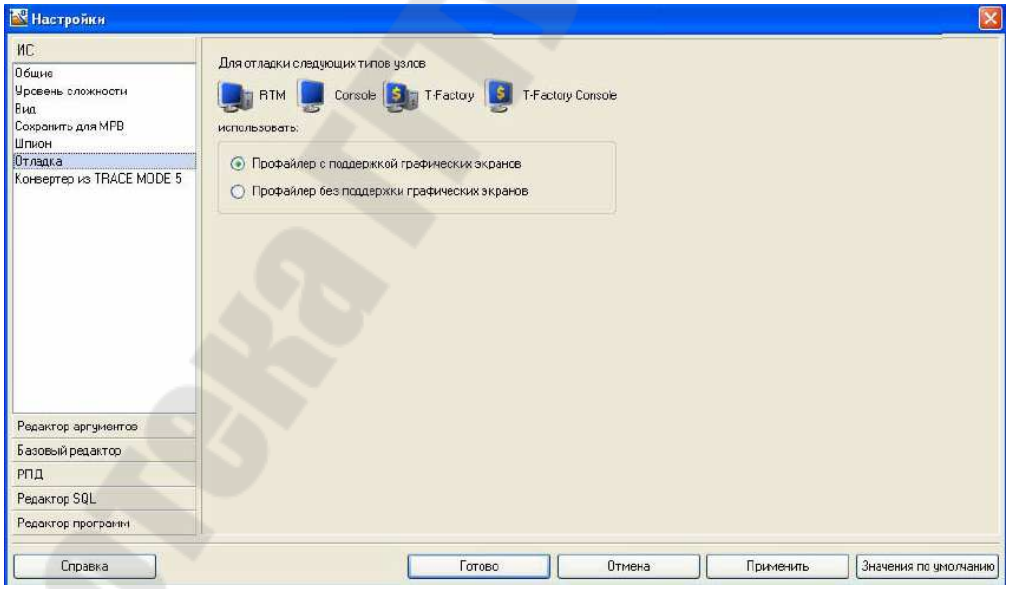


Рисунок 2.2 – Настройки «**Отладка»**

После проведенных настроек ИСР нажать кнопку «**Готово**». С помощью иконки  инструментальной панели создадим новый проект при этом в открывшемся на экране диалоге (рисунок 1.3).

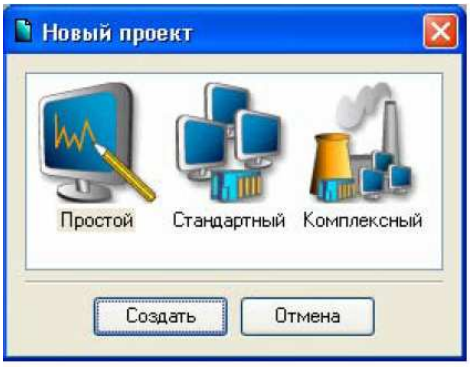


Рисунок 2.3 – Выбор типа проекта

Выберем «**Простой**» стиль разработки. После нажатия левой клавиши мыши (ЛК) на экранной кнопке «**Создать**», в левом окне Навигатора проекта появится дерево проекта с созданным узлом **АРМ RTM_1**. Откроем узел **RTM_1** двойным щелчком ЛК, в правом окне «**Навигатора проекта**» отобразится содержимое узла – пустая группа «**Каналы**» и один канал класса «Вызов» **Экран#1**, предназначенный для отображения на узле **АРМ** графического экрана (рисунок 2.4).

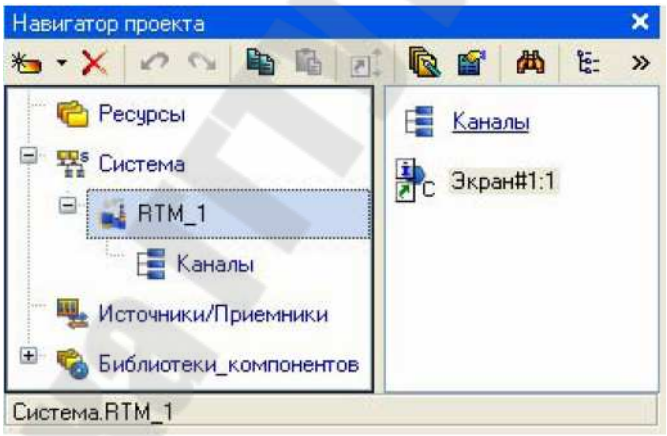


Рисунок 2.4 – Навигатор проекта

2.2 Создание графического интерфейса

Двойным щелчком ЛК на компоненте Экран#1 открываем окно графического редактора. Подготовим на экране вывод динамического текста для отображения численного значения входных переменных апериодического звена первого порядка: коэффициента усиления объекта (Kob), постоянной времени объекта (Tob), времени запаздывания (Tau); ПИД регулятора: коэффициент усиления регулятора (Кг), времени интегрирования (Ti), времени дифференцирования (Td); и величины задания (Zad).

Выберем на инструментальной панели графического редактора иконку ГЭ «Кнопка» -  С помощью мыши разместим семь элементов в поле экрана как показано на рисунке 2.5.

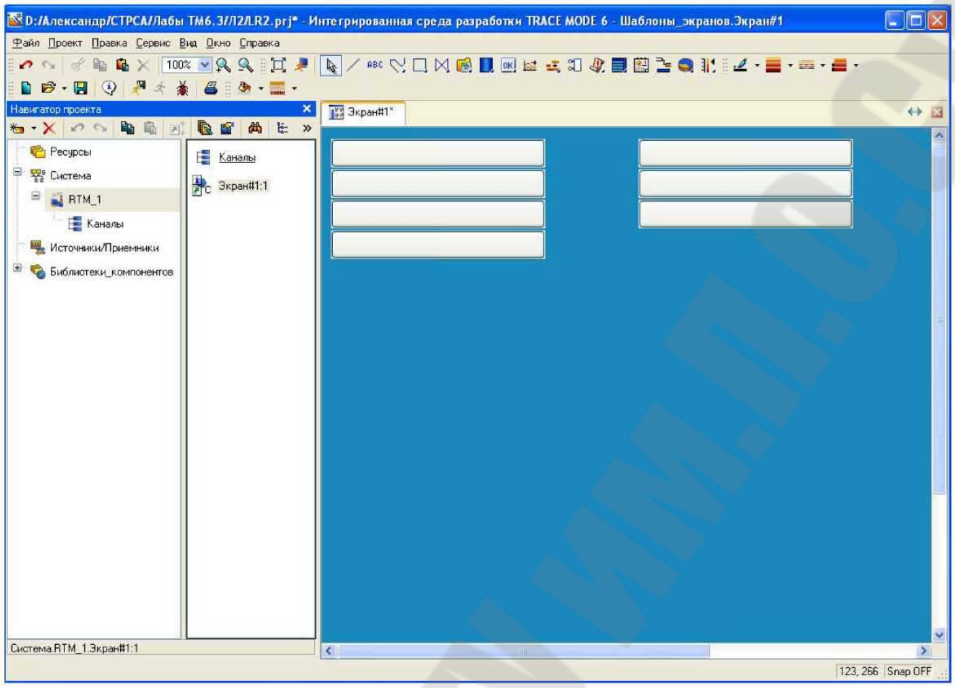


Рисунок 2.5 – Вид экрана оператора

Перейти в режим редактирования нажатием  кнопки, затем двойным щелчком ЛК вызвать окно свойств первого ГЭ  рисунок 2.6.

