ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

РОССИЙСКИЙ
ХИМИКО–ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
им. Д.И. МЕНДЕЛЕЕВА

НОВОМОСКОВСКИЙ ИНСТИТУТ

А. Г. ЛОПАТИН, П. А. КИРЕЕВ

МЕТОДИКА РАЗРАБОТКИ СИСТЕМ УПРАВЛЕНИЯ НА БАЗЕ SCADA СИСТЕМЫ TRACE MODE

Утверждено Редакционно–издательским советом
института в качестве учебно–методического пособия

Новомосковск 2007г.

УДК 681.322:62–52
ББК 32.973:32.965
Л 771

Рецензенты:
к.т.н., руководитель учебного центра Токарев А.Ю.
(AdAstrA Research Group, Ltd)
к.т.н., доцент Сидельников С.И.
(НИ РХТУ им. Д. И. Менделеева)

Лопатин А.Г., Киреев П.А.
Методика разработки систем управления на базе SCADA системы TRACE MODE: Учебно–методическое пособие / РХТУ им. Д. И. Менделеева, Новомосковский ин-т. Новомосковск, 2007.–112 с.
ISBN

Даны основы выбора SCADA систем в зависимости от поставленных задач. Рассмотрена методика программирования систем логического и аналогового управления в режиме эмуляции.

Предназначено для студентов, обучающихся по специальности 220300 "Автоматизация технологических процессов и производств (в химической промышленности)", а так же будет полезна всем лицам занимающиеся разработкой систем управления с использованием SCADA систем.

Табл. 2. Ил. 116. Библиогр.: 7 назв.

УДК 681.322:62–52
ББК 32.973:32.965

ISBN

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Создание высокоэффективных технологических процессов современных производств и управление ими, а так же модернизация действующих, в настоящие время не мыслима без создания систем управления с использованием SCADA систем, поэтому курс Современные технические решения в средствах автоматизации занимает особое место в системе подготовки специалистов по специальности 210200 «Автоматизация производственных процессов и производств». Именно он совместно с дисциплинами Теория автоматического управления и Технические средства автоматизации формирует профессиональное мировоззрение будущих инженеров, и как следствие по своему методическому назначению он должен заложить под последующие курсы их теоретические основы.

Концепция SCADA–систем позволяет достичь высокого уровня автоматизации в решении задач разработки систем управления, сбора, обработки, передачи, хранения и обработки информации.

Построение АСУ ТП на основе любой SCADA–системы резко сокращает набор необходимых знаний в области классического программирования, позволяя концентрировать усилия по освоению знаний в прикладной области.

Необходимая информация по данному курсу представлена в основном в Internet, в большом количестве различных журналов в виде отдельных статей, а также в help описаниях к конкретным SCADA–системам причем не всегда на русском языке. Этим и была вызвана необходимость разработки и написания данного учебно–методического пособия.

ВВЕДЕНИЕ

Развитие отечественных компьютерных автоматизированных систем управления технологическими процессами (АСУ ТП) можно подразделить на три основных этапа.

Первый этап создания АСУ ТП связан с использованием ЭВМ первого поколения, таких как «Урал», «УМ-1», «Минск».

На втором этапе применялись аналоги мэйнфреймов IBM (ЕС ЭВМ), клоны мини - компьютеров фирмы DEC (СМ ЭВМ).

Системы управления на этих этапах имели централизованную, структуру, и в большинстве случаев не обеспечивали достаточного быстродействия и работы в режиме реального времени.

Ситуация изменилась с момента массового распространения IBM PC совместимых компьютеров. Появилась возможность унифицировать ПО для операторских станций и промышленных контроллеров. В результате этой деятельности появились программные пакеты для создания человеко-машинного интерфейса (Human Machine Interface HMI) и программного обеспечения операторских станций АСУТП (Supervisory Control And Data Acquisition SCADA) – контроль управления и контроль сбора данных.

1 СИСТЕМЫ СБОРА ДАННЫХ И ОПЕРАТИВНОГО ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

1.1 Системы мониторинга и управления технологическими процессами

Современные интегрированные системы управления производством (рисунок 1) строятся по принципу пирамиды и охватывают весь цикл работы предприятия от систем управления нижнего уровня до систем управления предприятием в целом.

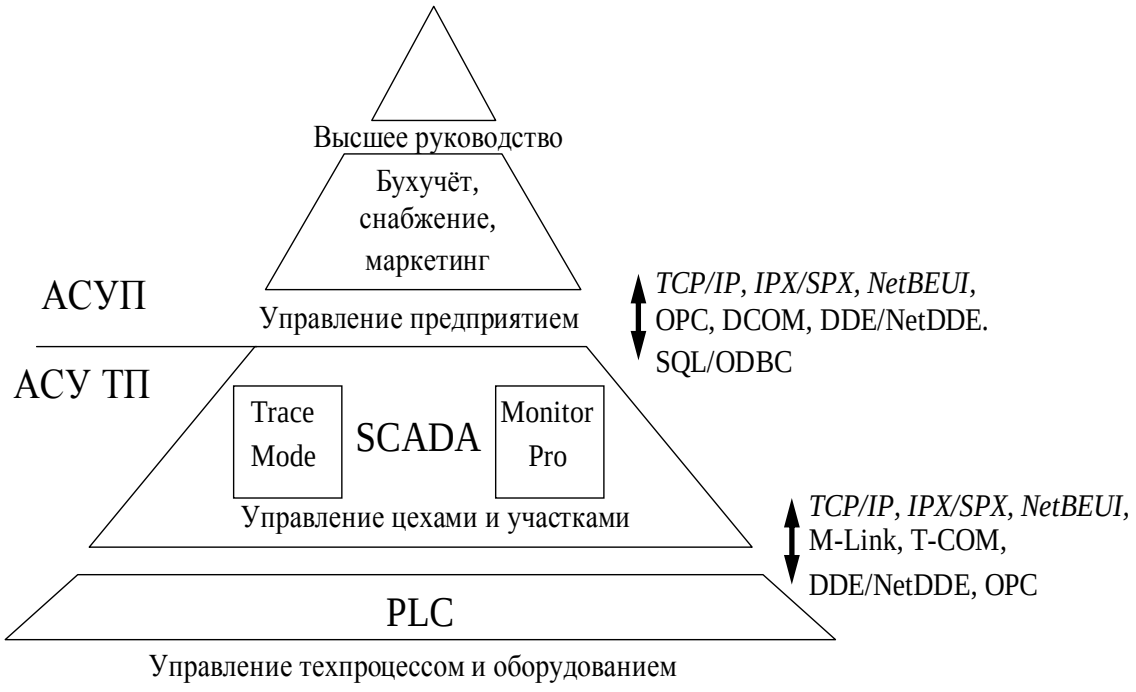


Рисунок 1 – Интегрированные системы управления производством

Архитектура АСУ ТП включает четыре уровня.

На первом уровне (уровне датчиков и исполнительных механизмов) аналоговый интерфейс 4...20 мА (0...5 мА) в настоящее время заменяется коммуникационной технологией, объединяющей датчики, исполнительные механизмы и контроллеры в единую цифровую сеть - Fieldbus (полевая шина или промышленная сеть).

Это позволяет большое число 2-, 3-, 4-проводных линий связи, идущих от множества датчиков и исполнительных механизмов к каналам ввода-вывода контроллеров, заменить на один «малопроводной» кабель. К приборам первого уровня по этому кабелю может передаваться также и электропитание. Все это дает серьезный выигрыш в стоимости. Кроме того, каждое устройство этого уровня оснащается самостоятельным вычислительным ресурсом и может выполнять функции управления, самонастройки и самодиагностики, что упрощает их обслуживание.

На втором уровне находятся устройства связи с объектом (УСО), принимающие и выдающие на объект группу аналоговых и дискретных сигналов и связанные через различные адаптеры с Fieldbus, программируемыми логическими контроллерами (ПЛК, или PLC) и индустриальными компьютерами (ИК). Устройства второго уровня являются безынициативными, работают под управлением ПЛК или ИК и располагаются вблизи с объектом управления. За счет использования этих УСО снижаются затраты на монтаж и кабельную связь.

На третьем уровне находятся промышленные ПЛК (Ремиконт, Modicon и др.) и SoftPLC (микроРС и др.). Для программирования PLC и SoftPLC Международный электротехнический комитет (МЭК, или IEC) принял стандарт IEC-61131/3, который описывает пять языков программирования: язык лестничных диаграмм или релейно-контактных схем (LD - Ladder Diagram); язык инструкций (IL -Instruction List); язык функциональных блоковых схем (FBD - Function Block Diagram); язык последовательных функциональных блоков (SFC - Sequential Function Chart); язык структурированного текста (ST - Structured Text). Связь между контроллерами и станциями управления верхнего уровня осуществляется по сети Ethernet, выполненной в промышленном варианте (Industrial Ethernet).

На рынке промышленной автоматизации все большим спросом пользуются встраиваемые в персональные компьютеры (ПК) модули, позволяющие непосредственно к компьютеру подключать датчики и исполнительные механизмы. Это направление получило название PC-based Control (управление на базе ПК).

Промышленные компьютеры представляют собой, как правило, программно-совместимые с обычными ПК машины, адаптированные для жестких условий эксплуатации - для установки на производстве, в цехах, на газокомпрессорных станциях и т. д. В качестве устройств сопряжения с объектом управления они комплектуются дополнительными платами (адаптерами) расширения. Для объектов управления, имеющих небольшое число входов-выходов, невысокие требования по надежности и по обеспечению

режима реального времени, подход PC-based Control с экономической точки зрения предпочтителен, так как уменьшаются затраты на аппаратные средства.

На четвертом уровне располагаются станции в виде PC совместимых промышленных компьютеров, которые обеспечивают диспетчеризацию технологического процесса и реализуют принцип безщитовой автоматики. Основу ПО этого уровня составляет SCADA–система.

Каждая SCADA-система содержит в своем составе две базовые подсистемы: инструментальный комплекс, представляющий собой средства разработки конкретных проектов; исполнительный комплекс, поддерживающий работу компьютерных пультов в системах управления технологическими процессами. Каждый инструментальный комплекс может обслужить при разработке несколько исполнительных комплексов.

Доминирующей операционной системой для АСУ ТП верхнего уровня является Windows NT. Стандартным механизмом взаимодействия программного обеспечения АСУ ТП признан стандарт OPC (OLE for Process Control), который основан на объектной модели COM/DCOM фирмы Microsoft (США).

Современная АСУ ТП обязательно должна предусматривать также информационную связь с корпоративными системами управления предприятием - уровнем АСУП (в современной терминологии ERP-системы (Enterprise Resource Planning) - планирование ресурсов предприятия или MRP-системы (Manufacturing Resource Planning) - планирование ресурсов производства). Системы ERP ориентированы на предприятие в целом, а MRP - на его технологические подразделения.

В интегрированных системах управления предприятием между системами SCADA и ERP присутствует промежуточная группа систем, называемая MES (Manufacturing Execution Systems). Эта группа возникла вследствие обособления задач, не относящихся к ранее определенным группам SCADA и ERP. К системам MES принято относить приложения, отвечающие:

за управление производственными и людскими ресурсами в рамках технологического процесса;

за планирование и контроль последовательности операций технологического процесса;

за управление качеством продукции;

за хранение исходных материалов и произведенной продукции по технологическим подразделениям;

за техническое обслуживание производственного оборудования;

за связь систем SCADA и ERP.

Одна из причин возникновения таких систем - это отделение в АСУ ТП тактических задач оперативного управления технологическими процессами от стратегических задач ведения процесса в целом. В частности, в химической, металлургической, пищевой, фармацевтической и некоторых других отраслях промышленности можно выделить задачи управления технологическими последовательностями (batch control). Их суть заключается в обеспечении выпуска продукции в нужном объеме с заданными технологическими

характеристиками при наличии возможности перехода на новый вид продукции.

За последние годы резко возрос интерес к проблемам построения высокоэффективных и высоконадежных систем диспетчерского управления и сбора данных – SCADA. С одной стороны, это связано со значительным прогрессом в области вычислительной техники, программного обеспечения и телекоммуникаций, что увеличивает возможности и расширяет сферу применения автоматизированных систем. С другой стороны, развитие информационных технологий, повышение степени автоматизации и перераспределение функций между человеком и аппаратурой обострило проблему взаимодействия человека-оператора с системой управления.

Можно выделить четыре функций человека-оператора в системе диспетчерского управления, в которых оператор:

1 планирует, какие следующие действия необходимо выполнить; обучает (программирует) компьютерную систему на последующие действия;

2 отслеживает результаты полуавтоматической работы системы;

3 вмешивается в процесс либо при наступлении критических событий, когда автоматика не может справиться, либо при необходимости подстройки (регулировки) параметров процесса;

4 обучается в процессе работы (получает опыт).

Основными особенностями процесса управления в диспетчерских системах управления являются следующие:

процесс SCADA применяется в системах, в которых обязательно наличие человека (оператора, диспетчера);

оператор несет, как правило, общую ответственность за управление системой, которая при нормальных условиях только изредка требует подстройки параметров для достижения оптимальной производительности;

активное участие оператора в процессе управления происходит нечасто и в непредсказуемые моменты времени, обычно в случае наступления критических событий (отказы, нештатные ситуации и пр.);

действия оператора в критических ситуациях могут быть жестко ограничены по времени (несколькими минутами или даже секундами).

В таблице 1 приведены некоторые из популярных на западном и российском рынках SCADA-систем, имеющих поддержку в России.

1.2 Этапы создания систем управления на базе SCADA – систем

Процесс создания системы диспетчерского контроля и управления состоит из следующих этапов.

1 Детализация технических требований на создание системы контроля и управления.

2 Разработка проектно-сметной документации (в полном или сокращенном объеме).

3 Сбор исходных данных.

4 Составление полного перечня переменных.

5 Комплектация системы.

Таблица 1 – Рынок SCADA–систем

SCADA	Фирма-изготовитель	Страна
InTouch	Wonderware	США
iFIX	Intellution	США
Factory Link	United States DATA Co.	США
Genesis	Iconics	США
RealFlex	BJ Software Systems	США
RSView	Rockwell Software Inc.	США
Simplicity	GE Fanuc Automation	США
Monitor Pro	Schneider Electric	Франция
Vijeo Look	Schneider Electric	Франция
WinCC	Siemens	Германия
Sitex	Jade Software	Великобритания
Citect	CiTechnologies	Австралия
Трейс Моуд	AdAstra	Россия, г.Москва
Круг-2000	НПФ «Круг»	Россия, г.Пенза
MasterSCADA	НПФ «ИнСАТ»	Россия, г.Москва
MIK$Sys	МИФИ	Россия, г.Москва

6 Разбиение объекта управления на технологические участки; компоновка переменных по участкам и группам.

7 Заполнение (генерация) базы данных.

8 «Рисование» статических частей мнемосхем.

9 Заполнение мнемосхем динамическими элементами.

10 Составление схемы переходов между мнемосхемами.

11 Генерация печатных документов.

12 Верификация базы данных.

13 Разработка эксплуатационной документации.

14 Тестирование системы в автономном режиме (без УСО).

15 Монтаж.

16 Тестирование системы в рабочем режиме (с УСО).

17 Внедрение, в том числе пусконаладка и обучение персонала. Возможно распараллеливание некоторых видов работ, что обеспечивает существенное сокращение срока создания системы.

На этапах 1, 2 составляются подробные технические требования (техническое задание) на систему контроля и управления, которые согласуются с конечным пользователем - теми, кто будет непосредственно эксплуатировать систему, например технологами. Для конкретизации технических требований необходимо выполнить следующее:

– описать конкретную структуру технических средств, используемых в проектируемой системе управления;

– указать конкретную информационную мощность системы (перечень измеряемых и управляющих переменных);

– привести перечень расчетных переменных, формулы расчета;

– привести конкретный перечень печатных документов;

– конкретизировать перечень требований к конечному пользователю системы.

При конкретизации технических характеристик системы, таких, как период опроса, время обновления информации на экране (и т.д.), необходимо учитывать, что их числовые значения зависят от следующих факторов:

– числа устройств связи с объектом;

– типов УСО и скорости их обмена с ПК;

– объема базы данных (информационная мощность);

– числа расчетных переменных;

– модели ПК (тип процессора, тактовая частота, объем кэш-памяти и т. п.).

Этап 3 – этап сбора данных - очень ответственный этап, так как от качества его выполнения в большой степени зависит срок и качество выполнения всей работы. Исходными данными при создании системы является следующая информация и документация:

1 функциональные схемы КИП и А;

2 описанием технологии;

3 перечень контролируемых и регулируемых параметров;

4 фотографии, рисунки, чертежи основных технологических агрегатов, которые помогают лучше и понятнее нарисовать мнемосхемы;

5 заполненные образцы отчетных документов (режимных листов, суточных ведомостей и т. п.).

Существуют три основных подхода к разработке системы контроля и управления:

– от графики;

– от структуры системы управления (аппаратуры);

– от структуры технологического объекта (технологии).

Подход от графики – это простейший подход к разработке. Он предполагает первичным создание пользовательского интерфейса оператора. Такой подход характерен для большинства западных пакетов, ориентирован на малоопытного разработчика и оправдывает себя при создании малых систем управления, состоящих из одного компьютера и стандартных ПЛК или модулей УСО. Здесь предполагается работа с небольшим числом сигналов, когда нет необходимости структурировать базу переменных проекта.

Подход от структуры системы управления требует большей квалификации от разработчика. Здесь в качестве базовой информации, от которой ведется разработка, является аппаратный слой проекта. Поэтому сначала описываются ПК и ПЛК, входящие в систему, и их коммуникации. Затем для каждого из описанных устройств указываются исполнительные модули, которые будут на нем работать. После этого разрабатываются фрагменты программного обеспечения проекта, которые будут запускаться на указанных исполнительных модулях.

Подход от структуры технологического объекта – это наиболее продвинутый подход к созданию систем контроля и управления. Он интегрирует в себе оба предыдущих подхода и добавляет анализ

автоматизируемого объекта для построения технологической иерархии (технологический слой проекта).

После описания технологической иерархии объекта управления для каждого ее элемента можно описать относящиеся к объекту элементы системы управления. Это могут быть сигналы, расчетные параметры, регламенты обслуживания, потребляемые ресурсы и производимая продукция, обслуживающий персонал, графическое представление, программы управления и прочая необходимая информация.

Далее на основании элементов проекта, привязанных к технологической структуре объекта, можно скомпоновать систему управления. Для этого следует сначала описать ее аппаратную и программную конфигурацию: входящие в систему ПК и ПЛК, запускаемые на них исполнительные модули. После этого надо поставить в соответствие аппаратным компонентам системы фрагменты технологической структуры. При этом выполняется автопостроение различных компонентов системы управления.

После описания технологического слоя проекта можно менять аппаратную или программную конфигурацию системы. Это приводит к незначительным затратам по доработке проекта, что особенно важно для системных интеграторов, автоматизирующих типовые технологические объекты на разнородном оборудовании.

SCADA-системы «закрывают» цеховой уровень автоматизации, связанный прежде всего с получением и визуализацией информации от ПЛК, распределенных систем управления.

Одновременно с этим SCADA-системы должны обеспечивать надежность и эффективность решения поставленных задач.

Надежность SCADA-систем – не только отсутствие ошибок в программном коде самого продукта, но и его устойчивость к ошибкам во внешних компонентах к некорректным действиям обслуживающего персонала.

Эффективность SCADA-систем сводится к тому, насколько мощный компьютер потребуется для реализации с его помощью конкретных проектов, т.е. в составе реальных проектов одновременно используется множество функций SCADA-систем; желательно, чтобы каждая из ее подсистем (графическая, ввод/вывод, архивирование) обладала необходимой функциональностью и быстродействием.

1.3 Функциональные характеристики SCADA – систем

Основными областями применения систем диспетчерского контроля и управления являются:

– производство, управление передачей и распределением электроэнергии;

– промышленное производство;

– водозабор, водоочистка и водораспределение;

– добыча, транспортировка и распределение нефти и газа;

– управление космическими объектами;

– управление на транспорте (все виды транспорта: авиа, метро, железнодорожный, автомобильный, водный);

– телекоммуникации;

– военная область.

1.3.1 Функциональные возможности

В силу тех требований, которые предъявляются к SCADA-системам, спектр их функциональных возможностей определен и реализован практически во всех пакетах.

Перечислим основные возможности и средства, присущие всем системам и различающиеся только техническими особенностями реализации:

– органы управления различных типов;

– представление текущей и исторической информации на дисплее (реализация динамизированных мнемосхем, гистограмм, анимационных изображений, таблиц, графиков, трендов, выделение аварийных ситуаций и т.д.);

– возможность создания архивов аварий, событий и поведения, переменных процесса во времени, а также полное или выборочное сохранение параметров процесса через заданный промежуток времени, постоянно или по условию и ее дальнейшая обработка;

– печать отчетов и протоколов произвольной формы в заданные моменты времени или по запросу оператора; вывод на экран, на внешние устройства, а так же регистрация аварийных ситуаций в моменты их возникновения;

– средства документирования, как самого алгоритма, так и технологического процесса;

– ввод и передача команд и сообщений оператора в контроллеры и другие устройства системы;

– взаимосвязь прикладных программ пользователя с текущей измеряемой информацией и управленческими решениями;

– информационные связи с серверами и другими операторскими станциями через разные сетевые структуры.

– упрощенный язык для реализации алгоритмов управления математических и логических вычислений;

– ядро или монитор реального времени, который обеспечивает детерминизм поведения системы или иными словами предсказуемое время отклика на внешние события;

– драйверы к оборудованию;

– сетевые функции;

– средства защиты от несанкционированного доступа в систему;

– многооконный графический интерфейс, импорт изображений и создание собственной библиотеки алгоритмов, динамических объектов и элементов мнемосхем.

– сбор текущей технологической информации от контроллеров или других приборов и устройств, связанных непосредственно или через сеть с пультом оператора;

– вычислительная и логическая обработка технологических данных, в том числе первичная обработка измерительной информации;

– решение прикладных программ пользователя и их взаимосвязь с текущей измеряемой информацией и управленческими решениями;

– информационные связи с серверами и другими рабочими станциями через различные сетевые структуры.

В целом технология проектирования систем автоматизации на основе различных SCADA-систем очень схожа и состоит в следующем:

– разработка архитектуры системы автоматизации в целом; на этом этапе определяется функциональное назначение каждого узла системы автоматизации;

– решение вопросов, связанных с возможной поддержкой распределенной архитектуры, необходимостью введения узлов с «горячим резервированием» и т. п.;

– создание прикладной системы управления для каждого узла; на этом этапе специалист в области автоматизируемых процессов наполняет узлы архитектуры алгоритмами, совокупность которых позволяет решать задачи автоматизации;

– приведение параметров прикладной системы в соответствие с информацией, которой обмениваются устройства нижнего уровня, например ПЛК с внешним миром (датчики температуры, давления, исполнительные устройства и др.);

– отладка созданной прикладной программы в режиме эмуляции и в реальном режиме.

Функциональные возможности систем SCADA в значительной мере определяют стоимость и сроки создания ПО, а также сроки ее окупаемости.

1.3.2 Программно-аппаратные платформы SCADA – систем

Анализ перечня таких платформ необходим, поскольку от него зависит ответ на вопросы распространения SCADA-системы на имеющиеся вычислительные средства, а также оценка стоимости эксплуатации системы. В различных SCADA-системах этот вопрос решен по-разному.

Подавляющее большинство SCADA-систем реализовано на MS Windows платформах. Именно такие системы предлагают наиболее полные и легко наращиваемые HMI средства. Даже разработчики многоплатформенных SCADA-систем приоритетным считают дальнейшее развитие своих SCADA-систем на платформе Windows. Быстрое развитие OPC-технологий, низкие цены аппаратного обеспечения, распространенность Windows в офисных системах предприятий вместе с ее солидными техническими характеристиками - главные причины того, что абсолютное большинство производителей SCADA-пакетов мигрировали в сторону этой операционной системы. Применение ОС реального времени становится все более очевидным в основном во встраиваемых системах.

1.3.3 Средства сетевой поддержки

Одной из основных черт современных систем автоматизации является их высокая степень интеграции. В любой из них могут быть задействованы

объекты управления, исполнительные механизмы, аппаратура, регистрирующая и обрабатывающая информацию, рабочие места операторов, серверы баз данных и т. д. Очевидно, что для эффективного функционирования в этой разнородной среде SCADA-система должна обеспечивать высокий уровень сетевого сервиса. Желательно, чтобы она поддерживала работу в стандартных сетевых средах (Arcnet, Ethernet и т. п.) с использованием стандартных протоколов (NETBIOS, TCP/IP и др.), а также обеспечивала поддержку наиболее популярных сетевых стандартов из класса промышленных интерфейсов (Profibus, Can-bus, Lon, Modbus Plus и т. д.). Обобщенная схема подобной системы приведена на рисунке 2. Этим требованиям в той или иной степени удовлетворяют практически все SCADA-системы, с тем лишь различием, что набор поддерживаемых сетевых интерфейсов разный.

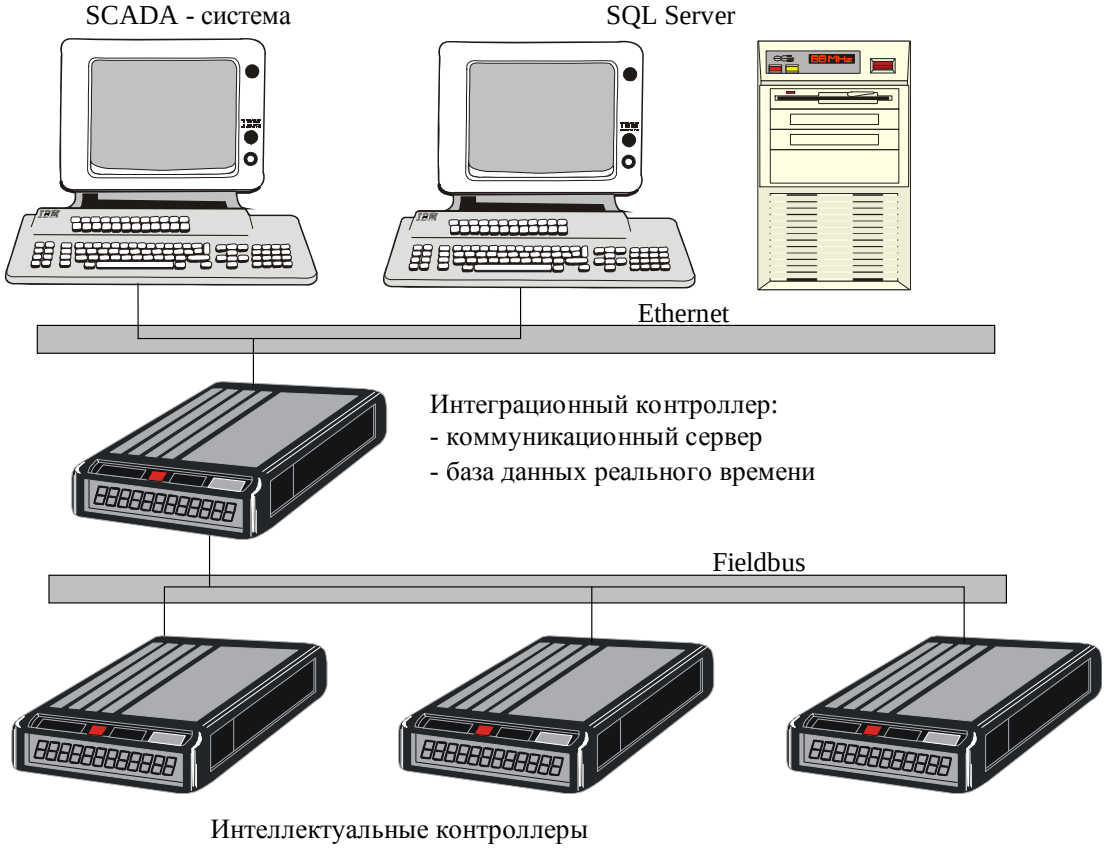


Рисунок 2 – Сетевое взаимодействие SCADA-системы

1.3.4 Встроенные командные языки

Встроенные технологические языки программирования - это инструмент, который предназначен для решения новых задач на базе системы контроля и управления технологическими процессами. Такими задачами являются:

– программно-логическое управление технологическим оборудованием;

– алгоритмы оптимального управления;

– расчет косвенных переменных по формулам;

– визуализация значений в цифровом виде трендов целевой обработки (текущие, средние или суммарные значения параметров по часам, сменам и суткам);

– формирование трендов целевой обработки из программы пользователя

постфактум;

– архивирование дат и времени событий;

– создание сценария динамики экрана;

– интегрирование мгновенных расходов под задачи дозирования;

– создание альтернативных фильтров входных переменных.

Использование стандартных алгоритмов значительно упрощает создание программ. Большинство SCADA-систем имеют встроенные языки высокого уровня. В современных версиях SCADA-систем функциональные возможности языков становятся существенно богаче. Явно выделяются два подхода:

1 ориентация встроенных языков программирования на технологов; функции в таких языках являются высокоуровневыми, не требующими профессиональных навыков программирования при их использовании;

2 ориентация на программиста-системного интегратора.

В каждом языке допускается расширение набора функций. В языках, ориентированных на технологов, это расширение достигается с помощью дополнительных инструментальных средств (Toolkits). Разработка дополнительных функций выполняется обычно программистами-профессионалами.

Разработка новых функций при втором подходе осуществляется обычно разработчиками приложений.

Полнота использования возможностей встроенных языков требует соответствующего уровня квалификации разработчика, если, конечно, в этом есть необходимость. Требования задачи могут быть не столь высокими, чтобы применять всю «мощь» встроенного языка.

Во всех языках функции разделяются на группы, часть из которых присутствует практически во всех языках: математические функции, функции работы со строками, обмен по SQL, и т. д.

1.3.5 Поддерживаемые базы данных

Практически все SCADA-системы используют синтаксис ANSII SQL, который является независимым от типа базы данных. При этом приложения виртуально изолированы, что позволяет менять базу данных без серьезного изменения самой прикладной задачи, создавать независимые программы для анализа информации, использовать уже наработанное программное обеспечение, ориентированное на обработку данных.

Однако для SCADA-систем требуются СУБД реального времени, так как скорость записи в реляционные БД недостаточна (SQL-server обрабатывает до 2000 переменных/сек, iHistorian - до 20 000/сек). Trace Mode использует свою БД, обеспечивающую запись до 600 000 переменных/сек.

1.3.6 Графические возможности

Средства визуализации – одно из базовых свойств SCADA-систем.

Для специалиста-разработчика системы автоматизации, так же, как и для специалиста-технолога, чье рабочее место создается, очень важен графический пользовательский интерфейс (GUI – Graphic Users Interface). В каждой SCADA–

системе существует графический объектно-ориентированный редактор с определенным набором анимационных функций. Используемая векторная графика дает возможность осуществлять широкий круг операций над выбранным объектом, а также быстро обновлять изображение на экране, используя средства анимации. Объекты могут быть простыми (линии, прямоугольники, текстовые объекты и т. д.) и сложными. Все SCADA-системы включают библиотеки стандартных графических символов, библиотеки сложных графических объектов, имеют целый ряд других стандартных возможностей.

Графический редактор позволяет создавать статическую часть технологических мнемосхем, их фрагменты и элементы, не изменяющиеся в процессе работы системы, и далее выполнять динамизацию мнемосхем, т. е. отображать такие атрибуты, как текущие значения параметров, границы сигнализации, состояния исполнительных механизмов и т. д. Динамически изменяемая информация на экране может представляться в удобном для пользователя виде: текстовое сообщение, числовые значения параметров в цифровой форме, или изображение вторичного прибора, состояние оборудования – в виде текста или изменяющих свой цвет и внешний вид фрагментов; состояние технологического процесса – в виде строк подсказок или изменяющихся по форме или цвету частей технологического оборудования и т. д.

1.3.7 Тренды и архивы в SCADA – системах

Тренд – это массив точек переменных, каждая из которых записывается в реальной системе в память ИК через определенный интервал времени.

Графическое представление значений технологических параметров во времени способствует лучшему пониманию динамики технологического процесса на предприятии. Поэтому подсистема создания трендов и хранения информации о параметрах с целью ее дальнейшего анализа и использования для управления является неотъемлемой частью любой SCADA-системы.

Тренды реального времени (Real Time) отображают динамические изменения параметра в текущем времени.

Исторические (архивные) тренды не являются динамическими Они обеспечивают «снимок» состояния данных за прошедшее время, т. е. по архивным данным. Тренды становятся историческими (Historical) после того, как данные будут записаны на диск, и можно будет использовать режим прокрутки предыдущих значений назад с целью посмотреть прошлые значения. Отображаемые данные тренда в таком режиме являются неподвижными и будут отображаться только за определенный период. Различают часовые, сменные и суточные тренды, которые используются для печати сменных документов.

1.3.8 Алармы и события в SCADA – системах

Состояние тревоги, в дальнейшем аларм (Alarm), - это некоторое сообщение, предупреждающее оператора, следящего за технологическим

процессом, о возникновении определенной ситуации, которая может привести к серьезным последствиям и потому требующая его внимания, а часто и вмешательства

Чтобы снять сомнения, принял ли оператор сообщение об аларме, в системах управления принято различать неподтвержденные и подтвержденные алармы. Аларм называется подтвержденным после того, как оператор отреагировал на сообщение об аларме. До этого аларм остается в состоянии неподтвержденного.

Наряду с алармами в SCADA-системах существует понятие событий. События представляют собой обычные статусные сообщения системы и не требуют реакции оператора. Обычно событие генерируется при возникновении в системе определенных условий (типа регистрации оператора в системе).

От эффективности подсистемы алармов зависит скорость идентификации неисправности, возникшей в системе, или технологического параметра, вышедшего за установленные регламентом границы. Быстродействие и надежность этой подсистемы могут существенно сократить время простоя технологического оборудования. Например, если оператор не получит вовремя информацию о том, что двигатель насоса перегрелся, это может привести в лучшем случае к выходу насоса из строя, а то и к крупной аварии.

Причины, вызывающие состояние аларма, могут быть самыми разными. Неисправность может возникнуть в самой SCADA-системе, в контроллерах, каналах связи, в технологическом оборудовании; может выйти из строя датчик или нарушатся его метрологические характеристики; параметры технологического процесса могут выйти за границы, установленные регламентом и т. д.

Подсистема алармов - это обязательный компонент любой SCADA-системы, но возможности подсистем алармов различных SCADA-систем разные. Однако когда речь идет о типах алармов, то все SCADA-системы поддерживают дискретные и аналоговые типы алармов.

Дискретные алармы срабатывают при изменении состояния дискретной переменной. При этом для срабатывания аларма можно использовать любое из двух состояний: TRUE / ON (1) или FALSE / OFF (0), По умолчанию дискретный аларм может срабатывать на ON или OFF в зависимости от конкретной SCADA-системы.

Аналоговые алармы базируются на анализе выхода значений переменной за указанные верхние и нижние пределы. Аналоговые алармы могут быть заданы в нескольких комбинациях:

– High и High High (верхний и выше верхнего);

– Low и Low Low (нижний и ниже нижнего);

– Deviation (отклонение от нормы);

– Rate of Change - ROC (скорость изменения).

Алармы типа ROC срабатывают, когда скорость изменения параметра становится больше предельно допустимой. Понятие «зона нечувствительности» (Deadband) к алармам этого типа не применяется.

SCADA-системы поддерживают возможность отображения, регистрации и

печати информации как об алармах, связанных с аналоговыми или логическими переменными, так и о системных событиях.

События также делятся в зависимости от их характеристик на несколько общих категорий (Event Types). Типы событий одинаковы как для стандартной, так и для распределенной систем алармов (таблица 2).

Таблица 2 – Типы событий

Тип	Событие
ACK	Аларм был подтвержден
ALM	Возникла аварийная ситуация
EVT	Возникло аварийное событие
RTN	Переменная перешла из аварийного состояния в обычное
SYS	Возникло системное событие
USER	Изменение значения переменной оператором
DDE	Получено значение переменной от DDE-клиента
LGC	Скрипт изменил значение переменной
OPR	Оператор ввел новое значение переменной

1.4 Эксплуатационные характеристики SCADA – систем

Эксплуатационные характеристики SCADA-системы имеют большое значение, поскольку от них зависит скорость освоения продукта и разработки прикладных систем. Они, в конечном итоге, отражаются на стоимости реализации проектов.

1.4.1 Надежность

Все успешно работающие системы обеспечивают контроль и управление, включая графический интерфейс оператора, обработку сигналов тревог, построение графиков, отчетов и обмен данными. Эти возможности способствуют улучшению эффективности работы предприятия и, следовательно, увеличению прибыли. Однако при разработке таких систем часто упускается из вида один существенный аспект - что произойдет, если какой-либо элемент аппаратуры выйдет из строя?

Локальная (рисунок 3) и распределенная (рисунок 4) АСУ ТП имеют одну общую особенность. Обе системы полностью выйдут из строя, если всего в одном компоненте системы (компьютере, соединенном с контроллерами или сетью контроллеров) возникнет неисправность.

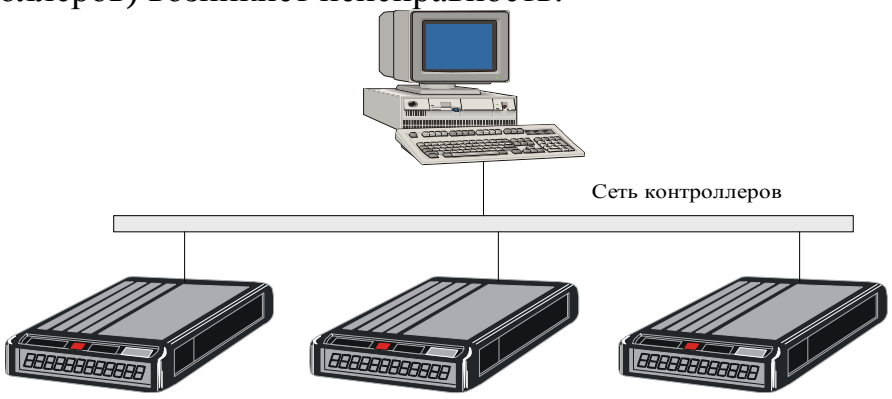


Рисунок 3 – Локальная АСУ ТП

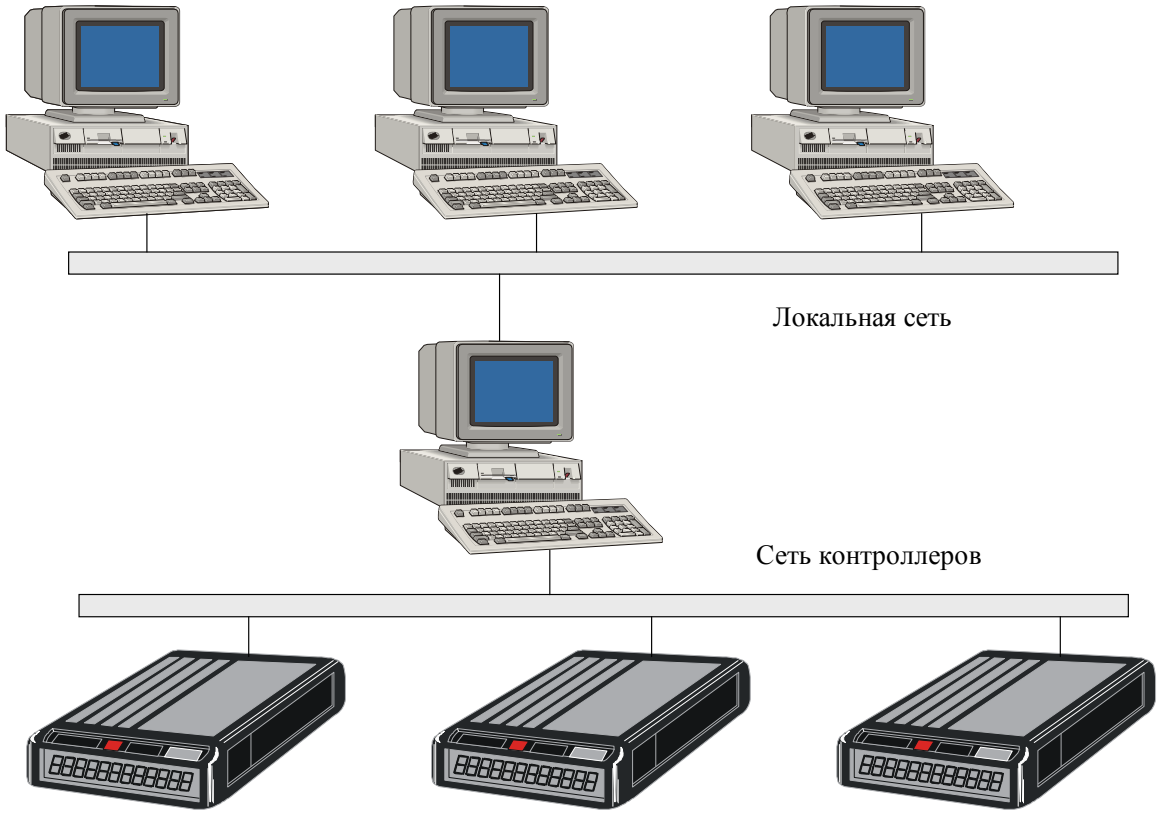


Рисунок 4 – Распределенная АСУ ТП

Большинство современных компьютеров обеспечивают хорошие показатели надежности, но, тем не менее, они также выходят из строя, особенно при эксплуатации в жестких производственных условиях. Если какие-либо компоненты производственного процесса (или весь процесс) являются критически важными, или стоимость остановки производства очень высока, возникает необходимость построения резервируемых систем. В системах, обеспечивающих резервирование, выход из строя одного компонента не влечет за собой остановку всей системы.

