

Министерство образования Республики Беларусь

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

Кафедра «Промышленная электроника»

А. В. Ковалев, Д. А. Литвинов

УПРАВЛЕНИЕ ПРОМЫШЛЕННЫМИ ОБЪЕКТАМИ

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименному курсу
для студентов специальности 1-36 04 02
«Промышленная электроника»
дневной и заочной форм обучения
В двух частях
Часть 1**

Гомель 2014

УДК 004(075.8)
ББК 32.965я73
К56

*Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(протокол № 12 от 24.06.2013 г.)*

Рецензент: зав. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *К. С. Курочка*

Ковалев, А. В.

К56 Управление промышленными объектами : лаборатор. практикум по одноим. курсу для студентов специальности 1-36 04 02 «Промышленная электроника» днев. и заоч. форм обучения : в 2 ч. Ч. 1. / А. В. Ковалев, Д. А. Литвинов. – Гомель : ГГТУ им. П. О. Сухого, 2014. – 183 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: [http:// library.gstu.by/StartEK/](http://library.gstu.by/StartEK/). – Загл. с титул. экрана.

Дается краткое описание и основные понятия системы TRACE MODE и DIRECT SOFT. Рассматривается интегрированная среда разработки проекта, создание графических интерфейсов и мнемосхем, программная обработка на промышленных языках программирования и обмен данными.

Для студентов специальности 1-36 04 02 «Промышленная электроника» дневной и заочной форм обучения.

УДК 004(075.8)
ББК 32.965я73

© Учреждение образования «Гомельский
государственный технический университет
имени П. О. Сухого», 2014

\СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	1
ВВЕДЕНИЕ	2
Лабораторная работа №1	7
Лабораторная работа №2	46
Лабораторная работа №3	74
Лабораторная работа №4	99
Лабораторная работа №5	112
Список литературы	181

ВВЕДЕНИЕ

1. ПОНЯТИЕ SCADA–СИСТЕМЫ

1.1 SCADA система TRACE MODE

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

SCADA-система TRACE MODE разработана российской фирмой AdAstra Research Group, LTD. Последний на данный момент продукт - это 5-й релиз 6-й версии TRACE MODE, который содержит полный набор программных средств для создания АСУТП и АСУП. SCADA-система TRACE MODE содержит средства разработки операторского интерфейса (SCADA/HMI). программирования контроллеров (Softlogic). управления основными фондами (EAM). персоналом (HRM) и производственными процессами (MES).

TRACE MODE 6 содержит множество библиотек ресурсов, готовых к использованию в прикладных проектах, бесплатные драйверы к более чем 1600 контроллерам и платам ввода/вывода, свыше 600 анимационных объектов, более 150 алгоритмов обработки данных и управления, комплексные технологические объекты.

1.2 Архитектура TRACE MODE

Все программы, входящие в TRACE MODE, подразделяются на две группы (рисунок 1.1): **инструментальную систему разработки и исполнительные модули**. Интегрированная среда разработки (ИС) – единая программная оболочка, содержащая все необходимые средства для разработки проекта. ИС включает в себя три редактора: *редактор базы каналов, редактор представления данных, редактор шаблонов*.

В редакторе базы каналов создается математическая основа системы управления: описываются конфигурации всех рабочих станций, контроллеров и УСО. а также настраиваются информационные потоки между ними. Здесь же описываются входные и выходные сигналы и их связь с устройствами сбора данных и управления: задаются периоды опроса или формирования сигналов, настраиваются законы

первичной обработки и управления, технологические границы, программы обработки данных и управления, осуществляется архивирование технологических параметров, сетевой обмен, а также решаются некоторые другие задачи.

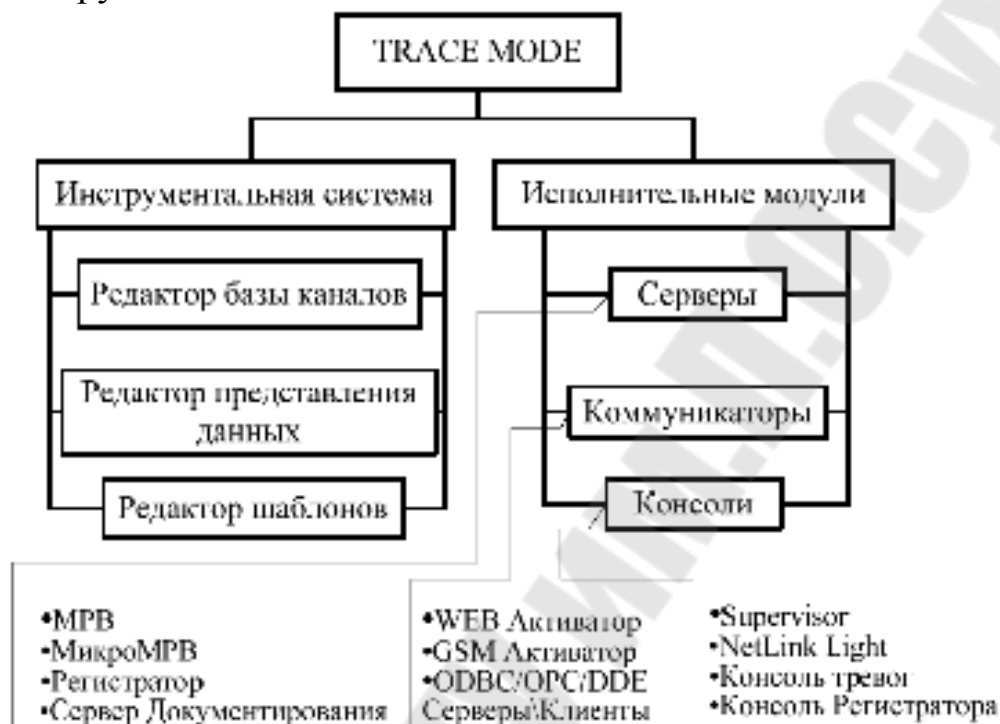


Рисунок – 1.1 Структурная схема TRACE MODE

Результатами работы в этом редакторе являются математическая и информационная структуры проекта АСУТП, которая включает набор баз каналов и файлов конфигурации для всех контроллеров и операторских станций (узлов) проекта, а также файл конфигурации всего проекта с расширением **cmt** (для версии 6 расширение - **prj**). Все остальные файлы проекта хранятся в рабочей директории в каталоге, имя которого совпадает с именем файла конфигурации.

В редакторе представления данных разрабатывается графическая часть проекта системы управления. Создается статичный рисунок технологического объекта, поверх которого размещаются динамические формы отображения и управления.

Основой **исполнительной системы** TRACE служат – мониторы (MPB). **Монитор** – программные модули различного назначения, под управлением которых в реальном времени выполняются составные части проекта, размещаемые на отдельных компьютерах или в контроллерах, предназначенные для работы на всех уровнях систем управления. Существует ряд программных модулей, назначение кото-

рых четко не привязано к функциям одного из перечисленных уровней систем управления. К таким модулям относятся: глобальный регистратор, сервер документирования, Web-активатор, GSM-активатор.

Глобальный регистратор предназначен для обеспечения надежного хранения архивов техпроцесса (ТП). Он архивирует данные, посылаемые ему по сети мониторами реального времени (64 000 параметров с дискретностью до 0.001 с), обеспечивает автоматическое восстановление данных после сбоя, а также может передавать архивные данные для просмотра мониторам. Глобальный регистратор может также выступать как OPC-сервер и DDE-сервер и поддерживает обмен с базами данных через ODBC.

Для документирования технологической информации в TRACE MODE используется специальный модуль – сервер документирования. Документирование осуществляется по шаблонам, создаваемых пользователем. Обычно подготовка отчетов синхронизируется с астрономическим временем.

Утилита **консоль тревог** предназначена для просмотра отчетов тревог разных МРВ одного проекта.

Любая рабочая станция системы TRACE MODE может выступать в качестве Web-сервера, что позволяет управлять технологическим процессом через Интернет (Internet). На удаленном компьютере необходимо иметь только доступ к сети Интернет и Web-браузер. Реализуется данный режим модулем Web-активатор.

Связь с серверами реального времени TRACE MODE также может осуществляться с помощью средств сотовой связи стандарта GSM, инфракрасный порт, сеть на основе интерфейса RS-232/485 или модем с использованием высоконадежного протокола TCP/IP.

Для обеспечения мобильных пользователей АСУ оперативной информацией в режиме реального времени используется - **GSM-активатор**. Он предназначен для дистанционного мониторинга и управления технологическими процессами, а также для получения оперативной информации. В реальном времени GSM-активатор может принимать информацию от 64 000 датчиков, осуществлять супервизорное управление, получать технико-экономическую информацию из баз данных через сервер, использующий стандартные интерфейсы SQL/ODBC, OPC, DDE и т. д. Вся входящая информация отображается графически в виде анимированных мнемосхем и трендов. GSM-активатор также можно применять в охранных службах.

В TRACE MODE 6 все редакторы системы вызываются из одной программы - **Интегрированной среды разработки (ИС)**. ИС - единая программная оболочка, содержащая все необходимые средства для разработки проекта.

Все переменные проекта, к чему бы они ни относились - к контроллеру, к операторской станции, к управлению техобслуживанием или производством хранятся в **единой базе данных проекта**. Логическая структура проекта полностью отделена от аппаратной части. Благодаря единому пространству распределенных переменных, переменные из разных узлов могут связываться между собой также легко, как и в пределах одного узла, любые изменения, вносимые в объект, автоматически применяются везде, где он был задействован.

1.3 Основные понятия SCADA-систем TRACE MODE

Проект системы управления - это совокупность всех математических и графических элементов системы, функционирующих на различных операторских станциях и контроллерах одной АСУ ТП, объединенных информационными связями и единой системой архивирования. 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 является создание файлов, содержащих информацию об алгоритмах работы АСУ. Эти файлы размещаются на аппаратных средствах (компьютерах и контроллерах) и выполняются под управлением исполнительных модулей TRACE MODE (мониторов).

Составная часть проекта, размещаемая на отдельном компьютере или в контроллере и выполняемая под управлением одного или нескольких исполнительных модулей TRACE MODE, называется **узлом проекта**. **УЗЕЛ** - любое устройство в рамках проекта, в котором запущено программное обеспечение TRACE MODE, реализующее серверные функции. Это может быть контроллер, операторская или архивная станция. В проекте может быть до 128 узлов. 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.

БАЗА КАНАЛОВ - совокупность всех каналов, математических объектов, FBD-программ и ПЛ-программ, созданных для каждого конкретного узла.

ОБЪЕКТ БАЗЫ КАНАЛОВ - совокупность любых каналов, которой приписан определенный набор свойств и атрибутов. Оформленные группы каналов могут быть подчинены друг другу и создавать таким образом иерархические структуры.

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

Лабораторная работа №1

Изучение среды TRACE MODE 6. Создание простого проекта

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

1. Теоретические сведения

1.1 Разработка графического интерфейса оператора

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

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

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

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

1.2 Классификация компонентов TRACE MODE

По функциональному назначению компоненты проекта можно разделить на виды:

- **каналы** – компоненты, определяющие алгоритм работы проекта;
- **шаблоны** – компоненты, которые при работе в реальном времени могут вызываться каналами с передачей параметров (аналог подпрограмм). Передача параметров настраивается при разработке проекта в ИС посредством привязки аргументов шаблона к каналам или источникам/приемникам;
- **источники/приемники** – шаблоны каналов обмена с различными устройствами и приложениями. Под устройствами здесь понимаются контроллеры, а также внешние и внутренние модули/платы различного назначения, обмен с которыми поддерживается мониторами TRACE MODE (в том числе через драйверы). Системные переменные TRACE MODE и встроенные генераторы также создаются в ИС как источники/приемники;
- **наборы ресурсов** – наборы текстов, изображений и видеоклипов, которые могут быть использованы при разработке шаблонов графических экранов;
- **графические объекты** – компоненты, представляющие собой в общем случае несколько графических элементов (из имеющихся в редакторе представления данных), сгруппированных в один. Графические объекты могут быть использованы при разработке шаблонов графических экранов;
- **последовательные порты** – параметры СОМ-портов;
- **словари сообщений** – наборы сообщений, генерируемых при возникновении различных событий;
- **клеммы** – эти компоненты, описывающие электрические контакты (например, монтажных шкафов), являются элементами схемы электрических соединений АСУ.

1.2.1 Каналы

По функциональному назначению классы каналов можно сгруппировать следующим образом:

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

1.2.2 Шаблоны

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

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

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

1.3 Классификация слоев

Предопределенные слои структуры проекта имеют следующее назначение:

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

- может играть роль узла – для него определена команда Сохранить узел для МРВ. Кроме того, для этого слоя определены команды взаимодействия с технологической базой данных;
- **Топология** – для разработки проекта от топологии (т.е. с группировкой компонентов по месту расположения);
 - **КИПиА** – для описания электрических соединений АСУ;
 - **Библиотеки компонентов** – для создания **библиотек объектов** – проектных решений отдельных задач. Этот слой содержит предопределенные группы «**Системные**» и «**Пользовательские**». В группе «**Системные**» содержатся библиотеки, подключенные к ИС по умолчанию.

1.4 Классификация узлов

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

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

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

1.5 Технология разработки проекта в ИС

Разработка проекта в ИС включает следующие этапы:

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

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

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

1.6 Интегрированная среда разработки

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

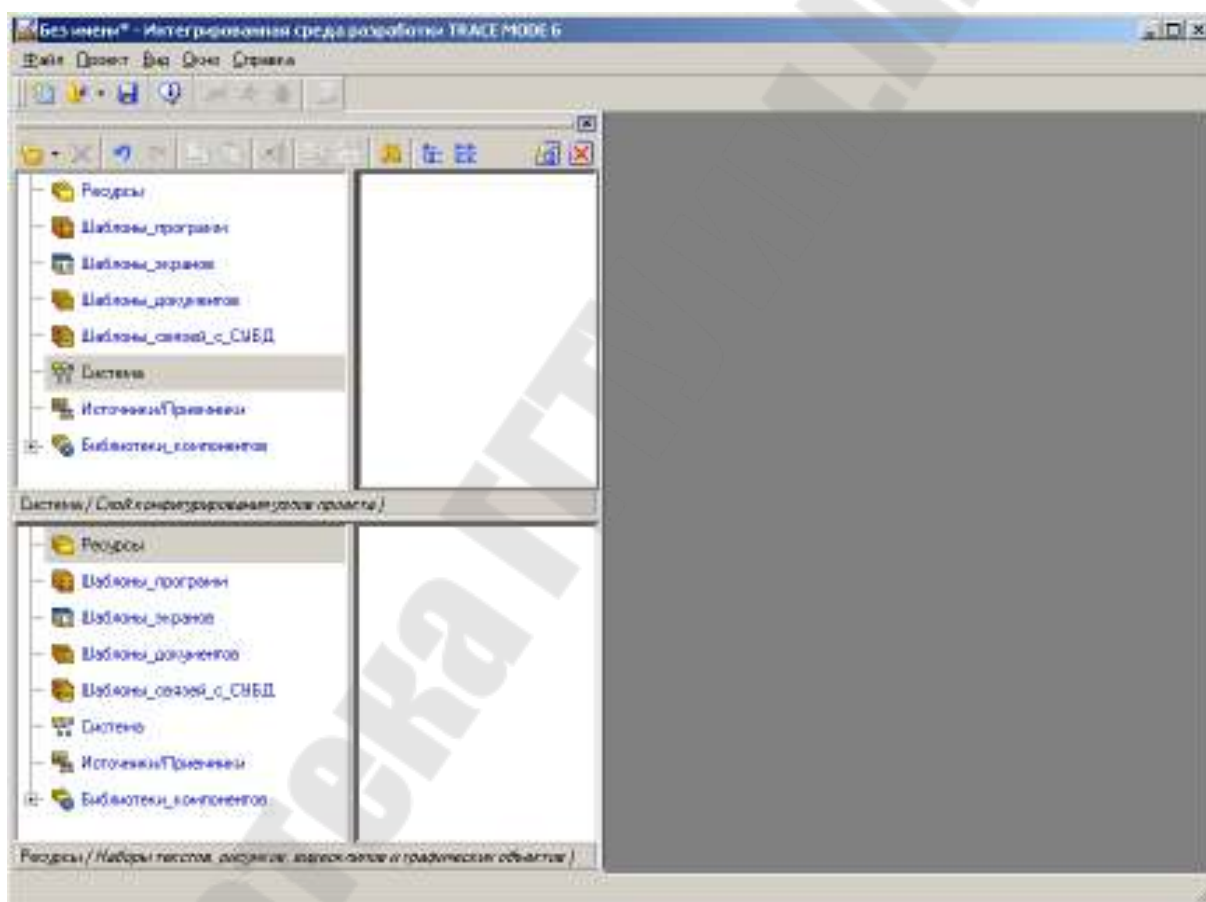


Рисунок 1.1 – Вид окна ИС разработки проекта

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

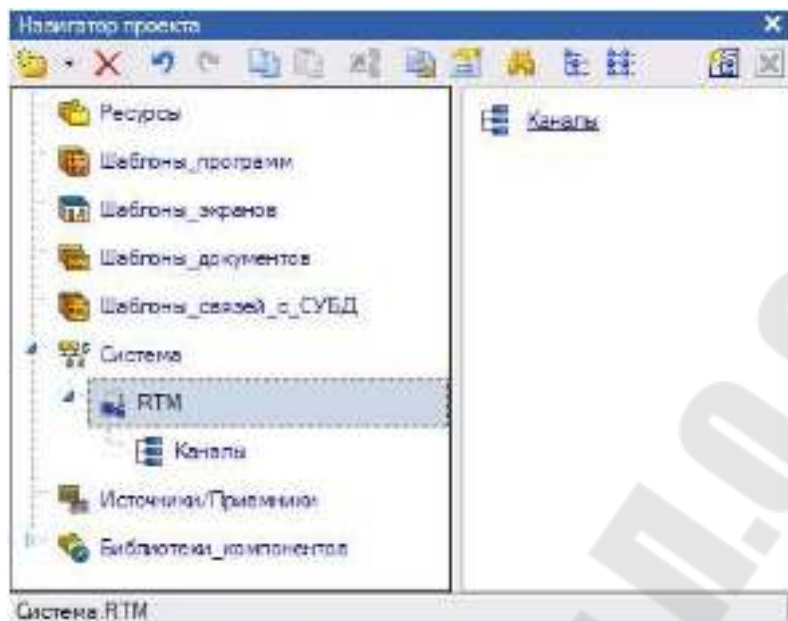




Рисунок 1.2 – Вид окна навигатора структуры проекта

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

1.7 Редактирование структуры проекта

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

1.8 Создание и редактирование свойств объектов структуры

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

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


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



Рисунок 1.3 – Вид окна свойств объекта канал, структуры проекта

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

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

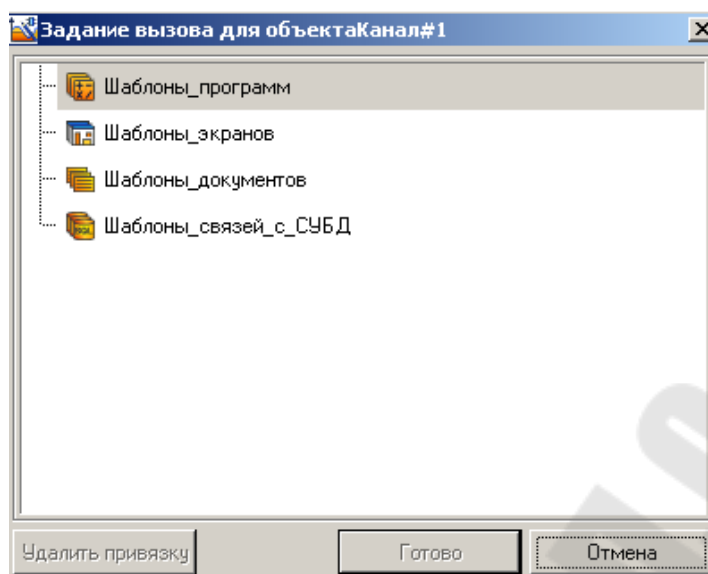


Рисунок 1.4 – Вид окна конфигурации свойства **вызов**

1.8 Аргументы. Табличный редактор аргументов

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

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

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

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

1.8 Поля редактора аргументов

Табличные редакторы аргументов имеют следующий набор полей:

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

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

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


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

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

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

Принадлежность - зарезервировано;

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

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

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

Комментарий - комментарий разработчика к аргументу.

1.9 Разновидности редактора аргументов

1.9.1 Редактор аргументов шаблона

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

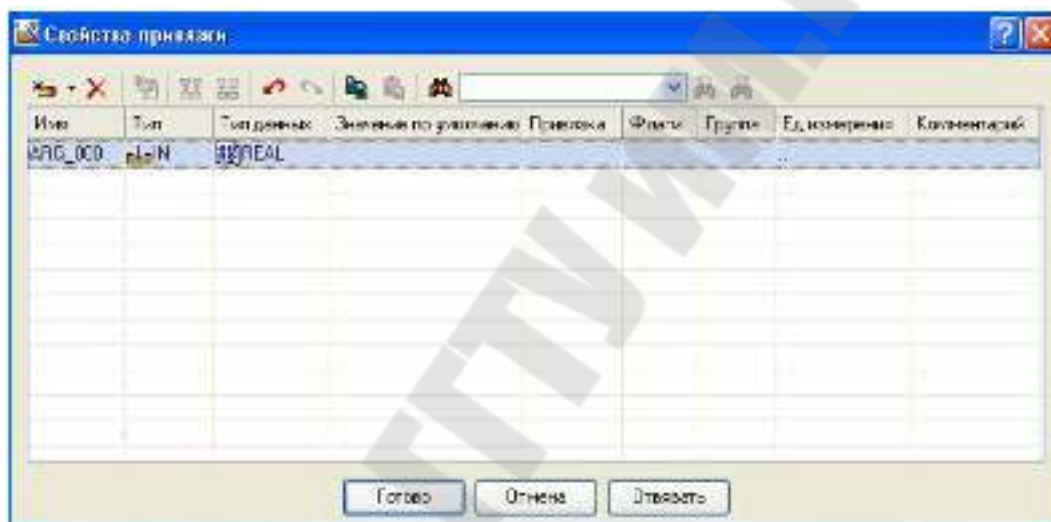



Рисунок 1.5 – Вид окна редактора аргументов шаблона


При задании имен аргументов шаблона нужно учитывать следующее:

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

1.9.2 Редактор аргументов канала CALL с ненастроенным свойством ВЫЗОВ

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

1.9.3 Редактор аргументов компонента с настроенным свойством ВЫЗОВ

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

2. Ход работы

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

2.1 Создание узла АРМ


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



Рисунок 2.1 – Интегрированная среда разработки

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

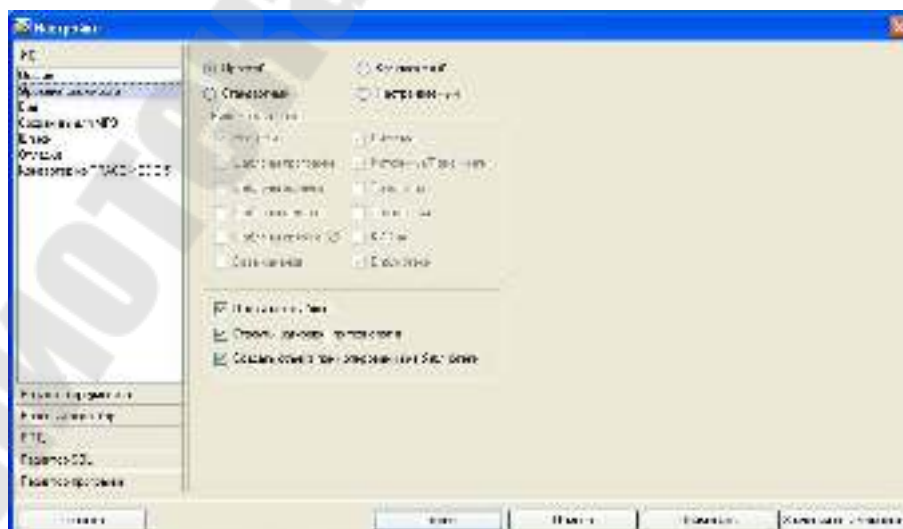


Рисунок 2.2 – Настройки уровня сложности проекта

Уровни сложности:

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

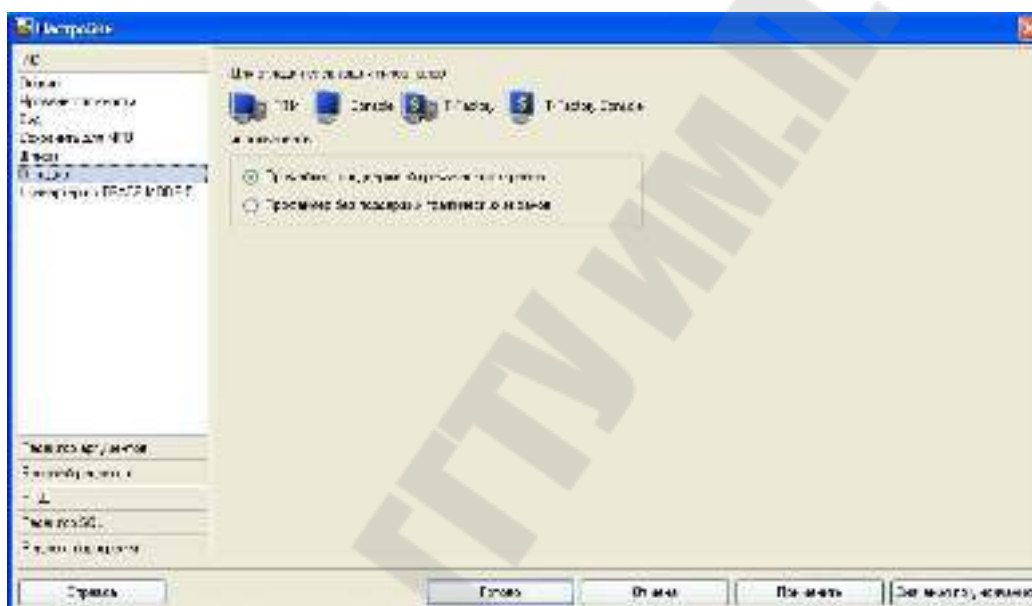



Рисунок 2.3 – Настройки отладки узлов проекта

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

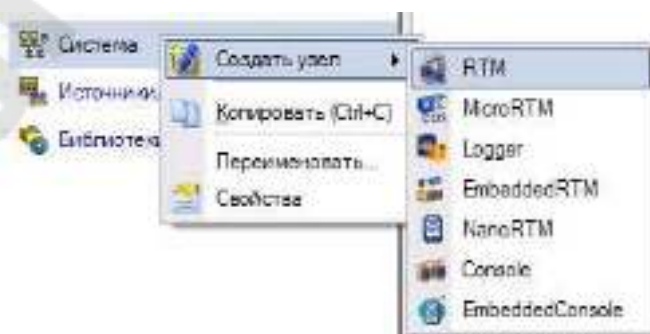


Рисунок 2.4 – Создание нового узла

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

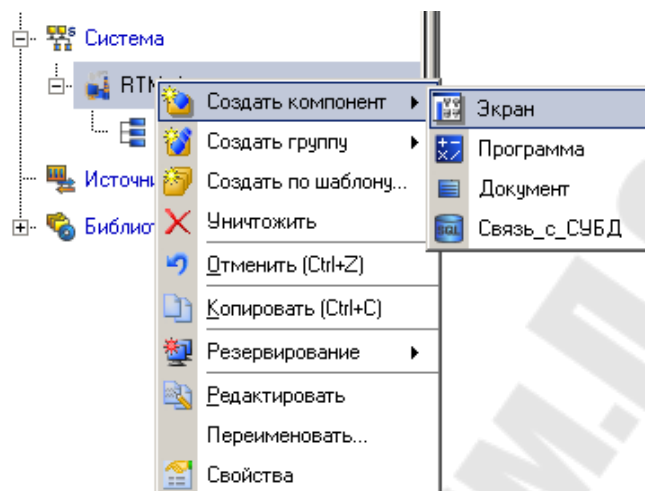



Рисунок 2.5 – Создание шаблона экрана

2.2 Создание графического экрана

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

2.2.1 Создание статического текста

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

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

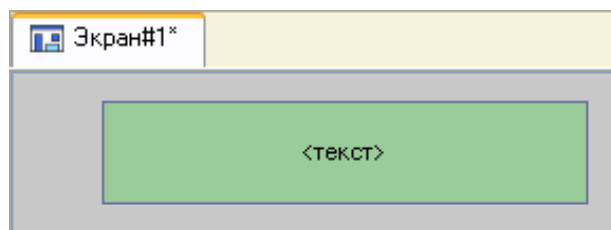



Рисунок 2.6 – Графический элемент «Текст»