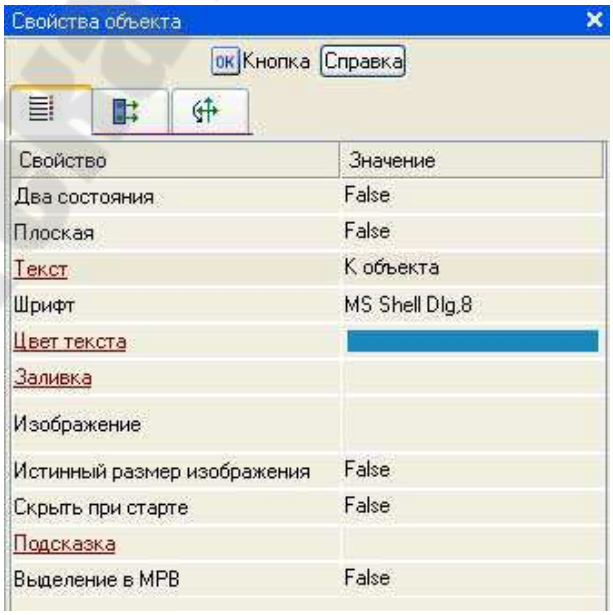


Рисунок 2.6 – Свойства графического элемента «**Кнопка**»

В поле «**Текст**» ввести «**К объекта**». Открыть закладку «**Действия**»  и (правой кнопкой мыши) ПК раскрыть меню «**События По нажатию (mousePressed)**». Выбрать из списка команду «**Добавить Передать значение**», раскрыть меню настроек выбранной команды рисунок 2.7.

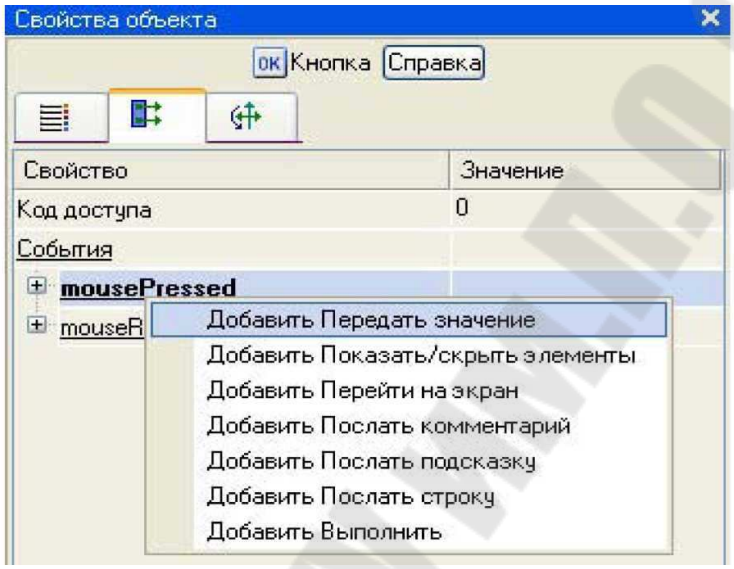


Рисунок 2.7 – Настройка типа передачи

В поле «**Тип передачи (Send Type)**» выбрать из списка «**Ввести и передать (Enter & Send)**» (рисунок 2.8).

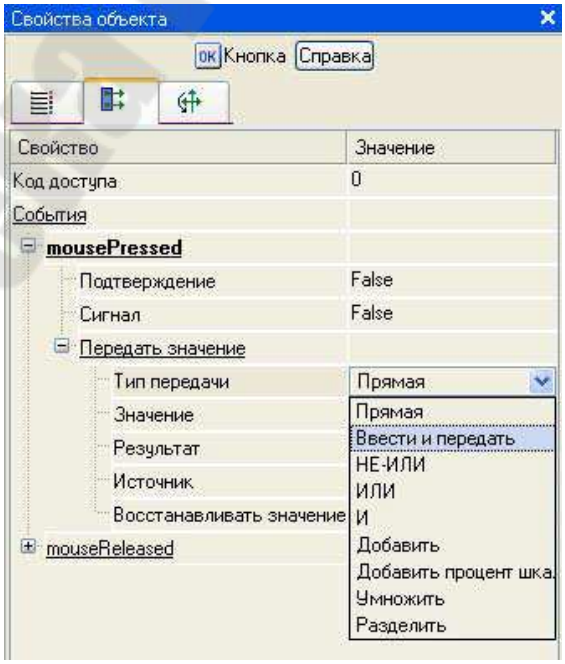


Рисунок 2.8 – Настройка типа передачи

ЛК в поле «**Результат**» вызвать табличный редактор аргументов. В открывшемся окне «**Свойство привязки**», нажав кнопку на его панели инструментов, создать восемь аргументов экрана 7 входных и один выходной. «**Входной (IN)**» или «**Выходной (OUT)**» будет аргумент, выбирается из низпадающего меню, появляющегося при двойном щелчке ЛК по типу аргумента (рисунок 2.9). Двойным щелчком ЛК выделить имя аргумента и изменить его, введя с клавиатуры «**Kob**» (завершить ввод нажатием клавиши Enter). Подтвердить связь с этим аргументом нажатием кнопки «**Готово**» Аналогичным образом настроить все остальные аргументы.

