

Министерство образования Российской Федерации
Ульяновский государственный технический университет

SCADA–система Trace Mode

методические указания к лабораторным работам

Составители:
И. П. Ефимов
Д. А. Солуянов

Ульяновск 2010

SCADA–система Trace Mode/ Сост. И. П. Ефимов, Д. А. Солуянов.—
Ульяновск: УлГТУ, 2010г.— 158 с.

Ключевые слова: автоматизация, процесс, SCADA, система.

Методические указания содержат описание существующих SCADA–систем, рассматривается отечественная система Trace Mode. Предлагаются лабораторные работы, позволяющие студентам познакомиться со SCADA–системой Trace Mode, научиться создавать статическое и динамическое изображение, производить программную обработку на языках программирования среды Trace Mode. В приложениях приводится пример отчета, созданный системой Trace Mode и FBD блоки, которые могут понадобиться студентам при выполнении лабораторных работ.

Содержание

Глава I SCADA–системы	5
Понятие SCADA–системы	5
Графический интерфейс SCADA–систем	9
Тревоги и события	13
Тренды	15
Программирование	17
Отчеты	21
Глава II Система Trace Mode	23
Система Trace Mode	23
Навигатор проекта	26
Редактирование канала	26
Привязка аргументов	28
Создание объектов экрана	29
Статическое изображение	33
Динамическое изображение	38
Программирование в Trace Mode	41
Язык Техно ST	43
Язык Техно FBD	55
Язык Техно SFC	59
Язык Техно IL	67
Отчет тревог	73
СПАД архив	82
Глава III Лабораторные работы	84

Установка системы Trace Mode 6	84
Работа 1. Создание проекта	88
Работа 2. Создание статического и динамического изображения	106
Работа 3. Программирование на языках Техно St и Техно FBD	115
Работа 4. Программирование на языках Техно IL и Техно SFC	127
Работа 5 Создание отчета тревог и СПАД архива	136
Приложение 1. Пример отчета Trace Mode	145
Приложение 2. Арифметические FBD блоки	148
Приложение 3. FBD блоки сравнения	150
Приложение 4. FBD блоки выбора	153
Приложение 5. FBD блоки-генераторы	156
Список литературы	158

Глава I SCADA–системы

Понятие SCADA–системы

Большую роль в повседневной жизни играют разного рода товары и услуги. Оказание различных услуг, производство товаров включает в себя различные процессы. В большинстве случаев бесконтрольное протекание процессов недопустимо из-за возможного нанесения травм, материального ущерба, создания аварийной ситуации. Нашел широкое применение контроль протекания технологических процессов. Процессы автоматизируются с использованием современной вычислительной техники, что позволяет точнее выставлять температуру в печи, к примеру, снижая расходы на топливо. Сегодня могут развиваться, конкурировать на рынке только те производители, которые используют современную технику, обеспечивая автоматизацию технологических процессов. По этим причинам все большее распространение получают различные автоматизированные процессы на производстве.

Автоматизация технологических процессов начиналась с разработки САР систем (Система Автоматического Регулирования). САР обеспечивали управление отдельными параметрами, агрегатами. Техника начинает отслеживать значения отдельного параметра, программно управлять процессом, стабилизировать различные параметры технологических процессов.

Дальнейшее развитие науки и техники приводит к созданию САУ (Система Автоматического Управления). Объектами управления становятся системы. САУ становятся способными воспроизводить сложные законы управления или регулирования, появляется возможность идентификации объектов и состояния системы. Системы включают в себя измерительные системы, исполнительные механизмы, средства отображения информации. Человек все больше удаляется от технологического процесса.

Дальнейшее развитие науки и техники приводит к распространению вычислительной техники. Вычислительная техника автоматизирует технологические процессы. Появляется АСУ ТП (Автоматизированная Система Управления Технологическим Процессом) Сначала использовались

микроконтроллеры, автоматизирующие технологические процессы. Подобные системы годились для автоматизации процессов, относительно простой визуализации, но не обеспечивали хранение, обработку измерительной информации, полноценное взаимодействие с оператором. Также промышленники отмечают следующие недостатки[6]: неповоротливость программных комплексов, низкая скорость обмена данными, недостаточный объем памяти. Появление персональных компьютеров позволило обеспечить не только автоматизацию технологических процессов, но и хранение, обработку поступающей информации, увеличить пропускную способность АСУ ТП, решить проблему малого объема памяти, улучшить визуализацию. АСУ ТП на базе современных вычислительных машин позволяет так же создавать различные документы, отчеты о состоянии процесса, дистанционно управлять процессом. В дальнейшем будем рассматривать АСУ ТП на базе персонального компьютера.

Обобщенная схема АСУ ТП изображена на рис. 1.1. Можно выделить три уровня: контроллеры нижнего уровня, контроллеры верхнего уровня, диспетчерский уровень.

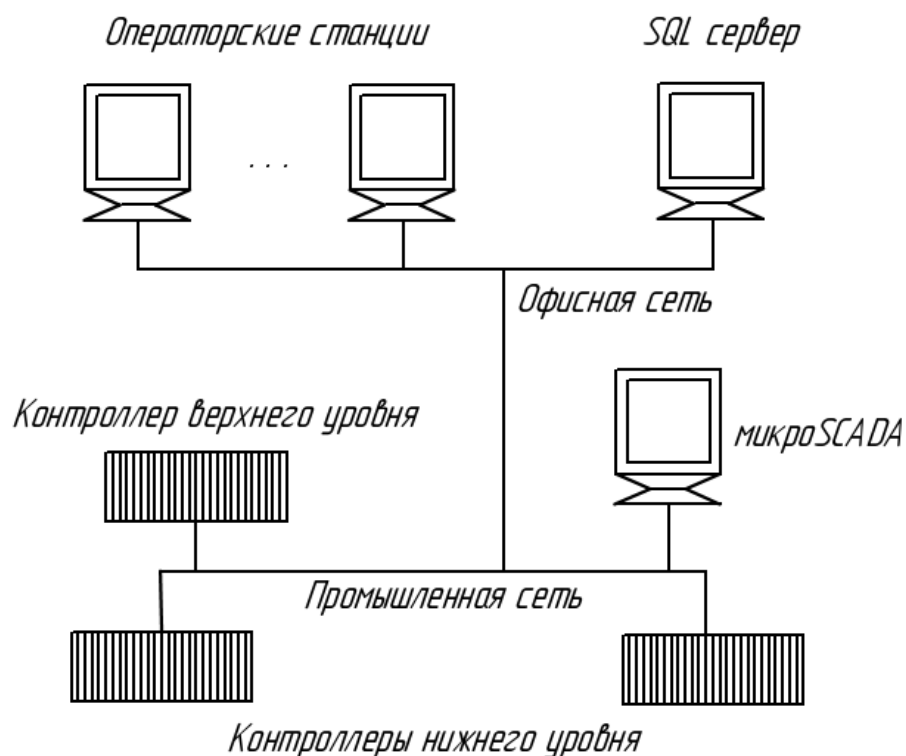


Рис. 1.1 Обобщенная схема АСУ ТП

Контроллеры нижнего уровня осуществляет следующие функции:

1. сбор данных о состоянии технологического процесса;
2. управление работой исполнительных механизмов;
3. автоматическое логическое управление.

Данные с контроллеров нижнего уровня могут поступать в офисную сеть диспетчерского уровня непосредственно или через контроллеры верхнего уровня. Контроллер верхнего уровня выполняет следующие функции:

1. сбор данных с контроллеров нижнего уровня;
2. обработка данных (масштабирование, к примеру);
3. синхронизация работы подсистем АСУ ТП;
4. создание архивов;
5. сохранение работоспособности при нарушении связи между контроллерами верхнего уровня и диспетчерским пунктом;
6. резервирование каналов, по которым происходит передача данных.

В качестве контроллеров верхнего уровня могут использоваться концентраторы, коммуникационные контроллеры.

микро–SCADA— программное обеспечение АСУ ТП, реализующее автоматическое управление и контроль технологического процесса, специализирующееся на автоматизации в определенной области.

Диспетчерский уровень представлен в первую очередь операторскими станциями, а также рабочими местами специалистов, сервером баз данных. Диспетчерские станции получают от подсистем и систем ввода/вывод различные данные о состоянии технологического процесса. Полученные данные необходимо обработать определенным образом, проанализировать, преподнести диспетчеру в той или иной форме информацию о состоянии технологического процесса, дать ему возможность управлять процессом. Помимо этого следует выполнять и другие функции, такие как создание документов и отчетов. Для выполнения указанных функций необходимо

программное обеспечение , которое обеспечит сбор, обработку, анализ данных о параметрах процесса, управление процессом.

Можно использовать программное обеспечение, написанное на языке высокого уровня или в специальной среде разработки АСУ ТП. Создание программ на языке высокого уровня требует не только знания языков программирования и навыков программирования, но и понимания технологического процесса, который необходимо автоматизировать. Программисты, которые создают программное обеспечение для управления технологическим процессом на языках высокого уровня, должны изучить автоматизируемый процесс, что приводит к длительности создания АСУ ТП, делает разработку дорогостоящей. Данную проблему позволяют решить SCADA–системы. Технолог, который хорошо знает технологический процесс, не имеет навыков программирования и не может написать программу на языках высокого уровня для АСУ ТП. Поэтому необходима специализированная система, позволяющая автоматизировать любой технологический процесс. Такие системы называют SCADA–системами.

Под SCADA–системой следует понимать специализированное программное обеспечение, реализующее интерфейс между человеком и системой управления, коммуникацию с внешним миром. Широкое распространение получили следующие SCADA–системы: Genesis, Trace Mode, InTouch, Citect, IGSS.

Практически все современные SCADA–системы выполняют следующие функции:

1. сбор информации о контролируемых технологически параметрах;
2. сохранение принятой информации в архивах;
3. вторичная обработка принятой информации;
4. графическое представление хода технологического процесса;
5. прием команд от оператора;
6. регистрация событий, связанных с контролируемым процессом;
7. оповещение персонала об аварийных ситуациях на производстве;
8. создание разного рода документов о ходе процесса;

9. автоматическое управление ходом технологического процесса.

SCADA–системы представляют следующие основные возможности:

1. предлагает кнопки, поворотные регуляторы и другие органы управления обеспечивая возможность управления технологическим процессом;
2. предлагает набор различных индикаторов, графиков, обеспечивая возможность индикации информации о процессах;
3. предоставляет возможность создания разного рода отчетов, архивов;
4. предлагает упрощенный язык для создания алгоритмов, что дает возможность создания АСУ ТП технологам, у которых нет опыта программирования на языках высокого уровня;
5. предлагает средства для документирования разрабатываемых алгоритмов и технологических процессов;
6. драйверы к оборудованию, обеспечивающие ввод, вывод аналоговых и дискретных сигналов;
7. сетевые функции, позволяющие производить обмен данными между вычислительными машинами, подключенными к одной сети, публиковать отчеты в сети или управлять процессом с удаленного компьютера через интернет.

Графический интерфейс SCADA–систем

Все современные SCADA–системы позволяют создавать графический интерфейс, что облегчает диалог оператора с машиной. Среди SCADA–систем распространена векторная графика, что позволяет создавать отдельные графические объекты, производить различные операции над ними, обеспечивать динамичность изображения за счет масштабирования, перемещения, вращения, изменения цвета объектов, образующих изображение.

Графический интерфейс может создаваться в специальном редакторе. Во многих случаях (InTouch, Genesis 32) необходимо запускать специальный редактор для создания интерфейса человек- машина. Редактор может быть встроен в систему, предназначенную для разработки АСУ ТП (Trace Mode, Genie).

Редакторы графического интерфейса позволяют создавать различные окна, отличающиеся размерами, цветом фона или изображением, используемым в качестве фона. SCADA–системы позволяют производить переключение между созданными окнами. Окно может создаваться двумя путями:

1. создается стандартное окно (Trace Mode, Genie), которое пользователь настраивает в соответствии с требованиями;
2. создается окно с предварительным выбором типа окна (InTouch, Citect).

В последнем случае пользователь выбирает тип (шаблон) окна, которое он хочет создать. Количество типов окон может быть различным от трех(InTouch) до двенадцати (Citect), к примеру. При большом количестве типов (шаблонов) окон, когда пользователь использует ряд окон, остается только указать имена переменных, которые связаны с создаваемым окном, что позволяет ускорить создание графического интерфейса человек-машина.

Все многообразие изображений, создаваемых в SCADA–системах можно поделить на две большие группы:

1. статическое изображение, которое не изменяется в течение времени (рис. 1.2);
2. динамическое изображение, которое изменяется, отображая ход того или иного технологического процесса или значение физической величины.

Сочетание динамического и статического изображения позволяет получить на экране интуитивно понятное изображение технологического процесса (рис. 1.3), состояние данного процесса. Мнемосхема, изображенная на рис. 1.3 содержит статическое изображение и динамическое. Статическое изображение представлено трубами, насосами, емкостями. Мнемосхема содержит также элементы управления— ползунок, поля для вывода данных,

таких как температура на определенном участке производства. Динамическое изображение, в данном случае, представлено разрезами емкостей, которые позволяют показать уровни жидкостей, которые находятся в них.

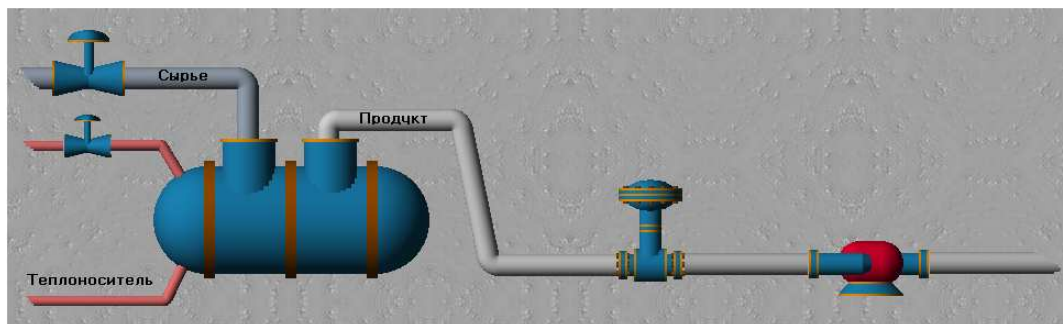


Рис. 1.2 Пример статического изображения



Рис. 1.3 Сочетание статического и динамического изображения

Среда INTouch предлагает инструменты, которые позволяют создавать графические примитивы (линии, кнопки, заполненные контуры,), графики (тренды). У каждого графического объекта можно изменить свойства такие, как цвет линии, заполнения, ориентация в пространстве, ширина. Настраивая данные свойства можно обеспечить статичность или динамичность изображения.

Графическая среда Cites в отличие от InTouch содержит специальный инструмент, который позволяет разместить на разрабатываемом окне объекты с динамическими свойствами. Предлагаются следующие стандартные свойства: перемещение (горизонтальное, вертикальное, вращательное), размер (горизонтальный, вертикальный), цвет заполнения и изменение цвета, команда, выполняемая по нажатию и так далее. Среда также позволяет разместить статические изображения из библиотек, что позволяет разместить такие стандартные изображения, как емкость, клапан, электродвигатель. Можно сохранить ряд графических объектов, объединив их в одну группу, что позволяет их разместить в последующем на создаваемом окне. Последнюю возможность предлагает инструмент джинни, что позволяет сократить разработку SCADA-системы, используя созданные ранее графические элементы, объединенные в одну группу.

Графическая среда Trace Mode предлагает инструменты для создания таких графических примитивов, как линия, ломанные, кривые, прямоугольники, плоские фигуры, объемные фигуры, а также различные кнопки, тренды (графики), выключатели, приборы для отображения значения величины, регулятор в виде ползунка, диаграммы. Ряд графических примитивов дают возможность настроить свойства динамического изображения, что позволяет сделать изображение динамическим. Можно выделить следующие виды динамического изображения: динамическая заливка, динамический контур, динамическая трансформация. Многие объекты позволяют настроить выполнения того или иного действия при нажатии или отпуске левой клавиши мыши. Многие графические объекты, позволяют изменять цвет заполнения замкнутой фигуры в зависимости от принадлежности параметра тому или иному диапазону.

Среда Genie предлагает инструменты, которые позволяют не только разместить графические примитивы, такие как линия, окружность, но и графики, стрелочные приборы, разного рода регуляторы, что позволяет пользователю не только отслеживать изменение параметров, но и управлять процессом. Графические примитивы позволяют изменять цвет отображения в зависимости от нажатия на объект или его активности (активен/неактивен).

Тревоги и события

На многих производствах необходимо контролировать тот или иной параметр для исключения аварии, выхода из строя оборудования. SCADA–системы позволяют контролировать значения параметров производственного процесса. Одного только контроля параметров не достаточно, необходимо во многих случаях сообщать оператору об аварийной ситуации, близости значения параметра к аварийному значению, вести учет всех имевших место аварийных ситуаций. Все современные SCADA–системы позволяют работать с тревогами (алармы) и событиями.

Тревога (аларм)— сообщение, предупреждающее оператора о возникновении ситуации близкой к критической, которая может привести к аварии, выходу из строя оборудования и требует внимания и, зачастую, срочного вмешательства оператора.

Событие— сообщение системы, которое не требует срочно вмешательства оператора.

Можно выделить аналоговые и дискретные тревоги. Дискретные тревоги происходят, при соответствующем изменении дискретного сигнала, то есть принимает значение истина или ложь. Аналоговые тревоги заключаются в том, что соответствующий параметр выходит за пределы заданного диапазона.

В SCADA–системе InTouch выделяют стандартную и распределенную систему тревог. Стандартная система тревог информирует об аварийных ситуациях, событиях, которые происходят в локальном InTouch приложении. Распределенная система тревог информирует о тревогах и событиях, которые генерируются другой удаленной InTouch системой, связь с которой осуществляется посредством сети.

Каждому событию и тревоге в системе InTouch ставят в соответствие приоритет. В качестве приоритета выступает целочисленное значение от 1 до 999. Чем меньше число, которое определяет приоритет, тем более важной является генерируемая тревога.

InTouch позволяет установить соответствие между переменной и группой тревог. Пользователь может задать иерархию созданных групп тревог.

InTouch допускает отображение тревог с помощью двух типов окон: текущие тревоги и архив тревог. Первый тип окон отображает текущую тревогу, а второй— выводит информацию о последних тревогах и событиях. Помимо вывода сведений о тревоге на экран предусмотрено сохранение сведений о тревоге в текстовом файле. Сведения о тревоге можно вывести на печать в данной SCADA–системе.

SCADA–система Citect также работает с тревогами и событиями. В данном случае выделяют тревогу аппаратную и конфигурируемую. Аппаратная тревога возникает в случае неисправности устройств АСУ ТП. Система Citect производит диагностические процедуры для проверки состояния собственного и периферийного оборудования. Конфигурируемые тревоги возникают при выходе параметра за пределы установленного диапазона. Можно выделить четыре типа тревог: дискретные, аналоговые, с меткой времени, составные. Тревога с меткой времени срабатывает при изменении дискретного параметра но, в отличие от дискретной тревоги, имеет точную привязку ко времени, что позволяет достаточно точно определить время возникновения тревоги. Составные тревоги генерируются, когда результат определенного выражения меняет свое значение с «ложь» на «истина».

Citect допускает классификацию тревог по различным признакам: участок производства, тип тревоги, имя и так далее. Пользователь может определить до 255 категорий тревог. У каждой тревоги свой приоритет. Важность приоритета уменьшается с увеличением его значения от 1 до 255. Таким образом, единице соответствует наивысший приоритет.

Для отображения тревог среда Citect позволяет использовать два объекта: страница текущих тревог и страница сводки тревог. Система позволяет вывести следующую информацию о тревоге: имя переменной, тревоги, описание тревоги, категория тревоги, справочная информация, уровень доступа, время или дата смены состояния, время и дата возникновения и окончания тревоги, длительность тревоги.

Система Trace Mode также позволяет работать с аналоговыми и дискретными тревогами. Система позволяет настроить аналоговые тревоги,

задавая различные диапазоны значений контролируемого параметра. Данная система для представления дискретных сигналов использует специальные типы данных— HEX16 и HEX32. Для настройки дискретных тревог необходимо установить соответствие между тревогой или событием и состоянием соответствующего бита значения, хранимого каналом HEX16(32). Система позволяет установить соответствие между событием или тревогой и категорией. Категория отображает степень важности для пользователя сообщения. В качестве примера можно привести следующие категории: без категории, ошибка, информация, тревога, сообщение, предупреждение.

Тревоги и события могут быть отображаться с помощью специального объекта— отчет тревог, который выводит на экран последнее сообщение о событии или тревоге. Информация о событии или тревоге может сохраняться в текстовом файле (отчет тревог). Сведения о тревогах и событиях могут также выводиться на принтер, отправляться на сотовый телефон в виде SMS, выкладывать в сеть. Возможно также воспроизведение определенного звукового файла при генерировании определенной тревоги или события.

Система Genie позволяет генерировать аналоговые тревоги. Тревоги генерируются, когда контролируемый параметр выходит за установленные границы. Сообщения о тревогах среда позволяет сохранить в файле архива тревог. Возможно также отображения информации о тревогах в специальном окне журнала событий. Система позволяет установить соответствие между объектом для вывода динамического изображения и блоком архива тревог, который генерирует сообщения о тревогах, что позволяет представить информацию об аварийной ситуации в графической форме.

Тренды

Динамику изменения технологического параметра во времени удобно представить в виде зависимости изменения данного параметра во времени. По этой причине в SCADA–системах нашли широкое распространение объекты, которые позволяют представить изменение определенного параметра во времени. Такие объекты называют трендами. Пример тренда

изображен на рис. 1.4 Можно выделить два типа трендов: тренды реального времени и архивные (исторические) тренды.

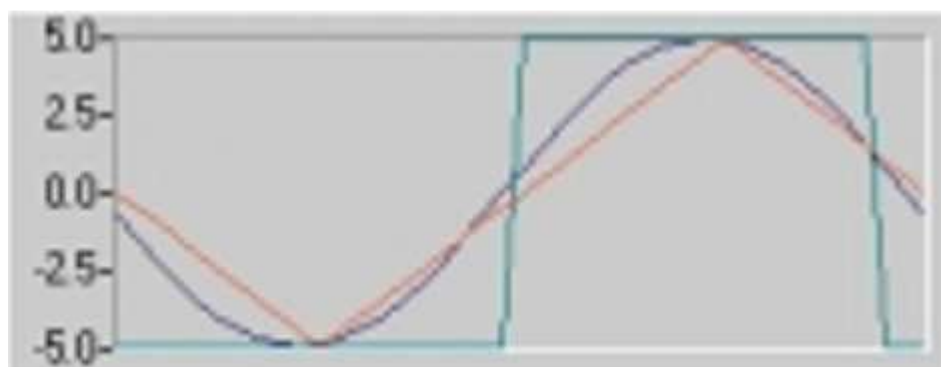


Рис. 1.4 Тренд

Тренд реального времени отображает в реальном времени изменение параметра. Происходит постоянный сдвиг зависимости влево, новые значения контролируемого параметра постоянно добавляются к построенной зависимости справа. Как правило, тренды содержат полосу прокрутки, что позволяет вернуться «назад» и посмотреть, что происходило ранее на производстве, или вернуться «в текущее время».

SCADA–системы позволяют создавать архив данных. Под архивом следует понимать бинарный файл, который содержит информацию об изменениях контролируемых параметров и времени, когда они произошли. Архивный тренд позволяет построить зависимость выбранного параметра от времени на основе данных, которые сохранены в архиве данных. Подобные тренды также содержат полосу прокрутки, что позволяет проанализировать весь объем данных, содержащихся в архиве.

В SCADA–системе InTouch выделяют тренды архивные и реального времени. Архивный тренд позволяет построить до 8 графических зависимостей параметров от времени. Тренд реального времени позволяет построить до четырех зависимостей. Данная среда поддерживает распределенные архивы, что проявляется в том, что тренд может получать данные от удаленных баз данных InTouch приложений.

SCADA–система Citect содержит единую распределенную систему трендов. В данной системе сбор, обработку и хранение информации об изменении параметров для графического представления осуществляет сервер

трендов. При необходимости построения тренда реального времени или архивного клиент обращается к серверу трендов и запрашивает данные.

SCADA–система Trace Mode позволяет строить диаграммы, тренды. Данная среда позволяет строить тренды реального времени и архивные, круговую диаграмму, гистограмму реального времени и архивную. Круговая диаграмма удобна, когда необходимо показать вклад каждой составляющей в сумму всех составляющих. Гистограмма позволяет отобразить значения нескольких параметров в виде столбцов, расположенных рядом друг с другом. Диаграммы удобны для сравнения нескольких параметров друг с другом. В отличие от InTouch и Citect есть тренд X-Y, что позволяет построить зависимость параметра Y от параметра X. Среда Trace Mode не накладывает ограничения на количество трендов или диаграмм, количество графиков на каждом тренде, столбцов гистограммы, секторов круговой диаграммы.

SCADA–система Genie позволяет строить зависимость параметра от времени, зависимость параметра X от Y как и среда Trace Mode. Рекомендуют строить не более 8 графиков на одном тренде.

Программирование

Все SCADA–системы содержат встроенные языки программирования. У каждой системы индивидуальный язык программирования. Все многообразие языков программирования SCADA–систем в зависимости от навыков программирования, которые требуются от пользователя для написания программы, можно разделить на две большие группы:

1. языки, ориентированные на технологов;
2. языки, ориентированные на системных интеграторов.

Поскольку у технологов на практике нет опыта программирования, то использовать языки высокого уровня они не могут. Для них разрабатываются особые языки, которые являются упрощенными, не требующими специальных навыков. Все большее распространение получают графические языки программирования (рис. 1.5). В таком языке программирования пользователь размещает на рабочем поле блоки, которые выполняют

отдельные функции. Каждый блок содержит определенные входы и выходы. Пользователю остается соединить соответствующие выходы с входами, задавая направления передачи данных.

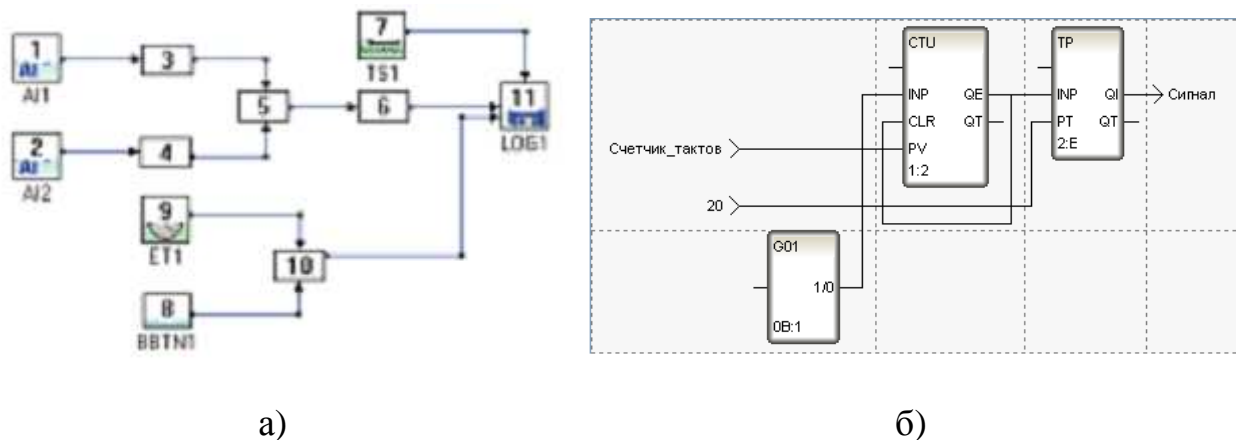


Рис. 1.5 Графические языки программирования

Языки, ориентированные на системных интеграторов, у которых есть навыки программирования, больше похожи на Visual Basic.

Во всех языках программирования SCADA–систем можно выделить математические, логические функции, для работы со строками, DDL и SQL обмена данными.

В SCADA–системе InTouch программирование построено на скриптах. Под скриптом следует понимать программные фрагменты, которые выполняются при определенных событиях, таких как нажатие кнопки, клавиши, открытие окна и так далее. Выделяют скрипты простые и сложные. Для простых скриптов характерно использование операндов, которые позволяют выполнять простые математические действия, сравнения, присваивание и так далее. Сложные скрипты, в отличие от простых, логические операции, такие как оператор сравнения `if` во многих языках программирования высокого уровня, циклы.

В системе InTouch используются также функции, которые могут быть связаны со скриптами, командами. Система предлагает пользователю различные функции для работы со строками, математическими функциями, системными функциями, тревогами, трендами. Особое место занимает функции Quick Functions, которая является скриптом, вызываемым из других скриптов.

Для системы Citect характерен язык программирования Cicode. Данный язык по своей структуре похож на VisualBasic и С. Большую роль играет команда, которая вызывается тем или иным образом на исполнение и решает определенную задачу. Команды допускают ручной запуск, то есть по нажатие кнопки на экране или клавиши на клавиатуре. Автоматически вызываются на исполнение команды при регистрации оператора системой (оператор ввел пароль при входе в систему), открывании и закрывании окон проекта, при срабатывании тревог или событий, формировании отчета.

В Citect выделяют также выражения, которые, в отличие от команд, не выполняют конкретные задачи, а оценивают их. Выражения включают в себя константы, значения переменных, результат вычислений. Большую роль играют также функции— набор выражений, констант, переменных, операторов, условий. В отличие от команд функция позволяет решать более сложные задачи.

SCADA–система Genie содержит редактор задач. В данном редакторе задается алгоритм обработки данных и управления технологическим процессом на основе графического языка (рис. 1.5, а). Редактор задач, как и любой графический язык программирования, позволяет легко создать АСУ ТП без навыков программирования. Помимо редактора задач используется Бейсик-сценарий— язык программирования, схожий с VisualBasic. Данный язык позволяет получить большую гибкость программирования, создавая небольшие подпрограммы.

SCADA–система Trace Mode использует модифицированные языки программирования, ST, IL, FBD, LD стандарта IEC61131-3. Язык TechnoST ориентирован на программистов, его синтаксис похож на Pascal и С. Язык позволяет легко писать программы как людям, знакомым с Pascal, так и с С. Это выражается в том, что присвоение, к примеру, можно производить оператором, который можно записать как «=», так и «:=». Данный язык позволяет использовать циклы, операторы безусловного перехода и с условием, выбора. Есть ряд стандартных функций аналогичных С.

Язык Techno IL похож на Ассемблер. Базовым понятием языка является аккумулятор. Аккумулятор— ячейка памяти, содержащая результат вычислений. В данную ячейку записываются результаты всех операций, оператор может туда записать значение любого операнда. Данный язык не содержит циклов, операторов выбора. Язык предоставляет пользователю

операторы для арифметических операций, сравнения, работы с логическими переменными, обращения к аккумулятору, безусловного перехода. Данный язык позволяет писать программы для контроллеров с низкой вычислительной мощностью.

Язык Техно FBD— графический язык программирования (рис. 1.5, б). В данном языке пользователь размещает на рабочем поле блоки, которые выполняют различные функции, и соединяет их входы и выходы, задавая направление передачи данных от блока к блоку.

Язык Техно LD предназначен для специалистов, которые привыкли составлять схемы релейной логики. Оператор размещает на рабочем столе катушки и контакты, блоки языка Техно FBD, соединяет их между собой, задавая алгоритм обработки.

Язык Техно SFC позволяет структурировать сложную программу. Внешне программа напоминает блок-схему (рис. 1.6). Такая программа содержит шаги и условия. Каждый шаг такой программы— подпрограмма на одном из языков Trace Mode(ТехноST, Техно IL, Техно FBD, Техно LD), реализующая те или иные действия. Переход к следующему шагу в такой программе производится, когда условие, соответствующее данному переходу, возвращает значение «истина». Условие перехода к следующему шагу представляет собой программу, написанную на одном из языков Trace Mode(ТехноST, Техно IL, Техно FBD, Техно LD). Таким образом, данный язык позволяет задать алгоритм вызова подпрограмм (шагов), которые могут быть написаны на разных языках Trace Mode.

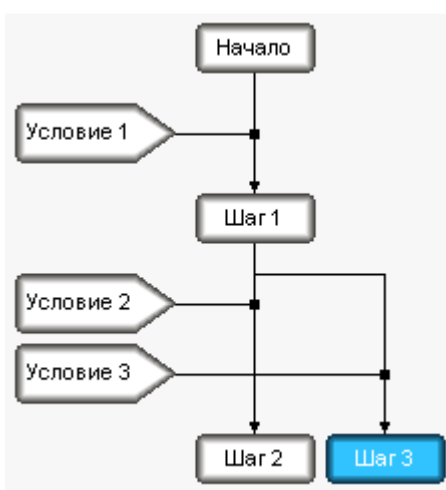


Рис. 1.6 Пример программы на языке Техно SFC

Отчеты

SCADA–системы позволяют создавать различные отчеты. Под отчетом следует понимать документ, содержащий сведения о состоянии технологических процессов, разного рода событиях и тревогах. Можно выделить два вида отчетов: отчет тревог, который рассматривался при рассмотрении тревог, и отчет о состоянии производства. Рассмотрим подробнее отчет о состоянии производства.

Данный отчет может содержать различные сведения о процессах, оборудовании, тревогах, событиях. Это могут быть как данные, получаемые с различных датчиков, так и результаты обработки данных SCADA–системой. Сам отчет может быть представлен по-разному: как таблица или как html страница, к примеру. SCADA–системы позволяют сохранить созданный отчет в виде книги MS Exsel, pdf-файла, html- страницы.

Системы позволяют следующие действия с созданным документом:

1. отправить на печать;
2. отправить по электронной почте получателя;
3. опубликовать в интернете;
4. отправлять по факсу;
5. сохранять в заданной директории.

Отчет может быть создан SCADA–системой в различных случаях:

1. по команде оператора;
2. периодически с заданным периодом генерации;
3. по событию;
4. при генерировании тревоги;
5. по результатам вычисления.

SCADA–система Trace Mode позволяет создавать отчет в виде-html страницы. Как следствие, отчет позволяет представить различные данные: табличные данные, графики, отчет тревог (вывести тревоги и события), текст,

значения аргументов. Поскольку создается html-страница, пользователь может задать любой фон отчета, размещать рисунки и так далее. Пользователь создает шаблон документа, который задает правила оформления, создания отчета SCADA–системой в последующем. SCADA–система может формировать отчет как по команде оператора, так и с заданной периодичностью, при различных событиях. Пользователь может написать программу, которая будет вызывать генерирование отчета в тех или иных случаях. Созданный отчет система Trace Mode позволяет вывести на печать, опубликовать на Web-сервере, сохранить в определенной директории. Отчет может быть создан на основе данных, полученной как в течение последнего часа, так и в течении года. Пример отчета системы Trace Mode приведен в приложении 1.

SCADA система Genie также позволяет создать отчет. Для создания отчета среда предусматривает сбор данных, которые сохраняются в файле базы данных. Система Genie 3.0 позволяет создавать отчеты только в табличной форме. Пользователь при задании формы отчета может вводить текст, задавать ключевые слова, определяя каждый столбец таблицы. В специальном планировщике пользователь задает моменты времени, когда должен генерироваться отчет. В заданные моменты времени формируется отчет о состоянии производства на основе созданной пользователем формы отчета, сведений, содержащихся в файле базы данных. Отчет может содержать данные за последние сутки, месяц, год. Пример отчета приведен на рис. 1.7.