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

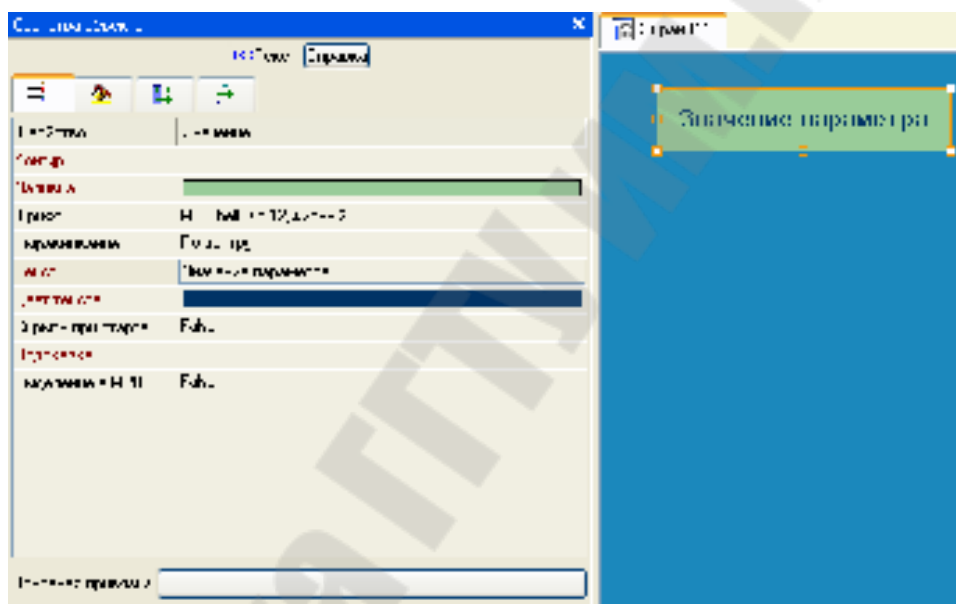


Рисунок 2.7 – Графический элемент «Текст»

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

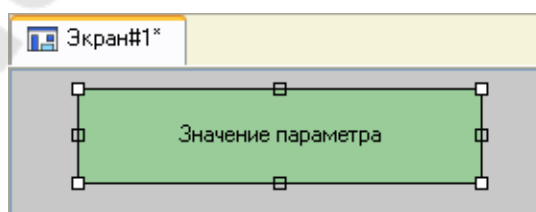


Рисунок 2.8 – Графический элемент «Текст»

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

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

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

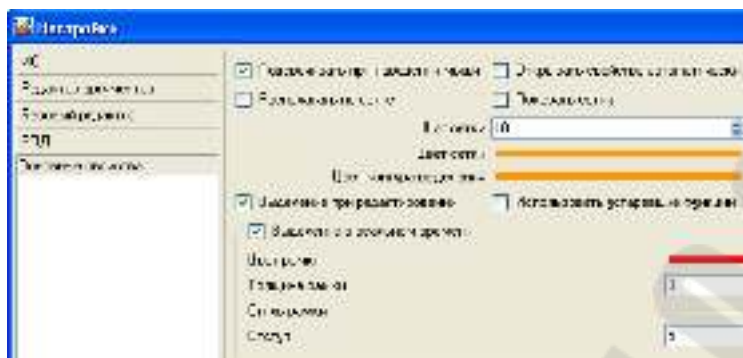


Рисунок 2.9 – Окно настройки автоматического вывода свойств ГЭ

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

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

- создадим и разместим новый ГЭ ABC справа от ГЭ с надписью «Значение параметра»;
- откроем свойства вновь размещенного ГЭ;
- двойным щелчком ЛК на строке «Текст» вызовем меню «Вид индикации» (рисунок 2.10);

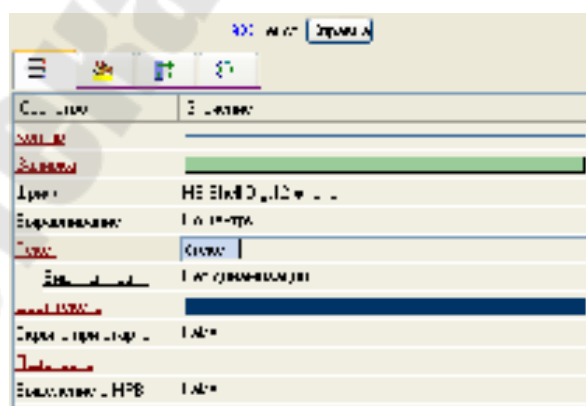


Рисунок 2.10 – Вид окна настройки

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

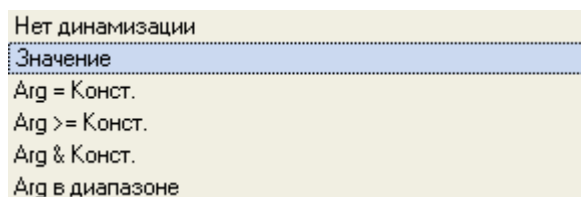



Рисунок 2.11 – Вид окна

- в открывшемся меню настройки параметров динамизации щелчком ЛК в правом поле строки **Привязка**;



Рисунок 2.12 – Вид окна

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

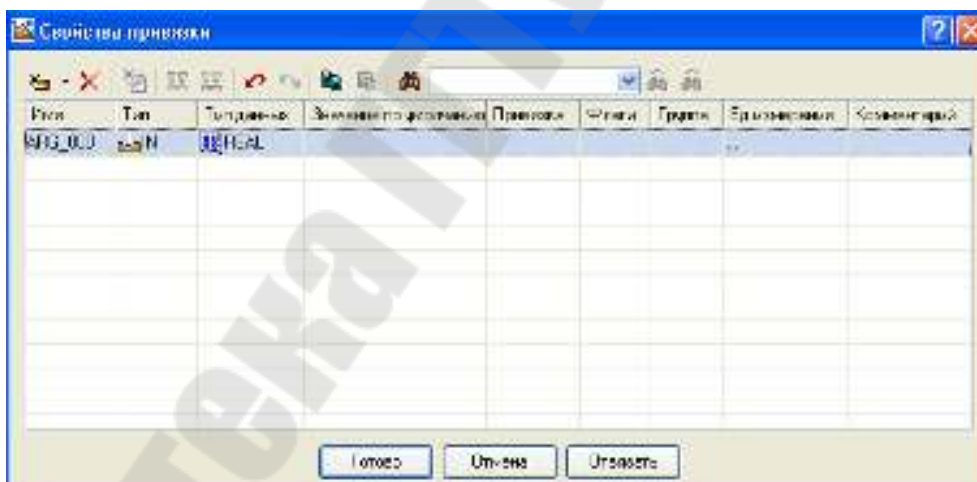


Рисунок 2.13 – Вид окна

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

Созданный графический экран будет иметь следующий вид:

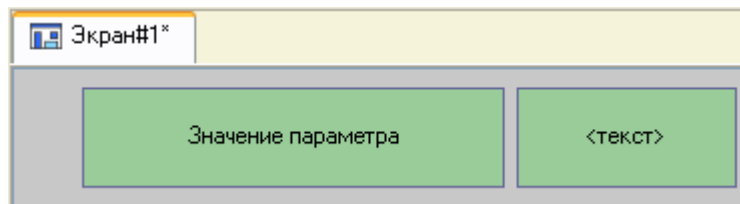







Рисунок 2.14 – Вид экрана

2.2.3 Создание стрелочного прибора, с привязкой показаний к аргументу

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

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

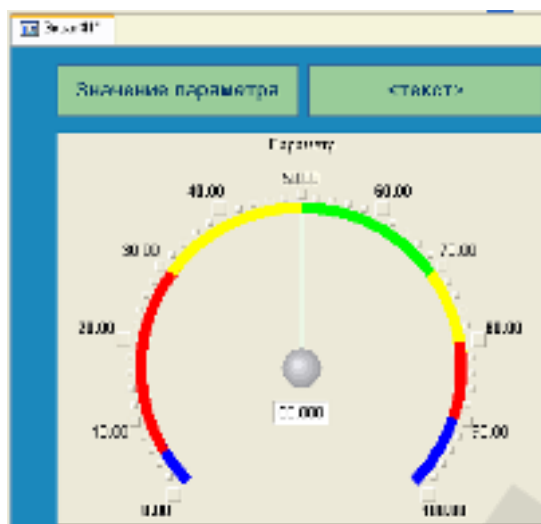




Рисунок 2.15 – Графический элемент «Стрелочный прибор»

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

Имя	Тип	Значение
Параметр	FLOAT	0

Рисунок 2.16 – Окно задания значения аргумента

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

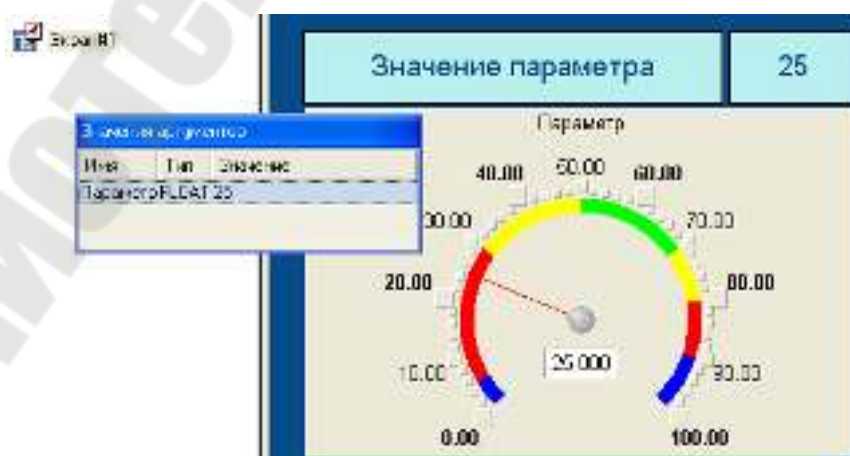



Рисунок 2.17 – Режим эмуляции

2.3. Автопостроение канала

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

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

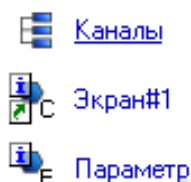


Рисунок 2.18 – Вид окна компонентов проекта

2.3.1 Задание границ и уставок

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

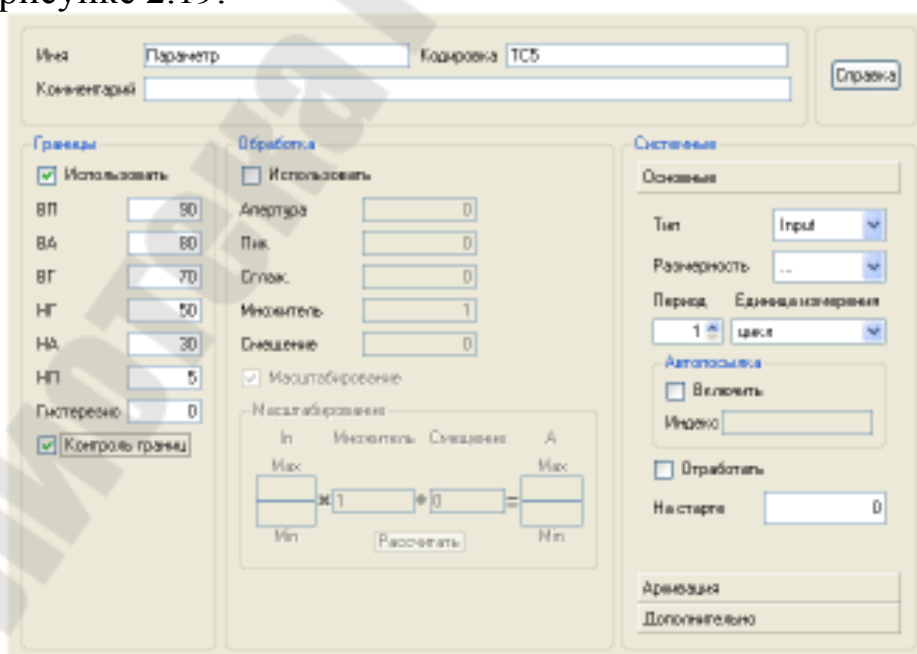


Рисунок 2.19 – Окно редактирования атрибутов канала

2.4. Создание генератора синуса и привязка его к каналу

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

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

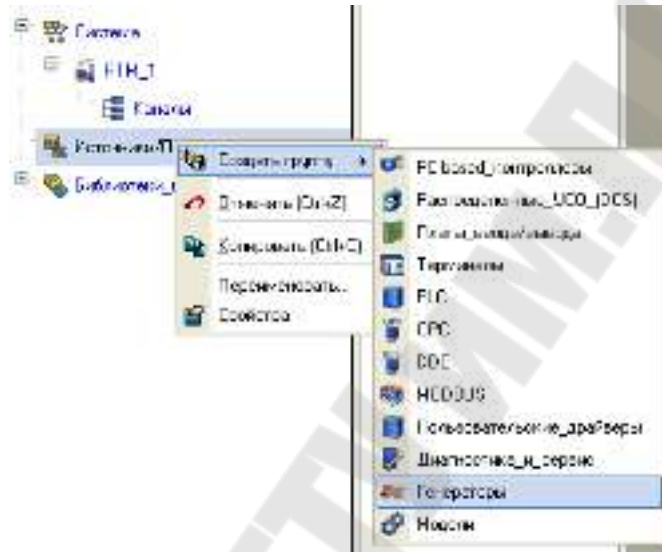


Рисунок 2.20 – Создание в группу компонентов «Генераторы»

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

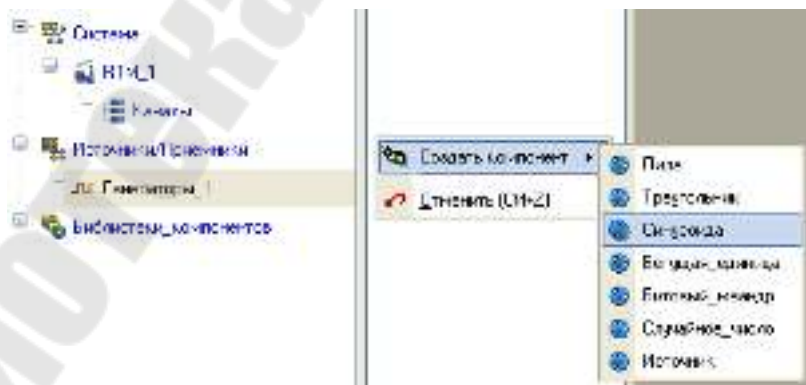


Рисунок 2.21 – Создание компонента «Синусоида»

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

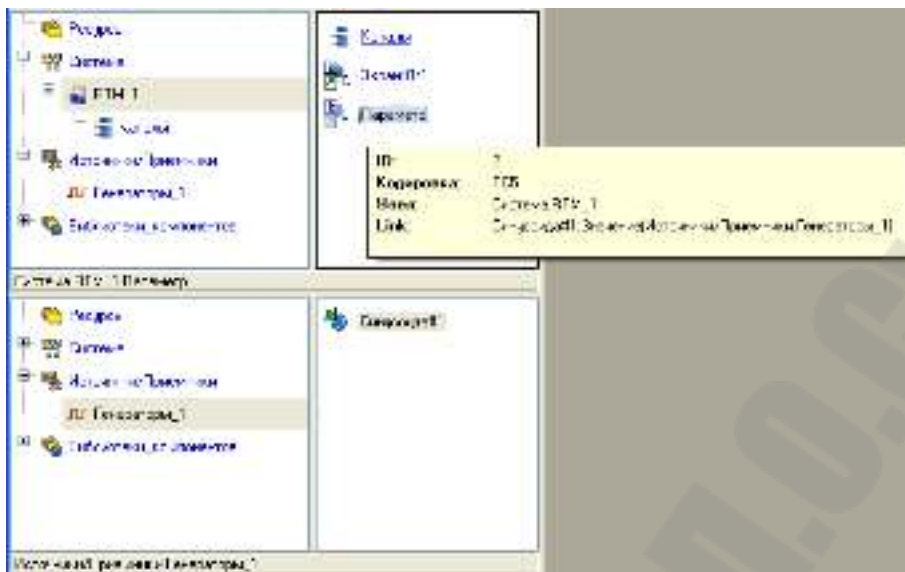





Рисунок 2.22 – Вид окна проекта после привязки генератора к каналу


2.5. Запуск проекта

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

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



Рисунок 2.23 – Вид окна профайлера

Для остановки работы профайлера кнопка .

2.6. Добавление функции управления

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




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



Рисунок 2.24 – Вид графического экрана после добавления элемента кнопка

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

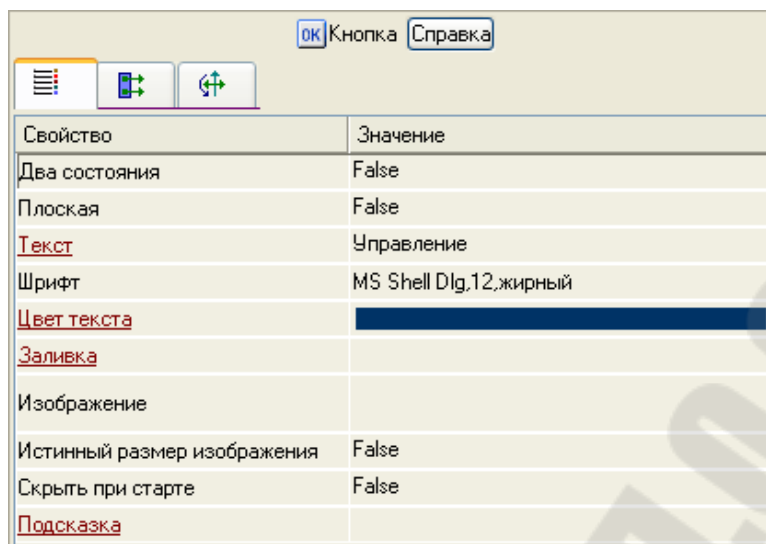


Рисунок 2.25 – Окно свойств элемента кнопка

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

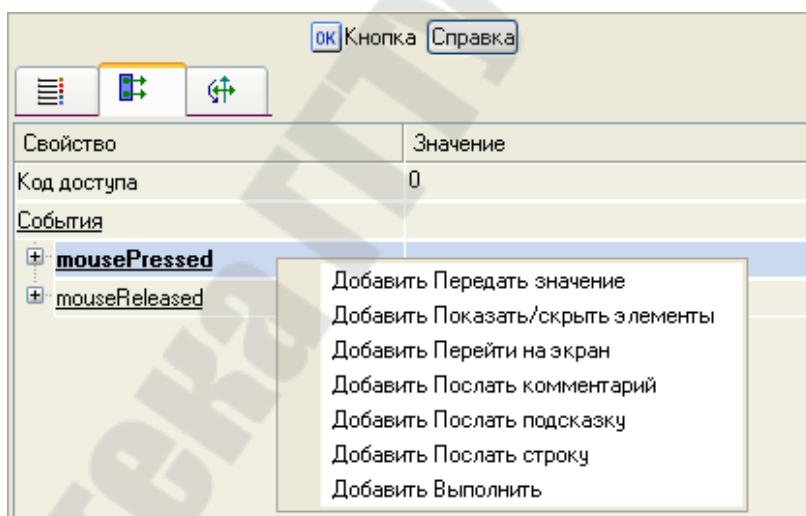


Рисунок 2.26 – Окно настройки событий элемента кнопка

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

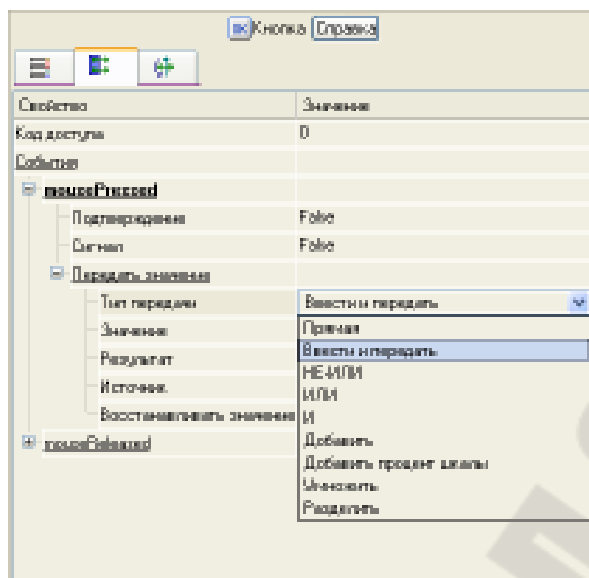


Рисунок 2.27 – Окно настройки событий элемента кнопка

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

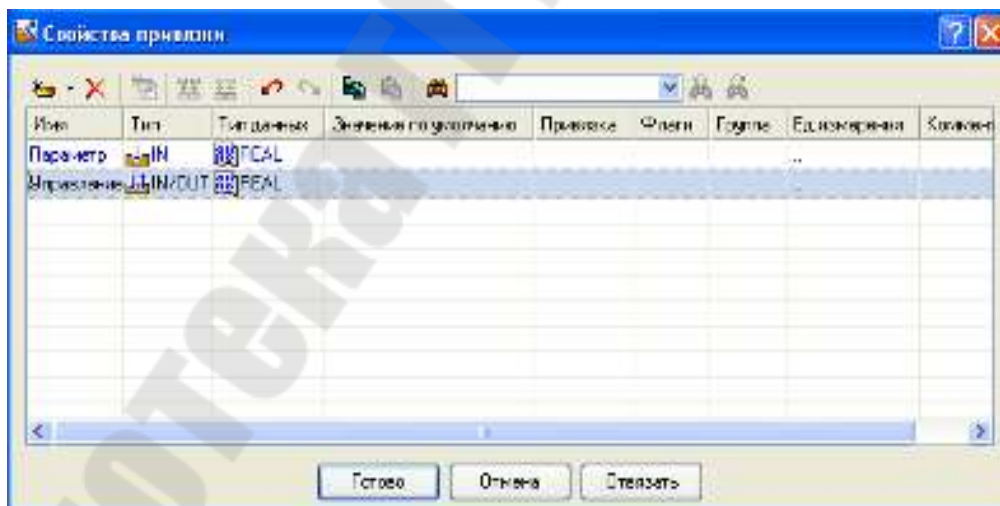


Рисунок 2.28 – Окно создания аргумента



- закроем окно свойств ГЭ с помощью щелчка ЛК по иконке

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

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



Рисунок 2.29 – Выделенный графический элемент

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

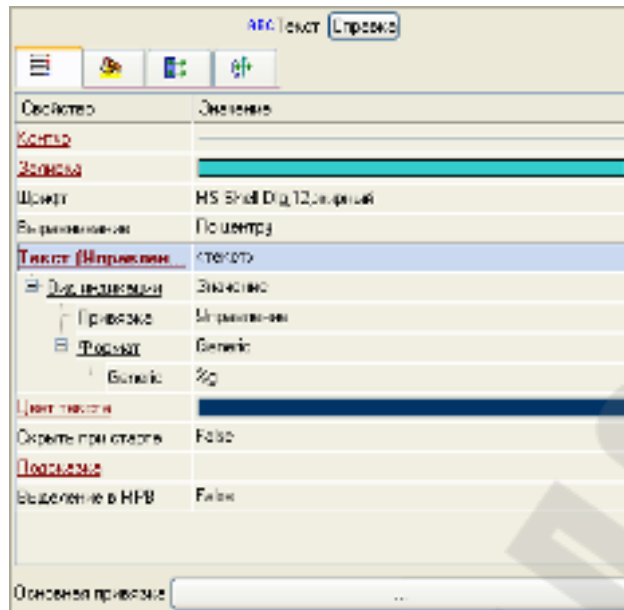


Рисунок 2.30 – Окно настройки привязки атрибута ГЭ «Текст»


2.7. Привязка аргумента экрана к каналу

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

- в слое «**Система**» откроем узел **RTM_1**;
- по щелчку ПК вызовем через контекстное меню свойства компонента **Экран#1**;