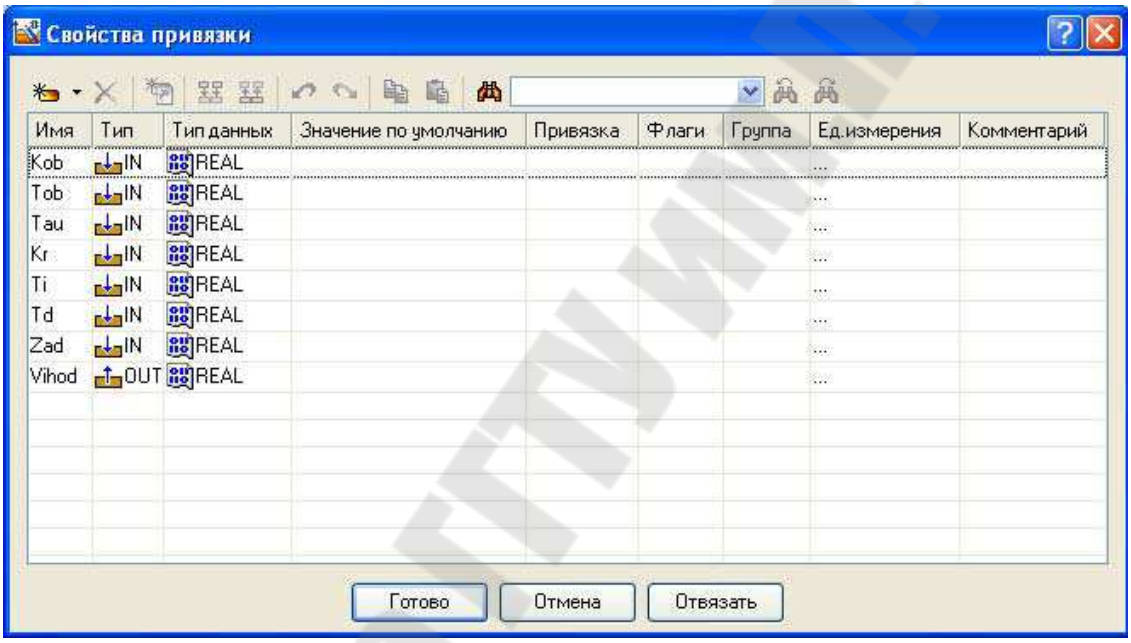


Рисунок 2.9 – Создание аргументов экрана

После привязки окно «**Свойства объекта**» будет выглядеть следующим образом – рисунок 2.10.

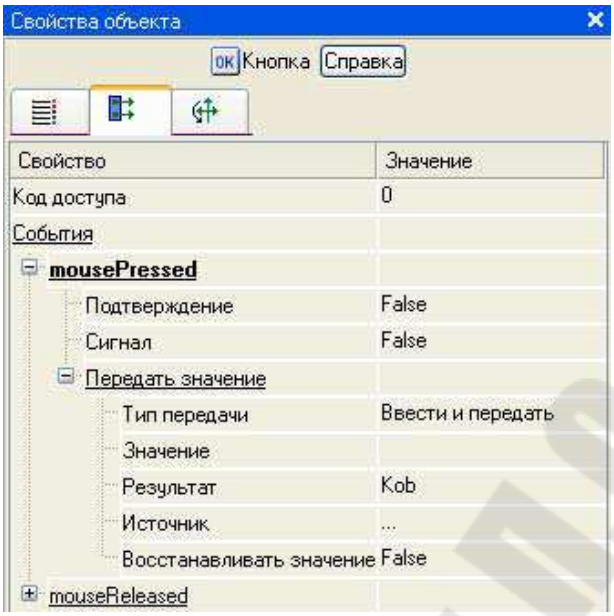


Рисунок 2.10 – Окончательный вид окна свойств графического элемента

Аналогичным образом настроить остальные кнопки Tob, Tau, Kr, Ti, Td, Zad.

Для отображения числового значения входных аргументов создадим и разместим семь ГЭ «Текст» ABC, как показано на рисунке 2.11. Для чего, активировав ГЭ ABC нажатием ЛК, рисуем области вывода динамического текста, задавая левый верхний и правый нижний углы щелчком ЛК.

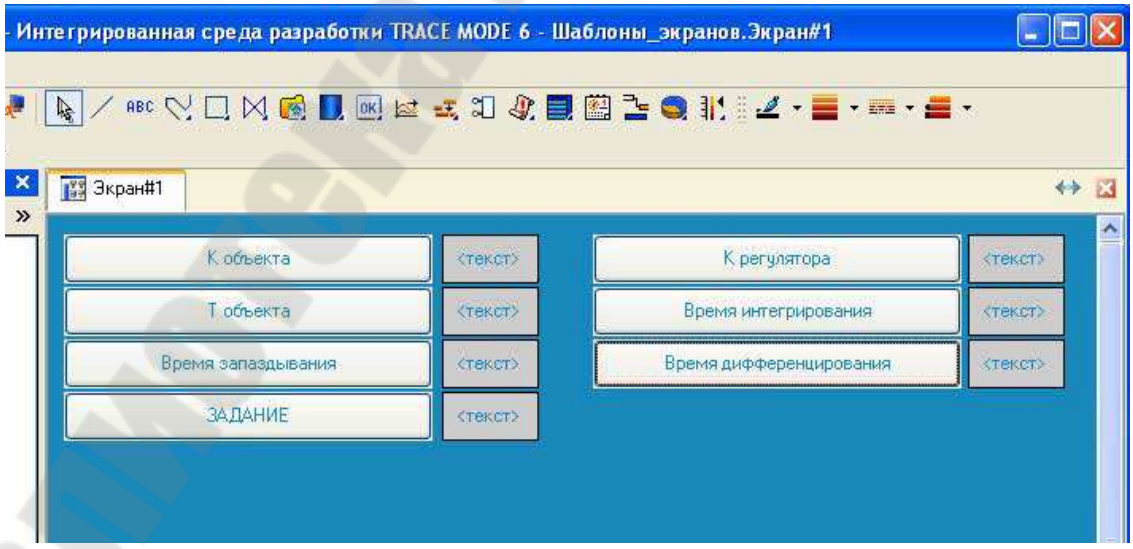


Рисунок 2.11 – Создание текстовых графических элементов

После этого необходимо переключиться в режим редактирования активацией ГЭ  . Двойным щелком ЛК на строке «**Текст**» вызвать меню «**Вид индикации**» (рисунок 2.12.)

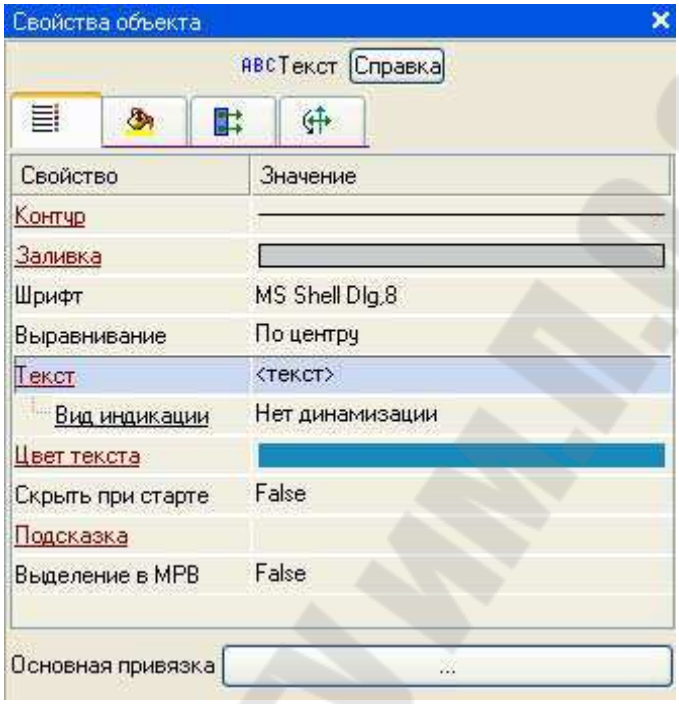


Рисунок 2.12 – Настройка индикации

В окне «**Свойства объекта**», двойным щелчком ЛК по свойству «**Текст**» раскрыть его элементы. В появившемся подпункте «**Вид индикации**» ,щелкнув по значению «**Нет динамизации**» выбрать тип «**Значение**» (рисунок 2.13).