SCADA-системы поддерживают реализацию резервирования большинства компонентов как вследствие особенности архитектуры, так и из-за наличия встроенных механизмов автоматического резервирования (Citect, iFIX, Trace Mode).

Распределение процессов управления и контроля по нескольким компьютерам, объединенным в локальную сеть, позволяет увеличить эффективность и скорость работы всей системы. В простой системе компьютер, соединенный с промышленным оборудованием, становится сервером, предназначенным для взаимодействия с контроллерами, а компьютеры в локальной сети - с клиентами (рисунок 5).

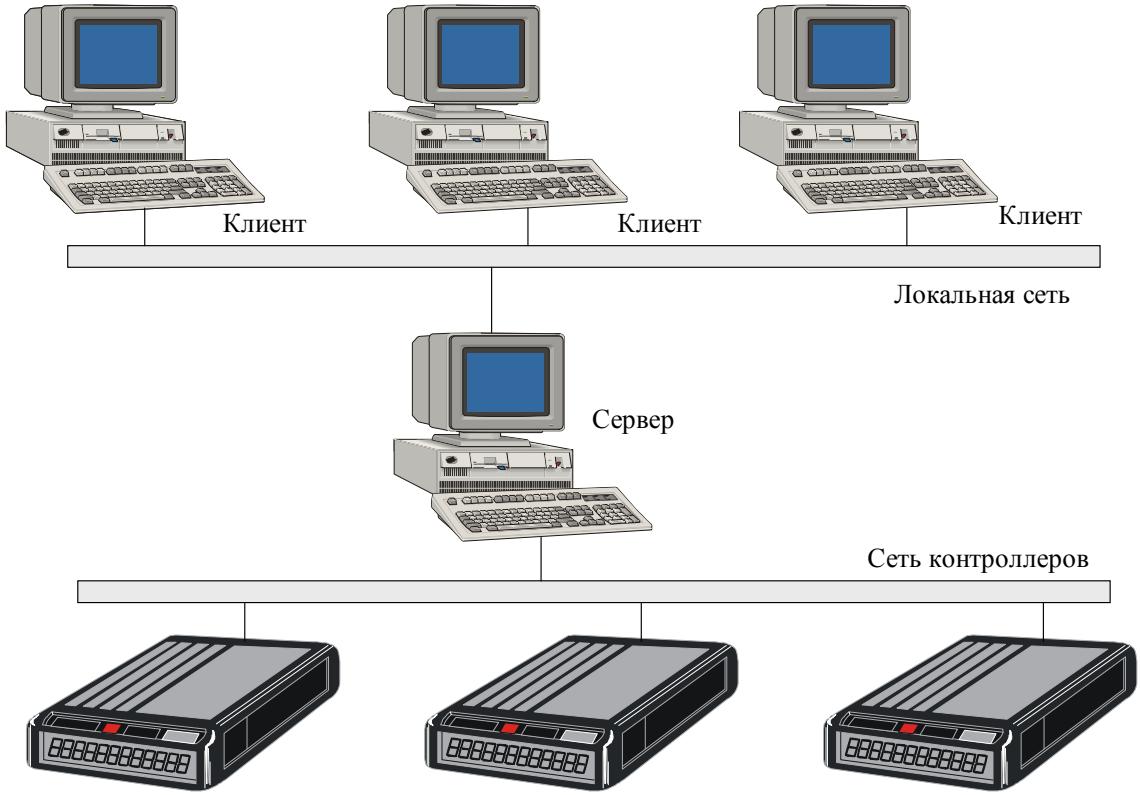


Рисунок 5 – Клиент-серверная архитектура простой системы

Когда клиенту требуются данные для отображения, он запрашивает их у сервера и затем обрабатывает локально. Для обеспечения резервирования в систему может быть добавлен второй (резервный) сервер (рисунок 6), также

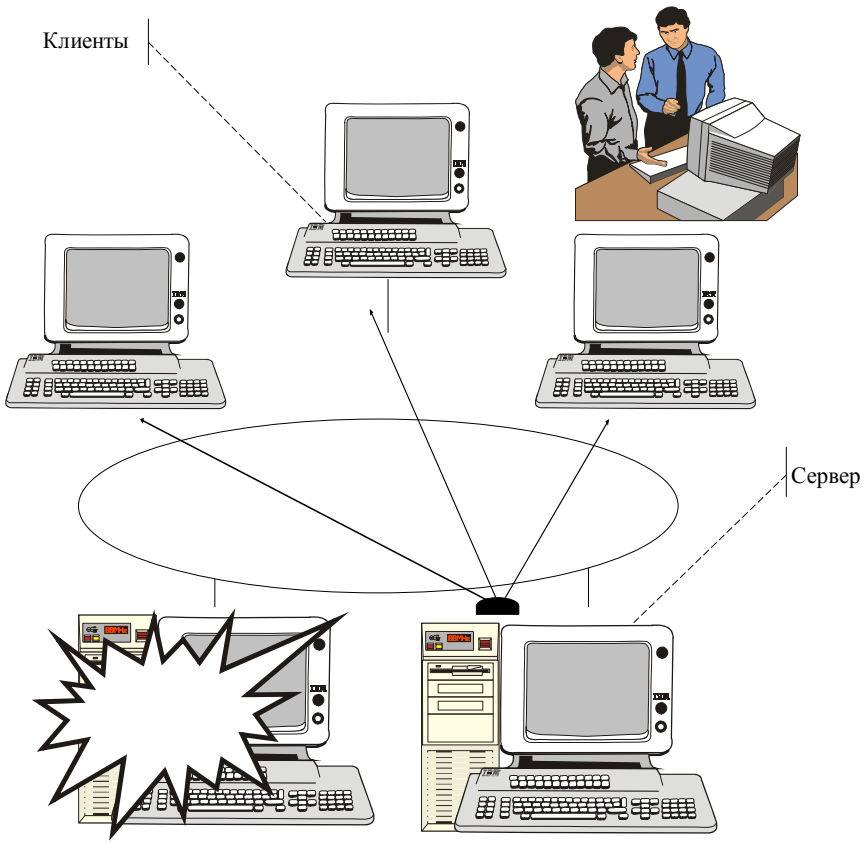


Рисунок 6 – Встроенный механизм автоматического резервирования

предназначенный для взаимодействия с промышленным оборудованием. Если основной сервер выходит из строя, запросы клиентов автоматически направляются к резервному серверу. Резервный сервер не должен при этом полностью дублировать работу основного, поскольку в этом случае оба сервера взаимодействуют с контроллерами, удваивая нагрузку на промышленную сеть и, следовательно, сокращая общую производительность.

В клиент-серверной архитектуре при наличии дублированных серверов ввода-вывода можно легко реализовать поддержку постоянной связи с промышленными устройствами, а также обеспечить сохранность и непрерывность данных тревог и графиков в случае возникновения неисправности.

1.4.2 Наличие и качество технической поддержки

Необходимо обращать внимание не только на наличие технической поддержки SCADA-систем как таковой, но и на ее качество. Основными показателями которой являются:

– наличие и состав, полнота документации на русском языке;

– наличие русифицированной версии SCADA-программы, полнота и уровень русификации;

– полнота и ясность представляемой документации;

– особенности сопровождения SCADA-программы;

– место, сроки и особенности обучения проектировщиков работе со SCADA-программой;

– ознакомление с новыми версиями SCADA-программы и политика их распространения;

– наличие и число фирм на территории России, осуществляющих распространение, поддержку и обслуживание SCADA-программы;

– наличие и число фирм, создающих дополнительное программное обеспечение для конкретных SCADA-программ.

1.4.3 Оценка стоимости инструментальных систем

Стоимость SCADA-систем, на первый взгляд, кажется достаточно высокой. При оценке стоимости SCADA-системы учитываются минимальные и рекомендуемые ресурсы компьютера, необходимые для ее установки. При этом в некоторых системах, число допустимых переменных напрямую зависит от числа доступных ОЗУ, Квалифицированное использование SCADA-пакета позволяет уменьшить требуемое число переменных процесса, которые влияют на стоимость.

Процедура освоения SCADA-систем достаточно проста с точки зрения программиста и не требует длительного времени, поэтому эти затраты относительно невелики. Основной составляющей стоимости является оплата труда программистов, осуществляющих эту работу.

Чтобы оценить время окупаемости SCADA-системы, необходимо учесть множество факторов, включая число проектов, реализуемых на основе этой системы, стоимость этих проектов и т. д. Ориентировочно, если речь идет о

бизнесе системного интегратора, реализация двух-трех проектов при приобретении системы разработки SCADA окупает ее (например, окупаемость проекта на Trace Mode составляет 0,5-3 месяца).

1.4.4 Открытость систем

Описанные выше характерные черты и особенности SCADA-систем являются достаточно устоявшимися. Но немаловажное значение имеют вновь появляющиеся особенности систем, связанные с их «открытостью», с интеграцией их в структуру комплексной автоматизации предприятия в целом. Свойство открытости всегда было одним из важных свойств SCADA, но сейчас оно дополняется новыми средствами передачи данных между процессами (OLE - Object Linking and Embedding - включение и встраивание объектов), стандартом общения с технологическими устройствами (OPC - OLE for Process Control), встраиваемыми программными объектами (ActiveX).

Система является открытой, если для нее определены и описаны используемые форматы данных и процедурный интерфейс, что позволяет подключить к ней «внешние», независимо разработанные компоненты, адаптировать пакет под конкретные нужды с минимальными затратами. В принципе любая SCADA – система является «открытым».

Современные SCADA – системы не ограничивают выбора аппаратуры нижнего уровня, так как предоставляют большой набор драйверов или серверов ввода-вывода и имеют хорошо развитые средства создания собственных программных модулей или драйверов новых устройств нижнего уровня.

Для подсоединения драйверов ввода-вывода к SCADA – системе используются следующие механизмы:

динамический обмен данными (DDE - Dynamic Data Exchange);

собственные протоколы фирм-производителей SCADA – систем, реально обеспечивающие самый скоростной обмен данными;

OPC - протокол, который, с одной стороны, является стандартным и поддерживается большинством SCADA – систем.

1.4.4.1 Технологии OPC

Взамен DDE компания Microsoft предложила более эффективное и надежное средство передачи данных между процессами -OLE (Object Linking and Embedding - включение и встраивание объектов). На базе OLE появляется новый стандарт OPC (OLE for Process Control), ориентированный на рынок промышленной автоматизации. Новый стандарт, во-первых, позволяет объединять на уровне объектов различные системы управления и контроля, функционирующие в распределенной гетерогенной среде; во-вторых, OPC устраняет необходимость использования различного нестандартного оборудования и соответствующих коммуникационных программных драйверов.

Основная цель стандарта OPC заключается в определении механизма доступа к данным с любого устройства из приложений. OPC позволяет производителям оборудования поставлять программные компоненты, которые

стандартным способом обеспечат клиентов данными с ПЛК.

С точки зрения SCADA-систем появление OPC-серверов означает разработку программных стандартов обмена с технологическими устройствами. OPC-интерфейс допускает различные варианты обмена: получение «сырых» данных с физических устройств, из распределенной системы управления или из любого приложения (рисунок 7).

При широком распространении OPC-стандарта появятся следующие преимущества:

OPC позволят определять на уровне объектов различные системы управления и контроля, работающие в распределенной гетерогенной среде;

OPC устранят необходимость использования различного нестандартного оборудования и соответствующих коммуникационных программных драйверов;

У потребителя появится больший выбор при разработке приложений.

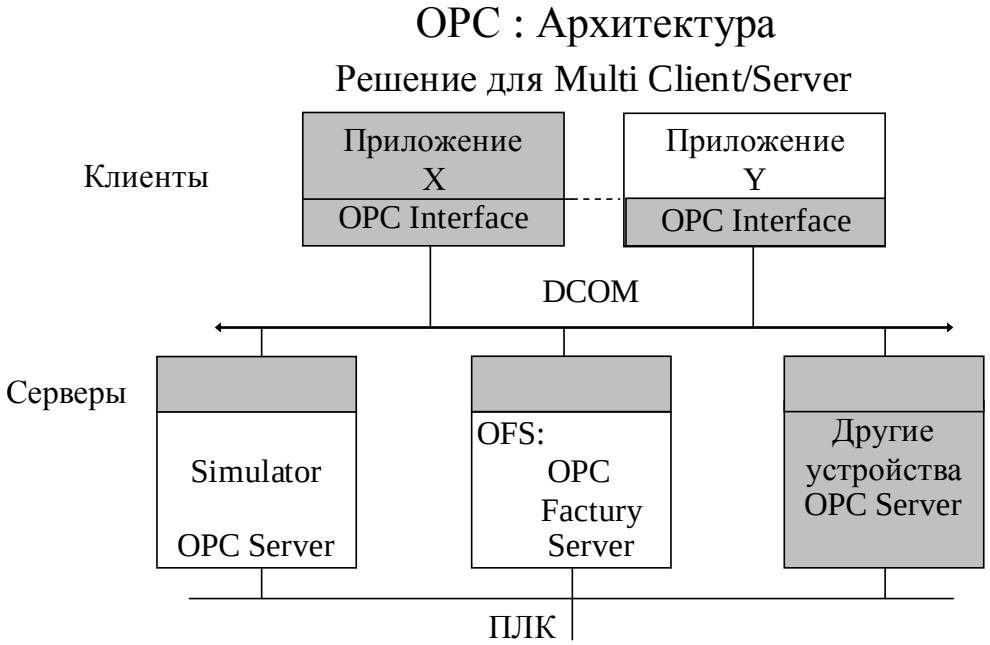


Рисунок 7 – Клиент-серверная архитектура системы OPC

С OPC-решениями интеграция в гетерогенные (неоднородные) системы становится достаточно простой. Применительно к SCADA-системам OPC-серверы, расположенные на всех компьютерах системы управления производственным предприятием, стандартным способом могут поставлять данные со скоростью 10...20 тыс. переменных/сек в программу визуализации, базы данных и т. п., уничтожая, в некотором смысле, само понятие неоднородной системы.

При обмене данными с OPC-сервером допускается два режима:

– периодический режим, когда с заданной частотой данные запрашиваются OPC-клиентом;

– режим по изменению значения, когда обмен происходит при изменении значения переменной на заданную (при конфигурировании обмена) величину.

Предпочтительным является второй тип обмена. Формат передаваемых

данных определяется OPC-протоколом как V (Value - значение), Q (Quality - качество), T (Timestamp - метка времени).

1.4.4.2 Аппаратная реализация связи с устройствами ввода-вывода

Для организации взаимодействия с контроллерами могут быть использованы следующие аппаратные средства:

COM-порты, в этом случае контроллер или объединенные сетью контроллеры подключаются по протоколам RS-232, RS-422, RS-485;

сетевые платы; использование такой аппаратной поддержки возможно, если соответствующие контроллеры снабжены интерфейсным выходом на Ethernet;

вставные платы; в этом случае протокол взаимодействия определяется платой и может быть уникальным; предлагаются их реализации в стандартах ISA, PCI, CompactPCI.

1.4.4.3 Технологии Active X

Говоря о технологиях Active X, можно выделить следующие аспекты:

– выбор типов Active X – объектов, используемых в конкретной SCADA – системе;

– ограничения, накладываемые на применения объектов Active X;

– простота применения в приложении.

Первый аспект является решающим и рассмотрение поддерживаемых типов важно при тестировании.

Объекты ActiveX – это объекты, в основе которых лежит Microsoft COM. Технология COM определяет общую схему взаимодействия компонентов программного обеспечения в среде Windows и предоставляет стандартную инфраструктуру, позволяющую объектам обмениваться данными и функциями между прикладными программами. Большинство SCADA – систем являются контейнерами, которые уведомляются ActiveX о происшедших событиях. Любые объекты ActiveX могут быть загружены в систему разработки большинства SCADA и использованы при создании прикладных программ. Управление объектами ActiveX осуществляется с помощью данных, методов и событийных функций, свойственных выбранному объекту.

Объект ActiveX играет роль сервера по отношению к контейнеру (SCADA – приложению), являющемуся клиентом. Объект ActiveX может быть реализован в двух основных режимах: как сервер, встроенный в процесс (in-process), и как сервер, исполняющийся в отдельном процессе (out-of-process). Этим двум способам исполнения соответствуют две реализации объектов ActiveX: в виде динамических библиотек и в виде исполняемых модулей. Обе реализации обладают и преимуществами, и недостатками.

Для передачи данных из одного процесса в другой вводится механизм «маршалинг» (Marshaling). Стандарт COM поддерживает маршалинг.

Динамически подключаемые библиотеки ActiveX (ActiveX DLL's) или встраиваемые ActiveX размещаются в пространстве процесса контейнерного приложения, поэтому нет необходимости в использовании механизма

«маршалинг» для организации передачи данных между приложением-контейнером и объектом ActiveX. Это уменьшает накладные расходы и увеличивает производительность. Существует ряд преимуществ в реализации объекта ActiveX как встраиваемого сервера.

Исполняемые в отдельном процессе объекты ActiveX (out-of-process) загружаются вне пространства приложения-контейнера. Для передачи данных между объектом ActiveX и контейнером используется механизм «маршалинг». Его применение заметно увеличивает накладные расходы и сильно влияет на производительность.

Многие компании занимаются разработкой драйверов, OPC-серверов, ActiveX-объектов и другого программного обеспечения для SCADA-систем. Этот факт очень важно оценивать при выборе SCADA-пакета, поскольку это расширяет область применения системы непрофессиональными программистами (нет необходимости разрабатывать программы с использованием языков Си или Basic).

1.5 Интеграция многоуровневых систем автоматизации

SCADA-системы ответственны за получение информации с нижнего уровня управления, «снизу», т. е. от различных датчиков через устройства сопряжения, от ПЛК, поставляющих информацию для непосредственного управления производственным процессом. Информация с уровня управления поступает на вход SCADA-систем. На SCADA-уровне осуществляется оперативное управление процессом, принятие тактических решений на основе полученной информации. Сам процесс поступления информации на производстве происходит и «сверху», и «снизу». «Сверху» формируется информация, отвечающая за работу предприятия в целом, осуществляется планирование производства.

Точная, своевременная, достоверная информация на каждом уровне производства позволяет оценить уровень издержек, качество и конкурентоспособность продукции.

Огромное стратегическое значение имеет то, насколько инструментальные системы АСУ ТП связаны с Microsoft BackOffice Suite, поскольку последний стал наиболее распространенным офисным программным продуктом. Составляющие SCADA-систем должны легко интегрироваться с такими продуктами, как Microsoft SQL Server, Windows NT Server, System Management Server, SNA Server и Mail-Server. Такие решения существенно расширяют возможности всего производственного персонала в смысле возможности доступа к полной информации о любом этапе производства.

Все более актуальным становится требование передачи на Web-узлы как статической (в определенные моменты времени), так и динамической (постоянно) информации. Объекты ActiveX позволяют передавать данные из SCADA-системы на Web-страницы. Но имеются и более многофункциональные компоненты типа Web-Client в Monitor Pro, Web-Aktivator в Trace Mode или Scout в Factory Suit, обеспечивающие возможность доступа к системам автоматизации через Internet/Intranet и позволяющие удаленному пользователю

взаимодействовать с прикладной задачей автоматизации как с простой Web-страницей.

Следует отметить тенденции включения SCADA-систем в системы комплексной автоматизации предприятия. Это обеспечивает точную и своевременную информацию на каждом уровне производства.

Применение в SCADA-системах новых технологий, разработка инструментальных средств комплексной автоматизации предприятия свидетельствуют о стремлении и возможности фирм-разработчиков постоянно совершенствовать свои продукты, что является немаловажным фактором при выборе инструментального средства, даже если не все его технологические решения в ближайшее время будут использованы разработчиком.

1.6 Сравнительный анализ и тестирование SCADA – систем

Выбор наиболее приемлемой SCADA-системы представляет собой сложную многокритериальную задачу, решением которой является компромисс между надежностью, стоимостью, техническим уровнем, полнотой программного обеспечения, комфортностью, затратами на сервисное обслуживание и т. д.

В большинстве SCADA – систем присутствуют известные базовые свойства, но технологии и средства их реализации достаточно сильно различаются. Именно мера реализации каждого свойства в SCADA – системе определяет необходимость в разработке дополнительного программного обеспечения (новые драйверы ввода-вывода, графические объекты; функции, расширяющие список базовых функций, встроенные библиотеки). Для минимизации этой процедуры важны три фактора: степень соответствия выбранного SCADA – пакета вашей задаче, понимание тонкостей реализации конкретной прикладной системы поставщиками SCADA – продукта и качество осуществляемой ими технической поддержки.

Существенное влияние на выбор SCADA – системы оказывают следующие свойства: тип, мощность, динамичность объекта автоматизации; учет дальнейшего распространения SCADA – системы на другие объекты автоматизации; класс систем автоматизации, контроль и учет; имеющаяся платформа; число и расположение пультов операторов; число и типы контроллеров; имеющаяся сетевая архитектура; число измеряемых величин на каждый пульт; необходимость обработки измерительной информации; надежность.

Еще не так давно SCADA-системы различались между собой такими параметрами, как: мощность векторной графики; особенность построения графиков, трендов; формат экспорта и импорта изображений; возможность работы с мультимедиа; тиражирование изображений; проектирование первичной переработки данных; написание пользователем программ; особенности отладки отдельных программ (эмуляция); возможность эмуляции объекта автоматизации; обучаемость персонала; открытость протоколов связи с контроллерами и сетями; наличие интерфейса с БД, электронными таблицами и Web-броузерами; перечень драйверов к контроллерам; полнота документации;

особенности технического сопровождения; цена базового комплекта.

В настоящее время практически все SCADA-системы работают под Windows и в связи с этим используют одну и ту же платформу, обладают примерно одинаковыми функциональными и графическими возможностями, а на первый план при сравнении выходят такие критерии, как надежность работы, обмен данными, удобство работы, техническая поддержка и цена. На рисунке 8 представлена иерархическая структура критериев, по которым оцениваются в последнее время SCADA-системы.

При выборе SCADA-системы необходимо идти от задачи, так как она во многом будет определять дальнейшее решение. Например, если понравившаяся вам SCADA-система не поддерживает имеющиеся у вас контроллеры, необходимо подумать, стоит ли вам браться за разработку драйверов.

Рисунок 8 – Критерии выбора SCADA-систем

SCADA-системы можно анализировать в разных срезах и один из них: кто выбирает SCADA-продукт - конечный пользователь, т. е. технолог, или системный интегратор, имеющий опыт в области создания проектов.

Процедура выбора должна включать в себя следующие этапы.

1. Составление технических требований к SCADA-системе.

2. Выделение двух (максимум трех) SCADA-систем, наиболее подходящих к объекту автоматизации.

3. Оценка выделенных SCADA-систем по отзывам пользователей.

4. Личное ознакомление со SCADA-системами, их тестирование, конкретизация состава пакета.

5. Определение наилучшей SCADA-системы и принятие решения.

Методики определения надежности SCADA-систем отсутствуют, хотя важность этого критерия составляет, по оценкам специалистов, около 70 %.

Косвенным показателем надежности считается количество инсталляций. Однако, по мнению ряда специалистов, роль этого показателя незначительна, если число инсталляций системы превышает 1000. В большей степени вас волнует вопрос: сколько внедрений имеет система не в мире, а в России, если вы - российский производитель, и сколько в Германии, если ваше производство размещено в Германии.

На втором месте, по мнению многих специалистов, должен быть такой критерий, как обмен данными. Здесь важными подкритериями являются поддержка стандартных сетевых протоколов и форматов данных, включая Web-технологии, наличие встроенных драйверов к отечественным и зарубежным контроллерам, а также производительность системы.

Важны протоколы, используемые для организации взаимодействия между компонентами, расположенными как на одном, так и на разных узлах. Какие протоколы, для какой SCADA-системы предпочтительны и почему? При тестировании они должны быть указаны, поскольку от них зависит как производительность текущего SCADA-приложения, так и параллельно загруженных программ. Подобный анализ необходим при тестировании конкретных модулей или возможностей системы.

Важнейшим российским критерием является цена. Как правило, критерий выбора - это соотношение функциональность/стоимость. Специалисты обращают внимание на зависимость цены системы от конфигурации, возможность получения новых версий и бесплатного обновления релизов, наличие бесплатной системы разработки.

В условиях повышенных требований к оптимальному использованию конкретного высокотехнологичного продукта становится более значительной роль технической поддержки при ужесточении требований к ее квалификации и компетентности менеджеров продуктов. Западные пакеты проигрывают по русификации документации и, тем более, программного обеспечения, по «либеральности» технической поддержки, наличию «горячей» линии, а главное, по возможности поддержки от разработчика. Последнее не может обеспечить ни один западный разработчик, так как работает в России через дистрибьюторов.

Несмотря на то, что критерий «удобство работы» в оценках специалистов имеет незначительный удельный вес, именно он вызывает наибольший интерес. Это возможность автоматического построения проекта, универсальность, наличие стандартных языков математического описания данных и процессов, удобство пользовательского интерфейса (работа с редакторами), качество графики и стандартных изображений, эмуляция работы.

Так как общее поле деятельности ведущих компаний-производителей сегодня концентрируется в области MS Windows NT, а общие технические возможности систем достаточно близки, главный упор делается на качество технической поддержки, на качество обучения пользователей, на концентрацию и качество дополнительных комплексных услуг по освоению и внедрению конечной системы управления, другими словами, на сокращение издержек разработчиков, на инжиниринг и менеджмент своих проектов, на уменьшение

стоимости сопровождения конечной системы. Именно эти показатели сегодня, в основном, влияют на рейтинг и рыночный успех той или иной SCADA-системы. Эти показатели даже более важны, чем абсолютные стоимостные характеристики SCADA-систем.

Тестирование SCADA-систем затрагивает актуальные вопросы языков программирования, коммуникационных протоколов, новых технологий. Практически каждый из них требует детального изучения и анализа, который должен стать основой для разработки методики испытаний с целью определения максимальной производительности, функциональных возможностей или недостатков конкретной SCADA-системы.

Выбор алгоритмов тестирования и критериев, конечно, зависит от того, кто проводит тестирование - журнал или организация для своих проектов. В первом случае ориентация делается на общие принципы построения продуктов, на многообразие поддерживаемых протоколов, на производительность и т. д., во втором случае оговариваются особые, ориентированные на определенные проекты условия, следовательно, возможны специализированные алгоритмы тестирования.

2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ

2.1 SCADA система TRACE MODE

TRACE MODE 6 – это программный комплекс, предназначенный для разработки и запуска в реальном времени распределенных автоматизированных систем управления технологическими процессами (АСУТП) и решения ряда задач управления предприятием (АСУП).

Для решения задач АСУП в TRACE MODE 6 интегрирован пакет **T-FACTORY**.

Комплекс программ TRACE MODE 6 можно разделить на 3 части.

Интегрированная среда разработки проекта (ИС) – единая программная оболочка, содержащая все необходимые средства для разработки проекта.

Под **проектом** в TRACE MODE 6 понимается вся совокупность данных и алгоритмов функционирования распределенной АСУ (АСУТП и/или T-FACTORY), заданных средствами TRACE MODE.

Итогом разработки проекта в ИС является создание файлов, содержащих необходимую информацию об алгоритмах работы АСУ. Эти файлы затем размещаются на аппаратных средствах (компьютерах и контроллерах) и выполняются под управлением исполнительных модулей TRACE MODE.

Исполнительные модули (мониторы, MPB) – программные модули различного назначения, под управлением которых в реальном времени выполняются составные части проекта, размещаемые на отдельных компьютерах или в контроллерах.

Составная часть проекта, размещаемая на отдельном компьютере или в контроллере и выполняемая под управлением одного или нескольких исполнительных модулей TRACE MODE, называется **узлом проекта**.

В общем случае размещение узла на том же аппаратном средстве, на котором он должен исполняться под управлением монитора, не является обязательным – мониторы могут загружать узлы с удаленных аппаратных средств.

Драйверы обмена – драйверы, используемые мониторами TRACE MODE для взаимодействия с устройствами, протоколы обмена с которыми не встроены в мониторы.

2.2 Принцип работы монитора. Канал TRACE MODE 6

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

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

Под записью значения в канал в общем случае понимается присвоение значения переменной (атрибуту) **Входное значение** этого канала.

Для канала могут быть сконфигурированы два важнейших свойства – **связь** и **вызов**.

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

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

Совокупность каналов узла называется **базой каналов** этого узла.

Класс канала определяет его общее назначение. При разработке проекта могут быть созданы каналы только предопределенных классов.

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

(мнемоническое обозначение). Идентификаторами атрибута являются его индекс и, в ряде случаев, короткое имя.

Каналы содержат внутри себя предопределенные алгоритмы (часть из них может настраиваться пользователем), в соответствии с которыми некоторые атрибуты канала устанавливаются или вычисляются монитором в зависимости от состояния или значения других атрибутов.

Исполнение внутренних алгоритмов канала и анализ его атрибутов монитором называется **пересчетом канала**.

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

При вызове программы числовым каналом (кроме канала TIME) может быть выполнено следующее преобразование его атрибутов:

аппаратное значение (A)<=>программа<=>реальное значение (R)

Направление преобразования зависит от типа канала:

INPUT: **A=>программа=>R**

OUTPUT: **R=>программа=>A**

Направление преобразования следует учитывать для корректной привязки атрибутов числового канала к аргументам программы.

По результатам анализа атрибутов монитор выполняет действия, заданные с помощью канала (например, вызов шаблона), – эта процедура называется **отработкой канала**. Отработка канала после его пересчета выполняется при определенных условиях. При пересчете базы каналов пересчет конкретного канала также выполняется при определенных условиях.

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

2.3 Обеспечение работы распределенных АСУ

Для обеспечения функционирования распределенных АСУ мониторы TRACE MODE поддерживают различные виды взаимодействия с аппаратными средствами и программными приложениями.

Мониторы поддерживают обмен между собой по протоколу **M-LINK** (открытый протокол фирмы ADASTRA) при связи компьютеров и/или контроллеров, на которых запущены, по последовательным интерфейсам RS-232/422/485, в том числе при подключении к COM-портам дополнительного оборудования, которое конфигурируется при разработке проекта в ИС:

модемов – при использовании для связи выделенных/коммутируемых телефонных линий;

радиомодемов – при использовании для связи радиоканала;

GSM-модемов – при использовании для связи GSM-сети;

конвертеров интерфейсов RS-232/422/485.

Под COM-портом здесь понимается как штатный последовательный порт устройства, так и, например, порты, доступные при установке расширителей портов в шины ISA/PCI компьютера.

Мониторы поддерживают обмен одновременно по 32 последовательным портам.

Сеть M-LINK – это сеть типа MASTER-SLAVE с одним ведущим узлом в одной сети в один момент времени.

Узел может иметь статус SLAVE только в сети M-LINK, при обмене по последовательному интерфейсу по любому другому протоколу узел имеет статус MASTER.

Мониторы поддерживают обмен между собой по протоколу TCP/IP при связи компьютеров и/или контроллеров, на которых запущены, по сети (физическая архитектура сети не имеет значения) – в этом случае на прикладном уровне используется протокол **I-NET** фирмы ADASTRA.

Мониторы поддерживают обмен одновременно по 4 сетевым адаптерам.

Мониторы поддерживают обмен с платами ввода/вывода, установленными в системные шины (ISA/PCI/PC-104) компьютеров/контроллеров, на которых запущены.

Мониторы поддерживают обмен с внешними устройствами (распределенными модулями):

по некоторым модификациям протокола MODBUS по последовательным интерфейсам;

по протоколу MODBUS TCP/IP по сети;

по последовательному интерфейсу по протоколам контроллеров/серий модулей LAGOON, ROBO, NuDAM-6000, I-7000, ADAM-4000, ADAM-5000/485, RIO-2000 и т.п. (в TRACE MODE эти протоколы называются протоколами **DCS**).

Мониторы поддерживают обмен с произвольными устройствами через драйверы как по стандартным интерфейсам (в том числе полевым шинам), так и при использовании дополнительных устройств, реализующих необходимые интерфейсы, – коммуникационных плат, преобразователей интерфейсов и т.п. Номенклатура драйверов TRACE MODE для обмена с различным оборудованием постоянно расширяется. Кроме того, программный интерфейс взаимодействия монитора с такими драйверами – интерфейс **TCOM** – является открытым, что позволяет пользователю разработать драйверы обмена с любым оборудованием.

Мониторы поддерживают обмен с клиентами/серверами OPC.

Мониторы поддерживают обмен между собой и с приложениями Windows по DDE/NetDDE.

Мониторы поддерживают обмен с локальными/удаленными базами данных как по ODBC, так и через собственные интерфейсы СУБД.

Таким образом, TRACE MODE не накладывает практически никаких ограничений на топологию систем управления и используемые в них аппаратные средства.

2.4 Резервирование

Резервирование – это метод экстенсивного повышения надежности АСУ посредством использования дополнительных (резервных) аппаратных средств (например, дополнительного сетевого адаптера, дополнительного COM-порта с подключенным к нему дополнительным контроллером и т.п.). Для контроля работоспособности оборудования мониторы используют ряд механизмов, в том числе анализируют каналы, связанные со специальными системными переменными TRACE MODE. При обнаружении отказа основного оборудования мониторы переключаются на резервное. Для узла может быть создано до 2 резервов, предусмотрена синхронизация данных архивов дублированных/троированных узлов и т.п.

2.5 Автопостроение

ИС содержит информацию о конструктивном исполнении ряда контроллеров, о платах расширения, которые могут быть вставлены в крейт того или иного контроллера, о внешних модулях, которые могут быть подключены к тому или иному контроллеру и т.д., а также об источниках/приемниках, имеющихся на платах/модулях. На базе этой информации в ИС реализованы различные механизмы **автопостроения** – например, источники/приемники платы, выбранной в списке, создаются автоматически, автоматически создаются каналы, связанные с источниками/приемниками и т.п.

Особой разновидностью автопостроения является автоматическое создание каналов мониторами – например, каналов, связанных с источниками/приемниками (такие каналы создаются мониторами в том случае, если в ИС задана связь источников/приемников с другими компонентами проекта через аргументы этих компонентов).

2.6 Математическая обработка данных

Любая АСУ требует математической обработки данных – как в измерительных информационных потоках (датчик => УСО => контроллер => операторская станция), так и в управляющих (операторская станция => контроллер => исполнительное устройство).

Для математической обработки данных в TRACE MODE 6 предусмотрены следующие средства:

внутренние алгоритмы числовых каналов;

программы. Для разработки программ в ИС встроены языки **Техно ST, Техно SFC**, **Техно FBD, Техно LD** и **Техно IL**, являющиеся модификациями языков **ST** (Structured Text), **SFC** (Sequential Function Chart), **FBD** (Function Block Diagram), **LD** (Ladder Diagram) и **IL** (Instruction List) стандарта IEC61131-3. Программы, разрабатываемые в ИС, позволяют использовать функции из внешних библиотек (DLL).

Эти средства обеспечивают возможность математической обработки данных в любом звене информационного потока.

2.7 Архивирование каналов узла

Для обеспечения архивирования параметров технологического процесса мониторы TRACE MODE поддерживают функцию записи значений атрибутов каналов в базы данных реального времени – архивы СУБД РВ SIAD/SQL 6 (в дальнейшем – архивы СПАД или архивы SIAD). Сообщения по каналу заносятся в архив при изменении его значения.

Мониторы, работающие в контроллерах, поддерживают индивидуальные архивы.

Для каждого узла в ИС могут быть определены 3 пользовательских архива SIAD (локальных или удаленных). Существует также системный архив, используемый мониторами для внутренних целей. При конфигурировании канала указывается, в какой из заданных пользовательских файлов он должен архивироваться.

Архивы SIAD имеют следующие основные характеристики:

точность значения времени – 1 мс;

скорость записи в архив для рабочей станции с процессором Pentium-4 с тактовой частотой 2 ГГц – свыше 600 тыс. параметров в секунду.

Архивные данные могут использоваться мониторами, экспортироваться в приложения Windows, а также отображаться на графическом экране (эту функцию реализует графический элемент **Тренд**).

2.8 Архивирование каналов проекта

Специализированный монитор **Logger** (Регистратор) может записывать в определенный для него архив SIAD значения атрибутов каналов всех узлов проекта. Данные в этот монитор могут быть переданы по протоколам I-NET и M-LINK.

В проекте может присутствовать до 3 регистраторов, в том числе имеющих резервы.

2.9 Отчет тревог и генерация сообщений

Мониторы могут генерировать сообщения в различных ситуациях при работе АСУ – например, при выходе значения канала класса FLOAT за установленную границу, при изменении статуса работника (т.е. при изменении соответствующего атрибута канала класса **Персонал**) и т.п. Эти сообщения заносятся в специальный текстовый файл – **отчет тревог** (**ОТ**), который конфигурируется для узла. В ОТ заносятся сообщения по каналам, для которых установлен соответствующий флаг.

Конфигурирование ОТ разрешает монитору генерацию сообщений. Тексты сообщений для событий могут быть заданы в словарях. Если канал связан со словарем, генерируются сообщения из словаря, в противном случае монитор генерирует сообщения по умолчанию. Для некоторых каналов критерии генерации сообщений зависят от параметров этих каналов.

В словаре могут быть заданы дополнительные направления передачи сообщений – например, в виде SMS-сообщений на указанный номер сотового телефона, по сети консолям и т.п.

Ряд графических элементов, используемых при разработке графических экранов, позволяет оператору заносить в отчет тревог произвольные сообщения, а также просматривать все сообщения ОТ и квитировать их (информация о квитировании также заносится в отчет тревог).

2.10 Графический интерфейс оператора

TRACE MODE 6 обеспечивает графическое представление хода выполнения техпроцесса, а также управление техпроцессом с помощью графических средств.

Графический интерфейс оператора реализуется в нескольких видах:

в виде набора **графических экранов**, шаблоны которых разрабатываются в редакторе представления данных (РПД; этот редактор входит в состав редакторов ИС), – для узлов, которые исполняются мониторами на аппаратных средствах, имеющих достаточную производительность и другие необходимые характеристики (например, при использовании объемной графики от видеосистемы требуется поддержка OpenGL 1.1). В состав TRACE MODE 6 входит большое количество ресурсов – текстов, изображений, видеоклипов, различных графических объектов, – которые могут использоваться при разработке графических экранов. Наборы ресурсов могут создаваться пользователем;

в виде набора **графических панелей**, шаблоны которых разрабатываются в модификации РПД, – для узлов, которые исполняются мониторами на аппаратных средствах, имеющих ограниченную производительность (например, в контроллерах с ОС Windows CE);

в виде **мнемосхем** – для узлов, исполняемых мониторами в среде DOS.

2.11 Генерация документов (отчетов)

Мониторы поддерживают функцию генерации файлов документов (отчетов) формата HTML. Эти документы (отчеты), в том числе, могут содержать информацию о текущих параметрах техпроцесса.

Документы генерируются по шаблонам, для разработки которых в ИС встроен соответствующий редактор.

МРВ поддерживают генерацию документов по ограниченному числу шаблонов. Для генерации документов без ограничений следует использовать специализированный монитор – **Сервер документирования**.

2.12 Защита проекта

Для защиты от несанкционированного доступа к редактированию проекта и/или управлению АСУ для каждого узла должны быть определены пользователи и заданы их права. Для этих целей существует специальные каналы – класса **Пользователь**. Мониторы контролируют права пользователей и записывают результаты контроля в архив и ОТ.

2.13 Технология разработки проекта в ИС

Разработка проекта в ИС включает следующие процедуры:

создание структуры проекта в навигаторе;

конфигурирование или разработка структурных составляющих – например, разработка шаблонов графических экранов оператора, разработка шаблонов программ, описание источников/приемников и т.д.;

конфигурирование информационных потоков;

выбор аппаратных средств АСУ (компьютеров, контроллеров и т.п.);

создание узлов в слое **Система** и их конфигурирование;

распределение каналов, созданных в различных слоях структуры, по узлам и конфигурирование интерфейсов взаимодействия компонентов в информационных потоках;

сохранение проекта в единый файл для последующего редактирования (с помощью команды **Сохранить** или **Сохранить как**;

экспорт узлов в наборы файлов для последующего запуска под управлением мониторов TRACE MODE (по команде **Сохранить для MPB)**.