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

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

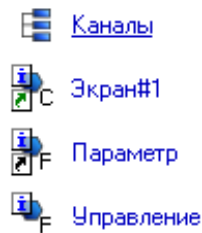


Рисунок 2.32 – Вид окна компонентов проекта

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

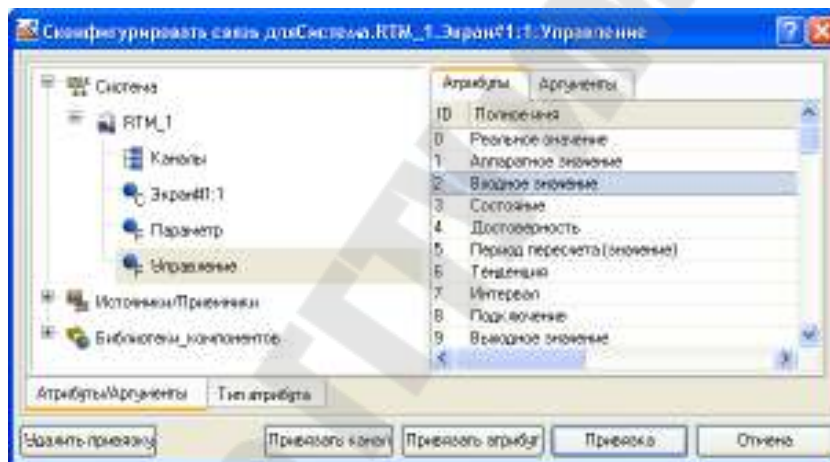





Рисунок 2.33 – Вид окна настройки аргумента с атрибутом канала

2.8. Размещение ГЭ Тренд

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

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

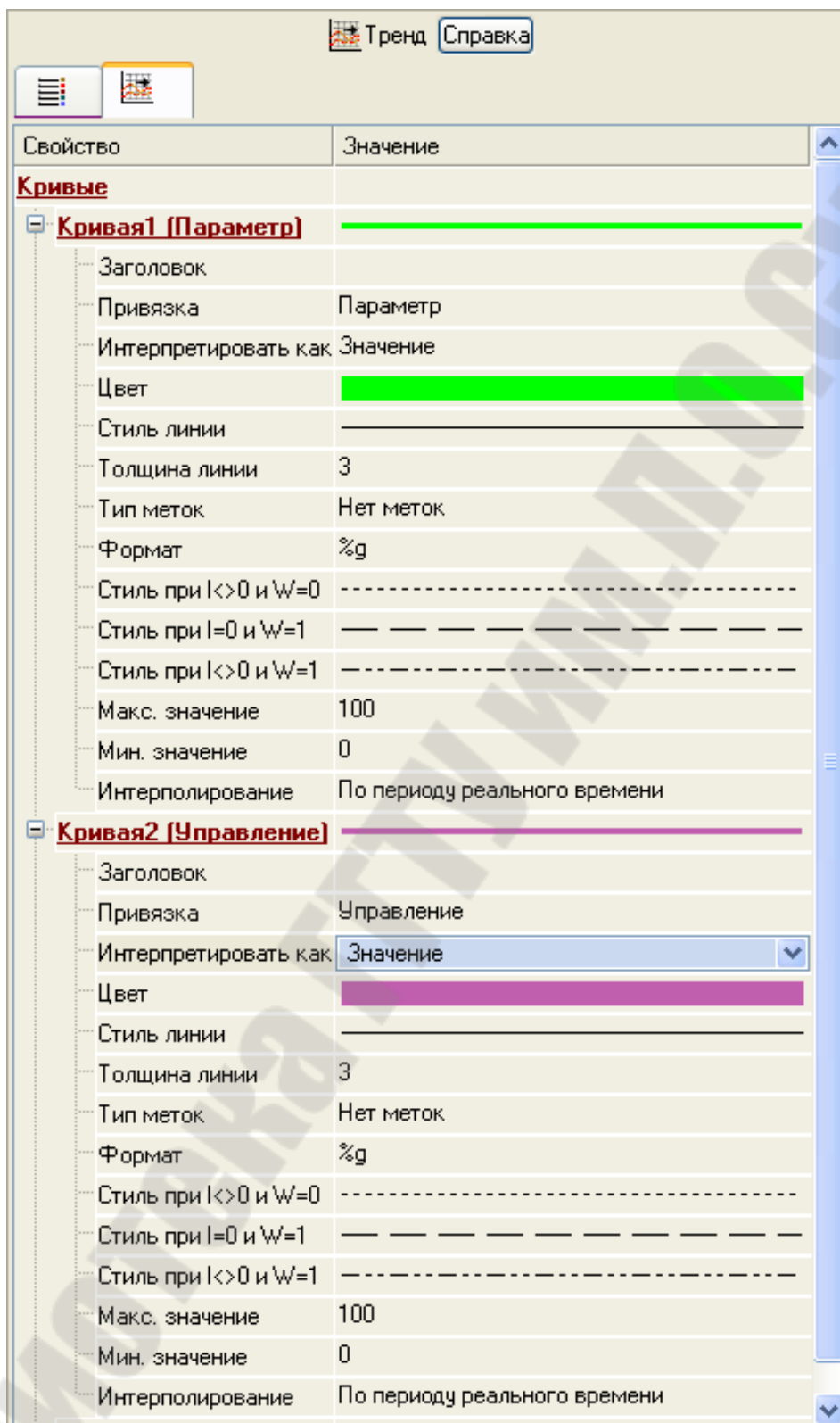


Рисунок 2.34 – Окно настройки ГЭ – тренд

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

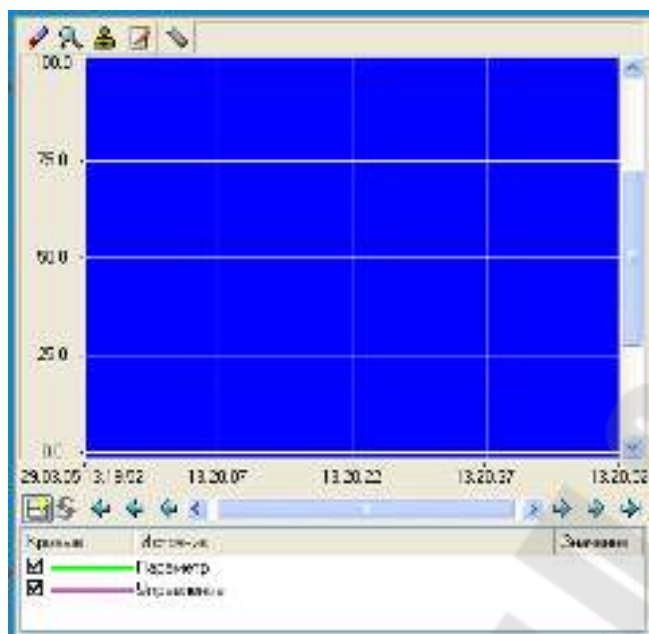





Рисунок 2.35 – Вид ГЭ – тренд

2.9 Запуск проекта

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

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

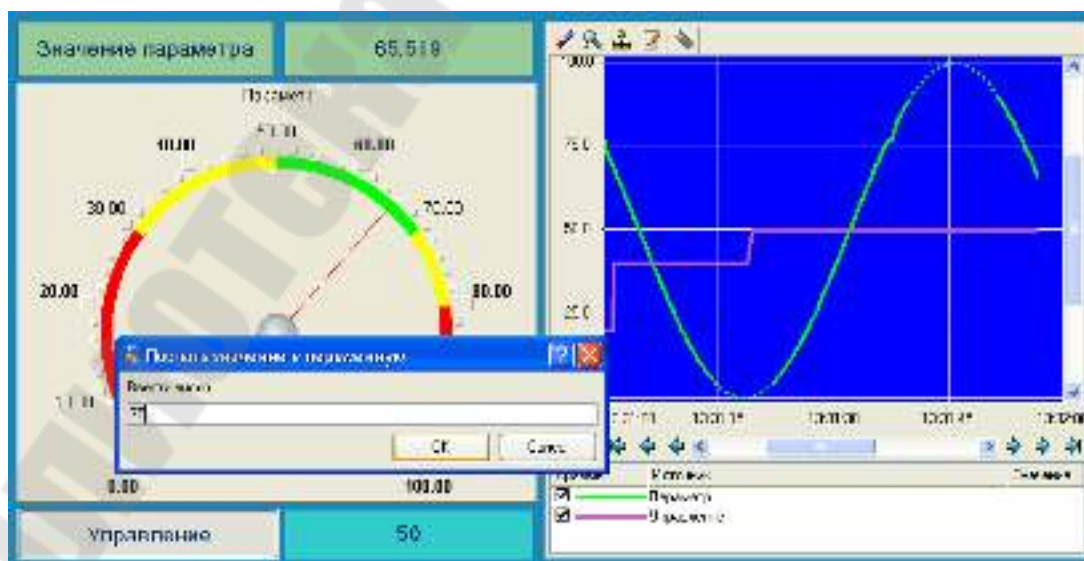


Рисунок 2.36 – Вид окна работы профайлера

2.10 Простейшая обработка данных

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

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



Рисунок 2.37 – Вид окна с новыми элементами

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

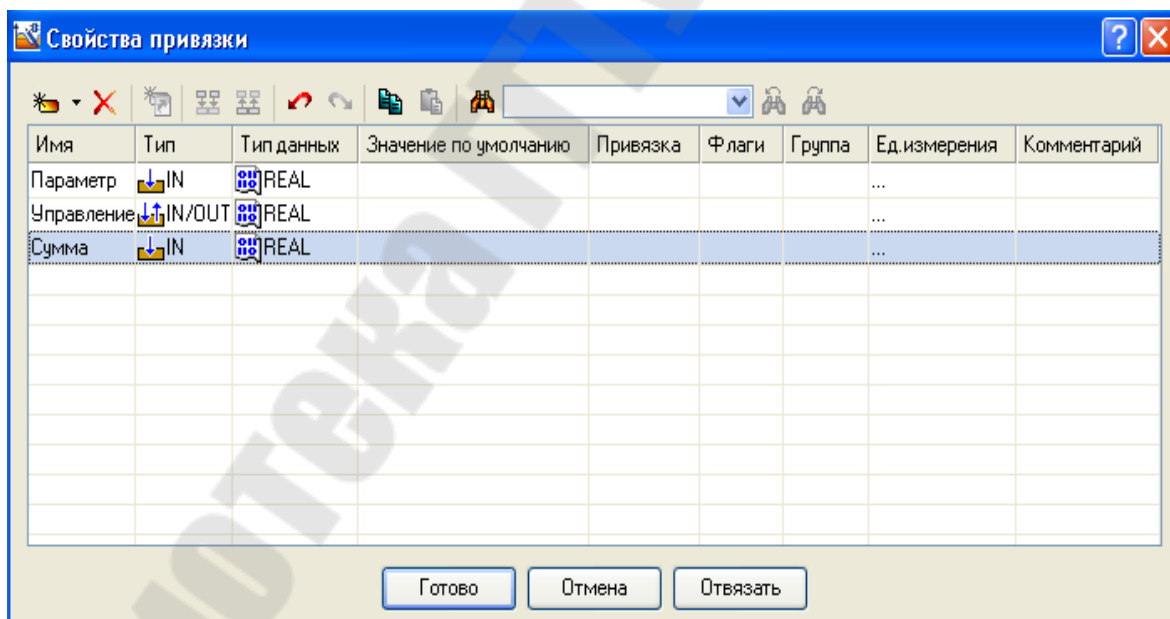


Рисунок 2.38 – Окно создания аргумента

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



Рисунок 2.39 – Окно настройки ГЭ – тренд

2.11 Создание программы на языке Техно ST

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

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

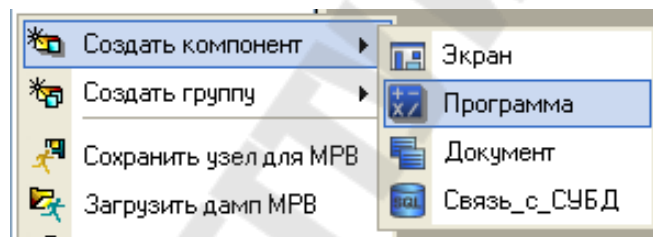


Рисунок 2.40 – Создание компонента – «Программа»

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

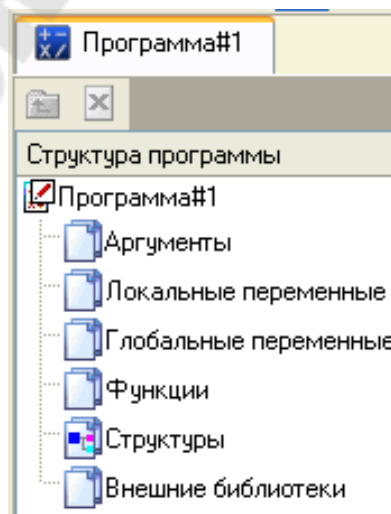

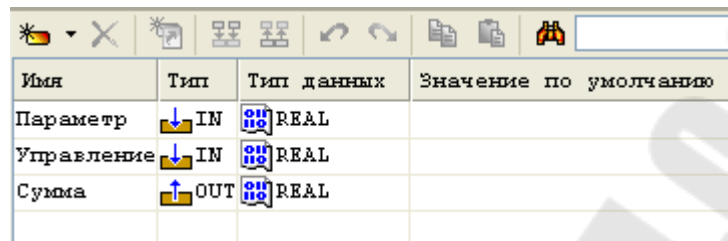


Рисунок 2.41 – Окно компонента – «Программа»

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



Имя	Тип	Тип данных	Значение по умолчанию
Параметр	IN	REAL	
Управление	IN	REAL	
Сумма	OUT	REAL	

Рисунок 2.42 – Окно аргументов компонента – «Программа»

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

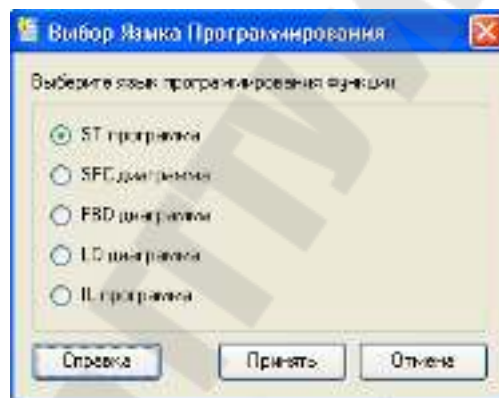


Рисунок 2.43 – Окно выбора языка программы

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

```



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

Сумма=Параметр+Управление;

END_PROGRAM

```

Рисунок 2.44 – Окно редактора программы

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

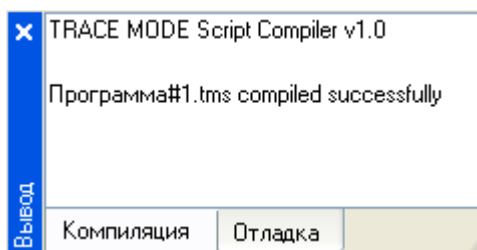


Рисунок 2.45 – Окно результатов компиляции программы

2.12 Привязка аргументов программы

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

- вызовем свойства компонента «**Программа#1**» через контекстное меню;
- выберем вкладку «**Аргументы**». Двойным нажатием в поле «**Привязка**» свяжем **аргументы** программы с **атрибутами** каналов – аргумент «**Параметр**» к **реальному значению** канала «**Параметр**», аргумент «**Управление**» к реальному значению канала «**Управление**»;

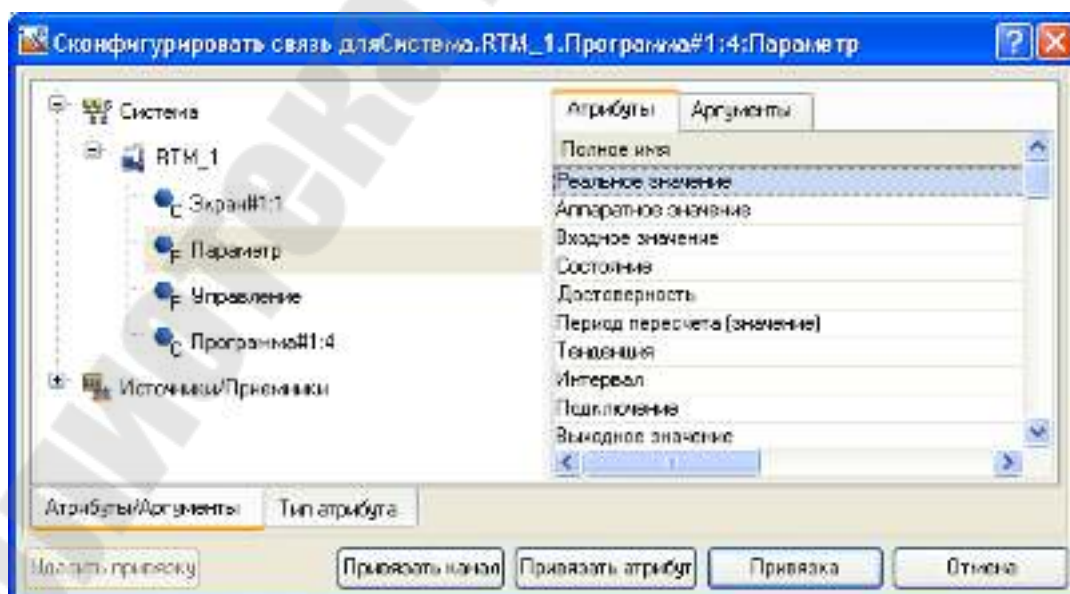


Рисунок 2.46 – Окно настройки привязки аргументы программы с атрибутами каналов

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

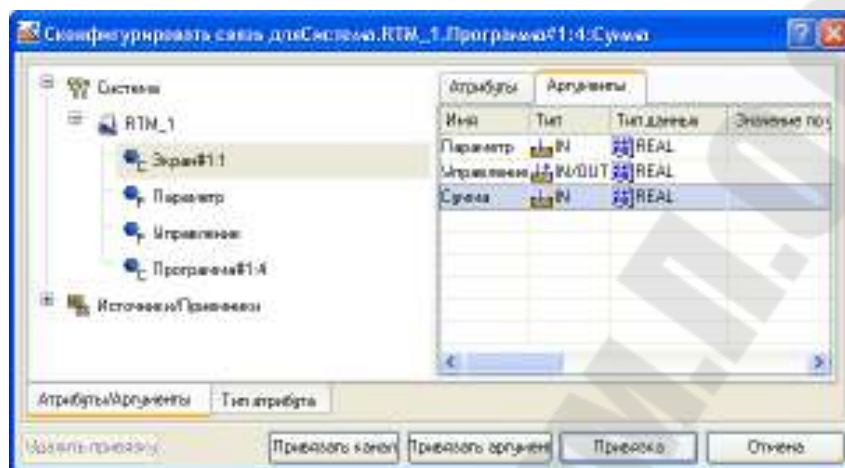





Рисунок 2.47 – Окно конфигурирования связи аргумента программы «Сумма»

В результате, будем иметь следующие связи – рисунок 2.48.



Рисунок 2.48 – Окно привязок компонента «Программа#1»

2.13 Запуск проекта

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

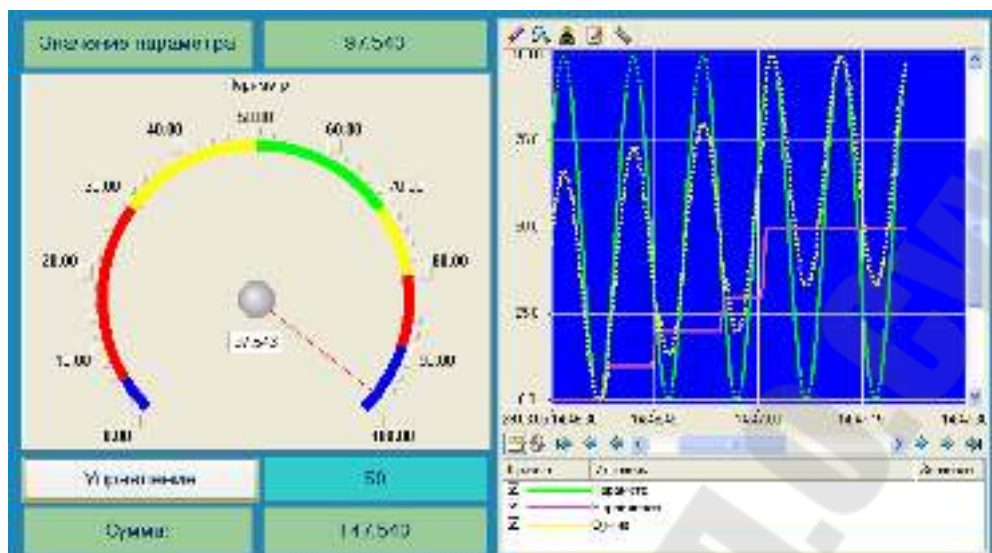


Рисунок 2.49 – Вид окна работы профайлера

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

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

- запуск приложения MS Excel;
- запишем в двух ячейках первого столбца запросы на получение данных:

=RTM0|GET!Параметр

=RTM0|GET!Управление

где **0** – индивидуальный номер узла в проекте;

- запустим на исполнение узел АРМ RTM_1;
- в меню таблицы MS Excel «**Правка**» выберем команду «**Связи**», выделим оба параметра и нажмем кнопку «**Обновить**», после чего закроем окно кнопкой ОК.

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

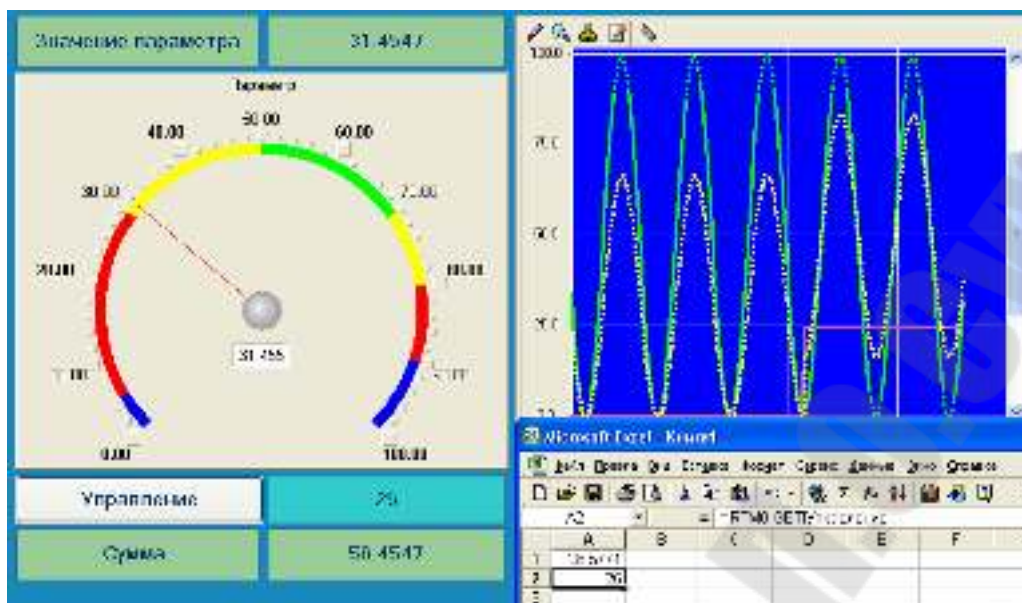


Рисунок 2.50 – Вид окна работы профайлера и MS Excel

3. Содержимое отчета

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

4. Контрольные вопросы

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

Лабораторная работа №2 Мониторы TRACE MODE. Числовые каналы.

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

1. Теоретические сведения

1.1 Работа монитора TRACE MODE. Канал данных

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

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

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

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

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

Совокупность каналов узла называется **базой каналов** этого узла. **Класс канала** определяет его общее назначение. При разработке проекта могут быть созданы каналы только определенных 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 predetermined алгоритмы (часть из них может настраиваться пользователем), в соответствии с которыми некоторые атрибуты канала устанавливаются или вычисляются монитором в зависимости от состояния или значения других атрибутов. Исполнение внутренних алгоритмов канала и анализ его атрибутов монитором называется **пересчетом канала**.

В этих алгоритмах, в общем случае, задействованы 4 атрибута – «**Входное значение**» (2, In), «**Аппаратное значение**» (1, A), «**Реальное значение**» (0, R) и «**Выходное значение**» (9, Q).

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

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

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

Направление преобразования зависит от типа канала:

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

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

налов пересчет конкретного канала также выполняется при определенных условиях.

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

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

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

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

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

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

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

1.2 Канал FLOAT. Обработка данных

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

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

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

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

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

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

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

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

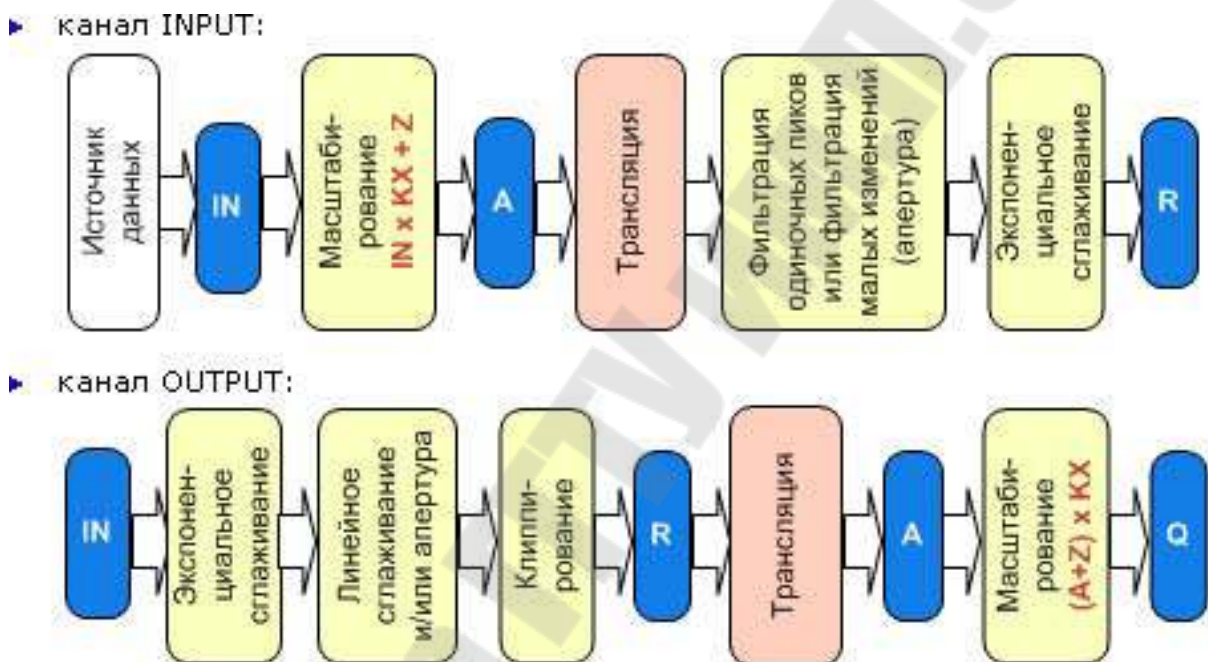


Рисунок 1.1 – Структура канала типа **FLOAT**

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

1.2.1 Масштабирование

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

- в канале типа **INPUT**: $A=IN \cdot KX+Z$;
- в канале типа **OUTPUT**: $Q=(A+Z) \cdot KX$;

1.2.2 Экспоненциальное сглаживание

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

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

При $PRS=1$ и $DSmoot \neq 0$ ($0 < DSmoot \leq 1$) в канале FLOAT выполняется экспоненциальное сглаживание (n - индекс указывает номер такта пересчета):

– в канале типа INPUT:

$$R_n = Result_n(1 - DSmoot_n) + DSmoot_n * R_{n-1}$$

$Result_n$ обозначает результат фильтрации пиков;

– в канале типа OUTPUT:

$$RESULT_n = IN_n(1 - DSmoot_n) + DSmoot_n * R_{n-1}$$

Результат экспоненциального сглаживания $RESULT_n$ далее обрабатывается процедурой линейного сглаживания. При $DSmoot \neq 0$ фильтрация малых изменений в канале FLOAT не выполняется.

1.2.3 Фильтрация пиков или фильтрация малых изменений в канале INPUT

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

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

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

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

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

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

- если $|\text{RES}_n - \text{R}_{n-1}| \leq \text{DPic}_n$, то выполняется фильтрация малых изменений:
 - если $|\text{RES}_n - \text{R}_{n-1}| < \text{APert}_n$, то $\text{Result}_n = \text{R}_{n-1}$;
 - если $|\text{RES}_n - \text{R}_{n-1}| \geq \text{APert}_n$, то $\text{Result}_n = \text{RES}_n$;
- если $|\text{RES}_n - \text{R}_{n-1}| > \text{DPic}_n$, то выполняется фильтрация одиночных пиков:
 - если на такте ($n-1$) фильтрация пиков не использовалась, $\text{Result}_n = \text{R}_{n-1}$ (первый пик фильтруется);
 - если на такте ($n-1$) фильтрация пиков использовалась, $\text{Result}_n = \text{RES}_n$ (второй подряд пик не фильтруется).

где

RES_n - результат процедуры трансляции или в ее отсутствие, аппаратное значение канала;

Result_n - результат описываемых алгоритмов (на n -ом такте пересчета).