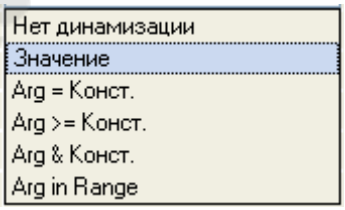


Рисунок 2.13 – Выбор типа динамизации

В открывшемся меню настройки параметров динамизации, выбрать свойство «**Привязка**» и связать с **Kob** (рисунок 2.14).

Рисунок 2.14 – Настройка привязки

Дополним созданный экран новым ГЭ для совместного просмотра изменений значений каналов узла во времени и отслеживании предыстории – трендом.

На экрана разместим ГЭ «**Тренд**» для вывода значений **Zad** и **Vihod** (рисунок 2.15). Основные свойства ГЭ оставим заданными по умолчанию.

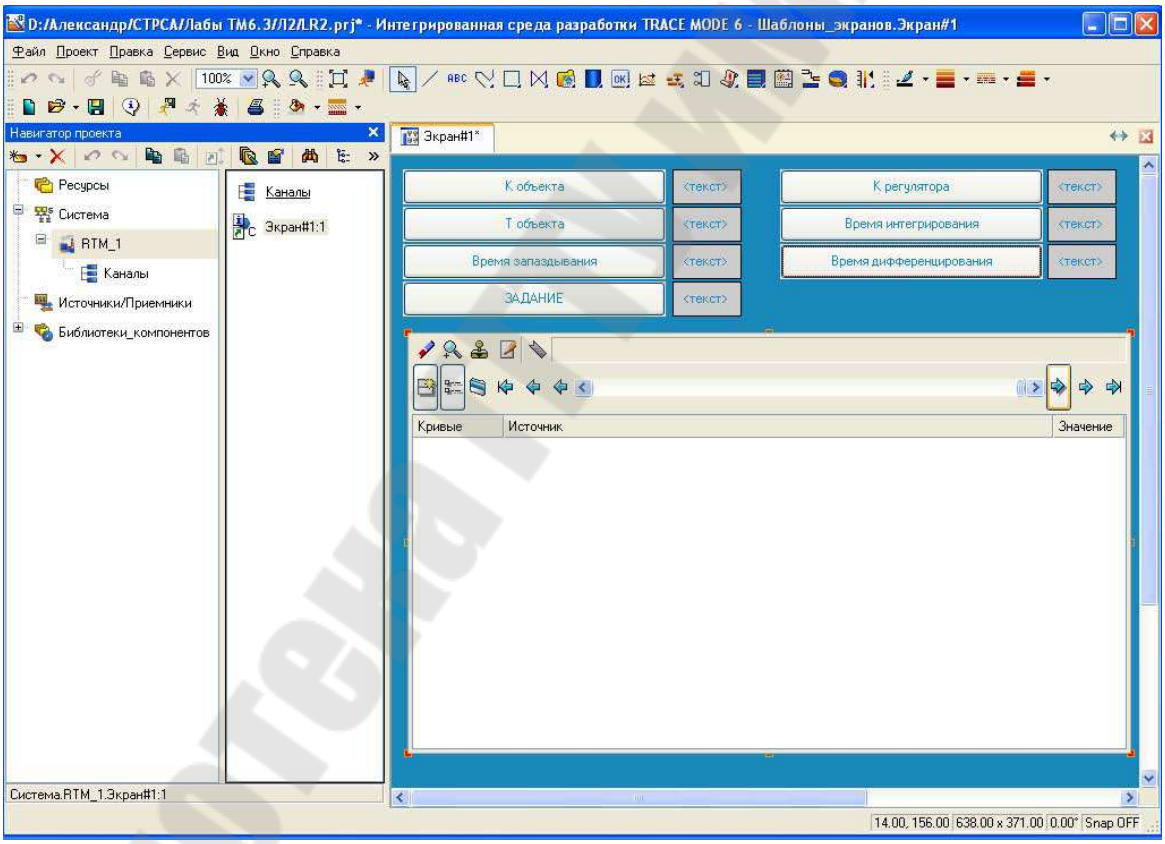


Рисунок 2.15 – Размещение тренда

Перейдем во вкладку и, выделив ЛК строку «**Кривые**», с помощью ПК создадим две новые кривые. Настроим их привязки к аргументам, толщину и цвет линий. Двойным щелчком ЛК мыши в поле тренда открываем окно «**Свойства тренда**» и переходим на вкладку «**Кривые**». Нажимая ПК мыши в поле «**Кривые**» задаем две кривые

Zad и **Vihod**, а в поле привязка привязываем созданные кривые к заданию и выходу как показано на рисунке 2.16.