Перечисленные процедуры (за исключением двух заключительных) и входящие в их состав операции могут выполняться в произвольном порядке. Например, можно начинать разработку проекта с разработки шаблонов графических экранов оператора, с создания узлов и их каналов в слое **Система** (если аппаратные средства АСУ известны заранее), можно конфигурировать каналы и информационные потоки после распределения каналов по узлам и т.п.

Чтобы получить представление о средствах разработки, которыми располагает ИС, рекомендуется рассмотреть нижеследующий пример создания проекта.

TRACE MODE располагает также средствами для объектного проектирования.

2.14 Классификация компонентов

По функциональному назначению компоненты проекта относятся к одному из следующих видов:

каналы – компоненты, определяющие алгоритм работы проекта. Каналы могут создаваться в различных слоях, однако их окончательное распределение по узлам в слое Система обязательно – в противном случае они не будут экспортированы для MPB;

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

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

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

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

последовательные порты – параметры COM-портов;

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

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

2.15 Классификация слоев

Предопределенные слои структуры проекта имеют следующее назначение:

Ресурсы – для создания пользовательских наборов текстов, изображений и видеоклипов, а также графических объектов;

Шаблоны программ – для создания шаблонов программ;

Шаблоны экранов – для создания шаблонов графических экранов и графических панелей;

Шаблоны связей с БД – для создания шаблонов связей с базами данных;

Шаблоны документов – для создания шаблонов документов (отчетов);

База каналов – этот слой является хранилищем всех каналов проекта. Выполнять операции с каналами (в том числе создавать их) можно в различных слоях, однако во всех случаях эти операции на самом деле реализуются в слое База каналов. В любом другом слое, где выполняется команда для совершения операции с каналом, ее результат только отображается – поэтому существуют команды удаления и уничтожения каналов;

Система – для конфигурирования узлов и их составляющих (узел создается как корневая группа этого слоя);

Источники/приемники – для создания встроенных генераторов, шаблонов каналов обмена с различными устройствами и программными приложениями, а также для конфигурирования системных переменных TRACE MODE 6,

Технология – для разработки проекта от технологии (т.е. с группировкой компонентов по признаку их принадлежности к технологическому объекту). При отладке проекта слой Технология может играть роль узла – для него определена команда Сохранить узел для МРВ. Кроме того, для этого слоя определены команды взаимодействия с технологической базой данных;

Топология – для разработки проекта от топологии (т.е. с группировкой компонентов по месту расположения);

КИПиА – для описания электрических соединений АСУ;

Библиотеки компонентов – для создания библиотек объектов – проектных решений отдельных задач. Этот слой содержит предопределенные группы Системные и Пользовательские. В группе Системные содержатся библиотеки, подключенные к ИС по умолчанию.

2.16 Классификация узлов

Узлы проекта создаются как корневые группы слоя **Система**. Предопределенное название узла указывает на семейство мониторов, для которых данный узел предназначен. Узел может содержать только те компоненты, которые поддерживаются мониторами соответствующего семейства.

В общем случае, узлы могут выполняться под управлением различных мониторов.

Как правило, узел выполняется на отдельном аппаратном средстве. В случае запуска двух и более узлов на одном аппаратном средстве оно должно быть оборудовано соответствующим количеством сетевых карт.

RTM

Узел **RTM** предназначен для запуска на компьютере под управлением исполнительных модулей семейства **RTM** (MPB) – мониторов с поддержкой отображения графических экранов оператора, поддержкой обмена по последовательному интерфейсу и сети с различным оборудованием и выполняющего пересчет каналов всех классов, кроме каналов T-FACTORY.

T-FACTORY

Узел **T-FACTORY** предназначен для запуска на компьютере под управлением исполнительных модулей семейства **T-FACTORY** – мониторов для решения задач АСУП.

MicroRTM

Узел **MicroRTM** предназначен для запуска на компьютере или в контроллере под управлением исполнительных модулей семейства **Micro RTM.** Основное отличие этих мониторов от MPB – отсутствие поддержки отображения графических экранов.

Logger

Узел **Logger** предназначен для запуска на компьютере под управлением исполнительного модуля **Logger** (регистратор) – монитора, способного вести архивы по каналам всех узлов проекта.

EmbeddedRTM

Узел **EmbeddedRTM** предназначен для запуска на компьютере или в контроллере под управлением исполнительных модулей семейства **Embedded RTM** – мониторов с поддержкой графических панелей, поддержкой обмена с оборудованием по различным протоколам и выполняющего пересчет каналов.

NanoRTM

Узел **NanoRTM** предназначен для запуска в контроллере под управлением исполнительного модуля **Nano RTM** – монитора, аналогичного **Micro RTM**, но предназначенного для работы с малым числом каналов.

Console

Узел **Console** предназначен для запуска на компьютере под управлением исполнительных модулей, которые, в отличие от MPB, не выполняют пересчет каналов, предназначенных для работы с данными. Консоли позволяют получать данные от других узлов проекта по сети, отображать их на графических экранах

и управлять технологическим процессом из графики. Консоли не могут взаимодействовать с узлами T-FACTORY.

TFactory_Console

Узел **TFactory_Console** предназначен для запуска на компьютере под управлением исполнительных модулей, аналогичных консолям, но, кроме того, способных взаимодействовать с узлами T-FACTORY.

TM_OPC_Server

Этот узел зарезервирован.

2.17 Назначение групп источников (приемников)

В слое **Источники/Приемники** могут быть созданы следующие предопределенные корневые группы:

1. PC-based контроллеры;
2. Распределенные УСО (DCS);
3. Платы ввода/вывода;
4. Терминалы;
5. PLC;
6. OPC-группа;
7. DDE-группа;
8. MODBUS-группа;
9. Пользовательские драйверы;
10. Диагностика и сервис;
11. Генераторы;
12. Модели.

Первые 9 групп предназначены для создания шаблонов каналов обмена с различными устройствами и программными приложениями, обмен с которыми поддерживается мониторами TRACE MODE 6. Источники/приемники устройств называются **аппаратными каналами** или **тегами**

При создании описаний конкретных плат/модулей шаблоны каналов обмена с ними создаются в навигаторе автоматически как компоненты проекта. Набор автоматически созданных источников/приемников может быть отредактирован вручную (например, с помощью команд контекстного меню).

При автоматическом создании для источников/приемников по умолчанию задается ряд параметров, которые могут быть отредактированы в соответствующих редакторах. Группы, непосредственно включающие источники/приемники, также имеют редакторы, с помощью которых возможно групповое редактирование параметров источников/приемников.

В навигаторе проекта поддерживается автопостроение каналов, связанных с источниками/приемниками, при копировании/вставке этих источников/приемников в группу каналов. Каналы, созданные таким способом, в ряде случаев требуют дополнительной конфигурации.

Если источники/приемники связаны с другими компонентами проекта через аргументы, то каналы, настроенные на эти источники/приемники, создаются мониторами в реальном времени.

2.18 Каналы класса FLOAT

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

в число, соответствующее амплитуде некоторого электрического сигнала (в том числе унифицированного – 0-10V, 4-20mA и т.д.);

в число, соответствующее проценту от диапазона изменения некоторого электрического сигнала;

в двоичный код (после АЦП).

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

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

канал INPUT:

масштабирование;

фильтрация одиночных пиков;

фильтрация малых изменений (апертура);

экспоненциальное сглаживание;

канал OUTPUT:

экспоненциальное сглаживание;

линейное сглаживание;

фильтрация малых изменений (апертура);

клиппирование;

масштабирование.

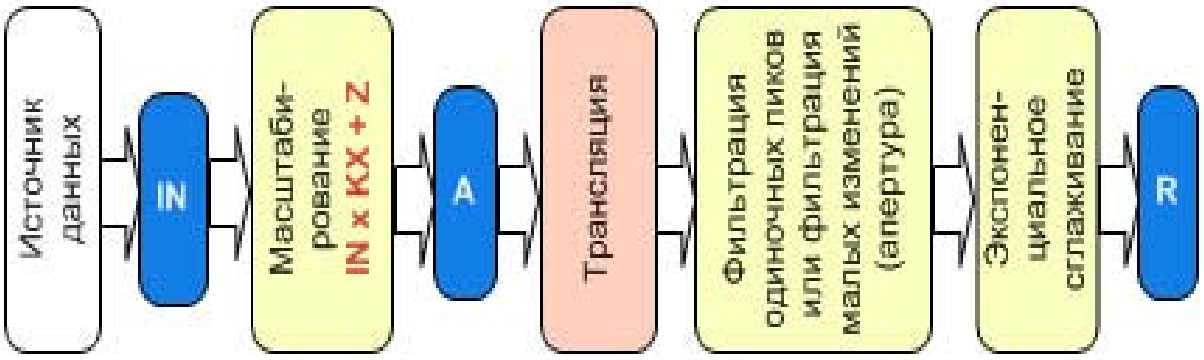
При использовании экспоненциального сглаживания фильтрация малых изменений в канале FLOAT не выполняется.

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

Атрибуты **Входное значение** (2, **In**), **Аппаратное значение** (1, **A**), **Реальное значение** (0, **R**) и **Выходное значение** (9, **Q**) задействованы во внутренних алгоритмах канала FLOAT следующим образом рисунок 9:

В указанных на рисунках формулах масштабирования **KX** и **Z** являются значениями атрибутов **Множитель** (33, **KX**) и **Смещение** (34, **Z Дрейф нуля**, МРВ – **ZERO**).

- канал INPUT:



- канал OUTPUT:

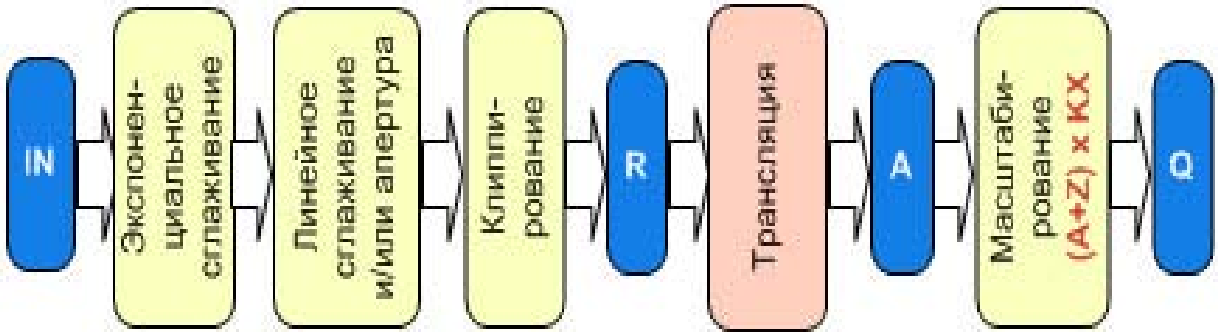


Рисунок 9 – Структура канала типа FLOAT

Масштабирование.

Эта процедура используется только в каналах, работающих с аналоговыми переменными. Она включает в себя две операции: умножение и смещение. Последовательность этих операций меняется в зависимости от типа канала:

У каналов типа INPUT входное значение умножается на заданный множитель и к полученному результату добавляется величина смещения. Результат присваивается аппаратному значению канала.

У каналов типа OUTPUT к аппаратному значению добавляется величина смещения, затем эта сумма умножается на заданный множитель, а результат присваивается выходному значению канала.

Трансляцией называется вызов программы числовым каналом (это единственное назначение свойства вызов числового канала).

При вызове программы числовым каналом (кроме канала TIME) может быть выполнено следующее преобразование его атрибутов:

аппаратное значение (A)<=>программа<=>реальное значение (R)

Направление преобразования зависит от типа канала:

INPUT: A=>программа=>R

OUTPUT: R=>программа=>A

Фильтрация

Данная процедура присутствует только у аналоговых каналов. Набор выполняемых ею операций отличается для входных и выходных каналов.

У каналов типа INPUT фильтрация выполняется после процедуры трансляции до формирования реального значения. Фильтрация включает в себя следующие операции:

подавление случайных всплесков в тракте измерения;

подавление малых колебаний значения канала;

экспоненциальное сглаживание;

контроль шкалы – отслеживание выхода реального значения канала за установленные границы шкалы;

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

ограничение скорости изменения реального значения;

подавление малых колебаний значения канала;

экспоненциальное сглаживание;

контроль шкалы – обрезание величины управляющего воздействия до границ шкалы канала.

Экспоненциальное сглаживание

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

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

Апертура

Данный метод обработки реализует фильтрацию малых изменений входного сигнала. Он обеспечивает зону нечувствительности. Если изменение выходной величины по отношению к предыдущему значению меньше зону нечувствительности, то выход не меняется.

Введение этого метода позволяет существенно сократить интенсивность информационных потоков, а также увеличить глубину сохранения и скорость доступа к данным в архивах

Клиппирование – ограничение реального значения в канале FLOAT типа OUTPUT по максимуму и минимуму.

2.19 Канал класса HEX16

Канал класса **HEX16** предназначен для работы с 2-байтовыми целыми числами.

Кроме атрибутов, которые имеют каналы всех классов, и атрибутов, общих для числовых каналов, каналы класса HEX16 имеют специфические атрибуты.

К специфическим атрибутам, которые могут быть заданы в редакторе канала, относятся следующие:

раздел "Параметры":

Размерность в битах – (56, **nBits**) **Number Bits** – данный атрибут задает число байт, участвующих в процедуре инверсии:

<= 8 – 1 байт;

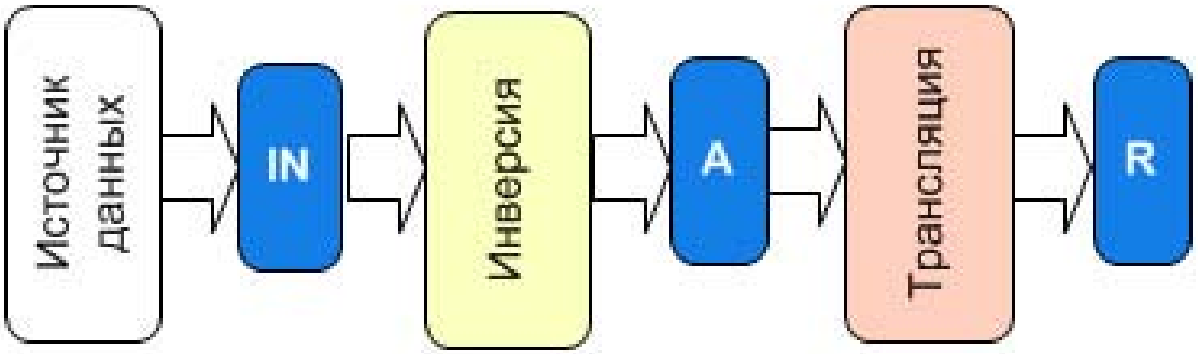
> 8 – 2 байта;

флаг **Инверсия** (40, **NM**) – если этот флаг установлен, инвертирование в канале разрешено;

флаг **DEC** (84, **HD**) – если этот флаг установлен (**HD**=1), значение канала отображается в профайлере в десятичном виде; если флаг не установлен (**HD**=0) – в шестнадцатеричном. От этого флага зависит также алгоритм записи сообщений в отчет тревог.

Атрибуты **Входное значение** (2, **In**), **Аппаратное значение** (1, **A**), **Реальное значение** (0, **R**) и **Выходное значение** (9, **Q**) канала HEX16 задействованы в его алгоритмах обработки следующим образом рисунок 10:

- канал типа INPUT:



- канал типа OUTPUT:

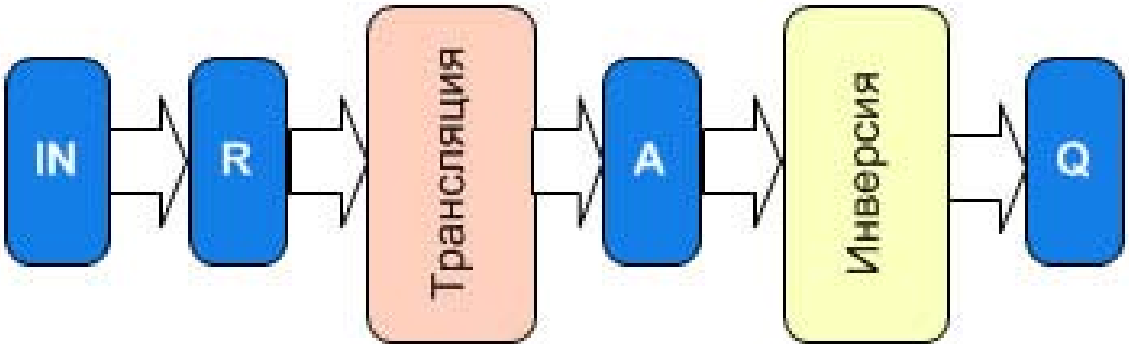


Рисунок 10 – Структура канала типа HEX16

В отсутствие процедуры трансляции, в каналах HEX16 не используется атрибут **Аппаратное значение** (1, **A**).

Инверсия

При обработке дискретных сигналов часто требуется использовать не полученное с датчика, а инвертированное значение. Операцию инверсии можно реализовать на стадии первичной обработки. Для этого каналу необходимо установить флаг инверсии и указать, значения каких битов его аппаратного значения надо инвертировать при выполнении процедуры трансляции.

2.20 Языки программирования в TRACE MODE

Для программирования алгоритмов функционирования разрабатываемого проекта АСУ в TRACE MODE 6 включены языки **Техно ST, Техно SFC, Техно FBD, Техно LD** и **Техно IL**. Данные языки являются модификациями языков **ST (Structured Text)**, **SFC (Sequential Function Chart), FBD (Function Block Diagram), LD (Ladder Diagram)** и **IL (Instruction List)** стандарта IEC61131-3.

Программы и некоторые их компоненты (функции, шаги и переходы SFC и т.п.) могут быть разработаны на любом из встроенных языков в

соответствующем редакторе, при этом языки для программы и ее компонентов выбираются независимо.

Для создания и редактирования свойств аргументов, переменных, функций и структурных типов программы, а также для использования в программе функций из внешних библиотек в интегрированную среду разработки проекта встроены специальные табличные редакторы.

TRACE MODE 6 имеет также средства для отладки программ.

Основным языком программирования TRACE MODE 6 является Техно ST. Программы, разработанные на языках Техно LD, Техно SFC и Техно FBD, перед компиляцией транслируются в Техно ST. IL-программы перед компиляцией частично транслируются в ST, частично – в ассемблер. Отсюда следует, например, что ключевые слова Техно ST являются таковыми и для всех других языков.

2.20.1 Техно ST

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

выражение – последовательность операндов, разделителей и символьных операторов, задающая вычисление без присвоения результата;

предложение – последовательность лексем, определяющая выполнение логически законченного промежуточного действия. Таким действием может быть присвоение переменной результата вычислений, вызов функции-блока и т.п. Операторы (кроме символьных) также образуют предложения.

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

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

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

Основная точка входа в программу определяется следующей конструкцией:

program
{определение аргументов}
{список предложений}
end_program

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

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

Основная точка входа создается автоматически при создании программы. Если для программы выбран язык **ST** или **IL**, конструкция **program ...**

end_program отображается в листинге. Если для программы выбран язык **SFC**, **LD** или **FBD**, основная точка создается во внутреннем представлении и недоступна для просмотра.

2.20.2 Техно SFC

Язык **Техно SFC** позволяет создавать программы в виде алгоритма, состоящего из SFC-шагов и SFC-переходов.

Для SFC-шагов задаются выполняемые действия, для SFC-переходов – условия переходов между шагами, поэтому в дальнейшем SFC-переход иногда называется SFC-условием.

Для перехода от одного шага к другому SFC-условие, действующее на этом переходе, должно быть истинным (т.е. возвращать TRUE или 1).

Направление перехода от одного шага к другому указывает линия со стрелкой. Линия, соединяющая SFC-условие с линией перехода между шагами, указывает, на каком переходе действует данное условие.

SFC-программа может выступать в роли основной программы и функции-блока. Запрограммировать функцию на языке **Техно SFC** нельзя.

2.20.3 Техно FBD

FBD-программа представляет собой цепочку (диаграмму) последовательно выполняемых **функциональных блоков**.

Функциональный блок – это графическое изображение вызова встроенной функции **Техно FBD** (FBD-блока) или функции (функции-блока), определенной пользователем.

В верхней части блока выводится обозначение функции, выполняемой блоком. Именованные отрезки слева, обозначают входы блока (аргументы, переменные или константы функции). Отрезок без имени слева обозначает вход, управляющий выполнением блока (в дальнейшем – вход **RUN**). Блок выполняется, если **RUN**=0 (значение по умолчанию).

Отрезки, примыкающие к блоку справа, обозначают выходы блока (возвращаемые функцией значения).

Кроме входов/выходов, некоторые встроенные FBD-блоки имеют внутренние переменные, недоступные пользователю. Переменные FBD-блока (входы/выходы и внутренние) являются глобальными, т.е. сохраняют свое значение между вызовами программы, в том числе при **RUN**=1.

В нижней части блока выводится его номер и, после двоеточия, номер следующего выполняемого блока. Номера блоков задаются последовательно при их размещении в рабочем поле редактора; номера следующих выполняемых блоков определяются автоматически при соединении входов и выходов блоков (образовании диаграммы). На блоке, который выполняется первым в программе, после его номера отображается символ **B**; на блоке, который выполняется последним, – символ **E**.

FBD-программа может выступать в роли основной программы, функции и функции-блока.

2.20.4 Техно LD

LD-программа представляет собой диаграмму последовательно выполняемых **функциональных блоков**.

Функциональный блок – это графическое изображение вызова встроенной функции **Техно LD** (LD-блока), функции (функции-блока), определенной пользователем, или FBD-блока.

Над блоком выводится имя **связанной переменной**.

Связанной переменной называется переменная, от значения которой зависит выполняемое блоком действие или значение которой устанавливается в процессе выполняемого блоком действия. Связанная переменная задается пользователем.

Если связанная переменная не задана, над блоком отображаются три звездочки:

В качестве изображения блока используется обозначение выполняемой этим блоком функции. Отрезок слева обозначает вход блока, отрезок справа – выход. Все LD-блоки имеют один вход (**in**) и один выход (**out**).

Под блоком выводится его номер и, после двоеточия, номер следующего выполняемого блока. Номера блоков задаются последовательно при их размещении в рабочем поле редактора; номера следующих выполняемых блоков определяются автоматически при размещении других блоков и соединении входов и выходов блоков (образовании диаграммы). На блоке, который выполняется первым в программе, после его номера отображается символ **B**; на блоке, который выполняется последним, – символ **E:**

Используемые в программе FBD-блоки, а также функции и функции-блоки отображаются на LD-диаграмме в виде, аналогичном виду функциональных блоков в FBD-редакторе.

Шины изображаются на диаграмме в виде вертикальных линий. В **Техно LD** используются две **основные** шины (левая и правая) и **вспомогательные** шины. Между основными шинами размещаются все функциональные блоки LD-программы; на вспомогательные шины могут замыкаться выходы блоков, расположенных один над другим.

Шины имеют следующее назначение:

значение левой основной шины всегда равно 1 (аналог положительной шины питания);

значение правой основной шины и вспомогательной шины формируется как логическая сумма (**OR**) значений выходов блоков, связанных с этой шиной

В процессе выполнения программы блоки пересчитываются последовательно в соответствии с их номерами. Значение правой основной шины и вспомогательной шины равно логической сумме значений выходов блоков, пересчитанных на текущий момент времени выполнения программы.

LD-программа может выступать в роли основной программы, функции и функции-блока.

2.20.5 Техно IL

Программа на языке **Техно IL** представляет собой последовательность **инструкций**. Каждая инструкция должна начинаться с новой строки и должна содержать **оператор** с опциональным **модификатором** и, для некоторых операций, один или более **операндов**, разделенных пробелами. Между инструкциями могут располагаться пустые строки. Компилятор не чувствителен к регистру, т.е. инструкции **add var_002** и **ADD VAR_002** равнозначны.

Под **аккумулятором** в **Техно IL** понимается хранилище текущего результата вычислений (в этом качестве выступает один из регистров процессора). Далее в описании языка **Техно IL** значение аккумулятора обозначается словом **result**. Функция на языке **Техно IL** возвращает **result**.

Техно IL поддерживает одноадресный и двухадресный режимы записи инструкций, которые оперируют с двумя операндами. В первом случае первым операндом является аккумулятор, который опускается при записи, во втором случае указываются два операнда.

В IL-программе могут использоваться метки и комментарии. Правила их задания аналогичны правилам **Техно ST.**

3 ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ

3.1 ЛАБОРАТОРНАЯ РАБОТА №1 «СОЗДАНИЕ ПРОСТЕЙШЕГО ПРОЕКТА»

Цель работы – освоить методику создания системы мониторинга, содержащую один узел АРМ с использованием механизма автопостроения каналов TRACE MODE методом «от шаблона экрана».

3.1.1 Создание узла АРМ

Разработка любого проекта автоматизации всегда начинается с запуска Интегрированной среды разработки (ИСР). Для ее запуска необходимо выполнить команду TRACE MODE IDE 6 (base) из группы установки инструментальной системы в меню Программы WINDOWS или двойным щелчком ЛК мыши по иконке  рабочего стола Windows рисунок 11.

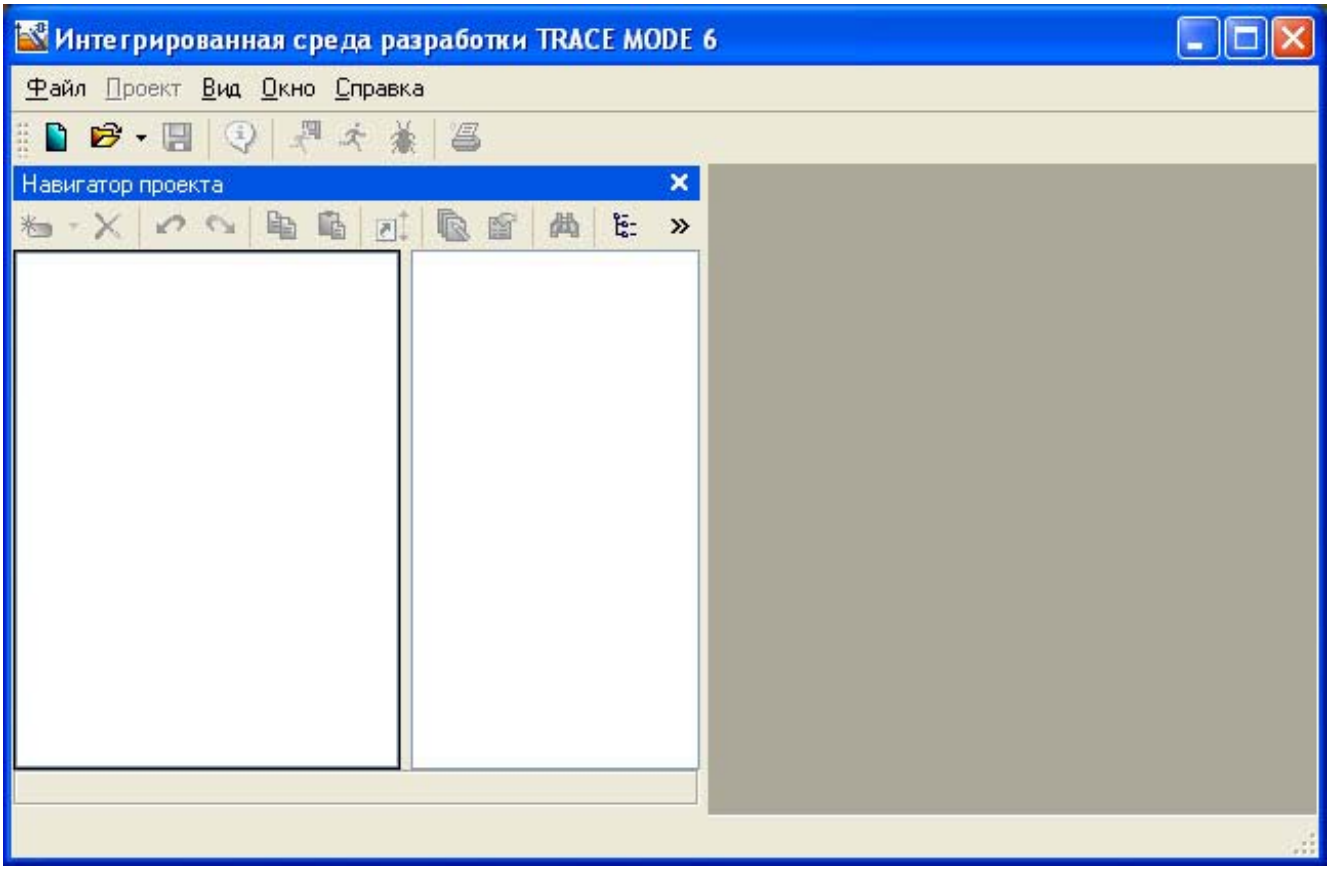


Рисунок 11– Интегрированная среда разработки

После запуска ИСР в меню **Файл** выбрать команду **Настройки ИС…**В появившемся окне выбрать **Уровень сложности** и настроить как показано на рисунке 12, а затем выбрать **Отладка** и настроить как на рисунке 13.

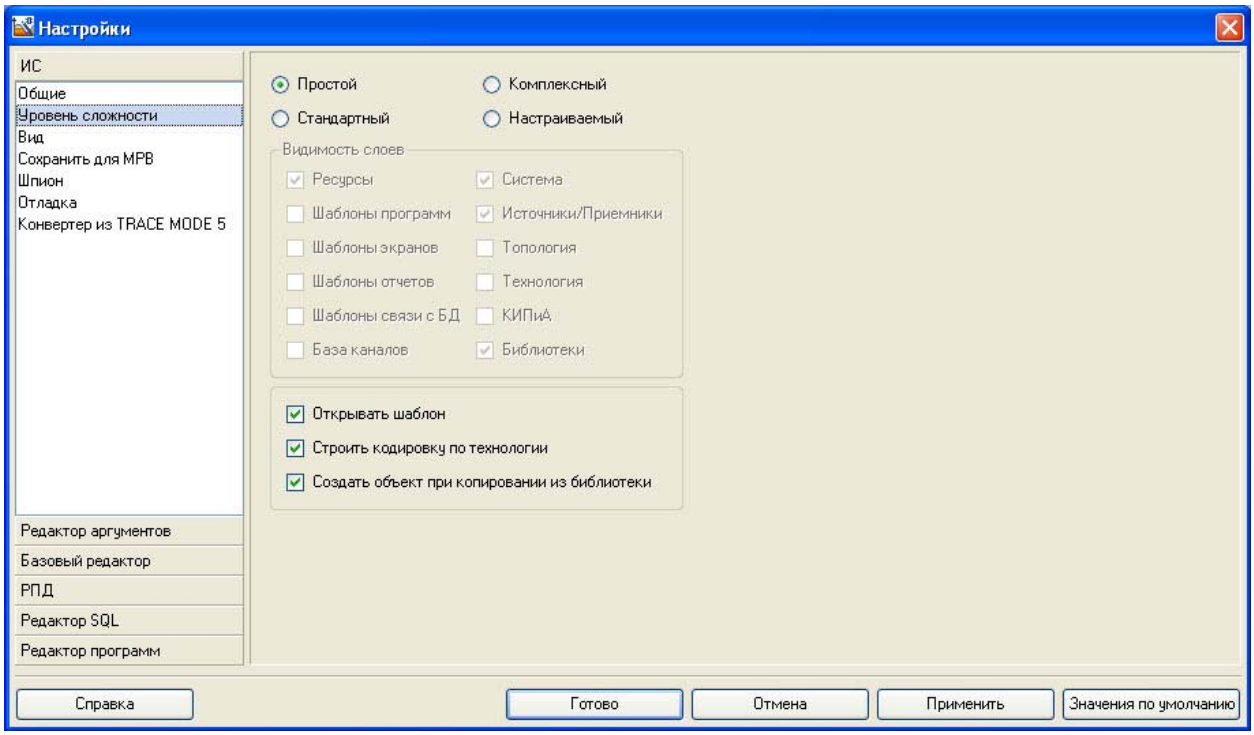


Рисунок 12 – Настройки ИСР

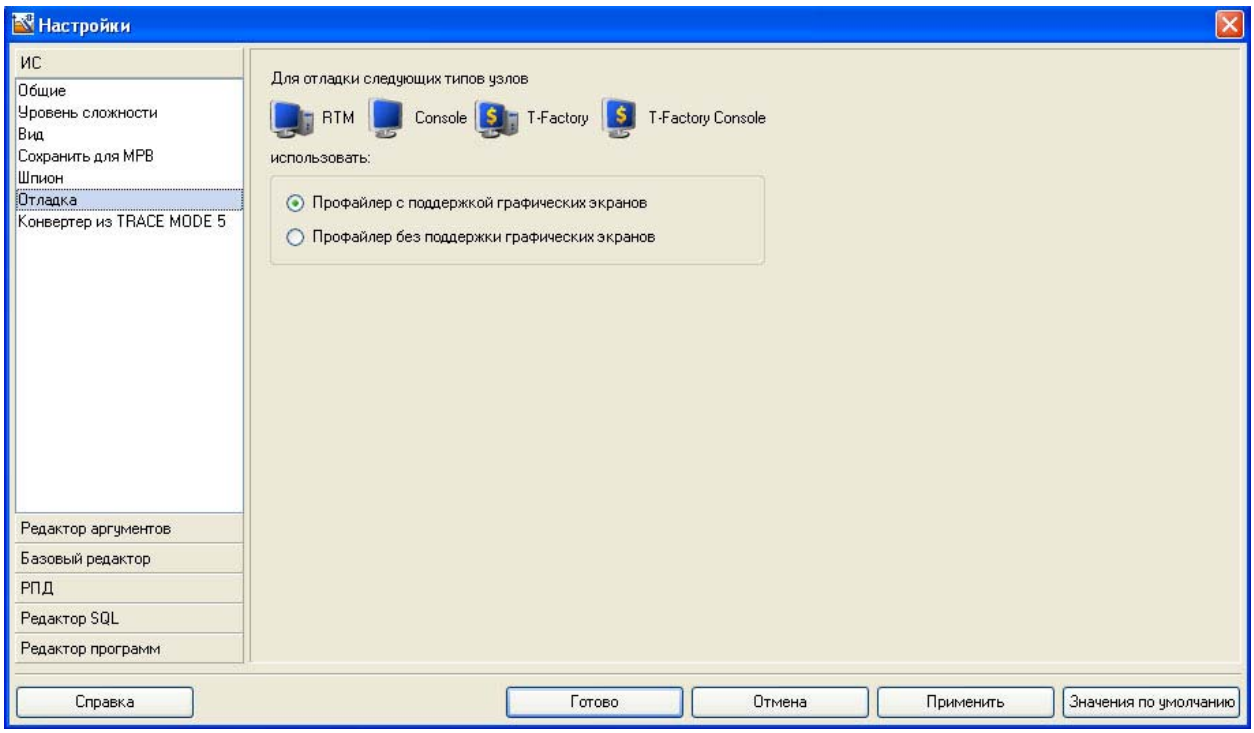


Рисунок 13 – Настройки ИСР

После проведенных настроек ИСР нажать кнопку Готово.

С помощью иконки  инструментальной панели создать новый проект, при этом в открывшемся на экране диалоге рисунок 14.

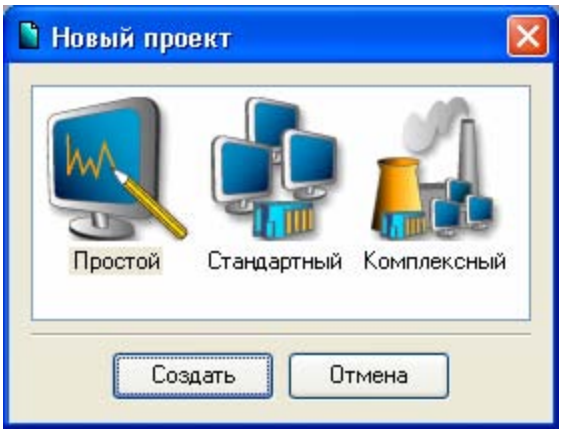


Рисунок 14 – Создание нового проекта

Выберем **Простой** стиль разработки. После нажатия левой клавиши мыши (ЛК) на экранной кнопке **Создать**, в левом окне Навигатора проекта появится дерево проекта с созданным узлом АРМ RTM_1 рисунок 15. Откроем узел RTM_1 двойным щелчком ЛК, в правом окне Навигатора проекта отобразится содержимое узла – пустая группа Каналы и один канал класса Вызов Экран#1, предназначенный для отображения на узле АРМ графического экрана;

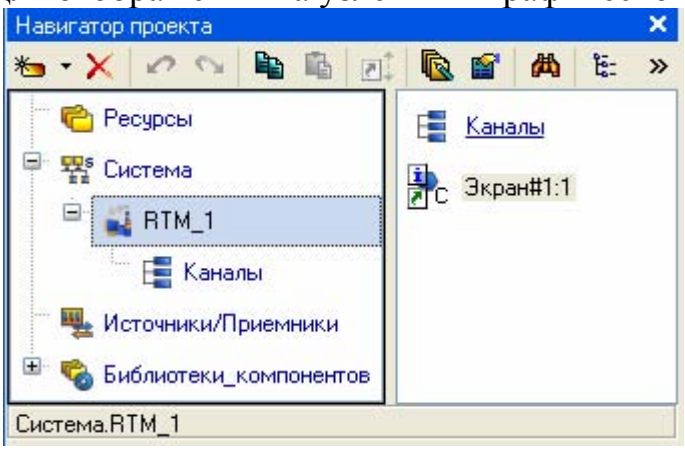


Рисунок 15 – Навигатор проекта

3.1.2 Создание графического экрана

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

Создание статического текста

Разместим в левом верхнем углу экрана статический текст - надпись «Значение параметра». На панели инструментов графического редактора выделить иконку графического элемента (ГЭ)  , на поле редактора установить прямоугольник ГЭ, для чего рисунок 16:

зафиксировать ЛК «точку привязки»;

развернуть прямоугольник движением курсора и

зафиксировать выбранный ГЭ;

Рисунок 16 – Размещение статического текста

Для перехода в режим редактирования элемента выделить на панели инструментов иконку  . Двойным щелчком ЛК по размещенному ГЭ открыть окно его свойств. В правом поле строки **Текст** набрать «Значение параметра» рисунок 17.

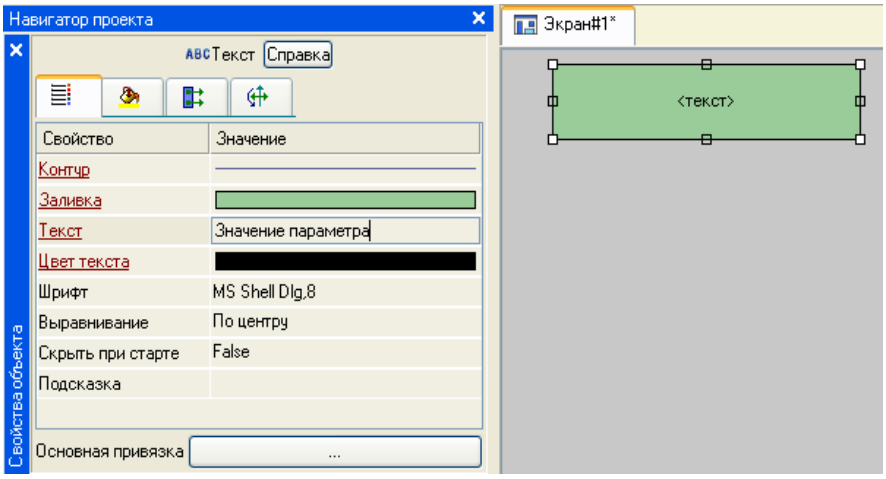


Рисунок 17 – Свойства статического текста

Закрыть окно свойств, ГЭ будет иметь вид рисунок 18:

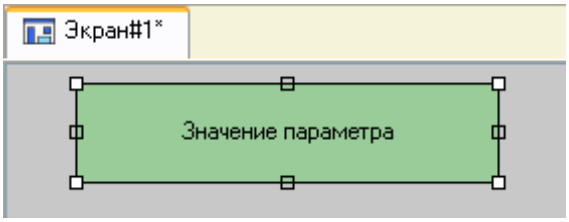


Рисунок 18 – Вид графического экрана

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

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

Подготовим на экране вывод динамического текста для отображения численного значения какого-либо источника сигнала – внешнего или внутреннего путем указания динамизации атрибута ГЭ. Определим назначение аргумента шаблона экрана. Создать и разместить новый ГЭ  справа от ГЭ с

надписью «Значение параметра». Двойным щелком ЛК на строке **Текст** вызвать меню **Вид индикации** рисунок 19.