	A	B	C	D	E
1	Время	ч	09:00	10:00	11:00
2	Расход в 1-й нитке	куб.м/ч	**	**	**
3	Расход во 2-й нитке	куб.м/ч	**	**	**
4	Давление в 1-й нитке	Атм	**	**	**
5	Давление во 2-й нитке	Атм	**	**	**
6					
7					
8					
9					
10					
11					
12					
13					

Рис. 1.7 Пример отчета Genie

Глава II Система Trace Mode

Система Trace Mode

Среди всего многообразия SCADA–систем можно выделить Trace Mode, созданную на территории Российской Федерации. Перед другими SCADA–системами у нее следующие преимущества на территории Российской Федерации:

1. осуществляется поддержка в Российской Федерации;
2. вся документация на русском языке;
3. полностью русифицированный программный продукт;
4. поддерживает не только IBM совместимые зарубежные контроллеры MicroPC, ADAM, PCL, MIC2000, но и отечественные МФК, «Крузиз», продукцию L-card.

Следующие понятия характерны для среды Trace Mode:

Проект— математические и графические элементы системы, которые функционируют на различных операторских станциях и контроллерах, входящих в одну АСУ ТП и объединенных информационными связями и системой архивирования.

Узел— любое устройство в рассматриваемом проекте на котором запущено программное обеспечение Trace Mode. Узлом может быть как станция оператора, так и микроконтроллер, осуществляющий сбор информации или управляющий технологическим процессом.

Канал— информационная структура, которая включает в себя переменные, константы, методы формирования и преобразования значений переменных.

База каналов— совокупность всех каналов, математических объектов, FBD-программ и IL-программ, созданных для каждого узла.

Объект базы каналов— совокупность любых каналов, которой приписан определенный набор свойств и атрибутов.

Все многообразие каналов можно разбить на входные (Input) и выходные (Output). У каждого канала есть набор атрибутов, то есть набор переменных, констант идентификаторов.

Есть четыре основных значения любого канала:

1. In— входное;
2. A— аппаратное;
3. R— реальное;
4. Q— выходное.

Входной канал получает значение от внешних источников (от микроконтроллера, платы ввода/вывода, к примеру) или от системной переменной (длина архива к примеру). Преобразование данных изображено на рис. 2.1.

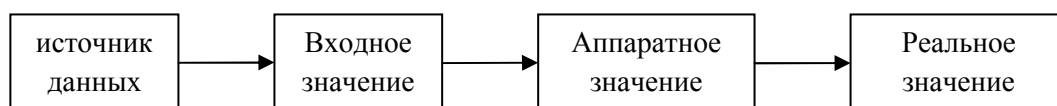


Рис. 2.1 Преобразование сигнала входным каналом

С источника данные поступают во входное значение. Затем происходит масштабирование по формуле:

$$A = In \cdot KX + Z, \text{ где}$$

KX— множитель;

Z— смещение.

После масштабирования значение поступает в аппаратное значение. Аппаратное значение проходит трансляцию (первичная математическая обработка), фильтрацию одиночных пиков или фильтрацию малых изменений, экспоненциальное сглаживание.

Фильтрация пиков заключается в том, что изменение значения игнорируется в течении одного такта пересчета, если изменение превысило установленное значение $DPic(\text{Пик})$.

Фильтрация малых изменений заключается в том, что игнорируются изменения значения, если это изменение меньше данной величины APert. Экспоненциальное сглаживание производится, если значение DSmoot принимает значение из диапазона (0;1]. Для отмены сглаживания можно установить DSmoot (*Сглаж.*) равным 0.

Результат фильтрации и сглаживания подается в реальное значение. Выходное значение входного канала всегда неопределенно.

Выходной канал передает данные внешнему или внутреннему приемнику. Преобразование данных в выходном канале изображено на рис. 2.2.

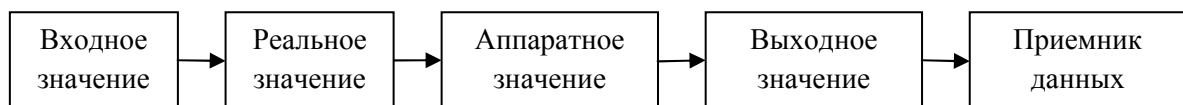


Рис. 2.2 Преобразование сигнала выходным каналом

Входное значение проходит экспоненциальное сглаживание, линейное сглаживание или апертуру, клиппирование (ограничение реального значения). Результат указанных преобразований поступает в реальное значение. Реальное значение после трансляции поступает в аппаратное значение. Аппаратное значение проходит масштабирование и поступает в выходное значение. Масштабирование производится по формуле:

$$Y = (A + Z) \cdot KX$$

Можно создать в Trace Mode следующие виды каналов:

1. канал FLOAT (вещественное число, 4 байта);
2. канал HEX 16 (целое число без знака, 2 байта);
3. канал HEX 32 (целое со знаком, 4 байта);
4. канал Double FLOAT (вещественное число, 8 байт);
5. канал TIME (дата, время);
6. событие (для мониторинга объекта с фиксацией возникновения/исчезновения события);
7. канал CALL (служит для вызова различных компонент).

Навигатор проекта

При создании проекта АСУ ТП необходимо использовать навигатор проекта. На рис. 2.3 изображено окно навигатора проекта. В левой части данного экрана отображается дерево проекта. В правой части окна навигатора отображаются подгруппы, компоненты выбранной группы. Вызов контекстного меню для выделенной группы или компоненты позволяет произвести редактирование выделенной единицы, создать новую группу или компонент. Для того чтобы свернуть или раскрыть слой (группу) необходимо дважды щелкнуть левой клавишей мыши по слою (группе).

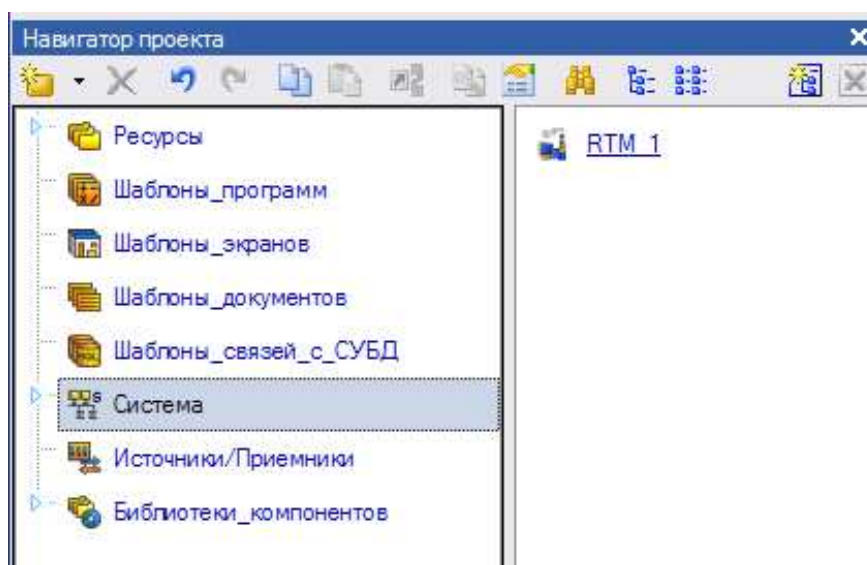


Рис. 2.3 Навигатор проекта

Редактирование канала

Помимо создания канала во многих случаях необходимо его редактирование. Для редактирования необходимо выделить канал, вызвать контекстное меню. В появившемся меню следует выбрать редактировать. Пример внешнего вида окна изображен на рис. 2.4. Внешний вид окна зависит от вида канала (float, hex 16 и так далее). В данном окне можно настроить системные свойства такие, как тип канала, период пересчета канала, создание архивов и так далее. Каналы для работы с вещественными

числами позволяют задать границы, которые необходимы для работы с тревогами. Можно выделить следующие границы:

1. ВП (HL)— значение верхнего предела;
2. ВА (HA)— значение верхней аварийной границы;
3. ВГ (HW)— значение верхней предупредительной границы;
4. НГ (LW)— значение нижней предупредительной границы;
5. НА (LA)— значение нижней аварийной границы;
6. НП (LL)— значение нижнего предела;
7. Гистерезис.

Диапазон [НП;ВП] соответствует достоверным данным, диапазоны (ВА; ВП) и [НП; НА) соответствуют аварийной ситуации, (ВА; ВГ) и (НГ; НА)— близости к аварийным значениям, [НГ; ВГ]— нормальному развитию событий.

Имя: Уровень Кодировка: TC5 Справка

Комментарий

Границы

☒ Использовать

ВП: 0.95

ВА: 0.9

ВГ: 0.8

НГ: 0.2

НА: 0.1

НП: 0.05

Гистерезис: 0

☒ Контроль границ

Обработка

☐ Использовать

Апертура: 0

Пик: 0

Сглаж.: 0

Множитель: 0.01

Смещение: 0

☒ Масштабирование

Масштабирование

In * 0.01 + 0 = A

Max Max

Min Min

Рассчитать

Системные

Основные

Тип: Input

Размерность: ...

Период: 1 Единица измерения: сек

Автопосылка

☒ Включить

Индекс:

☒ Отработать

На старте: 0


Архивация

Дополнительно

Рис. 2.4 Редактирование канала

Флаги **использовать** разрешают использование данных, которые введены на соответствующих панелях. На рис. 2.4 флаг **контроль границ** разрешает контроль заданных границ, а **масштабирование**— автоматический расчет множителя и смещения для заданных диапазонов входных и выходных значений.

Привязка аргументов

При создании проекта в Trace Mode создаются многочисленные каналы, программы. Весь обмен данными происходит через каналы. Связь между определенными значениями каналов, аргументами программы, экрана осуществляется с помощью механизма, который называется привязкой. При создании программы или экрана необходимо создать аргументы. Создание аргументов осуществляется заполнением таблицы, изображенной на рис. 2.5. Иконка  служит для создания нового аргумента. В таблице каждая строка— аргумент. В столбце **имя** можно задать произвольное имя для данного аргумента. Желательно избегать пробелов в именах аргументов. В столбце **тип** указывают тип аргумента: IN— для передачи в программу или вывода на экран, OUT— для передачи из программы или ввода данных через интерфейс человек-машина, IN/OUT— передача данных как в программу, так и из нее, вывод на экран и ввод данных. В столбце привязка производится привязка к определенному значению канала. Двойным щелчком левой клавиши мыши в данном столбце открывается окно, изображенное на рис. 2.6, где выбирается необходимый канал и его значения (атрибут).

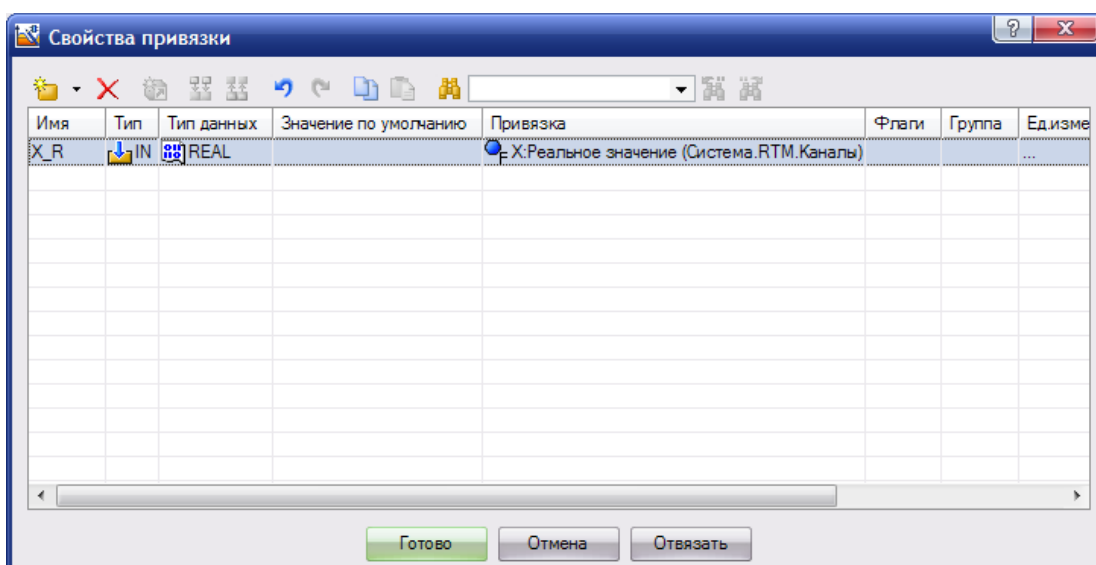


Рис. 2.5 Создание аргументов

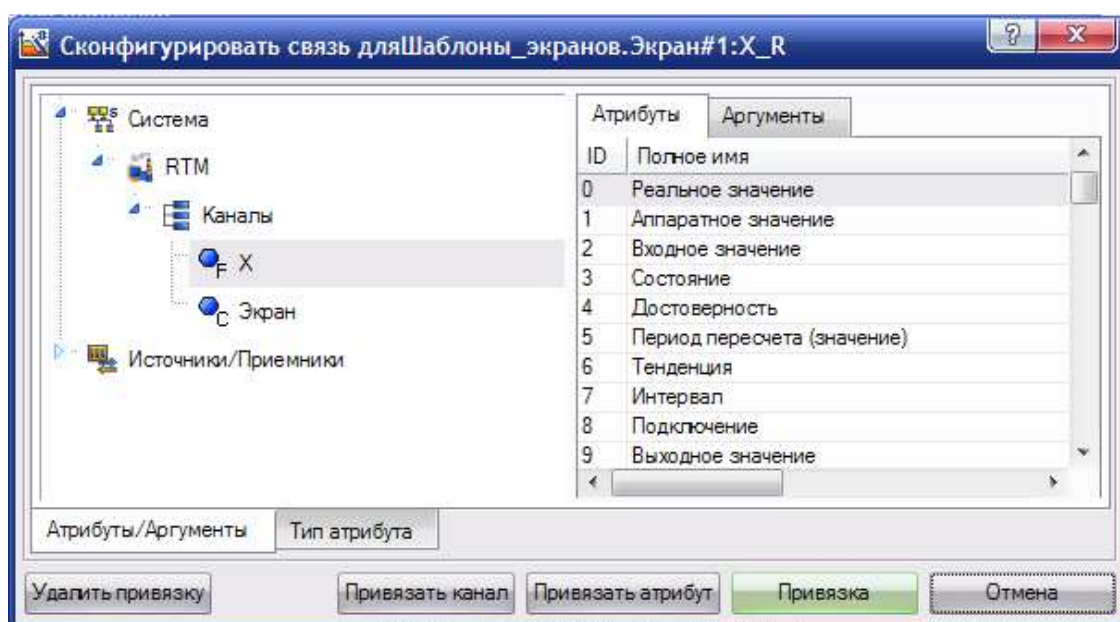


Рис. 2.6 Привязка к каналу

Создание объектов экрана

Для вывода данных на экран, управления системой оператором необходимы 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 2.1) на панели инструментов. В ряде случаев вместо него может отображаться инструмент для создания другого объекта, тогда следует щелкнуть правой клавишей мыши по иконке инструмента той же группы что и искомый. Среди предложенных инструментов следует выбрать необходимый. После выбора инструмента следует щелчком левой клавиши мыши задать два противоположных угла объекта. После создания всех объектов выбранным инструментом следует щелкнуть левой клавишей мыши по иконке  для перехода в режим редактирования. В режиме редактирования можно открыть окно свойств объекта (рис. 2.7), выделив сам объект. Если окно свойств не открыто, его можно открыть двойным щелчком левой клавиши по соответствующему объекту. В данном окне слева расположены имена полей, разделов (подчеркнуты). Для того, чтобы раскрыть или свернуть раздел достаточно дважды щелкнуть по нему левой клавишей мыши. Заполняя поля и разделы можно задать внешний вид объекта, логику его работы.



Таблица 2.1




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

Зачастую необходимо произвести привязку объекта к соответствующему значению (атрибуту) канала. Для привязки необходимо щелкнуть левой клавишей мыши в поле **привязка** или **результат, источник** и так далее. Откроется окно, изображенное на рис. 2.5. В данном окне следует создать аргумент, произвести его привязку к требуемому значению (атрибуту) канала. Если уже существует требуемый аргумент, то достаточно его выбрать в окне на рис. 2.5.

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 меню будет предложено создать кривую (сектор, столбец), чем и следует воспользоваться.

Ряд объектов содержат закладки **динамический контур** , **динамическая заливка** , **динамическая трансформация** . Данные закладки позволяют обеспечить динамичность изображения, которая подробнее рассматривается при рассмотрении динамического изображения.

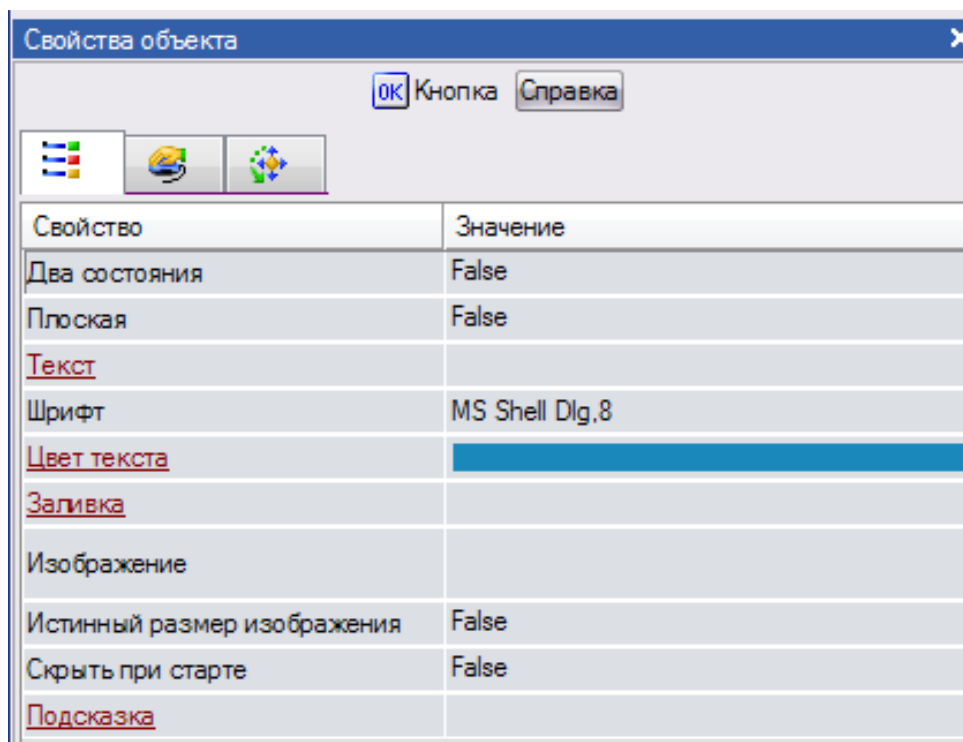



Рис. 2.7 Окно свойств объекта

Объекты могут содержать закладку **действия** , которая позволяет настроить действия, выполняемые при нажатии (*mousePressed*) или отпускании (*mouseReleased*) левой клавиши мыши. В данной закладке можно создать и настроить действие аналогично созданию кривой, вызывая контекстное меню для *mousePressed* или *mouseReleased*.

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) характерно то, что выключатель переводится в положение «вкл», когда **аргумент & константа** = true, в противном случае переключатель переводится в положение «выкл». При щелчке левой клавишей мыши по выключателю аргументу присваивается результат вычисления по формуле **аргумент ^ Значение**, где

^— побитовое исключающее ИЛИ).

При индикации ***Arg >= Конст*** выключатель находится в положении «вкл», 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***.

Статическое изображение



Для создания изображения можно использовать инструменты: линия, ломанные и кривые, прямоугольники, плоские фигуры, объемные фигуры.




Для выбора инструмента линия следует щелкнуть левой клавишей мыши по иконке . Для создания линии достаточно щелкнуть левой клавишей мыши мышкой там, где создаваемая линия должна начинаться и заканчиваться.

Инструмент ломанные и кривые позволяет создавать различные кривые и ломанные, используя объекты, приведенные в таблице 2.2. Инструмент на панели инструментов отображается иконкой выбранного инструмента. Если выбран необходимый инструмент, достаточно щелкнуть левой клавишей мыши на иконке инструмента. Когда отображается инструмент иконкой не того объекта, который нужен, необходимо щелкнуть правой клавишей мыши по иконке инструмента той же группы, что и искомый. Будут предложены объекты, среди которых следует выбрать необходимый.

Таблица 2.2

Ломанные и кривые

Иконка	Название объекта
	Ломаная линия
	Многоугольник

	Ломаная с заливкой
	Разомкнутая кривая
	Замкнутая кривая





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.

Инструмент прямоугольники позволяет создать различные прямоугольники, используя стандартные объекты, указанные в таблице 2.3. Иконка инструмента прямоугольники принимает вид иконки выбранного прямоугольника. Выбор необходимого инструмента среди всех прямоугольников производится аналогично выбору среди ломанных и кривых.

Для размещения прямоугольника необходимо щелчком левой клавиши мыши задать два противоположных угла создаваемого прямоугольника.

Таблица 2.3

Прямоугольники

Иконка	Название объекта
	Контур
	Прямоугольник
	Панель
	Рамка

Инструмент плоские фигуры позволяет создавать изображение клапана, стрелки, треугольника, овала, сектора, эллипса. Стандартные объекты, данного инструмента, приведены в таблице 2.4.

Плоские фигуры

Иконка	Название объекта
	Плоский клапан
	Треугольник
	Стрелка
	Овал
	Эллипс сектор

Плоские фигуры создаются аналогично прямоугольнику.

У плоских фигур, прямоугольников, ломаных и кривых можно настроить статический контур и заливку на закладке **основные свойства**. 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 задать цвет, стиль заполнения.

Инструмент объемные фигуры позволяет создать различные объекты, указанные в таблице 2.5. 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 так далее.

Рассмотрим задание толщины фигуры. Если **толщина стенок** равна 0, то на экране изображается, как выглядит объект внешне. Когда задана **толщина стенок** не равная 0 (толщина стенок больше 0) изображается разрез объекта. Пример изображения тора при толщине стенок равной 0 изображен на рис. 2.8, при толщине стенок неравной 0 — на рис. 2.9.

Рассмотрим настройки материала. Для настройки материала следует дважды щелкнуть левой клавишей мыши на подчеркнутой строчке **материал**, если данный раздел свернут. Если значение **выбрать из списка** установить равным **False**, то материал задается как цвет аналогично другим

объектам. Если значение ***выбрать из списка*** установить равным ***True***, то появляется возможность выбора материала, из которого выполнено изделие. В последнем случае можно выбрать такие материалы, как хром, бронза, медь, задать такие текстуры, как шлифовка или гравировка.

Таблица 2.5

Объемные фигуры

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

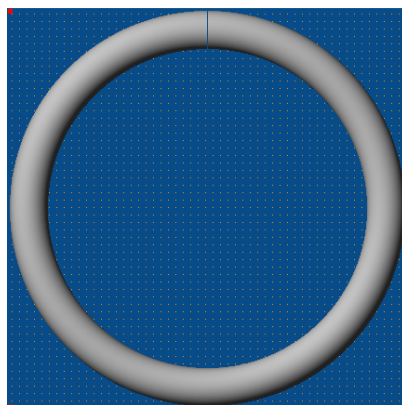


Рис. 2.8 Тор при толщине стенок равной 0

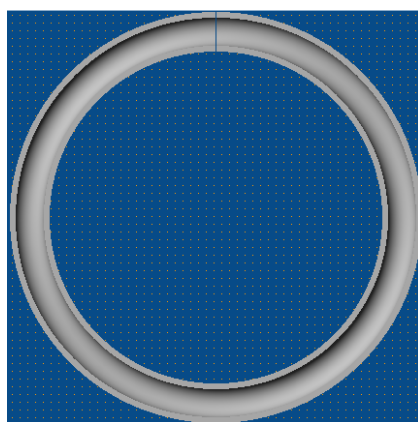


Рис. 2.9 Тор при толщине стенок неравной 0

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

1. создать группу **картинки** в разделе **ресурсы** навигатора проекта;
2. создать **библиотеку изображений** в группе **картинки**;
3. открыть **библиотеку изображений** и импортировать необходимые изображения;

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— его 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.

Динамическое изображение

Динамическое изображение отличается от статического тем, что данное изображение изменяется, его состояние определяется контролируемым процессом. Изображение может перемещаться, изменять размеры, поворачиваться, может происходить перемещение пунктиров по его контуру, 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 (рис. 2.10) два цвета: цвет штрихов и промежутка между ними, длина штриха, которая также определяет шаг перемещения штрихов. Происходит перемещение штрихов по контуру. Скорость их 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.

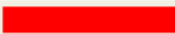


Свойство	Значение
Привязка	ARG_000
Цвет штриха	
Цвет промежутка	
Длина штриха	5
Промежуток./штрих	1

Рис. 2.10 Настройка динамического контура

Рассмотрим динамическую трансформацию (закладка динамическая трансформация ). Можно выделить динамическое перемещение, масштабирование, вращение, поставив соответствующий флаг. При

динамическом перемещении задается ломаная линия, вдоль которой происходит перемещение. Для ряда точек (узлов) задаются соответствующие им значения (рис. 2.11). Текущее положение объекта зависит от значения привязанного аргумента и значений, соответствующих узлам, флага перемещать плавно.

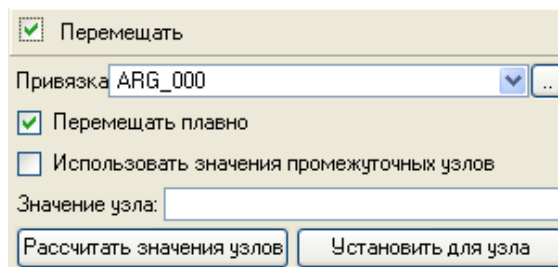


Рис. 2.11 Динамическое перемещение

При динамизации масштабирования задаются начальный и конечный размер объекта и значения привязанного аргумента, соответствующие заданным значениям (рис. 2.12) и центр относительно которого будет происходить масштабирование. Текущий размер зависит от значения аргумента.

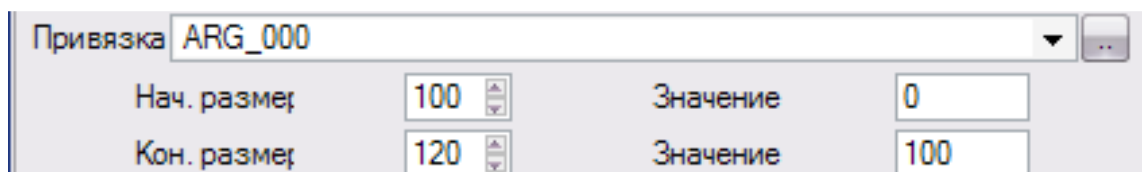


Рис. 2.12 Динамическое перемещение

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.

Рассмотрим подробнее динамическую заливку. Динамическая заливка заключается в том, что задается максимальное и минимальное значение привязываемого аргумента (рис. 2.13). 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*.

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 (рис. 2.14), задать диапазон.

Свойство	Значение
Направление	Вверх
Слой	
Слой (Уровень R)	
Имя	Уровень
Привязка	Уровень_R
Тип заливки	Изображение
Изображение	
Прозрачность	False
Макс.	1
Мин.	0
Мин = НП, Макс = ВП	False
Цвета для диапазонов	False
Предупреждение	
Авария	
Вне границ	

Рис.2.13 Динамическая заливка

Свойство	Значение
<u>Контур</u>	
<u>Заливка (ARG 000)</u>	
Тип заливки	Цвет
<u>Цвет заливки</u>	
Вид индикации	Arg в интервале
Привязка	ARG_000
Мигание	быстрое
Предупреждение	
Авария	
Вне границ	
Стиль	
Скрыть при старте	False
<u>Подсказка</u>	
Выделение в MPB	False

Рис. 2.14 Динамическое изменение цвета

Программирование в Trace Mode

SCADA–системы содержат специальные языки программирования, что позволяет разработчику описывать алгоритмы обработки измерительной информации, генерирования сигналов, управления процессом. Система Trace Mode поддерживает модифицированные языки стандарта IEC6113-3:

1. Texno ST (Structured Text);
2. Texno SFC (Sequential Function Chart);
3. Texno FBD (Function Block Diagram);
4. Texno LD (Ladder Diagram);
5. Texno IL (Instruction List).

Для обеспечения обмена данными между программой и каналами узла служат аргументы программы. Тип каждого аргумента определяет направление передачи данных. Так если аргумент служит для передачи

значения в программу, следует выбрать тип IN, если для передачи из программы— OUT. В ряде случаев нужен аргумент, который будет передавать данный как в программу, так и из нее, тогда следует выбрать тип IN/OUT. Для обмена данными программой необходимо произвести привязку каждого аргумента программы к тому или иному каналу (аргументу канала). Помимо аргументов можно создать локальные и глобальные переменные.

Окно редактирования программы (рис. 2.15) можно открыть двойным щелчком левой клавишей мыши по объекту *программа* в навигаторе проекта или вызвав контекстное меню для него и выбрав *редактировать шаблон*.

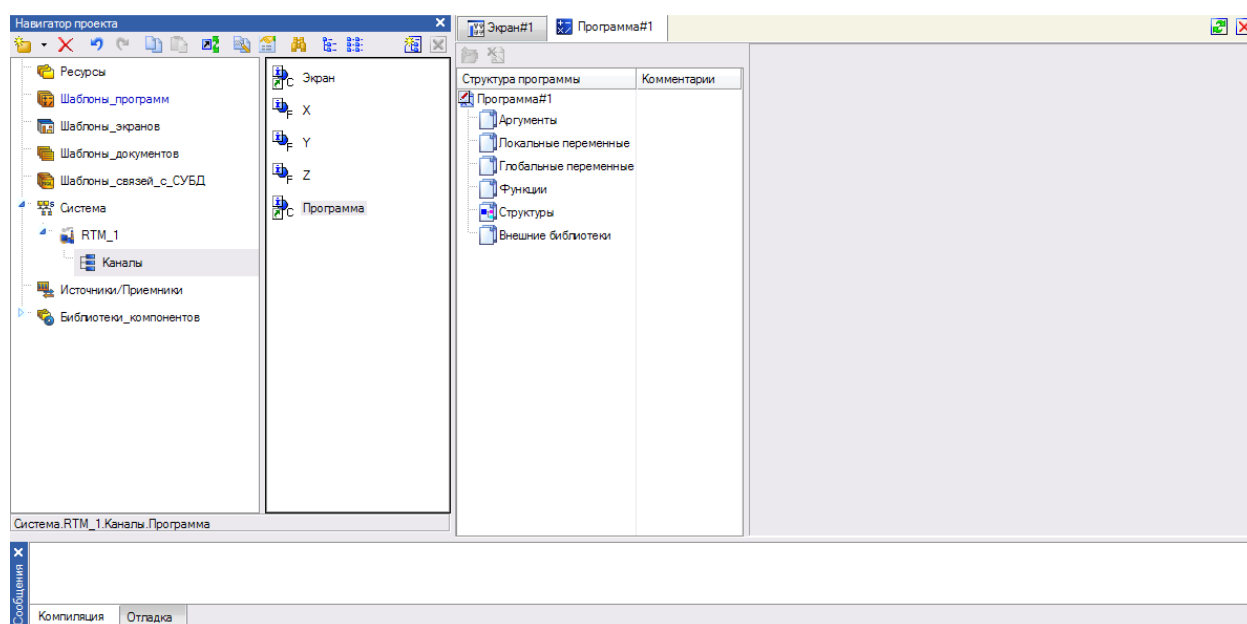


Рис. 2.15. Окно редактирования программы

Для создания аргументов необходимо выделить строку *аргументы* в структуре программы. Откроется окно, изображенное на рис. 2.5. Создание аргументов описано выше (см. привязка аргументов). Для создания локальной или глобальной переменной необходимо выделить соответствующую строку в структуре программы. Откроется окно аналогичное созданию аргументов (аналогично рис. 2.5). Переменные создаются аналогично созданию аргументов. Отличие создания переменных заключается в невозможности привязки переменной к значению (атрибуту) канала.

Для написания программы необходимо выбрать строку программа в структуре программы. Будет предложен выбор языка(рис. 2.16). Следует выбрать необходимый язык и приступит к написанию программы.

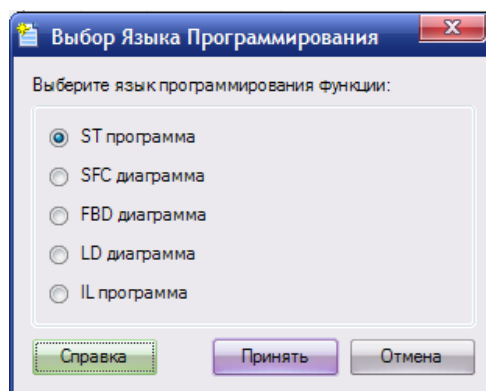


Рис. 2.16 Выбор языка

Рассмотрим подробнее языки программирования Texno St и Texno FBD, TexnoIL, TexnoSFC..

Язык 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, end_case, end_for, end_function, end_function_block, end_if, end_program, end_repeat, end_struct, end_type, end_var, end_while, exit, false, for, function, function_block, 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, uint, uint, until, usint, var, var_arg, var_global, var_inout, var_input, var_output, while, word, xor.

В качестве разделителей используются лексемы: + - * ** / < <= <> << > >= >> ! != = == : := & | ^ ~ % () [] . .. , ; .

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

В данном языке вводятся понятие выражения и предложения.

Выражение— последовательность операндов, разделителей и символьных операторов, задающая вычисление без присвоения результата.

Предложение— последовательность лексем, определяющая выполнение логически законченного промежуточного действия. Предложениями являются присвоение переменной результата каких-либо вычислений, вызов функции, операторы.

В конце предложения обязательно должен стоять символ «;». Исключением является определение переменной.

Программа на языке Техно ST можно представить в виде структуры:

PROGRAM

{ описание аргументов }

{ список предложений }

END_PROGRAM

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 Техно ST позволяет создавать константы. Рассмотрим их представление. Можно создать числовые и строковые константы. 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. 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>. 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>. 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>. К примеру, 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 2.6, побитовые— в таблице 2.7, операторы сравнения— в таблице 2.8, логические— в таблице 2.9.

Таблица 2.6

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 2.7

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
rot	Циклический сдвиг вправо на указанное число разрядов

Таблица 2.8

Операторы сравнения

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

Таблица 2.9

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}

и

{операнд} := {выражение}

В качестве операнда может использоваться имя переменной, массива, уточненное имя переменной объекта.

В таблице 5 приводится приоритет символьных операторов. Первыми будут выполняться выражения, заключенные в скобках, а затем в порядке возрастания номера строки в таблице 2.10.

Таблица 2.10

Приоритет символьных операторов

1	**
2	!, not, -, унарный −, унарный +
3	<<, >>, shl, shr, rol, ror
4	&, , ^, xor
5	*, /, %, mod
6	+, -
7	==, !=, <>, <, <=, >, >=
8	&&, and
9	, or
10	=, :=

В ST- программе можно использовать стандартные функции языка C: sin, cos, tan, asin, exp, log. Для работы с портом и каналом есть специальные функции в среде Trace Mode, о которых можно найти информацию в справке системы.