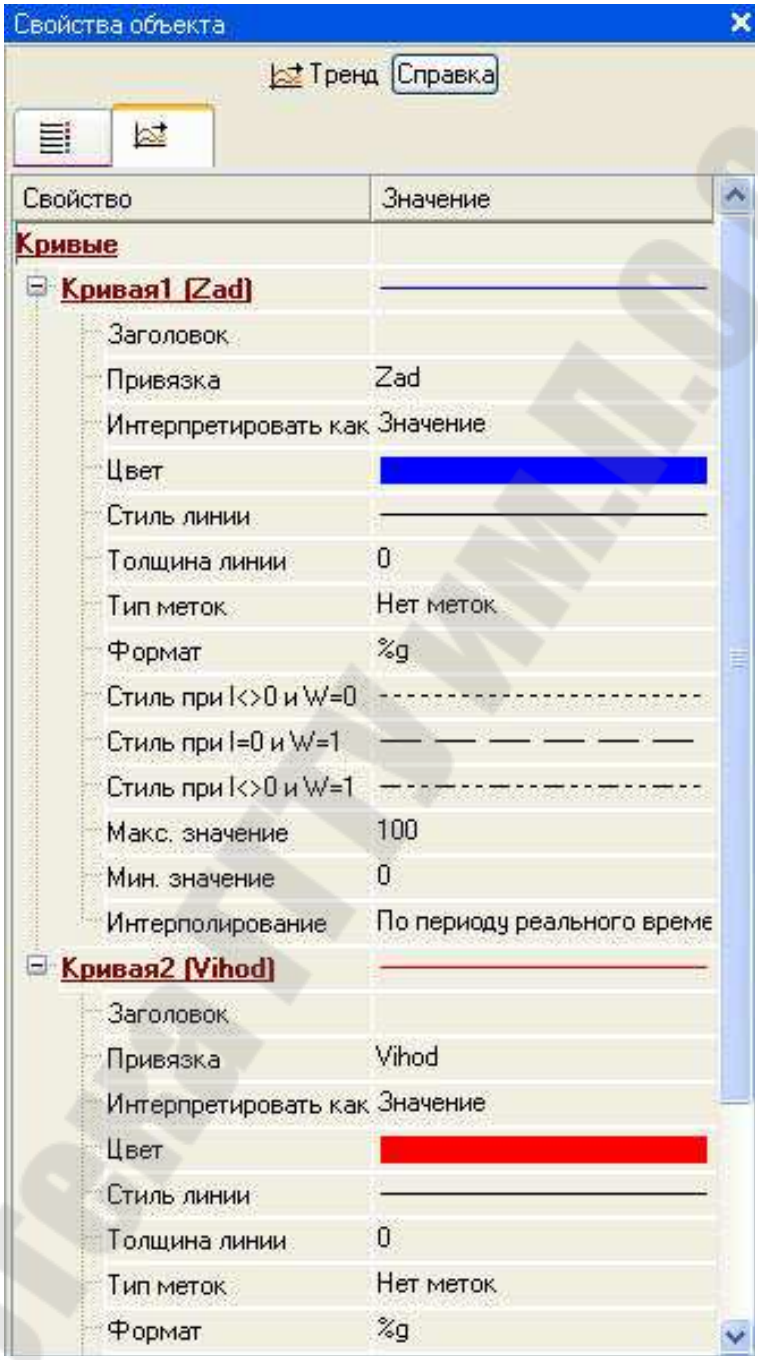


Рисунок 2.16 – Свойства тренда

2.3 Привязка аргумента экрана к каналу

Создадим по аргументам Kob, Tob, Tau, Kr, Ti, Td, Zad, Vihod, шаблона экрана новые каналы и отредактируем их привязку. В слое «**Система»** открыть узел **RTM**_1. С помощью ПК вызвать контекст-

ное меню свойства компонента **Экран**#1 и выбрать команду «**Свойства**» (рисунок 2.17). Выбрать вкладку «**Аргументы**», ЛК, при нажатой клавише Cntrl, выделить аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad, Vihod и с помощью иконки  создать новые каналы.

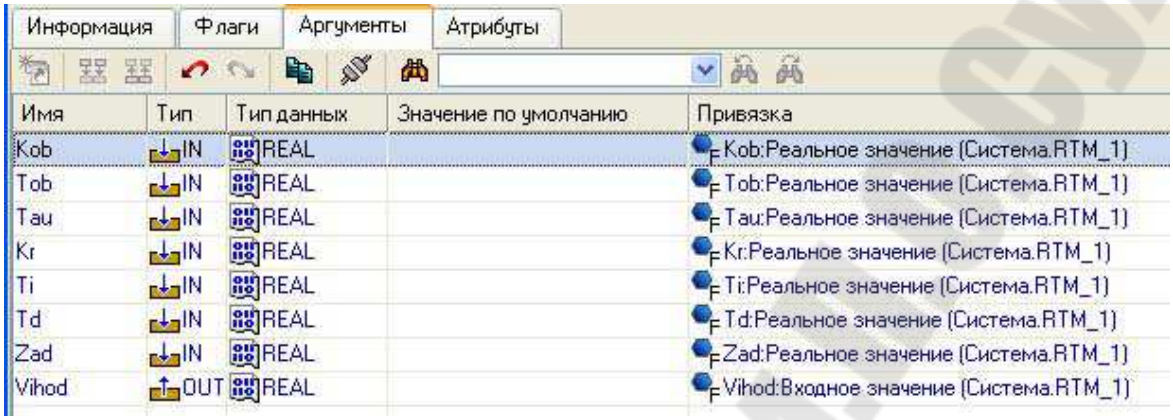


Рисунок 2.17 – Окно свойств экрана

В результате, в узле **RTM_1**, будут автопостроены следующие каналы – рисунок 2.18.

Рисунок 2.18 – Автопостроеные каналы

2.4 Создание программы на языке Техно FBD

Для создания FBD-программы и подключения ее к проекту нужно выполнить следующие операции:

1. разместить необходимые функциональные блоки в рабочем поле FBD-редактора;

2. соединить нужные входы и выходы блоков, образовав единую диаграмму;
3. задать аргументы, переменные и константы программы;
4. привязать входы/выходы FBD-диаграммы к аргументам, переменным и константам программы;
5. скомпилировать программу

Создадим программу, которая будет реализовывать одноконтурную систему автоматического регулирования. Для этого ПК по узлу RTM_1 вызвать контекстное меню выбрать подменю «**Создать компонент**» и выбрать ЛК в нем компонент «**Программа**» (рисунок 2.19).

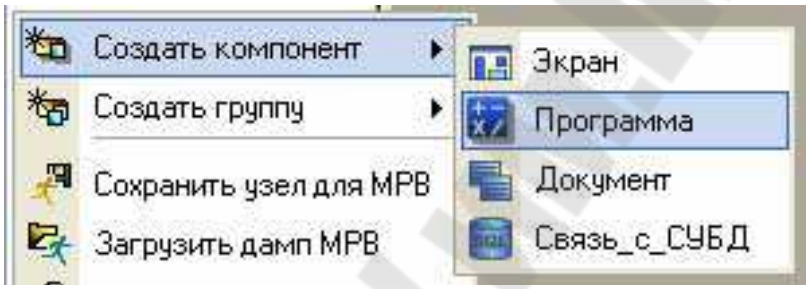


Рисунок 2.19 – Создание компонента «Программа»

Выделить компонент **Программа**#1 и ПК вызвать контекстное меню, выбрав в котором ЛК пункт «**Редактировать шаблон**», перейти в режим редактирования программы (рисунок 2.20).

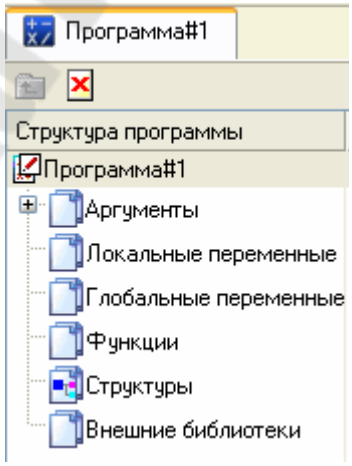


Рисунок 2.20 – Редактирование аргументов Программы

Выделением ЛК в дереве шаблона «**Программа**#1» строки «**Аргументы**» вызвать табличный редактор аргументов кнопкой  и создать в редакторе аргументов следующие аргументы программы

Kob, Tob, Tau, Kr, Ti, Td, Zad,. При этом аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad должны быть типа **IN**, а Vihod – **OUT** (рисунок 2.21).