1.2.4 Линейное сглаживание и фильтрация малых изменений в канале OUTPUT

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

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

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

- если $|\mathbf{RESULT}_n - \mathbf{R}_{n-1}| \leq \mathbf{DPic}_n$, то выполняется фильтрация малых изменений:
 - если $|\mathbf{RESULT}_n - \mathbf{R}_{n-1}| \leq \mathbf{APert}_n$, то $\mathbf{Res}_n = \mathbf{R}_{n-1}$;
 - если $|\mathbf{RESULT}_n - \mathbf{R}_{n-1}| > \mathbf{APert}_n$, то $\mathbf{Res}_n = \mathbf{RESULT}_n$;
- если $|\mathbf{RESULT}_n - \mathbf{R}_{n-1}| > \mathbf{DPic}_n$, то вычисляется результат линейного сглаживания $\mathbf{LIN}_n = \mathbf{R}_{n-1} + \mathbf{DPic}_n$ (если $\mathbf{RESULT}_n > \mathbf{R}_{n-1}$) или $\mathbf{LIN}_n = \mathbf{R}_{n-1} - \mathbf{DPic}_n$ (если $\mathbf{RESULT}_n < \mathbf{R}_{n-1}$) и далее выполняется фильтрация малых изменений:
 - если $|\mathbf{LIN}_n - \mathbf{R}_{n-1}| \leq \mathbf{APert}_n$, то: $\mathbf{Res}_n = \mathbf{R}_{n-1}$;
 - если $|\mathbf{LIN}_n - \mathbf{R}_{n-1}| > \mathbf{APert}_n$, то: $\mathbf{Res}_n = \mathbf{LIN}_n$;

где

\mathbf{RESULT}_n - результат предыдущей процедуры;

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

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

1.2.5 Клиппирование в канале OUTPUT

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

- если $\mathbf{Res} > \mathbf{HL}$, то $\mathbf{R} = \mathbf{HL}$;
- если $\mathbf{Res} < \mathbf{LL}$, то $\mathbf{R} = \mathbf{LL}$;
- если $\mathbf{LL} \leq \mathbf{Res} \leq \mathbf{HL}$, то $\mathbf{R} = \mathbf{Res}$.

где \mathbf{Res} - результат предыдущей процедуры обработки (линейного сглаживания и/или фильтрации малых изменений).

1.3 Атрибуты канала FLOAT

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

1. раздел «Границы»:

- флаг **Использовать** - (85, **BNDR**) - установка этого флага в редакторе равнозначна присвоению атрибуту **BNDR** значения 0, что разрешает монитору анализировать значения шести границ канала. При **BNDR=1** (флаг снят) анализ границ запрещен. От этого флага зависит исполнение алгоритма клиппирования в канале **OUTPUT**;
- **ВП** - (26, **HL**) - значение верхнего предела;
- **ВА** - (28, **HA**) - значение верхней аварийной границы;
- **ВГ** - (30, **HW**) - значение верхней предупредительной границы;
- **НГ** - (31, **LW**) - значение нижней предупредительной границы;
- **НА** - (29, **LA**) - значение нижней аварийной границы;
- **НП** - (27, **LL**) - значение нижнего предела;
- **Гистерезис** - (32, **Hyst**) - от этого параметра зависят условия генерации сообщений при переходе реальным значением канала заданных границ:

в сторону увеличения номера интервала:

<LL, <LA, <LW, >HW, >HA, >HL

в сторону уменьшения номера интервала:

>(LL+H), >(LA+H), >(LW+H), <(HW-H), <(HA-H), <(HL-H).

- флаг **Контроль границ** - (53, **SC_F**, **MPB - LMT**) – установка этого флага равнозначна присвоению атрибуту **LMT** значения 1. Действие флага различно для каналов типов **INPUT** и **OUTPUT**. В первом случае наличие флага означает разрешение установки каналу признака программной недостоверности (**FS=1**) в случае выхода **реального значения** канала за пределы диапазона [**LL**, **HL**] (при **BNDR=0**). При возврате реального значения в диапазон признак программной недостоверности автоматически сбрасывается (**FS=0**). Для типа **OUTPUT** установка флага «**Контроль границ**» разрешает клиппирование **реального значения** канала (при **BNDR=0**). При **LMT=0** или **BNDR=1** описанные алгоритмы не исполняются;

2. раздел «Обработка»:

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

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

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

In	Множитель	Смещение	A
Max 200	× 4.54	+ -154	Max 754
Min 100			Min 300

Рисунок 1.2 – Настройка масштабирования канала типа INPUT

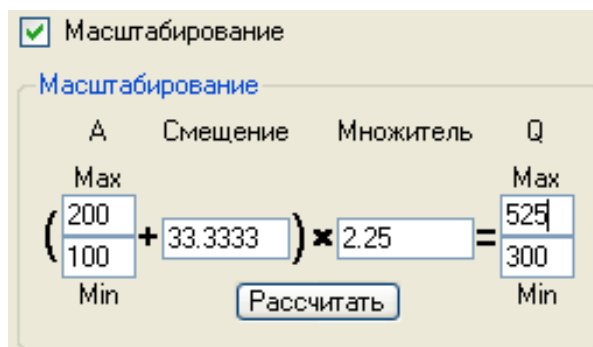


Рисунок 1.3 – Настройка масштабирования канала типа OUTPUT

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

1.4 Границы и интервалы канала FLOAT

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

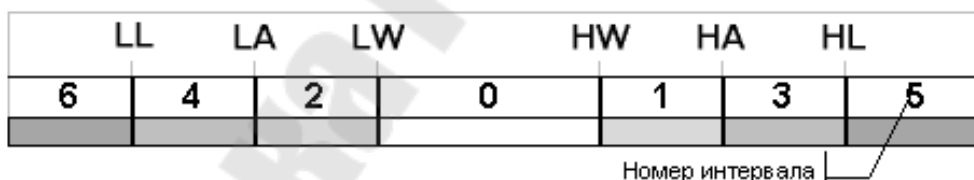


Рисунок 1.4 – Интервалы канала класса **FLOAT**

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

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

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

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

1.5 Генерация сообщений. Гистерезис

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

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

1.6 Канал класса DOUBLE FLOAT

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

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

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



Рисунок 1.5 – Интервалы канала класса DOUBLE FLOAT

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

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

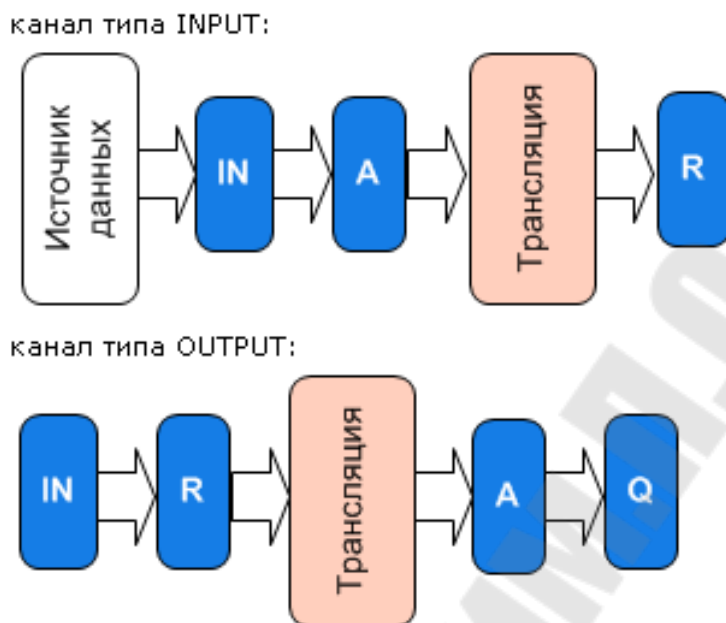


Рисунок 1.6 – Структура канала типа DOUBLE FLOAT

1.7 Канал класса HEX16 и HEX32

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

- **Размерность в битах** – (56, **nBits**) **Number Bits** – данный атрибут задает число байт, участвующих в процедуре инверсии:
 <= 8 – 1 байт (<=24-3 байта для HEX32);
 > 8 – 2 байта (>24-4 байта для HEX32);
 - флаг **Инверсия** (40, **NM**) – если этот флаг установлен, инвертирование в канале разрешено;
 - флаг **DEC** (84, **HD**) – если этот флаг установлен (**HD=1**), значение канала отображается в профайлере в десятичном виде; если флаг не установлен (**HD=0**) – в шестнадцатеричном. От этого флага зависит также алгоритм записи сообщений в отчет тревог.
- Атрибуты **Входное значение** (2, **In**), **Аппаратное значение** (1, **A**), **Реальное значение** (0, **R**) и **Выходное значение** (9, **Q**) канала