Помимо символьных операторов есть следующие операторы:

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 Техно 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 отсутствии 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;
```

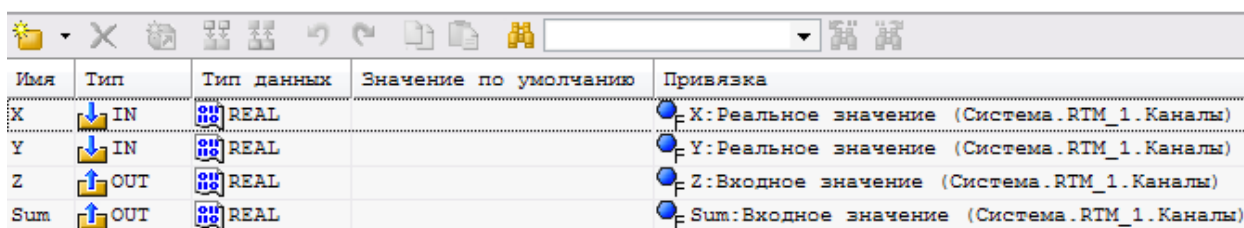
Данный оператор сперва присваивает переменной цикла с указанным именем результат вычисления выражения 1, выполняет последовательность предложений, если вычисленная переменная цикла не превысит значение

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 переменной 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.

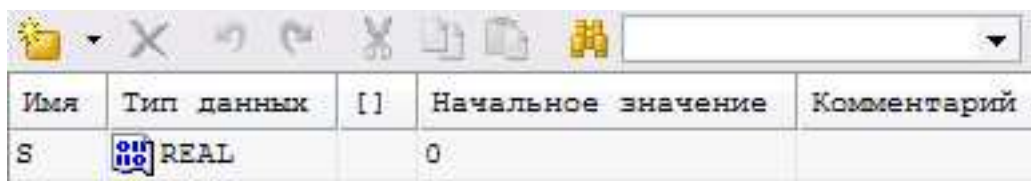
Оператор continue служит для перехода в конец цикла. При его вызове все следующие за ним до конца цикла предложения не выполняются.

Рассмотрим создание программы на языке Техно ST. Допустим, созданы аргументы, глобальная переменная (рис. 2.17, 2.18) После выбора языка откроется окно, изображенное на рис. 2.19. Все созданные аргументы, и локальные переменные будут описаны в начале программы. Глобальные переменные в отличие от локальных в самой программе не описываются, но без проблем могут использоваться как операнды.



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)

Рис. 2.17 Аргументы программы



Имя	Тип данных	[]	Начальное значение	Комментарий
S	REAL		0	

Рис. 2.18 Глобальные переменные программы

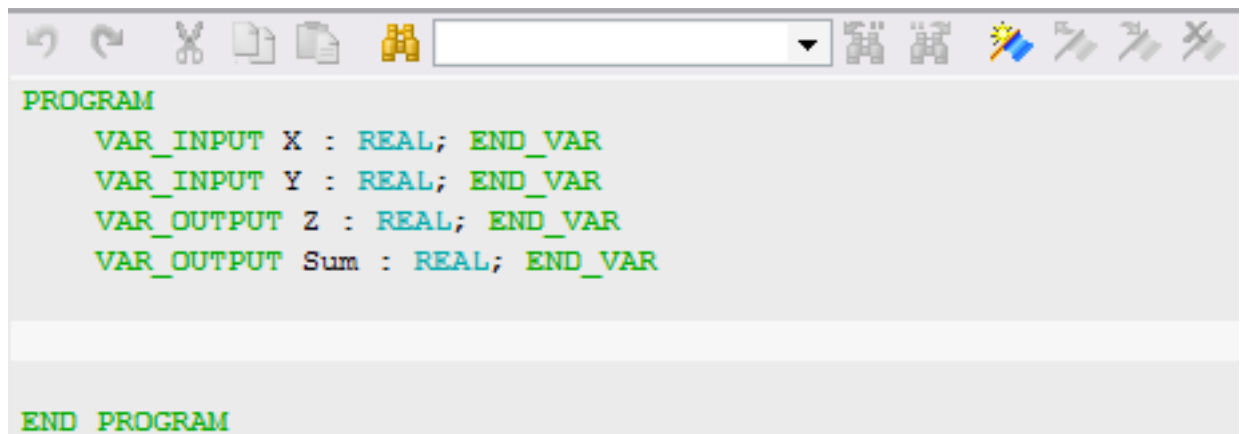


Рис. 2.19 Окно для создания программы

Произведем деление аргумента X на аргумент Y , произведя проверку на равенство Y нулю. Результат присвоим аргументу Z . Аргументу Sum присваивается сумма всех результатов деления. Программа будет следующая:

```
PROGRAM

VAR_INPUT X : REAL; END_VAR

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR



VAR_INOUT Sum : REAL; END_VAR

if Y == 0 then Z = X / 1e-9;
    else Z = X / Y;
end_if;

Sum = S+ Z;

S = Sum;

END_PROGRAM
```

После того, как написали текст программы необходимо проверить ее. Для этого нажмем на иконку «компиляция» . В окне «сообщения» будет выведен результат компиляции. Если окно «сообщения» закрыто, то следует щелкнуть по иконке  или выбрать в меню «вид» окно сообщения. Если, к примеру, не хватает запятой в программе, компилятор сообщит об ошибке (рис. 2.20). При отсутствии ошибок

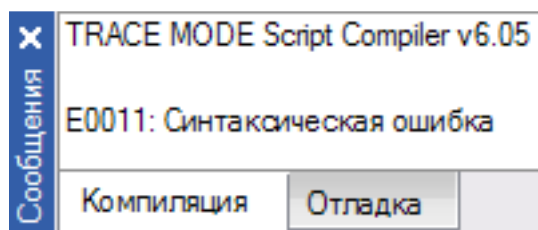


Рис. 2.20 Пример ошибки

При отсутствии ошибок будет написано: «*Программа#1.tms compiled successfully*», к примеру.

Язык Техно FBD

Рассмотрим FBD программу. Данная программа представляет собой совокупность функциональных блоков, которые соединены между собой.

Функциональный блок— изображение вызова функции Техно ST. В качестве примера рассмотрим функциональный блок, производящий сложение. Изображение его приведено на рис. 2.21.

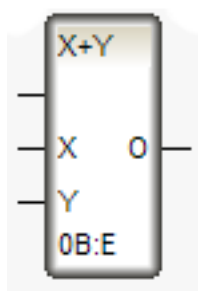



Рис. 2.21 Функциональный блок Техно ST

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 указывается 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 (рис. 2.22). 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.

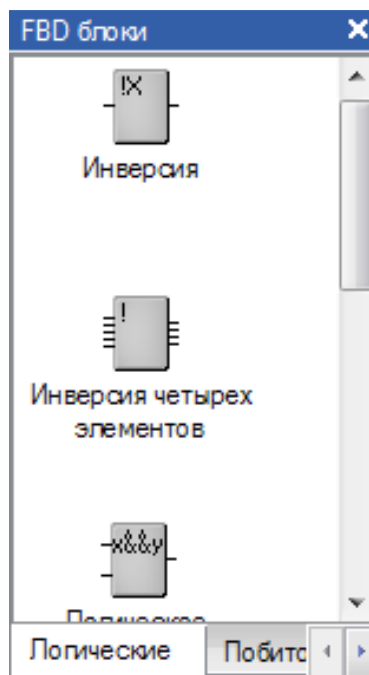


Рис. 2.22 Окно FBD блоки

Для соединения выхода одного функционального блока с входом другого следует привести 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 (рис. 2.23).

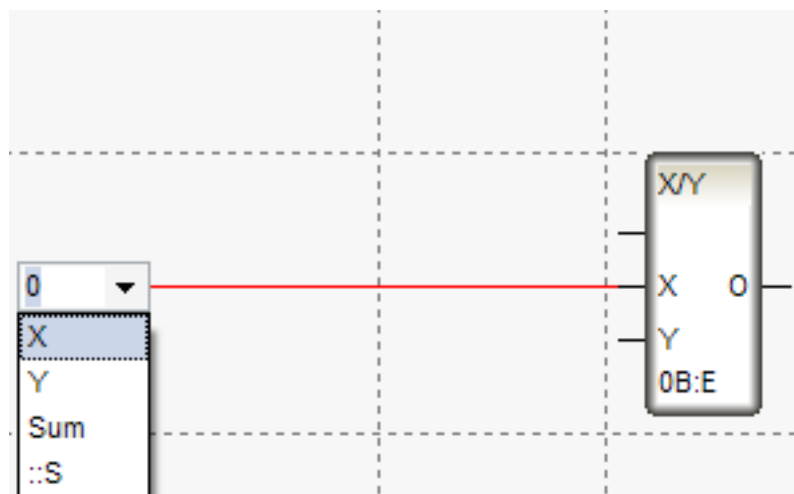


Рис. 2.23 Привязка входа (выхода) функционального блока

Создание константы рассмотрим на примере создания константы равной $1e-9$. Для этого следует выделить вход, на который необходимо подать константу, вызываем контекстное меню и выбираем **привязать**. Будет выведено окно, как указано на рис. 2.23. В появившемся 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 Техно ST, используя язык Техно FBD (рис. 24).

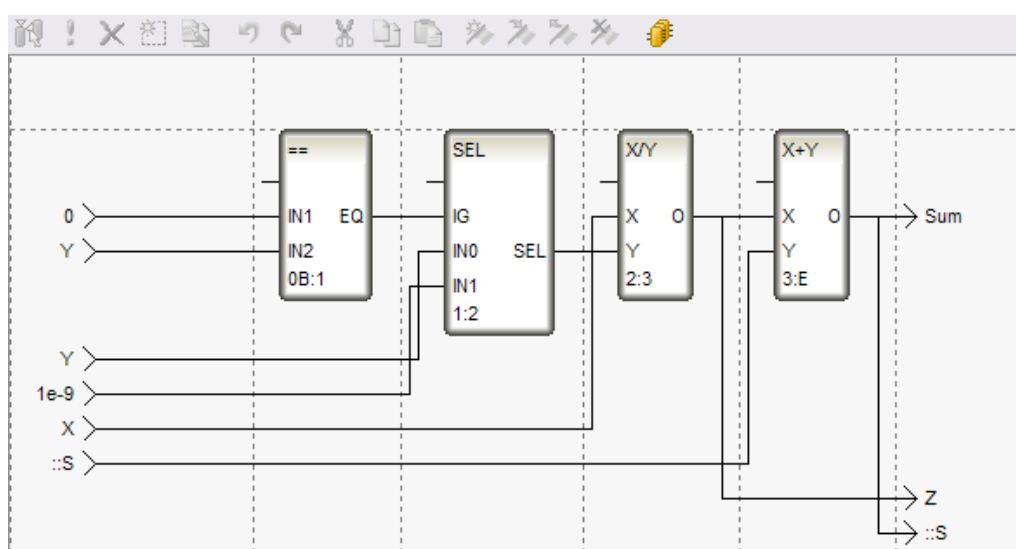


Рис. 2.24 Пример FBD программы

ЯзыкТехно SFC

SFC язык позволяет создавать программу, оперируя шагами и переходами. Под шагом следует понимать подпрограмму, написанную на одном из языков, доступных в среде Trace Mode, и выполняющую то или иное действие. Переход— условие, при выполнении которого выполняется определенный шаг. На рис. 2.25 изображены условные графические изображения шага и перехода

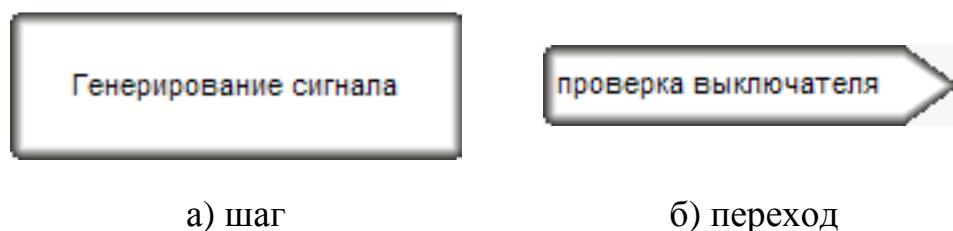


Рис. 2.25 Условные графические изображения шага и перехода

Пример SFC программы приведен на рис. 2.26. Направление перехода от одного шага к другому указывается линией и стрелкой. Последующий шаг выполняется в случае, если выполняется условие перехода. Линия со стрелкой, соединяющая переход с линией, отображающей направление перехода, указывает на шаг, который выполняется, если условие перехода выполняется.

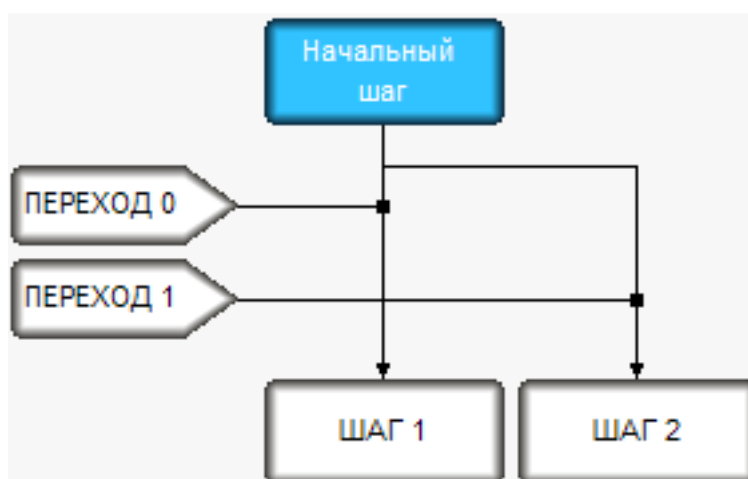


Рис. 2.26 SFC программа

Каждый шаг и переход можно выделить. Выделенный шаг (переход) отображается синим цветом. При выделении перехода выделяется дополнительно линия перехода, на которую распространяется условие

перехода. Можно выделить линию перехода. Выделенная линия изображается красным цветом. Линия перехода выделяется вместе с переходом, действующим на данной линии перехода и расположенной выше других переходов на выделенной линии перехода. Шаг (переход) можно переименовать, выделив его и дважды щелкнув левой клавишей мыши по нему, после чего можно ввести новое имя шага (перехода).

Рассмотрим подробнее создание SFC диаграммы. Объект программа, аргументы и переменные программы создаются аналогично другим программам. Окно для создания SFC диаграммы представлено на рис. 2.27.

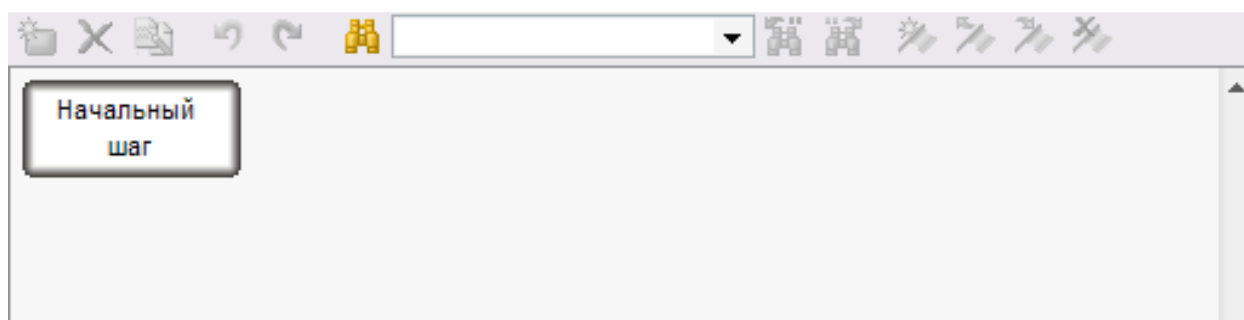


Рис. 2.27 Окно создания SFC диаграммы

Видно из рис. 2.27, что в новой SFC диаграмме присутствует один единственный шаг. Для создания нового шага надо выделить шаг, предшествующий ему. Можно либо щелкнуть левой клавишей мыши на иконке 📁 или вызвать контекстное меню и выбрать **создать шаг/переход**. Будет создан новый шаг, выполняемый после выделенного ранее шага, и переход, определяющий условие перехода к вновь созданному шагу. Результат создания нового шага приведен на рис. 2.28, а. Если существовал шаг, выполняемый после выделенного блока, то произойдет ветвление после создания нового шага (рис. 2.28, б).

В ряде случаев необходимо заикливание. Произведем заикливание начального шага (рис. 2.28, а). Шаг 1 будет выполняться после выхода из цикла. Для заикливания наведем курсор на нижний край блока, которым завершается тело цикла (в данном случае нижний край начального шага), нажмем на левую клавишу мыши. Наведем курсор мыши на верхний край блока, с которого начинается тело цикла (в данном случае верхний край начального шага) и отпустим клавишу мыши. Если все было правильно сделано, получится цикл (рис. 2.29). Условием выполнения цикла выступает переход, созданный вместе с циклом.



а) новый шаг

б) новый шаг с ветвлением

Рис. 2.28 Результат создания нового шага

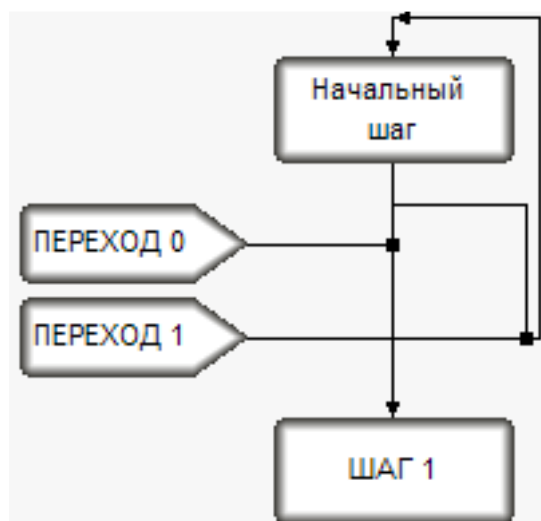



Рис. 2.29 Цикл на SFC диаграмме

Рассмотрим создание параллельного шага. Для создания шага, выполняемого параллельно какому-либо определенному шагу (параллельно шагу1, к примеру) следует выделить переход или линию выходящую из него, который определяет условие выполнения параллельных шагов (в данном случае переход 0— условие выполнение шага 1), щелкнуть левой клавишей мыши на иконке 🌟 или вызвать контекстное меню и выбрать **создать шаг/переход**. Будет создан новый шаг (рис. 2.30), выполняемый параллельно другому шагу, переход к которому был выделен при создании нового шага.

Все параллельные шаги должны быть связаны с одним и тем же последующим переходом. Допустим, создан шаг 3, выполняемый после шага

1. Тогда необходимо нажать левую клавишу мыши на нижнем крае шага, параллельного шагу 1 (шаг 2), переместить курсор, не отпуская клавиши мыши, на переход, определяющий условие вызова шага 3 или на линию перехода, отпустить клавишу мыши. Если все было правильно сделано, шаги 1 и 2 будут подключены параллельно.

После создания диаграммы необходимо описать каждый шаг или переход (произвести редактирование шага или перехода). Каждый шаг и переход— программа, написанная на одном из языков среды Trace Mode. Для редактирования шага (перехода) необходимо выделить соответствующий шаг (переход), щелкнуть левой клавишей мыши на иконке  или вызвать контекстное меню и выбрать **редактировать**. Будет предложен выбор программ, среди которых следует выбрать язык программирования, на котором будет описан данный шаг (условие перехода). Можно раскрыть раздел **SFC диаграмма** в **навигаторе проекта**. Появится раздел **шаги** и **переходы**, которые содержат все шаги и переходы программы. Раскрыв данные разделы и выбрав необходимый шаг (переход) можно произвести редактирование шага (перехода). После выбора шага или перехода в разделе **шаги** или **переходы** будет предложен выбор языка. После выбора языка можно приступить к написанию подпрограммы.

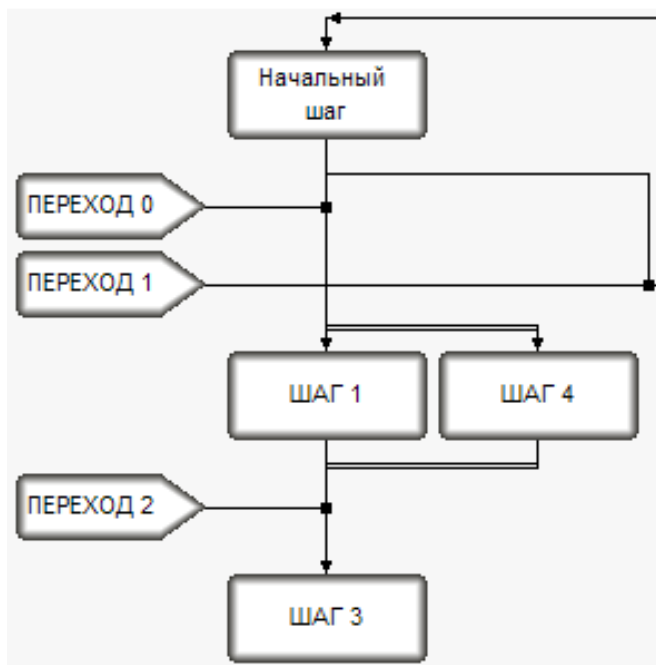
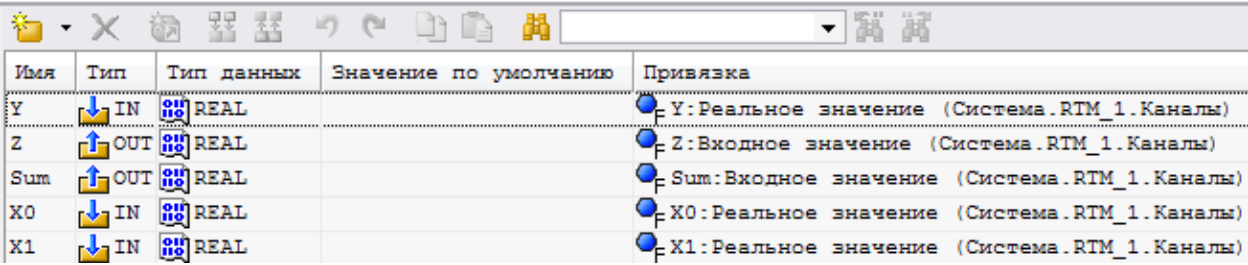


Рис. 2.30 Созданный параллельный шаг

В качестве примера рассмотрим деление суммы аргументов X0 и X1 на аргумент Y с проверкой на равенство аргумента Y нулю. В случае равенства

аргумента Y нулю произведем деление на константу 10^{-9} , результат деления присвоим аргументу Z. Программы будет также вычислять сумму всех результатов деления Sum. Первый шаг будет заключаться в нахождении суммы аргументов X0 и X1. Второй шаг— деление на Z, при условии перехода: Z не равен нулю. Третий шаг— деление на константу, 10^{-9} при условии: Z равен нулю. Аргументы программы приведены на рис. 2.31. Поскольку необходимо передать результат сложения из первого шага, выполняющего сложение X0 и X1, то создадим локальную переменную шаг1 (рис. 2.32). Так как необходимо найти суммы всех вычисленных отношений, создадим глобальную переменную S, хранящую предыдущую сумму всех найденных отношений (рис. 2.33).



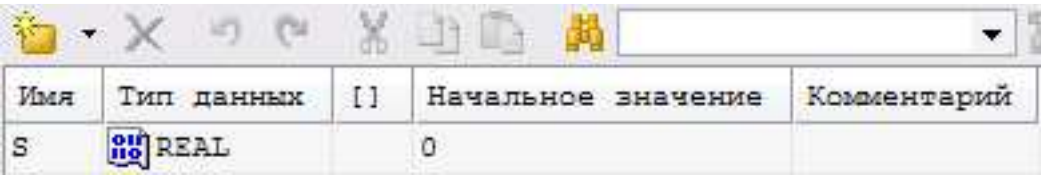
Имя	Тип	Тип данных	Значение по умолчанию	Привязка
Y	IN	REAL		Y: Реальное значение (Система.RTM_1.Каналы)
Z	OUT	REAL		Z: Входное значение (Система.RTM_1.Каналы)
Sum	OUT	REAL		Sum: Входное значение (Система.RTM_1.Каналы)
X0	IN	REAL		X0: Реальное значение (Система.RTM_1.Каналы)
X1	IN	REAL		X1: Реальное значение (Система.RTM_1.Каналы)

Рис. 2.31 Аргументы программы примера



Имя	Тип данных	[]	Начальное значение	Комментарий
Шаг1	REAL		0	

Рис. 2.32 Локальная переменная примера



Имя	Тип данных	[]	Начальное значение	Комментарий
S	REAL		0	

Рис. 2.33 Глобальная переменная примера

SFC диаграмма изображена на рис. 2.34. Каждый шаг и переход выполним, используя язык Техно ST.

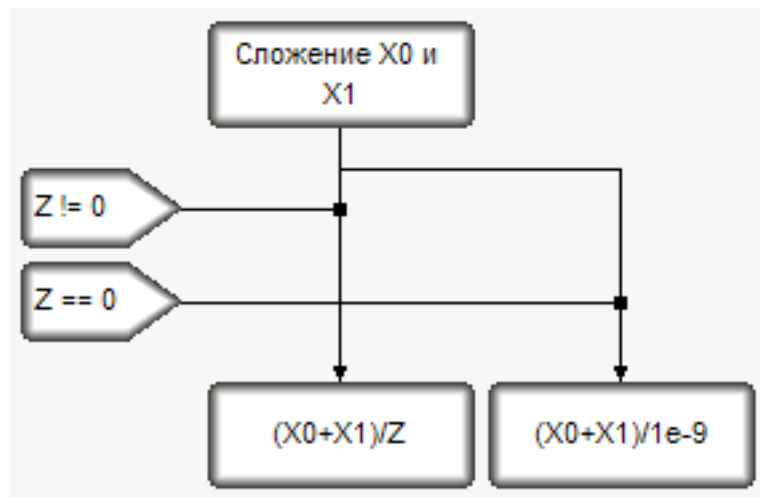


Рис. 2.33 SFC диаграмма примера

Первый шаг, производящий сложение аргументов X0 и X1 примет вид:

SFC_STEP "Сложение X0 и X1"

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR Шаг1 : REAL := 0; END_VAR

Шаг1 = X0 + X1;

END_SFC_STEP

Переход к шагу, вычисляющему деление на Z примет вид:

SFC_TRANSITION "Z != 0" FROM(INITIAL_STEP) TO(STEP_1)

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR Шаг1 : REAL := 0; END_VAR

Z != 0

END_SFC_TRANSITION

Внимание: следует обратить внимание на то, что после записанного условия точка с запятой не ставится!

Сам шаг, производящий деление на Z примет вид:

SFC_STEP "(X0+X1)/Z"

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR Шаг1 : REAL := 0; END_VAR

Sum = S + Шаг1 / Z;

S = Sum;

END_SFC_STEP

Переход к шагу, производящему деление на константу 10^{-9} примет вид:

```

SFC_TRANSITION "Z == 0" FROM( INITIAL_STEP ) TO( STEP_2 )

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR IHar1 : REAL := 0; END_VAR


Z == 0


END_SFC_TRANSITION

Шаг, производящий деление на константу  $10^{-9}$  примет вид:

SFC_STEP "(X0+X1)/1e-9"

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR IHar1 : REAL := 0; END_VAR


Sum = S + IHar1 / 1e-9;

S = Sum;