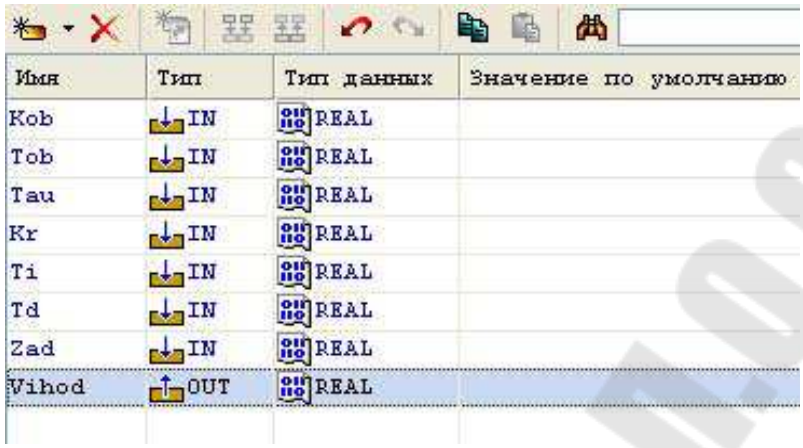


Рисунок 2.21 – Аргументы программы

Выделить в дереве шаблона строку **Программа#1** и в открывшемся диалоге «**Выбор языка**» выбрать язык **FBD (**рисунок 2.22).

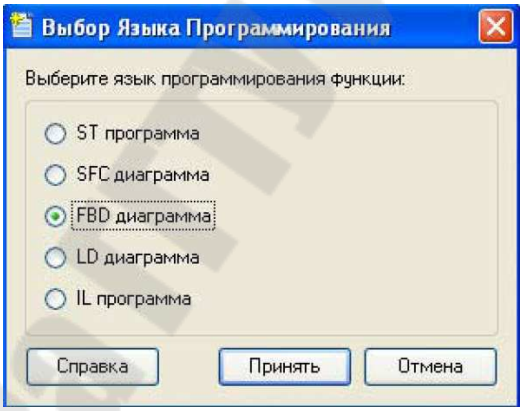


Рисунок 2.22 – Выбор языка программирования

По нажатию кнопки «**Принять**» в открывшемся окне редактора программ с объявленными переменными создать программу в соответствии с заданием (см. раздел «**Контрольное задание**»). Для выбора палитры FBD блоков необходимо ЛК мыши нажать на кнопку после чего появится окно выбора FBD блоков (рисунок 2.23). При разработки программы верхние входы FBD блоков не используются т.к. они предназначены для изменения порядка пересчета блоков, а информационными входами, являются входы начиная со второго.

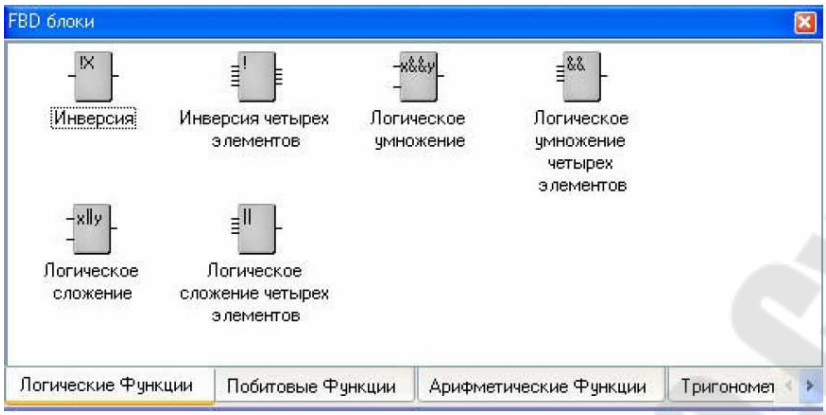


Рисунок 2.23 – Палитра FBD блоков

Для создания системы управления выберем следующие блоки: из раздела «**Арифметические Функции**» FBD блок **вычитание** (**X–Y**); из раздела «**Регулирование**» FBD блок **модель объекта** (**OBJ**) и звено **PID** (**PID**). После размещения всех блоков программа будет выглядеть следующим образом рисунок 2.24.

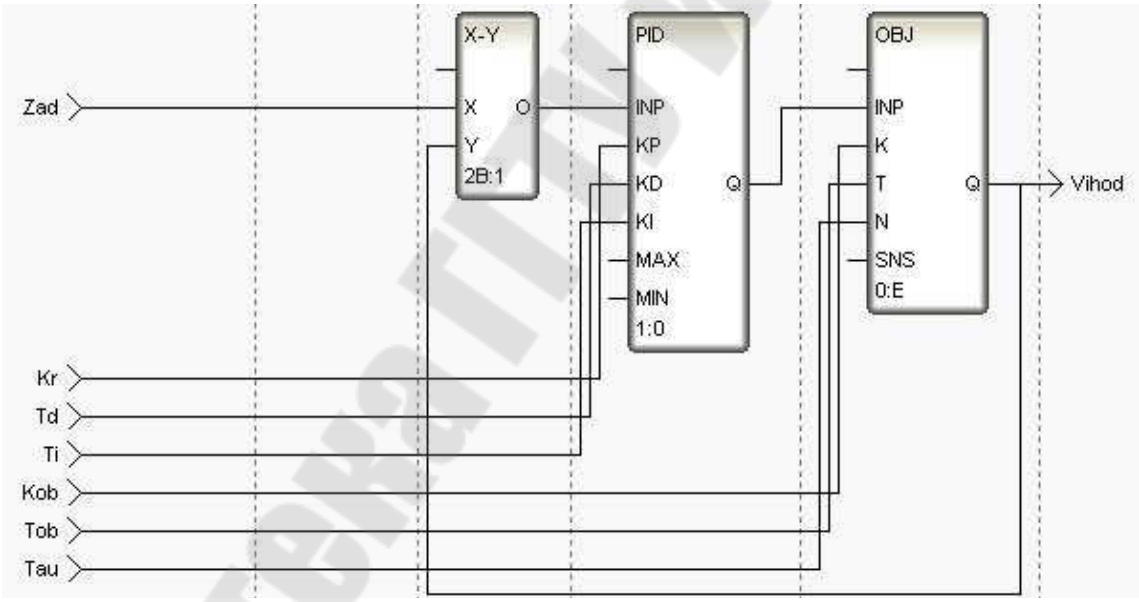


Рисунок 2.24 – Программа реализации системы управления

С помощью иконки  на инструментальной панели редактора или «горячей клавиши» F7, скомпилировать программу и убедиться в успешной компиляции в окне «**Выход**» (Output), вызываемого из инструментальной панели с помощью иконки  (рисунок 2.25).

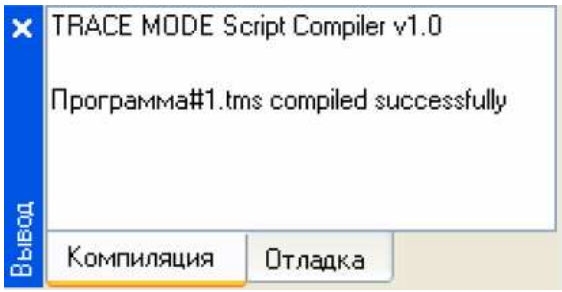


Рисунок 2.25 – Результат успешной компиляции программы

2.5 Привязка аргументов программы

Выполним привязку аргументов программы к атрибутам каналов. Вызвать свойства компонента **Программа**#1 через контекстное меню. Выбрать вкладку «**Аргументы»**.