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

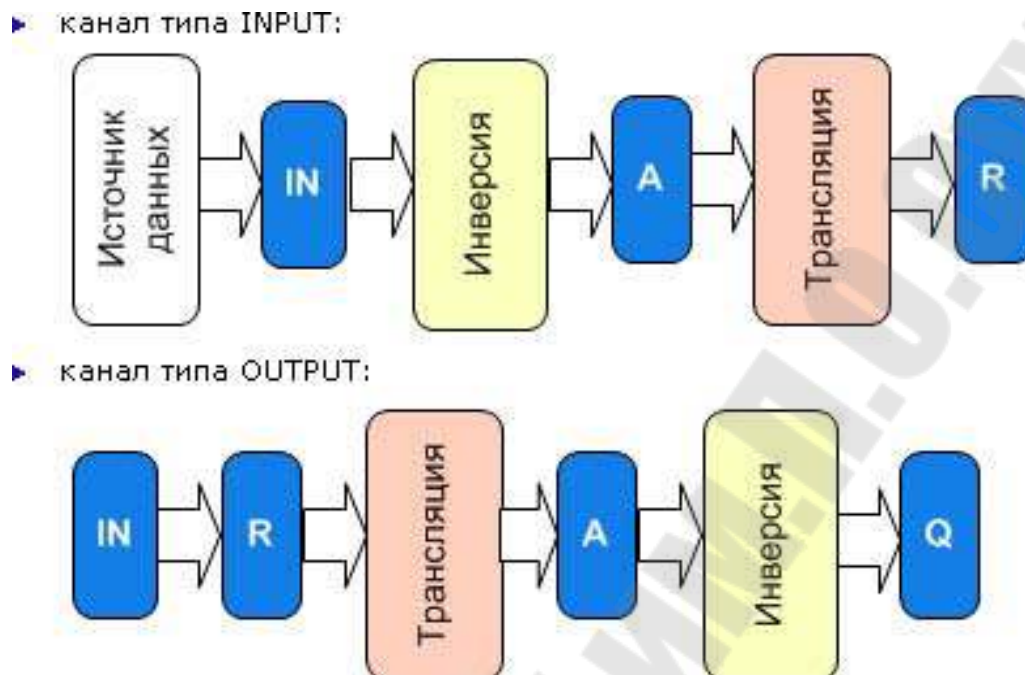


Рисунок 2.7 – Структура канала типа HEX

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

1.8 Пересчет базы каналов монитором

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

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

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

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

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

1.9 Время цикла монитора

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

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

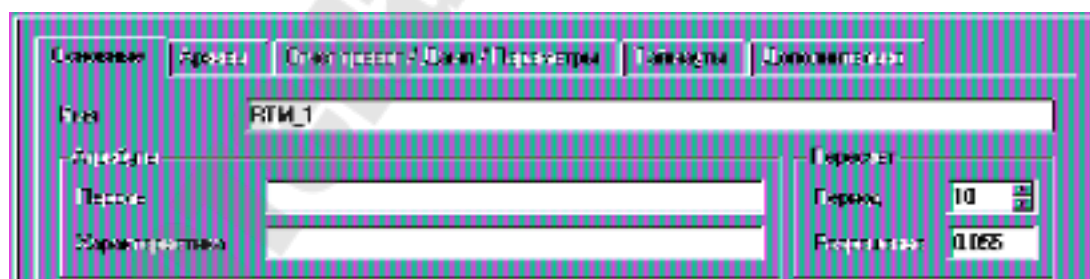


Рисунок 1.8 – Окно настройки времени цикла монитора

1.10 Период пересчета канала

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

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

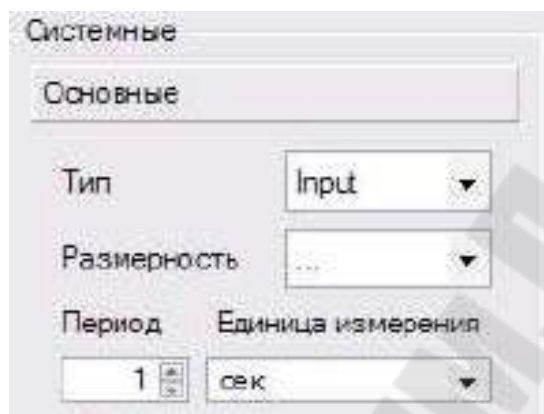


Рисунок 1.9 – Окно настройки пересчета канала

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

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

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

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

Пример




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

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

2. Ход работы

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

2.1. Создание проекта Trace Mode

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

2.2 Создание узла

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

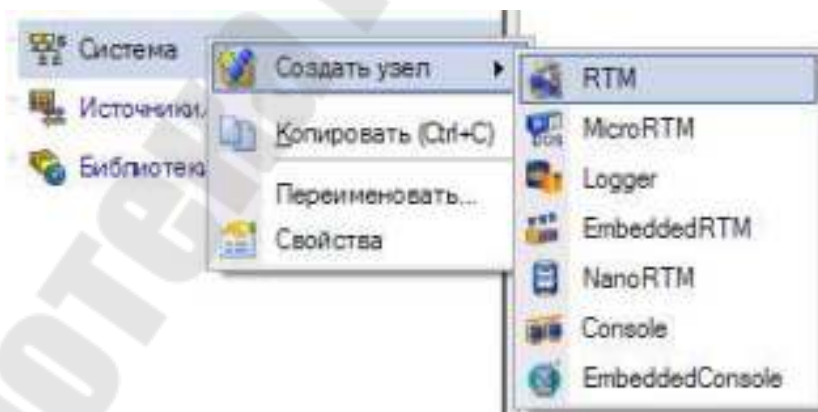


Рисунок 2.1 – Окно навигатора проекта

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

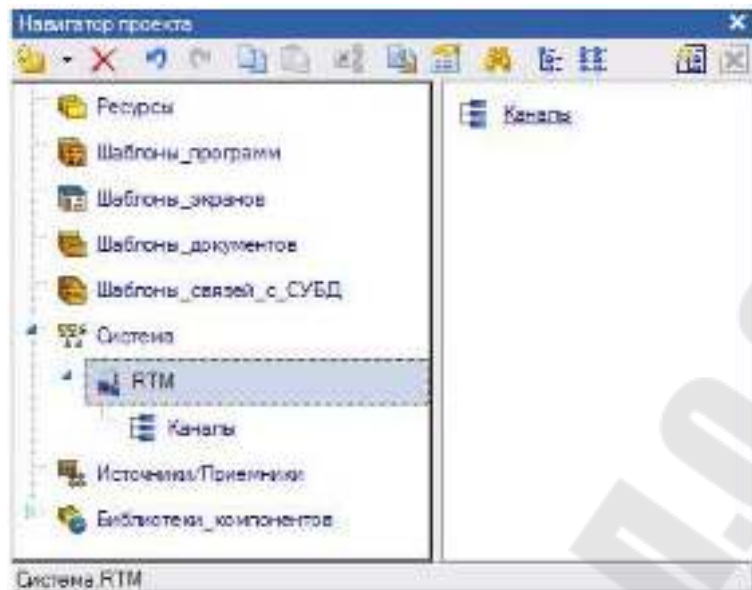


Рисунок 2.2 – Окно навигатора проекта с созданным узлом

2.3 Создание канала

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

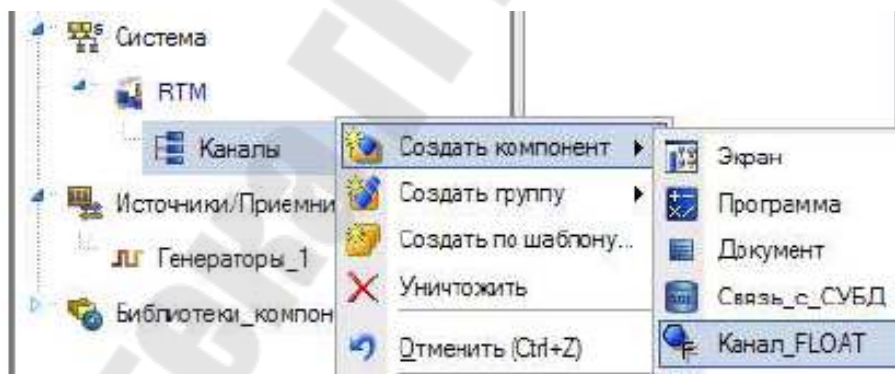


Рисунок 2.3 – Создание канала

2.4 Создание источника сигнала

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

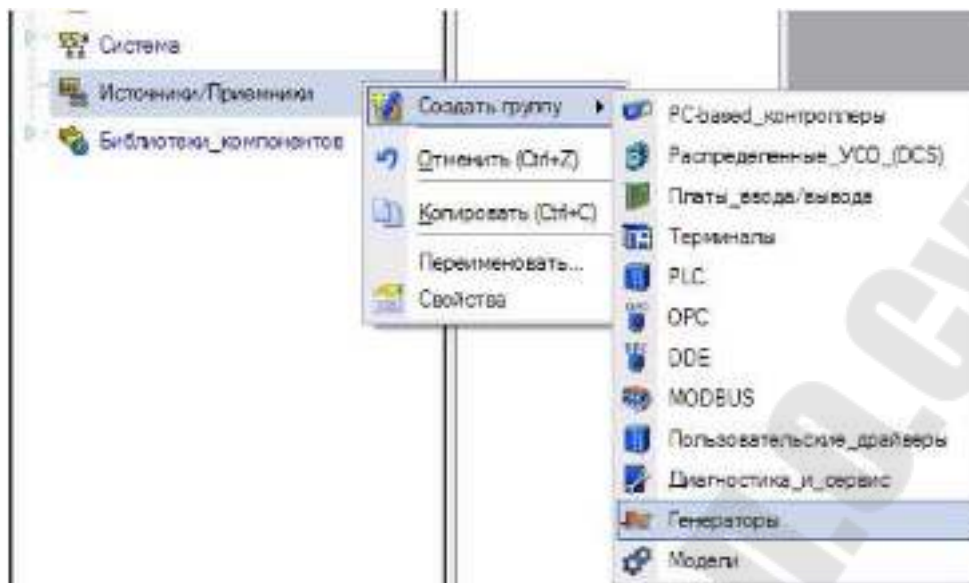


Рисунок 2.4 – Создание группы – Генераторы

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

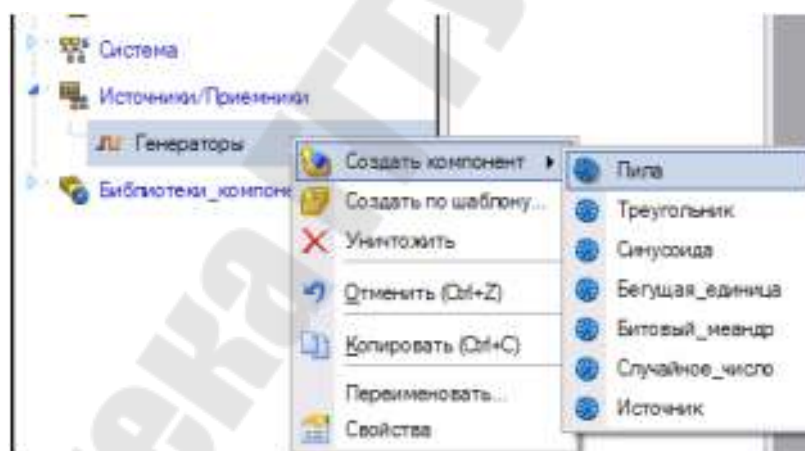


Рисунок 2.5 – Создание группы генератора пилы

2.5 Настройка масштабирования канала

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

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

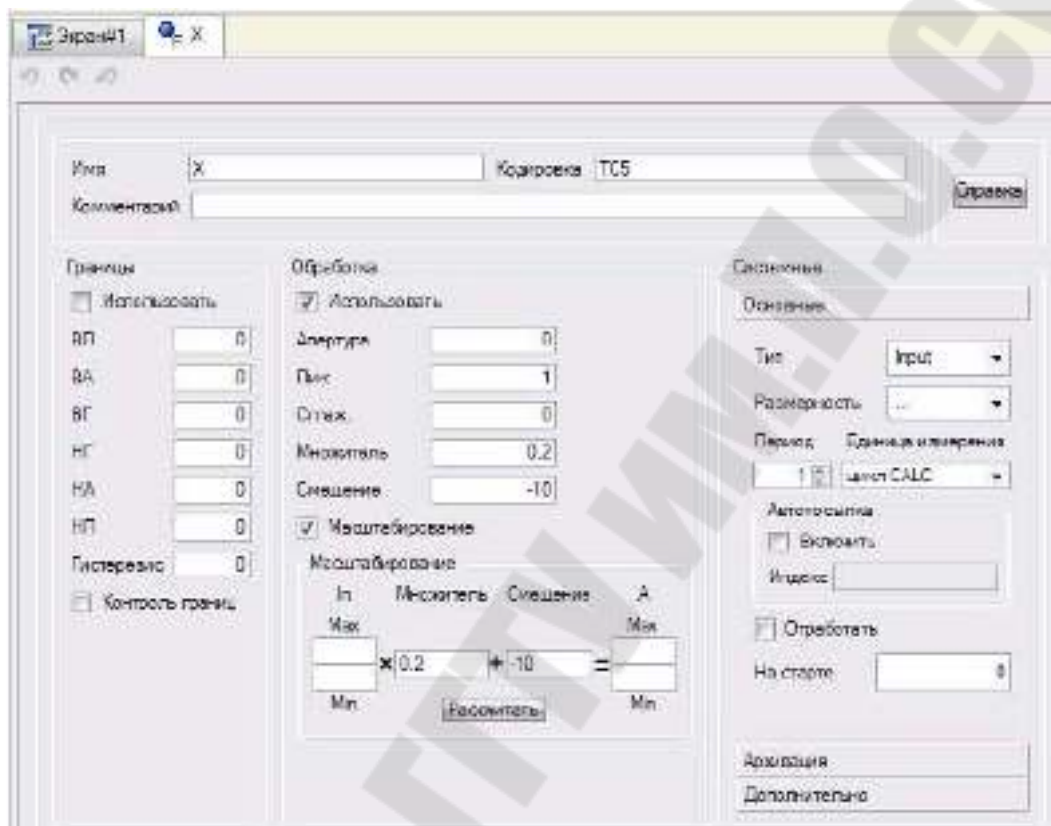



Рисунок 2.6 – Окно настроек атрибутов канала

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

2.6 Привязка генератора к каналу

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

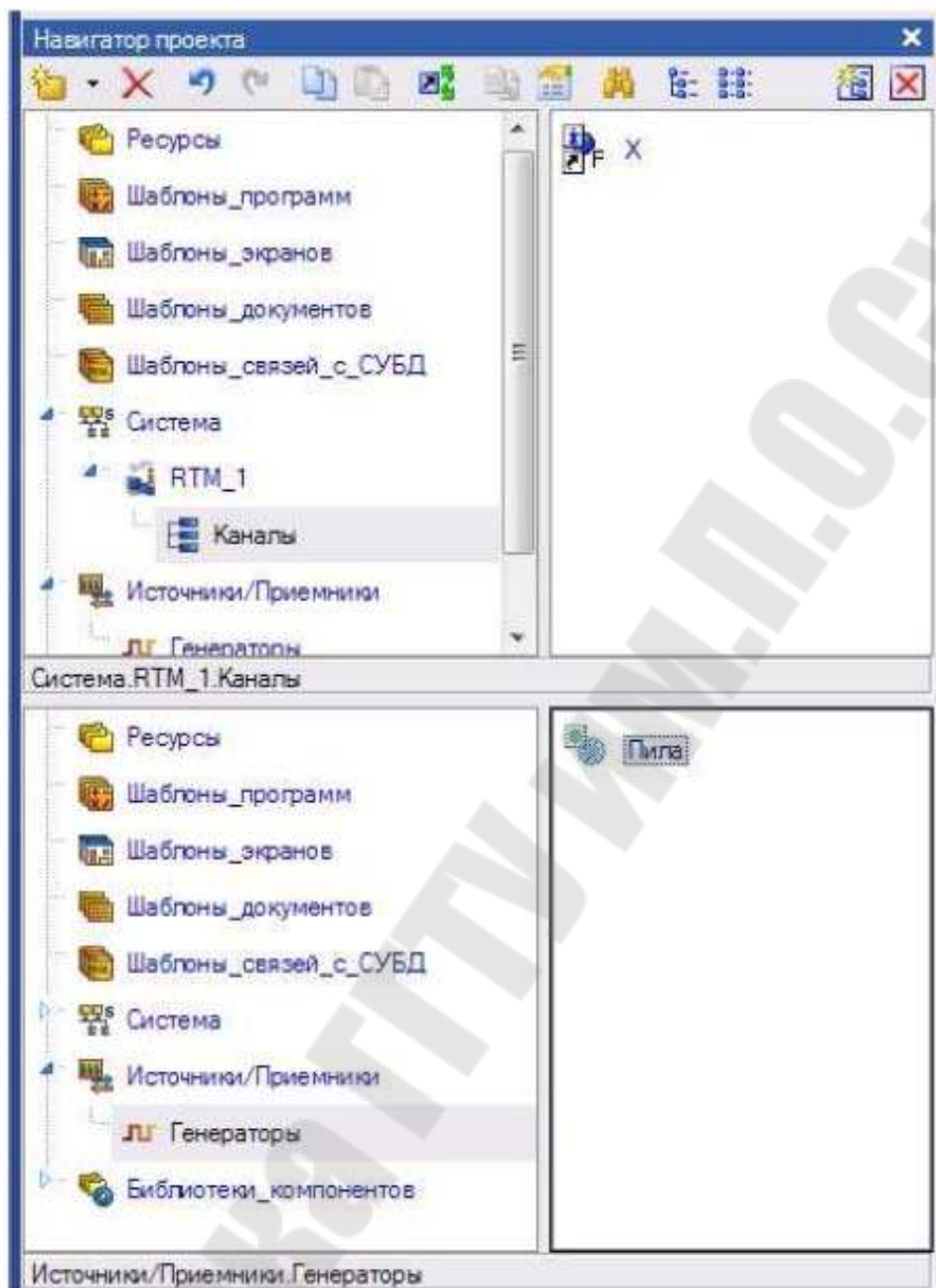



Рисунок 2.7 – Окно настройки привязки генератора к каналу

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

2.7 Создание шаблона экрана

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

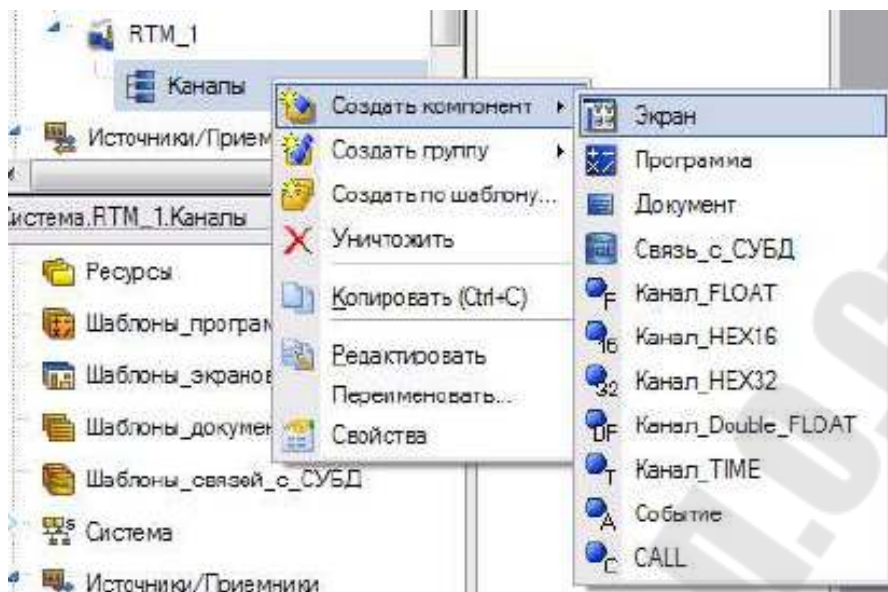




Рисунок 2.8 – Окно создания шаблона экрана

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

2.8 Создание и настройка стрелочного индикатора

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

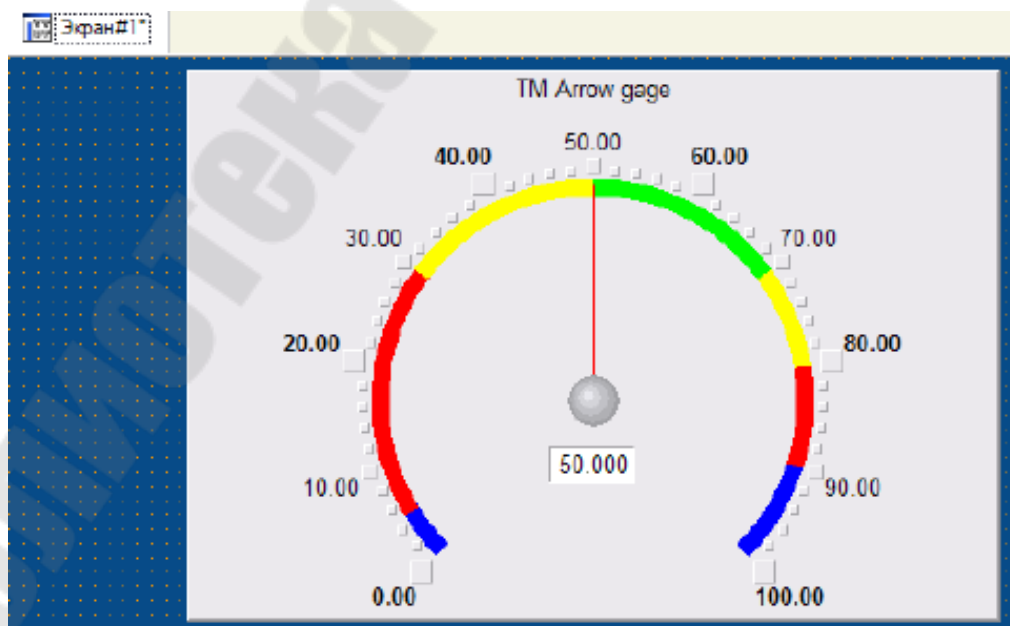


Рисунок 2.9 – Окно экрана с шаблоном индикатора

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

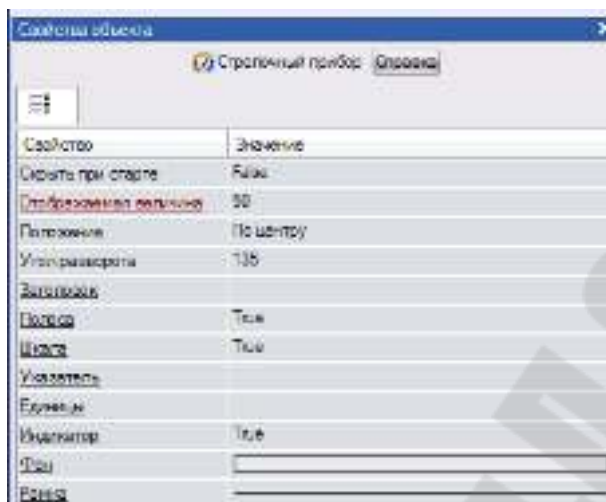


Рисунок 2.10 – Окно свойств индикатора

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

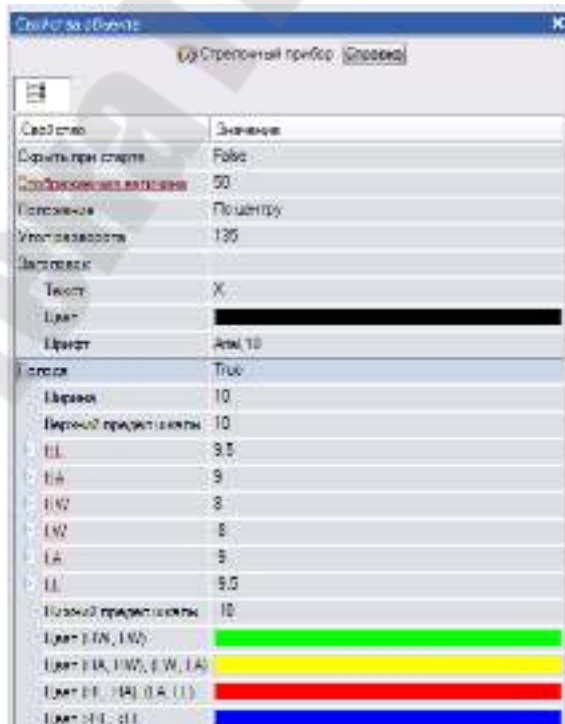


Рисунок 2.11 – Окно свойств индикатора

2.9 Привязка индикатора к каналу

Для привязки показаний индикатора к каналу необходимо:

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

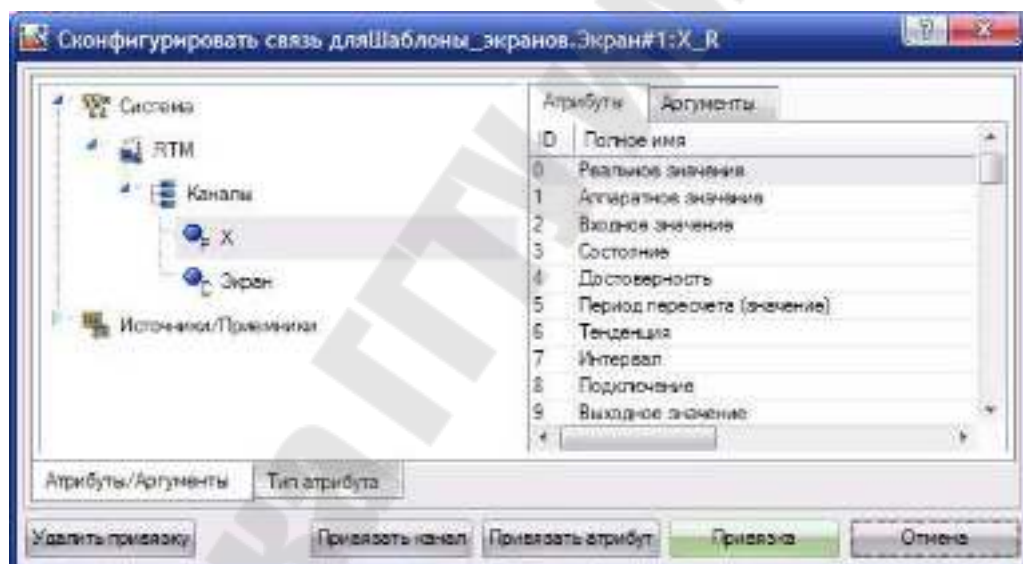


Рисунок 2.12 – Окно конфигурирования связи

Окончательно окно свойств привязки примет вид рисунок 2.13.

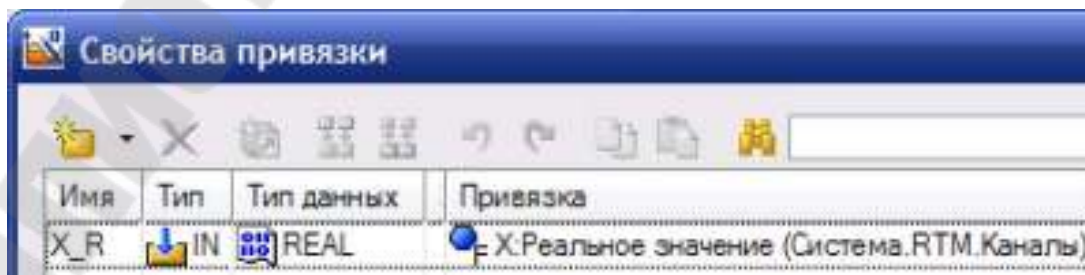







Рисунок 2.13 – Окно свойств привязки

2.10 Создание и настройка тренда

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

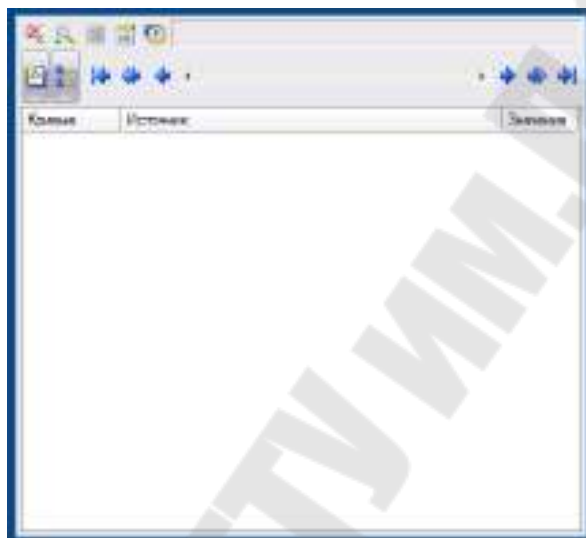




Рисунок 2.14 – Вид объекта – тренд

Откройте окно свойств тренда и настройте следующие параметры:

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

При необходимости скорректируйте размеры окна тренда (рисунок 2.15).

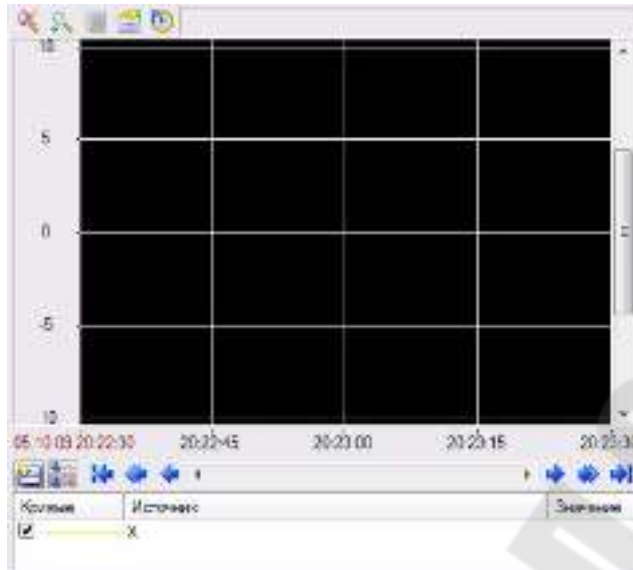





Рисунок 2.15 – Вид объекта – тренд

2.11 Запуск проекта.

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

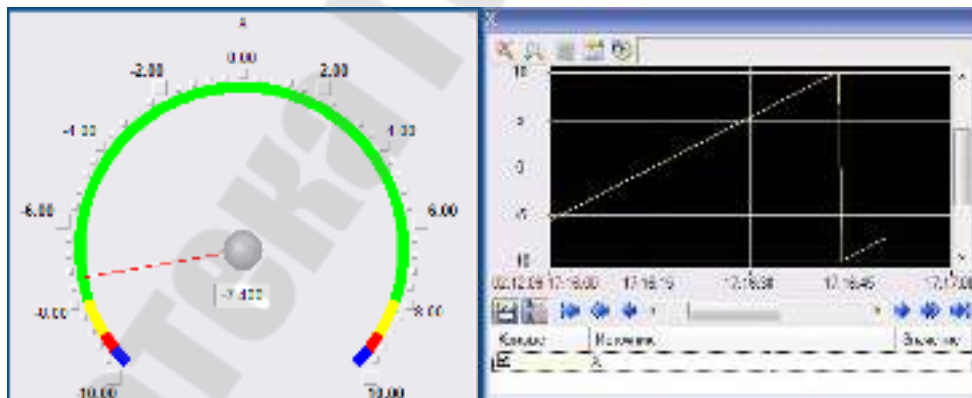


Рисунок 2.15 – Вид окна профайлера

Задание 2: Создать проект:

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

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

3. Содержимое отчета

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

4. Контрольные вопросы

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

Лабораторная работа №3 Разработка графического интерфейса оператора

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

1. Теоретические сведения

1.1 Создание и настройка объектов экрана

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











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

Таблица 1.1 – Объекты Trace Mode



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




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


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



Рисунок 1.1 – Вид окна создания аргумента

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

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


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

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

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


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

1.2 Статическое изображение

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

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

Таблица 1.2 – Ломанные и кривые

Иконка	Название объекта
	Ломаная линия






	Многоугольник
	Ломаная с заливкой
	Разомкнутая кривая
	Замкнутая кривая

Таблица 1.3 – Прямоугольники

Иконка	Название объекта
	Контур
	Прямоугольник
	Панель
	Рамка

Таблица 1.4 – Плоские фигуры

Иконка	Название объекта
	Плоский клапан
	Треугольник
	Стрелка
	Овал
	Эллипс сектор

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

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

Таблица 1.5 – Объемные фигуры

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

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



Рисунок 1.2 – Настройка толщины стенки фигуры

Для настройки вида материала следует дважды щелкнуть ЛК мыши по строчке **материал**, и настроить пункт **выбрать из списка**. Если установить его равным **False**, то материал задается как цвет аналогично другим объектам, если **True**, то можно выбрать материала, из которого выполнено изделие (хром, бронза, медь, текстуры шлифовка или гравировка).

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

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

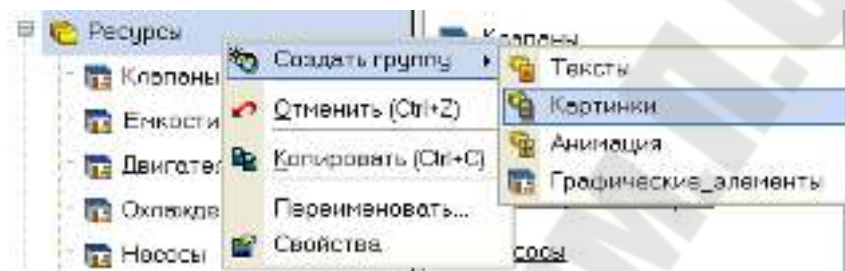


Рисунок 1.3 – Создание группы изображений



Рисунок 1.4 – Создание библиотеки изображений

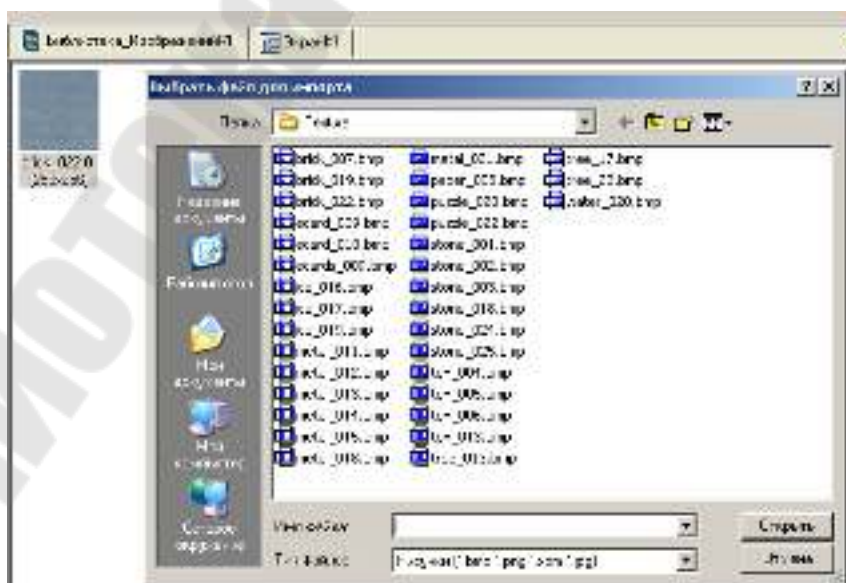


Рисунок 1.5 – Импорт изображений

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


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

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

1.3 Динамическое изображение

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

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

При динамическом контуре (закладка динамический контур ) задаются (рисунок 1.6) два цвета: цвет штрихов и промежутка между ними, длина штриха, которая также определяет шаг перемещения штрихов. При динамизации, происходит перемещение штрихов по контуру. Скорость перемещения определяется привязанным аргументом. Если аргумент равен 0, то перемещение отсутствует. Когда аргумент равен 1 происходит перемещение штрихов при каждом такте на один шаг. Если аргумент равен двум, к примеру, то штрихи перемещаются на один шаг один раз за 2 такта.


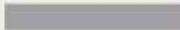

Свойство	Значение
Привязка	ARG_000
Цвет штриха	
Цвет промежутка	
Длина штриха	5
Промежуток/штрих	1

Рисунок 1.6 – Настройка динамического контура

Рассмотрим динамическую трансформацию (закладка динамическая трансформация ) . Можно выделить следующие виды – дина-

мическое перемещение, масштабирование, вращение. При динамическом перемещении задается ломаная линия, вдоль которой происходит перемещение. Для ряда точек (узлов) задаются соответствующие им значения (рисунок 1.7). Текущее положение объекта зависит от значения привязанного аргумента и значений, соответствующих узлам, флага – перемещать плавно.

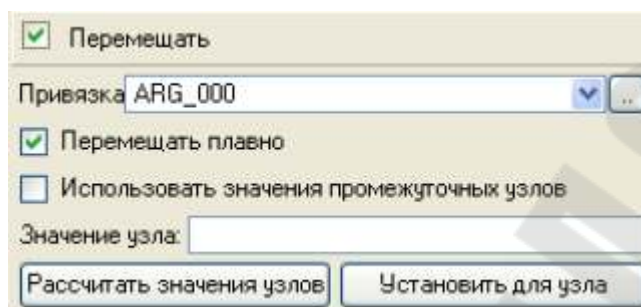


Рисунок 1.7 – Настройка динамического перемещения

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



Рисунок 1.8 – Настройка динамического масштабирования

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

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

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

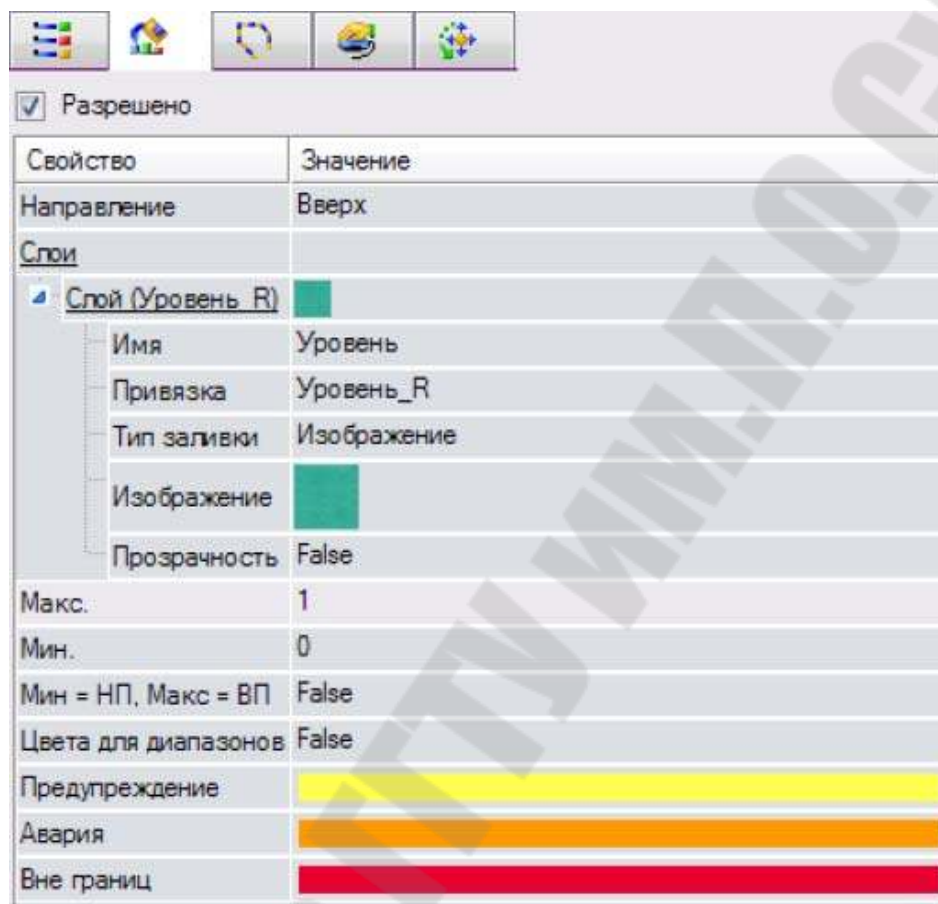


Рисунок 1.9 – Настройка динамической заливки

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





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


Рисунок 1.10 – Настройка динамического изменения цвета

2. Ход работы

Задание 1

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

2.1 Создание объектов

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

Создать следующие объекты проекта:

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

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

2.2 Создание статического изображения

2.2.1 Создание рамки







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



Рисунок 2.1 – Вид экрана информационной системы

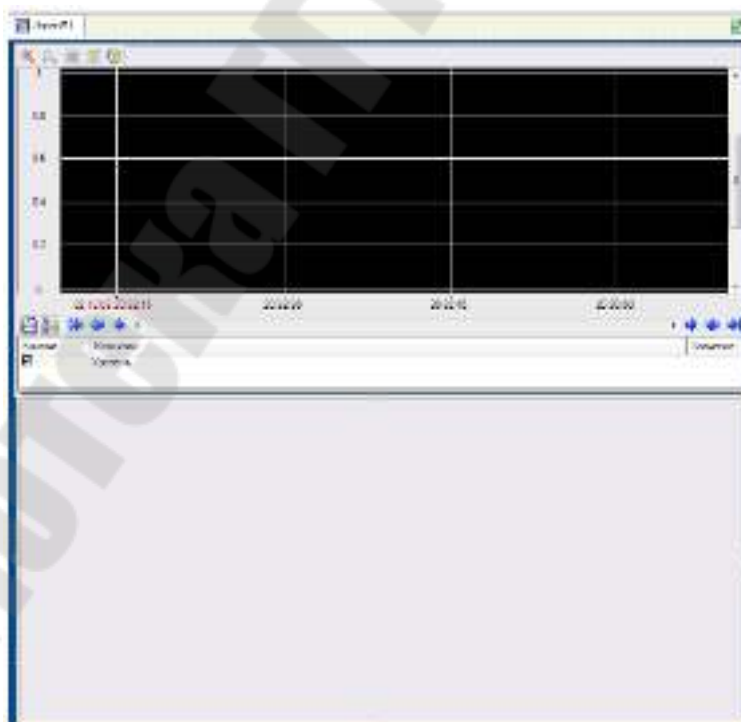
















Рисунок 2.2 – Вид экрана информационной системы

2.2.2 Создание емкости

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

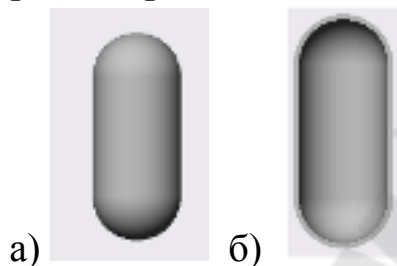






Рисунок 2.3 – Вид объекта «Емкость»
(а) – до редактирования свойств, (б) –

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

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



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

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



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

2.2.3 Создание насоса



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

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



Рисунок 2.5 – Вид объекта «Насос»

2.2.4 Создание труб

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

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

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



Рисунок 2.6 – Вид окна с размещенными статическими объектами

2.3 Создание динамического изображения

2.3.1 Импорт изображения

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

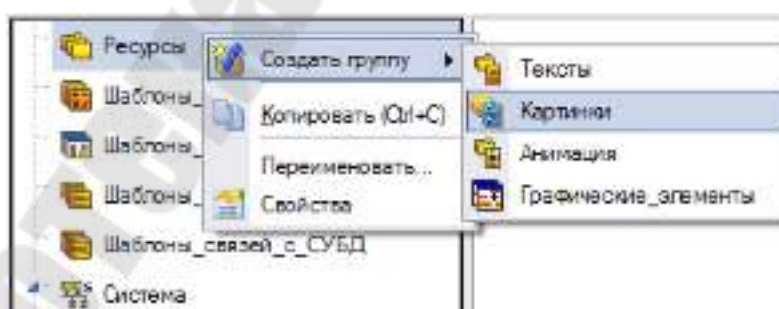


Рисунок 2.7 – Создание группы – «Картинки»