END_SFC_STEP

```

Язык Техно II

Программа на языке Техно II— последовательность инструкций. Каждая инструкция начинается с новой строки, содержит оператор с модификатором, в случае некоторых операторов приводится один или более операндов. Компилятор не чувствителен к регистру. Инструкции ADD 10 15 и Add 10 15 равнозначны.

В данном языке вводится понятие аккумулятора, под которым следует понимать хранилище текущего результата.

В качестве операндов могут выступать:

1. переменные;
2. константы (см. язык Техно ST);
3. имя метки;
4. имя функции.

Под модификатором следует понимать литеры N, C, X, которые приписываются справа к имени некоторых операторов. Модификатор N обозначает логическое отрицание операнда, C обозначает, что инструкция выполняется, если результат предыдущей операции сравнения истинен, модификатор X указывает на то, что инструкция выполняется, если аккумулятор содержит значение true.

Пример использования модификатора:

OR b//аккумулятор ИЛИ b

ORN b// аккумулятор ИЛИ НЕ b

Операторы, которые предполагают наличие одного или двух операндов (арифметические, логические операции) допускают опустить первый операнд, тогда в качестве первого операнда выступает значение, хранящееся в аккумуляторе. Случай, когда используются оба операнда называется двухадресным режимом, случай, когда первый операнд заменяется аккумулятором— одноадресным режимом.

Пример двухадресного режима:

ADD 10 15//сложение 10 и 15

Пример того же сложения при одноадресном режиме:

LD 10//присвоение аккумулятору числа 10

ADD 15//сложение 10 и 15

Рассмотрим операторы языка Техно IL. Операторы для обмена с аккумулятором приведены в таблице 2.11.

Таблица 2.11

Операторы для обмена с аккумулятором

Обозначение оператора	Синтаксис оператора	Допустимые модификаторы	Действие
LD	LD {операнд}	N	присваивает аккумулятору значение операнда
ST	ST {операнд}	N	присваивает операнду значение аккумулятора

Пример обмена данными с аккумулятором:

LD X//присвоение аккумулятору операнда X

MULT K//Умножение X на K

ST X//присвоение операнду X значения, хранящегося в аккумуляторе.

Логические операторы приведены в таблице 2.12

Таблица 2.12

Логические операторы

Обозначение оператора	Синтаксис оператора	Допустимые модификаторы	Действие
S	S {операнд}	—	присваивает операнду значение

			true
R	R {операнд}	—	присваивает операнду значение false
AND	AND {операнд 1} {операнд 2}	N	логическое «И»
OR	OR {операнд 1} {операнд 2}	N	логическое «ИЛИ»
XOR	XOR {операнд 1} {операнд 2}	N	Логическое «исключающее ИЛИ»

Пример логических операций:

S X//присвоение операнду X значения true

R Y//присвоение операнду Y значения false

AND X Y//X «И» Y

Арифметические операторы приведены в таблице 2.13.

Таблица 2.13

Арифметические операторы

Обозначение оператора	Синтаксис оператора	Допустимые модификаторы	Действие
ADD	ADD {операнд 1} {операнд 2}	—	сложение операндов
SUB	SUB {операнд 1} {операнд 2}	—	вычитание операнда 2 из операнда 1

MUL	MUL {операнд 1} {операнд 2}	—	произведение операндов
DIV	DIV {операнд 1} {операнд 2}	—	Деление операнда 1 на операнд 2

Пример выполнения арифметических операций:

MUL X K

ADD C

Операторы сравнения приведены в таблице 2.14.

Таблица 2.14

Операторы сравнения

Обозначение оператора	Синтаксис	Допустимые модификаторы	Действие
GT	GT {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 больше операнда 2
GE	GE {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 не меньше операнда 2
EQ	EQ {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 равен операнду 2
NE	NE {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 не равен операнду 2
LE	LE {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 не больше операнда 2

LT	LT {операнд 1} {операнд 2}	—	возвращает true, если операнд 1 меньше операнда 2
----	-------------------------------	---	---

Операторы перехода и вызова функций приведены в таблице 2.15.

Таблица 2.15

Операторы перехода и вызова

Обозначение оператора	Синтаксис	Допустимые модификаторы	Действие
JMP	JMP {метка}	C, X	переход к строке с указанной меткой
CAL	CAL {имя функции (параметры)}	C, X	Вызов функции
RET	RET	C, X	Выход из программы

Возможны следующие модификации операторов перехода и вызова:

JMPC— условный переход, выполняемый если результат предыдущей операции сравнения истинен;

JMPX— условный переход, выполняемый если аккумулятор содержит значение true;

CALC— условный вызов функции, выполняемый если результат предыдущего сравнения истинен;

CALX— условный вызов функции, выполняемый если аккумулятор содержит значение true

RETC— условный выход из программы, выполняемый если результат предыдущей операции сравнения истинен;

RETХ— условный выход из программы, выполняемый, когда аккумулятор содержит значение true.

Пример выполнения сравнения и перехода:

EQ Z 0//проверка на равенство Z нулю

JMPC label//при равенстве нулю переход к строке с меткой label

...

label: LD 1e-9//присвоение аргументу Z значения 10^{-9}

ST Z

Рассмотрим реализацию примера деления, приведенного на языке Техно SFC, с использованием языка Техно IL. Поскольку существует оператор S в языке Техно IL, глобальную переменную переименуем с S на Sum0. Тогда программа на языке Техно IL примет вид:

PROGRAM

VAR_INPUT Y : REAL; END_VAR

VAR_OUTPUT Z : REAL; END_VAR

VAR_OUTPUT Sum : REAL; END_VAR

VAR_INPUT X0 : REAL; END_VAR

VAR_INPUT X1 : REAL; END_VAR

VAR Шаг1 : REAL := 0; END_VAR

ADD X0 X1//Сложение аргументов

ST Шаг1//присвоение результата сложения переменной Шаг1

NE Y 0//Проверка на равенство нулю

JMPC Деление_на_Y//Переход к делению на Z, если Z не равен нулю

//Деление на 1e-9

DIV Шаг1 1e-9

ST Z//присвоение аргументу Z вычисленного отношения

JMP Сложение//Переход к сложению

Деление_на_Y: DIV Шаг1 Y//деление на Z

ST Z//присвоение аргументу Z вычисленного отношения

Сложение: ADD Z Sum0//Получение суммы всех результатов отношения

ST Sum0//присвоение суммы всех результатов сложения аргументу Sumи глобальное переменной S и глобальное переменной S

ST Sum

END_PROGRAM

Отчет тревог Trace Mode

Для SDADA–системы Trace Mode характерна возможность создания отчета тревог. Отчет тревог— текстовый документ, который содержит сообщения, генерируемые при различных ситуациях в процессе работе Trace Mode.

В отчете могут быть записаны тревоги следующих видов:

1. системные сообщения;
2. сообщения по каналам;
3. сообщения, генерируемые с помощью системной переменной @Message;
4. интерактивные сообщения оператора.

Строка сообщения в отчете тревог содержит поля, разделенные пробелом:

Date Time Category Name Coding Text UserID T_ack N

Можно выделить следующие поля:

Date— дата события;

Time— время события;

Category— категория сообщения. Можно выделить следующие категории сообщений:

1. <>— без категории;
2. <M>— сообщение;
3. <W>— предупреждение;
4. <E>— ошибка;
5. <I>— информация;
6. <A>— тревога;
7. <R>— изменение атрибутов;
8. <S>— пользовательское
9. <_>— невидимое (не передается в графику)
10. <->— неквитируемое;
11. <!>— командное;
12. <?>— резерв

Name— в случае сообщений канала— имя канала; при системном сообщении— <имя файла prj без разрешения>_<порядковый номер узла>

Coding— кодировка канала;

Text— текст сообщения или пользовательский комментарий;

UserID— идентификатор пользователя;

T_ack— время квитирования сообщения в формате DD_HH_MM:SS;

N— порядковый номер строки.

Системными сообщениями являются:

1. события, связанные с каналами:

- 1.1 Error— установка каналу признака аппаратной недостоверности;
- 1.2 Login— корректная авторизация при старте;
- 1.3 Logout— корректное окончание сеанса пользователем;
- 1.4 Failed Login— некорректная авторизация при старте;
- 1.5 Failed Logout— некорректная авторизация при окончании сеанса пользователем;

2. события, не связанные с каналами:

- 2.1 Start— запуск монитора реального времени;
- 2.2 Continue— невозможность выполнения файловой операции;
- 2.3 Stop— останов монитора реального времени;

3. прочие события— неопределенное событие.

Для создания отчета надо настроить анализ границ канала при его редактировании. У любого канала, содержащего числовые данные можно задать следующие границы:

ВП (HL)— значение верхнего предела;

ВА (HA)— значение верхней аварийной границы;

ВГ (HW)— значение верхней предупредительной границы;

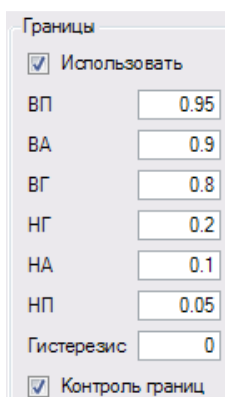
НГ (LW)— значение нижней предупредительной границы;

НА (LA)— значение нижней аварийной границы;

НП (LL)— значение нижнего предела;

Гистерезис.

Если в окне редактирования канала установлен флаг **использовать** на панели границы, то система Trace Mode будет производить анализ границ каналов (рис.2.34).



Границы	
<input checked="" type="checkbox"/> Использовать	
ВП	0.95
ВА	0.9
ВГ	0.8
НГ	0.2
НА	0.1
НП	0.05
Гистерезис	0
<input checked="" type="checkbox"/> Контроль границ	

Рис. 2.34

Флаг **контроль границ** на той же панели окна редактирования разрешает установку признака программной достоверности. Так если данный флаг установлен и значение канала лежит в диапазоне [НП;ВП], то атрибут данного канала примет значение 1, что указывает на достоверность данных, если значение канала выйдет за пределы указанного диапазона, атрибут **программная достоверность** примет значение 0, что будет указывать на недостоверность данных. Таким образом, диапазон (ВП; ∞) и $(-\infty; НП)$ задают недостоверные значения канала, которые могут быть получены при неисправности датчиковой аппаратуры или устройств согласования. При недостоверности данных и установленном флаге **программная достоверность** в результате клиппирования, канал будет возвращать предельное значение НП или ВП.

При разрешенном анализе, когда значение канала лежит в диапазоне (ВА; ВП] или [НП; НА) генерируется сообщение об аварии. Если значение канала лежит в диапазоне [ВА; ВГ) или (НГ; НА] генерируется сообщение, предупреждающее о близости к аварийной границе. Когда значение лежит в диапазоне [НГ; ВГ] генерируется сообщение о нормальном протекании процесса.

При небольших колебаниях параметра вблизи одной из границ будет создаваться большое количество сообщений при многократных переходах через границу. Колебаний могут быть настолько малы, что переходом через

границы можно пренебречь и генерируемые сообщения будут только мешать, тогда следует использовать гистерезис, который позволяет исключить подобное ненужное генерирование сообщений. Переход в сторону развития аварийной ситуации фиксируется при заданных границах. При обратном изменении значения канала граница корректируется на величину установленного гистерезиса в соответствующем направлении. Генерация сообщений будет происходить с использованием скорректированных границ.

Словарь сообщений (рис. 2.35) позволяет установить соответствие между диапазонами, задаваемыми при настройке контроля границ, и сообщениями, которые генерируются при принадлежности тому или иному диапазону, направлением передачи данных о тревоге. В словаре сообщений используются направления. Под направлением следует понимать параметр, задающие направление передачи сообщения.

Направление	Категория	Текст
AR+G	<M> Сообщение	Все в порядке
AR+G	<W> Предупреждение	Выход за верхнюю предупредительную границу
AR+G	<W> Предупреждение	Выход за нижнюю предупредительную границу
AR+G	<A> Тревога	Выход за верхнюю аварийную границу
AR+G	<A> Тревога	Выход за нижнюю аварийную границу
AR+G	<E> Ошибка	Выход за верхний предел
AR+G	<E> Ошибка	Выход за нижний предел

Рис. 2.35 Словарь сообщений

Можно выделить следующие направления передачи сообщений:

1. AR + G + Prn;
2. AR + Prn;
3. AR;
4. AR + G + GSM;
5. AR + GSM;
6. AR + GSM + PRN;
7. AR + G + GSM + PRN;
8. G;
9. AR + Net;
10. AR + G + Net;
11. AR + GSM + Net;
12. Net;
13. Net + GSM;
14. AR + Play;
15. AR + G + Play;
16. AR + Net + Play;
17. Play;
18. AR + G + PlayStop;
19. AR + G + PlayLoopStop.

При задании направления используются следующие обозначения направлений передачи сообщений:

1. AR— в файл тревог;
2. G— в исполнительные модули, которые способны отображать отчет тревог;

3. PRN— на принтер;
4. GSM— на сотовые телефоны в виде SMS;
5. Net— в сеть;
6. Play, PlayStop, PlayLoopStop— воспроизведение файла с именем: {текст сообщения}.wav.

Степень важности сообщения задается категориями, которые указаны выше.

Среда предполагает заполнение таблицы словаря сообщений с помощью окна, изображенного на рис. 2.36. Для заполнения данного окна необходимо дважды щелкнуть левой клавишей мыши на строчке, которую необходимо заполнить или скорректировать.

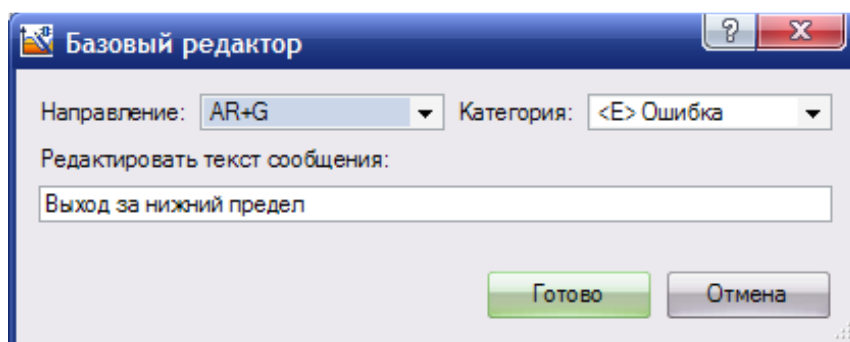


Рис. 2.36 Заполнение словаря сообщений

Для создания отчета тревог необходимо произвести редактирование узла на закладке *отчет тревог/дамп/параметры* (2.37). В поле *максимум записей* указывается предельное количество записей в файле отчета тревог. При переполнении файла отчета тревог происходит запись новых тревог с начала файла отчета начиная со второй строки. Поле *формат даты* задает формат записи даты и времени генерирования тревоги в файле отчета тревог. В поле состояние следует выбрать *true*.

Отчет тревог

Имя файла:

Максимум записей: Состояние:

Формат даты: Считывать при старте:

Рис. 2.37 Редактирование узла

При задании формата времени и даты можно использовать форматы приведенные в таблице 2.16.

Таблица 2.16

Форматы вывода даты и времени

Обознач.	Формат
%a	сокращенное наименование дня недели
%A	полное наименование дня недели
%b	сокращенное наименование месяца
%B	Полное наименование месяца
%c	дата и время в соответствии с региональными настройками ОС
%d	день месяца как целое число
%H	часы в формате (00—23)
%I	часы в формате (01— 12)
%j	день года как целое число (001 до 366)
%m	месяц как целое число (01— 12)
%M	минуты как целое число (00— 59)
%p	индикатор AM/PM для часов в формате 01— 12

%S	секунды как целое число
%U	неделя как целое число (00— 53); первый день недели— воскресенье
%w	день недели как целое число (0—6, 0— воскресенье)
%W	неделя как целое число (00—53); первый день недели— понедельник
%x	дата в соответствии с региональными настройками ОС
%y	год без века как целое число (00— 99)
%Y	год с веком как целое число (0000— 9999)

Для установления связи между созданным словарем сообщений и каналом следует произвести редактирование канала. На закладке архивация следует поставить флажок **отчет тревог**, в поле индекс аварийного словаря выбрать нужный словарь сообщений.

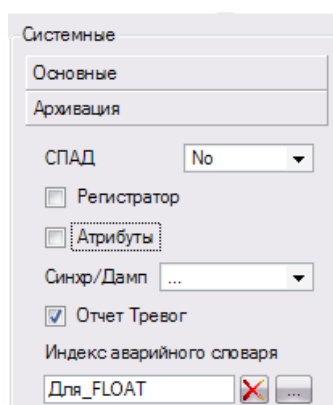


Рис. 2.38 Редактирование канала

СПАД архив

Монитор реального времени программы Trace Mode поддерживает запись значений атрибутов канала в СПАД архивы (база данных реального времени).

Основные характеристики СПАД архива:

- точность значения времени— 1 мс;
- скорость записи в архив для рабочей станции с процессором Pentium-4 с тактовой частотой 2 ГГц— свыше 600 000 параметров в секунду.

Архивные данные могут использоваться монитором реального времени, экспортироваться в файл, отображаться графически.

Каждое сообщение в архиве содержит идентификаторы канала и атрибута, новое значение атрибута и время его изменения, а также некоторые другие параметры.

Настройка архива производится при редактировании узла и канала. При редактировании узла (рис. 2.39) на закладке **архивы** можно задать имя файла, в котором будет создан архив, размер файла, период сохранения, ограничить размер очереди на запись в архив. Для успешной работы с архивом следует установить значение *состояние архива* равным *true*.

СПАД 1

Имя файла:

Состояние архива:

Размер файла, Мб:

Кэш

Период сохранения: Размер, Мб:

Максимум очереди записей:

Рис. 2.39 Редактирование узла

При редактировании канала (рис. 2.40) на закладке **архивация** следует выбрать в поле **СПАД** номер архива, который настраивался при редактировании узла. Для разрешения записи атрибутов канала в архив

следует поставить флажок *атрибуты*. Флажок *регистратор* разрешает архивацию атрибутов в регистратор.

Глобальный регистратор служит для обеспечения надежного хранения архивов. Он архивирует данные, обеспечивает автоматическое восстановление данных после сбоя. Флаг *атрибуты* разрешает/запрещает запись в архив сообщения об изменении границ и атрибутов обработки канала помимо сообщения об изменении атрибута **R** (реальное значение) канала. Так если флаг атрибуты не установлен, то в архив заносятся сообщения об изменении атрибута **R** канала, границ, атрибутов обработки канала. Если же флаг атрибуты не установлен, то в архив заносятся только сообщения об изменении атрибута **R** канала.

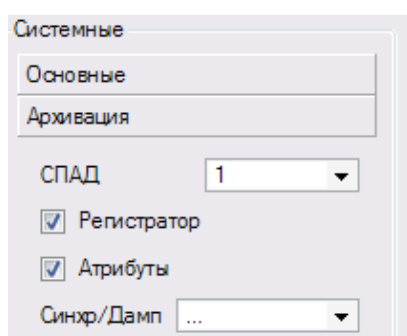


Рис. 2.40 Редактирование канала

Глава III Лабораторные работы

Установка системы Trace Mode 6

Запустите файл setup1.msi. Будет запущена инсталляция программы. В появившемся окне (рис. 3.1) щелкните левой клавишей мыши по кнопке *Next*.

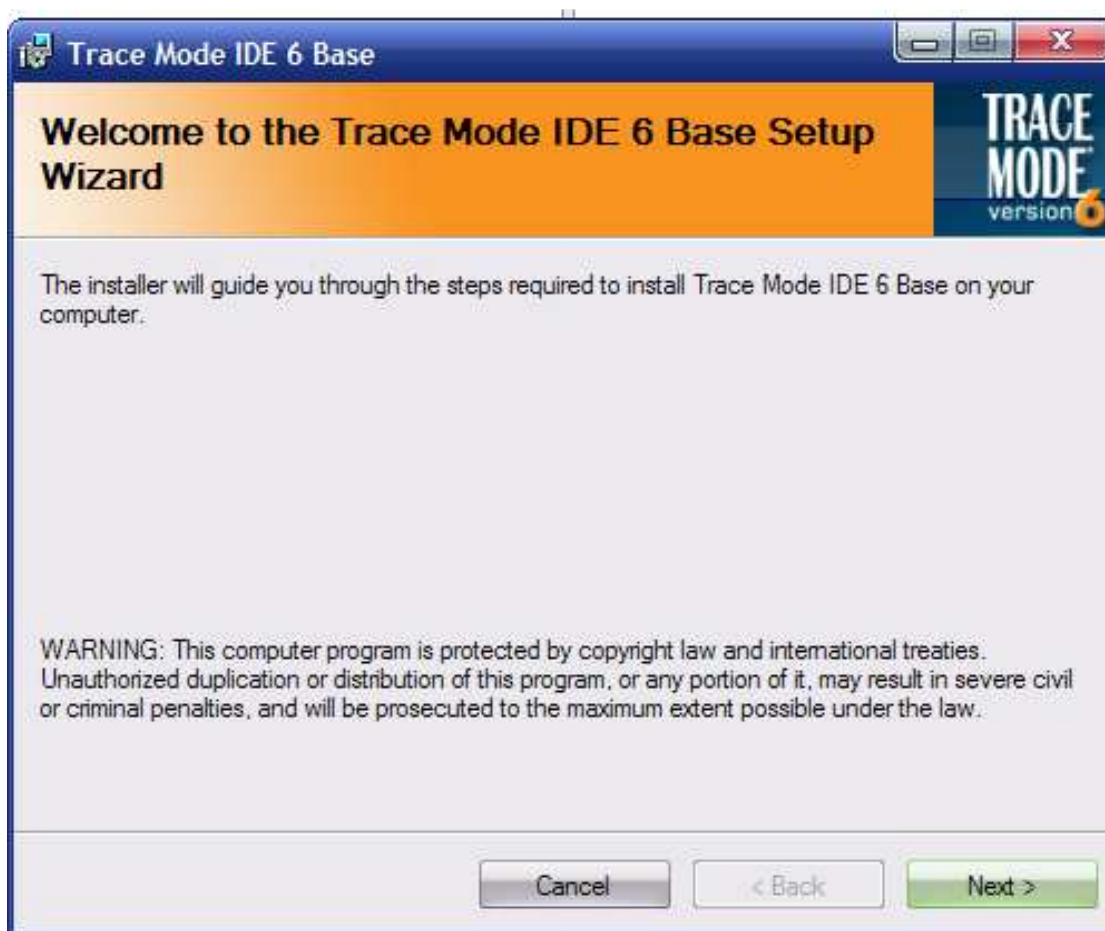


Рис. 3.1

Будет предложено лицензионное соглашение. Познакомившись с лицензионным соглашением, выберите *I agree* и щелкните левой клавишей мыши по кнопке *Next* для продолжения установки (рис. 3.2).



Рис. 3.2

Среди предложенных стран выберите Россию (рис. 3.3).

Укажите директорию, куда необходимо установить программное обеспечение. Выберите **Everyone** (все будут использовать программу) или **Just me** (программу будет использовать только один пользователь). Щелкните левой клавишей мыши по кнопке **Next** (рис. 3.4).

Введите сведения о пользователе: имя пользователя, название организации. Снова щелкните левой клавишей мыши по кнопке **Next** (рис. 3.5). Начнется установка программного обеспечения. По завершению установки откроется окно, указанное на рис. 3.6. Для завершения установки щелкните левой клавишей мыши по кнопке **Close**.