Двойным щелчком в поле «**Привязка**» аргумента программы Y вызвать окно настройки связи, выбрать в левом окне канал класса «**Вызов**» **Экран**#1, а в правом окне выбрать вкладку «**Аргументы**» и указать в ней аргумент **Vihod** и кнопкой «**Привязка**» подтвердить связь (рисунок 2.26).

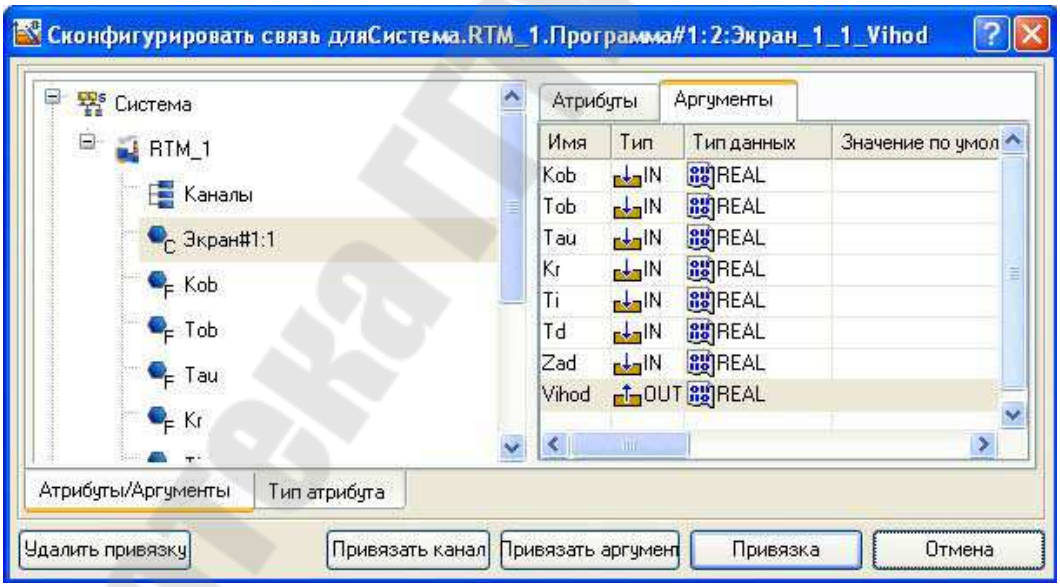


Рисунок 2.26 – Связь выходных аргументов

Аналогично в поле «**Привязка**» привязать аргументы программы к атрибутам каналов – аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad к реальному значению каналов Kob, Tob, Tau, Kr, Ti, Td, Zad (рисунок 2.27).

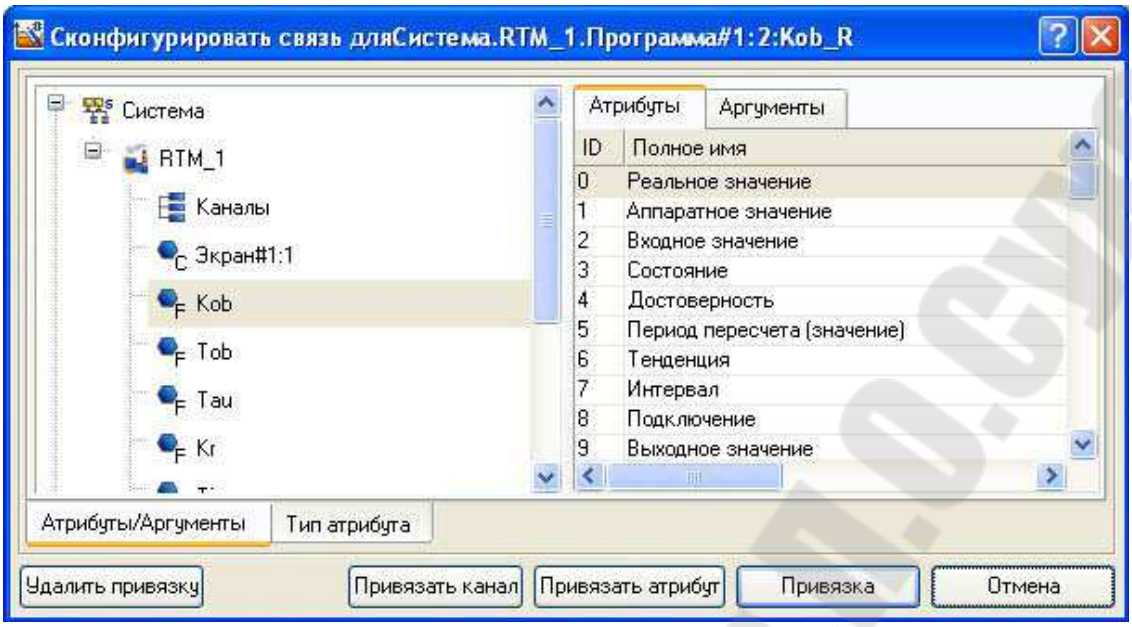


Рисунок 2.27 – Связь аргументов программы с аргументами экрана

В результате, получим следующие привязки – рисунок 2.28

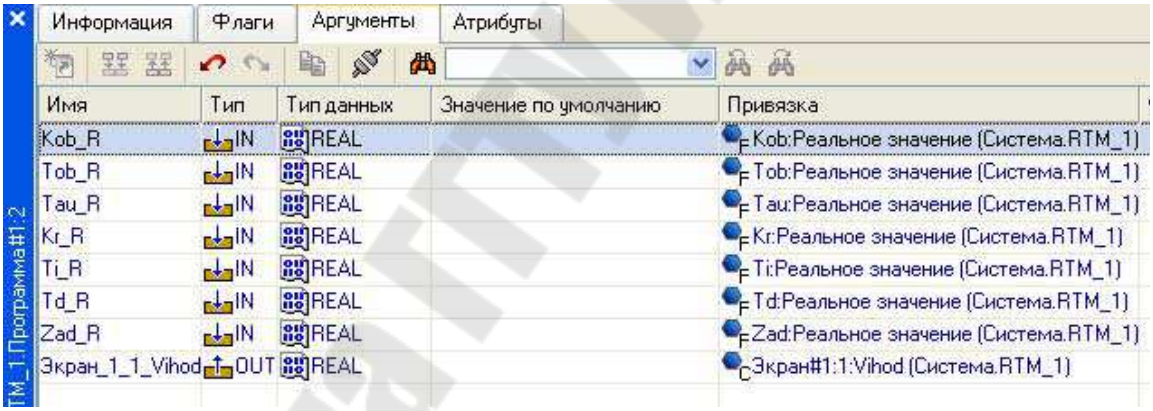


Рисунок 2.28 – Окончательная настройка связи

После закрыть окно свойств компонента **Программа#1.**

2.6 Запуск проекта

Сохранить проект с помощью кнопки . На инструментальной панели выберем иконку и подготовим проект для запуска в реальном времени. ЛК выделим в слое «**Система**» узел **RTM_1** (рисунок 2.29), а после, нажав ЛК иконку на инструментальной панели, запустим профайлер. Запуск/останов профайлера осуществляется с по-

мощью иконки  на его инструментальной панели или клавишной комбинации **Ctrl+R**. .