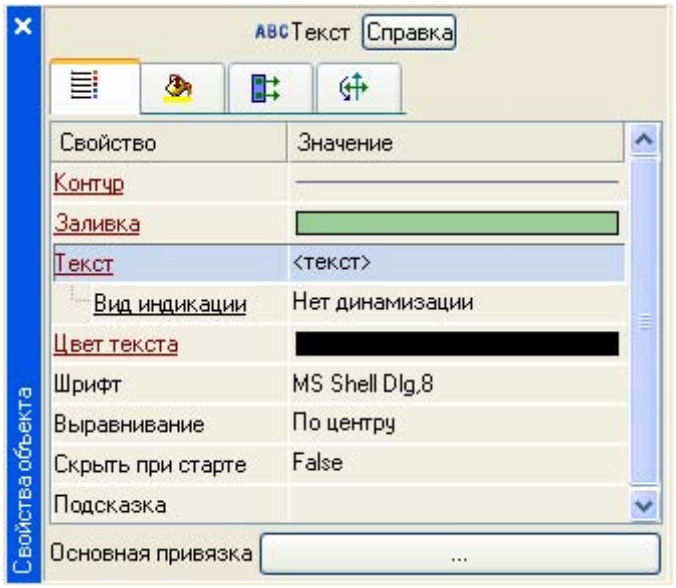


Рисунок 11 – Настройка динамизации9

В правом поле строки нажать ЛК и вызвать список доступных типов, выбрать тип **Значение** рисунок 20.

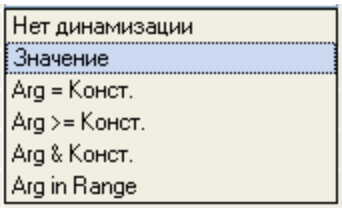


Рисунок 20 – Тип динамизации

В открывшемся меню настройки параметров динамизации рисунок 21 выбрать свойство **Привязка**.

Рисунок 21 – Настройка параметров динамизации

В открывшемся окне **Свойство привязки**, нажав кнопку  на его панели инструментов, создать аргумент экрана рисунок 22.

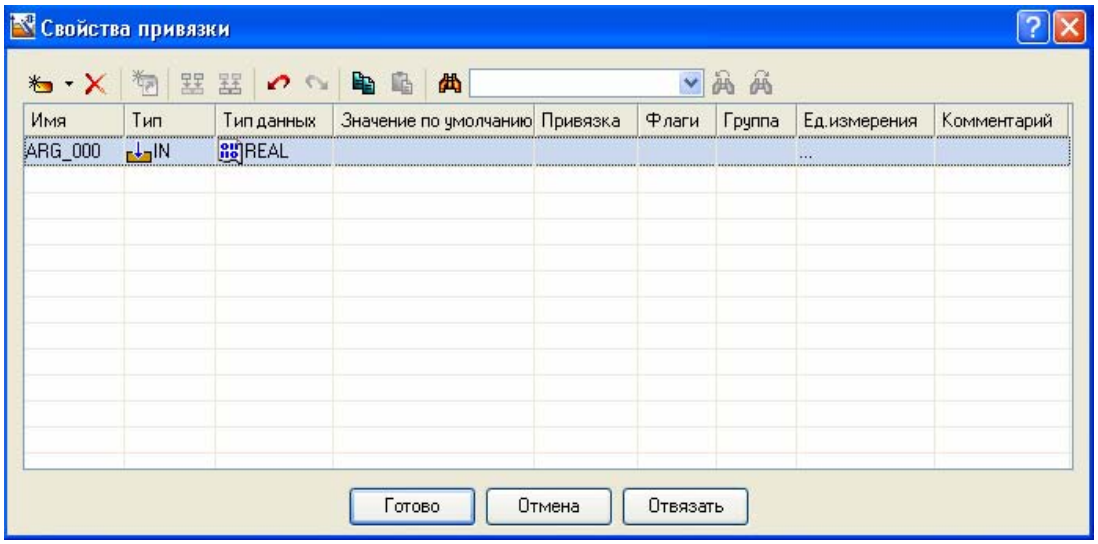


Рисунок 22 – Создание аргумента экрана

Двойным щелчком ЛК выделить имя аргумента и изменить его, введя с клавиатуры «Параметр» (завершить ввод нажатием клавиши Enter). Подтвердить связь с этим аргументом нажатием кнопки **Готово**. Закрыть окно свойств ГЭ, графический экран будет иметь следующий вид рисунок 23.

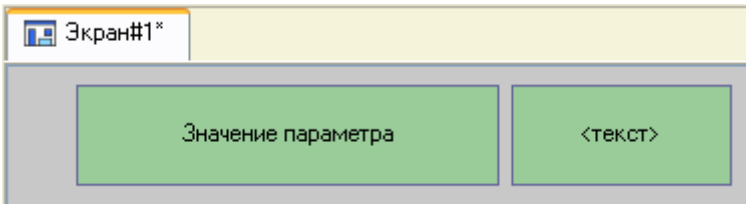


Рисунок 23 – Вид графического экрана

3.1.4 Создание стрелочного прибора, привязка к тому же аргументу

Применим для отображения параметра новый тип ГЭ – Стрелочный прибор рисунок 23. Выделить двойным щелчком ЛК на инструментальной панели редактора графики иконку  и выбрать в появившемся меню иконку стрелочного прибора  . Установить ГЭ Стрелочный прибор, выбрав его размер таким, чтобы все элементы графики и текста на нем было разборчивы и симметричны. Перейти в режим редактирования и открыть окно свойств ГЭ Стрелочный прибор. Щелчком ЛК на кнопке **Основная привязка** открыть окно табличного редактора аргументов. ЛК выбрать аргумент шаблона экрана Параметр. Подтвердить выбор ЛК на кнопке **Готово**. Двойным щелчком ЛК открыть свойство **Заголовок** и в строке **Текст** ввести слово «Параметр». Закрыть окно свойств.

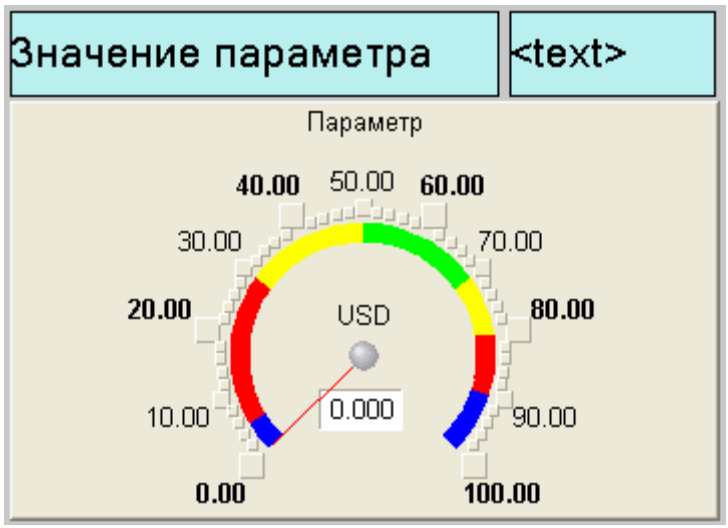


Рисунок 23 – Стрелочный прибор

3.1.5 Автопостроение канала

Для создания канала в узле проекта по аргументу шаблона экрана воспользуемся процедурой автопостроения. В слое **Система** открыть узел RTM_1. С помощью ПК вызвать контекстное меню и ЛК открыть свойства компонента Экран#1 рисунок 24.

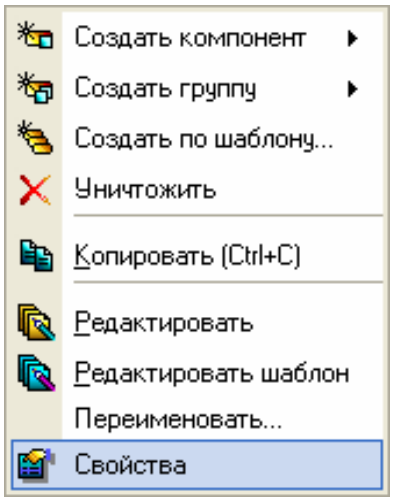


Рисунок 24 – Выбор свойств экрана

Выбрать вкладку **Аргументы**, выделить ЛК аргумент Параметр и с помощью иконки  создать канал класса **Float** типа **Input** с именем **Параметр**.

Создание генератора синуса и привязка его к каналу

Введем в состав проекта источник сигнала – внутренний генератор синусоиды, свяжем его с созданным каналом и опробуем выполненные средства отображения.

Открыть слой **Источники/Приемники** и через ПК создать в нем группу **Генераторы** рисунок 25.

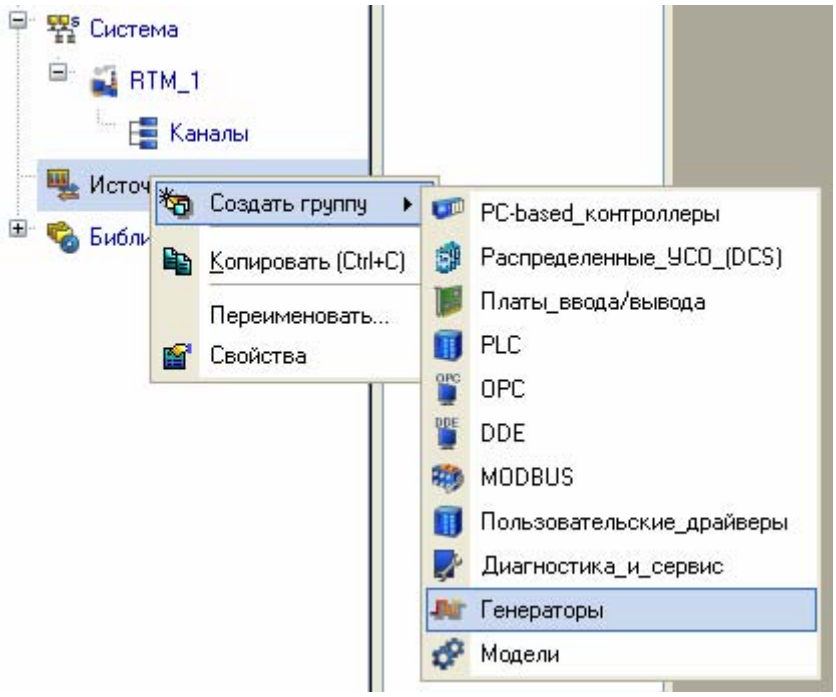


Рисунок 25 – Добавление группы Генераторы

Двойным щелчком ЛК открыть группу **Генераторы** и через ПК создать в ней компонент **Синусоида** рисунок 26.

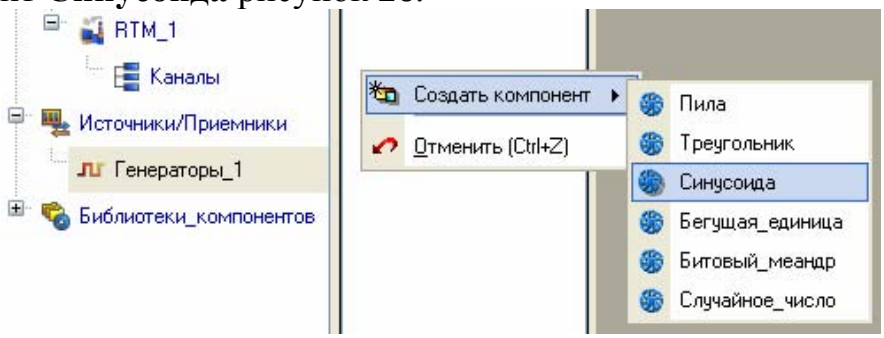


Рисунок 26 – Создание компонента Синусоида

Захватить с помощью ЛК созданный источник и, не отпуская ЛК, перетащить курсор на узел RTM_1 в слое **Система**, а затем, в открывшемся окне компонентов RTM_1, на канал **Параметр** рисунок 27. Отпустить ЛК.

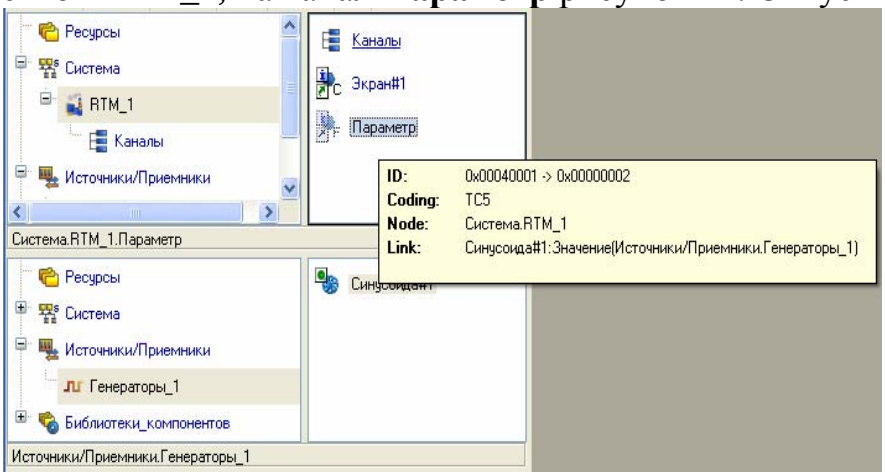


Рисунок 27 – Связь компонента Синусоида с каналом Параметр

Запуск проекта

Закрыть окно графического редактора. Сохранить проект с помощью иконки . На инструментальной панели выбрать команду и скомпилировать проект для запуска в реальном времени. ЛК выделить в слое **Система** узел RTM_1, выбрать иконку на инструментальной панели и запустить режим исполнения. В открывшемся окне ГЭ справа от надписи «Значение параметра» должно показываться изменение синусоидального сигнала. То же значение должен отображать и стрелочный прибор рисунок 28.

Рисунок 28 – Результат имитационного запуска проекта

3.1.6 Добавление функции управления

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

Вызвать графический экран на редактирование. Выбрать на инструментальной панели графического редактора иконку ГЭ Кнопка - . С помощью мыши разместить его в поле экрана под ГЭ Стрелочный прибор рисунок 29.

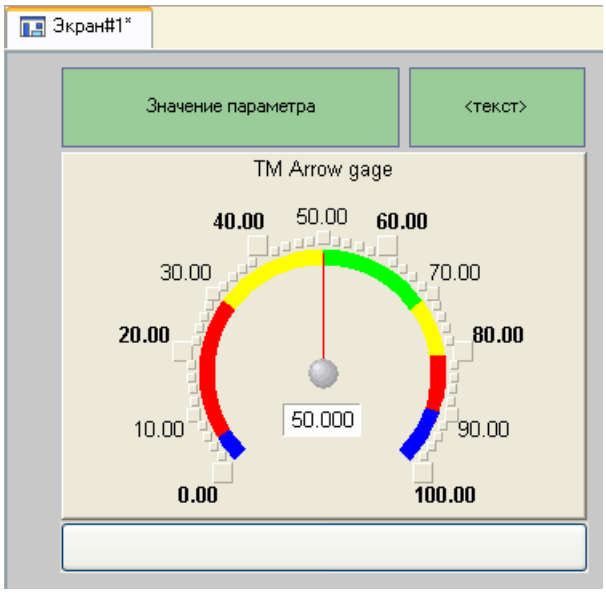


Рисунок 29 – Вид графического экрана

Перейти в режим редактирования , вызвать окно свойств ГЭ рисунок 30.

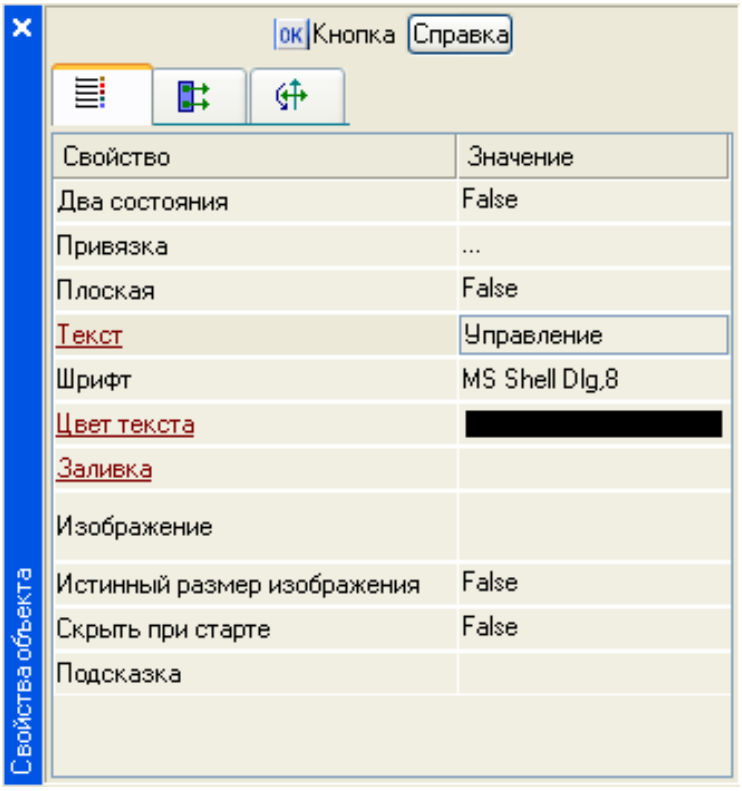


Рисунок 30 – Свойства графического элемента кнопка

В поле **Текст** ввести «Управление». Открыть бланк **События** и ПК раскрыть меню **По нажатию (pressed)**. Выбрать из списка команду **Добавить Send Value**, раскрыть меню настроек выбранной команды рисунок 31.

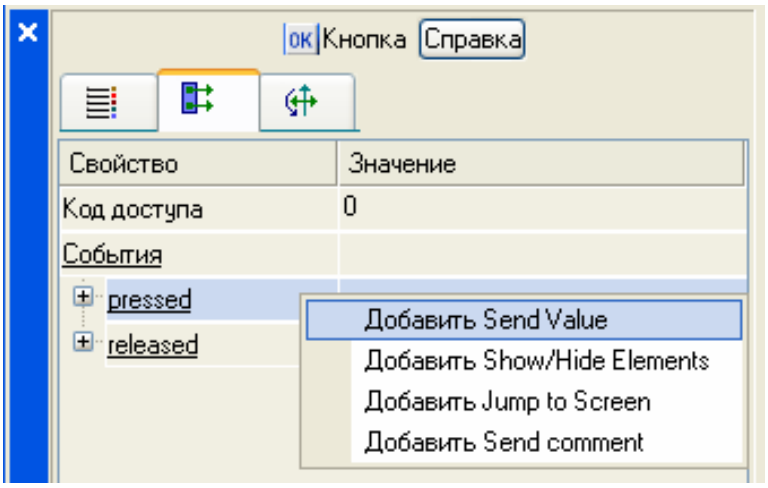


Рисунок 31 – Настройка графического элемента Кнопка

В поле **Тип передачи (Send Type)** выбрать из списка **Ввести и передать (Enter & Send)** рисунок 32.

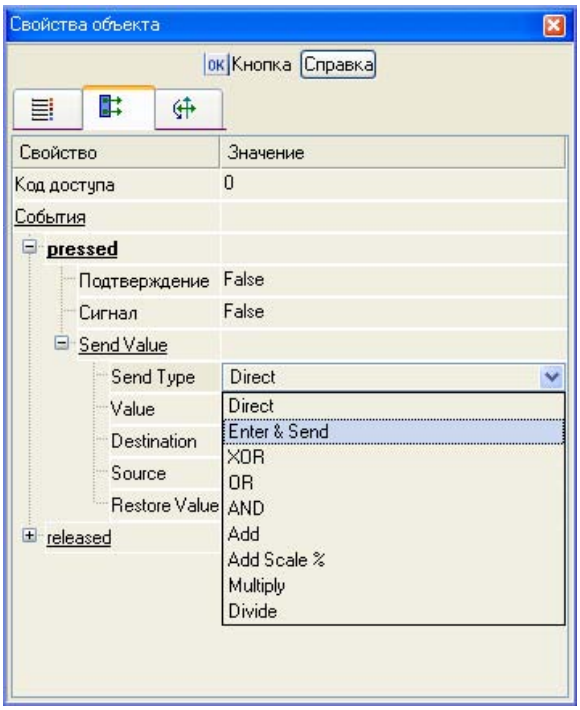


Рисунок 32 – Настройка графического элемента Кнопка

ЛК в поле **Результат** вызвать табличный редактор аргументов. Создать еще один аргумент и задать ему имя **Управление**. Изменить тип аргумента на **IN/OUT**, кнопкой Готово подтвердить привязку атрибута ГЭ к этому аргументу рисунок 33.

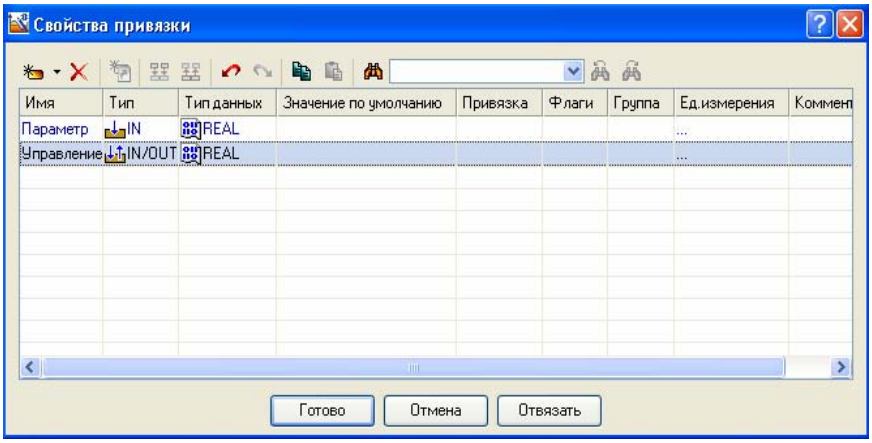


Рисунок 33 – Создание аргумента Управление

Закрыть окно свойств ГЭ с помощью щелчка ЛК по иконке  . Выделить ЛК ГЭ Текст, служащий для отображения значения Параметра рисунок 34

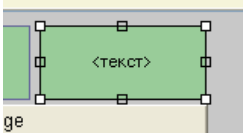


Рисунок 34 – Графический элемент текст

С помощью иконки  на панели инструментов или комбинацией клавиш **Ctrl+C** скопировать выделенный ГЭ Текст в буфер обмена. Далее с помощью иконки  или комбинацией клавиш **Ctrl+V** извлечь копию ГЭ из буфера обмена и поместить ее на графический экран. Переместить, удерживая нажатой ЛК, копию ГЭ **Текст** справа от размещенного на экране ГЭ Кнопка. Двойным

щелчком ЛК на перемещенном ГЭ Текст открыть окно его свойств рисунок 35. Двойным щелчком ЛК на строке Текст вкладки основных свойств перейти к настройке динамизации данного атрибута ГЭ. В правом поле строки **Привязка** щелчком ЛК открыть табличный редактор аргументов шаблона экрана. Выделить ЛК в списке аргумент **Управление** и щелчком ЛК по экранной кнопке Готово подтвердить привязку атрибута ГЭ Текст к данному аргументу шаблона экрана. Закрыть окно свойств ГЭ Текст.

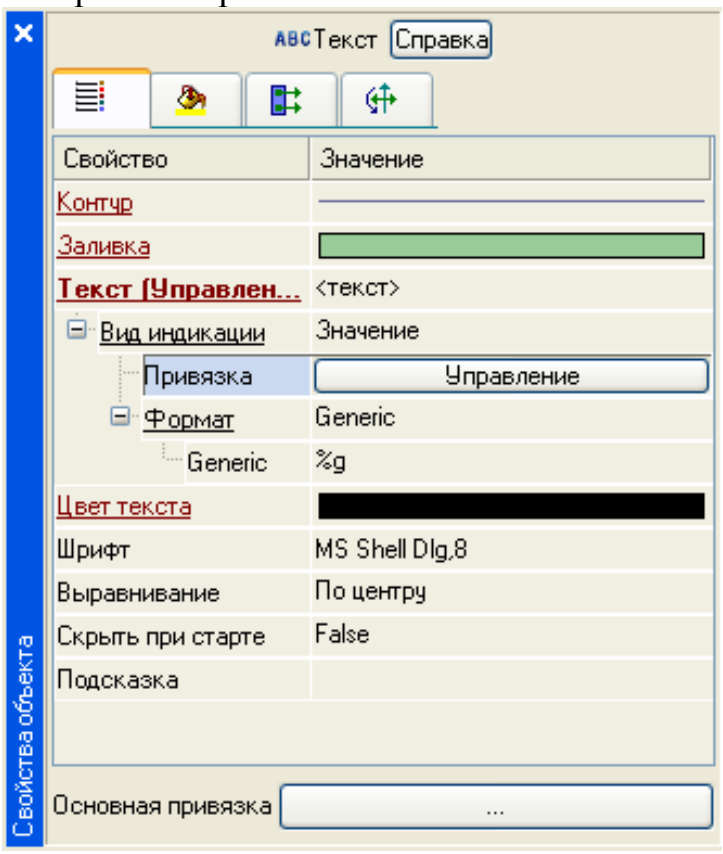


Рисунок 35 – Свойства графического элемента Текст

Закрыть окно графического редактора.

3.1.7 Привязка аргумента экрана к каналу

Создадим по аргументу Управление шаблона экрана новый канал, отредактируем его привязку. В слое **Система** открыть узел RTM_1. С помощью ПК вызвать через контекстное меню свойства компонента Экран#1 рисунок 36.

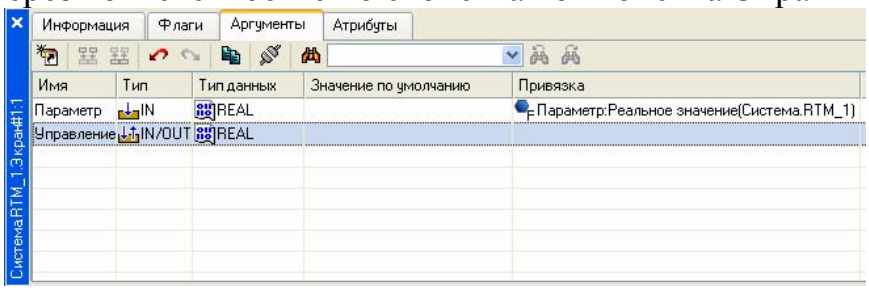


Рисунок 36 – Окно свойств экрана

Выбрать вкладку **Аргументы**, ЛК выделить аргумент **Управление** и с помощью иконки создать новый канал. В результате, в узле RTM_1 ,будет автопостроен канал с именем **Управление** рисунок 37.

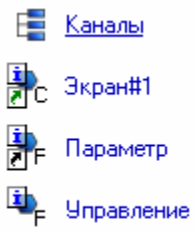


Рисунок 37 – Автопостроенный канал Управление

Двойным щелчком в поле **Привязка** аргумента **Управление** вызвать окно настройки связи, выбрать в нем атрибут **Входное значение** канала **Управление** и кнопкой **Привязка** подтвердить связь аргумента экрана **Управление** с атрибутом **Входное значение** канала **Управление** рисунок 38.

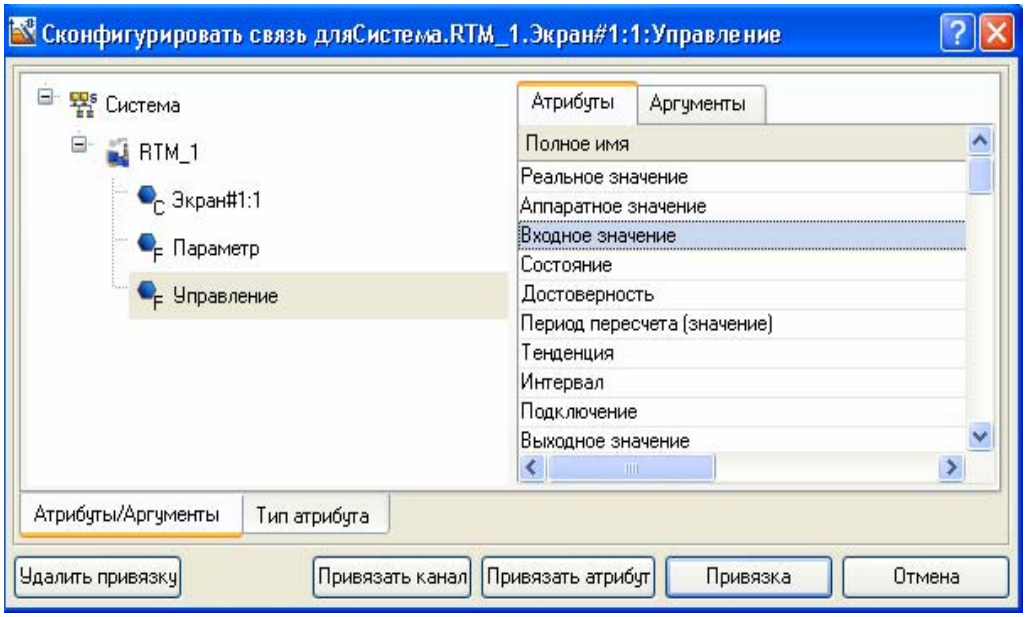


Рисунок 38 – Конфигурация системы

Закрыть окно свойств компонента Экран#1.

3.1.8 Размещение ГЭ Тренд

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

В правой части графического экрана разместим ГЭ Тренд для вывода значений **Параметр** и **Управление**. Основные свойства ГЭ оставим заданными по умолчанию. Перейдем во вкладку и, выделив ЛК строку **Кривые**, с помощью ПК создадим две новые кривые. Настроим их привязки к аргументам, толщину и цвет линий рисунок 39.

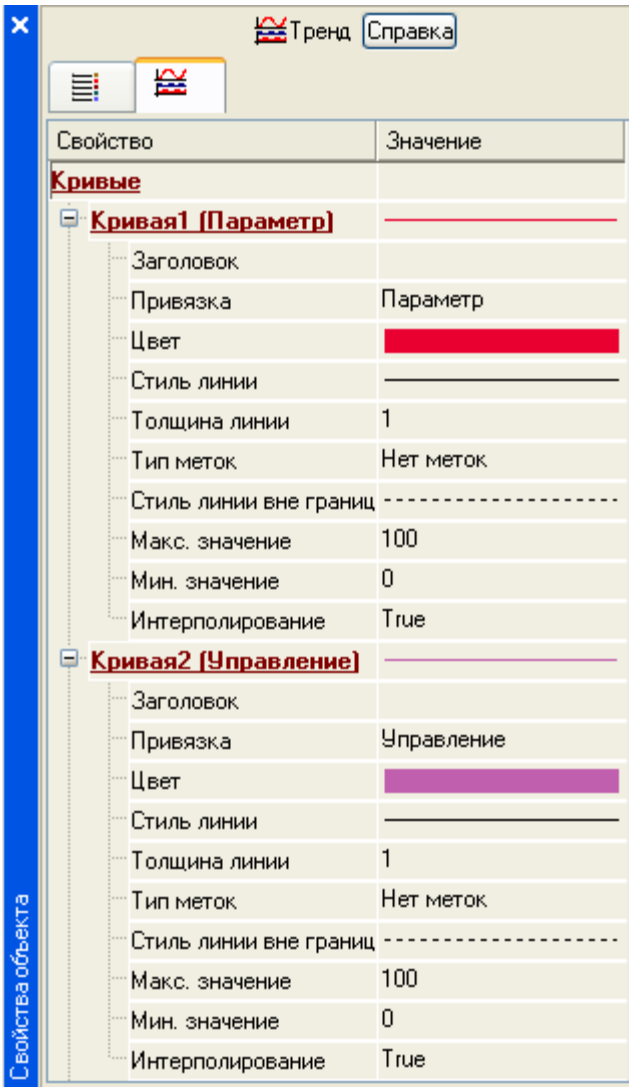


Рисунок 39 – Свойства графического элемента Тренд

ГЭ примет вид рисунок 40.

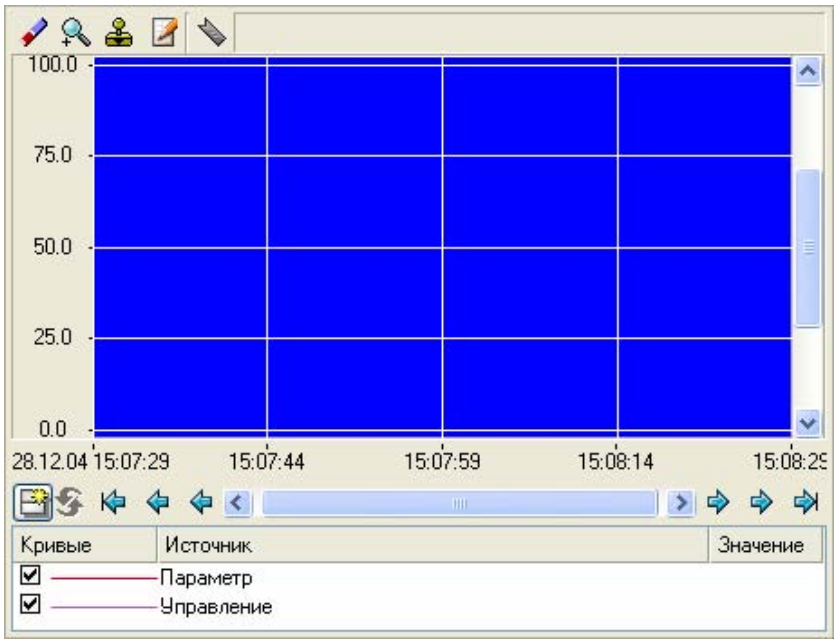


Рисунок 40 – Вид графического элемента Тренд

3.1.9 Запуск проекта

Сохранить проект с помощью иконки . На инструментальной панели выбрать команду и скомпилировать тем самым проект для запуска в реальном времени. Выбрать иконку на инструментальной панели и запустить режим исполнения. С помощью кнопки «Управление» ввести величину «управляющего воздействия» и наблюдать результат на соседнем индикаторе и тренде.

3.1.10 Простейшая обработка данных

С помощью создания нового компонента проекта – шаблона программы свяжем операцией сложения два имеющихся канала. Будем суммировать реальные значения каналов Параметр и Управление, а результат помещать во вновь созданный аргумент экрана Сумма (с отображением на ГЭ Текст и Тренд) без создания дополнительного канала в узле проекта.

Скопировать два первых ГЭ – «Значение параметра» и «text» и разместить их ниже ГЭ Кнопка рисунок 41.

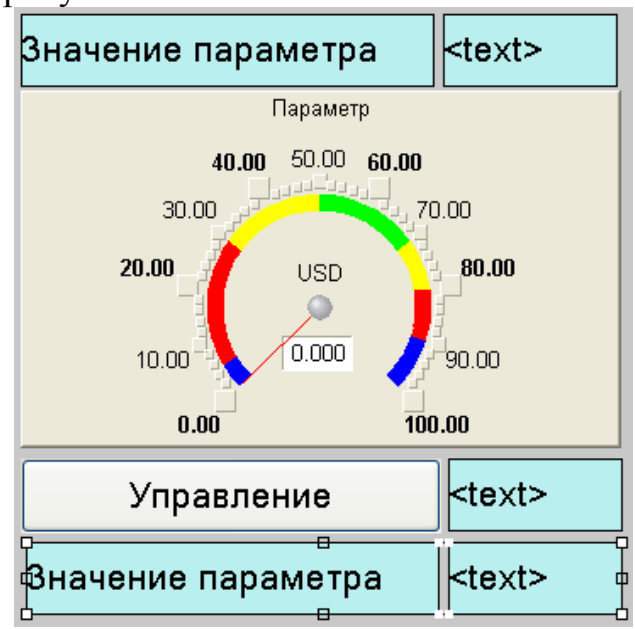


Рисунок 41 – Вид графического экрана

Изменить статический текст первого ГЭ на «Сумма :» рисунок 42.

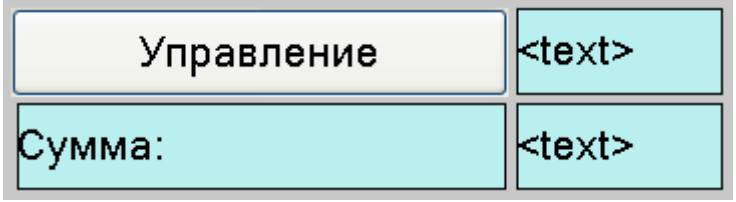


Рисунок 42 – Вид графического экрана

А динамику второго ГЭ привязать к третьему аргументу экрана типа **IN** с именем **Сумма**, который создать в процессе привязки рисунок 43.

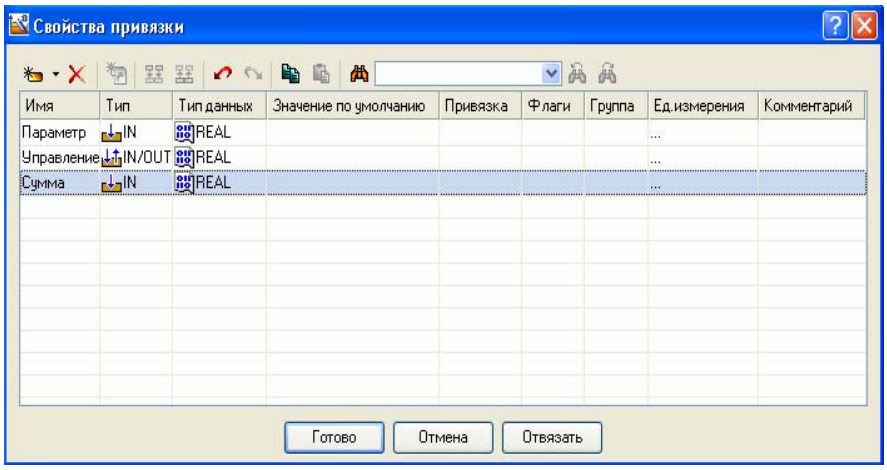


Рисунок 43 – Свойства привязки

Добавить еще одну кривую на тренд с привязкой к аргументу **Сумма** рисунок 44.

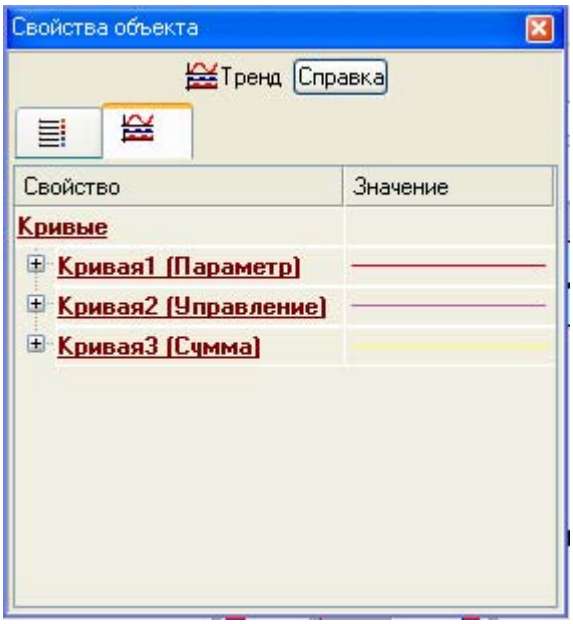


Рисунок 44 – Свойства графического элемента Тренд

3.1.11 Создание программы на языке Техно ST

Создадим программу, в которой сумма двух аргументов, связанных с атрибутами **Реальное значение** каналов Параметр и Управление, будет помещается в третий аргумент с именем Сумма. В дальнейшем, воспользуемся возможностью связывания аргументов шаблонов для вывода на экран результата работы программы без создания дополнительного канала.

Двойным щелчком ЛК открыть узел RTM_1 и создать в нем компонент **Программа** рисунок 45.

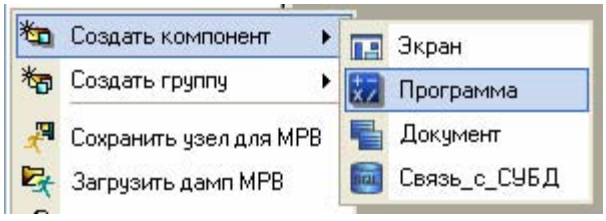


Рисунок 45 – Создание компонента программа

выделить компонент Программа#1 и ПК вызвать контекстное меню, выбрав в котором ЛК пункт **Редактировать шаблон**, перейти в режим редактирования программы рисунок 46.

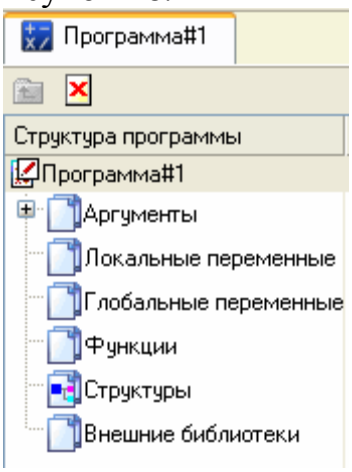


Рисунок 46 – Режим редактирования программы

Выделением ЛК в дереве шаблона Программа#1 строки **Аргументы** вызвать табличный редактор аргументов. Иконкой создать в редакторе аргументов три аргумента с именами Параметр, Управление и Сумма. При этом первые 2 аргумента должны быть типа **IN**, а третий – **OUT** рисунок 47.