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

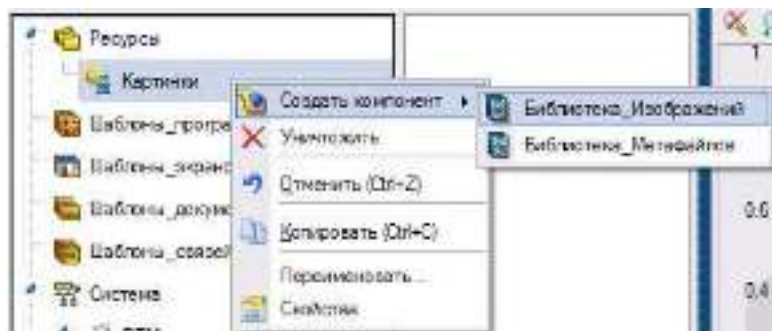










Рисунок 2.8 – Создание библиотеки изображений

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

2.3.2 Создание динамической заливки

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

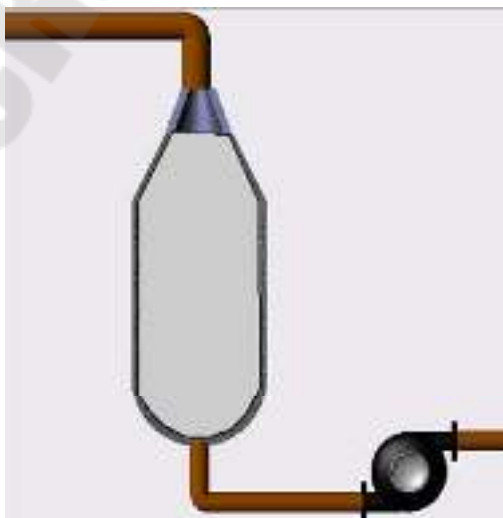



Рисунок 2.9 – Создание многоугольника для заливки

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

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

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

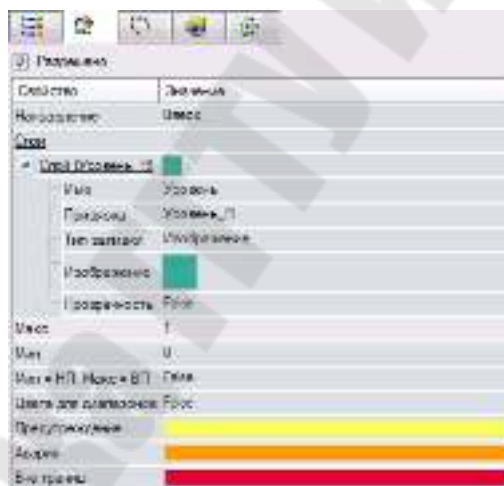





Рисунок 2.10 – Окно настройки параметров заливки



Рисунок 2.11 – Выбор изображения для заливки

2.4 Запуск проекта

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

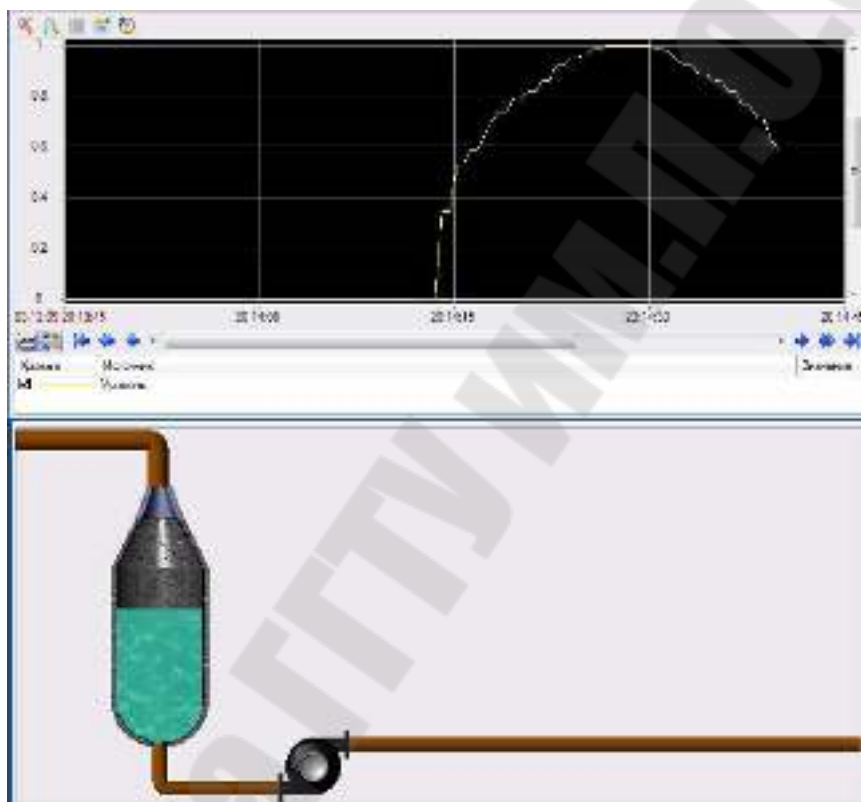



Рисунок 2.11 – Окно профайлера

3. Создание технологических объектов

3.1 Создание проекта

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

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



Рисунок 3.1 – Подключение библиотеки пользователя

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

3.2 Создание экрана оператора

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

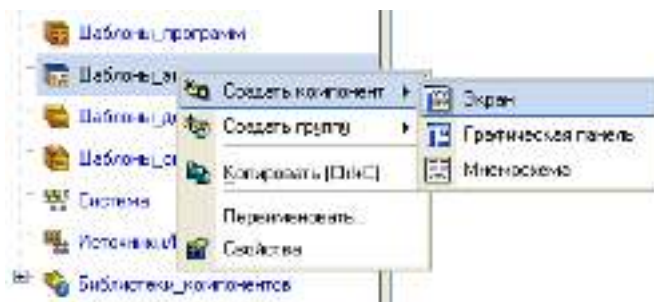


Рисунок 3.2 – Создание шаблона экрана




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



Рисунок 3.3 – Окно панели оператора

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

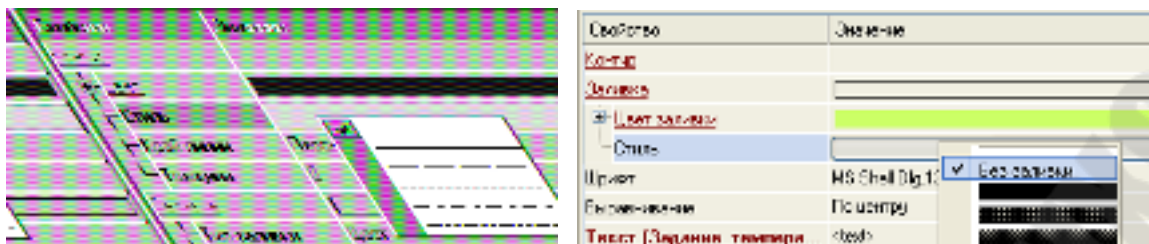



Рисунок 3.4 – Настройка свойств надписей

3.3 Создание аргументов экрана, каналов

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

Имя	Тип	Тип данных	Значение по умолчанию	Привязка
Уровень	IN	REAL	0	
Температура	IN	REAL	0	

Рисунок 3.5 – Создание аргументов шаблона экрана

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



Рисунок 3.6 – Окно панели оператора

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

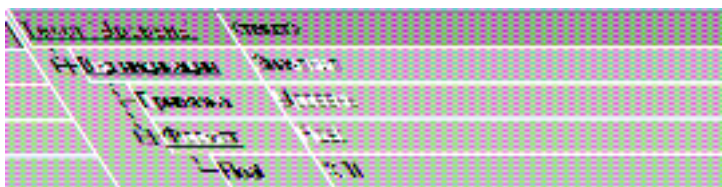



Рисунок 3.7 – Настройка свойств ГЭ 

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

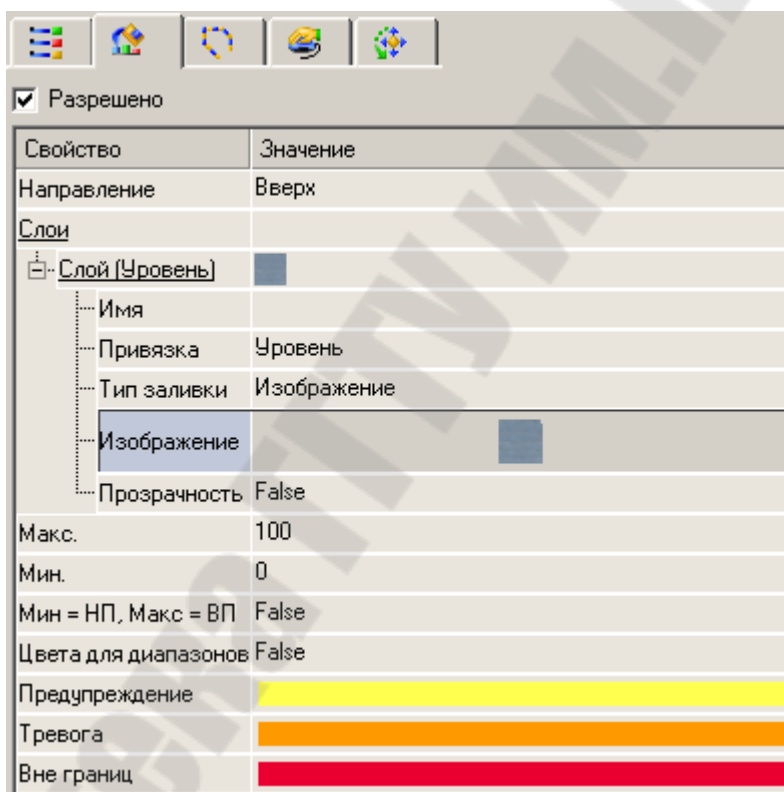



Рисунок 3.8 – Настройка свойств ГЭ 

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

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

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

Имя	Тип	Тип данных	Значение по умолчанию	Привязка
Уровень	IN	REAL	0	Уровень:Реальное значение (Система.RTM_1.Каналы)
Температура	IN	REAL	0	Температура:Реальное значение (Система.RTM_1.Каналы)

Рисунок 3.9 – Создание каналов по аргументам

3.4 Создание источников сигналов

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

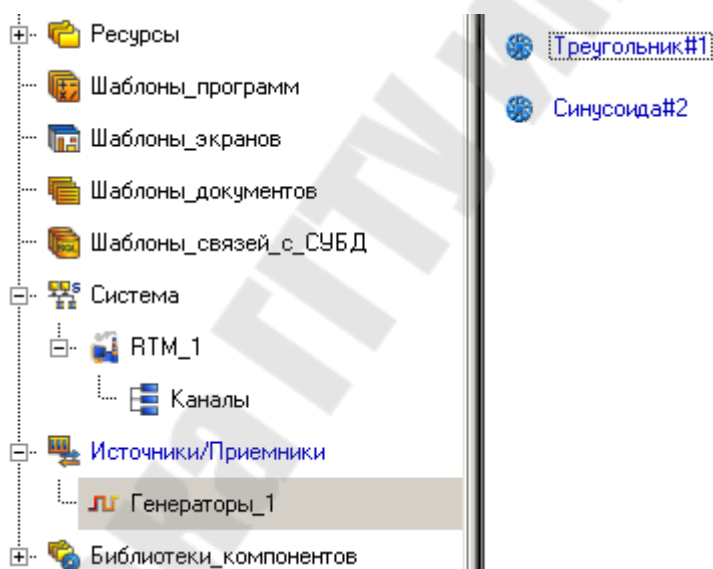




Рисунок 3.10 – Создание источников сигналов

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

3.5 Запуск проекта

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










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



Рисунок 3.11 – Создание источников сигналов

4. Контрольные задания

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

5. Содержимое отчета

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

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

6. Контрольные вопросы

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

Лабораторная работа №4
Распределенные системы мониторинга.
Работа с модулями удаленного ввода/вывода сигналов

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

1. Теоретические сведения

1.1 Идеология распределенных комплексов

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

1.1.1. Уровень контроллеров

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

1.1.2. Оперативный уровень

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

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

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

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

1.1.3. Административный уровень

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

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

При работе с архивами SUPERVISOR реализует следующие функции: отображение последних изменений значений каналов: просмотр архивов в режиме **PLAYBACK**: просмотр на заданное архивное время с пошаговым переходом по времени.

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

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

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

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

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

1.2 Обмен данными в SCADA-системе TRACE MODE

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

1.2.1. Последовательный интерфейс обмена данными

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

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

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

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

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

1.2.2. Обмен по протоколу M- Link

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

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

Для связи двух мониторов можно использовать интерфейс RS-232. Для организации взаимодействия с несколькими удаленными узлами по этому интерфейсу - необходимо иметь соответствующее ко-

личество последовательных портов. Это позволяет организовать связь типа "звезда". Такая конфигурация может потребовать дополнительных затрат на многоканальные платы. Однако она позволяет быстрее передавать данные за счет распараллеливания обмена с разными удаленными узлами. TRACE MODE поддерживает обмен одновременно по 32 последовательным портам.

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

1.2.3. Организация ввода/вывода данных. Настройка каналов

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

1.2.4. Настройка MPB для обмена по M-Link

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

Параметры обмена по протоколу M_Link настраиваются в бланках «**Основные**» и «**Параметры**» последовательных портов диалога «**Параметры узла**». Для входа в этот диалог необходимо нажать ПК на изображении настраиваемого узла в навигаторе проекта. Статус узла при обмене по протоколу M_Link задается в бланке «**Основные**»

диалога «**Параметры узла**». Чтобы узел поддерживал статус Master, необходимо установить флаг M_Link в разделе **Host Mode** данного бланка, а для поддержки режима SLAVE - тот же флаг в разделе **Slave Mode**.


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

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

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

2. Ход работы

2.1 Создание проекта

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

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

2.2 Подключение модуля удаленного ввода сигналов

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



Рисунок 2.1 – Подключение модуля удаленного ввода I-7033D



Рисунок 2.2 – Настройка модуля удаленного ввода I-7033D

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

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

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

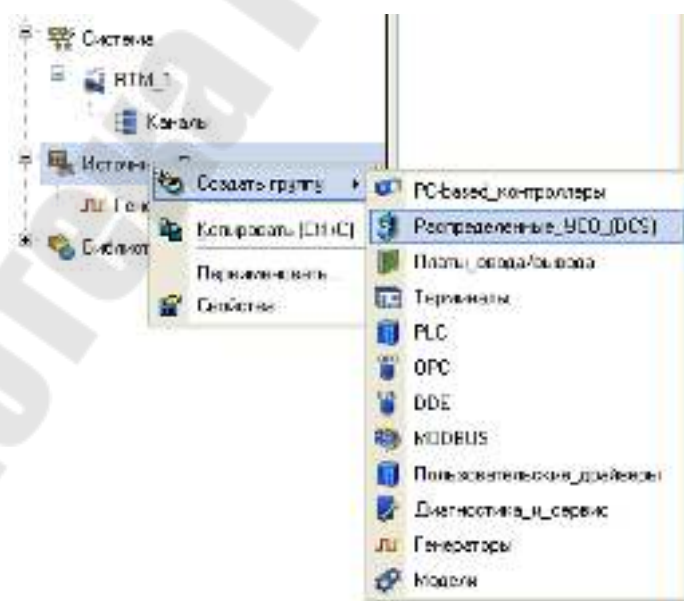


Рисунок 2.3 – Создание группы **Распределенное УСО**

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

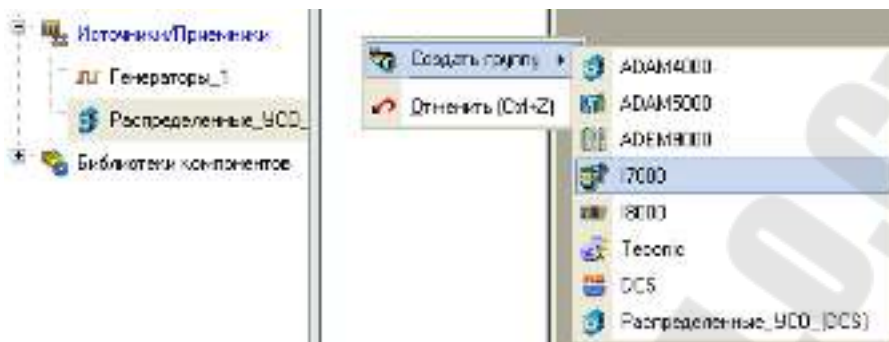


Рисунок 2.4 – Создание группы **I-7000**

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



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

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

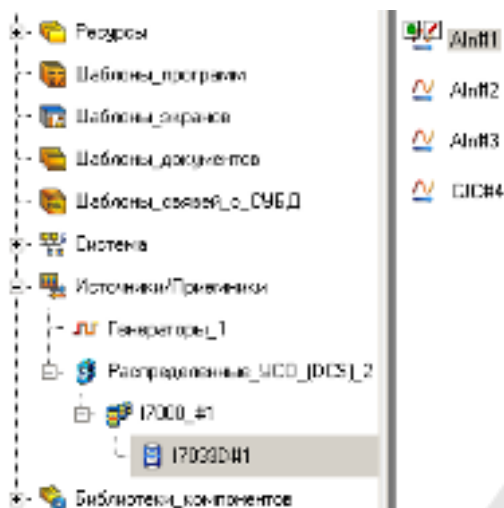


Рисунок 2.6 – Компоненты УСО I7033D

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

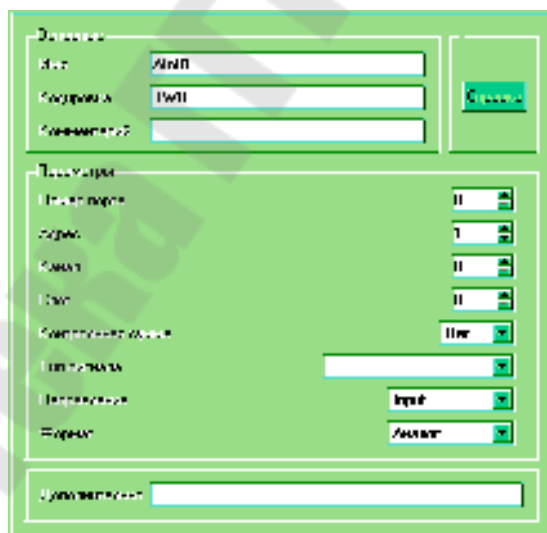


Рисунок 2.7 – Редактирование атрибутов УСО I7033D

2.4 Создание и настройка COM-порта

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

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



Рисунок 2.8 – Создание группы СОМ-порты

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



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

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

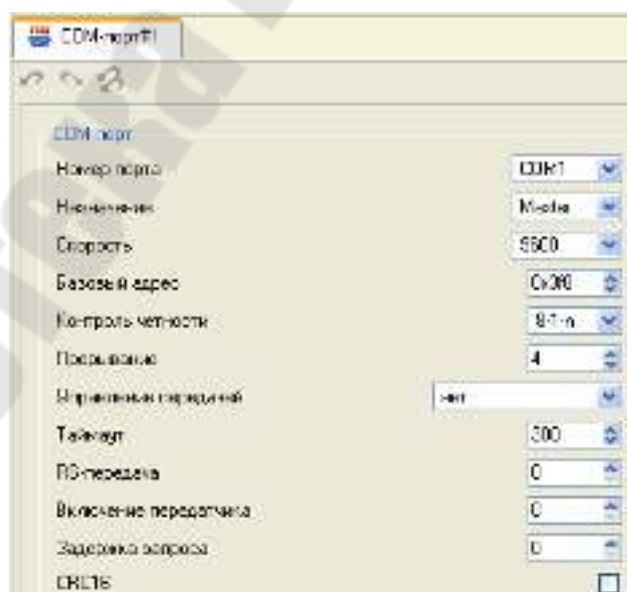






Рисунок 2.10 – Окно атрибутов объекта СОМ-порт#1

2.5 Привязка канала к источнику данных

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

2.6 Запуск проекта

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

3. Контрольные задания

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

4. Содержимое отчета

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

5. Контрольные вопросы

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

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

Лабораторная работа №5
Бинарное и аналоговое управление объектом при помощи
программируемого логического контроллера.
Управление шаговым двигателем посредством
ПЛК Direct Logic 06

Цель работы: Изучить технологию создания системы управления шаговым двигателем с использованием аппаратного обеспечения ПЛК DL06 и программного обеспечения Direct Soft.

1. Теоретические сведения

1.1 Аппаратное обеспечение

Микроконтроллеры серии DL06 комбинируют в себе мощные возможности и компактные размеры. Серия DL06 предлагает расширяемый ввод/вывод, высокоскоростной счетчик, арифметику с плавающей точкой, ПИД-регулирование, ряд коммуникационных возможностей, LCD панель и многое другое.

Для программирования микроконтроллеров доступны два языка: RLL (Relay Ladder Logic – язык релейной логики) и RLLPLUS. RLLPLUS объединяет стандартный язык релейной логики с возможностями стадийного программирования (StageTM). Оба языка одинаково хорошо поддерживаются как ручным программатором, так и пакетом программирования DirectSoftTM.

Все микроконтроллеры DL06 имеют встроенный порт для программирования с использованием ручного программатора (D2-HPP), такой же программатор используется с сериями DL05, DL105 и DL205. Ручной программатор может быть использован для создания, изменения или отладки прикладной программы. Для программирования DL06 нам понадобится D2-HPP со встроенным программным обеспечением версии 2.2 или выше.

Большая часть соединений, индикаторов и обозначений расположены на передней панели микро ПЛК DL06. Порты связи находятся на передней стороне контроллера, так же как слоты для дополнительных модулей и переключатель выбора режимов. См. рисунок 1.1.

Выходные разъемы и разъемы питания принимают внешнее питание AC(L) - AC(N), логическую землю, заземление корпуса на

обозначенных контактах. Оставшиеся клеммы для подключения общих проводов С0-С3 и выходов с Y0 до Y17. Клеммы 16 выходов пронумерованы в восьмеричной системе счисления, от Y0 до Y7 и от Y10 до Y17. В моделях с выходами постоянного тока крайняя правая клемма обеспечивает питание для выходных цепей. Входные разъемы обеспечивают подключение входов X0 и X23 и общих проводов С0-С4.

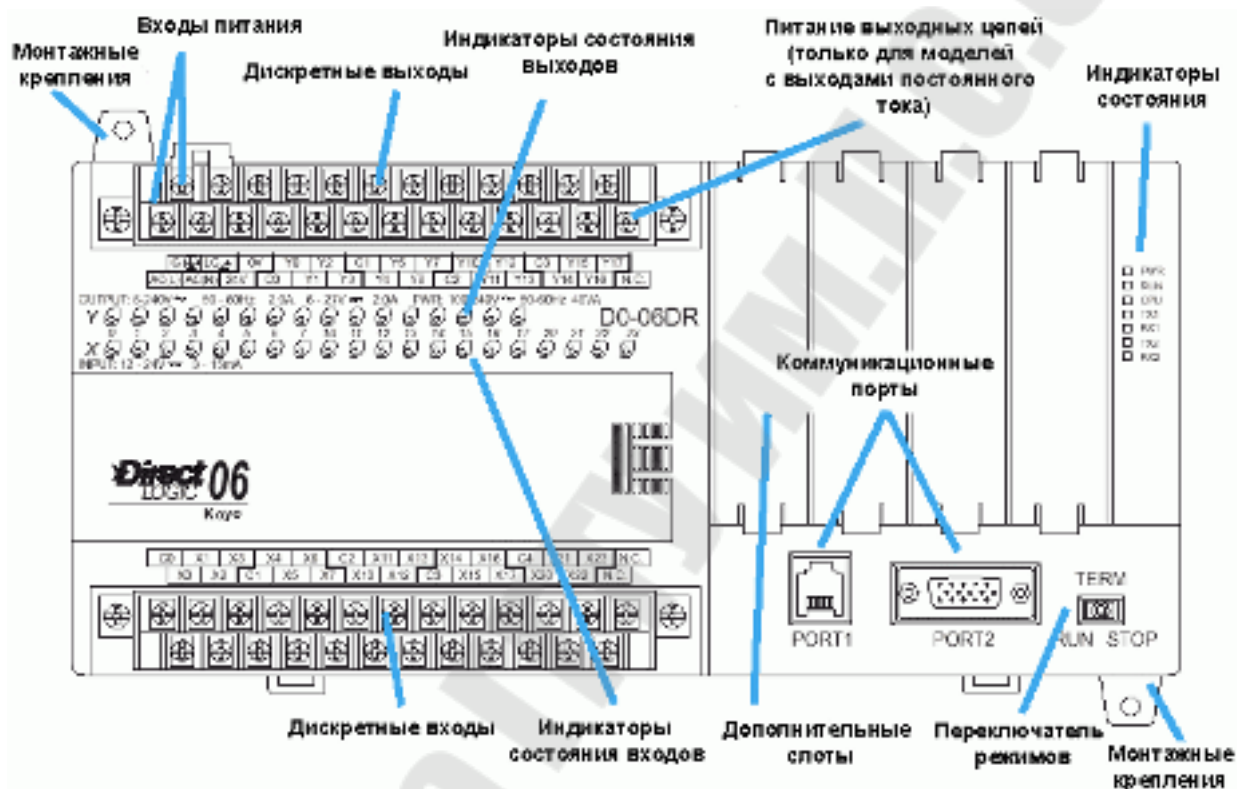


Рисунок 1.1 – Передняя панель DL06

Перед дальнейшим изучением схем подключения мы должны иметь полное понимание понятий «приемник» и «источник» тока. Эти термины часто используются при описании дискретных входных и выходных цепей постоянного тока. Цель данного раздела — облегчить понимание этих понятий. Сначала приводится краткое определение, за которым следуют практические приложения.

Приемник = предусматривает замыкание цепи на землю (-)

Источник = предусматривает замыкание цепи на источник питания (+)

Обратим внимание на полярность. Понятия «приемник» и

«источник» относятся только к цепям постоянного тока. Входные и выходные точки, которые являются приемниками либо источниками, могут проводить ток только в одном направлении. Это означает, что можно подсоединить внешний источник питания и полевые устройства к точке Ввода/Вывода с обратным направлением тока, и эти цепи не будут работать. Таким образом, мы можем приступить к монтажу только при понимании понятий «Приемник» и «Источник». Например, на рисунке 1.2 справа показан вход — «приемник». Для правильного подсоединения внешнего источника питания, вы должны осуществить это соединение таким образом, чтобы вход обеспечивал проводимость к заземлению(-).

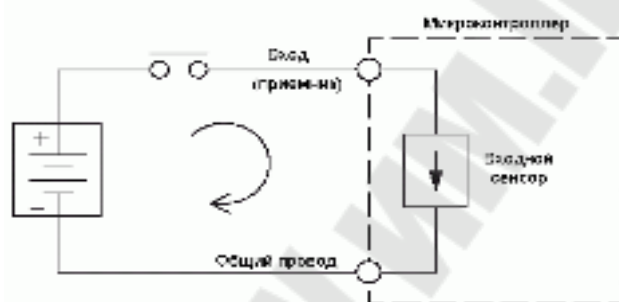


Рисунок 1.2 – Схема подключения источника информации ко входу ПЛК

Начнем с входной клеммы ПЛК, продолжаем через цепи считывания входа ПЛК и закончим на общем полюсе, соединим источник питания (-) с общим полюсом. После добавления выключателя между источником питания (+) и входом мы замкнем цепь. Если замкнуть выключатель, ток потечет в направлении, указанном стрелками. Применяя указанный принцип построения цепи, получим показанные ниже четыре возможные схемы входного/выходного приемника/источника. У микроконтроллеров DL06 входы могут быть приемником или источником и выходы или приемником или источником. Любая пара схем ввода/вывода, показанных ниже, возможна с одной из моделей DL06.

Для того чтобы цепи Ввода/Вывода работали, необходимо, чтобы ток входил в одну клемму и выходил через другую. Поэтому с каждым дискретным Вводом/Выводом связаны, по крайней мере, две клеммы. На рисунке справа клемма для Входа или Выхода предназначена для питающего провода тока. Еще одна клемма необходима для обратного провода (к источнику питания).

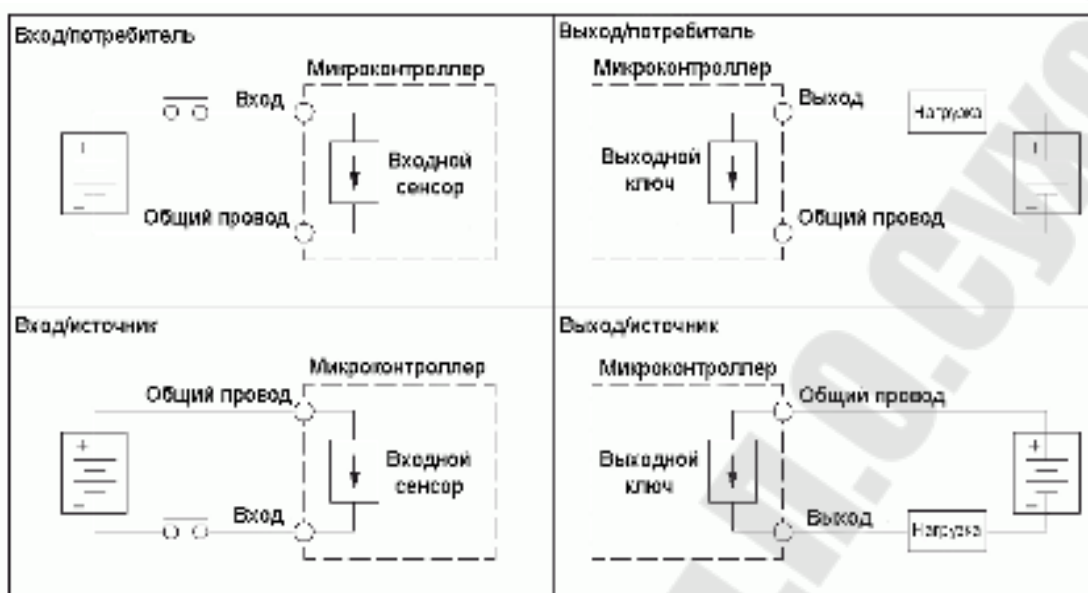


Рисунок 1.3 – Варианты схем подключения входов/выходов ПЛК

Практически все дискретные входы или выходы на ПЛК совместно используют путь возврата с другими точками входа/выхода. На рисунке 1.4 справа показана группа (или блок из 4 входов, которые совместно используют обратный общий провод. В этом случае четыре входа требуют только пять клемм вместо восьми.