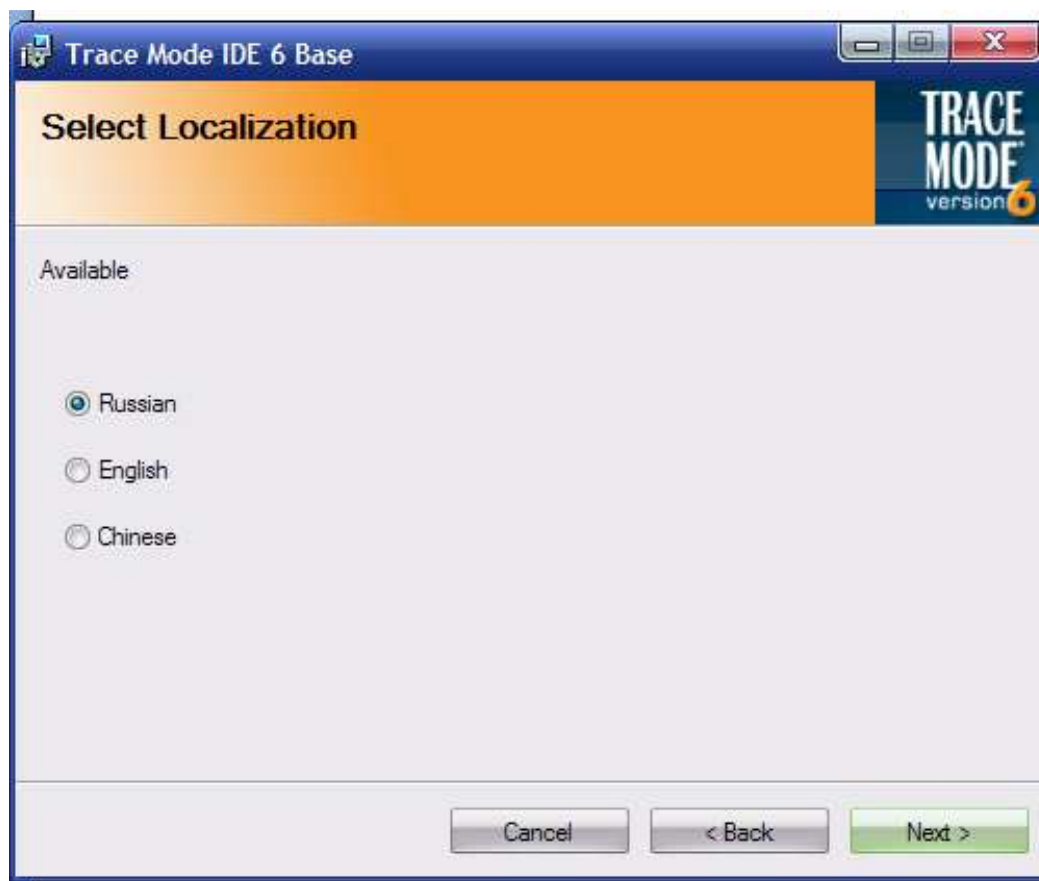


Рис. 3.3 Начало установки

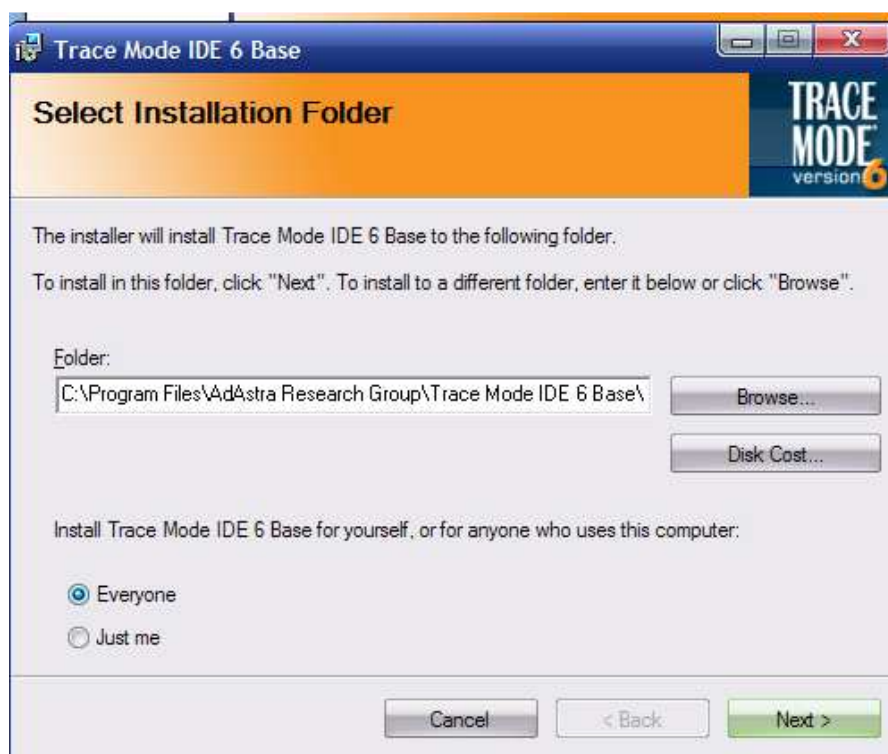


Рис. 3.4 Выбор директории

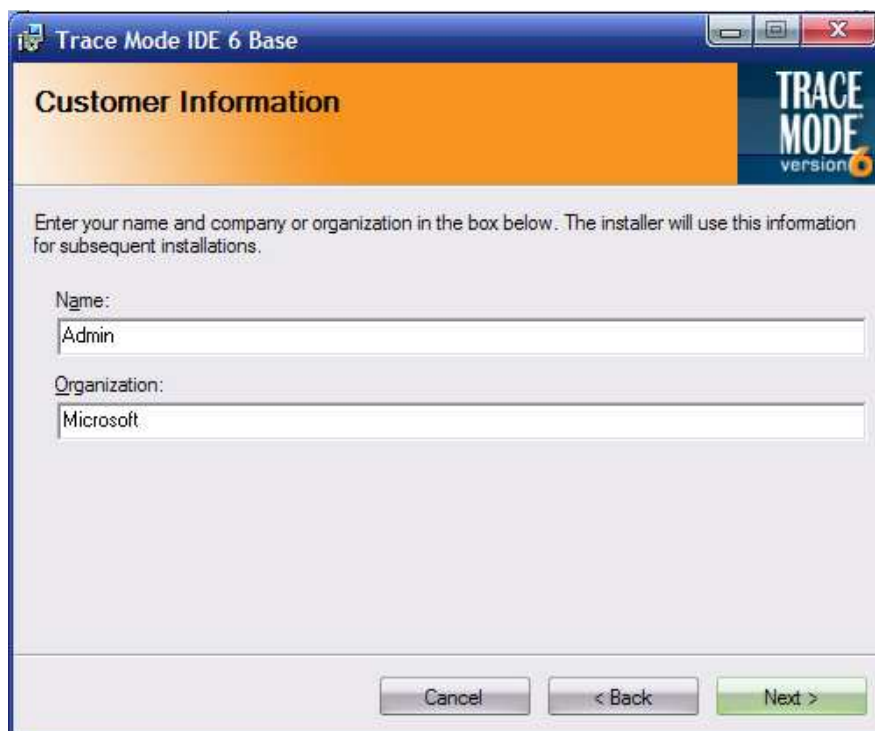


Рис. 3.5 Ввод сведений о пользователе





Рис. 3.6 Завершение установки

Работа 1 Создание проекта

Цель работы: познакомиться с SCADA–системой TRACE MODE 6, научиться создавать и настраивать канал, выводить информацию на экран в среде TRACE MODE.

Задание: создать канал, генератор заданного сигнала (синусоидального, пилообразного, треугольного), произвести привязку генератора к созданному каналу, масштабирование сигнала, обеспечив заданный диапазон; на экране вывести отмасштабированный сигнал с помощью стрелочного прибора, тренда, текста.

Ход работы

1. **Создание проекта Trace Mode.** Запустите программу Trace Mode 6 (файл tmdevenv.exe). Для создания нового проекта щелкните левой клавишей мыши на иконке  или на строчке **Новый** в меню **Файл**. Созданный проект примет вид, изображенный на рис. 3.7. При отсутствии навигатора проекта щелкните левой клавишей мыши на строчке **Навигатор проекта** меню **Вид**. Сохраните созданный проект, щелкнув левой клавишей мыши на иконке  или на строчке **Сохранить** или **Сохранить как** в меню **Файл**.

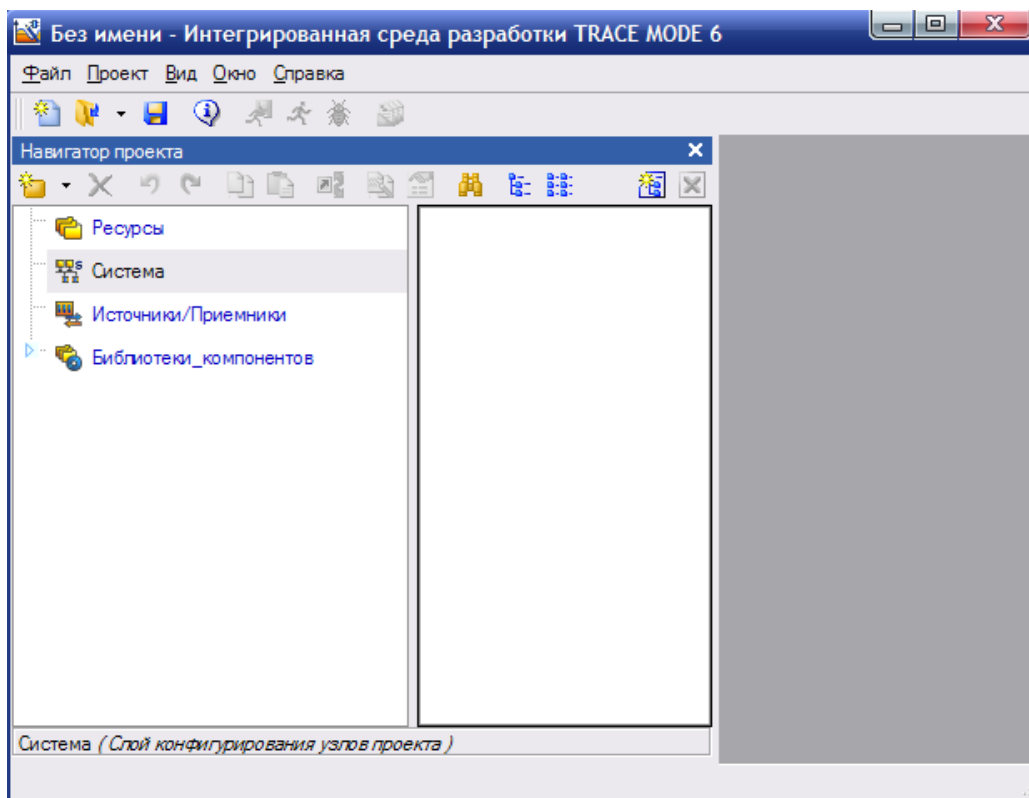


Рис. 3.7 Новы проект Trace Mode

2. **Создание узла.** Выделите строку *система* в навигаторе проекта (рис. 3.8). Вызовите контекстное меню щелчком правой клавиши мыши. В контекстном меню выберите строку *создать узел*. Среди предложенных типов узлов выберите **RTM**.

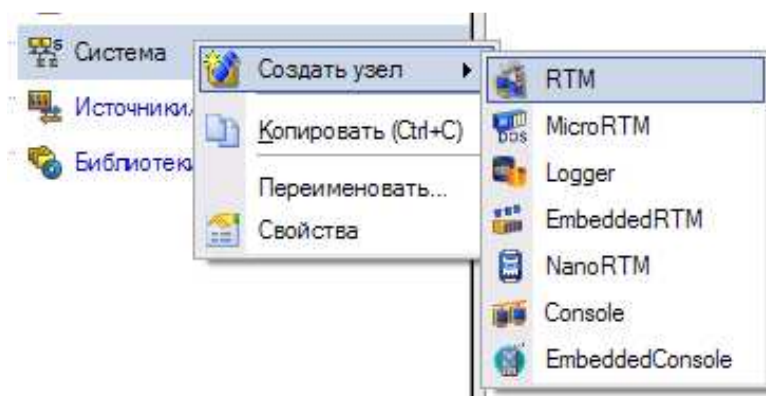


Рис. 3.8 Создание узла

Навигатор проекта в разделе *система* отобразит созданный узел. Выделите созданный узел **RTM** и щелкните левой клавишей мыши по нему. Появится возможность изменить имя узла. Пример созданного узла RTM изображен на рис. 3.9.

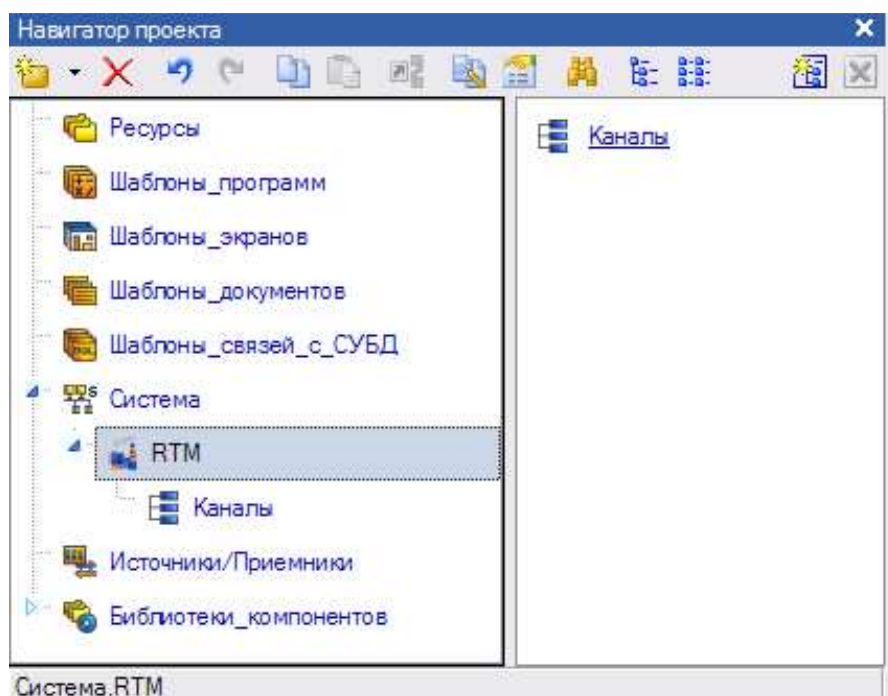


Рис. 3.9 Узел

3. **Создание канала.** Выделите группу *каналы* RTM узла. Вызовите контекстное меню. В появившемся контекстном меню выберите строку *создать компонент*. Среди предложенных компонентов выберите *канал_FLOAT* (рис. 3.10).

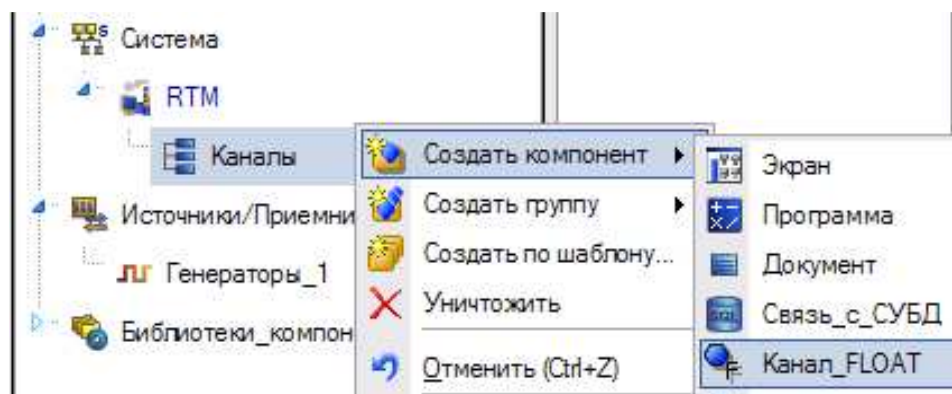


Рис. 3.10 Создание канала

4. **Произведите расчет множителя K и смещения Z .**
Преобразование сигнала будет производиться по формуле:

$$Y = K \cdot X + Z, \text{ где}$$

Y — результат преобразования;

X — входное значение канала;

K — множитель;

Z — смещение.

Входное значение канала изменяется в диапазоне $[0; 100]$. Найдите значение множителя K и смещения Z , которые позволят получить требуемый диапазон изменения результата преобразования (диапазон сигнала).

Пример: необходимо обеспечить диапазон сигнала $[-10; 10]$. Множитель K составит 0,2, смещение Z — -10.

5. **Редактирование созданного канала.** Дважды щелкните левой клавишей мыши на имени созданного канала (канал#1 на рис. 3.11). Откроется окно для редактирования канала (рис. 3.12)

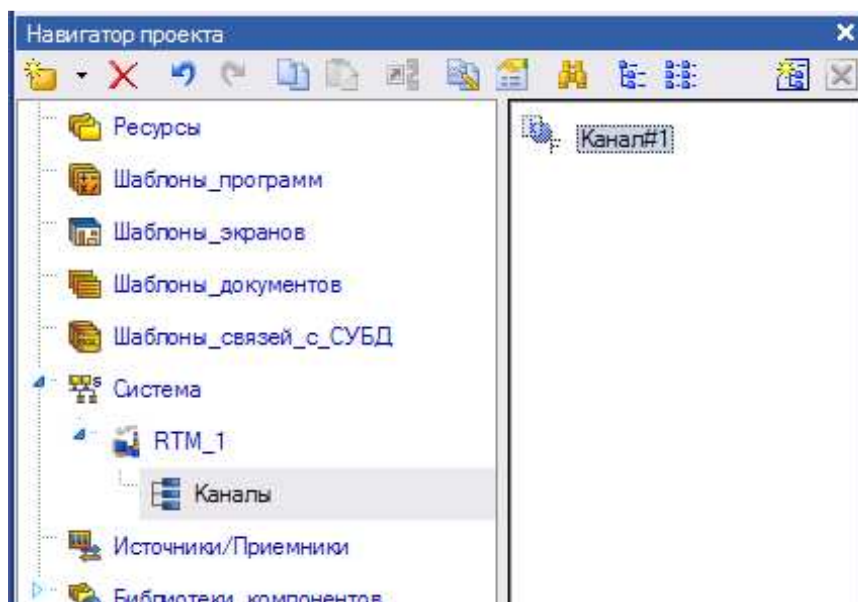


Рис. 3.11 Редактирование канала

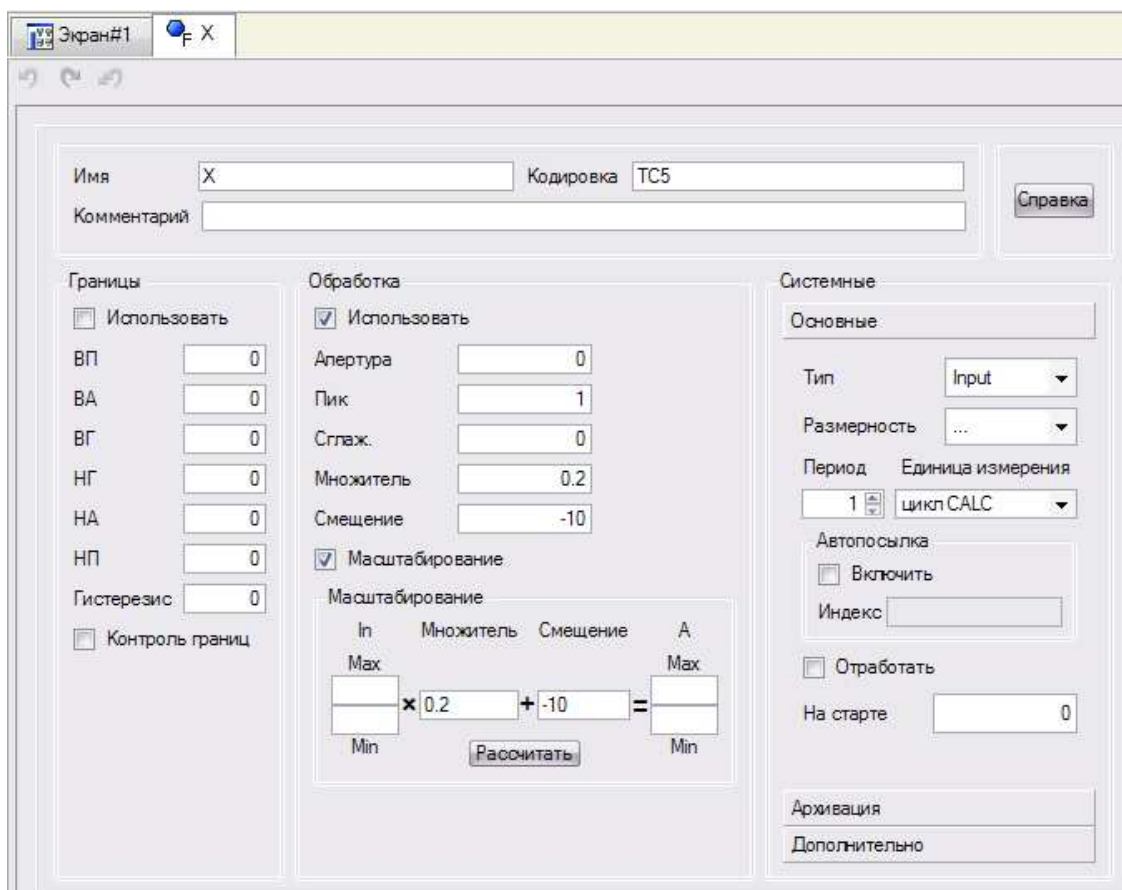


Рис. 3.12 Редактирование канала

В поле **имя** введите новое имя канала. 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**.

6. **Генерация сигнала.** Создайте группу генераторы. Для этого выделите строку **источники/приемники** навигатора проекта. Выделите контекстное меню. В появившемся меню выберите строку **создать группу**. Среди предложенных групп выберите **генераторы** (рис. 3.13). Выделив созданную группу (генераторы 1) и щелкнув левой клавишей мыши, измените имя, к примеру, на «генератор».

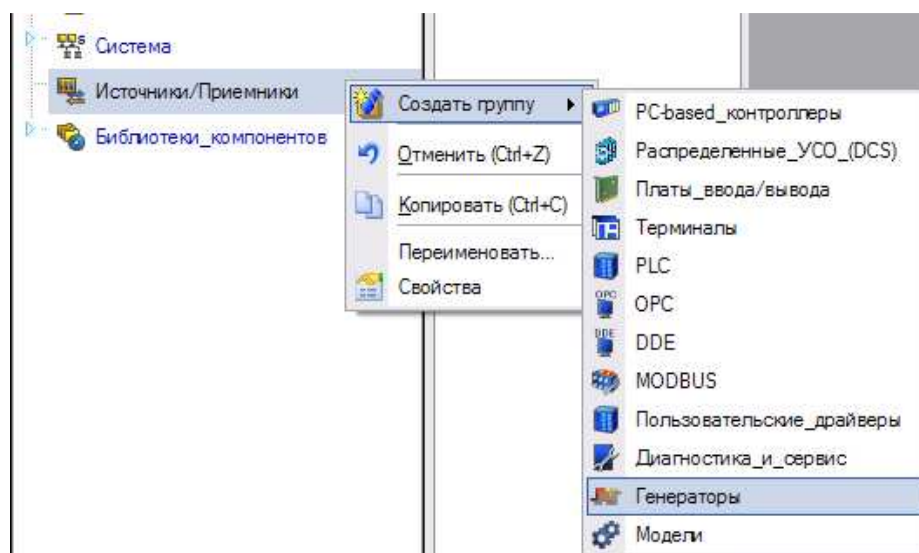


Рис. 3.13 Создание группы для генераторов

Выделите созданную группу. Вызовите контекстное меню. В контекстном меню выберите строку **создать компонент**. Среди предложенных генераторов выберите требуемый тип генератора (рис. 3.14). Выделив созданный генератор и щелкнув левой клавишей мыши, измените имя генератора, к примеру, на «Пила».

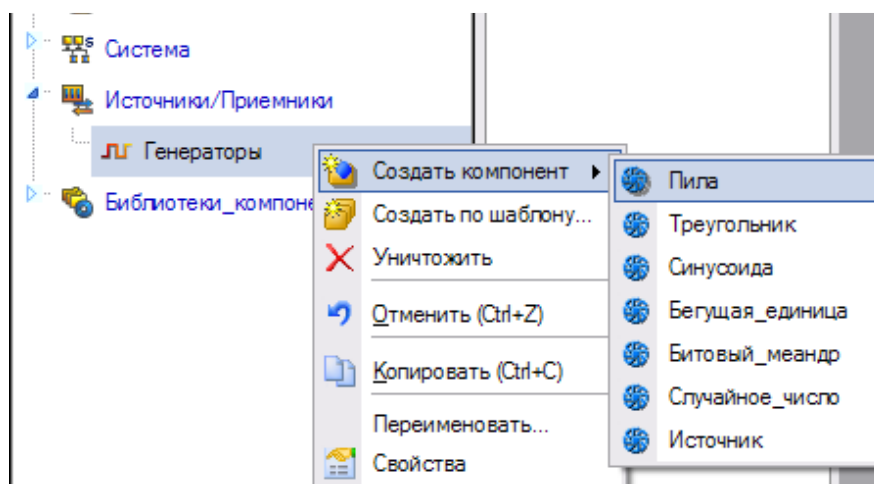




Рис. 3.14 Создание генератора

7. **Привязка созданного генератора.** Щелкните левой клавишей мыши по иконке . Откроется еще одно окно навигатора проекта. В левой части верхнего навигатора проекта выберите группу **каналы** RTM узла. В левой части нижнего навигатора проекта выберите группу **генераторы** группы **источники/приемники** (рис. 3.9). Нажмите левую клавишу мыши на созданном ранее генераторе. Не отпуская левой клавиши мыши, наведите

курсор мыши на созданный канал и отпустите левую клавишу мыши. Иконка канала должна измениться на  (рис. 3.15).

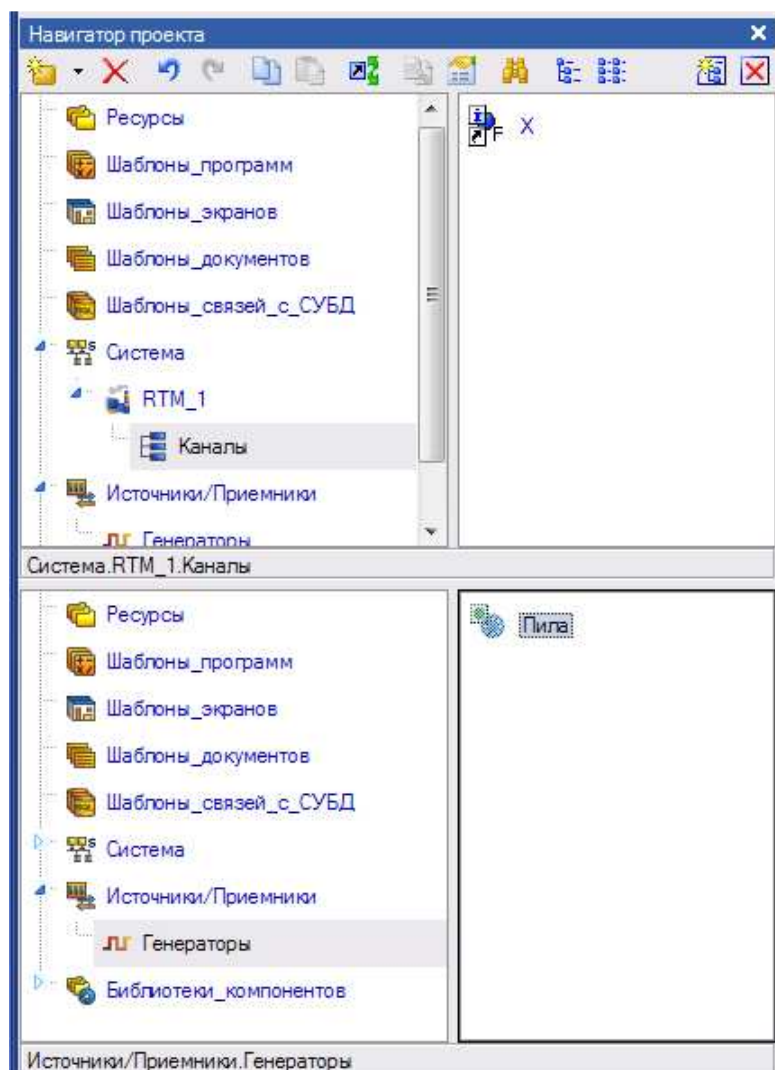



Рис. 3.15 Привязка генератора к каналу

Закрывать одно из созданных окон навигатора проекта можно щелкнув левой клавишей мыши по иконке .

8. **Создание экрана.** Выделите группу *каналы*, вызовите контекстное меню. В контекстном меню выберите строку *создать компонент*. Среди предложенных компонентов следует выбрать *экран* (рис. 3.16).

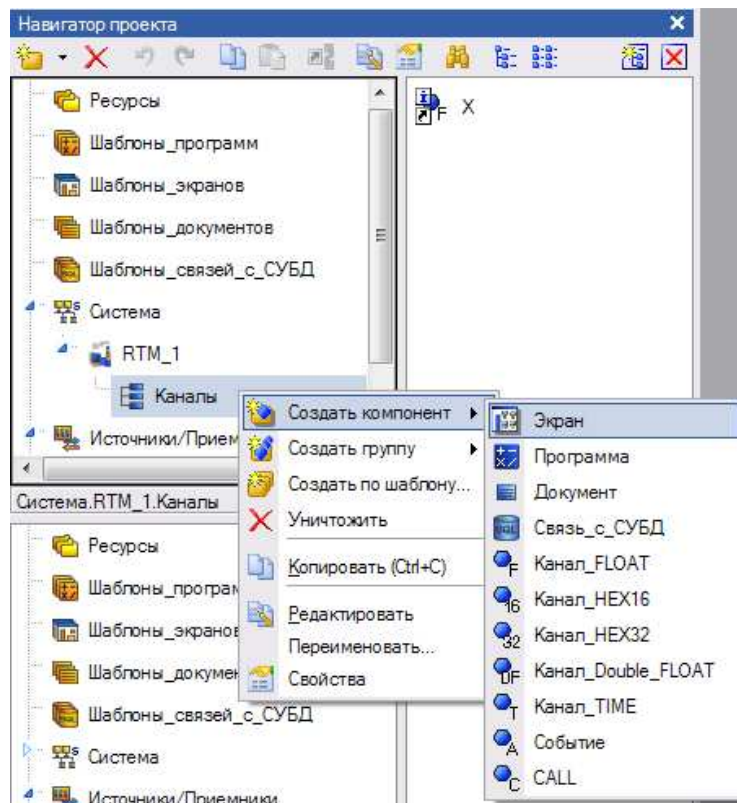


Рис. 3.16 Создание экрана

Выделите созданный экран и щелкните левой клавишей мыши, измените имя экрана. К примеру, можно изменить имя на «экран». Дважды щелкните левой клавишей мыши по созданному экрану. Откроется окно для редактирования шаблона экрана (рис. 3.17).

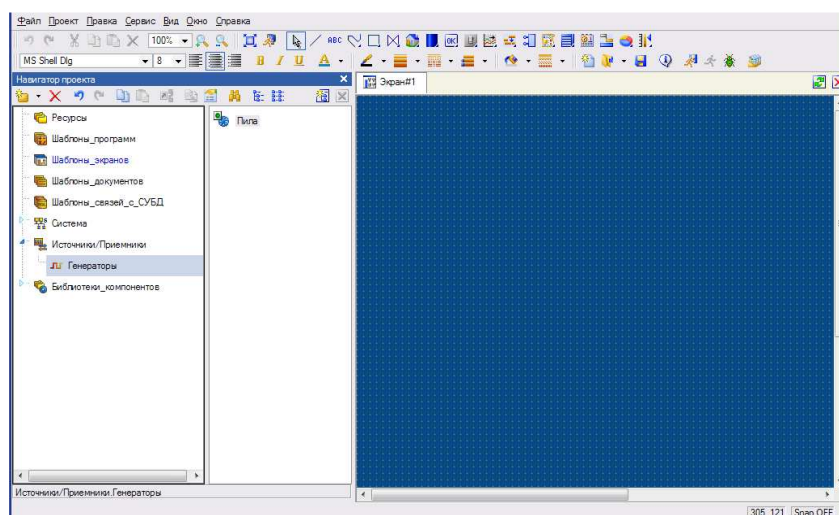








Рис. 3.17 Редактирование шаблона экрана

9. **Размещение объекта «стрелочный прибор».** Щелкните левой клавишей мыши по иконке . Если на панели инструментов вместо иконки  присутствует иконка , щелкните правой клавишей мыши по иконке . Среди предложенных объектов выберите стрелочный прибор . После выбора  щелкните левой клавишей мыши в том месте экрана, где должен располагаться один из углов стрелочного прибора. Переместите курсор в положение, соответствующее положению противоположного угла стрелочного прибора и щелкните левой клавишей мыши. Будет создан стрелочный прибор на экране (рис. 3.18).

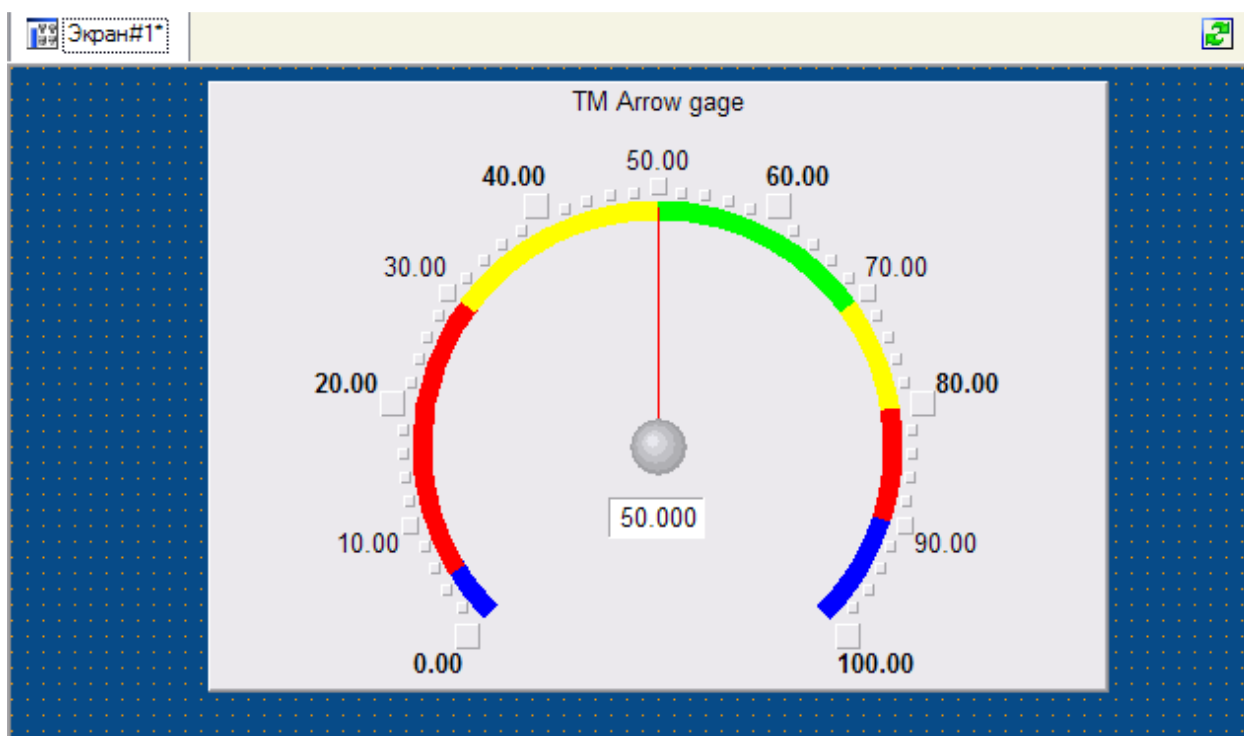


Рис. 3.18 Создание стрелочного прибора

Щелкните левой клавишей мыши по иконке .

10. **Настройка стрелочного прибора.** Если окно свойств объекта открыто выделите стрелочный прибор, в противном случае дважды щелкните левой клавишей мыши по созданному стрелочному прибору. Откроется окно свойств объекта (рис. 3.19). Раскройте раздел **заголовок** двойным щелчком левой клавиши мыши по подчеркнутой строке **заголовок**. В появившемся поле **текст** введите текст заголовка, который будет выведен на стрелочном приборе. Раскройте раздел **полоса** двойным щелчком левой клавиши мыши по подчеркнутой строке **полоса**. В поле **верхний предел шкалы** введите

верхнюю границу диапазона, в поле **нижний предел шкалы**— нижнюю границу диапазона. Заполните поля HL, HA, HW, LL, LA, LW любыми значениями, удовлетворяющими условию:

Нижний предел шкалы < LL < LA < LW < HW < HA < HL < Верхний предел шкалы

Пример заполнения полей свойств стрелочного прибора приведен на рис. 3.20.

Свойство	Значение
Скрыть при старте	False
<u>Отображаемая величина</u>	50
Положение	По центру
Угол разворота	135
<u>Заголовок</u>	
<u>Полоса</u>	True
<u>Шкала</u>	True
<u>Указатель</u>	
<u>Единицы</u>	
<u>Индикатор</u>	True
<u>Фон</u>	
<u>Рамка</u>	

Рис. 3.19 Настройка стрелочного прибора

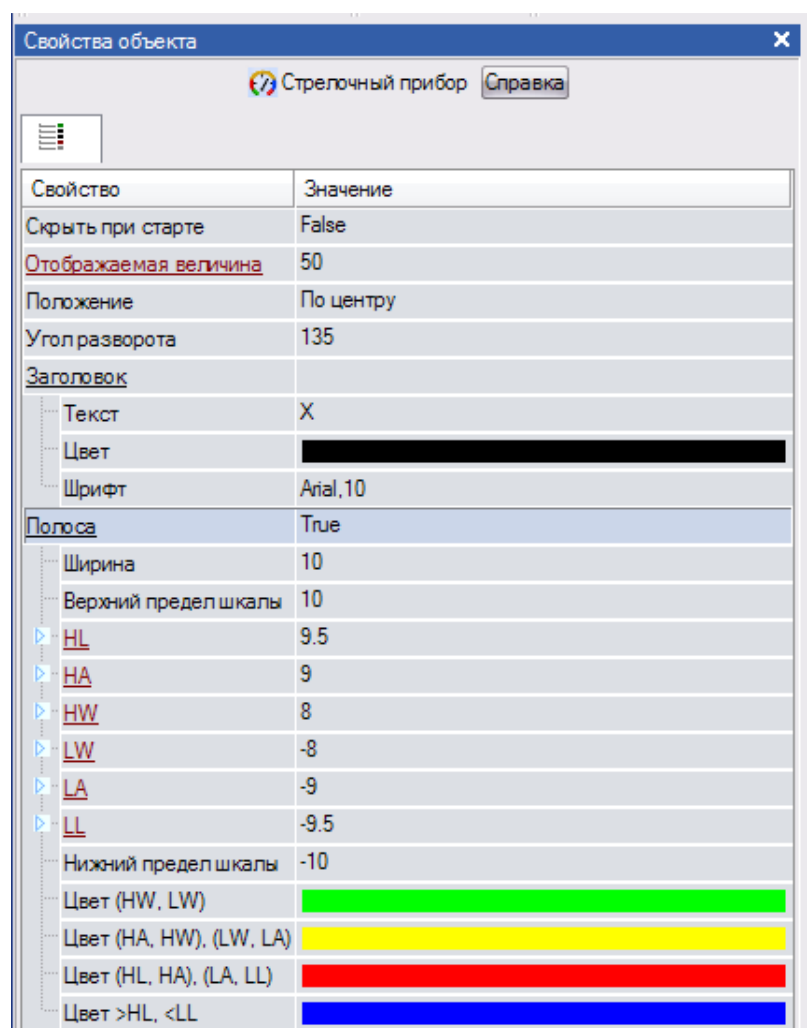


Рис. 3.20

11. **Привязка объекта к каналу.** Раскройте раздел **отображаемая величина**. Для этого дважды щелкните левой клавишей мыши на подчеркнутой строчке **отображаемая величина**. Щелкните левой клавишей мыши в поле **привязка**. Откроется окно свойств привязки с пустой таблицей. Щелкните левой клавишей мыши на иконке . Установите тип IN у созданного аргумента ARG_000. Дважды щелкните левой клавишей мыши в столбце **привязка** таблицы. Откроется окно привязки. В левой части открытого окна выделите канал RTM узла, созданный ранее. В правой части окна выберите аргумент реальное значение (рис. 3.21). Щелкните левой клавишей мыши по кнопке **привязка**. Окно свойств привязки примет вид, изображенное на рис. 3.22. В поле **имя** можно ввести новое имя канала аргумента. Щелкните левой клавишей мыши по кнопке **готово**. В поле привязка будет выведено имя созданного аргумента.

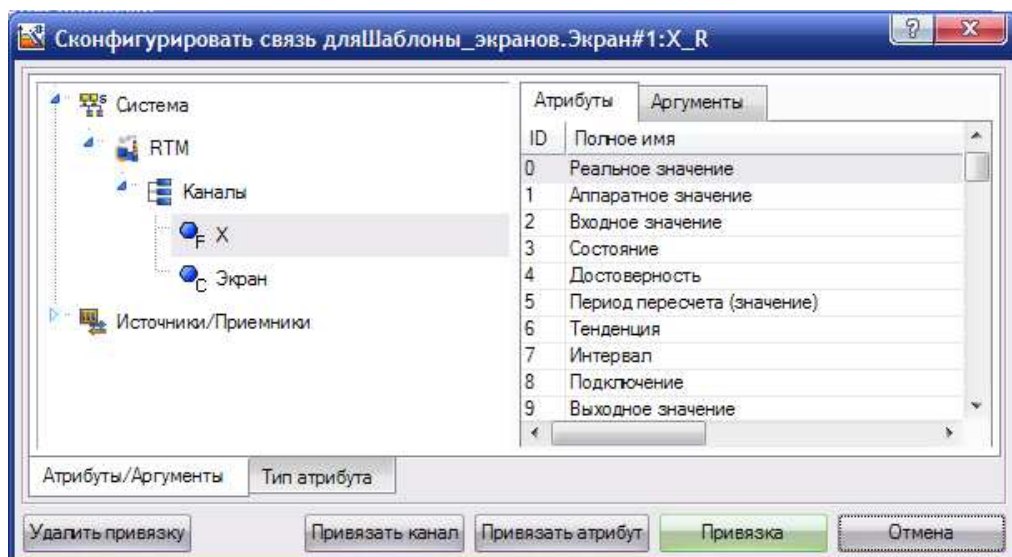


Рис. 3. 21 Привязка

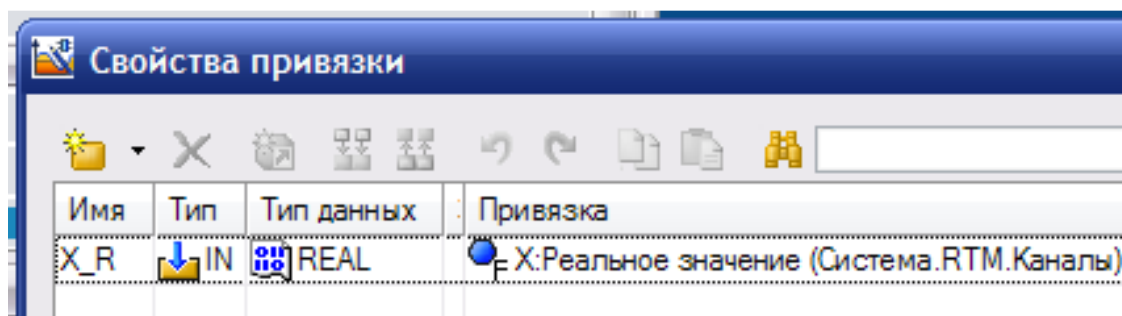








Рис. 3.22 Результат привязки

12. **Создание тренда.** Для создания тренда щелкните левой клавишей имени 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 . щелкните левой клавишей мыши по иконке .

Для размещения тренда на экране щелкните левой клавишей мыши там, где должен располагаться один из углов тренда. Переведите курсор в положение, где должен располагаться противоположный угол тренда и щелкните левой клавишей мыши (рис. 3.23).

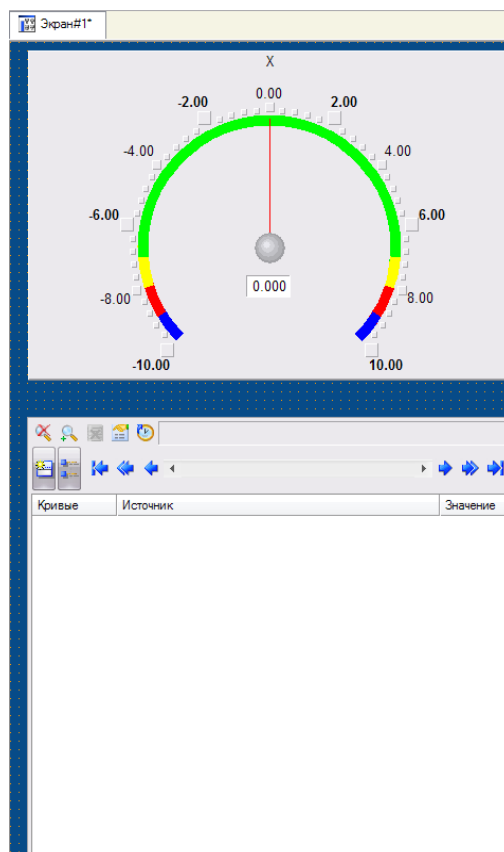


Рис. 3.23 Создание тренда

13. **Настройка тренда.** Выделите созданный тренд. Откроется окно свойств тренда. Изначально будет открыта закладка **основные свойства**. В поле **заголовок** введите текст, который вы хотите вывести в качестве заголовка. Перейдите на закладку **кривые**. Выделите строку **кривые** и вызовите контекстное меню. В появившемся контекстном меню выберите **кривая** (рис. 3.24).

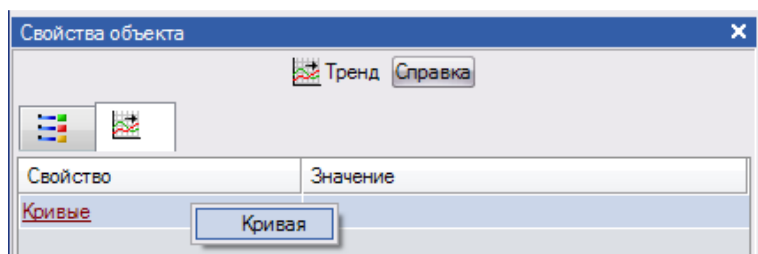


Рис. 3.24 Создание кривой

В появившемся поле **имя** введите имя созданной кривой. В поле **макс. значение** введите значение верхней границы диапазона, в поле **мин. значение**— нижней границы диапазона. Для привязки кривой щелкните

левой клавишей мыши в поле **привязка**. Откроется окно, изображенное на рис. 3.22. Выберите созданный ранее аргумент и щелкните левой клавишей мыши по кнопке **готово**. Тренд примет внешний вид, изображенный на рис. 3.25.

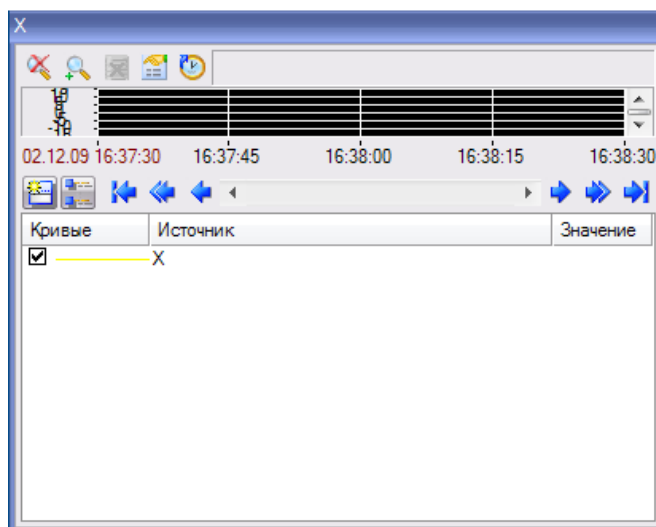


Рис. 3.25 Созданный тренд

Для увеличения размеров графика наведите курсор на границу легенды и графика. Курсор примет форму двух параллельных линий со стрелками. Нажмите левую клавишу мыши. Не отпуская левой клавиши мыши, опустите курсор вниз и отпустите левую клавишу мыши. Тренд примет вид, изображенный на рис. 3.26.

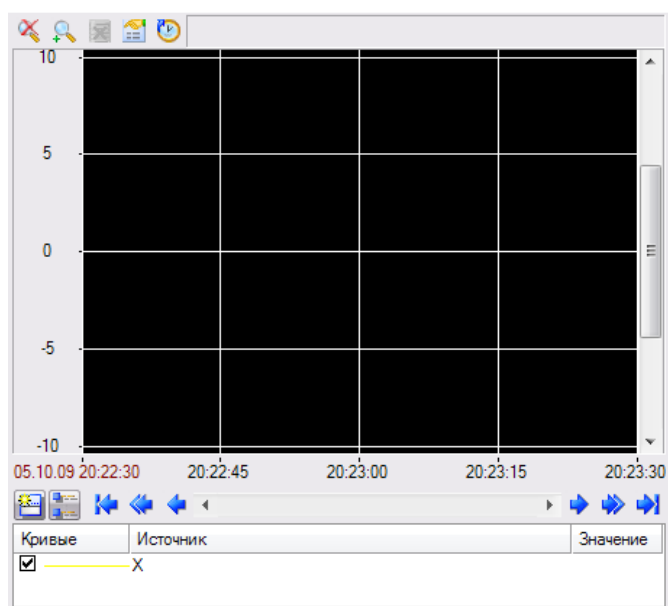



Рис. 3.26 Результат создания тренда

14. **Создание объекта текст.** Щелкните левой клавишей мыши на иконке **ABC**. Щелкните левой клавишей мыши там, где должен располагаться один из углов объекта текст. Переведите курсор мыши в положение, где должен располагаться другой край объекта текст и щелкните левой клавишей мыши. Рядом аналогично разместите еще один объект текст (рис. 3.27). Щелкните левой клавишей мыши по иконке .

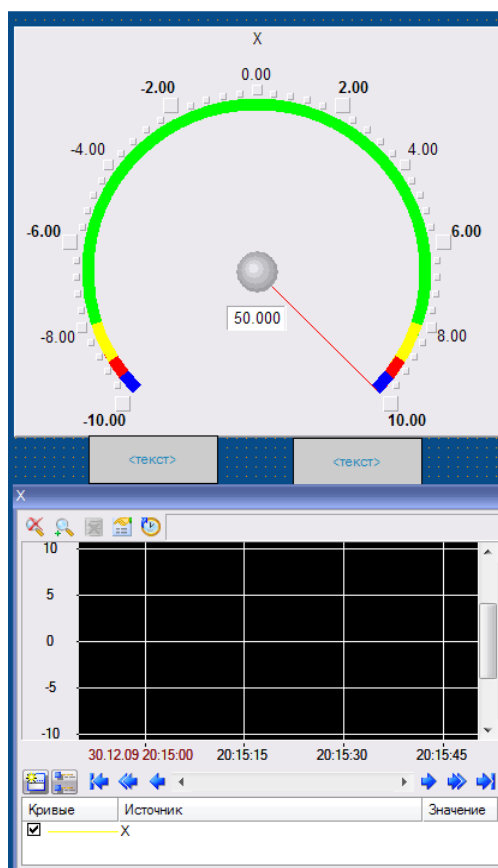


Рис. 3.27 Пример размещения объектов текст

15. **Настройка объектов текст.** Выделите объект текст, расположенный слева. В поле **текст** введите имя канала RTM узла. Выделите объект текст, расположенный справа. Откройте раздел **текст** двойным щелчком по подчеркнутой строчке **текст**. В поле **вид динамизации** выберите **значение** (рис. 3.28). Щелкните левой клавишей мыши в появившемся поле **привязка**. Откроется окно, изображенное на рис. 3.22. Выберите созданный ранее аргумент и щелкните левой клавишей мыши по кнопке **готово**. Объекты текст примут вид, изображенный на рис. 3.29.

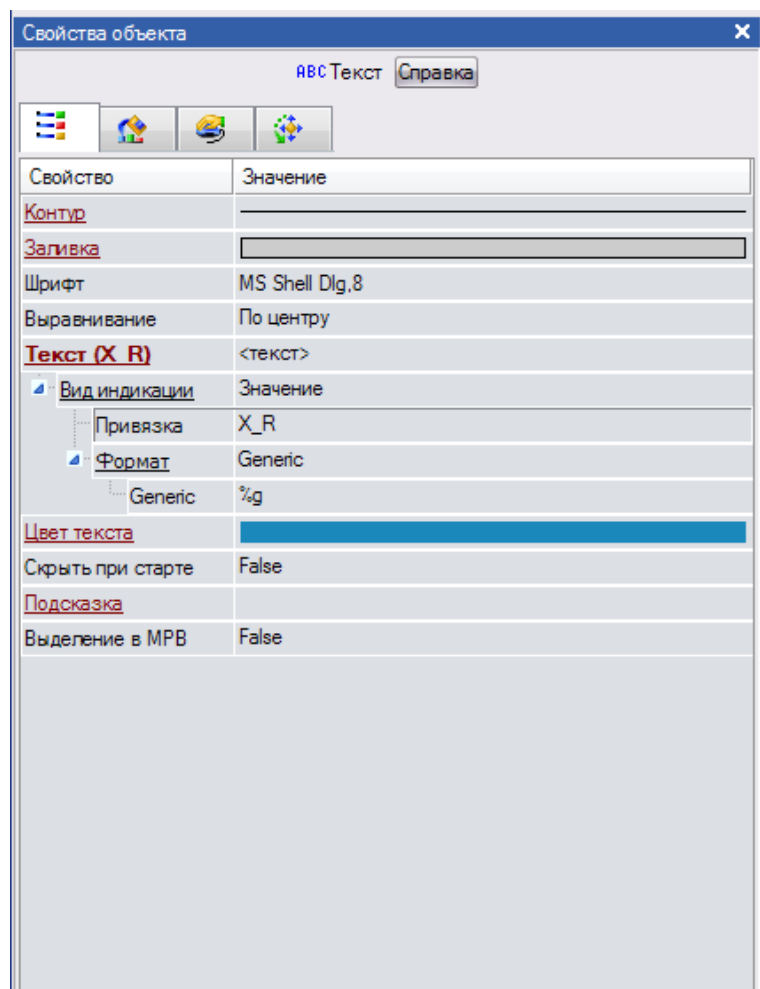






Рис. 3.28 Редактирование объекта текст



Рис. 3.29 Объекты текст

16. **Запуск проекта.** Щелкните левой клавишей мыши по иконке  или по строчке *сохранить для MPB* в меню *файл*. В навигаторе проекта выделите созданный RTM узел. Щелчком левой клавиши мыши по иконке  откройте профайлер. Если профайлер не запустился, запустите файл *rtc.exe*, щелкните левой клавишей мыши по иконке  или строчке *открыть* в меню *файл*. Для запуска проекта щелкните левой клавишей мыши по иконке . В результате запуска профайлера стрелка стрелочного прибора должна

перемещаться, тренд строить график, а объект текст выводить текущее значение созданного канала (рис. 3.30).

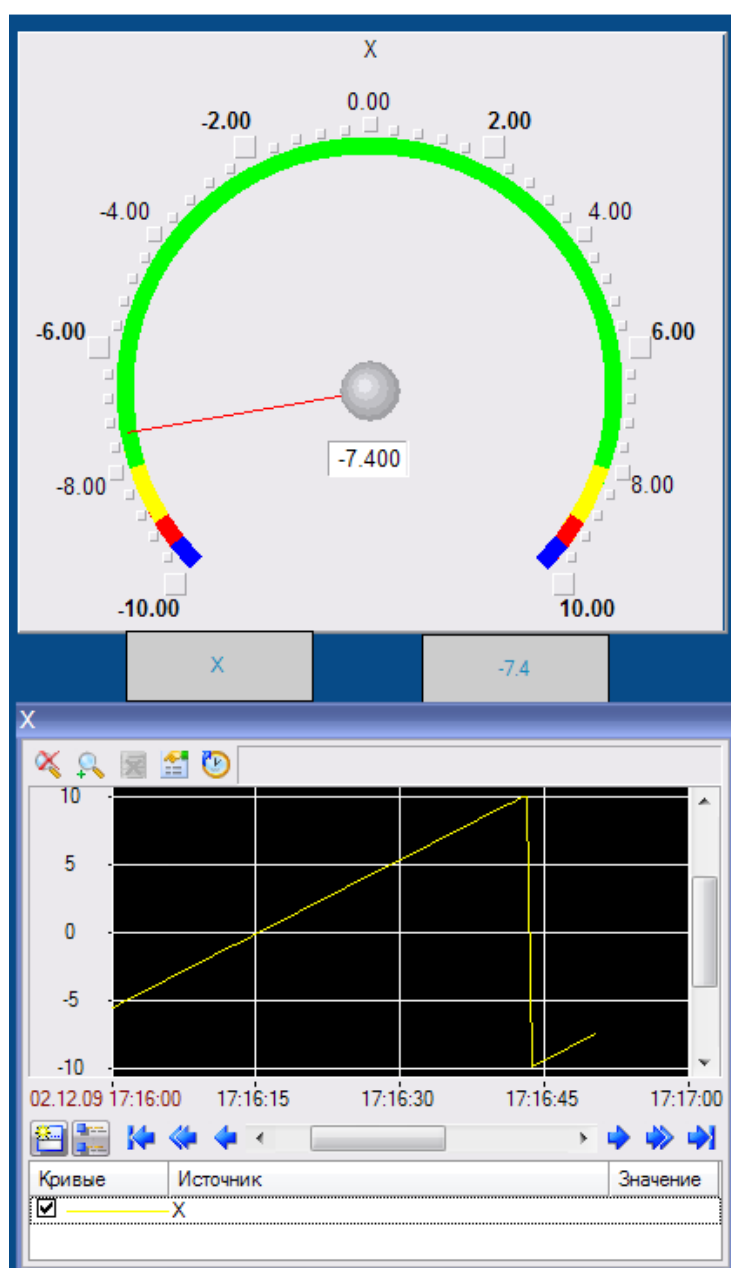


Рис. 3.30 Результат запуска проекта

Контрольные вопросы

1. Что такое SCADA-система.
2. Основные функции SCADA-систем.
3. Причины появления SCADA-систем.
4. Что такое АСУ ТП, их назначение?
5. Что такое узел?
6. Что такое канал?
7. Что такое база каналов?
8. Что такое объекты базы каналов?
9. Функции и назначение контроллеров нижнего уровня АСУ ТП.
10. Функции и назначение контроллеров верхнего уровня.
11. Что такое микроSCADA?
12. Опишите диспетчерский уровень АСУ ТП.

Работа 2

Создание статического и динамического изображения

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

Задание:

1. создать новый проект. В данном проекте создать генератор, привязанный к каналу, значение которого определяется уровнем продукта в емкости;
2. создать экран, расположить на нем тренд, строящий зависимость уровня продукта во времени.
3. 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.
4. создать динамический объект, имитирующий заполнение емкости, используя графический файл.

Ход работы

1. **Создайте канал, который будет пропорционален уровню продукта в емкости.** Назовем данный канал как уровень, для лучшего восприятия.
2. **Создайте генератор синусоидального сигнала.**
3. **Произведите привязку созданного генератора к созданному каналу.**
4. **Создайте экран.**
5. **Создайте тренд, настройте кривую и произведите привязка к созданному каналу .** Пример экрана, созданного при выполнении пунктов 1— 5 приведен на рис. 3.31

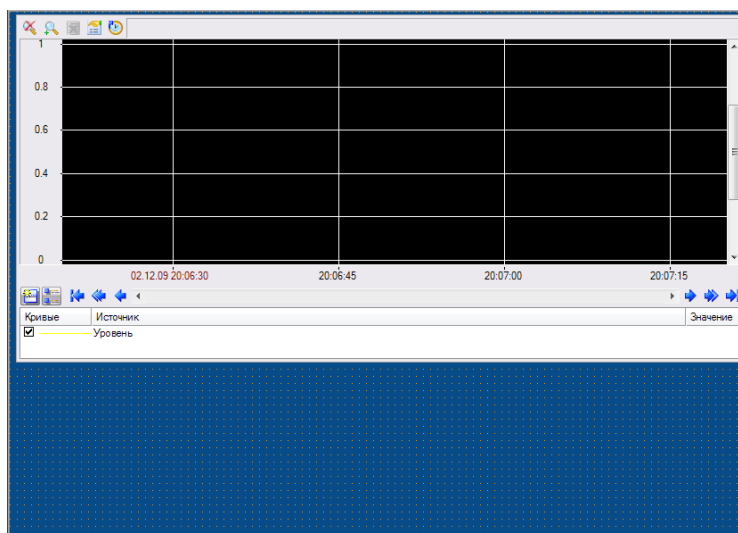








Рис. 3.31 Результат выполнения пунктов 1—5

6. Создание статического изображения.

а. Создание рамки. Для создания рамки щелкните левой клавишей мыши по иконке . Если нет указанной иконки на панели инструментов, то щелкните правой клавишей мыши по одной из иконок: , , или . Среди предложенных объектов выберите рамку . (рис. 3.32). Щечком левой клавишей мыши задайте противоположные углы рамки. Для перехода в режим редактирования щелкните левой клавишей мыши по иконке .

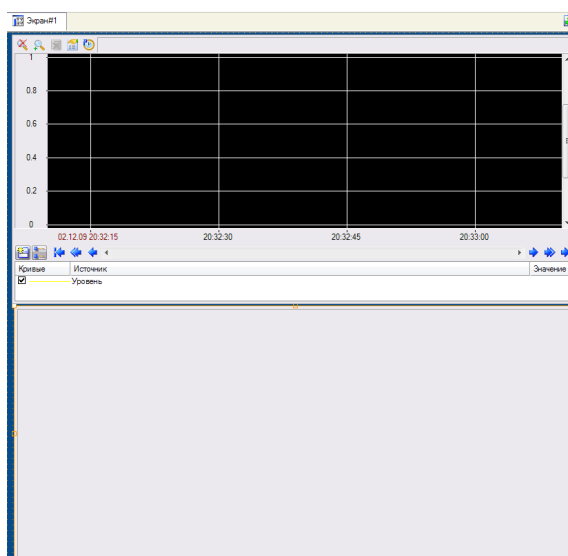

















Рис. 3.32 Размещение рамки

б. Создание емкости.

Щелкните левой клавишей мыши по иконке . Если данной иконки нет на панели инструментов, то щелкните правой кнопкой мыши по одной из иконок: , , , , , , , , , или . Среди предложенных объектов выберите емкость . После выбора инструмента поместите емкость на экране. Задайте противоположные углы емкости щелчком левой клавиши мыши. щелкните левой клавишей мыши по иконке . Емкость примет вид, изображенный на рис. 3.33. Выделите созданную емкость или дважды щелкните по ней левой клавишей мыши. Откроется окно свойств объекта. В поле **толщина стенок** задайте толщину больше 0. Емкость будет изображена в разрезе (рис. 3.34). В поле **верхний** и **нижний край** выберите необходимый вид края. К примеру, верхний край примет вид , а нижний— . Для задания материала емкости раскройте раздел материал, дважды щелкнув левой клавишей мыши по подчеркнутой строчке **материал**, если данный раздел не раскрыт. В поле **выбрать из списка** выберите значение **true**. В поле **материал** выберите необходимый материал, к примеру, хром. В поле **стандартная текстура** выберите необходимую текстуру, к примеру, гравировку. При необходимости можно добавить другие объемные фигуры.

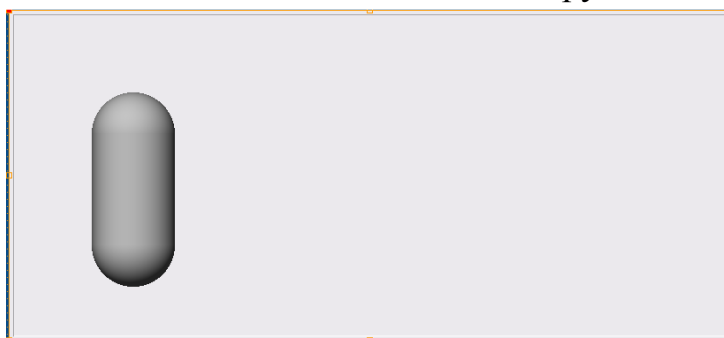


Рис. 3.33 Создание емкости



Рис. 3.34 Задание толщины стенок емкости







Для размещения конуса щелкните правой клавишей мыши по иконке  и среди предложенных инструментов выберите . Разместите на экране конус, задав противоположные углы прямоугольника в который будет вписан конус щелчком левой клавиши мыши. Щелкните левой клавишей мыши по иконке . Выделите созданный конус. Если не открылось окно свойств конуса, дважды щелкните левой клавишей мыши по нему. В поле **толщина стенок** задайте ту же толщину, что и у емкости. Раскройте раздел **материал** двойным щелчком мыши по подчеркнутой строчке **материал**, если слой еще не раскрыт. Выберите значение **true** в поле **выбрать из списка**. Выберите требуемый материал в поле **материал**, к примеру, олово. В поле **стандартная гравировка** задайте гравировку, к примеру, шлифовку. Емкость примет вид, изображенный на рис. 3.35



Рис. 3.35 Пример емкости

с. Создание насоса. Щелкните правой клавишей мыши по иконке . Среди предложенных инструментов выберите . Для размещения насоса на экране задайте противоположные углы прямоугольника, в который будет вписан насос щелчком левой клавиши мыши. Щелкните левой клавишей мыши по иконке . Выделите насос. Если не открылось окно свойств насоса, дважды щелкните левой клавишей мыши по созданному насосу. Раскройте раздел **материал**, двойным щелчком по подчеркнутому тексту **материал**, если он не раскрыт еще. В поле **выбрать из списка** установит значение **true**. В поле **материал** выберите необходимый материал, к примеру, пластик черный. В поле **форма насоса** выберите нужную форму насоса. Статическое изображение примет вид, изображенный на рис. 3.36

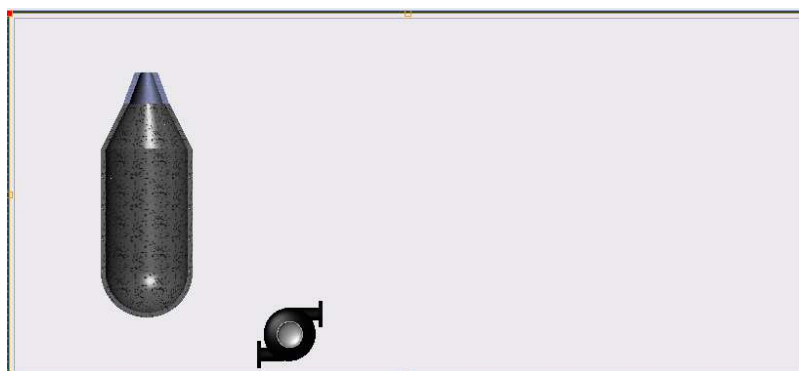





Рис. 3.36 Размещение насоса