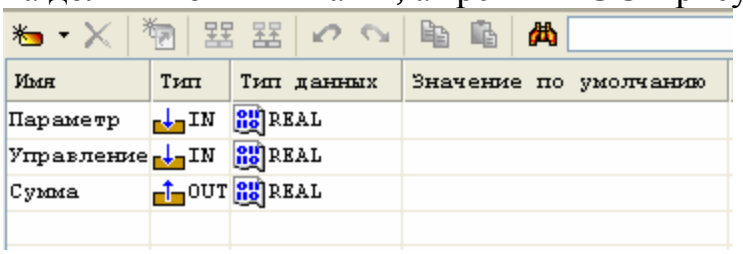


Рисунок 47 – Создание каналов программы

Выделить в дереве шаблона строку Программа#1 и в открывшемся диалоге **Выбор языка** выбрать язык ST рисунок 48.

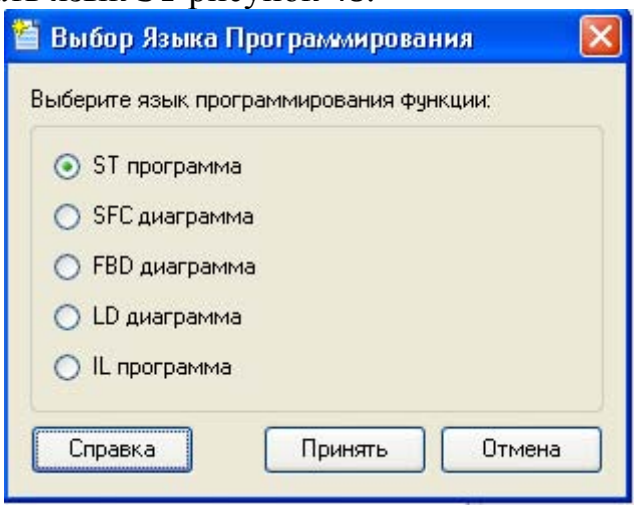


Рисунок 48 – Выбор языка программирования

По нажатию экранной кнопки **Принять** в открывшемся окне редактора программ с объявленными переменными набрать следующую строку рисунок 49.

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

Сумма=Параметр+Управление;