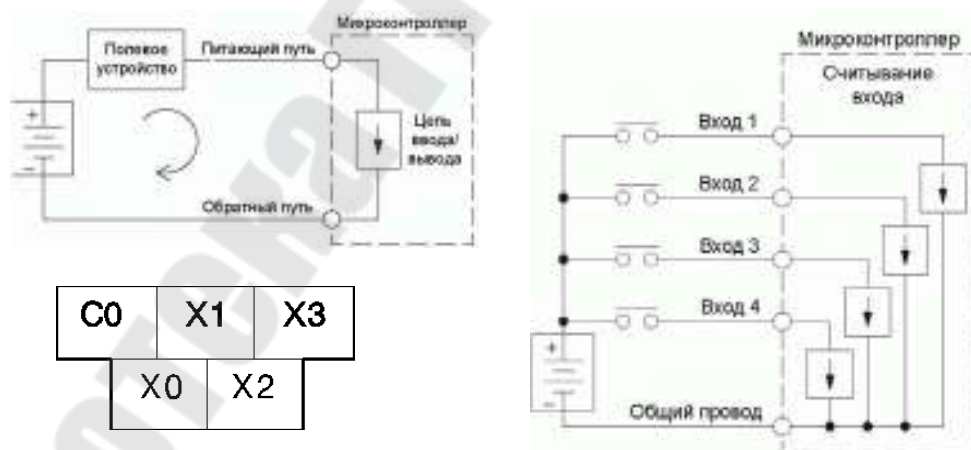


Рисунок 1.4 – Вариант схемы с объединением общего провода источников

Большинство входных и выходных цепей DL06 объединены в блоки, которые совместно используют обратный провод. Хорошим указателем такой группы Ввода/Вывода являются монтажные мет-

ки. Общие блоки отделены жирной линией. Более тонкая линия отделяет входы связанные с тем общим проводом. На рисунке справа X0, X1, X2 и X3 совместно используют общий провод C0, расположенный слева от X1.

Приведенная ниже маркировка показывает наличие пяти групп по 4 входа и четырех групп по 4 выхода. Для каждой группы существует один общий канал.

G (D)	LG	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17	
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	N.C.
C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

Рисунок 1.5 – Группировка входов/выходов DL06

1.2 Программное обеспечение

1.2.1. Пакет программирования DirectSOFT32 под Windows

Микроконтроллеры DL06 можно программировать с помощью графического пакета DirectSOFT (версия 4 или выше), который поддерживает многие хорошо известные функции Windows, такие как «вырезать-вставить» между приложениями, редактирование по щелчку мыши, просмотр и редактирование нескольких прикладных программ одновременно, и т. д. DirectSOFT32 (PC-PGMSW) поддерживает все серии контроллеров DirectLOGIC. Таким образом, мы можете использовать один и тот же пакет для программирования DL05, DL06, DL105, DL205 и DL405. (При переходе на новый процессор может понадобиться обновление версии программного обеспечения). описывается в отдельном руководстве. Для программирования DL06 нам потребуется версия 4.0 или выше.

1.2.2. Варианты программирования DL06

Микроконтроллер DL06 можно запрограммировать с помощью ручного программатора или компьютерной программы DirectSOFT.

Подсоединим DL06 к компьютеру с помощью кабеля, показанного на рисунке 1.6.

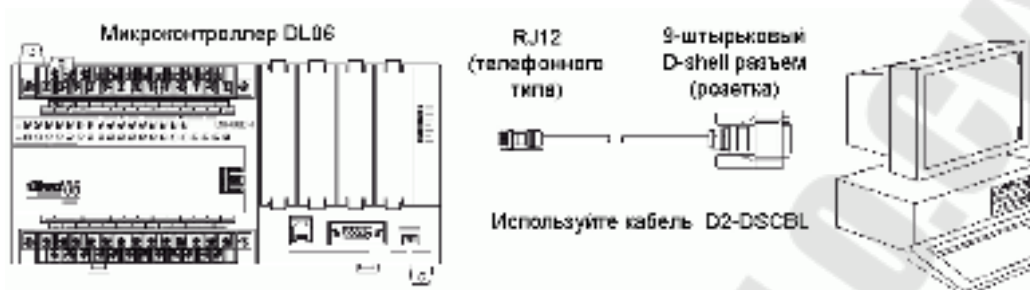


Рисунок 1.6 – Вариант программирования с использованием ПК



Рисунок 1.7 – Вариант программирования с использованием ручного программатора D2-NPP

1.2.3. Встроенное решение по управлению движением

Для многих приложений по управлению машинами требуются различные типы простого высокоскоростного слежения и управления. Эти приложения обычно включают отдельные типы управления движением или высокоскоростные прерывания для критичных по времени событий. Микроконтроллер DL06 решает эти традиционно сложные задачи с помощью встроенных технических возможностей процессора.

Функциями высокоскоростного ввода являются:

- Высокоскоростной счетчик (максимум 7КГц) с 24 заданными параметрами и встроенной подпрограммой прерывания, со счетом в одном направлении и сбросом;
- Входы квадратурного кодировщика угла поворота (эн-

кодера) для отсчета в направлениях по часовой стрелке или против часовой стрелки (до 7КГц), со счетом в прямом/обратном направлениях и сбросом.

- Высокоскоростной вход прерывания для немедленной реакции в критичных по времени задачах;

- Функция защелки (фиксации) импульсов при отслеживании одной входной точки с длительностью импульса до 100 мкс (0.1 мс);

- Программируемая дискретная фильтрация (с задержкой по времени как на включение, так и на выключение сигнала до 99 мс), обеспечивающая целостность сигнала (этот режим установлен по умолчанию для входов X0 – X3);

Функциями импульсного вывода являются:

- Одноканальный программируемый импульсный выход (максимум 10КГц) с тремя типами профилей, включая трапецеидальное движение, точное совмещение и управление скоростью.

Применение функций имеет следующие ограничения:

- *Варианты высокоскоростного ввода доступны только для DL06 с входами постоянного тока;*

- *Варианты импульсного вывода доступны только для DL06 с выходами постоянного тока;*

- *Одновременно может использоваться только одна функция высокоскоростного ввода/вывода. Вы не можете использовать высокоскоростной вход импульсный выход в одно и то же время.*

1.2.4. Специализированная схема высокоскоростного ввода/вывода

Основной внутренней задачей процессора является выполнение пользовательской программы и чтение/запись входов и выходов в каждом цикле сканирования. Для обслуживания событий высокоскоростного ввода/вывода DL06 имеет специальную схему, которая предназначена для части входных/выходных точек. См. блок-схему на рисунке 1.8.

Схема высокоскоростного ввода/вывода (HSIO) предназначена для первых четырех входов (X0 – X3) и первых двух выходов (Y0 - Y1). Эту схему можно рассматривать как «помощник процессора».

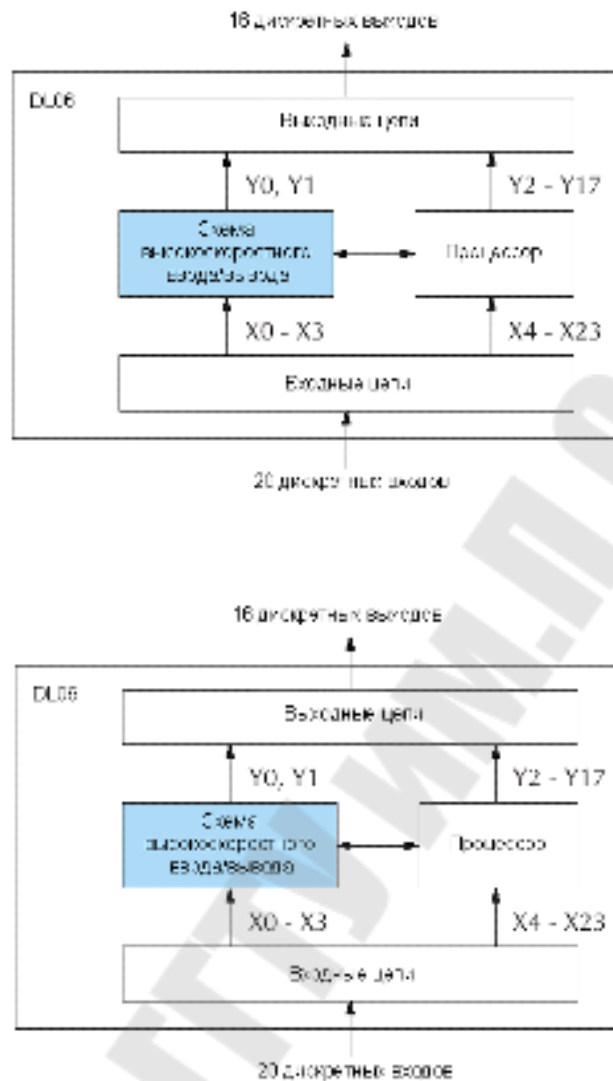


Рисунок 1.8 – Схема взаимодействия блока высокоскоростного ввода/вывода

В режиме по умолчанию (называемым «Режим 60») схема высокоскоростного ввода/вывода только пропускает входные/выходные сигналы в процессор и из процессора, поэтому все двадцать входов работают одинаково, и все шестнадцать выходов также работают одинаково. Когда процессор конфигурируется для какого-либо режима высокоскоростного ввода/вывода, то эта схема получает специализированную функцию для указанной части входов и выходов. Схема высокоскоростного ввода/вывода функционирует независимо от сканирования программ процессором. Это позволяет проводить точные измерения и осуществлять работу высокоскоростного ввода/вывода, пока процессор занят выполнением пользовательской программы.

1.2.5. Режимы работы блока высокоскоростного ввода/вывода

Схема высокоскоростного ввода/вывода работает в одном из 6 базовых режимов, перечисленных ниже в таблице 1. Число в левом столбце является номером режима (позжемы будем использовать этот номер при конфигурировании). Выберем один из режимов в соответствии с основной функцией, которую мы хотим получить от схемы высокоскоростного ввода/вывода. Мы можем просто использовать все двадцать входов и шестнадцать выходов как обычные точки ввода/вывода в режиме 60.

Таблица 1 –Режимы работы блока высокосортного ввода/вывода

Наименование режима		Характеристика режима
10	Высокоскоростной счетчик	Два счетчика 7КГц с 24 заданными параметрами, со счетом в одном направлении, со сбросом и прерывание по заданному значению
20	Реверсивный счетчик	Реверсивный счетчик 7КГц с 24 заданными параметрами, со сбросом и прерыванием по заданному значению
		Квадратурный вход 7КГц по каналу А / каналу В, со счетом в прямом и обратном направлениях
30	Импульсный выход	Управление шаговым приводом - импульсные сигналы и сигналы направления, программируемый профиль движения
40	Высокоскоростное прерывание	Генерирует прерывание исходя из перехода состояния входа или по времени
50	Зашелка (фиксатор)	Захватывает короткие импульсы в выбранном входе
60	Фильтрованный вход	Отфильтровывает короткие импульсы в выбранном входе

При выборе одного из шести режимов высокоскоростного ввода/вывода перечисленные в таблице ниже точки входа/выхода реализуют только указанные в таблице функции. Если вход не используется специально для поддержания конкретного режима, то он обычно работает по умолчанию как фильтрованный вход. Аналогично выход функционирует как обычный выход, если не

выбран режим импульсного выхода.

1.2.5. Режим по умолчанию

Режим 60 (Фильтрованные входы) является режимом по умолчанию. DL06 устанавливается в этом режиме в заводских условиях, и в любой момент мы можем восстановить память с этой информацией. В режиме по умолчанию X0 – X3 являются фильтрованными входами (задержка 10мс), а Y0 - Y1 являются стандартными выходами. Если мы выбрали подходящий для нашего приложения режим высокоскоростного ввода/вывода, то можно произвести соответствующее конфигурирование работы ПЛК. На приведенной ниже блок-схеме обращаем внимание на элемент V-памяти в блоке Процессор. Ячейка V-памяти V7633 определяет функциональный режим высокоскоростного ввода/вывода. Это — наиболее важная величина для выбора функций высокоскоростного ввода/вывода!

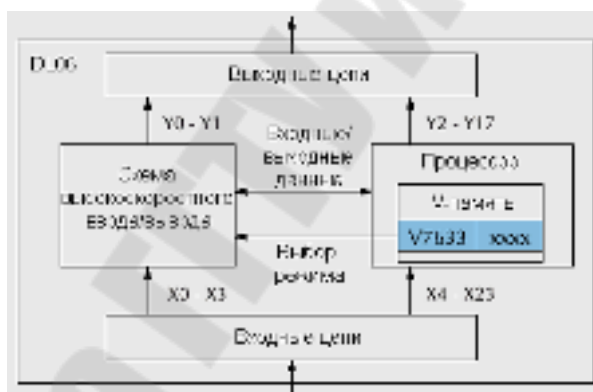


Рисунок 1.9 – Схема взаимодействия данных

Ячейка V7633 содержит 16-битовое слово, которое должно вводиться в двоично-десятичном формате. На рисунке ниже показано, что представляет собой каждая 4-битовая цифра этого двоично-десятичного слова. Биты 0 - 7 определяют указанный ранее номер режима. Например, значение «2050» показывает выбор режима 50 - фиксация импульсов (в двоично-десятичном формате = 50).

Кроме настройки ячейки V7633 для режима HSIO, нам необходимо запрограммировать следующие четыре ячейки для заданного режима в соответствии с желаемой функцией входных точек X0 – X3. Может потребоваться конфигурирование других ячеек V-памяти в зависимости от режима HSIO.

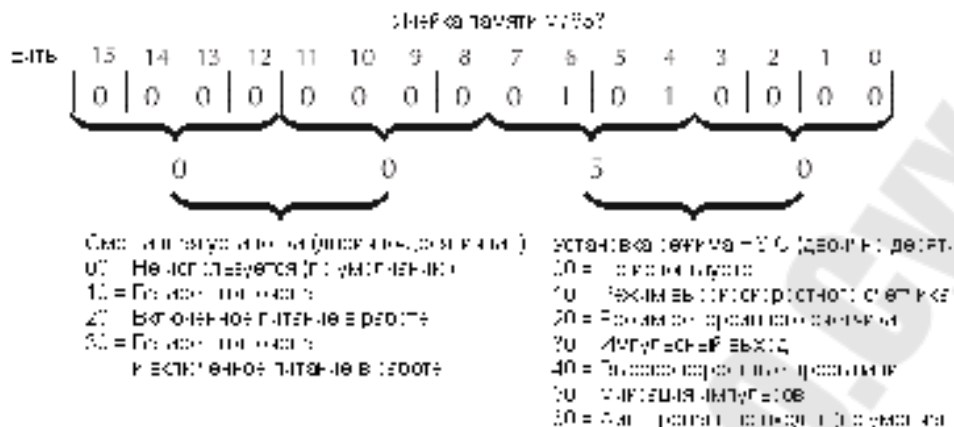


Рисунок 1.10 – Структура ячейки памяти V7633

Функциональная блок-схема приведена на рисунке 1.11. Если младший байт регистра режимов HSIO V7633 содержит двоично-десятичное число «10», то в схеме HSIO включается высокоскоростной суммирующий счетчик. X0 и X1 автоматически становятся входами «генератора импульсов» для высокоскоростного счетчика, наращивая его при каждом переходе «выключено — включено». По умолчанию X2 и X3 в конфигурации для режима 10 являются входами внешнего сброса.

Мы можем сконфигурировать X2 и X3 как обычные фильтрованные входы, вместо того чтобы использовать их как специализированные входы сброса. В этом случае сброс счетчика должен вырабатываться программой релейной логики.

Ниже показана обобщенная схема подсоединения счетчиков/кодировщиков к DL06 в режиме 10. Могут использоваться многие типы устройств генерирующих импульсы, такие как бесконтактные переключатели, одноканальные кодировщики, магнитные и оптические чувствительные элементы и др. Устройства с выходами-потребителями тока (транзисторы с открытым коллектором NPN). Если счетчик является источником питания для входов, он должен иметь на выходе от 12 до 24 В постоянного тока. Обращаем внимание на то, что устройства с выходами 5В-источник, не будут работать с входами DL06.

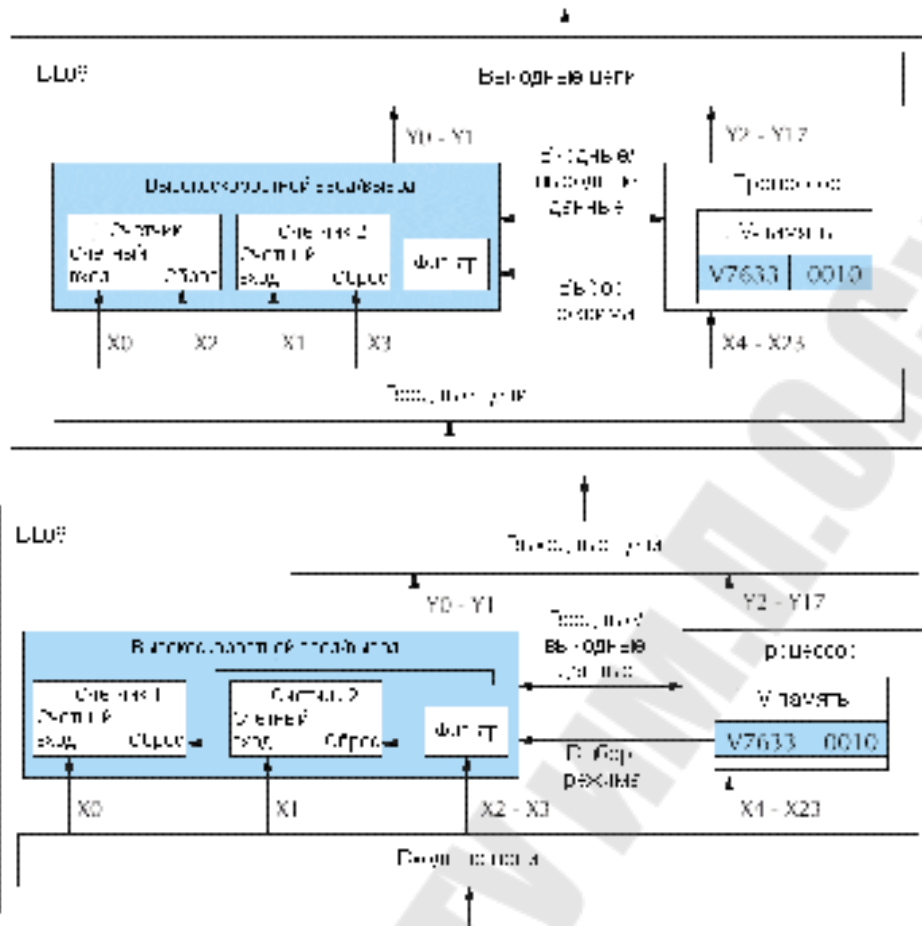


Рисунок 1.11 – Функциональная блок схема вариантов работы DL06

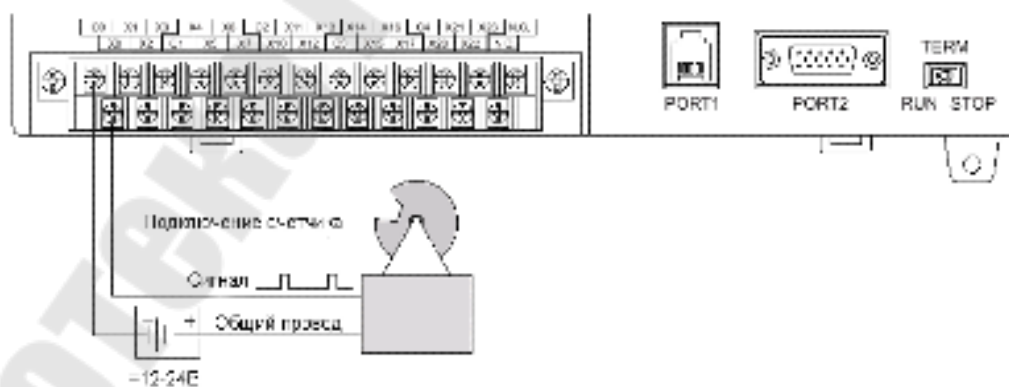


Рисунок 1.12 – Схема подключения источника импульсов

1.3. Особенности программирования DL06

1.3.1. Заданные значения и специальные реле

Предварительная установка используется, чтобы выполнить оп-

ределенное действие, когда при подсчете будет достигнуто заданное значение. Посмотрим рисунок 1.13. Каждый счетчик может иметь до 24 предварительно заданных значений (Preset Value), которые могут быть запрограммированы. Предварительно заданные значения представляют двойное слово так, что они занимают два регистра V-памяти. Пользователь выбирает предварительно установленные значения, и счетчик непрерывно сравнивает текущий счет с предварительно установленным. Когда они равны, контакты специального реле замыкаются, и происходит переход к выполнению подпрограммы прерывания. Рекомендуется использовать специальное(ые) реле в сервисной подпрограмме прерывания, чтобы обеспечить немедленное выполнение желательных действий. После завершения сервисной подпрограммы прерывания процессор возвращается в пользовательскую программу и продолжает ее выполнения с точки прерывания. Далее производится сравнение со следующим заданным значением.

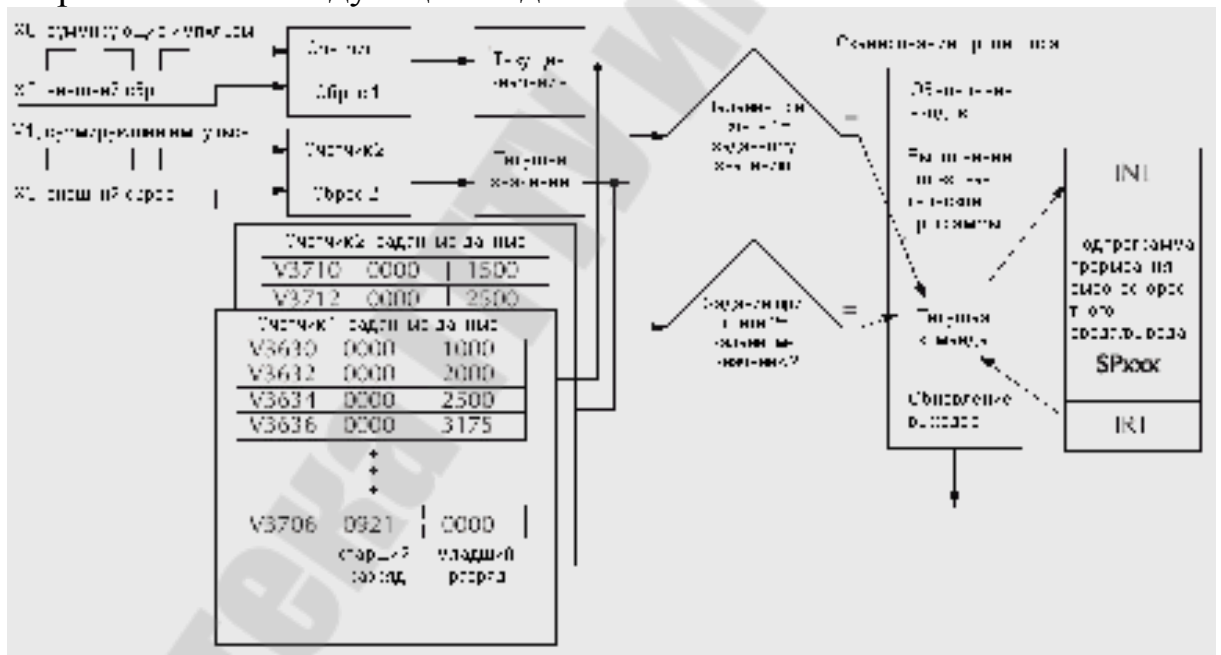


Рисунок 1.13 – Схема работы счетчиков

1.3.2. Абсолютные и инкрементальные предустановленные значения

Доступны два режима предустановленных значений: абсолютный и инкрементальный. Предустановленные значения предвари-

тельно записываются в регистрах V- памяти. В абсолютном режиме, каждое значение обрабатывается как итоговое. В инкрементном режиме предварительно заданное значение используется как накопленное. Инкрементные значения представляют собой значения счетчика между событиями.

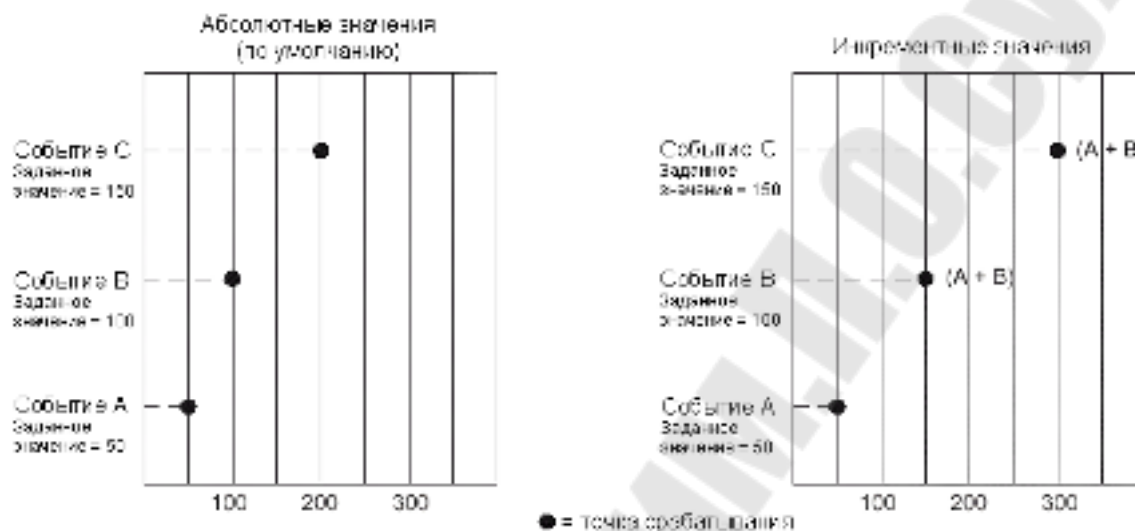


Рисунок 1.14 – Пример использования абсолютных и инкрементных значений

В приведенном примере заданные значения равны 50, 100 и 150. Различие между абсолютным и инкрементальным режимами показано. Абсолютные уставки запускают события по заданным значениям: 50, 100 и 200. Инкрементальные уставки запускают события по накопленным значениям: 50, 150 и 300.

1.3.3. Начальная ячейка заданных данных

V7630 является ячейкой V-памяти указателя, которая указывает на Таблицу Заданных Данных. По умолчанию начальной ячейкой для Таблицы Заданных Данных является V3630 (значение по умолчанию устанавливается после инициации оперативной V-памяти). Однако мы можем программным путем изменить в V7630 адрес этой ячейки. Используйте команды LDA и OUT следующим образом:

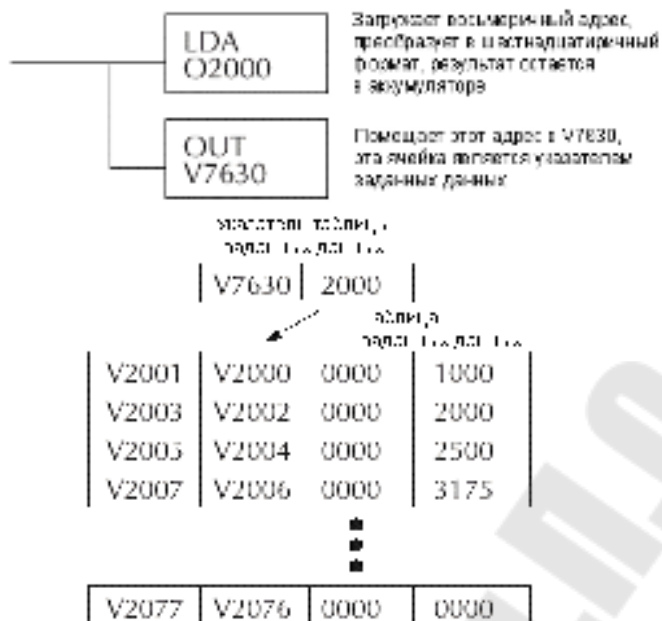


Рисунок 1.15 – Программный фрагмент начальной инициализации

1.3.4. Использование менее 24 заданных значений

Если используются все 24 заданных значения, процессор заведомо знает, когда достигается конец таблицы заданных данных. Когда используется менее 24 значений необходимо оповестить процессор о том, что достигнуто последнее значение. Способ оповещения о конце блока предварительно заданных значений состоит в том, чтобы вставить один из следующих кодов «конец таблицы» в следующую доступную пару регистров.