d. Создание труб. Щелкните правой клавишей мыши по иконке . Среди предложенных инструментов выберите . Создайте трубу по которой продукт поступает в емкость и по которой из емкости течет в насос. Для этого щелчком левой клавиши мыши отметьте местоположение начала трубы. Переведите курсор мыши в положение изгиба трубы и снова щелкните левой клавишей мыши. Таким образом, отмечаются все точки изгиба трубы. Когда курсор переведен в положение, где размещается конец трубы, щелкните правой кнопкой мыши, завершая создание текущей трубы. Создайте аналогично трубу, по которой продукт поступает в насос и вытекает из него. Для редактирования свойств каждой трубы выделите трубу. Если не открылось окно свойств, дважды щелкните левой клавишей мыши по трубе. В поле толщина подберите толщину каждой трубы, которая лучше будет подходить для рисунка. В поле базовый цвет выберите необходимый цвет, к примеру . Статическое изображение примет вид, указанный на рис. 3.37

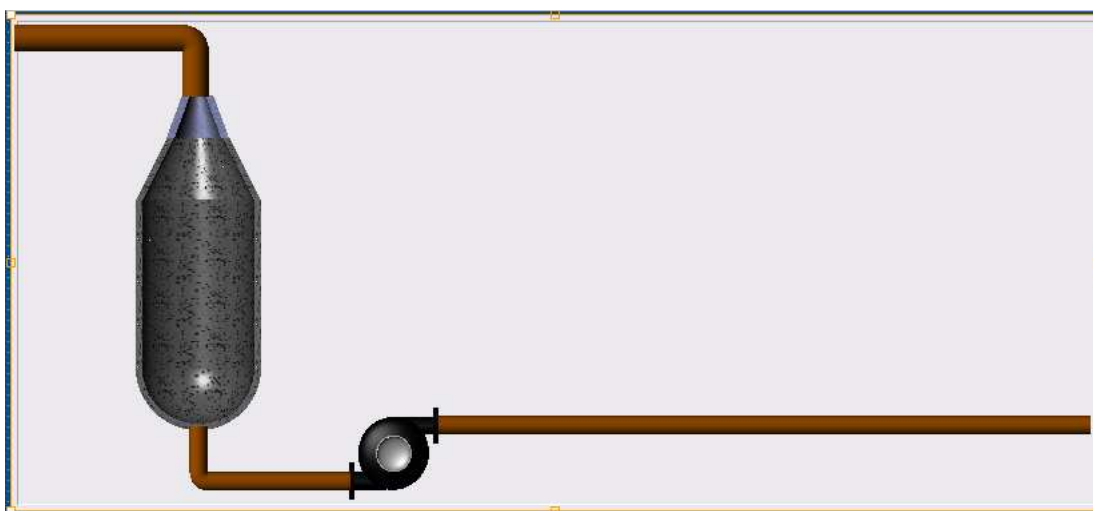


Рис. 3.37 Создание труб

7. Создание динамического изображения.

а. Импорт изображения. Выделите строку **ресурсы** навигатора проекта. Вызовите контекстное меню. Выберите строку **создать группу**. Среди предложенных групп выберите **картинки** (рис. 3.38).

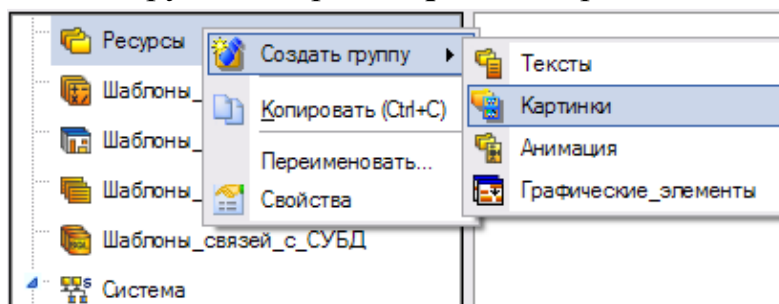


Рис. 3.38 Создание группы картинки

Выделите созданную группу картинки, вызовите контекстное меню, выберите **создать компонент**. Среди предложенных компонентов выберите библиотеку изображений (рис. 3.39).

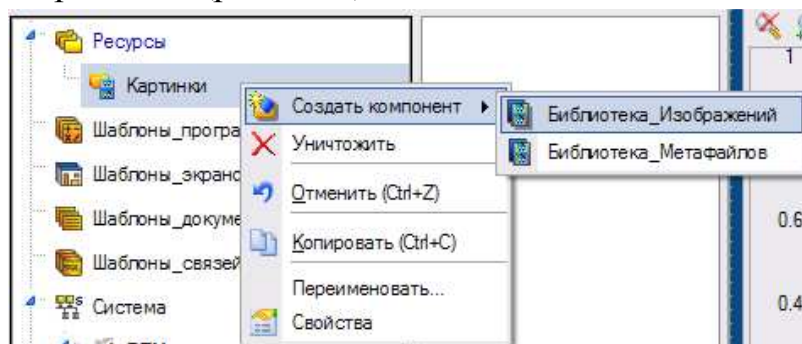









Рис. 3.39 Создание библиотеки изображений

Для импортирования изображения дважды щелкните левой клавишей мыши по созданной **библиотеке изображений**. Откроется пустое окно. В данном пустом окне вызовите контекстное меню и выберите **импортировать**. Откроется диалоговое окно для открывания графического файла. Откройте графический файл изображения, которое будет использоваться в дальнейшем.

б. Создание динамической заливки. Создайте многоугольник. Щелкните левой клавишей мыши по иконке . Если на инструментальной панели нет иконки , щелкните правой клавишей мыши по одной из иконок: , , , . Среди предложенных инструментов выберите . Для размещения многоугольника щелкните левой клавишей мыши там, где должен располагаться один из углов многоугольника. Задайте точки излом щелчком левой клавишей мыши там, где будут располагаться другие углы

многоугольника. Последний угол многоугольника следует отмечать щелчком правой клавиши мыши. Результат создания многоугольника приведено на рис. 3.40

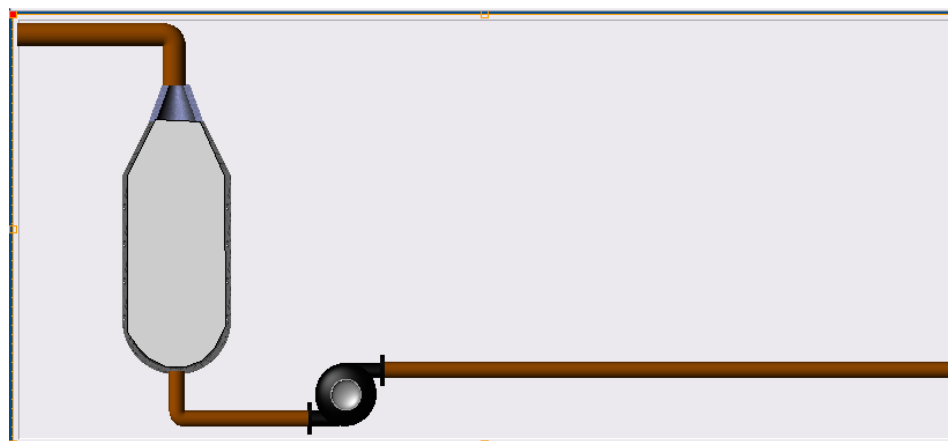


Рис. 3.40 Создание многоугольника для заливки

Выделите созданный многоугольник. Если окно свойств не открылось, дважды щелкните левой клавишей мыши по созданному многоугольнику. Раскройте раздел **заливка** двойным щелчком левой клавиши мыши, если он не раскрыт. В появившемся поле **стиль** выберите без заливки. В результате изображение примет вид, указанный на рис. 3.37.


Перейдите на закладку динамическая заливка . Поставьте флажок **разрешено**. Раскройте раздел «**слой**». Для этого дважды щелкните левой клавишей мыши на тексте **слой**. В появившемся поле **имя** введите имя для слоя. Произведите привязку слоя к аргументу, посредством которого произведена привязка тренда к канала, хранящему значение уровня продукта в емкости. В поле **тип заливки** выберите изображение. Щелкните в поле **изображение**. Откроется окно, предлагающее выбор изображений, хранящихся в библиотеке изображений, созданной ранее (рис. 3.41). Выберите необходимое изображение и щелкните левой клавишей мыши по кнопке **готово**. В поле **Макс.** установите значение верхней границы диапазона значений, хранимых в канале, созданном ранее, а в поле Мин.— нижней границы.



Рис. 3.41 Выбор изображения для заливки

8. **Запуск проекта.** Произведите запуск проекта, аналогично первой работе. Пример результата исполнения созданного проекта приведен на рис. 3.42.

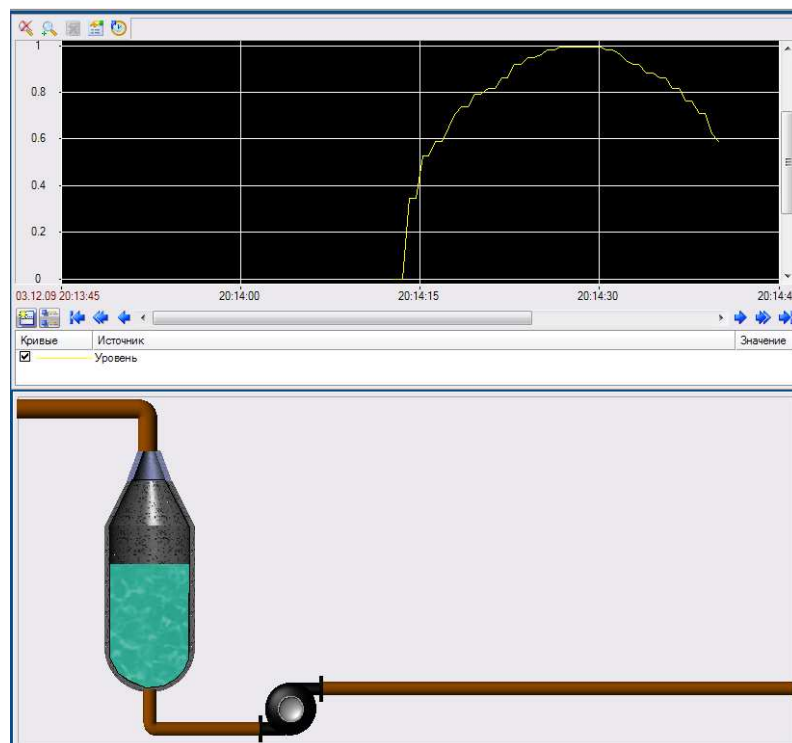


Рис. 3.42 Пример результата запуска

Вопросы для самоконтроля:

1. Опишите различие входных и выходных каналов.
2. Что такое привязка, зачем она нужна, как производится?
3. Что такое навигатор проекта?
4. Как импортируется изображение?
5. Как создаются компоненты базы каналов в Trace Mode?
6. Как размещается объект на экране?
7. Зачем нужно окно свойств объекта, что оно дает?
8. Чем отличается статическое изображение от динамического?
9. Как создается статическое изображение?
10. Как создается динамическое изображение?
11. Какие элементы изображения могут быть динамическими в Trace Mode?

Работа 3 Программирование на языках Техно ST и Техно FBD.

Цель работы: изучить языки Техно ST и Техно FBD среды Trace Mode, реализовать систему АСУ ТП с использованием программной обработки.

Задание:

1. Взять за основу проект, созданный при выполнении второй лабораторной работы (статическое и динамическое изображение), добавить каналы, которые передают значение стоимости продукта, расхода, суммарного расхода, суммарной стоимости продукта, периода генерации сигнала, удалить использовавшийся во второй лабораторной работе генератор. У всех каналов следует снять флаг *использовать* на панели *обработка* при редактировании канала.
2. Добавить, кнопки, при нажатии на которые будет вводиться стоимость продукта, период генерации уровня продукта в емкости, расход, объекты текст для отображения суммарного расхода продукта, стоимости израсходованного продукта, стоимости продукта, периода генерации уровня продукта в емкости, расхода и стоимости.
3. Добавить FBD диаграмму, генерирующую уровень продукта в емкости. Если период генерации меньше 10, то программа должна установить период равный 10. Программа должна возвращать отмасштабированный сигнал, то есть значения должны изменяться в заданном диапазоне.
4. Добавить программу ST, которая проверяет значение стоимости и расхода и производит расчеты. Если значение стоимости или расхода меньше 1, то устанавливает соответствующий параметр равным 1. Производит расчет суммарного расхода продукта и суммарной стоимости израсходованного продукта.

Примечания:

1. У всех каналов и у программ должен быть установлен при редактировании одинаковый период 1— 3 секунды.

2. Вычислять суммарный расход следует по формуле:

$$\text{Суммарный_расход} = \text{Предыдущий_расход} + \text{Расход} \cdot \text{Период_Обработки}, \text{ где}$$

Суммарный_расход— аргумент программы возвращающий значение суммарного расхода входному значению канала, который передает значение суммарного расхода;

Предыдущий_расход— глобальная переменная, содержащая значение суммарного расхода, вычисленное при предыдущем вызове программы;

Период_Обработки— аргумент привязанный к аргументу *период пересчета* канала, который передает значение расхода.

3. После вычисления суммарного расхода следует глобальной переменной Предыдущий_расход присвоить вычисленное значение Суммарный_Расход, суммарную стоимость продукта вычислить как произведение вычисленного значения суммарного расхода на стоимость продукта.





Ход работы

1. **Создание проекта.** Сохраните проект, созданный при выполнении работы 2 под новым именем. Удалите генератор, привязанный к каналу Уровень . Выделите объект *экр*ан и вызовите контекстное меню. Выберите *редактировать*. В открывшемся окне редактирования канала выберите *период* равным 1, единицу измерения— сек. Аналогично измените период пересчета канала, который хранит уровень продукта в емкости.

2. **Создание необходимых каналов.** Добавьте каналы хранящие значение стоимости продукта, расхода продукта, периода генерации. Для лучшего восприятия назовем канал, хранящий стоимость продукта как стоимость, хранящий расход продукта— расход, период генерации— период

генерации. Установите тип каждого канала— Input (см. работу 1). Добавьте каналы, хранящие результаты вычислений: суммарный расход продукта, суммарная стоимость продукта. Для удобства восприятия назовем канал, хранящий суммарную стоимость продукта, как суммарная стоимость, а суммарный расход продукта— суммарный расход. Установите тип указанных каналов— Input. Для всех каналов установим одинаковый период пересчета значения. Для этого вызовите окно редактирования для каждого канала. В поле **период** установите необходимый период пересчета 1 и **единицу измерения**— секунду (сек).

3. **Создание кнопок для задания необходимых параметров.**

Щелкните левой кнопкой мыши по иконке . Если данной иконки нет панели инструментов, то щелкните правой клавишей мыши по одной из иконок:  или . Разместите кнопки на экране. Для этого задайте противоположные углы кнопок щелчком левой кнопки мыши (рис. 3.43). Для каждой кнопки откройте окно свойств. В поле **текст** введите назначение кнопки. Так, для кнопки, которая устанавливает период генерации можно ввести «Установить период», устанавливает расход продукта— «установить расход», устанавливает стоимость продукта— «Установить стоимость», к примеру. Перейдите на закладку **действия** . Выделите строку **mousePressed** и вызовите контекстное меню (рис. 3.44). Выберите в появившемся контекстном меню строку **передать значение**. В появившемся поле **тип передачи** установите **ввести и передать**. В поле **результат** произведите привязку к входному значению канала, которому будет передаваться вводимое значение (пример настройки кнопки— рис. 3.45). В результате привязки будут созданы четыре аргумента в таблице окна свойств привязки (рис. 3.46).

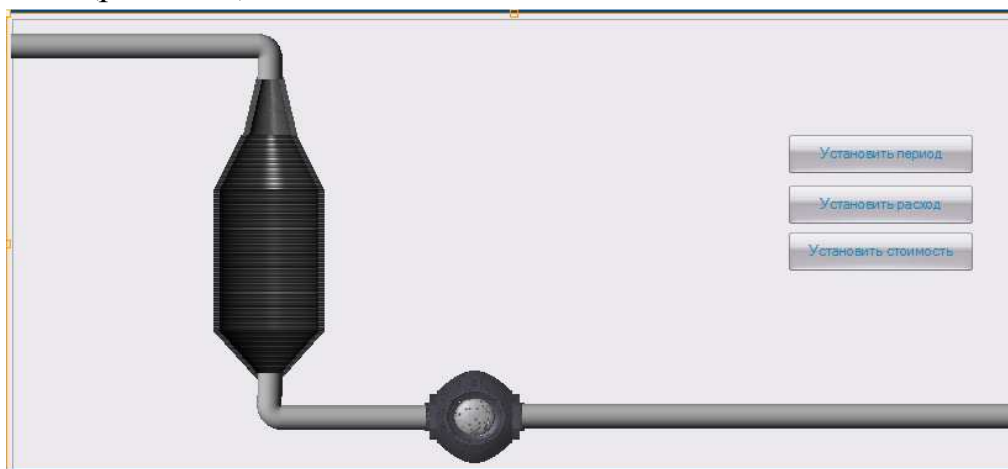


Рис. 3.43 Размещение кнопок

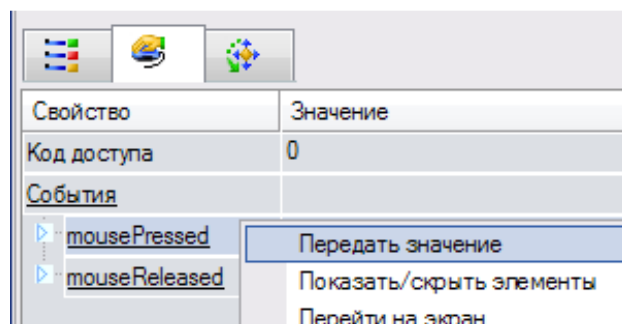


Рис. 3.44 Создание события

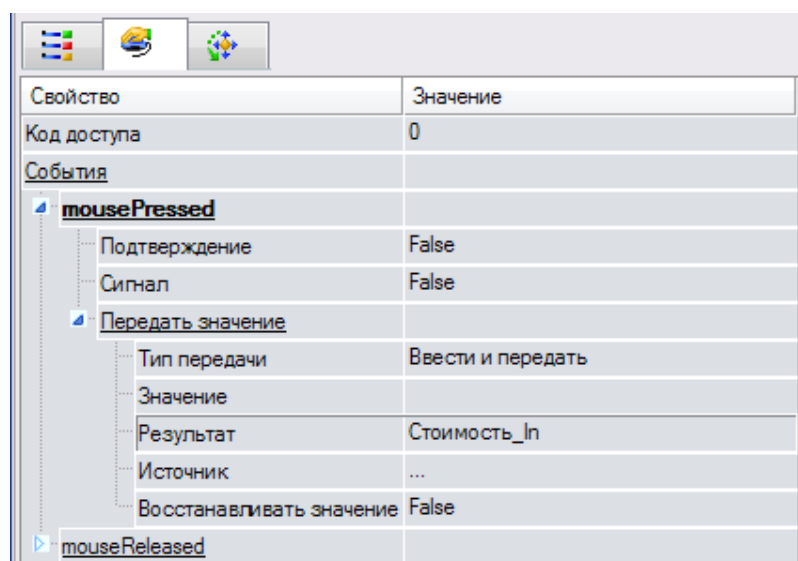


Рис. 3.45 Настройка события

Имя	Тип	Тип данных	Привязка	Фла
Уровень_R	IN	REAL	Уровень:Реальное значение (Система.RTM.Каналы)	
Период_генерации_In	OUT	REAL	Период генерации:Входное значение (Система.RTM.Каналы)	
Расход_In	OUT	REAL	Расход:Входное значение (Система.RTM.Каналы)	
Стоимость_In	OUT	REAL	Стоимость:Входное значение (Система.RTM.Каналы)	

Рис. 3.46 Атрибуты для ввода данных

4. **Создание объектов текст.** Для вывода значений, хранимых в ряде каналов, создайте объекты текст. Пример создания объектов текст изображен на рис. 3.47. Объекты текст, расположенные слева, выводят подсказку (где какой параметр выведен), а справа— значения, хранимые каналами. Привязку объектов текст, расположенных в правом столбце,

производите к реальным значениям соответствующих каналов. Для привязки, при необходимости, создайте дополнительные аргументы, как показано на рис. 3.48

Период генерации, с	<текст>	Установить период
Расход, л/с	<текст>	Установить расход
Стоимость руб/л	<текст>	Установить стоимость
Суммарный расход, л	<текст>	
Суммарная стоимость, руб	<текст>	

Рис. 3.47 Создание объектов текст

Имя	Тип	Тип данных	Привязка
Уровень_R	IN	REAL	Уровень:Реальное значение (Система.RTM.Каналы)
Период_генерации_In	OUT	REAL	Период генерации:Входное значение (Система.RTM.Каналы)
Расход_In	OUT	REAL	Расход:Входное значение (Система.RTM.Каналы)
Стоимость_In	OUT	REAL	Стоимость:Входное значение (Система.RTM.Каналы)
Период_генерации_R	IN	REAL	Период генерации:Реальное значение (Система.RTM.Каналы)
Расход_R	IN	REAL	Расход:Реальное значение (Система.RTM.Каналы)
Стоимость_R	IN	REAL	Стоимость:Реальное значение (Система.RTM.Каналы)
Суммарный_расход_R	IN	REAL	Суммарный расход:Реальное значение (Система.RTM.Каналы)
Суммарная_стоимость_R	IN	REAL	Суммарная стоимость:Реальное значение (Система.RTM.Каналы)

Рис. 3.48 Атрибуты экрана

5. Произведите расчет множителя K и смещения C .

Масштабирование сигнала в FBD программе будет производиться по формуле:

$$Y = K \cdot X + C, \text{ где}$$

Y — результат преобразования;

X — входное значение канала;

K — множитель;

C — смещение.

FBD блоки, производящие генерацию сигнала возвращают сигнал, изменяющийся в диапазоне $[-1; +1]$. Найдите значение множителя K и смещения Z , которые позволят получить требуемый диапазон изменения результата преобразования (диапазон сигнала).

Пример: необходимо обеспечить диапазон сигнала $[0; 1]$. Множитель K составит 0,5, смещение Z — 0,5.

6. **Создание FBD диаграммы.** Выделите группу **каналы** RTM узла и вызовите контекстное меню. В появившемся меню выберите **создать компонент**. Среди предложенных вариантов выберите **программа** (рис. 3.49). Выделив созданную программу и, щелкнув левой клавишей мыши, можете изменить имя программы, например, на «синусоида». Выделите созданный объект-программу и вызовите контекстное меню. В появившемся меню выберите **редактировать**. Откроется окно, аналогичное редактированию канала float. В поле период выберите **1**, выберите единицу измерения— **секунду** (сек). После настройки периода пересчета окно редактирования канала можно закрыть. Двойным щелчком левой клавиши мыши по созданной программе или вызвав контекстное меню и выбрав строчку **редактировать шаблон**, вызовите окно для создания (или редактирования) программы. Откроется окно, представленное на рис. 3.50.

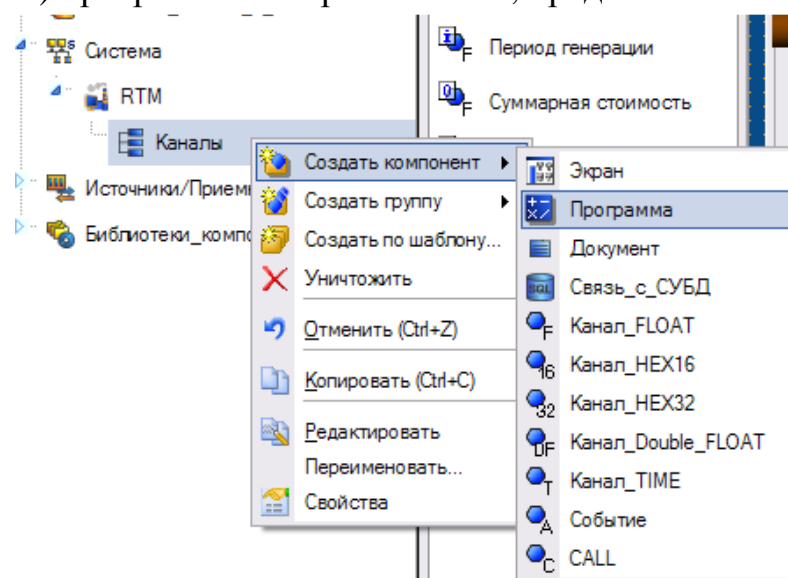


Рис. 3.49 Создание объекта программа

В структуре программы выделите строчку **аргументы**. Заполните появившуюся таблицу. Каждая строка— аргумент программы. Установите тип In для аргументов, которые будут передавать данные в программу, Out— которые будут передавать данные из программы, In/Out— которые будут передавать в программу и из нее. Пример создания аргументов программы для генерирования синусоидального сигнала приведен на рис. 3.51.

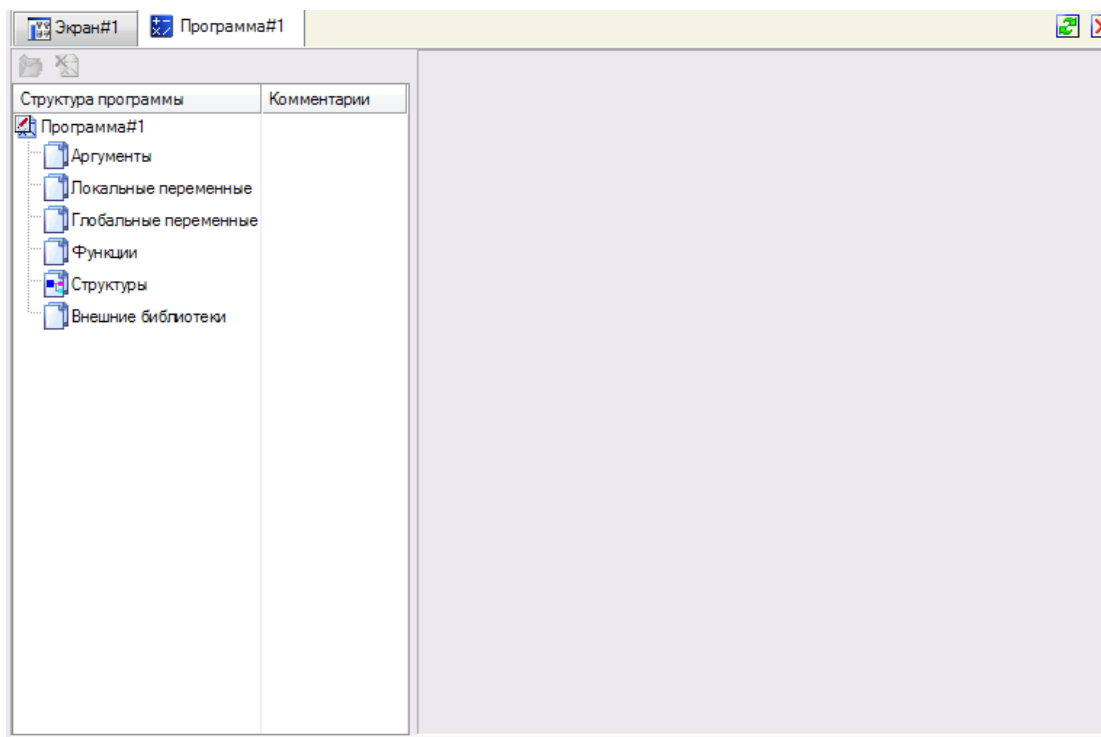


Рис. 3.50 Окно редактирования шаблона программы

Имя	Тип	Тип данных	Привязка
Период генерации_R	IN/OUT	REAL	Период генерации: Реальное значение (Система.RTM.Каналы)
Уровень_In	OUT	REAL	Уровень: Входное значение (Система.RTM.Каналы)

Рис. 3.51 Атрибуты программы на языке Техно FBD

Выделите строку **программа#** в структуре программы. Среди предложенных языков программирования выберите FBD диаграмму (рис. 3.52).

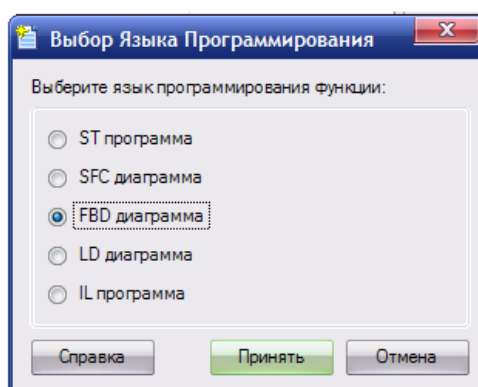



Рис. 3.52 Выбор языка

Щелкните левой клавишей мыши по иконке  или выберите строку **палитра FBD блоков** в меню **вид**, если палитра FBD блоков (рис. 3.52) не открыта.

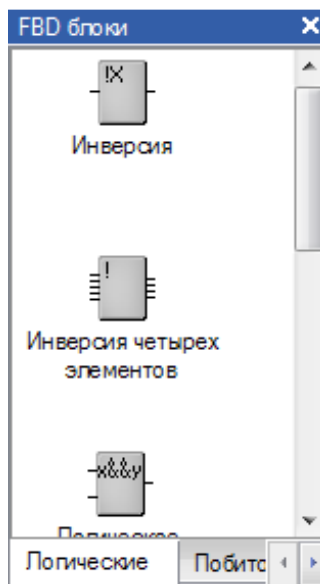




Рис. 3.52 Окно выбора FBD блоков

Перейдите на закладку **сравнение**. Нажмите левую клавишу мыши на блоке **больше или равно**. Переместите его на рабочее поле и отпустите клавишу мыши. Аналогично разместите на рабочем поле FBD блоки (рис. 3.53): **выбор из двух** (закладка **выбор**), **синусоидальный сигнал** (закладка **синусоидальный сигнал**), **масштабирование** (закладка **арифметические**).

Произведите привязку блоков между собой. Для привязки выхода одного блока к входу другого наведите курсор мыши на соответствующий выход (вход) и нажмите левую клавишу мыши. Не отпуская левой клавиши мыши, наведите курсор на необходимый вход (выход) и отпустите левую клавишу мыши. Для привязки аргумента программы к соответствующему входу(выходу) выделите соответствующий вход (выход) и вызовите контекстное меню. В появившемся меню выберите **привязать**. Среди предлагаемых аргументов выберите необходимый (рис. 3.53). Для создания константы выберите вход, вызовите контекстное меню и выберите **привязать**. В появившемся поле вместо выбора аргумента программы введите константу. На вход **IN2** блока **больше или равно** и **IN0** блока **выбора из двух** следует подавать константу **10**, согласно заданию. Вход **IN2** блока **больше или равно** и **IN0** блока **выбора из двух** следует привязать к аргументу, который передает период генерации в программу. Вычисленные ранее значения констант **K** и **C** следует подать на соответствующие входы

блока *масштабирование*. Программа, генерирующая сигнал, изменяющийся в диапазоне от [0;1] примет вид, изображенный на рис. 3.53. После создание программы проверьте ее. Для этого щелкните левой клавишей мыши по иконке «компиляция» . В окне «сообщения» будет выведен результат компиляции. Если окно «сообщения» закрыто, то следует щелкнуть по иконке  или выбрать в меню «вид» окно сообщения. При отсутствии ошибок будет написано: «*Программа#1.tms compiled successfully*», к примеру. При наличии ошибки будет написано: «*E0011 Синтаксическая ошибка*», к примеру.

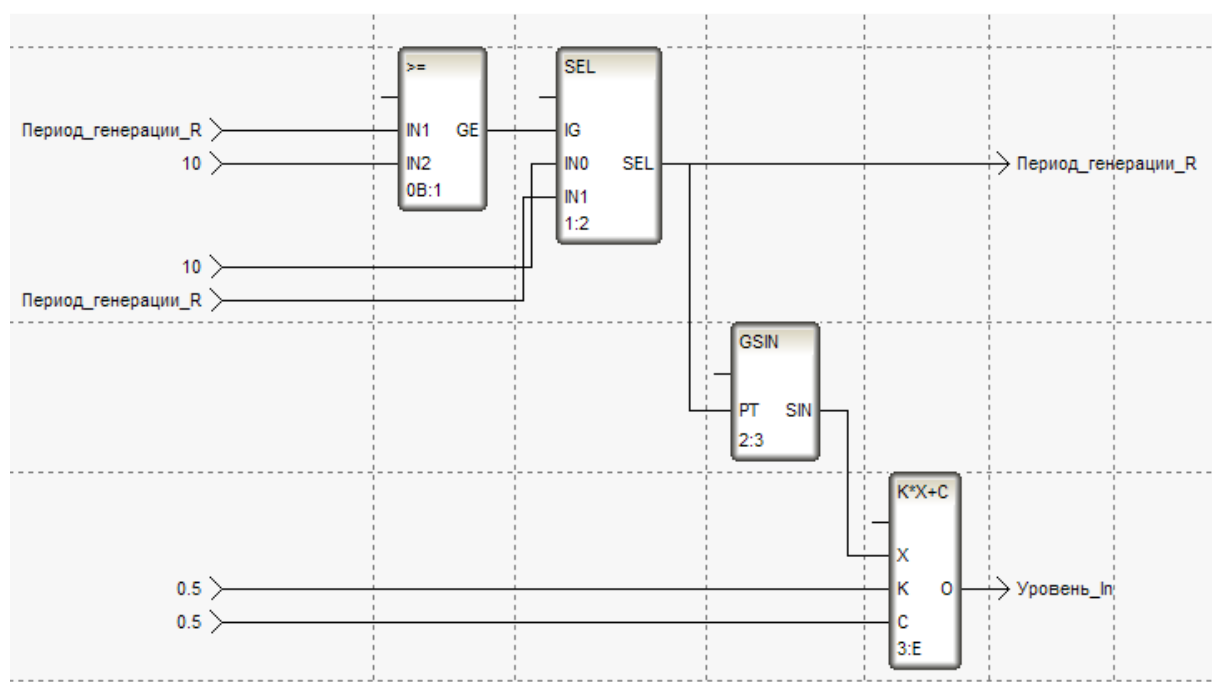


Рис. 3.53 Программа на языке Техно FBD

7. **Создание ST программы.** Создайте объект *программа* в группе *каналы* RTM узла. Название программы можно изменить на «обработка», к примеру. Установите период пересчета равным 1 секунде. Откройте программу двойным щелчком левой клавиши мыши по созданному объекту программы. В структуре программы выберите *аргументы*. Создайте аргументы, которые будут передавать в программу расход продукта, стоимость продукта, период пересчета каналов, а также аргументы, которые будут возвращать из программы суммарный расход продукта и суммарную стоимость всего израсходованного продукта (рис. 3.54).

Имя	Тип	Тип данных	Привязка
Расход	IN/OUT	REAL	Расход: Реальное значение (Система.RTM.Каналы)
Стоимость	IN/OUT	REAL	Стоимость: Реальное значение (Система.RTM.Каналы)
T	IN	REAL	Уровень: Период пересчета (значение) (Система.RTM.Каналы)
Суммарный_расход	OUT	REAL	Суммарный расход: Входящее значение (Система.RTM.Каналы)
Суммарная_стоимость	OUT	REAL	Суммарная стоимость: Входящее значение (Система.RTM.Каналы)

Рис. 3.54 Аргументы программы на языке Techno ST

Выберите строку **глобальные переменные**. Аналогично созданию аргументов создайте глобальную переменную, которая будет хранить значение суммарного расхода, вычисленного при предыдущем вызове программы, указав **начальное значение** равным нулю (рис. 3.55).

Имя	Тип данных
Предыдущий_суммарный_расход	REAL

Рис. 3.55 Глобальная переменная

Выделите строку **программа#**. В открывшемся окне наберите ST-программу. При указанных выше именах аргументов и глобальной переменной текст программы примет вид, указанный на рис. 3/56.