END_PROGRAM
```

Рисунок 49 – Вид программы на языке ST

С помощью иконки  на инструментальной панели редактора или «горячей клавишей» F7 скомпилировать программу и убедиться в успешной компиляции в окне **Выход (Output)**, вызываемого из инструментальной панели с помощью иконки  рисунок 50.

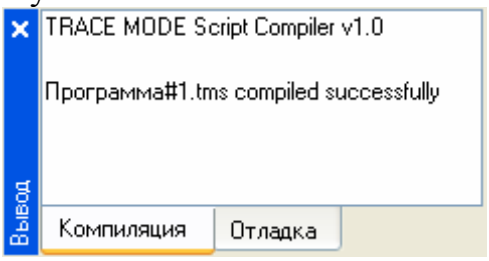


Рисунок 50 – Результат удачной компиляции программы

3.1.12 Привязка аргументов программы

Выполним привязку аргументов программы к атрибутам каналов. Вызвать свойства компонента Программа#1 через контекстное меню. Выбрать вкладку **Аргументы**. Двойным нажатием в поле Привязка привязать аргументы программы к атрибутам каналов – аргумент **Параметр** к реальному значению канала Параметр, аргумент Управление к реальному значению канала **Управление** рисунок 51.

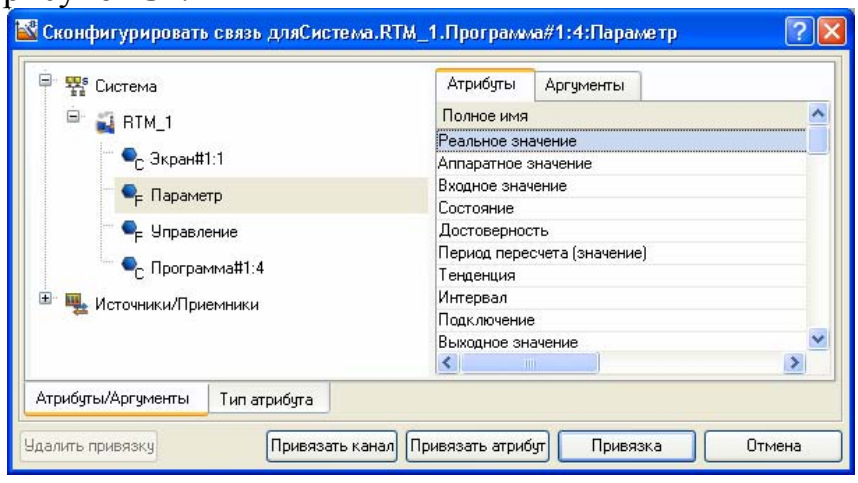


Рисунок 52 – Привязка аргументов программы к атрибутам каналов

Двойным щелчком в поле **Привязка** аргумента программы **Сумма** вызвать окно настройки связи, выбрать в левом окне канал класса **Вызов** Экран#1, а в правом окне выбрать вкладку **Аргументы** и указать в ней аргумент **Сумма** и кнопкой **Привязка** подтвердить связь рисунок 53.

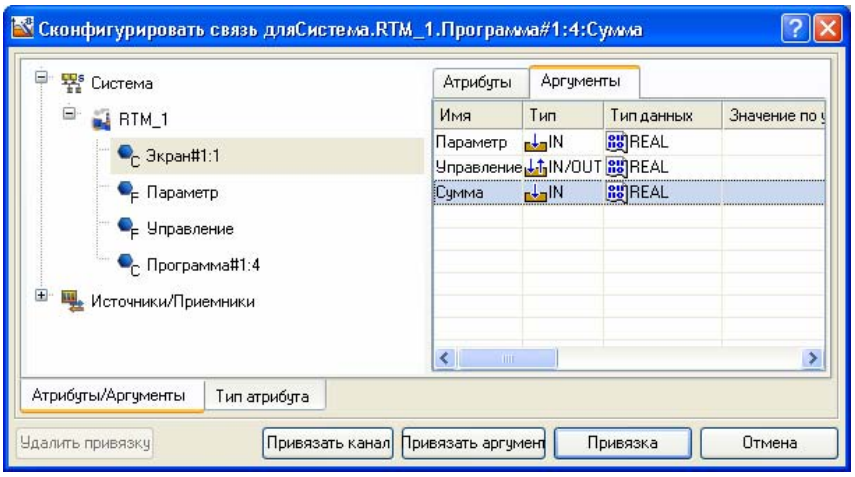


Рисунок 53 – Привязка аргументов программы к атрибутам каналов

В результате, будем иметь рисунок 54.

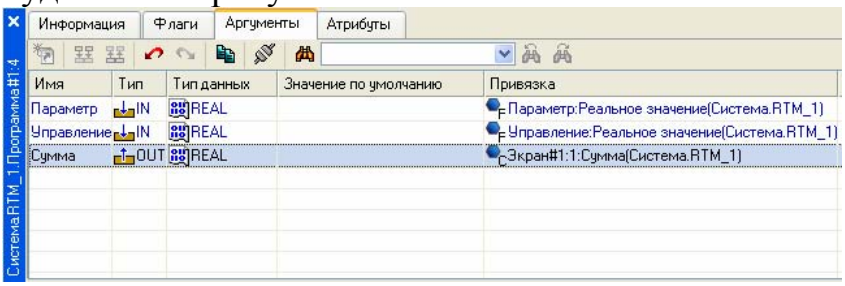


Рисунок 54 – Результаты конфигурации

После закрыть окно свойств компонента Программа#1.

3.1.13 Запуск проекта

Сохранить проект с помощью иконки  . На инструментальной панели выбрать команду  и скомпилировать тем самым проект для запуска в реальном времени. Выбрать иконку  на инструментальной панели и запустить режим исполнения. С помощью кнопки «Управление» ввести «управляющее воздействие» и наблюдать соответствующее изменение сигнала «Управление» и смещение сигнала «Сумма» рисунок 55.

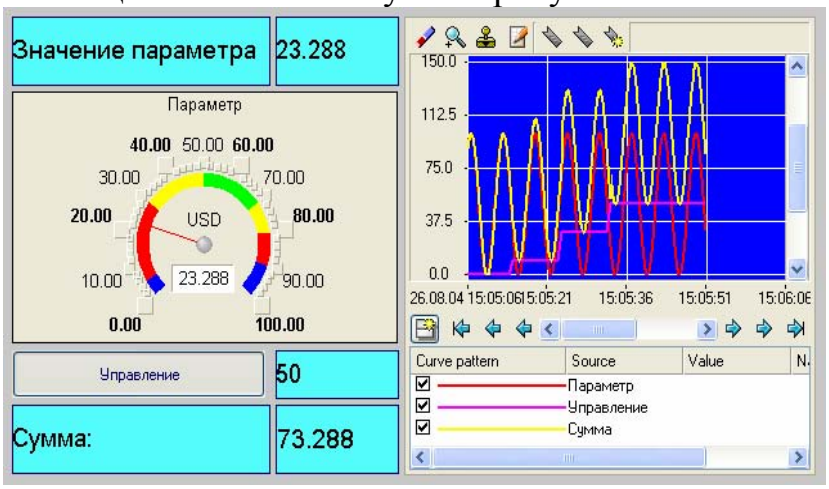


Рисунок 55 – Результат имитационного запуска проекта

3.1.14 Связь по протоколу DDE с приложением MS Windows на примере Excel MPB как DDE – сервер

Организуем запрос реальных значений каналов узла разработанного проекта приложением MS Windows в качестве которого выберем книгу MS Excel.

Запустить MS Excel. Записать в двух ячейках первого столбца запросы на получение данных:

=RTM0|GET!Параметр

=RTM0|GET!Управление

где 0 – индивидуальный номер узла в проекте;

Запустить на исполнение узел АРМ RTM_1. В меню таблицы MS Excel **Правка** выбрать команду **Связи**, выделить все три параметра и нажать кнопку **Обновить**, после чего закрыть окно кнопкой **ОК**. Убедиться, что значения в ячейках книги Excel изменяются вместе с соответствующими реальными значениями каналов узла (значения канала **Параметр** меняется постоянно, а канала **Управление** – после введения нового значения с помощью ГЭ Кнопка) рисунок 56.

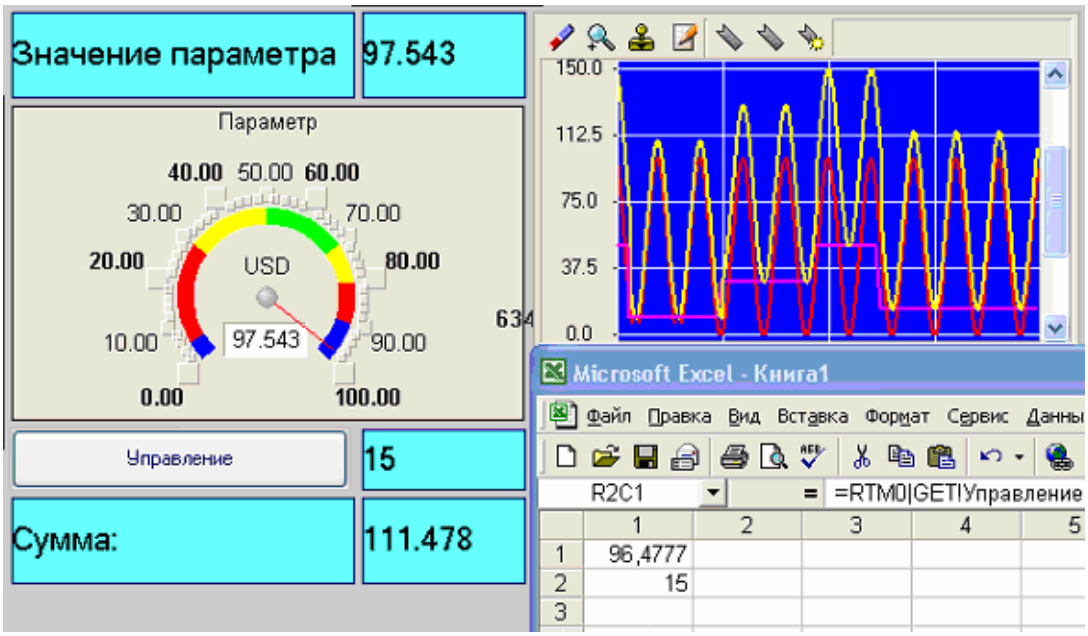


Рисунок 56 – Результат имитационного запуска проекта

MPB как DDE-клиент

В том случае, когда требуется получать данные от внешнего приложения по протоколу DDE, MPB должен выступать в роли DDE-клиента. Например, если необходимо вводить во вновь создаваемый канал (в его атрибут **Входное значение**) Из_таблицы узла RTM_1 данные из ячейки R3C3 книги MS Excel, надо в слое **Источники/Приемники** создать новую группу DDE, а в ней – компонент DDE#1 и отредактировать его следующим образом рисунок 57.

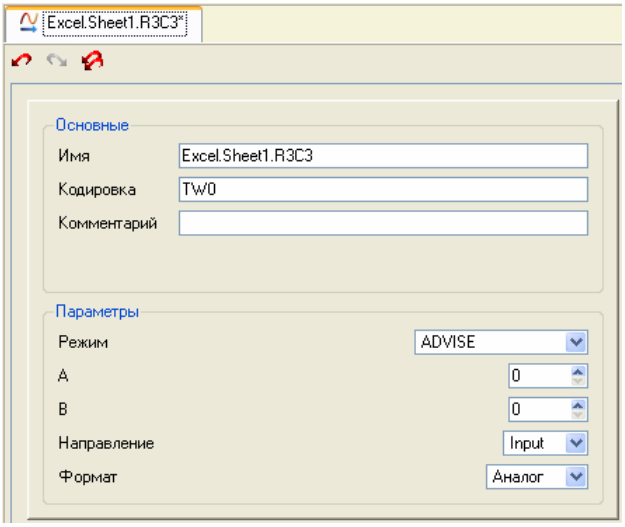


Рисунок 57 – Создание группы DDE#1

где в атрибуте **Имя**:

Excel – имя приложения;

Sheet1 – имя листа книги Excel;

R3C3 – адрес ячейки.

ADVISE – режим посылки клиенту значения при каждом его изменении.

После необходимо создать канал класса **Float** типа **Input** с именем Из_таблицы и привязать к нему с помощью механизма **drag-and-drop** источник Excel.Sheet1.R3C3. После процедур сохранения проекта и подготовки его к запуску в реальном времени, запустим Excel, а затем узел АРМ RTM_1. Вводя в ячейку R3C3 произвольные значения, их можно наблюдать в атрибутах канала Из_таблицы с помощью окна просмотра компонентов, открываемое через основное меню отладчика рисунок 58.

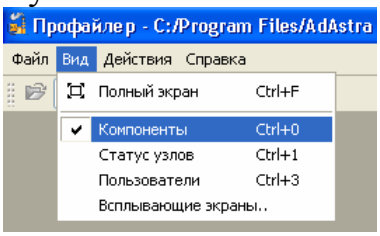


Рисунок 58 – Окно профайлера

Таким образом, в результате будем иметь следующее рисунок 59

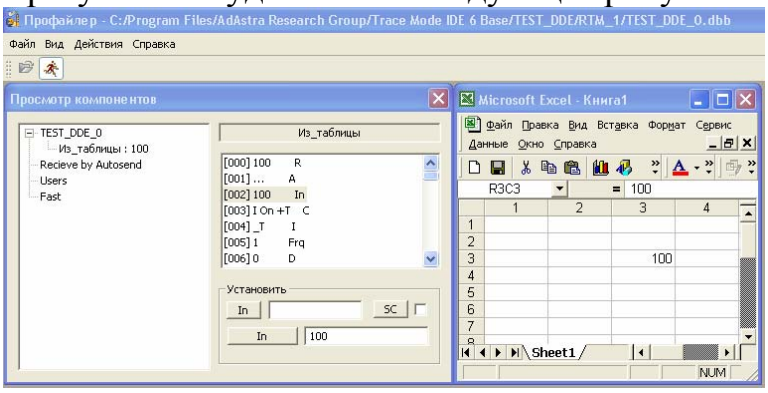


Рисунок 59 – Результат имитационного запуска проекта

Для составления отчета необходимо в меню **Файл** выполнить команду **Документировать проект** полученный *.html файл необходимо распечатать.

3.1.15 Вопросы для защиты лабораторной работы №1

1. Системы мониторинга и управления технологическими процессами
2. Этапы создания систем управления на базае SCADA–систем
3. Функциональные характеристики SCADA-систем
4. Функциональные возможности
5. Программно-аппаратные платформы SCADA-систем
6. Средства сетевой поддержки
7. Встроенные командные языки
8. Поддерживаемые базы данных
9. Графические возможности
10. Тренды и архивы в SCADA-системах
11. Алармы и события в SCADA-системах
12. Эксплуатационные характеристики SCADA-систем
13. Надежность
14. Наличие и качество технической поддержки
15. Оценка стоимости инструментальных систем
16. Открытость систем
17. Технологии OPC
18. Аппаратная реализация связи с устройствами ввода-вывода
19. Технологии Active X
20. Интеграция многоуровневых систем автоматизации
21. Сравнительный анализ и тестирование SCADA-систем
22. SCADA система TRACE MODE
23. Языки программирования в TRACE MODE
24. Техно ST
25. Техно SFC
26. Техно FBD
27. Техно LD
28. Техно IL

3.2 ЛАБОРАТОРНАЯ РАБОТА №2 «РЕАЛИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ ПРИ ПОМОЩИ SCADA–СИСТЕМЫ TRACE MODE»

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

3.2.1 Порядок выполнения лабораторной работы

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

Разработка любого проекта автоматизации всегда начинается с запуска Интегрированной среды разработки (ИСР). Для ее запуска необходимо выполнить команду TRACE MODE IDE 6 (base) из группы установки инструментальной системы в меню Программы WINDOWS или двойным щелчком ЛК мыши по иконке  рабочего стола Windows рисунок 60;

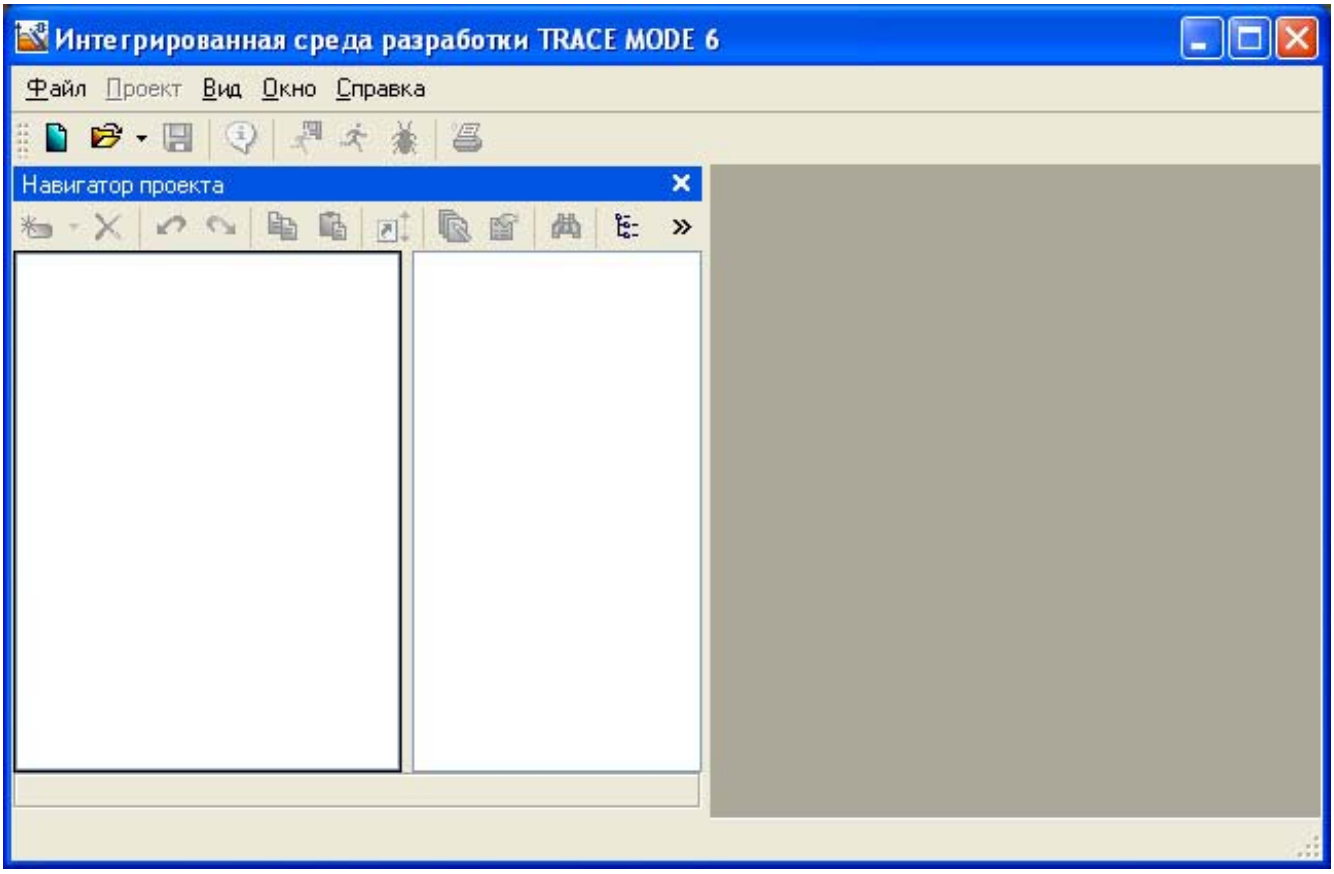


Рисунок 60– Интегрированная среда разработки

После запуска ИСР в меню **Файл** выбрать команду **Настройки ИС…**В появившемся окне выбрать **Уровень сложности** и настроить как показано на рисунке 61, а затем выбрать **Отладка** и настроить как на рисунке 62.

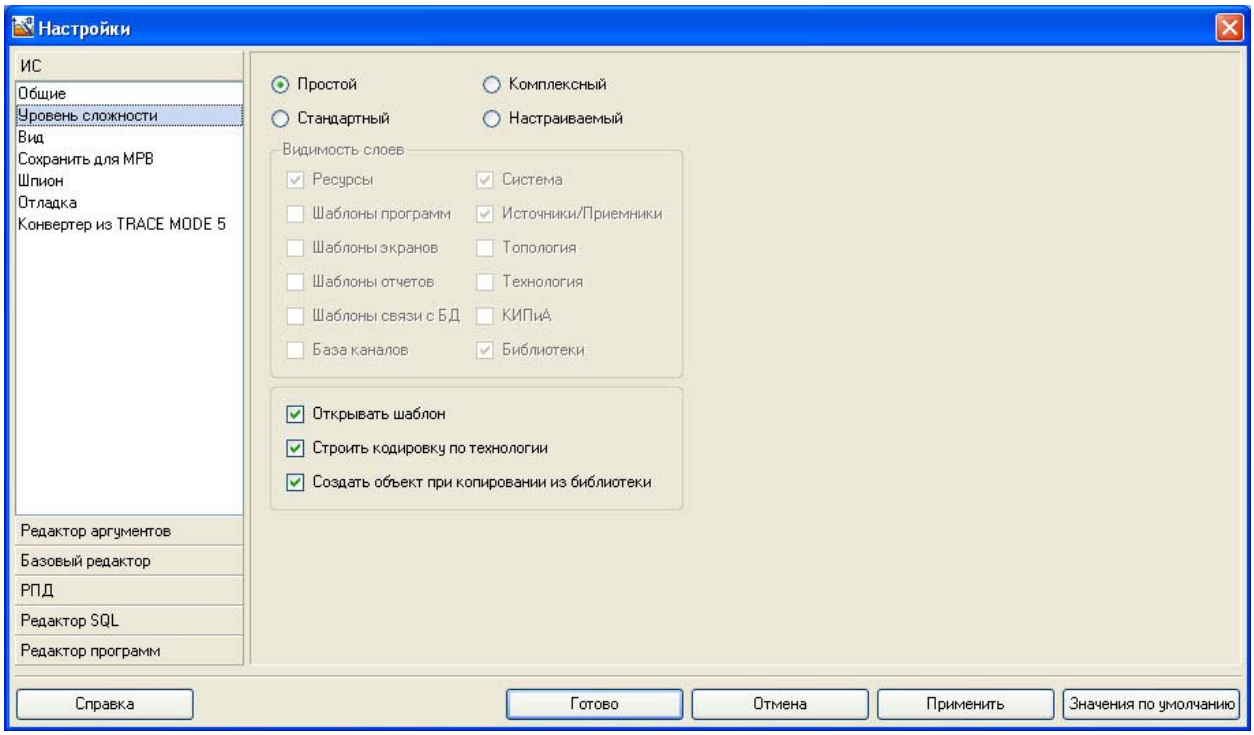


Рисунок 61 – Настройки ИСР

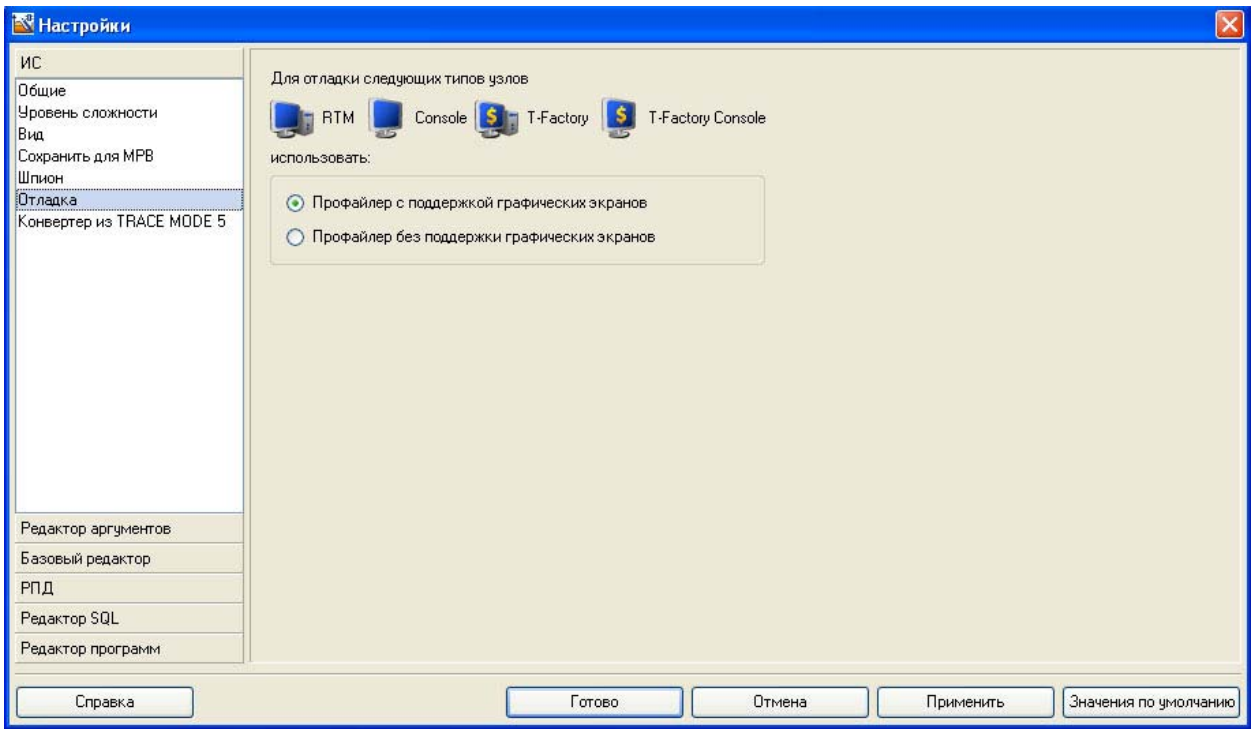


Рисунок 62 – Настройки ИСР

После проведенных настроек ИСР нажать кнопку Готово.

С помощью иконки  инструментальной панели создадим новый проект при этом в открывшемся на экране диалоге рисунок 63.

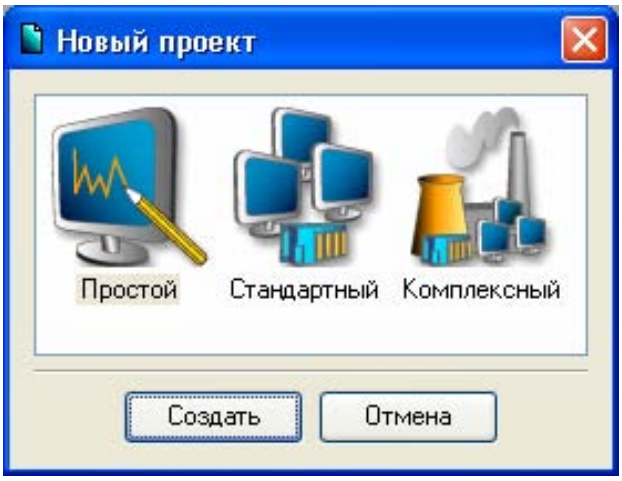


Рисунок 63 – Выбор типа проекта

Выберем **Простой** стиль разработки. После нажатия левой клавиши мыши (ЛК) на экранной кнопке **Создать**, в левом окне Навигатора проекта появится дерево проекта с созданным узлом **АРМ RTM_1**. Откроем узел **RTM_1** двойным щелчком ЛК, в правом окне **Навигатора проекта** отобразится содержимое узла – пустая группа **Каналы** и один канал класса «Вызов» **Экран#1**, предназначенный для отображения на узле **АРМ** графического экрана рисунок 64.

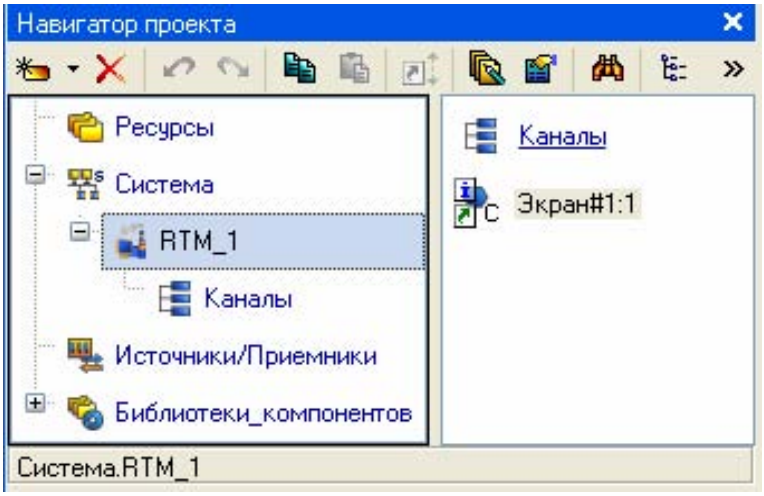


Рисунок 64 – Навигатор проекта

3.2.2 Создание графического экрана

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

Подготовим на экране вывод динамического текста для отображения численного значения входных переменных X1, X2, X3, X4 и выходного значения Y путем указания динамизации атрибута графического элемента (ГЭ). Определим назначение аргумента шаблона экрана.

Создадим и разместим четыре ГЭ  как показано на рисунке 65.

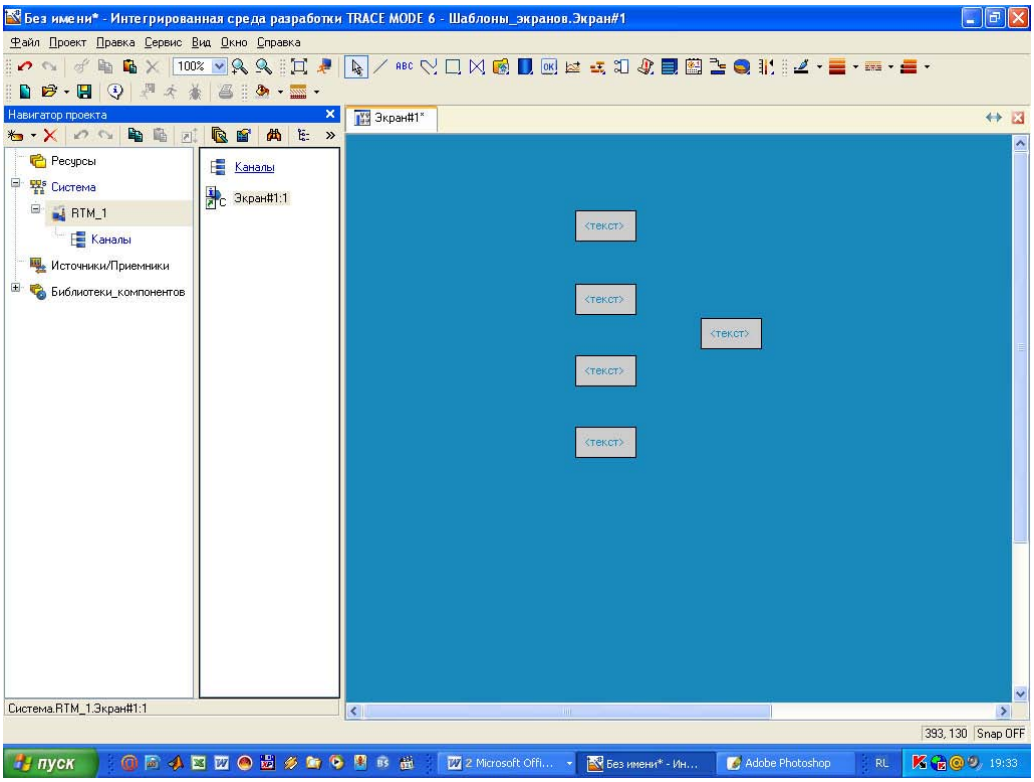


Рисунок 65 – Создание графических элементов

Двойным щелком ЛК на строке Текст вызвать меню **Вид индикации** рисунок 66;

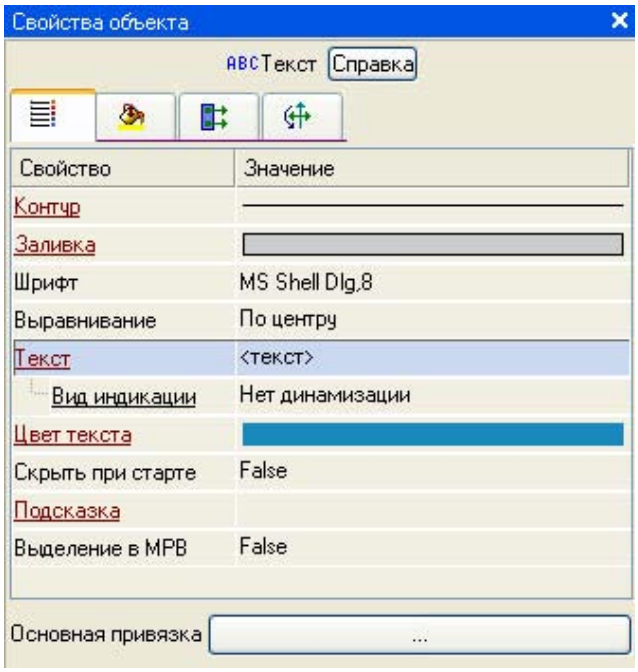


Рисунок 66 – Настройка индикации

В правом поле строки нажать ЛК и вызвать список доступных типов. Выбрать тип **Значение** рисунок 67.

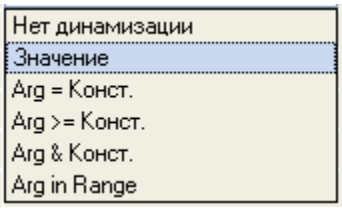


Рисунок 67 – Выбор типа динамизации

В открывшемся меню настройки параметров динамизации рисунок 68.

Рисунок 68 – Окно привязки

Выбрать свойство **Привязка.**

В открывшемся окне **Свойство привязки**, нажав кнопку  на его панели инструментов, создать пять аргументов экрана 4 входных и один выходной рисунок 69.

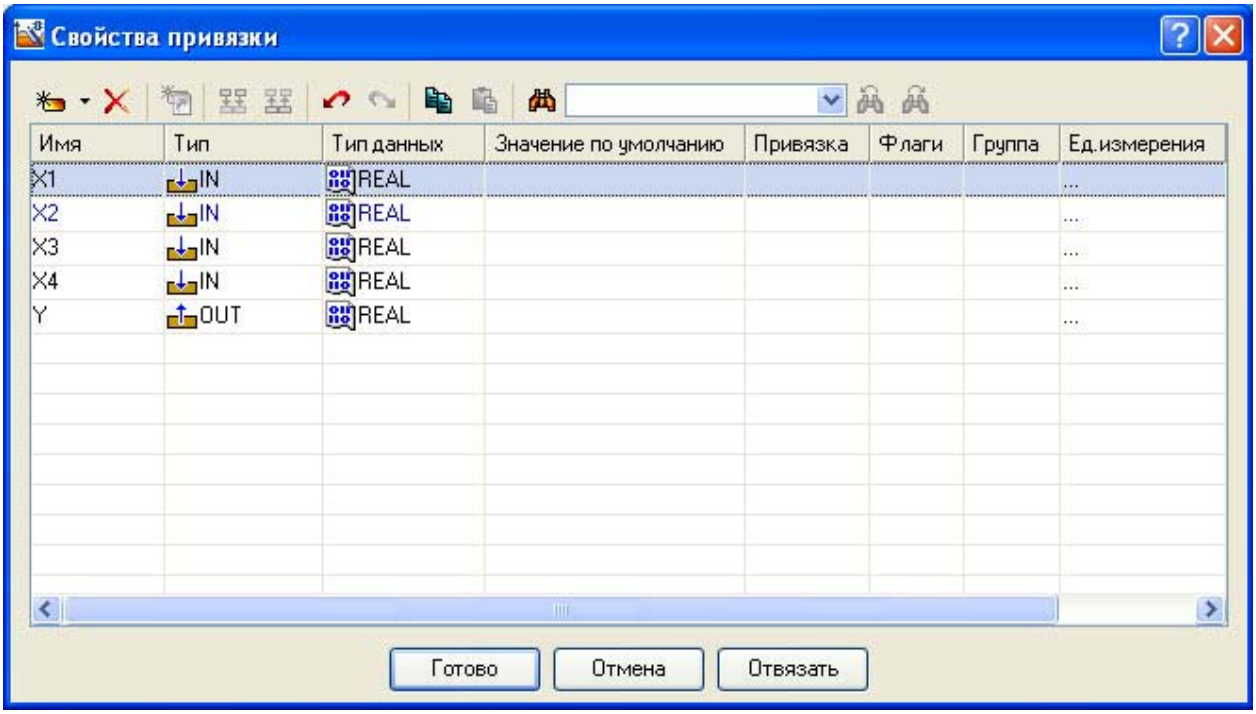


Рисунок 69 – Создание аргументов экрана

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

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

Выбрать на инструментальной панели графического редактора иконку ГЭ Кнопка -  . С помощью мыши разместить его в поле экрана как показано на рисунке 70.

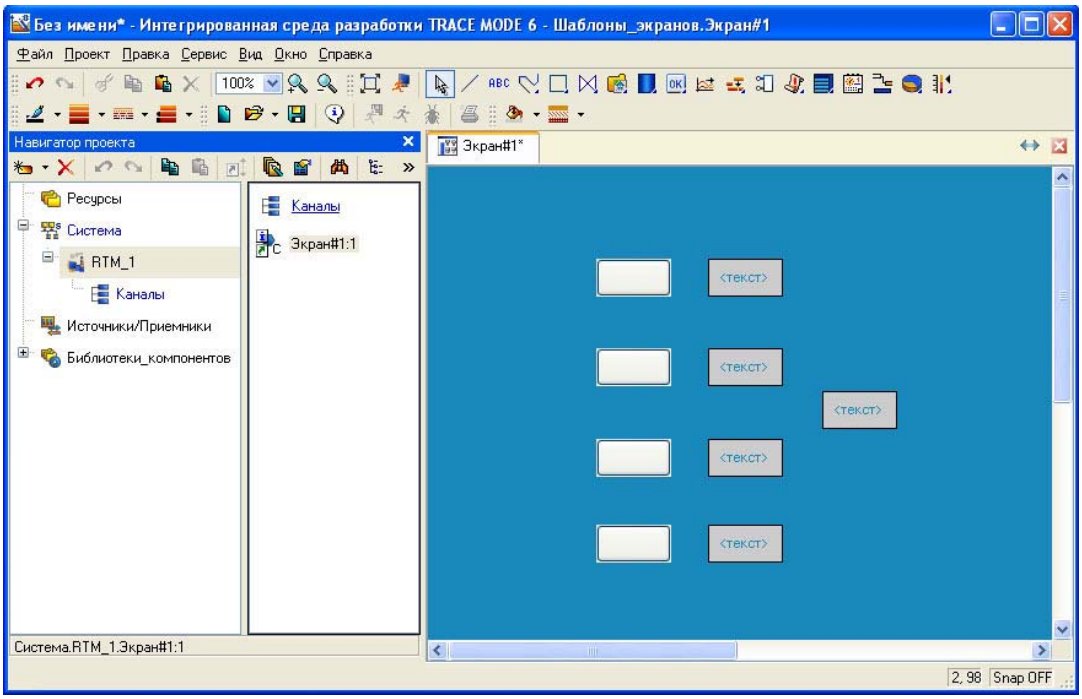


Рисунок 70 – Графический экран

Перейти в режим редактирования первой кнопки, вызвать окно свойств ГЭ рисунок 71.

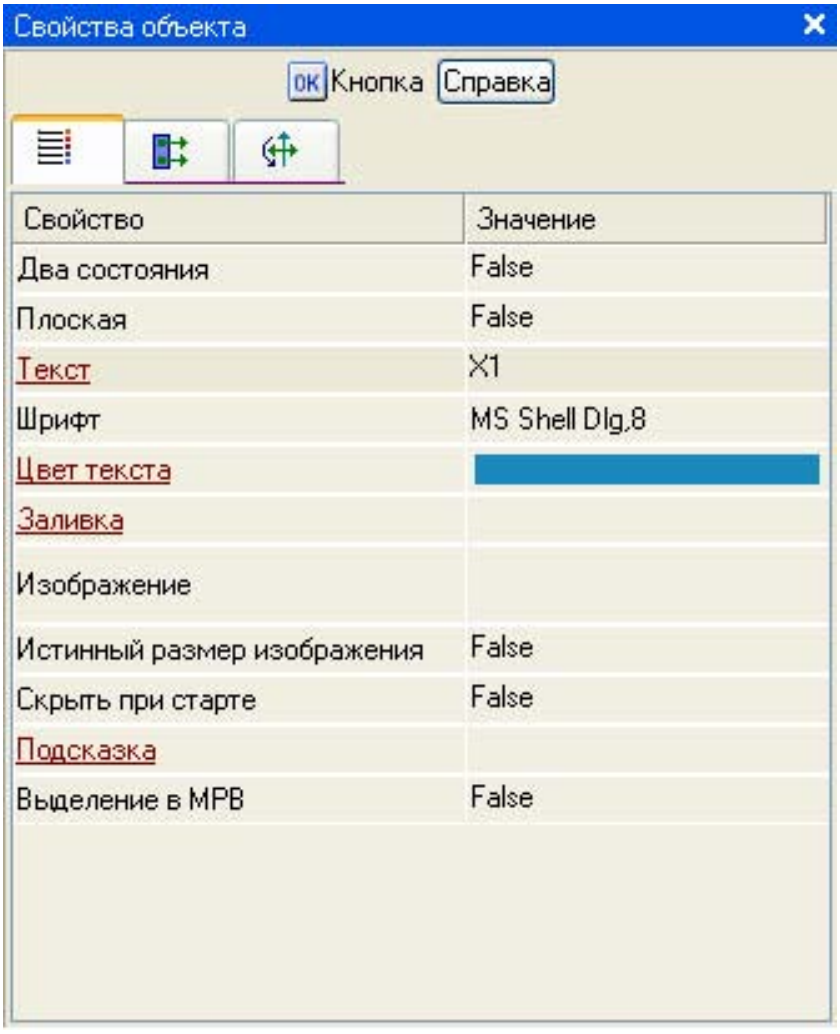


Рисунок 71 – Свойства графического элемента

В поле **Текст** ввести «X1». Открыть бланк **События** и ПК раскрыть меню **По нажатию** (**pressed**). Выбрать из списка команду **Добавить Передать значение**, раскрыть меню настроек выбранной команды рисунок 72.

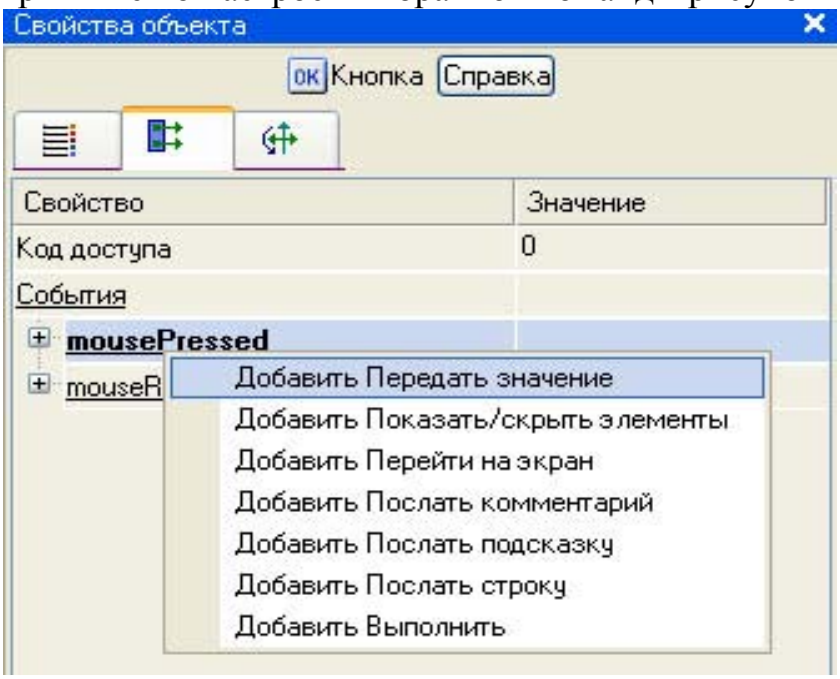


Рисунок 72 – Настройка типа передачи

В поле **Тип передачи (Send Type)** выбрать из списка **Ввести и передать (Enter & Send)** рисунок 73.

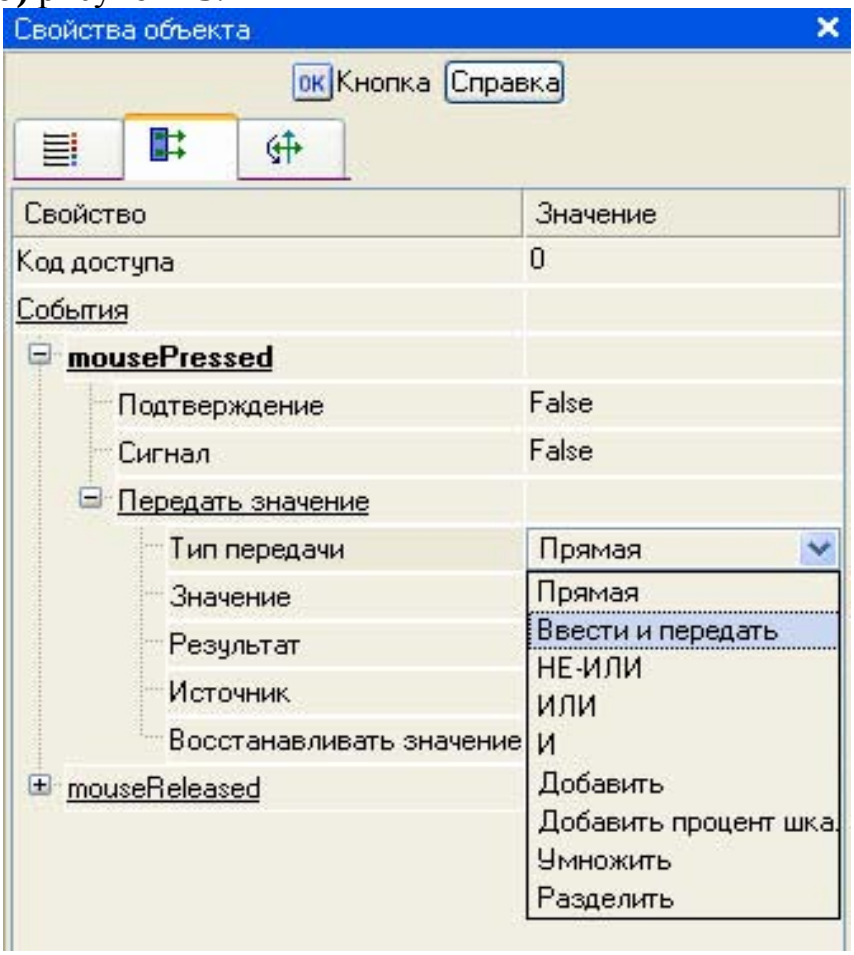


Рисунок 73 – Настройка типа передачи

ЛК в поле **Результат** вызвать табличный редактор аргументов и произвести привязку к аргументу X1. После привязки окно Свойства объекта выглядит следующим образом рисунок 74.

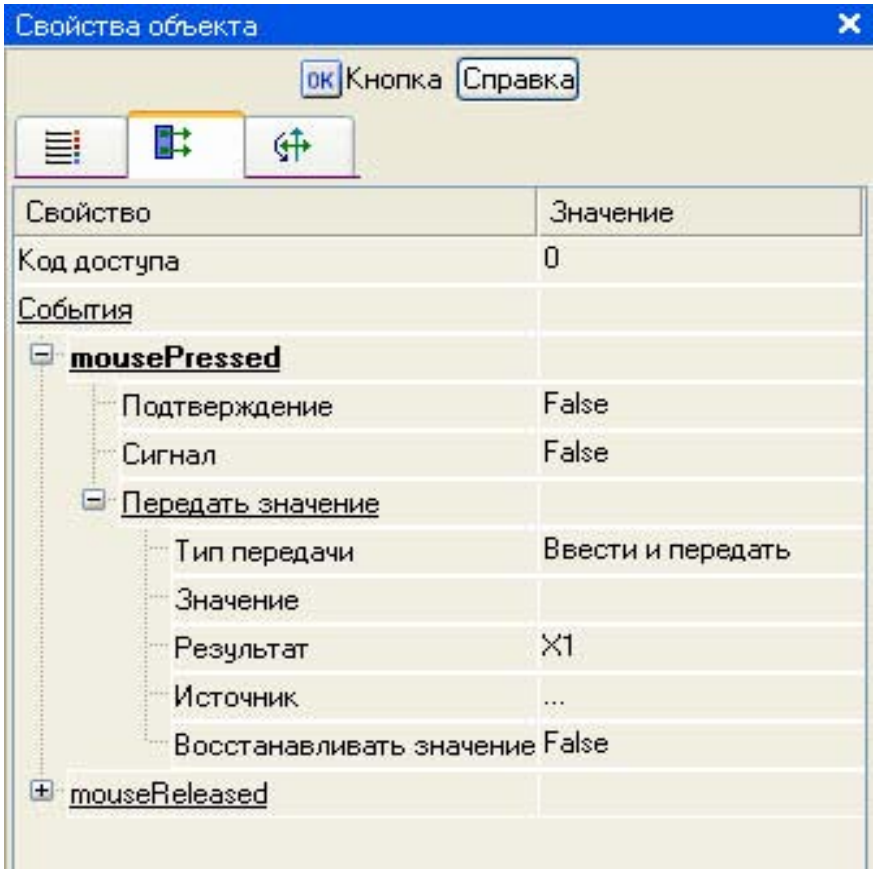


Рисунок 74 – Окончательный вид окна свойств графического элемента

Аналогичным образом настроить остальные кнопки x2, x3, x4.

3.2.3 Привязка аргумента экрана к каналу

Создадим по аргументам X1, X2, X3, X4 и выходу Y шаблона экрана новые каналы и отредактируем их привязку. В слое **Система** открыть узел **RTM_1**. С помощью ПК вызвать через контекстное меню свойства компонента **Экран#1** рисунок 75.

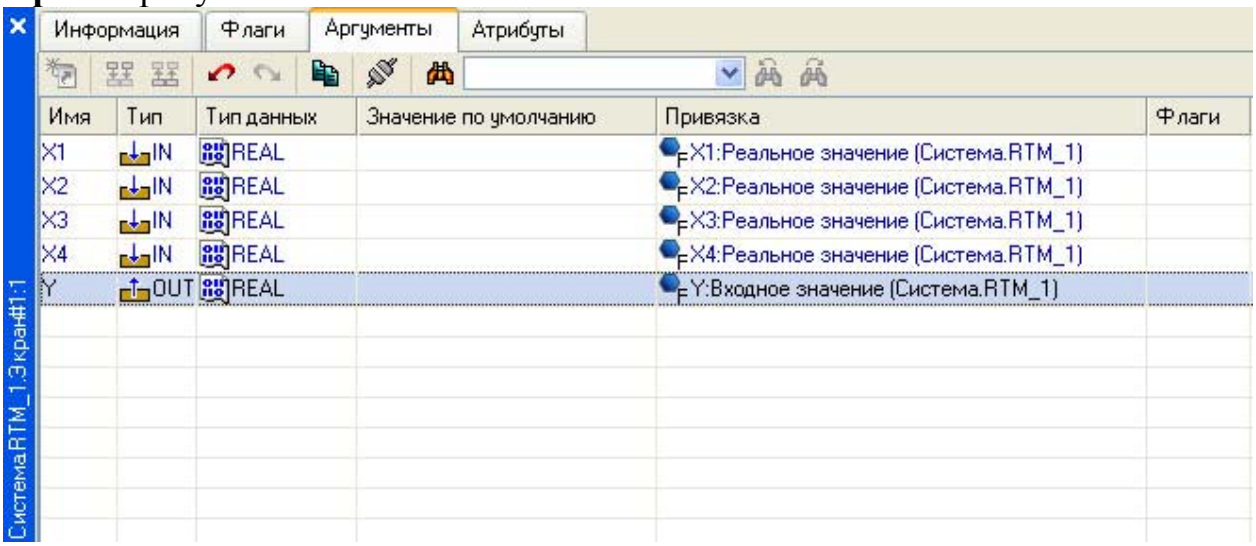


Рисунок 75 – Окно свойств экрана

Выбрать вкладку Аргументы, ЛК выделить аргументы X1, X2, X3, X4 и выход Y и с помощью иконки  создать новый канал. В результате, в узле **RTM_1** ,будут автопостроены следующие каналы рисунок 75.

Рисунок 75 – Автопостроены каналы

3.2.4 Создание программы на языке Техно FBD

Создадим программу, которая будет реализовывать заданную логическую функцию. Двойным щелчком ЛК открыть узел **RTM_1** и создать в нем компонент **Программа** рисунок 76.

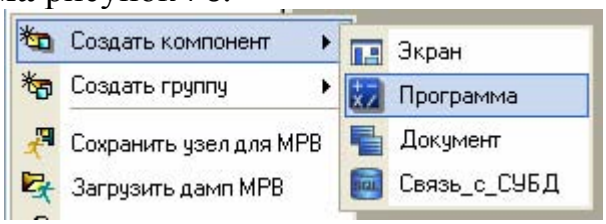


Рисунок 76 – Создание компонента Программа

Выделить компонент Программа#1 и ПК вызвать контекстное меню, выбрав в котором ЛК пункт **Редактировать шаблон**, перейти в режим редактирования программы рисунок 77.

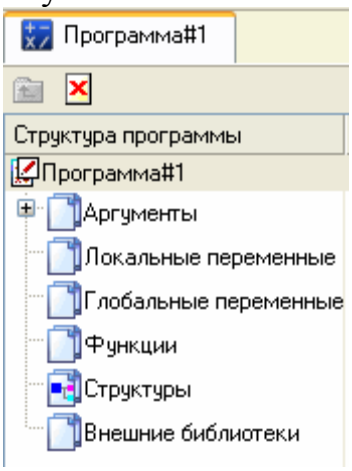


Рисунок 77 – Редактирование аргументов Программы

Выделением ЛК в дереве шаблона **Программа#1** строки Аргументы вызвать табличный редактор аргументов иконкой  создать в редакторе

аргументов четыре аргумента X1, X2, X3, X4 и выход Y. При этом первые 4 аргумента должны быть типа **IN**, а последний – **OUT** рисунок 78.

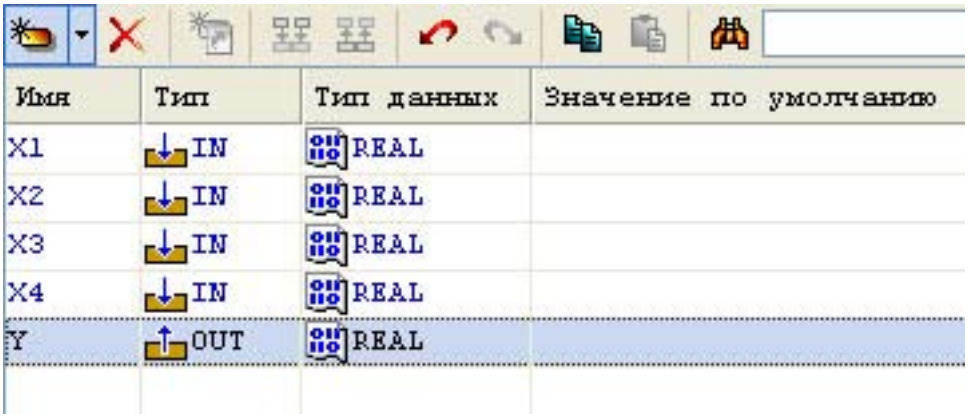


Рисунок 78 – Аргументы программы

Выделить в дереве шаблона строку **Программа#1** и в открывшемся диалоге Выбор языка выбрать язык **FBD** рисунок 79.

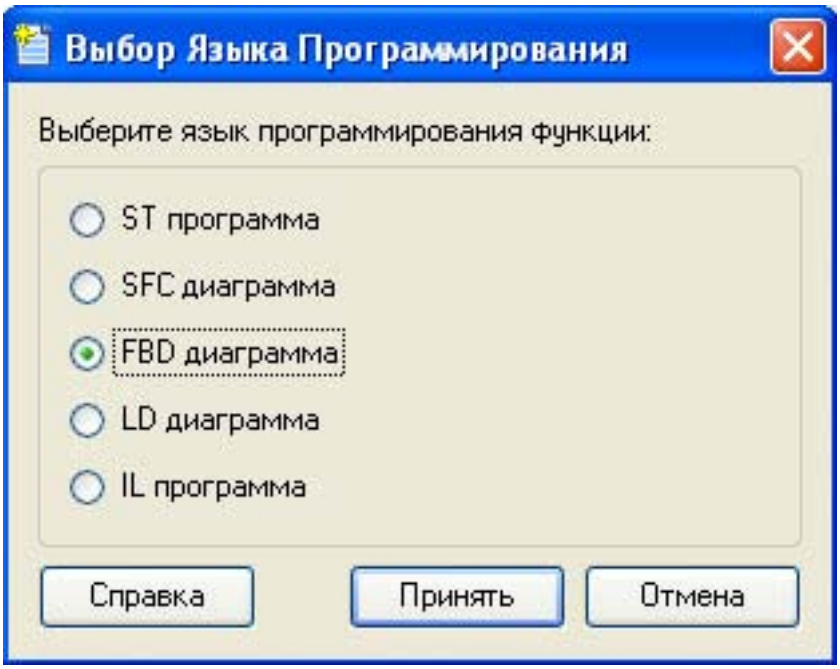


Рисунок 79 – Выбор языка программирования

По нажатию экранной кнопки **Принять** в открывшемся окне редактора программ с объявленными переменными создать программу в соответствии с заданием. Для выбора палитры FBD блоков необходимо ЛК мыши нажать на кнопку  после чего появится окно выбора FBD блоков рисунок 80. При разработки программы верхние входы FBD блоков не используются т.к. они предназначены для изменения порядка пересчета блоков, а информационными входами, являются входы начиная со второго.

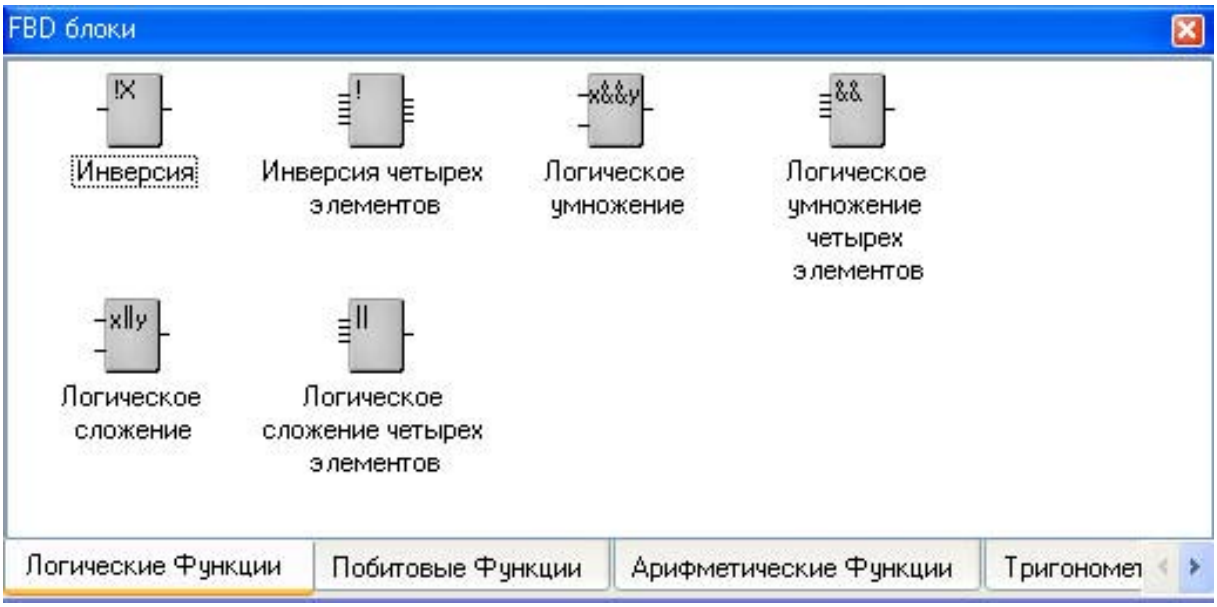


Рисунок 80 – Палитра FBD блоков

Для реализации логической функции выберем следующие блоки: из раздела *Логические Функции* **FBD блоки** инверсия (!X), логическое умножение (X&&Y и &&), логическое сложение (||). После размещения всех блоков для рассматриваемого примера программа будет выглядеть следующим образом рисунок 81.

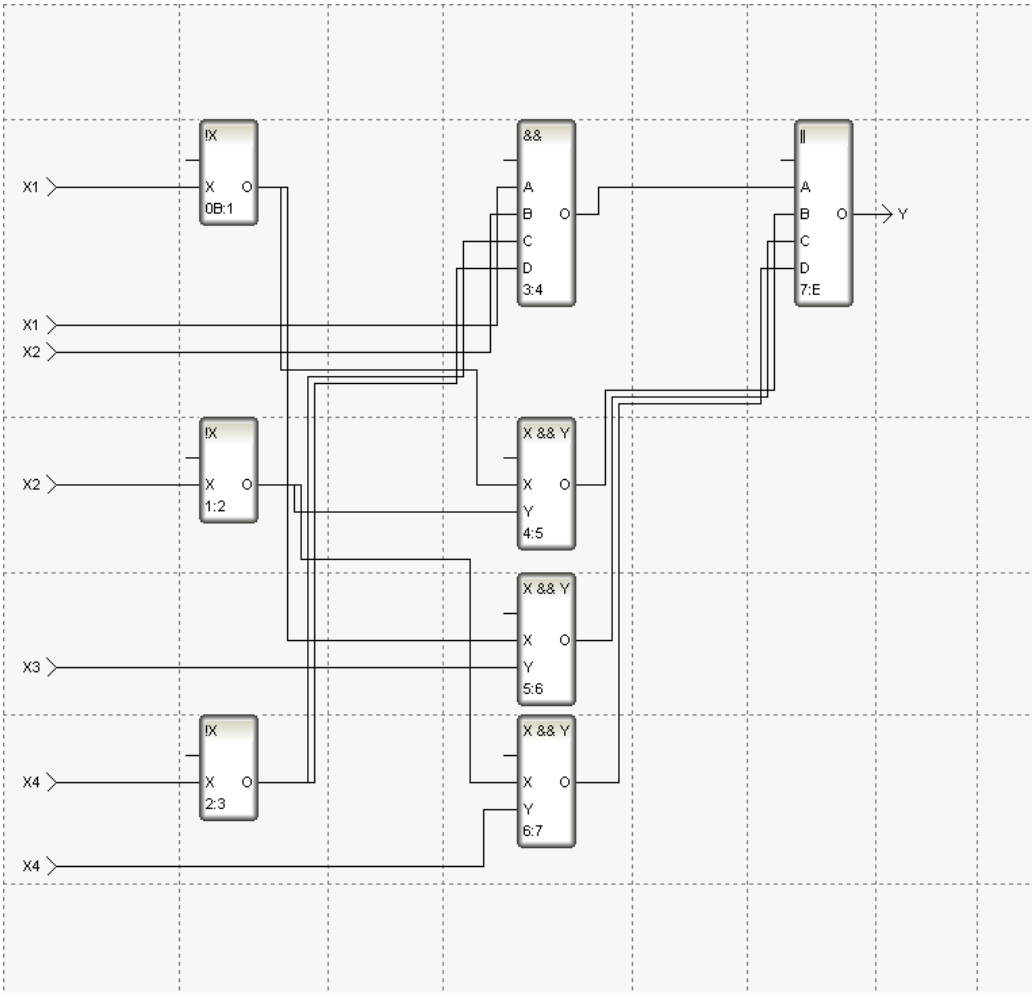


Рисунок 81 – Программа на языке **Техно FBD**

С помощью иконки  на инструментальной панели редактора или «горячей клавишей» F7 скомпилировать программу и убедиться в успешной компиляции в окне Выход (Output), вызываемого из инструментальной панели с помощью иконки  рисунок 82.

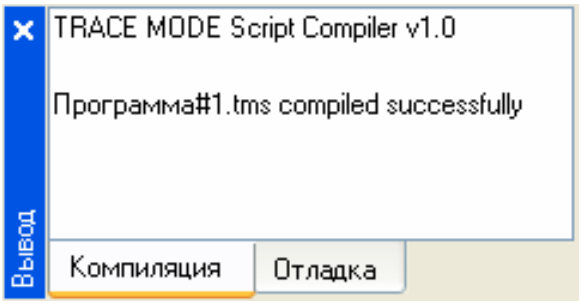


Рисунок 82 – Результат успешной компиляции программы

3.2.5 Привязка аргументов программы