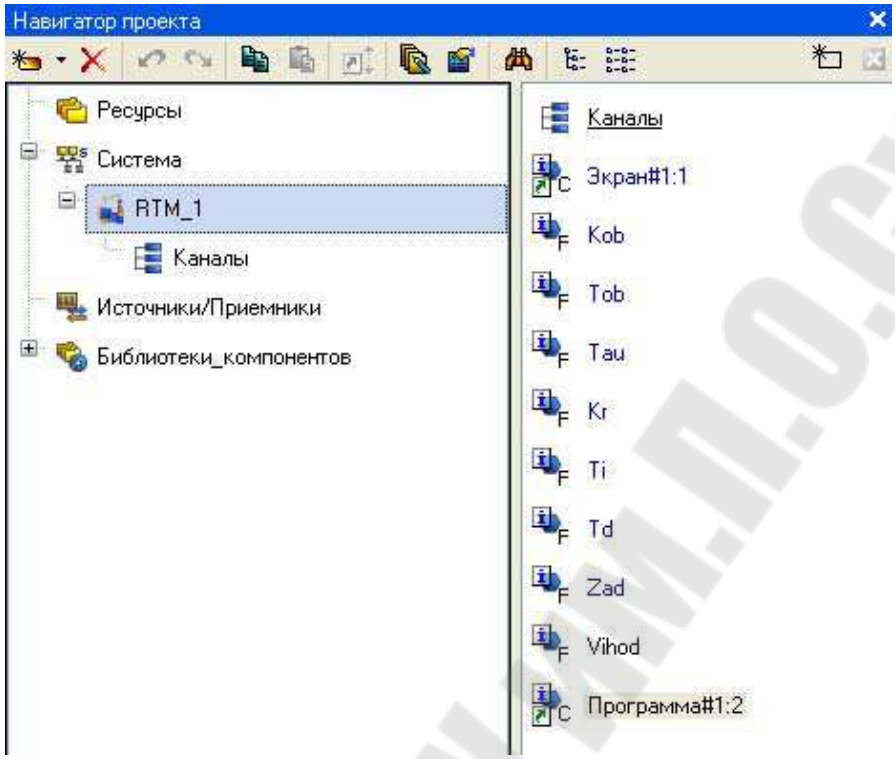


Рисунок 2.29 – Навигатор проекта

Для составления отчета необходимо в меню **Файл** выполнить команду «**Документировать проект»** полученный *.html файл необходимо внести в отчет.

3 Контрольные задания

Варианты заданий выбираются по порядковому номеру в списке группы.

Вариант задания выбирается исходя из номера буквы в алфавите по инициалам студента. Номер первой буквы **Фамилии** в алфавите – К коэффициент усиления объекта, **Имени** – Т постоянная времени объекта, **Отчества** – τ время запаздывания.

$$W(s) = \frac{K \cdot e^{-\tau \cdot s}}{T \cdot s + 1}.$$

Расчет настроек ПИД регулятора производится по формулам:

$$Kr = \frac{1{,}2}{K\frac{\tau}{T}}., \; Ти = \frac{2\tau}{Kr}, \; Тд = 0.4\tau \cdot Kr.$$

В Trace Mode вместо времени интегрирования вводится коэффициент равный **1/Ти**.

Таблица 1.1 – Данные для расчета передаточной функции

0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**0.10**	**0.11**	**0.12**	**0.13**	**0.14**	**0.15**	**0.16**	**0.17**
А	**Б**	**В**	**Г**	**Д**	**Е**	**Ё**	**Ж**	**З**	**И**	**Й**	**К**	**Л**	**М**	**Н**	**О**	**П**
0.18	**0.19**	**0.20**	**0.21**	**0.22**	**0.23**	**0.24**	**0.25**	**0.26**	**0.27**	**0.28**	**0.29**	**0.30**	**0.31**	**0.32**	**0.33**	
Р	**С**	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ъ**	**Ы**	**Ь**	**Э**	**Ю**	**Я**	

4 Содержимое отчета

В отчете описать основные элементы проекта, аргументы элементов операторского интерфейса, программы, созданные каналы и привязки. Привести скриншоты основного окна операторского интерфейса в режиме симуляции и **FBD** программы.

Подобрать настройки ПИД регулятора самостоятельно и сравнить переходные процессы с расчетными настройками. Выполнить несколько различных настроек (не менее 3). В отчете привести тренды.

5 Контрольные вопросы

1. Языки МЭК 61131-3. Общие характеристики;
2. Принципы построения программ на языках МЭК 61131-3;
3. Язык FBD. Блок задания модели объекта OBJ;
4. Язык FBD. Блок ПИД регулятора.

Список литературы

1. Андреев, Е. Б. SCADA-спстемы: взгляд изнутри/ Е. Б. Андреев, Н. А. Куцевич. О. В. Синенко. – М.: издательство РТСофт. 2004. – 176 с.
2. Букреев, В. Г. Основы инструментальной системы разработки АСУ/ В. Г. Букреев, А. В. Цхе. – Томск: издательство ТПУ, 2003. – 127.
3. Локотов, А. Что должна уметь система SCADA/ А. Локотов// Современные технологии автоматизации. – 1998. – 3. – с. 44.
4. Руководство пользователя. TRACE MODE 6 & T-FACTORY. Быстрый старт. Издание пятое (к релизу 6.03.1). Москва 2006. AdAstrA Research Group. Ltd.-l 63 с.
5. Руководство пользователя. TRACE MODE 6. Издание седьмое (к релизу 6.03.1). Москва 2006. AdAstiA Research Group. Ltd. Том 1-554 с. Том 2-598 с.
6. Деменков, Н. П. SCADA – системы как инструмент проектирования АСУ ТП. М. : Изд-во МГТУ им. Н.Э. Баумана. 2004. – 328 с.
7. Управляющие вычислительные комплексы: учеб. пособие / под ред. Н. Л. Прохорова. – М. : Финансы и статистика, 2003. – 352 с.

Ковалев Алексей Викторович
Литвинов Дмитрий Александрович
Ростокина Ольга Михайловна

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ

Практикум
для студентов специальности
1-53 01 07 «Информационные технологии и управление в технических системах»
дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 16.11.22.
Рег. № 30Е.
http://www.gstu.by