```

PROGRAM
  VAR_INOUT Расход : REAL; END_VAR
  VAR_INOUT Стоимость : REAL; END_VAR
  VAR_INPUT T : REAL; END_VAR
  VAR_OUTPUT Суммарный_расход : REAL; END_VAR
  VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR

  If Расход < 1 Then
    Расход = 1;
  End_if;
  If Стоимость < 1 Then
    Стоимость = 1;
  End_if;
  Суммарный_расход = Предыдущий_суммарный_расход + Расход * T;
  Предыдущий_суммарный_расход = Суммарный_расход;
  Суммарная_стоимость = Суммарный_расход * Стоимость;

END_PROGRAM

```

Рис. 3.56 Программа на языке Techno ST

Проверьте программу, щелкнув левой клавишей мыши по иконке .

8. **Запустите проект.** Запустите проект, аналогично предыдущим работам (см. работу 1). Пример результата запуска приведен на рис. 3.57

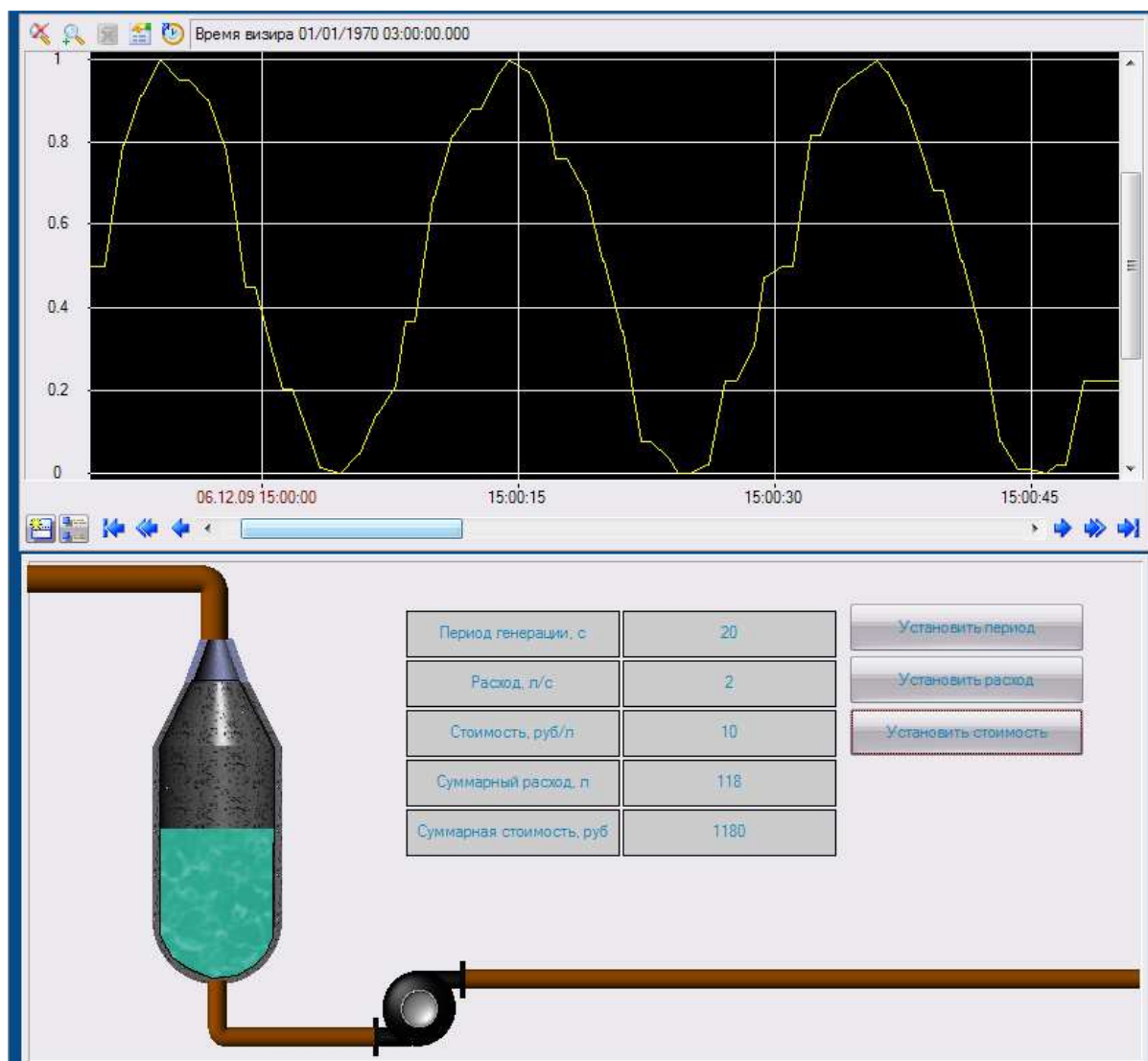


Рис. 2.54 Пример результата запуска проекта

Вопросы для самоконтроля:

1. Опишите создание объекта кнопка и ее редактирование.
2. Опишите создание программы в среде Trace Mode.
3. Опишите создание атрибутов программы и переменных.
4. Опишите правила создания числовых констант.
5. Опишите правила создания строковых констант.
6. Синтаксис, назначение оператора if в языке ST.
7. Синтаксис, назначение оператора case в языке ST.
8. Синтаксис, назначение оператора while в языке ST.
9. Синтаксис, назначение оператора repeat в языке ST.
10. Синтаксис, назначение оператора for в языке ST.
11. Что представляет из себя FBD программа и как она работает?
12. Опишите выбор FBD блока, размещение его на рабочем поле.
13. Опишите привязку FBD блока к переменным, атрибутам, создание констант.
14. Какие группы языков программирования можно выделить в SCADA-системах?
15. Приведите примеры графических языков программирования.
- 16.

Работа 4 Программирование на языках

Texno IL и Texno SFC

Цель работы: изучить языки программирования Texno IL и Texno SFC, создать АСУ ТП, с использованием указанных языков Texno IL, SFC, FBD.

Задание:










1. взять за основу АСУ ТП, созданную при выполнении третьей лабораторной работы (программирование в среде Trace Mode на языках Texno ST и Texno FBD). Удалить обе программы, написанные на языках Texno ST и Texno FBD;
2. добавить выключатель на экране, создать канал, определяющий необходимость определения суммарного расхода и суммарной стоимости продукта и к которому производится привязка выключателя;
3. создать программу на языке Texno SFC, которая будет содержать следующие шаги и переходы:
 - 3.1 первый шаг: выполнение проверок значения периода дискретизации и значения расхода, стоимости продукта на языке Texno IL; если период дискретизации окажется меньше 10, то установите значение по умолчанию (10), если расход или стоимость продукта окажутся меньше 1, то установите соответствующий параметр равным 1, измененные значения программа должна возвращать соответствующим каналам;
 - 3.2 Второй шаг: генерирование уровня в емкости на языке Texno FBD аналогично третьей лабораторной работе без проверки периода генерации (он проверен на первом шаге); в качестве условия перехода следует записать константу true;
 - 3.3 третий шаг: выполняемый, когда выключатель возвращает 1, реализованный, с использованием языка Texno IL: вычисление суммарного расхода, суммарной стоимости продукта аналогично третьей лабораторной работе с использованием глобальной переменной для хранения предыдущего значения суммарного расхода, вычисленные

значения суммарного расхода, суммарной стоимости программа должна возвращать соответствующим каналам;

3.4 четвертый шаг: выполняемый, когда выключатель возвращает 0, реализованный с помощью языка Техно Л: вычисляется суммарный расход продукта, результат вычисления не возвращается программой соответствующему каналу, а только присваивается глобальной переменной, предназначенной для хранения предыдущего значения суммарного расход.

Примечание: период пересчета у всех каналов, программы, должен быть один и тот же в пределах 1— 3 с.

Ход работы

1. **Создание проекта.** Сохраните проект, созданный в ходе выполнения работы 3 под новым именем. Удалите обе программы.
2. **Создание выключателя.** Создайте еще один канал . Установите период пересчета равным 1 секунде. Для удобства восприятия назовем новый канал «выключатель». Для создания выключателя необходимо щелкнуть левой клавишей мыши по иконке выключателя на рабочем столе. На панели инструментов может быть одна из следующих иконок: , , , , , , , , . Все приведенные переключатели обладают одинаковыми возможностями. Поэтому не важно, какой переключатель будет расположен на экране. Достаточно выбрать тот, который размещен на панели инструментов. После выбора инструмента для создания выключателя щелкните левой клавишей мыши там, где хотите поместить выключатель. Пример размещения выключателя приведен на рис. 2.55.

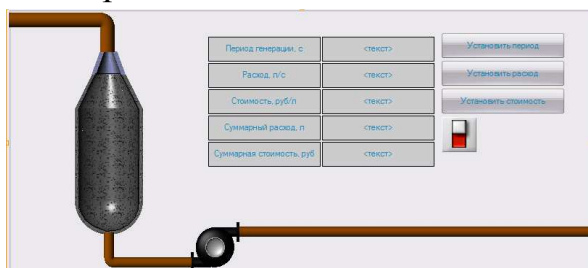


Рис. 3.55 Размещение переключателя

Откройте окно свойств объекта «выключатель». Произведите настройку, как показано на рис. 3.56, где Выключатель_b1— аргумент, который служит для привязки к первому биту канала выключатель(рис. 3.57).

Свойство	Значение
Привязка	Выключатель_b1
Вид индикации	Arg & Конст
Инверсия	False
Константа	0x1
Код доступа	0
Значение	0x1
Подсказка	
Выделение в MPB	False


Рис. 3.56 Настройка переключателя

Имя	Тип	Тип данных	Привязка
Уровень_R	IN	REAL	Уровень:Реальное значение (Система.RTM.Каналы)
Период_генерации_In	OUT	REAL	Период генерации:Входное значение (Система.RTM.Каналы)
Расход_In	OUT	REAL	Расход:Входное значение (Система.RTM.Каналы)
Стоимость_In	OUT	REAL	Стоимость:Входное значение (Система.RTM.Каналы)
Период_генерации_R	IN	REAL	Период генерации:Реальное значение (Система.RTM.Каналы)
Расход_R	IN	REAL	Расход:Реальное значение (Система.RTM.Каналы)
Стоимость_R	IN	REAL	Стоимость:Реальное значение (Система.RTM.Каналы)
Суммарный_расход_R	IN	REAL	Суммарный расход:Реальное значение (Система.RTM.Каналы)
Суммарная_стоимость_R	IN	REAL	Суммарная стоимость:Реальное значение (Система.RTM.Каналы)
Выключатель_b1	IN/OUT	BOOL	Выключатель:Бит 1 (Система.RTM.Каналы)

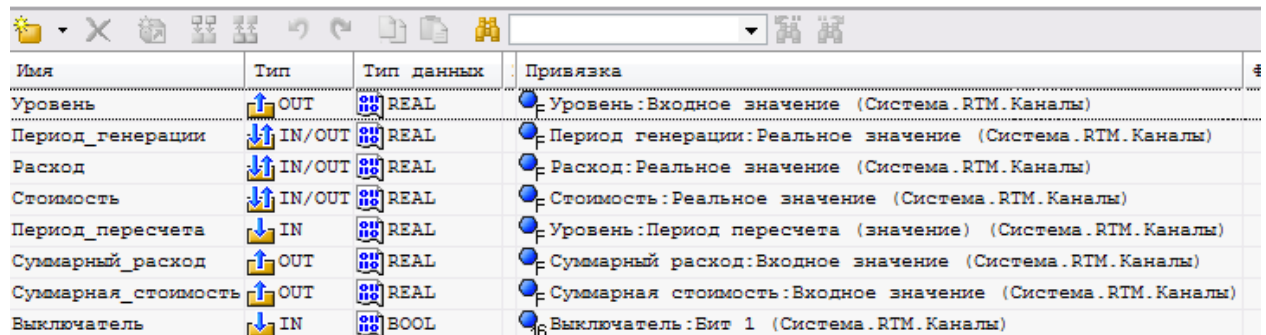
Рис. 3.57 Аргументы экрана

3. **Создание SFC диаграммы.** Создайте объект программа . Создайте атрибуты, аналогично работе 3. Помимо атрибутов, использовавшихся в программах работы 3, понадобится атрибут, передающий в программу реальное значение канала, который хранит состояние выключателя. Пример создания атрибутов приведен на рис. 3.58.

Создайте глобальную переменную, для хранения результата вычисления суммарного расхода при предыдущем вызове программы (рис. 3.55).

Выделите строку **программа#** в структуре программы. Среди предложенных языков программирования выберите SFC диаграмму. На рабочем поле отображается только один шаг: начальный шаг (рис. 3.59). Дважды щелкните по данному шагу левой клавишей мыши. Введите новое имя шага, к примеру, «проверка исходных данных». Для создания нового шага выделите единственный существующий шаг. Щелкните левой клавишей мыши по иконке  или вызовите контекстное меню и выберите **создать шаг/переход**. Будет создан новый шаг и переход. Двойным

щелчком по созданному шагу и переходу измените их имя. Шаг можно назвать как «генерирование сигнала», к примеру, переход— «true». Выделите шаг «генерирование сигнала» и создайте новый шаг и переход. Созданный шаг и переход можно назвать как «полный расчет» и «реальное значение выключателя», к примеру. Выделите шаг «генерирование сигнала». Создайте новый шаг и переход. Новый шаг и переход можно назвать как «расчет только предыдущего значения суммарного значения» и «реальное значение выключателя с отрицанием», к примеру. Внешний вид созданной SFC диаграммы приведен на рис. 3.60.



Имя	Тип	Тип данных	Привязка
Уровень	OUT	REAL	Уровень:Входное значение (Система.RTM.Каналы)
Период_генерации	IN/OUT	REAL	Период генерации:Реальное значение (Система.RTM.Каналы)
Расход	IN/OUT	REAL	Расход:Реальное значение (Система.RTM.Каналы)
Стоимость	IN/OUT	REAL	Стоимость:Реальное значение (Система.RTM.Каналы)
Период_пересчета	IN	REAL	Уровень:Период пересчета (значение) (Система.RTM.Каналы)
Суммарный_расход	OUT	REAL	Суммарный расход:Входное значение (Система.RTM.Каналы)
Суммарная_стоимость	OUT	REAL	Суммарная стоимость:Входное значение (Система.RTM.Каналы)
Выключатель	IN	BOOL	Выключатель:Бит 1 (Система.RTM.Каналы)

Рис. 3.58 Аргументы программы

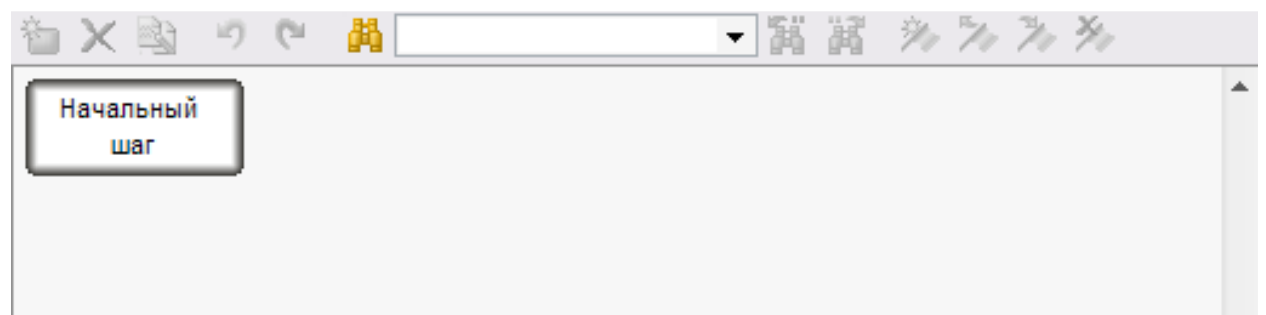


Рис. 3.59 Новая программа на языке Texno SFC

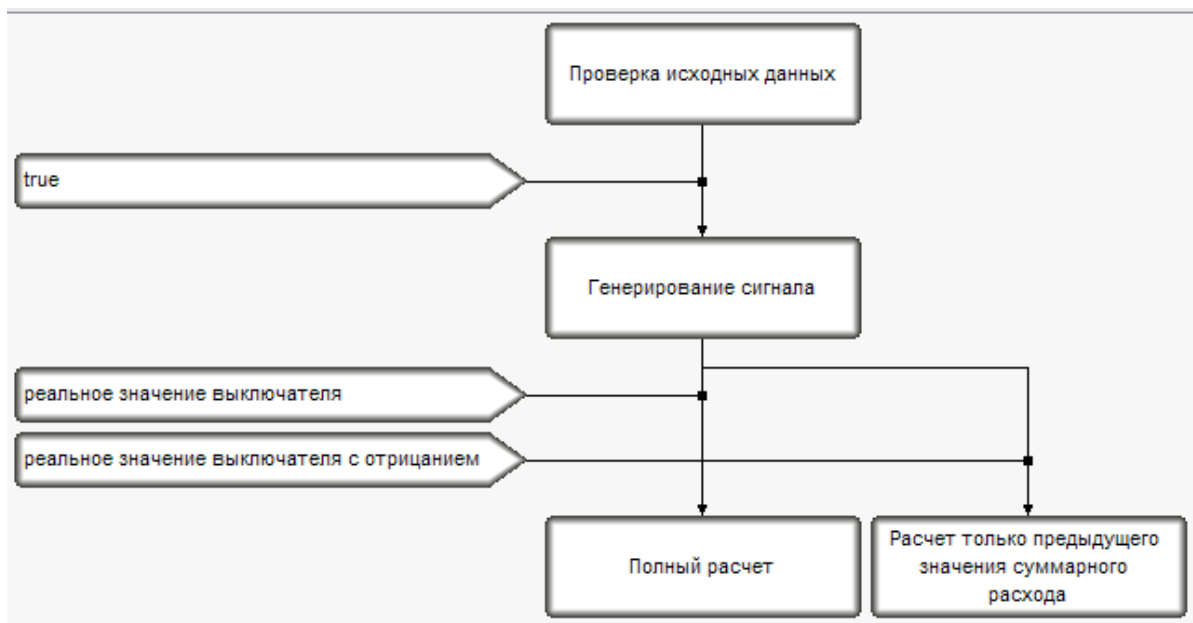


Рис. 3.60 Программа на языке SFC

4. Создание шагов

а. **Создание шага «проверка исходных данных».** В структуре программы раскройте раздел *SFC диаграмма*. Раскройте раздел *шаги* и *переходы*. Выделите строку *проверка исходных данных*. Среди предложенных языков выберите *TechnoIL*.

При указанных выше именах атрибутов и глобальной переменной текст программы, производящей проверку исходных данных примет вид:

```

SFC_STEP "Проверка исходных данных"
VAR_OUTPUT Уровень : REAL; END_VAR
VAR_INOUT Период_генерации : REAL; END_VAR
VAR_INOUT Расход : REAL; END_VAR
VAR_INOUT Стоимость : REAL; END_VAR
VAR_INPUT Период_пересчета : REAL; END_VAR
VAR_OUTPUT Суммарный_расход : REAL; END_VAR
VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR
VAR_INPUT Выключатель : REAL; END_VAR
  
```

```

GE Расход 1//проверка расхода
  
```

```

JMPC Проверка_стоимости//переход к проверке стоимости при
расходе > 1
  
```

```

LD 1
  
```

ST Расход//Расход по умолчанию
 Проверка_стоимости: GE Стоимость 1
 JMPC Проверка_периода//переход к проверке периода генерации,
 когда стоимость > 1
 LD 1
 ST Стоимость//Стоимость по умолчанию
 Проверка_периода: GE Период_генерации 10
 JMPC конец//выход при периоде > 10
 LD 10
 ST Период_генерации//Период генерации по умолчанию
 конец:

END_SFC_STEP

в. **Создание шага «генерирование сигнала».** Выделите строчку **генерирование сигнала**. Среди предложенных языков выберите **FBD**.
 Создайте FBD программу, как показано на рис. 3.61

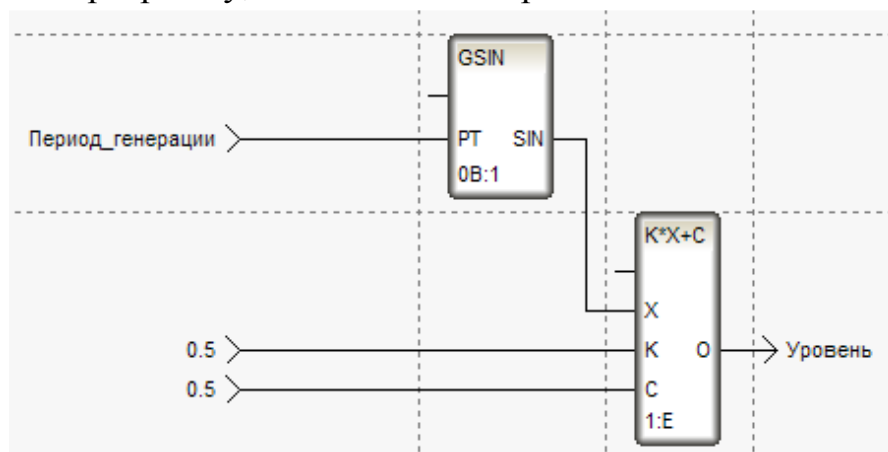


Рис. 3.61 Шаг для генерации сигнала

Для создания перехода к созданному шагу выделите строчку true в разделе **переходы**. среди предложенных языков выберите **ST**. Текст программы примет вид:

```

SFC_TRANSITION "true" FROM( INITIAL_STEP ) TO(
STEP_1 )

VAR_OUTPUT Уровень : REAL; END_VAR
VAR_INOUT Период_генерации : REAL; END_VAR
VAR_INOUT Расход : REAL; END_VAR
VAR_INOUT Стоимость : REAL; END_VAR
VAR_INPUT Период_пересчета : REAL; END_VAR
  
```

```
VAR_OUTPUT Суммарный_расход : REAL; END_VAR
VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR
VAR_INPUT Выключатель : REAL; END_VAR
```

true

END_SFC_TRANSITION

с. **Создание шага «полный расчет».** Выберите строку **полный расчет**. Среди предложенных языков выберите **IL**. Текст программы примет вид:

```
SFC_STEP "Полный расчет"
VAR_OUTPUT Уровень : REAL; END_VAR
VAR_INOUT Период_генерации : REAL; END_VAR
VAR_INOUT Расход : REAL; END_VAR
VAR_INOUT Стоимость : REAL; END_VAR
VAR_INPUT Период_пересчета : REAL; END_VAR
VAR_OUTPUT Суммарный_расход : REAL; END_VAR
VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR
VAR_INPUT Выключатель: REAL; END_VAR

MUL Период_пересчета Расход//вычисление расхода за период
пересчета
ADD Предыдущий_суммарный_расход//Нахождение суммарного
расхода
ST Предыдущий_суммарный_расход//присвоение вычисленного
значения суммарного расчета
ST Суммарный_расход
MUL Стоимость//вычисление суммарной стоимости
ST Суммарная_стоимость//присвоение аргументу результата
вычисления суммарной стоимости
```

END_SFC_STEP

Для создания перехода к созданному шагу выделите строку **реальное значение выключателя**. Среди предложенных языков выберите **ST**. Текст программы примет вид:

SFC_TRANSITION "Реальное значение выключателя" FROM(STEP_1) TO(STEP_2)

VAR_OUTPUT Уровень : REAL; END_VAR

VAR_INOUT Период_генерации : REAL; END_VAR

VAR_INOUT Расход : REAL; END_VAR

VAR_INOUT Стоимость : REAL; END_VAR

VAR_INPUT Период_пересчета : REAL; END_VAR

VAR_OUTPUT Суммарный_расход : REAL; END_VAR

VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR

VAR_INPUT Выключатель : BOOL; END_VAR

Выключатель

END_SFC_TRANSITION

d. Создание шага «расчет только предыдущего значения суммарного расчета». Выделите строку *расчет только предыдущего значения суммарного расчета*. Среди предложенных языков выберите *IL*. Текст программы примет вид:

SFC_STEP "Расчет только предыдущего значения суммарного расхода"

VAR_OUTPUT Уровень : REAL; END_VAR

VAR_INOUT Период_генерации : REAL; END_VAR

VAR_INOUT Расход : REAL; END_VAR

VAR_INOUT Стоимость : REAL; END_VAR

VAR_INPUT Период_пересчета : REAL; END_VAR

VAR_OUTPUT Суммарный_расход : REAL; END_VAR

VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR

VAR_INPUT Выключатель : REAL; END_VAR

MUL Период_пересчета Расход//вычисление расхода за период пересчета

ADD Предыдущий_суммарный_расход//Нахождение суммарного расхода

END_SFC_STEP

Для создания перехода к созданному шагу выделите строку *реальное значение выключателя с отрицанием*. Среди предложенных языков выберите *ST*. Текст программы примет вид:

```
SFC_TRANSITION "Реальное значение выключателя с
отрицанием" FROM( STEP_1 ) TO( STEP_3 )
VAR_OUTPUT Уровень : REAL; END_VAR
VAR_INOUT Период_генерации : REAL; END_VAR
VAR_INOUT Расход : REAL; END_VAR
VAR_INOUT Стоимость : REAL; END_VAR
VAR_INPUT Период_пересчета : REAL; END_VAR
VAR_OUTPUT Суммарный_расход : REAL; END_VAR
VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR
VAR_INPUT Выключатель : BOOL; END_VAR
```

!Выключатель

END_SFC_TRANSITION

5. **Запуск проекта.** Запустите проект аналогично первой работе. При обоих положениях переключателя должно происходить заполнение емкости аналогично работе 3. При одном из положений переключателя должен производиться вывод новых полученных значений суммарного расхода и стоимости, при другом— новые значения суммарного расхода и стоимости не выводятся.

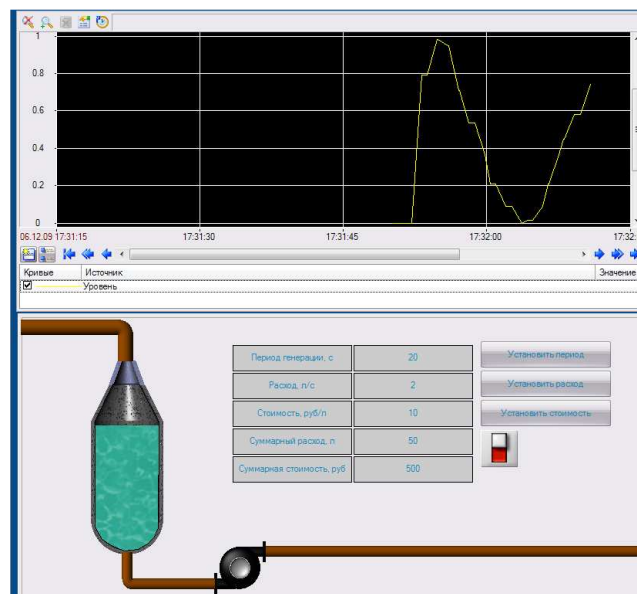


Рис. 3.62 Пример исполнения программы

Вопросы для самоконтроля

1. Что представляет из себя SFC программа и как она работает.
2. Опишите создание нового шага, цикла, параллельных шагов.
3. Опишите операнды и модификаторы языка IL.
4. Опишите синтаксис и назначения операторов для обмена данными с аккумулятором.
5. Опишите синтаксис и назначение арифметических операторов.
6. Опишите синтаксис и назначение операторов перехода и вызова.
7. Что такое аккумулятор в языке IL и как с ним работать.
8. Какие группы языков программирования можно выделить в SCADA-системах?
9. Приведите примеры графических языков программирования.

Работа 5 Программирование на языках

Создание отчета тревог и СПАД архива

Цель работы: познакомиться с отчетом тревог, СПАД архивом, создать отчет тревог, архив значений.

Задание:

1. взять за основу четвертую лабораторную работу (программирование в среде Trace Mode на языках Техно IL и Техно SFC);
2. заменить тренд на архивный тренд;
3. настроить анализ границ канала, передающего уровень продукта в емкости;

4. создать словарь сообщений для канала, передающего уровень продукта в емкости, настроить узел для создания отчета тревог;
5. изменить динамическую заливку емкости изображением на заливку цветом и настроить выбор цвета в соответствии с принадлежностью значения уровня тому или иному диапазону;
6. настроить SPAD архивирование канала, хранящего уровень продукта.

Ход работы

1. **Создание проекта.** Сохраните проект, созданный при выполнении работы №4. Удалите тренд. Разместите на экране архивный тренд, который настройте на вывод уровня продукта в емкости. Желательно изменить период колебаний по умолчанию с 10 на 40. Шаг, производящий проверку исходных данных примет вид:

SFC_STEP "Проверка исходных данных"

VAR_OUTPUT Уровень : REAL; END_VAR

VAR_INOUT Период_генерации : REAL; END_VAR

VAR_INOUT Расход : REAL; END_VAR

VAR_INOUT Стоимость : REAL; END_VAR

VAR_INPUT Период_пересчета : REAL; END_VAR

VAR_OUTPUT Суммарный_расход : REAL; END_VAR

VAR_OUTPUT Суммарная_стоимость : REAL; END_VAR

VAR_INPUT Выключатель : BOOL; END_VAR

GE Расход 1

JMPC Проверка_стоимости

LD 1

ST Расход

Проверка_стоимости: GE Стоимость 1

JMPC Проверка_периода

LD 1

ST Стоимость

Проверка_периода: GE Период_генерации 10

JMPC Конец

LD 40

ST Период_генерации

Конец:

END_SFC_STEP

2. **Настройка контроля границ.** Выделите канала **уровень** в группе **каналы** RTM узла и вызовите контекстное меню. Выберите строку **редактировать** в появившемся меню. На панели **границы** поставьте флаг **использовать**, установите границы диапазонов, как показано на рис. 3.63. Установите флаг контроль границ.

Границы	
<input checked="" type="checkbox"/> Использовать	
ВП	1
ВА	0.9
ВГ	0.8
НГ	0.2
НА	0.1
НП	0
Гистерезис	0
<input checked="" type="checkbox"/> Контроль границ	

Рис. 3.63 Задание границ канала

3. **Создание словаря сообщений.** Выделите узел **RTM** и вызовите контекстное меню (рис. 3.64). Выберите строку **создать группу**. Среди

предложенных групп выберите строку **словари_сообщений**. Выделите созданную группу **словари_сообщений** и вызовите контекстное меню. Среди предложенных словарей выберите **словарь_для_FLOAT** (рис. 3.65).

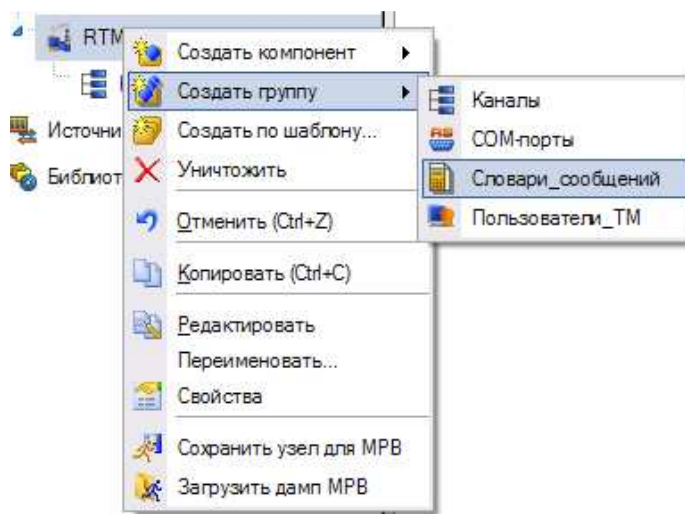


Рис. 3.64 Создание группы для словарей сообщений

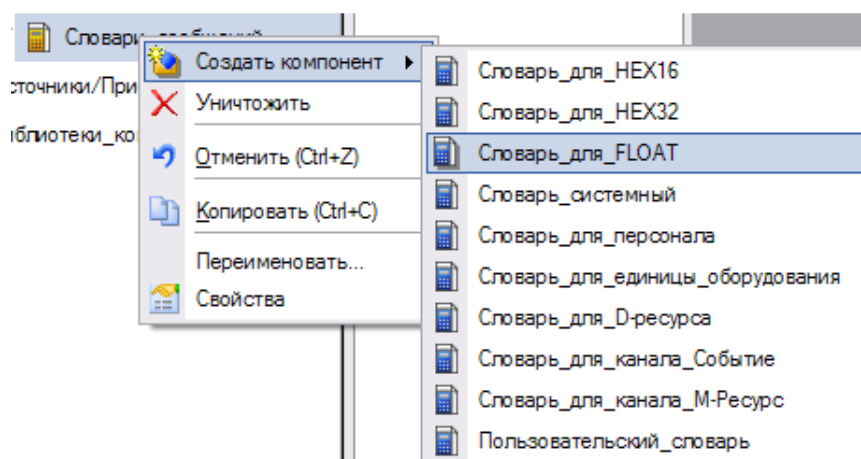


Рис. 3.65 Создание словаря сообщений

4. **Редактирование словаря.** Двойным щелчком по созданному словарю сообщений или выделив его, вызвав контекстное меню и выбрав **редактировать** откройте окно для редактирования словаря. В поле **имя** введите имя словаря. Двойным щелчком левой клавиши мыши на каждой строчке откройте окно для редактирования сообщения (рис. 3.66). Редактируя все сообщения, настройте словарь сообщений, как показано на рис. 3.67

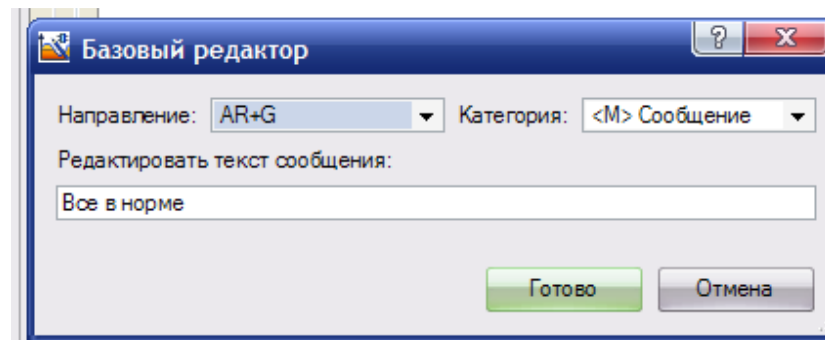


Рис. 3.66 Окно для задания сообщения

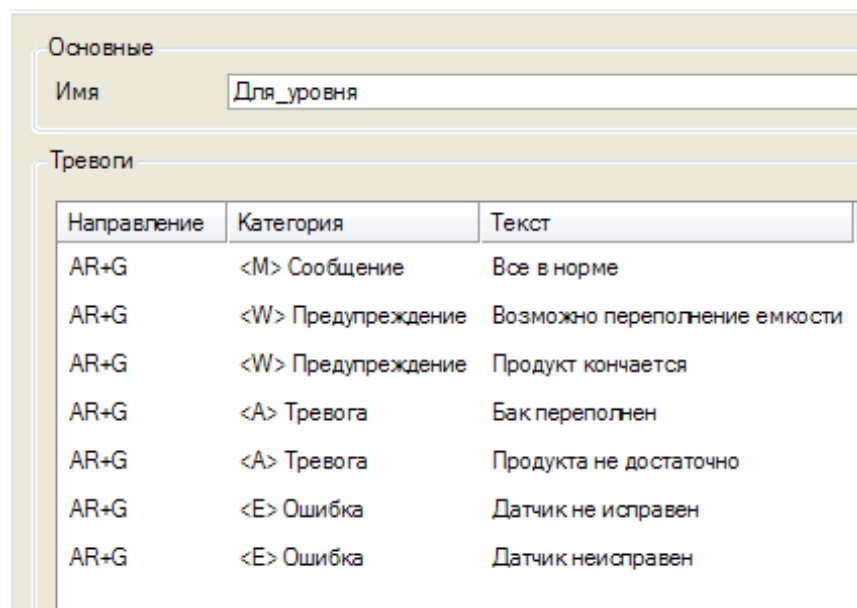


Рис. 3.67 Редактор словаря сообщений

5. **Редактирование узла.** Выделите узел **RTM**. Вызовите контекстное меню и выберите **редактировать**. Перейдите на закладку **отчет тревог/дамп/параметры**. Произведите заполнение полей, как показано на рис. 3.68. Перейдите на закладку **архивы**. Произведите заполнение полей, как показано на рис. 3.69.

6.

Рис. 3.68 Редактирование узла

Рис. 3.69 Редактирования узла

7. **Редактирование канала.** В окне редактирования канала *уровень* задайте границы, на закладке *архивация* настройте архивацию и создание отчета тревог, как показано на рис. 3.70.

Рис. 3.70 Редактирование канала

8. **Настройка заливки емкости.** Откройте окно свойств объекта **многоугольник** (элемент изображения, обеспечивающий динамическую заливку). Для получения зависимости цвета заливки от состояния контролируемого процесса произведите настройку, как показано на рис. 3.71.

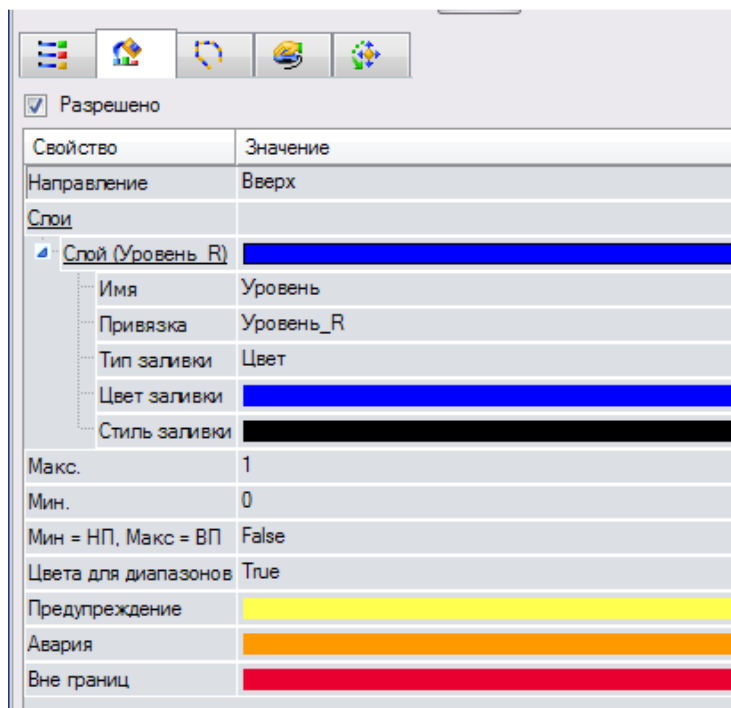


Рис. 3.71 Настройка изменения цвета динамической заливки

9. **Запустите проект на исполнение.** Результат запуска проекта на исполнение будет отличаться тем, что будет выводиться результат предыдущих запусков проекта на исполнение, периодически будет происходить обновление тренда, который строит зависимость на основе данных архива. Цвет заливки емкости будет определяться уровнем продукта в емкости (рис. 3.72). Посмотрите созданный отчет тревог, который размещается в директории с именем узла, размещенной в директории проекта.

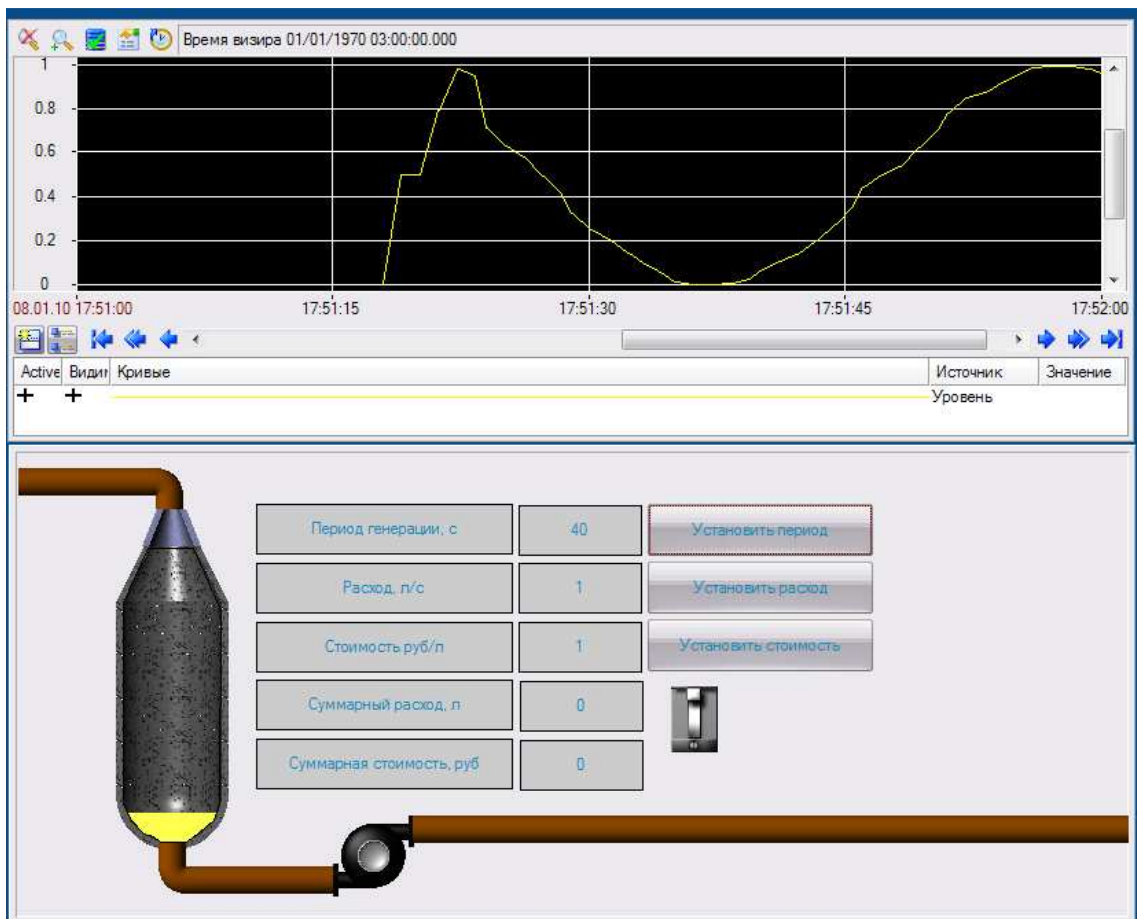


Рис. 3.72 Пример запуска проекта

Вопросы для самоконтроля

1. Отчет в SCADA–системах.
2. Отчет тревог в среде Trace Mode.
3. Создание отчета тревог в среде Trace Mode.
4. Границы канала и их анализ в среде Trace Mode.
5. Словарь сообщений в среде Trace Mode.
6. СПАД архив и его создание в среде Trace Mode.
7. Изменение цвета динамической заливки в среде Trace Mode.
8. Отличие события от тревоги.

9. Аналоговые тревоги SCADA–систем.
10. Дискретные тревоги SCADA–систем.
11. Отчет тревог.
12. Как пользователь может получить сведения о сгенерированных тревогах.
13. Что такое отчет SCADA–систем, зачем его используют.
14. Как пользователь может получить отчет тревог?
15. При каких условиях SCADA–системы позволяют сформировать отчет.
16. Опишите отчет в виде html страницы.
17. Опишите табличную форму отчета.

Пример отчета Trace Mode

Ведомость расхода тепла, холодной и горячей воды за 01.08.2005

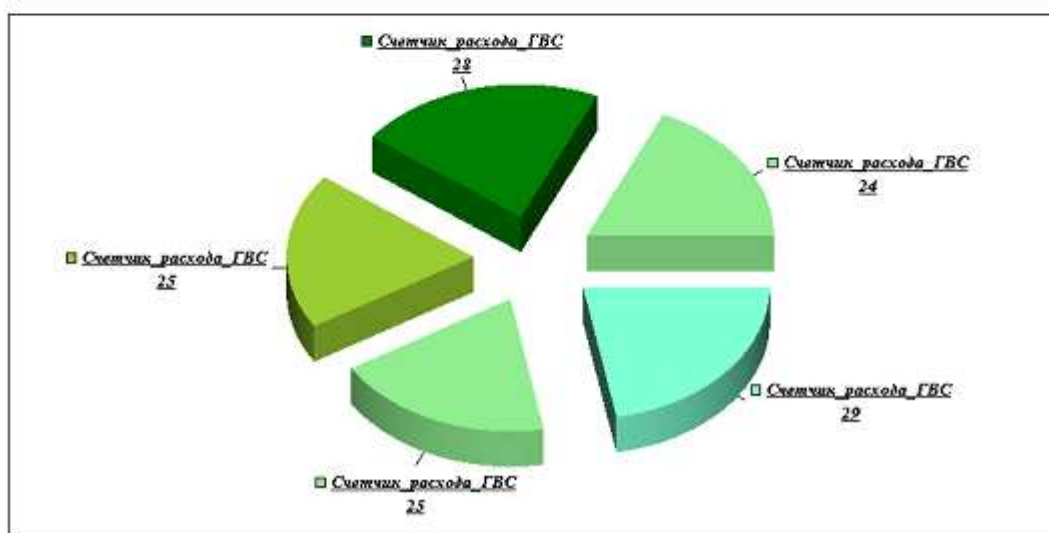
[пример отчета Сервера документирования TRACE MODE 6]

Шаблон отчета подготовлен системой документирования TRACE MODE 6. Шаблон отчета создан в Интегрированной среде разработки, генерирование отчета осуществлено в ДокМРВ+.

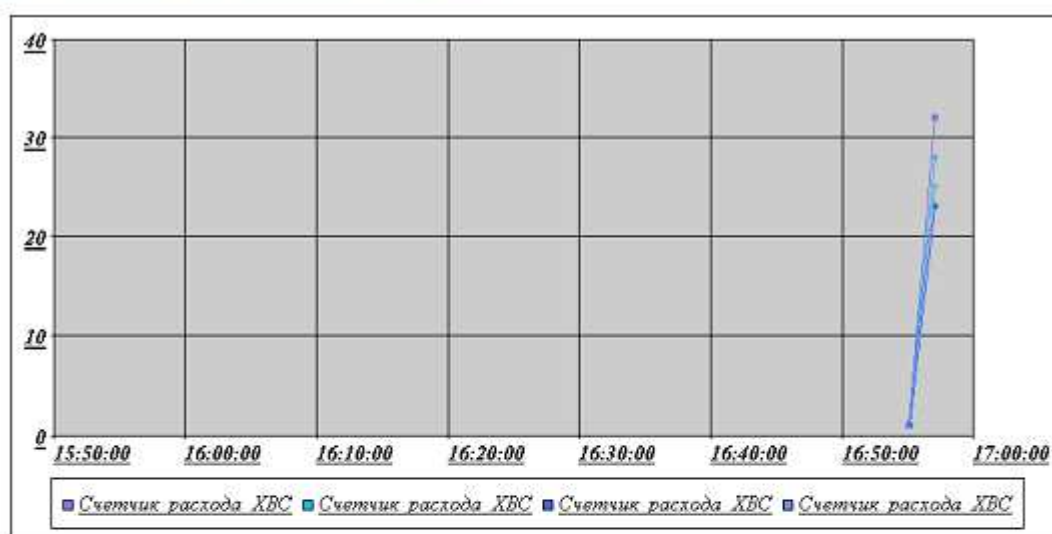
Данная ведомость демонстрирует четыре типа отображения информации в документе: таблица, вертикальные бар-гистограммы, круговые диаграммы и тренд.

Исходный документ может быть распечатан и/или сохранен в файл в формате HTML.

Горячее водоснабжение (м3)



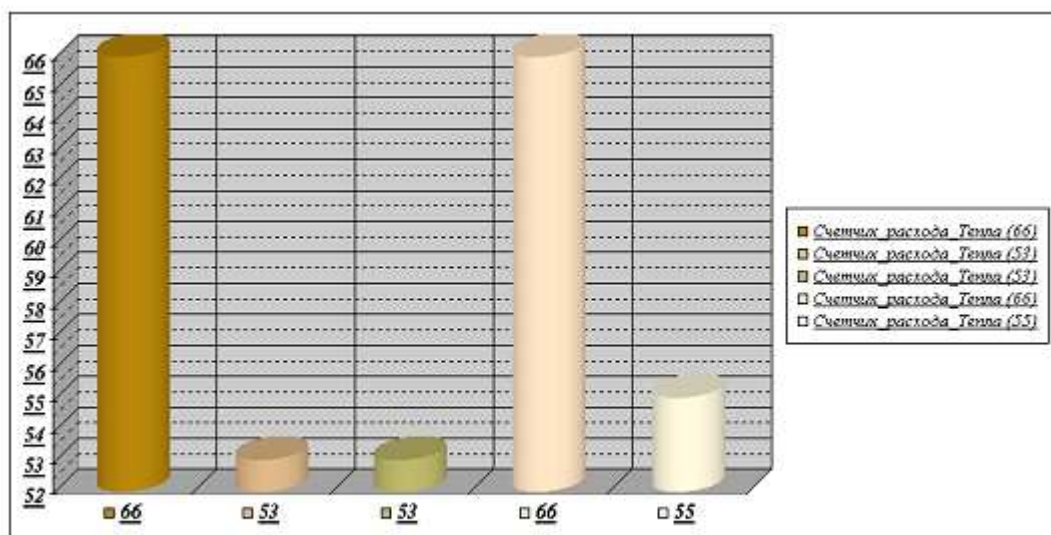
Холодное водоснабжение (м3)



Электрознергия (КВт/час)

№ квартиры	Номер счетчика	Значение (КВт/час)	С учетом погрешности (КВт/час)	Стоимость
1	123378994	20	32	1344
2	234223423	16	25.6	1075.2
3	334445555	21	33.6	1411.2
4	435345344	24	38.4	1612.8
5	343453336	16	25.6	1075.2

Тепло (ГКал)



Сервер документирования Trace Mode 6 (C) AdAstra Research Group, Ltd. 2005

Приложение 2

Арифметические FBD блоки

Название	Обозначение	Действие	Входы и выходы
1	2	3	4
Сложение	$X+Y$	Сложение X и Y	X, Y — слагаемые, O — сумма
Сложение четырех элементов	$+$	Сложение A, B, C, D	A, B, C, D — слагаемые, O — сумма
Вычитание	$X-Y$	Вычитание Y из X	X — уменьшаемое, Y — вычитаемое, O — разность
Умножение	$X*Y$	Умножение X на Y	X, Y — множители, O — произведение
Деление	X/Y	Деление X на Y	X — делимое, Y — делитель, O — частное
Остаток от деления	$X\%Y$	Остаток от деления X на Y	X — делимое, Y — делитель, O — остаток от деления
Возведение в степень	$X**Y$	Возведение X в степень Y	X — основание, Y — показатель степени, O — результат возведения в степень
Абсолютное значение	ABS	Возвращает абсолютное значение	X — вход, O — выход
Инверсия знака	$-X$	Меняет знак входного значения	X — вход, O — выход
Целая часть	$FLOOR$	Возвращает целую	X — исходное число,

		часть числа	О— целая часть
Обратная величина	$1/X$	Вычисляет обратную величину: $O = 1 / X$	X— входное значение, О— обратная величина
Квадратный корень	SQRT	Извлекает квадратный корень из X	X— входное значение, О— квадратный корень из X
Возведение в квадрат	$X**2$	Возведение X в квадрат	X— входное значение, О— квадрат X
Сумма квадратов	HYPOT	Вычисляет сумму квадратов	X, Y— входные значения, О— сумма квадратов
Масштабирование	$K*X+C$	Производит масштабирование входного значения	X— входное значение, K— множитель, C— смещение

FBD блоки сравнения

Название	Обозначение	Описание	Входы и выходы
Равенство	==	Возвращает 1, если IN1 равен IN2, в противном случае возвращает 0	IN1, IN2—сравниваемые значения, EQ—результат сравнения
Неравенств	≠	Возвращает 1, если IN1 не равен IN2, в противном случае возвращает 0	IN1, IN2—сравниваемые значения, NE—результат сравнения
Больше	>	Возвращает 1, если IN1 больше IN2, в противном случае возвращает 0	IN1, IN2—сравниваемые значения, GT—результат сравнения
Меньше	<	Возвращает 1, если IN1 меньше IN2, в противном случае возвращает 0	IN1, IN2—сравниваемые значения, LT—результат сравнения
Больше или равно	>=	Возвращает 1, если IN1 не меньше IN2, в противном случае возвращает 0	IN1, IN2—сравниваемые значения, GE—результат сравнения
Меньше или равно	<=	Возвращает 1, если IN1 не больше IN2, в	IN1, IN2—сравниваемые

		противном случае возвращает 0	значения, LE— результат сравнения
Равенство нулю	$==0$	Возвращает 1, если INP равен 0, в противном случае возвращает 0	INP— сравниваемое значение, $==0$ — результат сравнения
Неравенство нулю	$\neq 0$	Возвращает 1, если INP неравен 0, в противном случае возвращает 0	INP— сравниваемое значение, $\neq 0$ — результат сравнения
Знаковая функция	SIGN	Если $INP > 0$, то $SN+=1$, а $SN-=0$; если $INP < 0$, то $SN+=0$, а $SN-=1$. Если $INP = 0$, то $SN+ =$ $SN- = 0$.	INP— входное значение, $SN+$ и SN — выходные значения
Анализ на равенство	CMP	Сравнивается вход INP с входами PD_k , возвращается наименьший из номеров входов PD_k , чьи значения равны INP. если все значения PD_k не равны INP, то значение Q не меняется.	INP, PD_k — входы, Q— выход
Анализ совпадения	CMPN	Если INP неравен Pd, то при каждом вызове QT увеличивается на 1, если $QT > PT$, то QE	INP, PD, PT— входы, QT, QE— выходы

		возвращает 1, увеличение QT прекращается даже при INP неравном PD. Выходы QT, QE принимает 0 при любом изменении INP.	
Анализ совпадения	CMPE	Отличается от CMPN анализом совпадения INP с PD, сбросом QT и QE в 0 при любом изменении как INP, так и PD	INP, PD, PT— входы, QT, QE— выходы
Управление по астрономическому времени	ALARM	Формирует 1 при совпадении текущего астрономического времени с заданным на входах	MON— месяц; DAY— день месяца; DOF— день недели; H— часы; M—минуты; S— секунды

FBD блоки выбора

Название	Обозначение	Действие	Входы и выходы
1	2	3	4
Выбор из двух	SEL	Выбирает IN0, если IG= 0 и IN1, если IG= 1	IG— критерий выбора, IN0, IN1— значения, SEL— результат выбора
Выбор из пяти	nSEL	Возвращает значение одного из входов и номер входа. Если REG= 0 возвращается минимальное значения, если REG= 1— наибольшее. Первые пять битов числа BLK определяют участие входов в выборе. Если бит равен 0, то соответствующий ему по номеру вход рассматривается, если 1— игнорируется	REG— вход, определяющий условие выбора, BLK— вход, определяющий участие входов I0...I5 в выборе; I0...I5— входы для выбора; VAL— выбранное значение; NUM— номер выбранного канала
Выбор максимального	MAX	Возвращает наибольшее значение из входных	IN1, IN2— входные значения, MAX— максимальное из входных значений
Выбор минимального	MIN	Возвращает наименьшее значение из входных	IN1, IN2— входные значения, MIN— минимальное из

			входных значений
Ограничение	LIMIT	Клиппирует входной сигнал	<p>INP— входной сигнал</p> <p>вход MAX— максимальное значение входного сигнала, вход MIN— минимальное значение входного сигнала</p>
Выбор из трех	MUX	Возвращает IN_{NUM} , если $NUM = 0, 1, 2$ в противном случае выход Q не меняется	<p>Вход NUM— номер входа для выбора;</p> <p>$IN_0 \dots IN_2$— входы для выбора;</p> <p>Q— результат выбора</p>
Выбор из семи	MUX7	Аналогичен MUX, отличается количеством входов IN	<p>Вход NUM— номер входа для выбора;</p> <p>$IN_0 \dots IN_6$ входы для выбора;</p> <p>Q— результат выбора</p>
Интервал	NLIM	<p>Возвращает 1, если $INP > MAX$;</p> <p>возвращает 0, если $MIN \leq INP$;</p> <p>возвращает 2, если $INP < MAX$</p>	<p>Вход MIN— минимальное значение;</p> <p>Вход INP— входное значение;</p> <p>Вход MAX—</p>

			минимальное значение
Запаздывание	LTN	<p>Реализация запаздывания:</p> $Q0_i = INP_i;$ $Q1_i = INP_{i-1};$ $Q2_i = INP_{i-2};$ $Q3_i = INP_{i-3}, \text{ где}$ <p>i- номер текущего вызова блока</p>	<p>INP— вход;</p> <p>$Q0...Q3$— задержанный сигнал INP</p>
Предсказание	FRWD	<p>Реализует экстраполяцию входного значения по первой и второй производным, выходу Q присваивается предполагаемое значение INP при следующем вызове блока</p>	<p>INP— вход;</p> <p>Q— предполагаемое значение, поступающее на вход блока при следующем вызове</p>

FBD блоки-генераторы

Название	Обозначение	Действие	Входы и выходы
1	2	3	4
Меандр	G01	Генерирует прямоугольный сигнал с максимальным значением равным 1	1/0— результат генерации
Бегущая единица	G1	8-битовый выход принимает последовательно значения: $0, 2^0, 2^1, 2^2, \dots, 2^7, 2^6, \dots, 2^0, 2^1$ и т. д.	Q— результат генерации сигнала
Случайная величина в диапазоне [0;1]	RND	Генерирует случайную величину в диапазоне [0;1] с нормальным законом распределения	Q— результат генерации сигнала
Пилообразный сигнал	PILA	Генерируется пилообразный сигнал с максимальным значением PV	PV— максимальное значение; Q— результат генерации сигнала
Единица с заданной вероятностью	GP01	Генерируются 0 и 1 с вероятностью генерации 1	PRB— вероятность генерации 1;

		равной PRB. PRB— целое число в диапазоне от 0 до 1000	0/1— результат генерации
Астрономическое время	TIME	Возвращает текущее астрономическое время	S— секунды; M— минуты; H— часы
Астрономическая дата	DATE	Возвращает текущее значение даты	DAY— день месяца; MON— месяц; YR— текущий год
Период вызова программы	TSTEP	Измеряет период вызова программы в миллисекундах	ПМ— период вызова программы
Синусоидальный сигнал	GSIN	Генерирует синусоидальный сигнал единичной амплитуды с периодом колебаний PT (количество вызовов блока, которые приходятся на один период)	PT— период колебаний; SIN— результат генерации

Список литературы

1. Андреев Е. Б., Куцевич Н. А. Синенко О. В. SCADA-системы: взгляд изнутри/ Е. Б. Андреев, Н. А. Куцевич, О. В. Синенко.— М.: издательство РТСофт, 2004.— 176с;
2. Букреев В. Г., Цхе А. В. 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. Анизимиров Л., Айзин В., Фридлянд А. Новая версия Trace Mode для Windows NT/ Л. Анизимиров, В. Айзин, А. Фридлянд// Современные технологии автоматизации.— 1998.— 3.—с 56;
5. Анизимиров Л. Windows- компоненты Trace Mode 4.20/ Л. Анизимиров// Современные технологии автоматизации.— 1996.— 1.—с 102;
6. Волобуев Ю. АСУ ТП в металлургии: проблемы и решения/ Ю. Волобуев Ю.// Современные технологии автоматизации.— 2000.— 1.—с 38;
7. Кузнецов А. SACADA- системы: программистом можешь ты не быть.../А. Волобуев// Современные технологии автоматизации.— 1996.— 1.—с 32;