Выполним привязку аргументов программы к атрибутам каналов. Вызвать свойства компонента **Программа#1** через контекстное меню. Выбрать вкладку **Аргументы**. Двойным нажатием в поле **Привязка** привязать аргументы программы к атрибутам каналов – аргументы X1, X2, X3, X4 к реальному значению каналов X1, X2, X3, X4 рисунок 83.

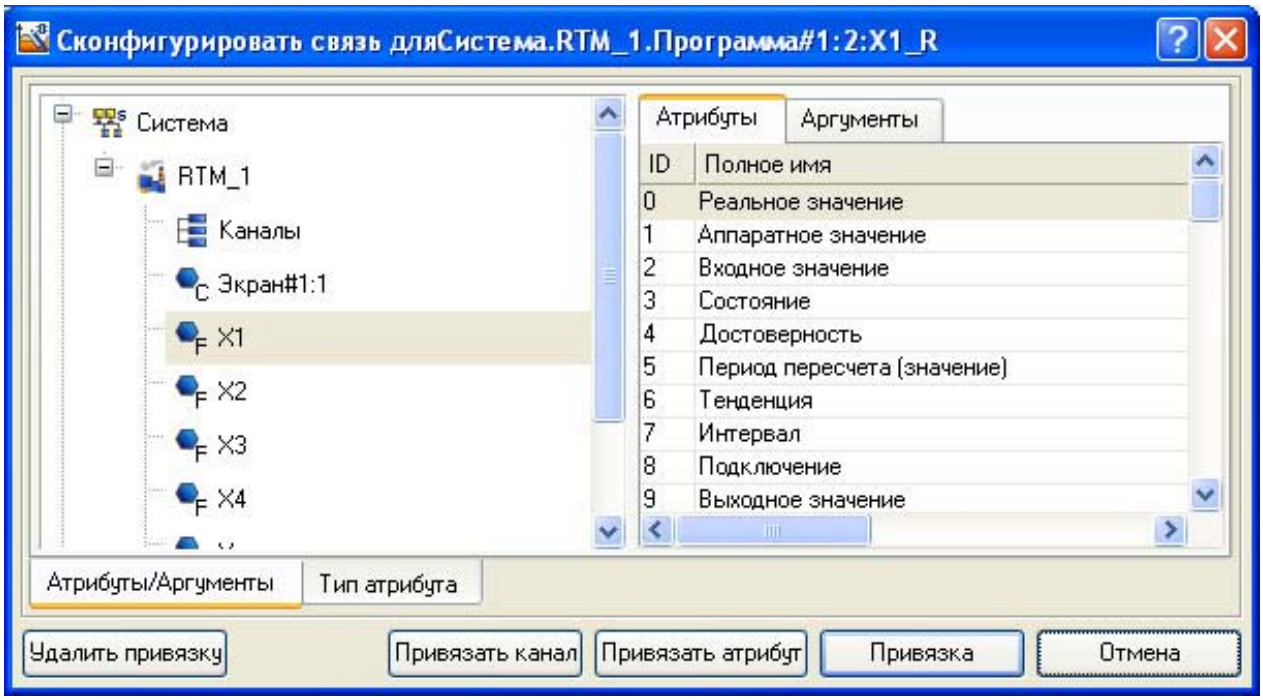


Рисунок 83 – Связь аргументов программы с аргументами экрана

Двойным щелчком в поле **Привязка** аргумента программы Y вызвать окно настройки связи, выбрать в левом окне канал класса «Вызов» **Экран#1**, а в правом окне выбрать вкладку **Аргументы** и указать в ней аргумент Y и кнопкой **Привязка** подтвердить связь рисунок 84.

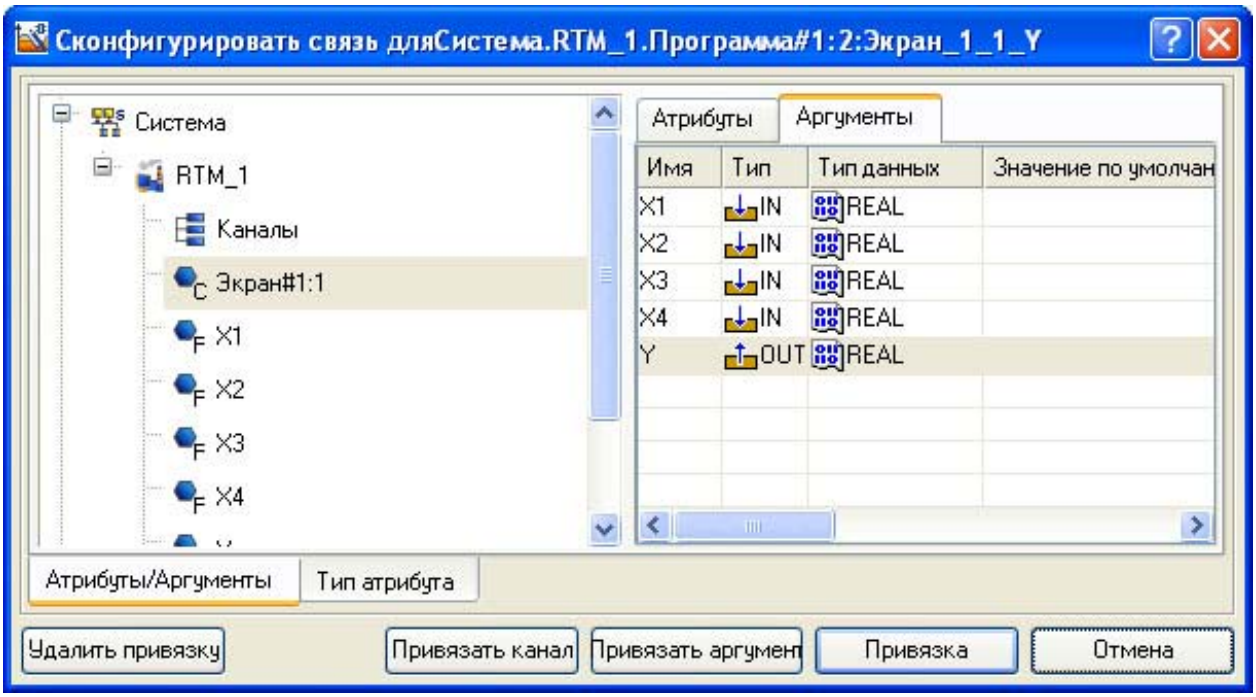


Рисунок 84 – Связь выходных аргументов

В результате, будем иметь рисунок 85.

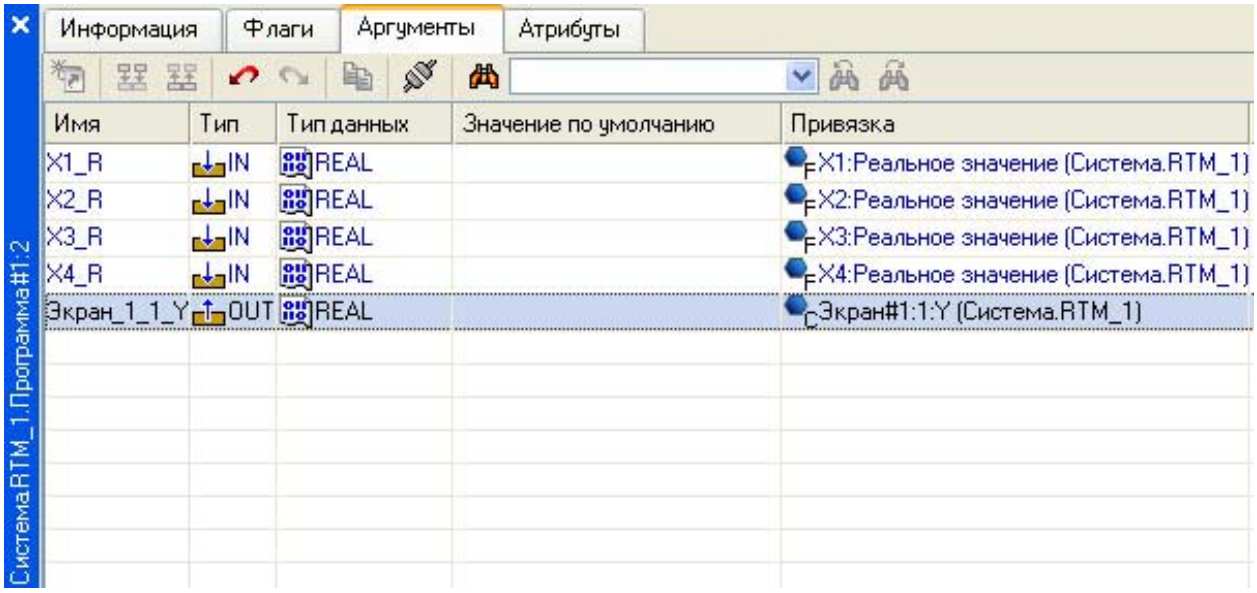


Рисунок 85 – Окончательная настройка связи

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

3.2.6 Запуск проекта

Сохранить проект с помощью иконки . На инструментальной панели выбрать команду и скомпилировать тем самым проект для запуска в реальном времени. В навигаторе проекта выделить узел RTM_1 рисунок 86.

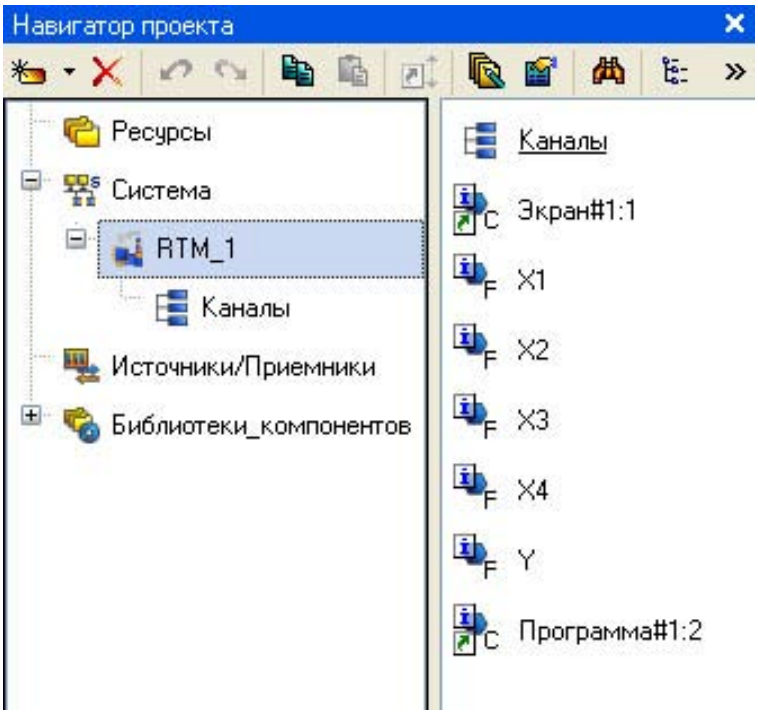


Рисунок 86 – Навигатор проекта

Выбрать иконку  на инструментальной панели и запустить режим исполнения.

Для составления отчета необходимо в меню **Файл** выполнить команду **Документировать проект** полученный *.html файл необходимо распечатать.

3.2.7 Задания для лабораторной работы

Варианты заданий выбираются по порядковому номеру в списке группы.

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

17. $f = x_1x_3 \cup \overline{x_1}x_2\overline{x_4} \cup \overline{x_1x_2x_4} \cup x_2x_3 \cup x_3x_4$
18. $f = \overline{x_1}x_3 \cup \overline{x_1x_2} \cup \overline{x_1}x_4 \cup \overline{x_2x_4} \cup x_2\overline{x_3}x_4$
19. $f = x_1x_3 \cup \overline{x_1x_2x_3} \cup x_2\overline{x_3x_4}$
20. $f = x_1x_2x_4 \cup x_1x_3x_4 \cup \overline{x_1x_4} \cup \overline{x_1x_2x_3} \cup x_2x_3\overline{x_4}$
21. $f = x_1x_3 \cup \overline{x_1x_2x_3} \cup \overline{x_1x_3x_4} \cup \overline{x_2x_4}$
22. $f = x_1x_2x_4 \cup x_1x_3x_4 \cup \overline{x_1x_2x_3} \cup \overline{x_1x_2x_4} \cup x_2x_3x_4 \cup \overline{x_2x_3x_4}$
23. $f = x_1x_2x_3 \cup \overline{x_1x_2} \cup \overline{x_1x_4} \cup \overline{x_2x_3x_4} \cup x_3\overline{x_4}$
24. $f = x_1\overline{x_2}x_3 \cup x_1x_3\overline{x_4} \cup \overline{x_1}x_4 \cup x_2\overline{x_3}$
25. $f = x_1x_2\overline{x_4} \cup \overline{x_1x_2} \cup \overline{x_1}x_3 \cup \overline{x_2}x_4$

3.2.8 Вопросы для защиты лабораторной работы №2

1. SCADA система TRACE MODE
2. Принцип работы монитора. Канал TRACE MODE 6
3. Обеспечение работы распределенных АСУ
4. Резервирование
5. Автопостроение
6. Математическая обработка данных
7. Архивирование каналов узла
8. Архивирование каналов проекта
9. Отчет тревог и генерация сообщений
10. Графический интерфейс оператора
11. Генерация документов (отчетов)
12. Защита проекта
13. Технология разработки проекта в ИС
14. Классификация компонентов
15. Классификация слоев
16. Классификация узлов
17. Назначение групп источников (приемников)
18. Канал класса HEX16
19. Понятие логической обработки.
20. Понятие трансляции.
21. Перечислите основные понятия языка программирования техно-FBD.
22. Типы входа/выхода FBD-блоков.
23. Порядок написания FBD-программ.
24. Назначение входов/выходов FBD-блоков используемых в лабораторной работе.
25. Понятие переменной.
26. Понятие функционального блока.
27. Понятие выполняемой функции.
28. Понятие номера блока.
29. Описание входов и выходов блоков.
30. Понятие пересчета блоков.
31. Описание функциональных блоков.

3.3 ЛАБОРАТОРНАЯ РАБОТА №3 «РЕАЛИЗАЦИЯ ОДНОКОНТУРНОЙ СИСТЕМЫ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ ПРИ ПОМОЩИ SCADA–СИСТЕМЫ TRACE MODE»

Цель работы: освоить методику программирования одноконтурной системы автоматического регулирования при помощи SCADA–системы TRACE MODE на языке Техно FBD.

3.3.1 Порядок выполнения лабораторной работы

Рассмотрим реализацию одноконтурной системы автоматического регулирования при помощи SCADA–системы TRACE MODE на языке Техно FBD.

Разработка любого проекта автоматизации всегда начинается с запуска Интегрированной среды разработки (ИСР). Для ее запуска необходимо выполнить команду: TRACE MODE IDE 6 (base) из группы установки инструментальной системы в меню Программы WINDOWS или двойным щелчком ЛК мыши по иконке  рабочего стола Windows рисунок 87.

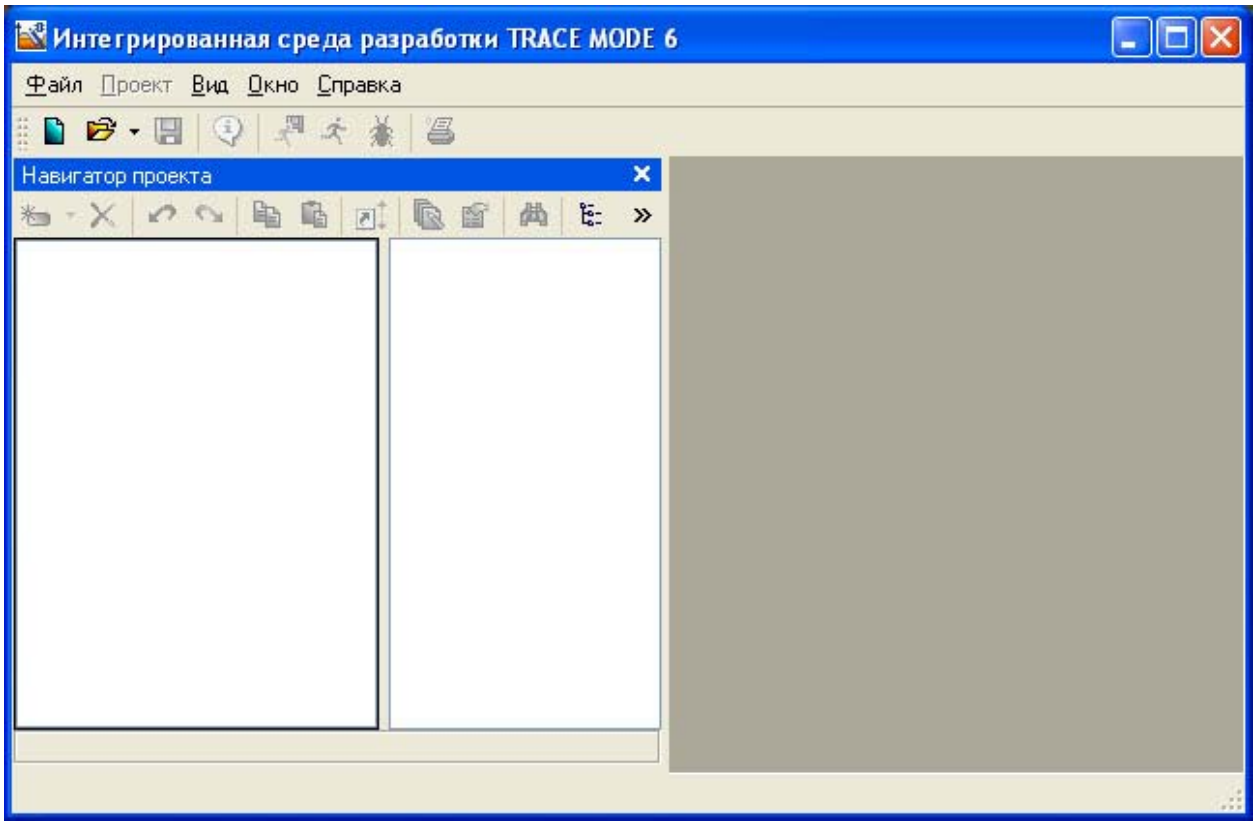


Рисунок 87– Интегрированная среда разработки

После запуска ИСР в меню **Файл** выбрать команду *Настройки ИС…*

В появившемся окне выбрать *Уровень сложности* и настроить как показано на рисунке 88:

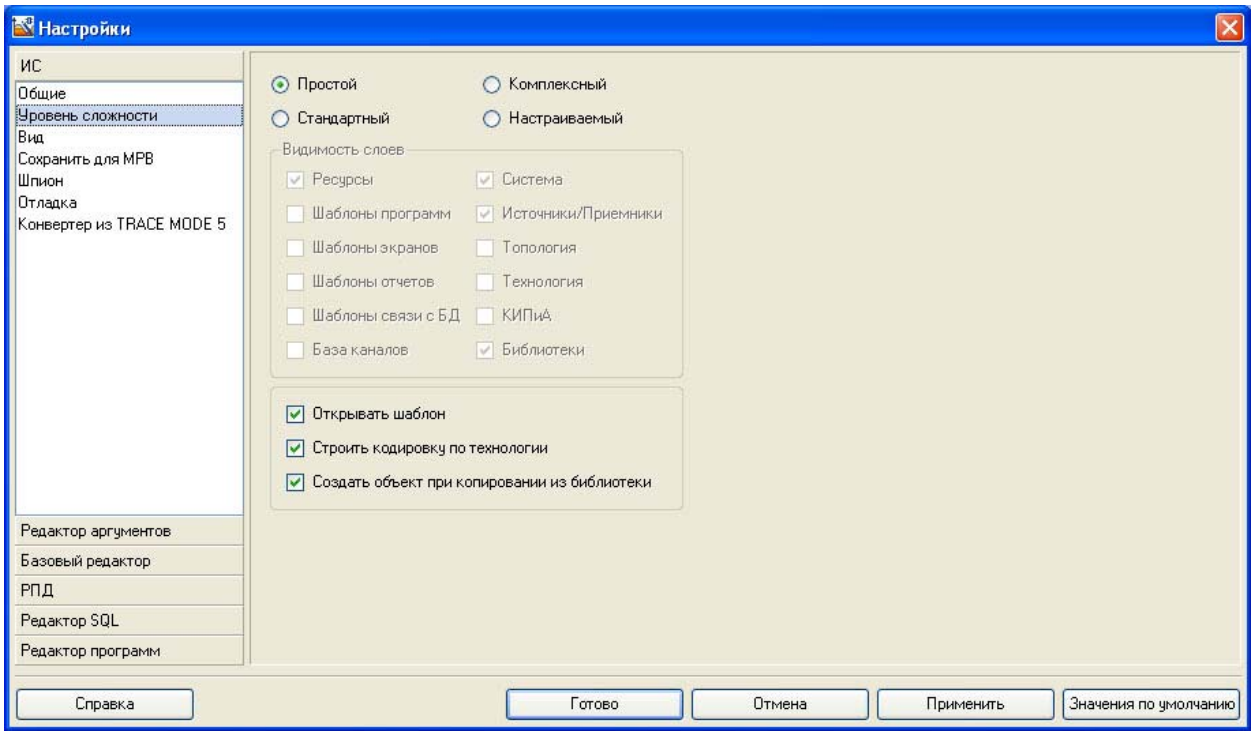


Рисунок 88 – Настройки ИСР

затем выбрать *Отладка* и настроить как показано на рисунке 89.

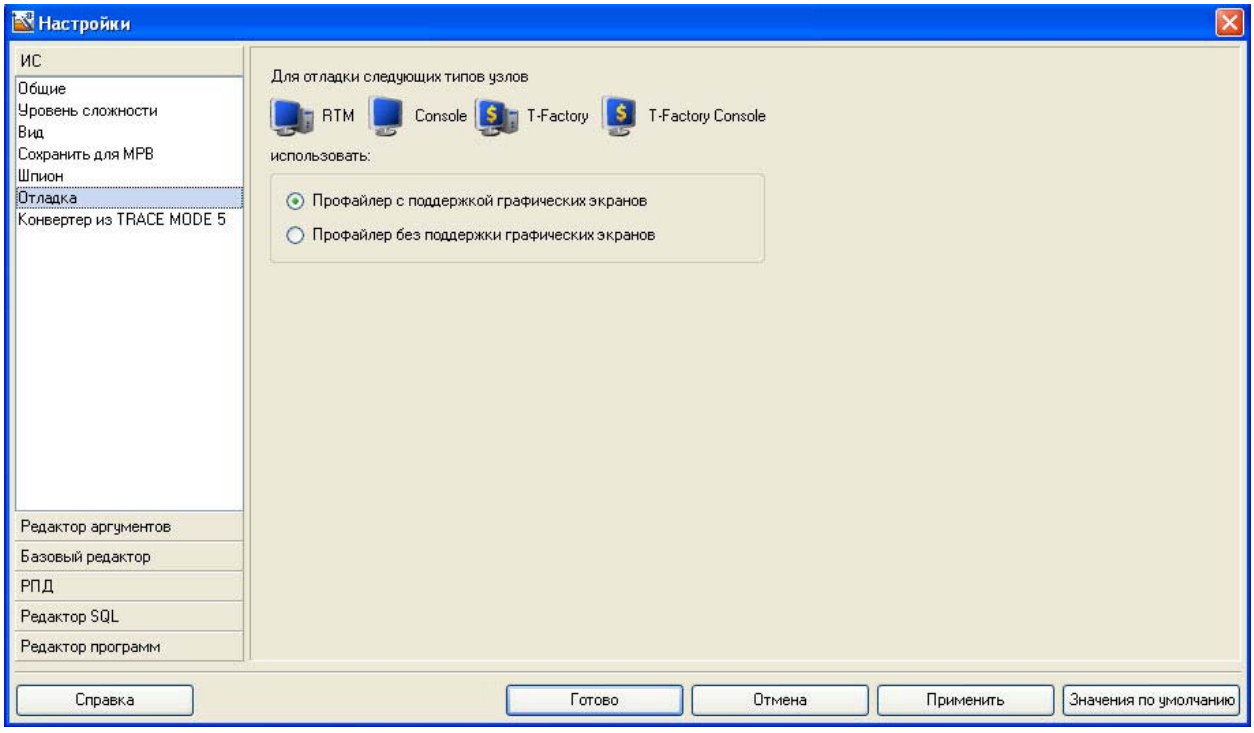


Рисунок 89 – Настройки ИСР

После проведенных настроек ИСР нажать кнопку Готово.

Нажатием левой клавиши мыши по иконке  инструментальной панели создадим новый проект при этом в открывшемся на экране диалоге рисунок 90.

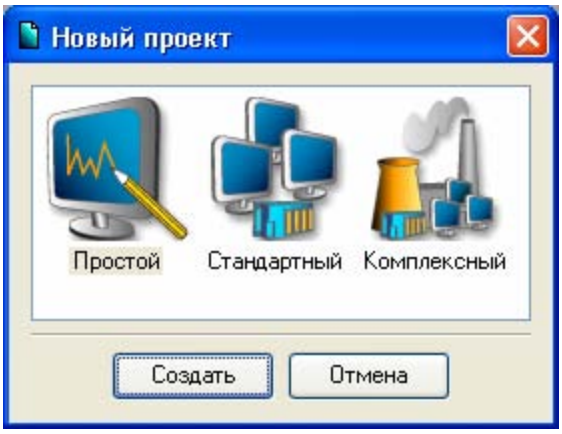


Рисунок 90 – Выбор типа проекта

Выберем **Простой** стиль разработки. После нажатия левой клавиши мыши (ЛК) на экранной кнопке ***Создать***, в левом окне Навигатора проекта появится дерево проекта, с созданным узлом **АРМ RTM_1**. Откроем узел **RTM_1** двойным щелчком ЛК, при этом, в правом окне **Навигатора проекта** отобразится содержимое узла – пустая группа **Каналы** и один канал класса «Вызов» **Экран#1**, предназначенный для отображения на узле **АРМ** графического экрана рисунок 91.

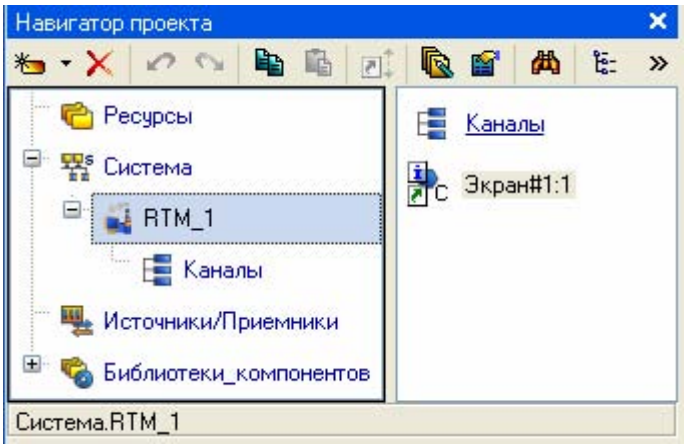


Рисунок 91 – Навигатор проекта

3.3.2 Создание графического экрана

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

Подготовим на экране вывод динамического текста для отображения численного значения входных переменных апериодического звена первого порядка: коэффициента усиления объекта (Kob), постоянной времени объекта (Tob), времени запаздывания (Tau); ПИД регулятора: коэффициент усиления регулятора (Kr), времени интегрирования (Ti), времени дифференцирования (Td); и величины задания (Zad).

Выбрать на инструментальной панели графического редактора иконку ГЭ Кнопка -  . С помощью мыши разместить семь элементов в поле экрана как показано на рисунке 92.

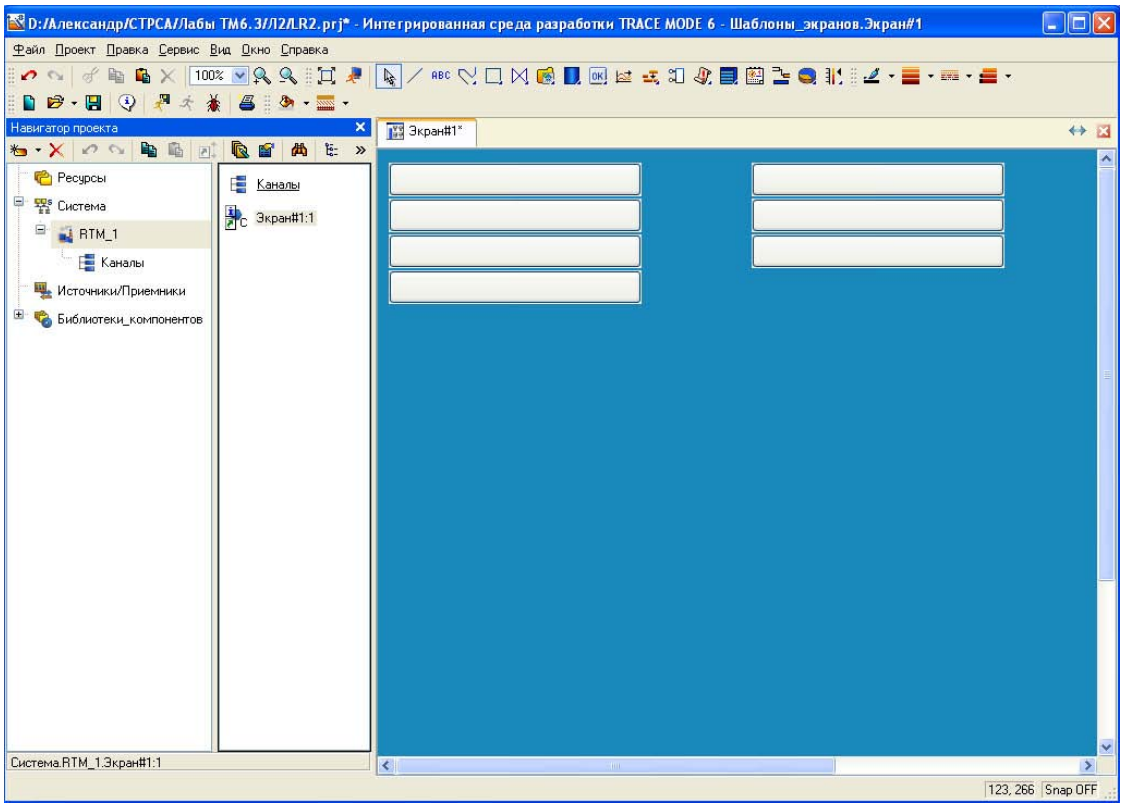


Рисунок 92 – Графический экран

Перейти в режим редактирования нажатием кнопки, затем двойным щелчком ЛК вызвать окно свойств первого ГЭ рисунок 93.

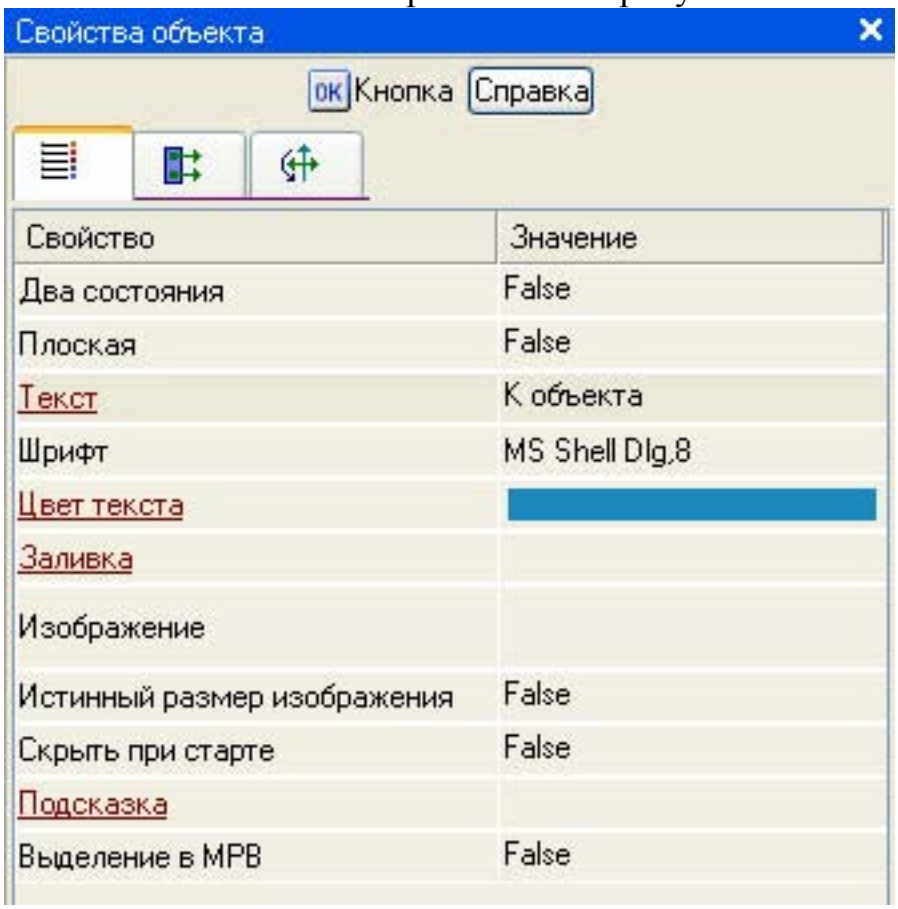


Рисунок 93 – Свойства графического элемента

В поле **Текст** ввести «К объекта». Открыть бланк **Действия** и (правой кнопкой мыши) ПК раскрыть меню События **По нажатию** (**mousePressed**). Выбрать из списка команду **Добавить Передать значение**, раскрыть меню настроек выбранной команды рисунок 94.

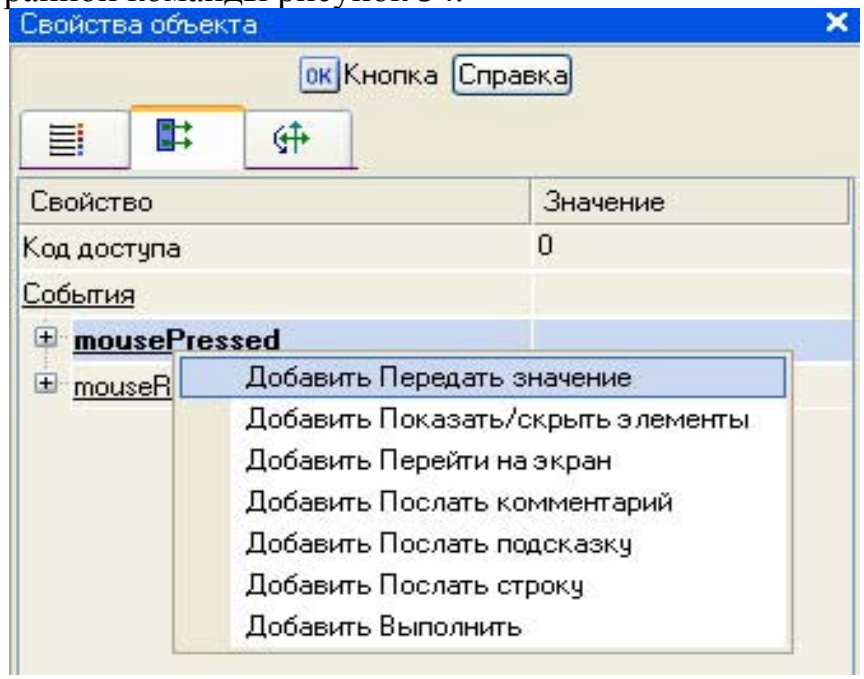


Рисунок 94 – Настройка типа передачи

В поле **Тип передачи (Send Type)** выбрать из списка **Ввести и передать (Enter & Send)** рисунок 95.

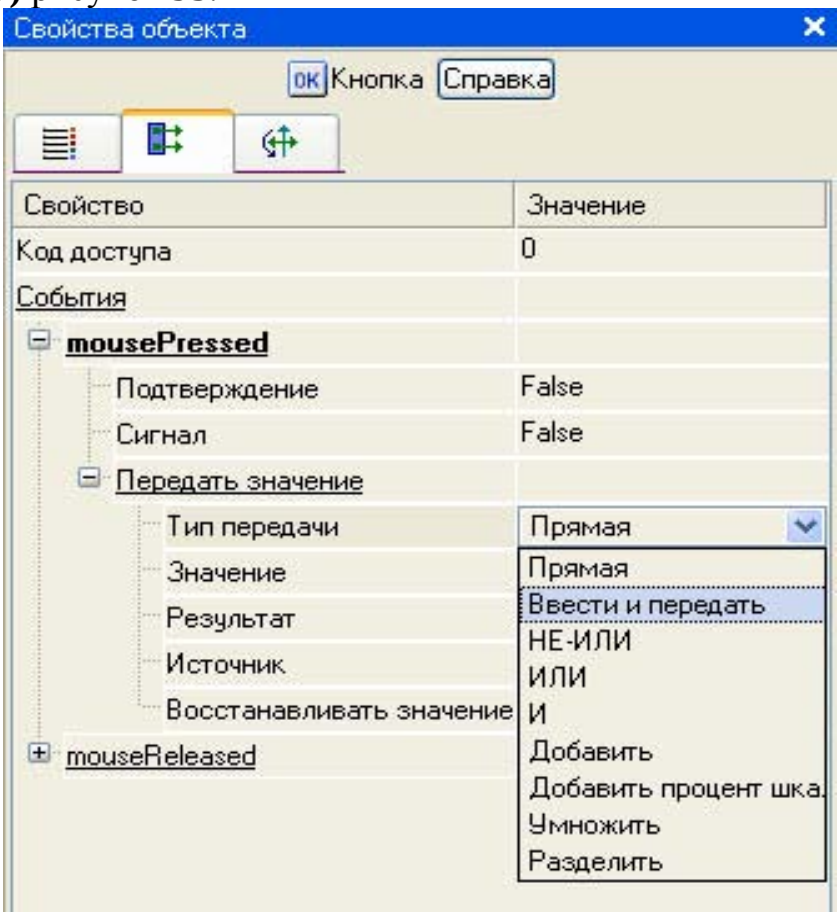


Рисунок 95 – Настройка типа передачи

ЛК в поле **Результат** вызвать табличный редактор аргументов. В открывшемся окне **Свойство привязки**, нажав кнопку  на его панели инструментов, создать восемь аргументов экрана 7 входных и один выходной. Входной (IN) или Выходной (OUT) будет аргумент, выбирается из низпадающего меню, появляющегося при двойном щелчке ЛК по типу аргумента (рисунок 96). Двойным щелчком ЛК выделить имя аргумента и изменить его, введя с клавиатуры «Kob» (завершить ввод нажатием клавиши Enter). Подтвердить связь с этим аргументом нажатием кнопки **Готово** Аналогичным образом настроить все остальные аргументы.

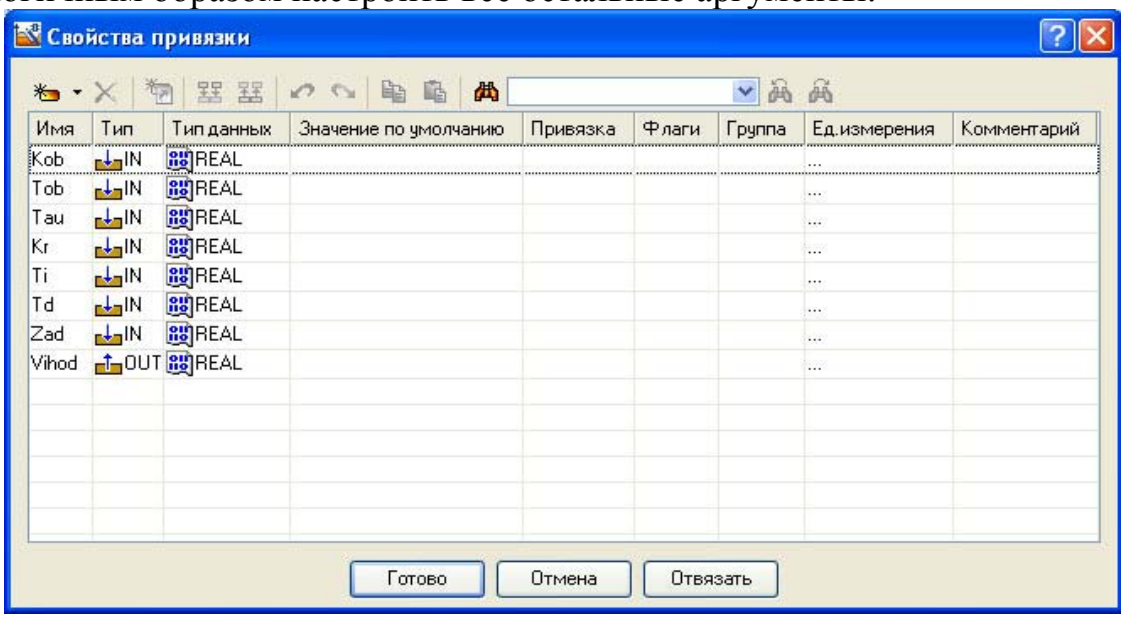


Рисунок 96 – Создание аргументов экрана

После привязки окно Свойства объекта выглядит следующим образом рисунок 97.

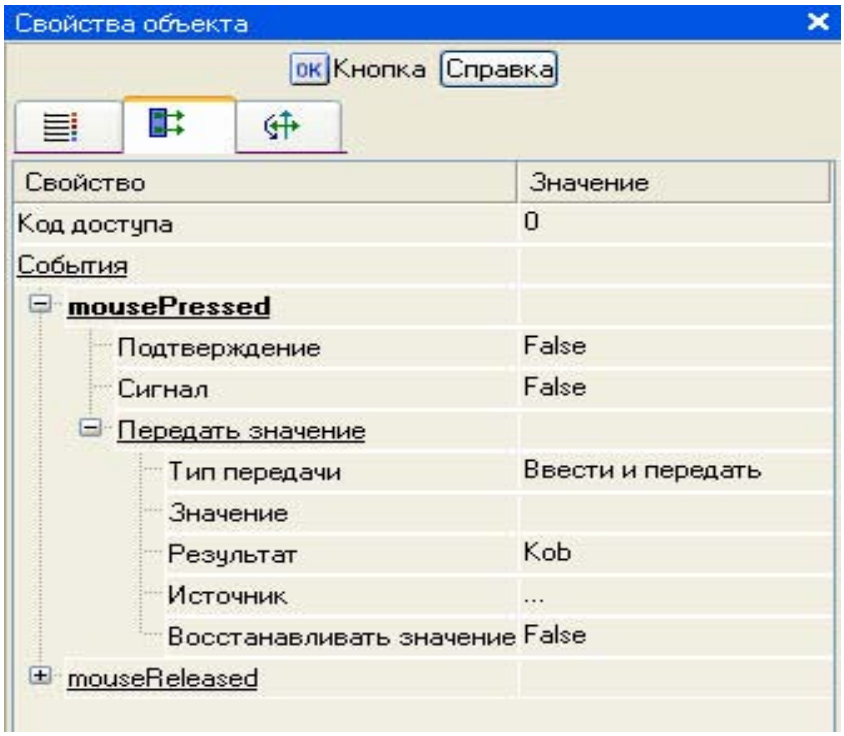


Рисунок 97 – Окончательный вид окна свойств графического элемента

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

Для отображения числового значения входных аргументов создадим и разместим семь ГЭ  как показано на рисунке 98.

Для этого активировав ГЭ  нажатием ЛК, рисуем области вывода динамического текста, задавая левый верхний и правый нижний углы щелчком ЛК.

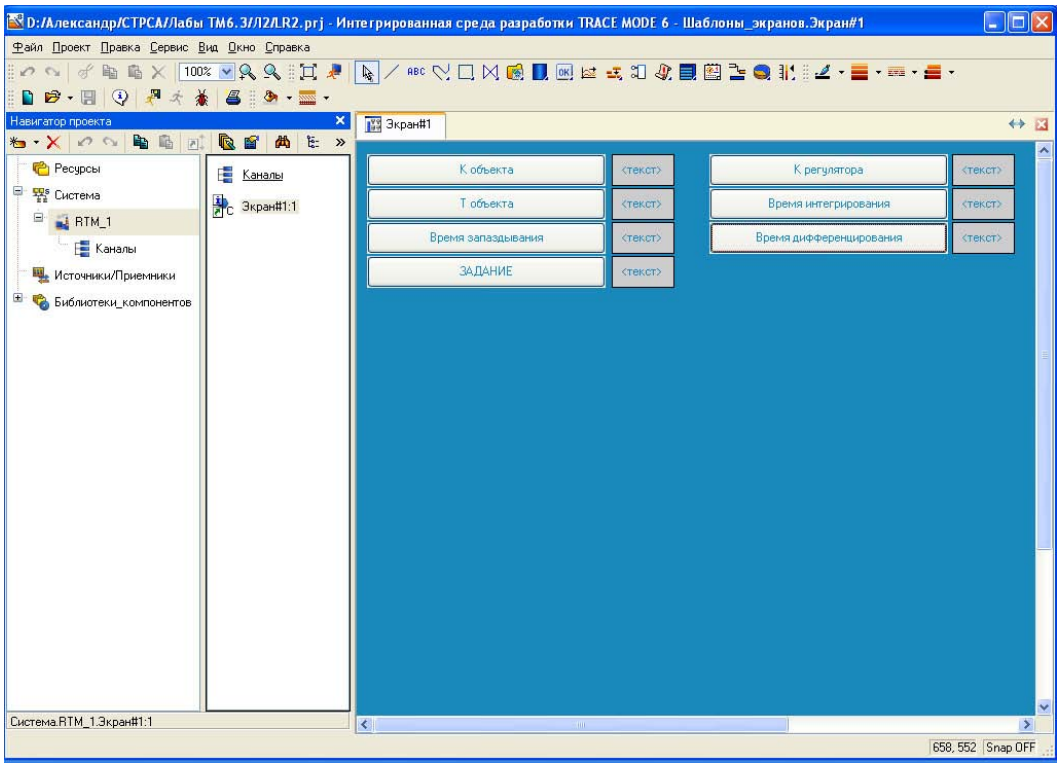


Рисунок 98 – Создание графических элементов

После этого необходимо переключиться в режим редактирования активацией ГЭ  . Двойным щелком ЛК на строке Текст вызвать меню **Вид индикации** рисунок 99;

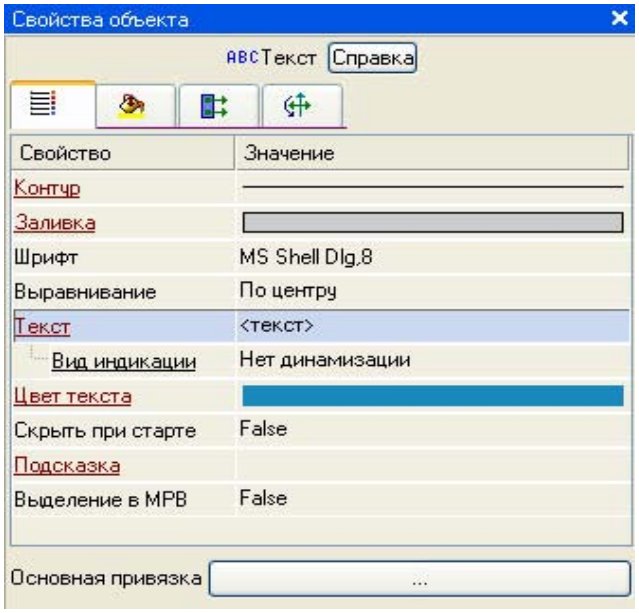


Рисунок 99 – Настройка индикации

В окне «Свойства объекта», двойным щелчком ЛК по свойству **Текст** развернуть его. В появившемся подпункте **Вид индикации** щелкнув по значению **Нет динамизации** выбрать его тип **Значение** (рисунок 100).

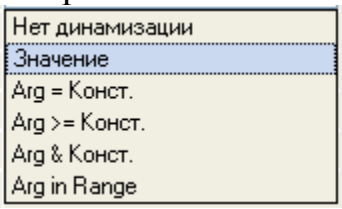


Рисунок 100 – Выбор типа динамизации

В открывшемся меню настройки параметров динамизации рисунок 101.

Рисунок 101 – Окно привязки

Выбрать свойство **Привязка** и связать с Kob.

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

На экрана разместим ГЭ Тренд для вывода значений **Zad** и **Vihod** рисунок 102. Основные свойства ГЭ оставим заданными по умолчанию. Перейдем во вкладку и, выделив ЛК строку **Кривые**, с помощью ПК создадим две новые кривые. Настроим их привязки к аргументам, толщину и цвет линий:

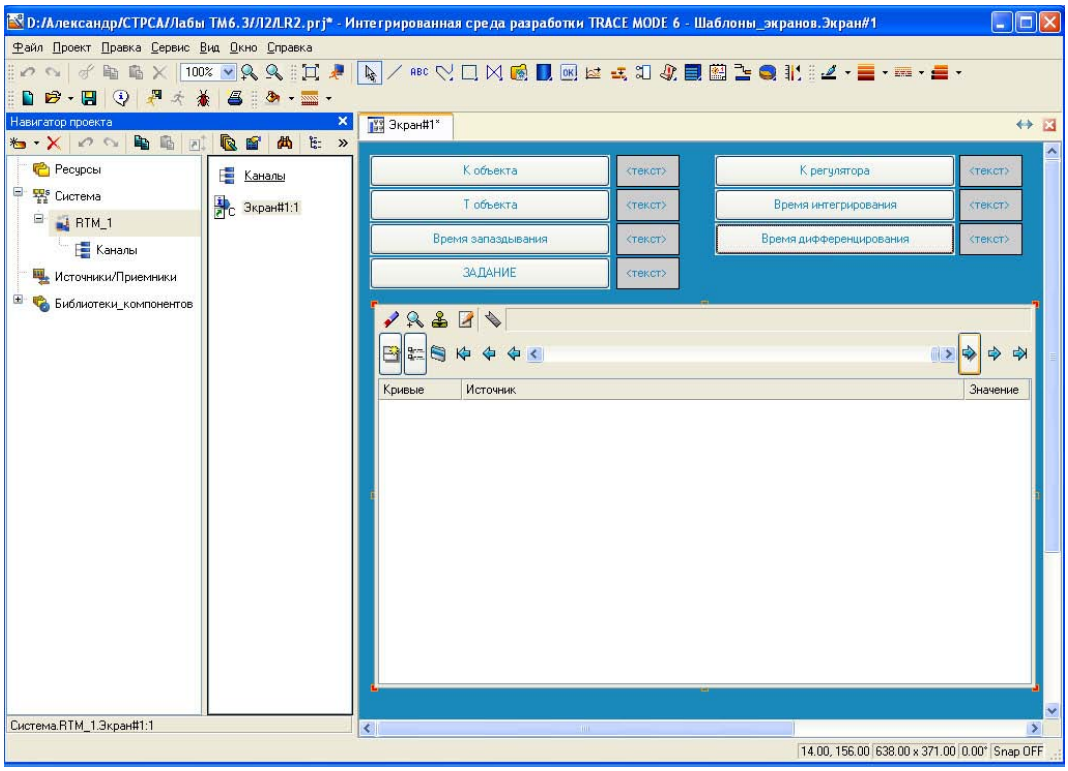


Рисунок 102 – Размещение тренда

Двойным щелчком ЛК мыши в поле тренда открываем окно **Свойства тренда** и переходим на вкладку **Кривые**. Нажимая ПК мыши в поле кривые задаем две кривые Zad и Vihod, а в поле привязка привязываем созданные кривые к заданию и выходу как показано на рисунке 103.

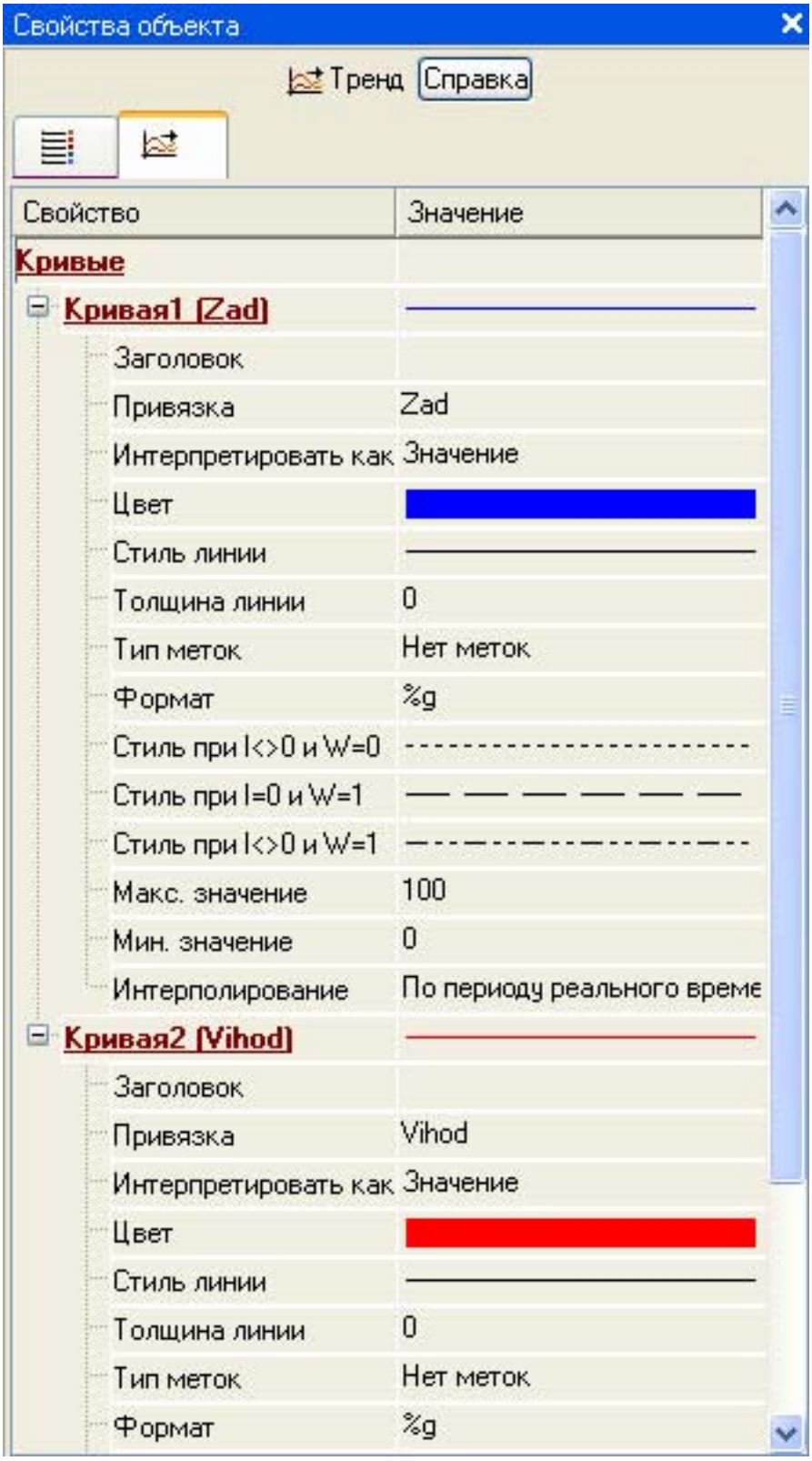


Рисунок 103 – Свойства тренда

3.3.3 Привязка аргумента экрана к каналу

Создадим по аргументам Kob, Tob, Tau, Kr, Ti, Td, Zad, Vihod, шаблона экрана новые каналы и отредактируем их привязку. В слое **Система** открыть узел **RTM_1**. С помощью ПК вызвать контекстное меню свойства компонента **Экран#1** и выбрать команду **Свойства** рисунок 104. Выбрать вкладку Аргументы, ЛК, при нажатой клавише Cntrl, выделить аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad, Vihod и с помощью иконки  создать новые каналы.

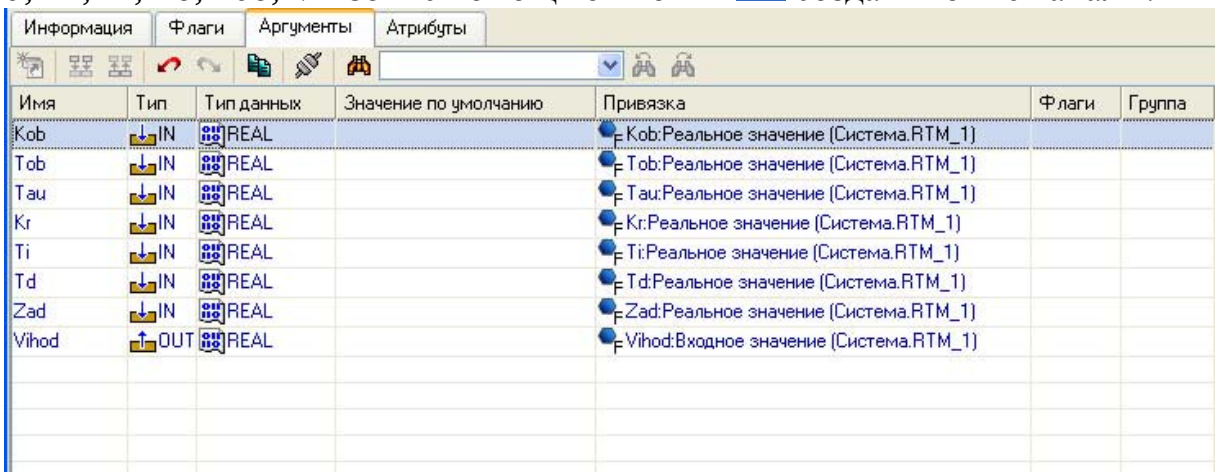


Рисунок 104 – Окно свойств экрана

В результате, в узле **RTM_1**, будут автопостроены следующие каналы рисунок 105.

Рисунок 105 – Автопостроены каналы

3.3.4 Создание программы на языке Техно FBD

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

– разместить необходимые функциональные блоки в рабочем поле FBD-редактора;
– соединить нужные входы и выходы блоков, образовав единую диаграмму;
– задать аргументы, переменные и константы программы;
– привязать входы/выходы FBD-диаграммы к аргументам, переменным и константам программы;
– скомпилировать программу

Создадим программу, которая будет реализовывать одноконтурную систему автоматического регулирования. Для этого ПК по узлу **RTM_1 вызвать контекстное меню выбрать подменю «Создать компонент»** и выбрать ЛК в нем компонент **Программа** рисунок 106.

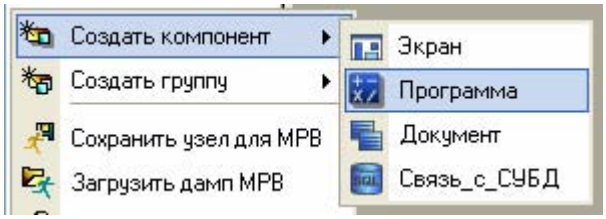


Рисунок 106 – Создание компонента Программа

Выделить компонент Программа#1 и ПК вызвать контекстное меню, выбрав в котором ЛК пункт **Редактировать шаблон**, перейти в режим редактирования программы рисунок 107.

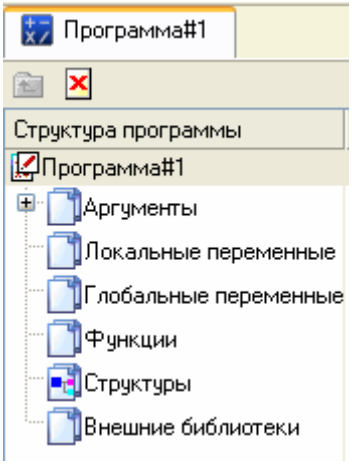


Рисунок 107 – Редактирование аргументов Программы

Выделением ЛК в дереве шаблона **Программа#1** строки **Аргументы** вызвать табличный редактор аргументов иконкой  и создать в редакторе аргументов следующие аргументы программы Kob, Tob, Tau, Kr, Ti, Td, Zad,. При этом аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad должны быть типа **IN**, а Vihod – **OUT** рисунок 108.

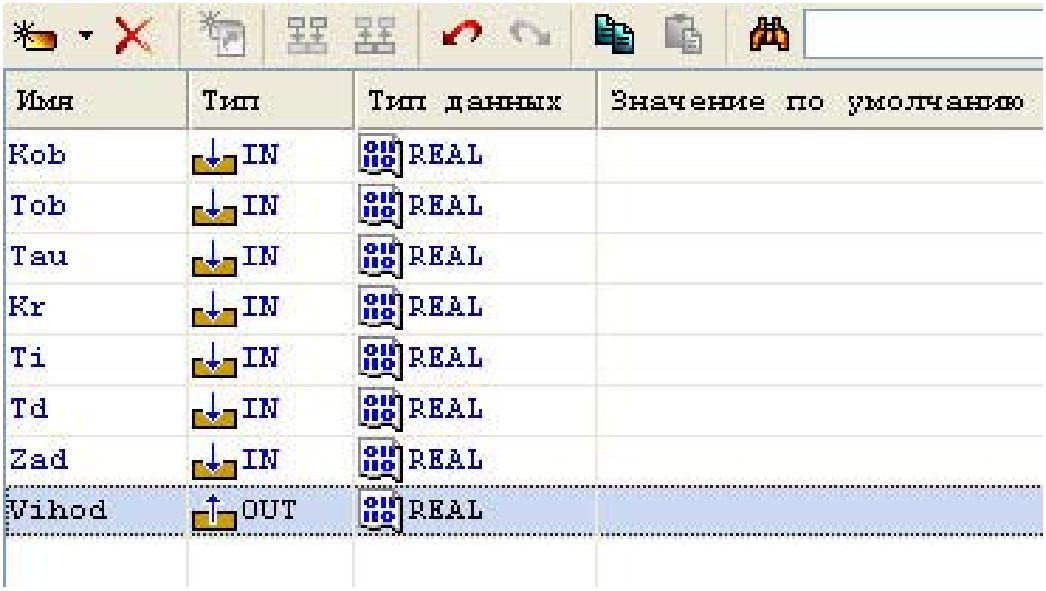


Рисунок 108 – Аргументы программы

Щелкнув ЛК в дереве шаблона строку **Программа#1** и в открывшемся диалоге Выбор языка выбрать язык **FBD** рисунок 109.

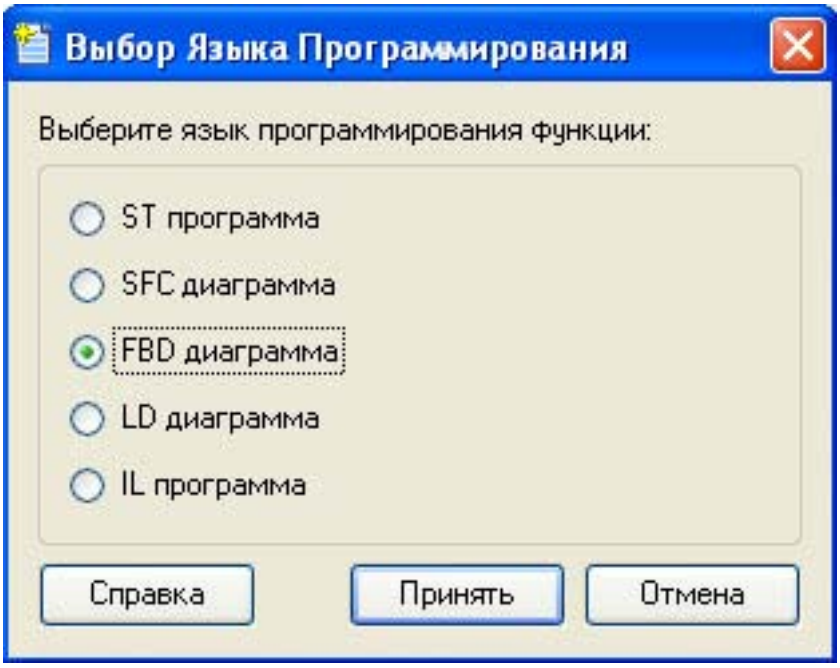


Рисунок 109 – Выбор языка программирования

По нажатию экранной кнопки **Принять** в открывшемся окне редактора программ с объявленными переменными создать программу в соответствии с заданием. Для выбора палитры FBD блоков необходимо ЛК мыши нажать на кнопку  после чего появится окно выбора FBD блоков рисунок 110. При разработки программы верхние входы FBD блоков не используются т.к. они предназначены для изменения порядка пересчета блоков, а информационными входами, являются входы начиная со второго.

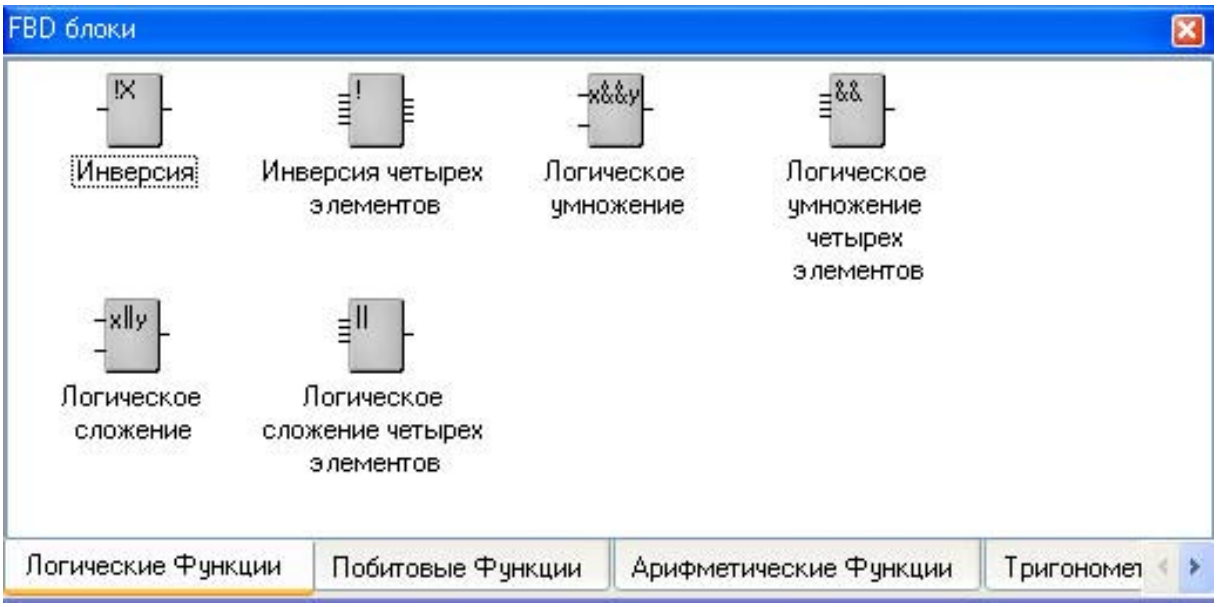


Рисунок 110 – Палитра FBD блоков

Для создания системы управления выберем следующие блоки: из раздела *Арифметические Функции* **FBD блок** вычитание (X–Y); из раздела *Регулирование* **FBD блок** модель объекта (OBJ) и звено PID (PID). После размещения всех блоков программа будет выглядеть следующим образом рисунок 111.

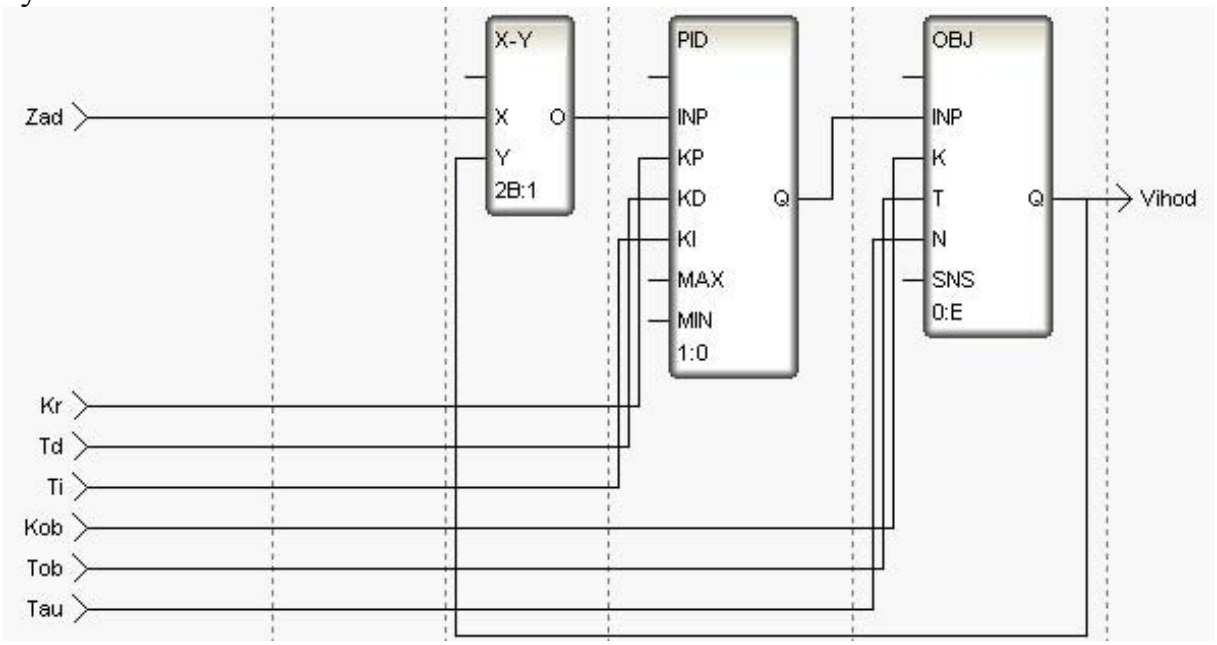


Рисунок 111 – Программа реализации системы управления

С помощью иконки  на инструментальной панели редактора или «горячей клавишей» F7 скомпилировать программу и убедиться в успешной компиляции в окне Выход (Output), вызываемого из инструментальной панели с помощью иконки  рисунок 112.

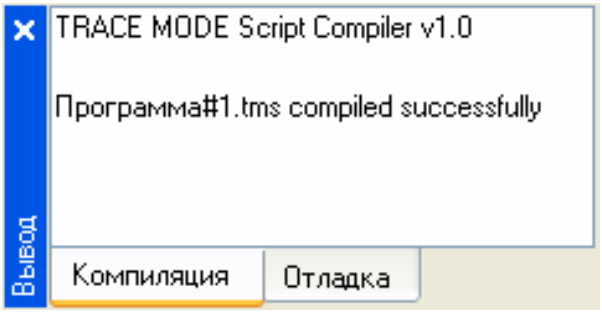


Рисунок 112 – Результат успешной компиляции программы

3.3.4 Привязка аргументов программы

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

Двойным щелчком в поле **Привязка** аргумента программы Y вызвать окно настройки связи, выбрать в левом окне канал класса «Вызов» **Экран#1**, а в правом окне выбрать вкладку **Аргументы** и указать в ней аргумент Vihod и кнопкой **Привязка** подтвердить связь рисунок 113.

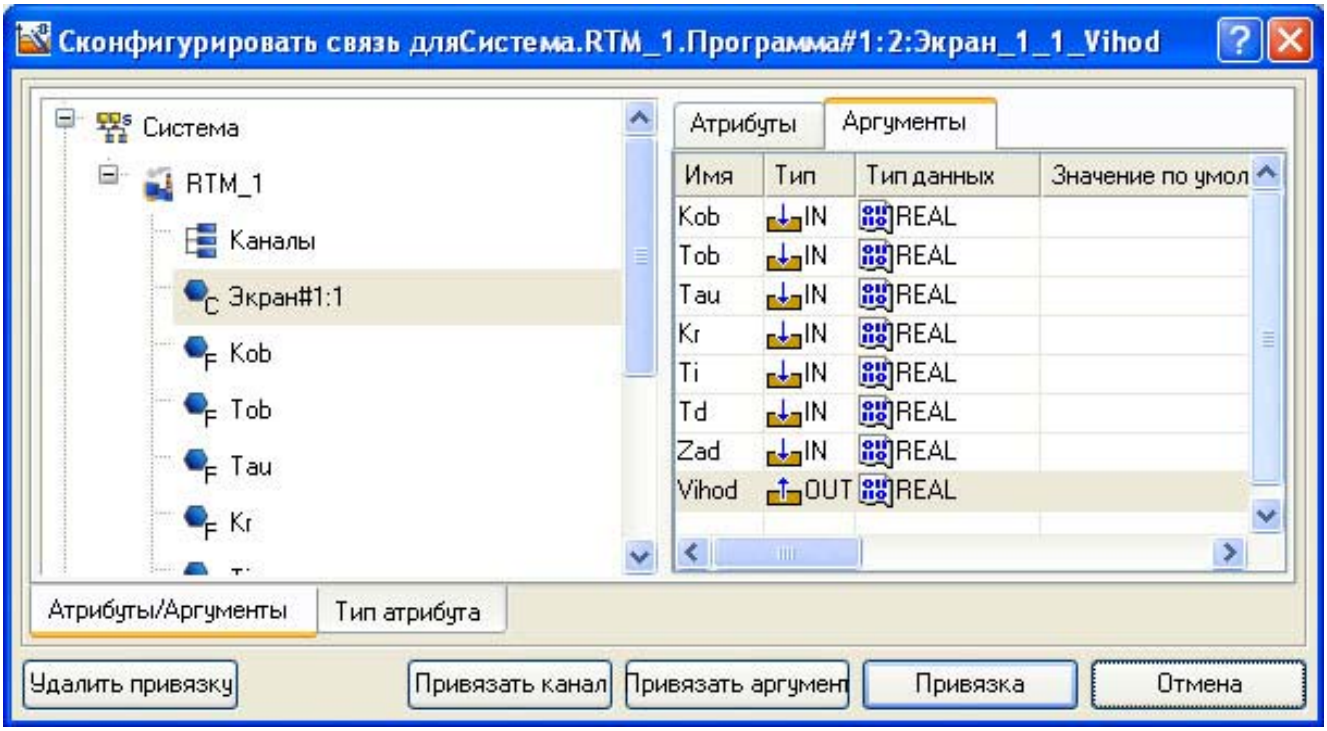


Рисунок 113 – Связь выходных аргументов

Аналогично в поле **Привязка** привязать аргументы программы к атрибутам каналов – аргументы Kob, Tob, Tau, Kr, Ti, Td, Zad к реальному значению каналов Kob, Tob, Tau, Kr, Ti, Td, Zad рисунок 114.

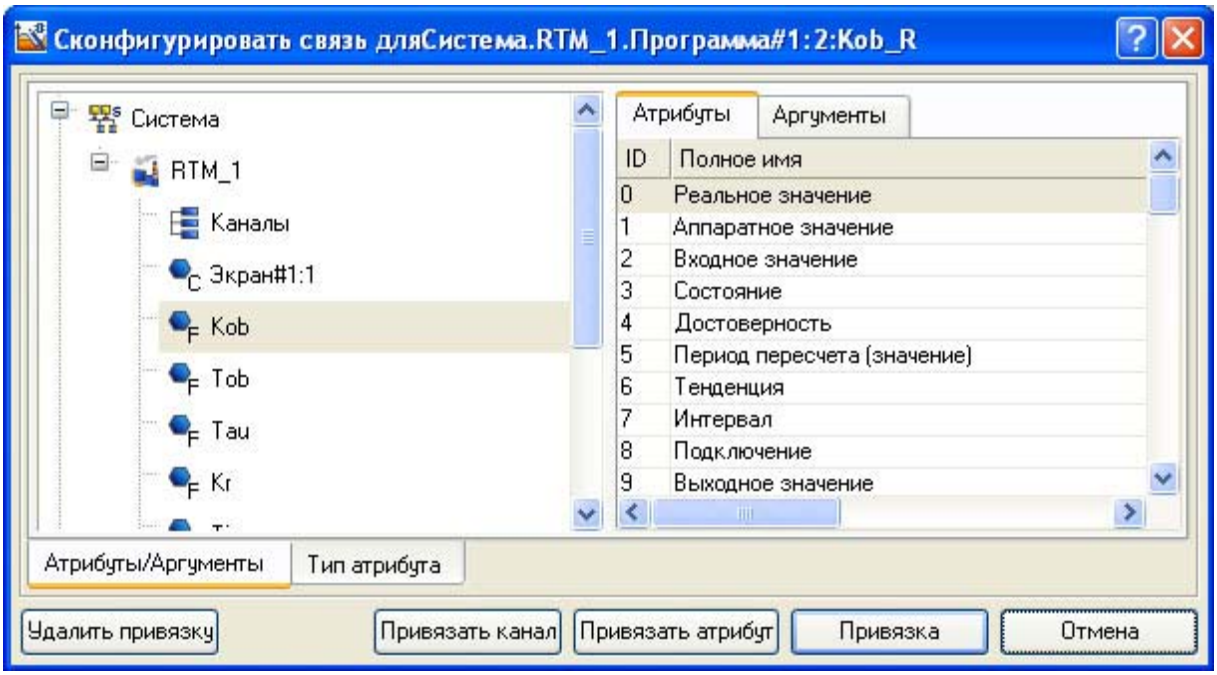


Рисунок 114 – Связь аргументов программы с аргументами экрана

В результате, будем иметь рисунок 115.

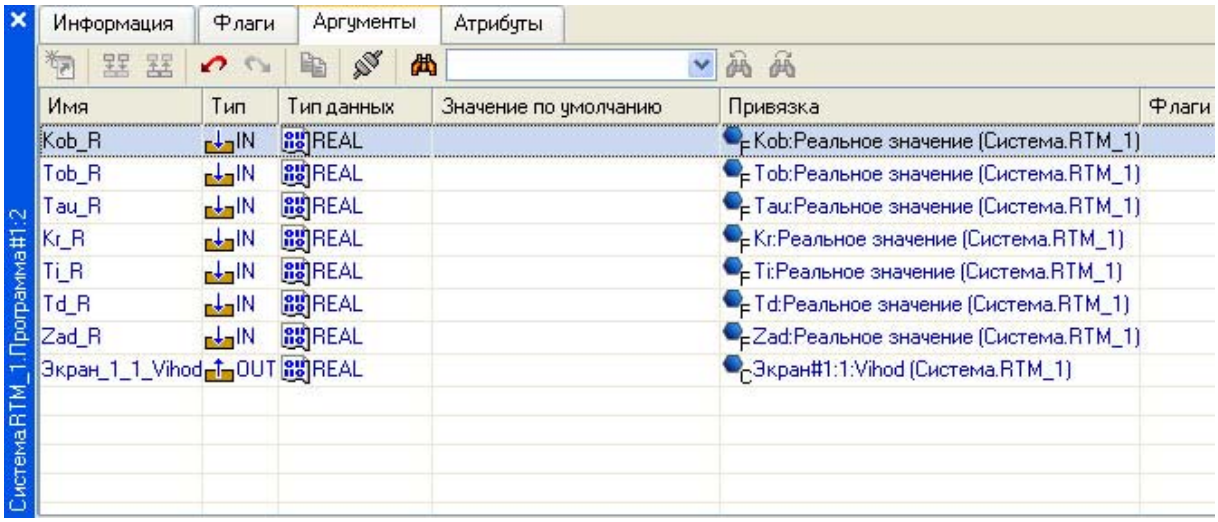


Рисунок 115 – Окончательная настройка связи

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

3.3.5 Запуск проекта

Сохранить проект с помощью иконки . На инструментальной панели выбрать команду (сохранить для МРВ) и скомпилировать тем самым проект для запуска в реальном времени. В навигаторе проекта выделить узел RTM_1 рисунок 116.

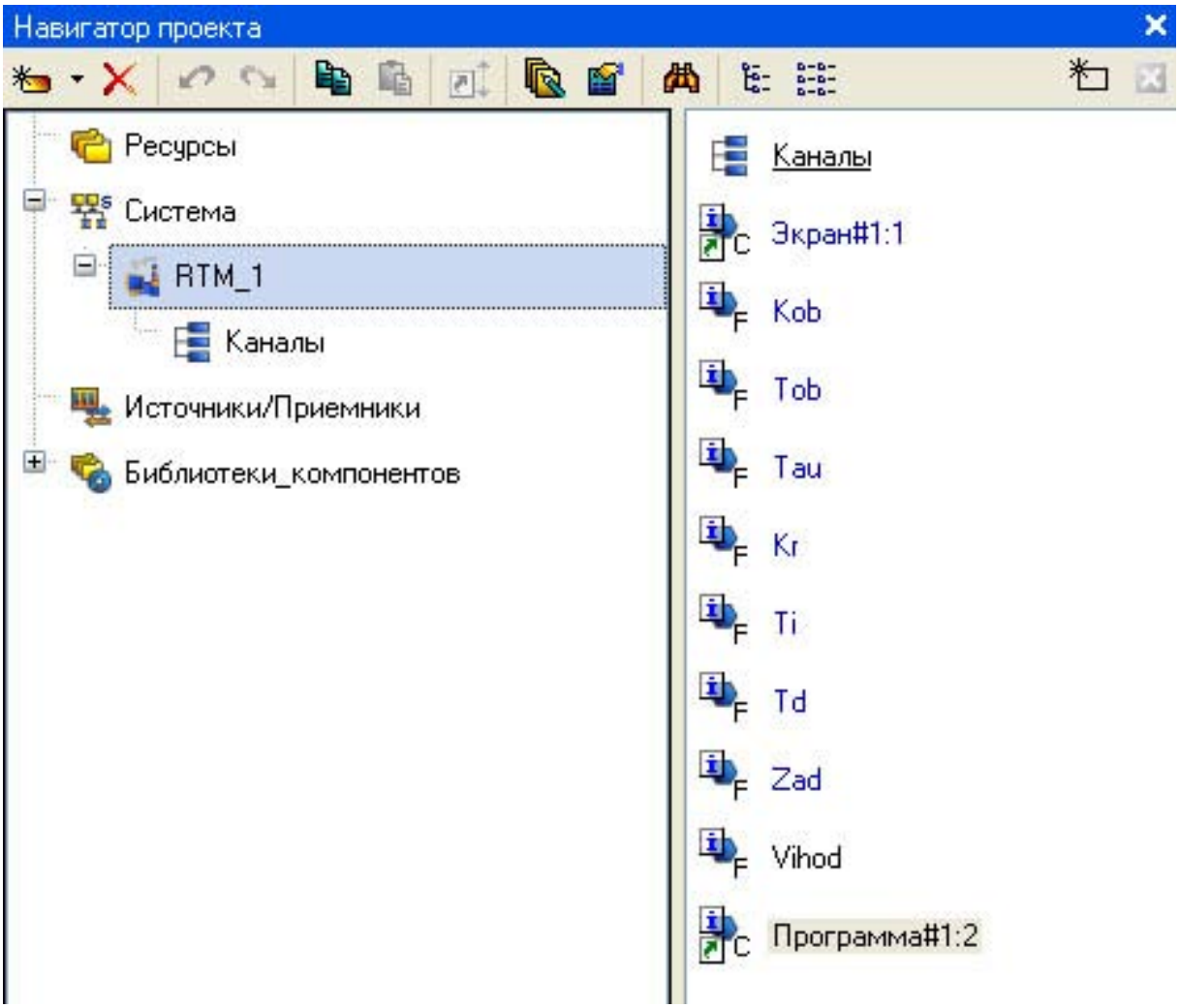


Рисунок 116 – Навигатор проекта

Выбрать иконку  на инструментальной панели и запустить профайлер. В режиме исполнения запустить программу с помощью иконки  .

Для составления отчета необходимо в меню **Файл** выполнить команду **Документировать проект** полученный *.html файл необходимо распечатать.

3.3.6 Задания для лабораторной работы

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

Например: Иванов Сергей Васильевич

И–10

С–19

В–3

Получаем передаточную функцию $W(S)=\frac{10\times e^{-3S}}{19S+1}$

Расчет настроек регулятора производится по формулам

$$Kr=\frac{1.2}{K\frac{\tau}{T}}\quad T_{ИЗ}=2\tau\quad T_{ПР}=0.4\tau$$

$$T_И = \frac{T_{ИЗ}}{Kr} \quad T_Д = T_{ПР} K_r$$

В Trace Mode вместо времени интегрирования вводится коэффициент равный $\frac{1}{Ti}$.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
18	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	

3.3.7 Вопросы для защиты лабораторной работы №3

1. Принцип работы монитора. Канал TRACE MODE 6
2. Обеспечение работы распределенных АСУ
3. Резервирование
4. Автопостроение
5. Математическая обработка данных
6. Архивирование каналов узла
7. Архивирование каналов проекта
8. Отчет тревог и генерация сообщений
9. Графический интерфейс оператора
10. Генерация документов (отчетов)
11. Защита проекта
12. Технология разработки проекта в ИС
13. Классификация компонентов
14. Классификация слоев
15. Классификация узлов
16. Назначение групп источников (приемников)
17. Обработка в канале FLOAT
18. Форматы значений каналов.
19. Понятие масштабирования.
20. Понятие логической обработки.
21. Понятие трансляции.
22. Перечислите основные понятия языка программирования техно-FBD.
23. Типы входа/выхода FBD-блоков.
24. Назначение входов/выходов FBD-блоков используемых в лабораторной работе.
25. Понятие функционального блока.
26. Понятие выполняемой функции.
27. Понятие номера блока.
28. Описание входов и выходов блоков.
29. Понятие пересчета блоков.
30. Описание функциональных блоков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство пользователя. TRACE MODE 6 & T–FACTORY. Быстрый старт. Издание пятое (к релизу 6.03.1). Москва 2006. AdAstrA Research Group, Ltd.–163 с.
2. Руководство пользователя. TRACE MODE 6. Издание седьмое (к релизу 6.03.1). Москва 2006. AdAstrA Research Group, Ltd., том 1–554 с. том 2–598 с.
3. Деменков Н.П. SCADA – системы как инструмент проектирования АСУ ТП. М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 328 с.
4. Управляющие вычислительные комплексы : Учеб. пособие/ Под ред. Н.Л. Прохорова. М.: Финансы и статистика, 2003. –352 с.
5. М.Ю. Рачков Технические средства автоматизации. М.:МГИУ, 2006 – с.185
6. Лапшенков Г.И., Полоцкий Л.М. Автоматизация производственных процессов в химической промышленности. М.: Химия, 1988 – с.296.
7. Сидельников С.И. Синтез комбинационных схем. Новомосковск: НФ РХТУ им. Менделеева, 1993 – 37с.

ПРИЛОЖЕНИЕ А

ПРИМЕР ОФОРМЛЕНИЯ ТИТУЛЬНОГО ЛИСТА ЛАБОРАТОРНОЙ РАБОТЫ

Федеральное агентство по образованию

Российский Химико–Технологический Университет
им. Д. И. Менделеева

Новомосковский институт

Кафедра АПП

Лабораторная работа № 1

по курсу СТРСА

«Построение логических систем управления при помощи SCADA–системы TRACE MODE»

студенты	допуск	выполнение	защита

Группа

Преподаватель

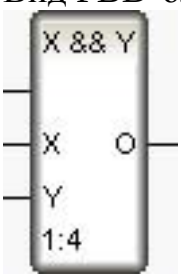
Новомосковск 2006 г.

ПРИЛОЖЕНИЕ Б

ОПИСАНИЕ FBD–БЛОКОВ ИСПОЛЬЗУЕМЫХ ПРИ ВЫПОЛНЕНИИ ЛАБОРАТОРНЫХ РАБОТ

Функциональный блок – это графическое изображение вызова встроенной функции **Техно FBD** (FBD-блока) или функции (функции-блока), определенной пользователем.

Вид FBD-блока показан на следующем рисунке.



В верхней части блока выводится обозначение функции, выполняемой блоком (**X && Y** на рисунке). Именованные отрезки слева (**X и Y**), обозначают входы блока (аргументы, переменные или константы функции). Отрезок без имени слева обозначает вход, управляющий выполнением блока (в дальнейшем – вход **RUN**). Блок выполняется, если **RUN=0** (значение по умолчанию).

Отрезки, примыкающие к блоку справа, обозначают выходы блока (возвращаемые функцией значения).

Кроме входов/выходов, некоторые встроенные **FBD-блоки** имеют внутренние переменные, недоступные пользователю. Переменные **FBD-блока** (входы/выходы и внутренние) являются глобальными, т.е. сохраняют свое значение между вызовами программы, в том числе при **RUN=1**.

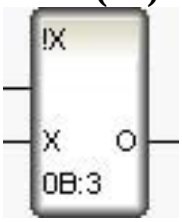
В нижней части блока выводится его номер и, после двоеточия, номер следующего выполняемого блока (**1:4** на рисунке). Номера блоков задаются последовательно при их размещении в рабочем поле редактора; номера следующих выполняемых блоков определяются автоматически при соединении входов и выходов блоков (образовании диаграммы). На блоке, который выполняется первым в программе, после его номера отображается символ B; на блоке, который выполняется последним, – символ E.

FBD-программа может выступать в роли основной программы, функции и функции-блока.

FBD–блоки выполняющие *Логические операции*

На вход блоков этого раздела можно подавать числовые значения, а также значения типа BOOL или STRING. В последнем случае в операции участвует длина строки.

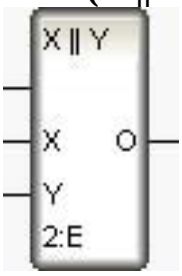
НЕ (!X)



O = NOT X

O=1, если X=0, во всех остальных случаях O=0.

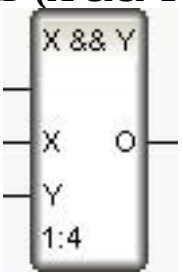
ИЛИ (X || Y)



O = X OR Y

O=0, если одновременно X=0 и Y=0, во всех остальных случаях O=1.

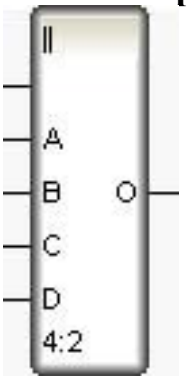
И (X && Y)



O = X AND Y

O=1, если X и Y одновременно отличны от нуля, во всех остальных случаях O=0.

Логическое сложение четырех элементов (||)



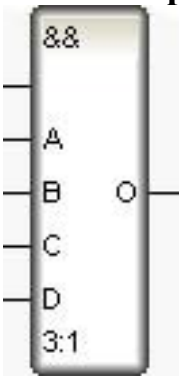
O = A OR B OR C OR D

O=1, если хотя бы один из входов отличен от нуля.

O=0, если A=B=C=D=0.

Если вход не определен, его значение принимается равным 0.

Логическое умножение четырех элементов (&&)



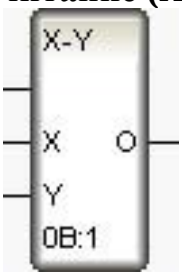
O = A AND B AND C AND D

O=1, если все входы одновременно отличны от 0, во всех остальных случаях O=0.

Если вход не определен, его значение принимается равным 0.

FBD–блок выполняющий *Арифметические операции*

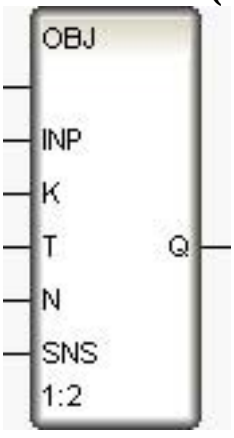
Вычитание (X-Y)



O = X – Y

FBD–блоки выполняющие *Операции регулирования*

Модель объекта (OBJ)



Данный блок моделирует объект управления для отладки алгоритмов регулирования или подготовки демонстрационных проектов. Он представляет собой комбинацию апериодического (инерционного) звена первого порядка и

звена запаздывания, т.е. передаточная функция блока (при входной функции e^{st}) имеет вид:

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

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

Кроме того, на выходной сигнал блока можно наложить помеху в виде случайной составляющей, синусоидального сигнала или случайных бросков. Здесь же можно задать случайное колебание динамических характеристик объекта.

Входным по отношению к моделируемому объекту является вход **INP**. Входы **K**, **T** и **N** используются для задания соответственно коэффициента усиления, постоянной времени и времени запаздывания. Последние два параметра задаются в тактах пересчета, максимальное значение времени запаздывания – 4.

Вход **SNS** предназначен для управления случайными помехами, вносимыми в работу объекта. Значение 1 отдельных битов этого входа включает следующие помехи:

–1 бит – добавление к выходному сигналу случайной величины в диапазоне от 0 до 1%;

–2 бит – формирование пика величиной 25% от значения выхода с вероятностью 0,01;

– 3 бит – добавление к выходу синусоидального сигнала с амплитудой 2% от значения выхода;

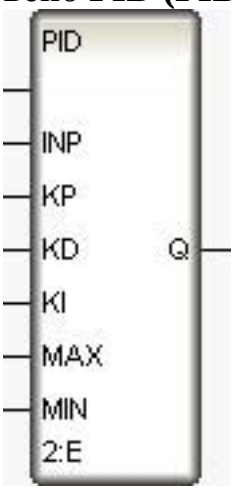
– 5 бит – случайное увеличение коэффициента усиления в диапазоне от 0 до 2%;

–6 бит – случайное увеличение постоянной времени в диапазоне от 0 до 2%;

–7 бит – случайное изменение на 1 запаздывание.

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

Звено PID (PID)



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

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

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

Модуль подаваемого на вход **KI** отрицательного значения передается на выход. Далее при подаче на вход **KI** неотрицательного значения регулирование начинается с установленной величины.

Для ограничения величины управляющего воздействия используются входы блока **MIN** и **MAX**. Если величина управления меньше **MIN**, то **Q = MIN**, если величина управления больше **MAX**, то **Q = MAX**, при этом в обоих случаях накопление интегральной составляющей закона регулирования прекращается.

Данный блок вычисляет величину управляющего воздействия по значению рассогласования регулируемой величины и задания, которое предварительно надо вычислять с помощью блока **X-Y**.

Введение в алгоритм параметра Δt исключает необходимость пересчета настроек регулятора при смене периода пересчета.

ПРИЛОЖЕНИЕ В
ГЛОССАРИЙ

АРМ – автоматизированное рабочее место
АСУ – автоматическая система управления
АСУП – АСУ предприятием
АСУ ТП – АСУ технологическим процессом
АЦП – аналого–цифровой преобразователь
БД – база данных
МРВ – монитор реального времени
ОЗУ – оперативное запоминающие устройство
ПЗУ – постоянно запоминающие устройство
ПК – промышленный компьютер
ПЛК – программируемый логический контроллер
ПО – программное обеспечение
УВК – управляющий вычислительный комплекс
УСО – устройство связи с объектом
ЦАП – цифроаналоговый преобразователь
Active X – это объекты, в основе которых лежит Microsoft COM
COM – (Component Object Model) – модель многокомпонентных объектов
DCOM – (Distributed COM) – распределенная COM
FBD – (Function Block Diagram) – язык функциональных блоковых схем
GUI – (Graphic Users Interface) – графический пользовательский интерфейс
HMI – (Human Machine Interface) – человеко–машинный интерфейс
IL – (Instruction List) – язык инструкций
LAN – (Local Area Network) – локальная вычислительная сеть
LD – (Ladder Diagram) – язык лестничных диаграмм или релейно-контактных схем
MRP – (Manufacturing Resource Planning) – планирование ресурсов производства
ERP – (Enterprise Resource Planning) – планирование ресурсов предприятия
OLE – (Object Linking and Embedding) – связывание и внедрение объектов
OPC – (OLE for Process Control) – OLE для управления процессами
ROC – (Rate of Change) – скорость изменения
RTM – (Real Time Monitor) – монитор реального времени
SCADA – (Supervisory Control And Data Acquisition) – контроль управления и контроль сбора данных
SFC – (Sequential Function Chart) – язык последовательных функциональных блоков
ST – (Structured Text) – язык структурированного текста

Учебно–методическое издание

ЛОПАТИН Александр Геннадиевич
КИРЕЕВ Павел Анатольевич

Методика разработки систем управления на базе SCADA системы TRACE MODE

Редактор
Лицензия ЛР № 020714 от 02.02.98
Подписано в печать . . . Формат 60x84 1/16
Бумага типографская №2. Отпечатано на ризографе
Усл. печ. л. . Уч.- изд. л. . Тираж 50 экз.
Заказ № .

Российский химико-технологический университет
им. Д. И. Менделеева
Новомосковский институт. Издательский центр
Адрес университета:125047, Москва, Миусская пл., 9
Адрес института: 301670, Новомосковск, Тульской обл., Дружбы 8