Как показано в таблице выше, каждый из сигналов «конец таблицы» имеет различное значение. Используйте команду LDDKfff, чтобы вставить код конца таблицы в следующую пару регистров для выхода из таблицы. В примере используется четыре заданных значения. 0000 FFFF в V3641-V3640 указывает, что предыдущее заданное значение было последним значением.

В абсолютном режиме счетчик и накопленное значение каждый раз сбрасываются, когда достигнуто заданное значение. В инкрементном режиме, мы можем выбрать: не сбрасывать счетчик или накопленное значение, можем сбросить только счетчик или сбросить и счетчик и накопленное значение, когда код таблица-конец

прочитан. В примере FFFF был помещен в V3640, так как последнее заданное значение было в V3636 и мы используем меньше чем 24 предварительно заданных значений.

Код конца таблицы	Применяемый режим	Значение
0000 FFFF	Абсолютный и инкрементный	Сообщение о достижении последнего значения
0000 00FF	Инкрементный	Сообщение о достижении последнего значения и повторный запуск заданных значений. Не сбрасывается накопленный отчет импульсов СТ174 или СТ176.
0000 FF00	Инкрементный	Сообщение о достижении последнего значения, повторный запуск заданных значений и сброс накопленного отчета импульсов СТ174 или СТ176.



Рисунок 1.16 – Программный фрагмент вставки кода конца таблицы

1.3.4. Номера эквивалентных реле

В приводимой таблице 2 перечислены все 24 установленные по умолчанию ячейки с регистрами заданных значений для каждого высокоскоростного счетчика. Каждое такое значение занимает два 16-битовых регистра в V-памяти. Номера контактов соответствующих специальных реле приведены в следующем столбце. Можно назвать эти контакты реле «эквивалентными», так как они ис-

тинны (замкнуты), когда текущее значение высокоскоростного счетчика равно заданному значению. Каждый контакт остается закрытым до тех пор, пока значение счетчика не станет равным следующему заданному значению.

Последовательные адреса показанные выше для каждого реле – адреса назначенные процессором по умолчанию. Указатель начального адреса хранится процессором в ячейке V7630. Если мы уже использовали эти адреса, мы можем изменить блок адресов по умолчанию программным способом, изменяя указатель значения в ячейке V7630. Для изменения положения таблицы используем команды LDA и OUT, как показано на предыдущей странице.

Таблица 2 – Номера эквивалентных реле

Счетчик 1 Заданное значение	Регистр заданного значения в V-памяти	Номер специального реле	Счетчик 2 Заданное значение	Регистр заданного значения в V-памяти	Номер специального реле
1	V3631 / V3630	SP540	1	V3711/V3710	SP570
2	V3633 / V3632	SP541	2	V3713/V3712	SP571
3	V3635 / V3634	SP542	3	V3715/V3714	SP572
4	V3637 / V3636	SP543	4	V3717/V3716	SP573
5	V3641 / V3640	SP544	5	V3721/V3720	SP574
6	V3643 / V3642	SP545	6	V3723/V3722	SP575
7	V3645 / V3644	SP546	7	V3725/V3724	SP576
8	V3647 / V3646	SP547	8	V3727/V3726	SP577
9	V3651 / V3650	SP550	9	V3731/V3730	SP600
10	V3653 / V3652	SP551	10	V3733/V3732	SP601
11	V3655 / V3654	SP552	11	V3735/V3734	SP602
12	V3657 / V3656	SP553	12	V3737/V3736	SP603
13	V3661 / V3660	SP554	13	V3741/V3740	SP604
14	V3663 / V3662	SP555	14	V3743/V3742	SP605
15	V3665 / V3664	SP556	15	V3745/V3744	SP606
16	V3667 / V3666	SP557	16	V3747/V3746	SP607
17	V3671 / V3670	SP560	17	V3751/V3750	SP610
18	V3673 / V3672	SP561	18	V3753/V3752	SP611
19	V3675 / V3674	SP562	19	V3755/V3754	SP612
20	V3677 / V3676	SP563	20	V3757/V3756	SP613
21	V3701 / V3700	SP564	21	V3761/V3760	SP614
22	V3703 / V3702	SP565	22	V3763/V3762	SP615
23	V3705 / V3704	SP566	23	V3765/V3764	SP616
24	V3707 / V3706	SP567	24	V3767/V3766	SP617

1.3.5. Конфигурирование входа X

Варианты конфигурируемого дискретного входа для режима высокоскоростного счетчика приведены в таблице ниже. Вход X0 выделен как вход импульсов первого счетчика. Вход X1

может быть импульсным входом для второго счетчика или фильтрованным.

Таблица 3 – Конфигурация входа X

Вход	Регистр конфигурации	Функция	Требуемый шестнадцатеричный код
X0	V7634	Счетные импульсы #1	0001 (абсолютный) (по умолчанию)
			0101 (инкрементный)
X1	V7635	Счетные импульсы #2	0001(абсолютный) (по умолчанию)
			0101 (инкрементный)
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)
X2	V7636	Сброс счет-	0007* (по умолчанию)
		Сброс счет-	0107*
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)
X3	V7637	Сброс	0007* (по умолчанию)
		Сброс	0107*
		Прерывание	0004
		Импульсный вход	0005
		Фильтрованный вход	xx06, xx= время фильтрации от 0 до 99 мс (двоично-десятичных)

Входы X2 и X3 могут быть сконфигурированы как сброс счетчика, с прерыванием или без него. Вариант с прерыванием позволяет

входу сброса (X2 и X3) вызвать прерывание аналогично тому, как это делает заданное значение, но без замыкания контактов специального реле (вместо этого X2 и X3 будут в состоянии включен в течение работы подпрограммы прерывания в одном цикле сканирования). X2 и X3 могут просто остаться как фильтрованный вход.

1.4. Работа со средой Direct Soft

Для того чтобы начать редактировать программу, нужно открыть DirectSOFT 5.

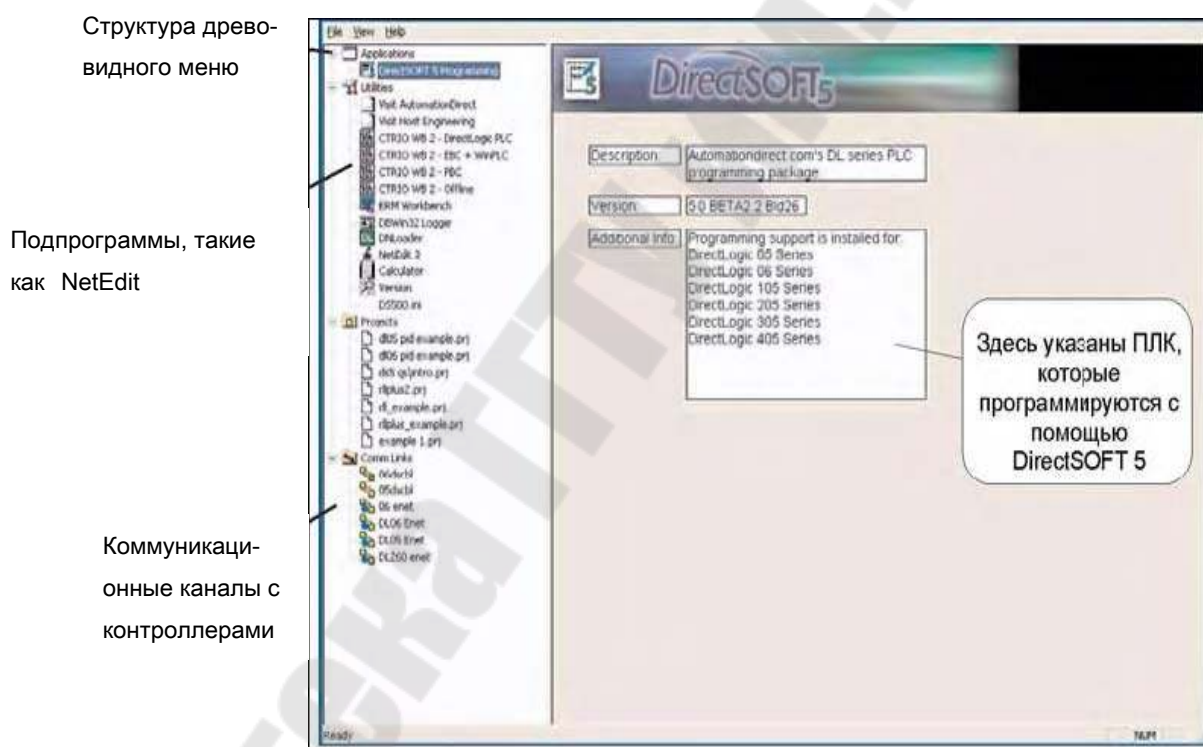


Рисунок 1.18 – Начальная страница Direct Soft

Из этого окна, как из единого центра, могут быть запущены дополнительные подпрограммы, такие как NetEdit, STRIO WB и другие. Это окно может быть также использовано для создания программ ПЛК и управления этими программами, а также для управления коммуникационными каналами между компьютером и ПЛК. Обратите

внимание на различные области, которые указаны слева от окна запуска.

- Applications (Приложения) – Это приложения, к настоящему времени установленные в DirectSOFT 5. Ярлычки этих приложений можно увидеть в древовидном меню под папкой/иконкой Applications, эти ярлычки связаны с приложениями, которые разработаны для запуска из DirectSOFT 5. Например, для создания новой программы дважды щелкните по имени DirectSOFT 5 Programming.

- Utilities (Подпрограммы) – Под этой папкой/иконкой имеется несколько подпрограмм. Если вы уже установили такие подпрограммы, как NetEdit, STRIO Workbench, то они будут показаны здесь. Обратите внимание, что теперь имеются три новые подпрограммы: ярлычок к калькулятору Microsoft®, который (калькулятор) поставляется с Microsoft® Windows, прямое соединение с сайтом AutomationDirect и прямое соединение с сайтом Host Engineering. Ярлычки к наиболее часто используемым подпрограммам можно добавить в раздел Utilities файла DS500.ini.

- Projects (Проекты) – Это программы, которые создаются с помощью DirectSOFT 5. Проект – это общее название для создаваемой программы и сопутствующей ей документации. Когда вы создаете новый проект или работаете над существующим проектом, то имя этого проекта будет в древовидном меню в списке проектов под папкой/иконкой Projects. Для того чтобы открыть существующий проект, дважды щелкните по имени проекта. Для того чтобы открыть проект, который не находится в списке проектов, щелкните правой клавишей мыши по Projects, затем выберите Browse (Обзор) для поиска проекта, потом выберите его.

- Comm Links (Коммуникационные каналы) - Под этой папкой/иконкой указываются коммуникационные каналы, используемые для связи компьютера с одним или несколькими ПЛК. Если в предыдущей версии DirectSOFT уже были созданы коммуникационные каналы, то они появятся здесь. Новые коммуникационные каналы также появятся в этой папке после того, как они будут созданы.

Для того чтобы начать проект дважды щелкните по DirectSOFT 5 Programming под папкой Applications в древовидном меню. Появится окно, приведенное ниже. Это окно New Project (Новый проект) предназначено для ввода основной информации о новом проекте. Дайте имя новому проекту, затем передвиньте курсор в область Family (Семейство) и выберите семейство, к которому принадлежит программ-

руемый вами ПЛК. Потом выберите тип процессорного модуля. После ввода всей этой информации щелкните по ОК. Имейте в виду, что имеющиеся в DirectSOFT мнемосхемы, правила обработки и инструментальные панели будут согласованы с вашим выбором семейства и типа процессорного модуля.

Окно для программирования немного отличается от такого же окна в предыдущих версиях пакета программирования DirectSOFT. Этот вид окна появляется по умолчанию всегда, когда открывается новый проект. Панели инструментов “Online” (Программирование ПЛК в реальном времени) и “Offline” (Программирование, когда ПЛК отключен) расположены так же, как и в окнах программирования предыдущей версии DirectSOFT, только кнопки на них имеют другие пиктограммы. Обратите внимание на то, что некоторые пиктограммы на кнопках панели инструментов бледно-серого цвета – эти кнопки недоступны, когда как другие кнопки с цветными пиктограммами активны. Во время редактирования программы некоторые из бледно-серых кнопок становятся доступными (цветными). Панель инструментов Online – серая и остается в таком виде до тех пор, пока компьютер не связан с ПЛК.

По умолчанию, когда открывается новый проект, окно программирования по является в виде двух окон. Одно окно Cross Reference View (Окно перекрестных ссылок) размещено слева и другое Ladder View (Окно программы) - справа. Окно перекрестных ссылок – это одно из новых перемещаемых («плавающих») окон в DirectSOFT 5. Оно включает в себя также окна Data Views (Окно данных) и Output (Окно вывода). Эти окна могут быть закреплены («пришвартованы») к любому краю окна программирования или они могут быть «отшвартованы» и перемещены в любую часть экрана или даже на другой дисплей, если у вас к компьютеру подключено несколько мониторов. Если окно пришвартовано, то его можно спрятать, щелкнув по кнопке свертывания в правом верхнем углу этого окна. Окно при этом свернется слева от окна программы в закладку, на которой будет написано имя окна. Чтобы снова развернуть окно, наведите курсор мыши на имя окна в закладке. Если окно совсем не нужно, то закройте его, щелкнув по кнопке X, расположенной справа от кнопки свертывания окна. Обратите внимание, что панель инструментов программирования расположена справа от окна программы. Если не активирован режим редактирования, то кнопки элементов программы на панели инструментов программирования будут серыми. Для активации панели

инструментов программирования щелкните по любой кнопке EDIT Mode (Режим редактирования); одна кнопка расположена на панели инструментов Offline и другая – на панели инструментов редактирования сверху. Эту панель инструментов программирования можно расположить в любом месте экрана, захватив курсором мыши полосу перетаскивания этой панели и передвигая ее в новое место.

С окном программы можно работать в двух режимах: в режиме просмотра - Display Mode и в режиме редактирования – Edit Mode. Когда открывается новая или существующая программа, окно программы находится в режиме просмотра. В этом режиме программу нельзя редактировать. Для редактирования программы необходимо перейти в режим редактирования. Для включения режима редактирования или щелкните по кнопке режима редактирования на панели инструментов Offline, или щелкните по кнопке режима редактирования на панели инструментов программирования. Когда режим редактирования включен, курсор в виде прямоугольника заливается сплошным однородным цветом, вокруг кнопок включения режима редактирования появляется окантовка в виде прямоугольника и элементы на панели инструментов программирования подсвечиваются.

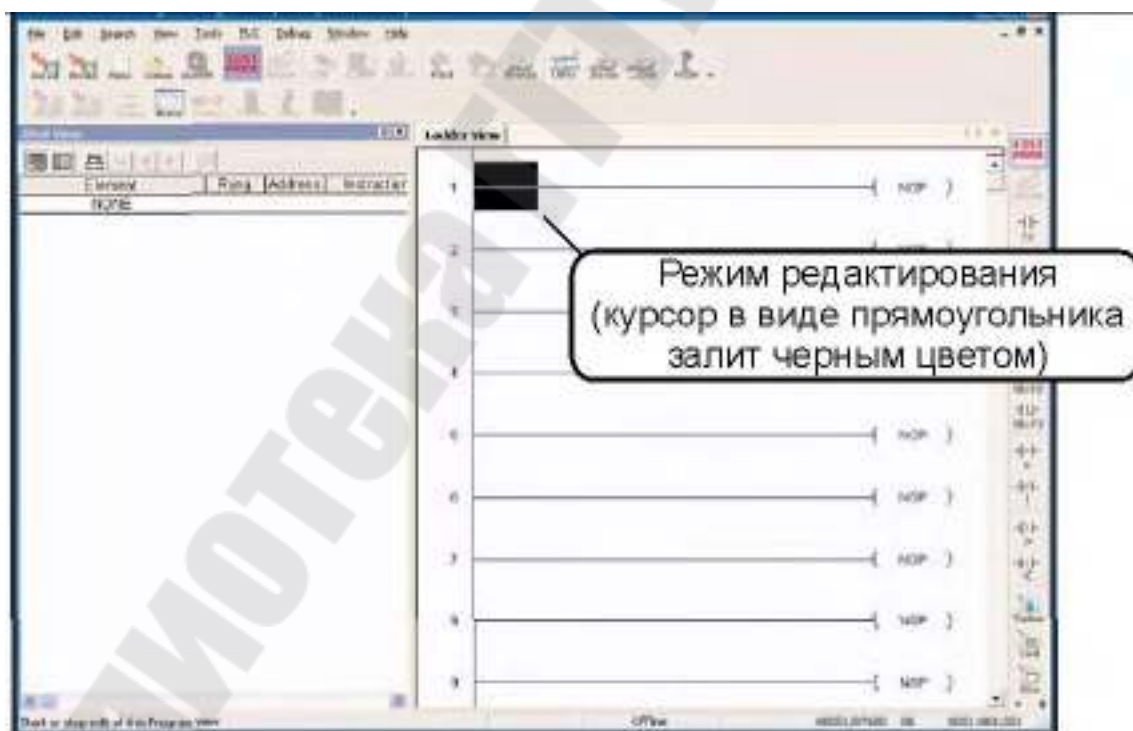


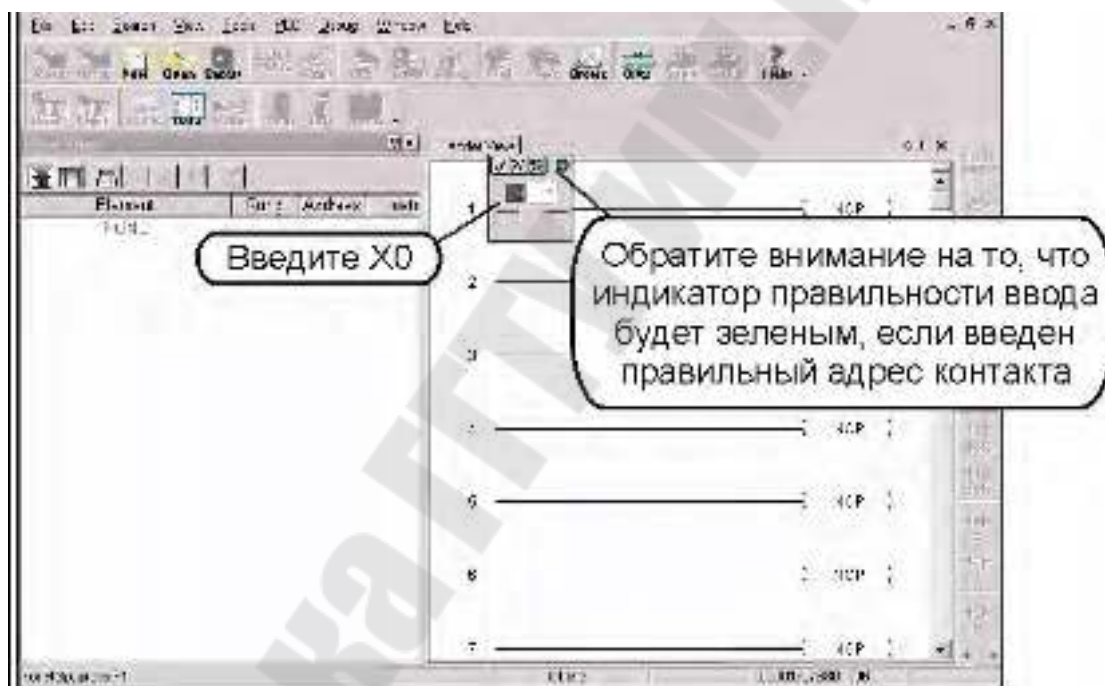
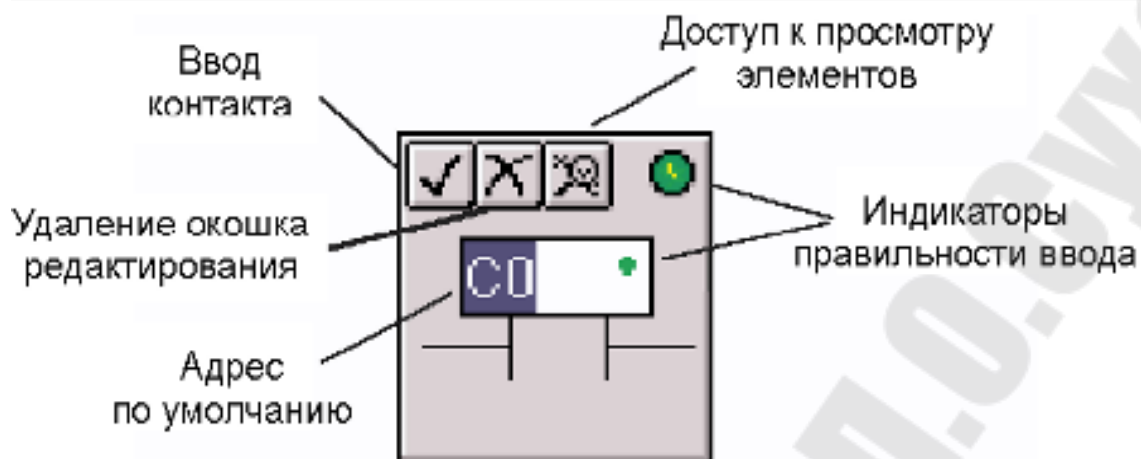
Рисунок 1.18 – Режим редактирования

Панель инструментов программирования, показанная ниже, может выглядеть не совсем так, как вы ее увидите на экране компьютера. Набор инструментов на панели зависит от типа процессорного модуля в ПЛК. В примере на рисунке показаны только инструменты, общие для всех процессорных модулей.



Рисунок 1.19 – Панель инструментов

Для ввода первой команды в программе воспользуйтесь панелью инструментов для программирования. Сначала щелкните по одной из кнопок Edit Mode (Режим редактирования). После щелчка прямоугольный курсор редактирования будет залит однородным цветом. Курсор редактирования следует поместить слева в начале ступеньки 1. Первая команда программы – обычно релейный контакт или элемент - будет помещена в месте расположения курсора. Щелкните по символу Normally Open Contact (Нормально разомкнутый контакт) на панели инструментов программирования. Курсор превратится в окошко ввода элемента, на котором будет нарисован разомкнутый релейный контакт, а также помещены текстовое поле с мерцающим курсором после подсвеченного адреса C0 и зеленые индикаторы подтверждения правильности ввода.



Р

исунок 1.20 – Панель редактирования команды

Если зеленый цвет индикатора правильности ввода меняется на красный, то это значит, что введен неверный адрес, недействительный или неправильный символ. Например, если вы ввели букву «О» вместо цифры «0», то индикатор станет красным и будет красным до тех пор, пока не будет исправлена ошибка. В нашем примере введите X0 вместо C0. Индикатор правильности ввода должен остаться зеленым, означая, что адрес введен правильно. Теперь или щелкните по «галочке» (☑☑), или нажмите клавишу Enter (Ввод).

Элемент будет введен, и курсор передвинется в следующую позицию ввода. Слева от ступеньки 1 появится желтая вертикальная полоса. Эта желтая полоса указывает на то, что вы уже ввели команду или команды, но программа еще не принята (не скомпилирована).

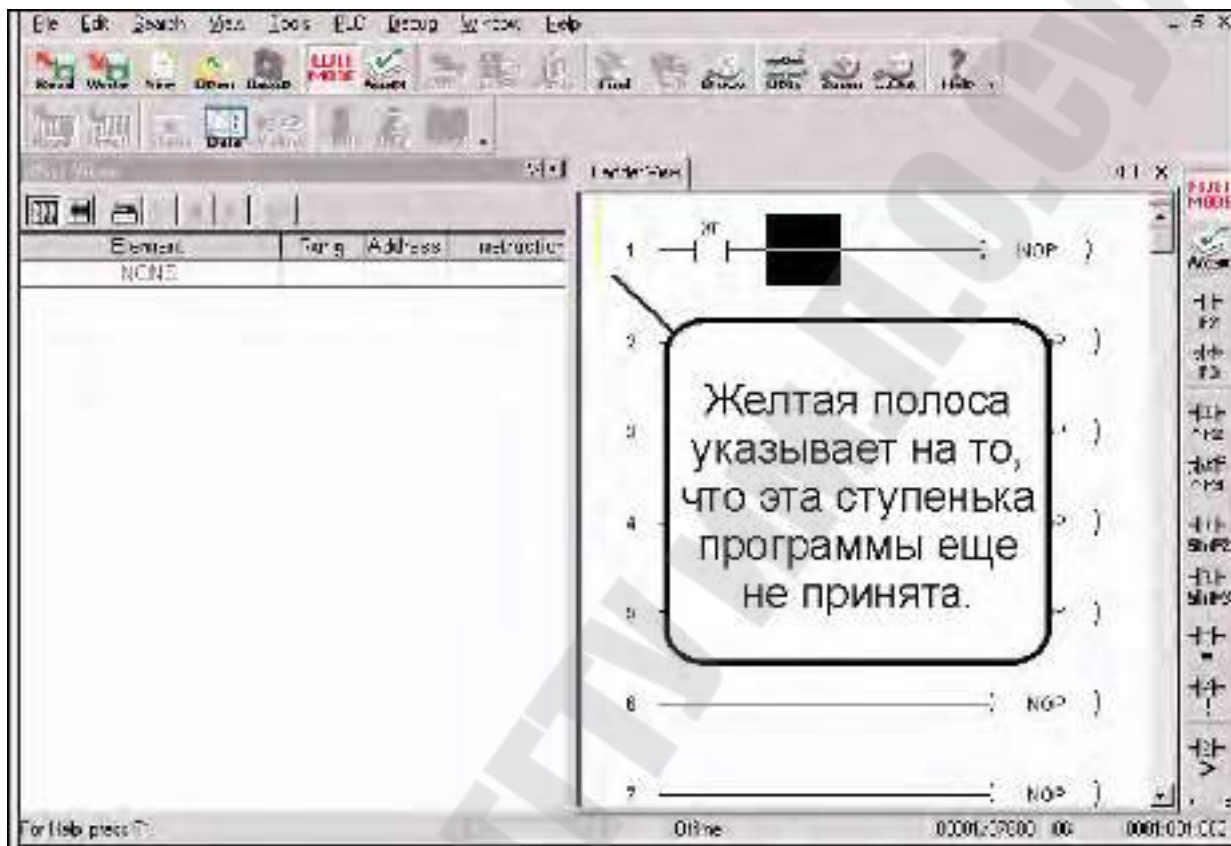


Рисунок 1.21 – Промежуточный вид окна редактирования

Теперь передвиньте курсор в конец ступеньки, поместив его на NOP (No Operation – Нет команды). Щелкните по кнопке Browse Coils (Просмотр обмоток) на панели инструментов программирования. Появится окно Instruction Browser (Просмотр команд), в котором по умолчанию выбрана Standard Coil (Стандартная обмотка). Щелкните по ОК для ввода стандартной обмотки. Имейте в виду, что вы можете выбрать какую-либо другую обмотку.

Окно Instruction Browser (Просмотр команд) сменится окошком ввода элемента. Адрес по умолчанию, C0, будет подсвечен. Введите Y0 и проверьте, что индикатор правильности ввода остался зеленым, означая, что адрес введен правильно. Теперь или щелкните по «галочке» (☐☐), или нажмите клавишу Enter (Ввод) для ввода элемента.

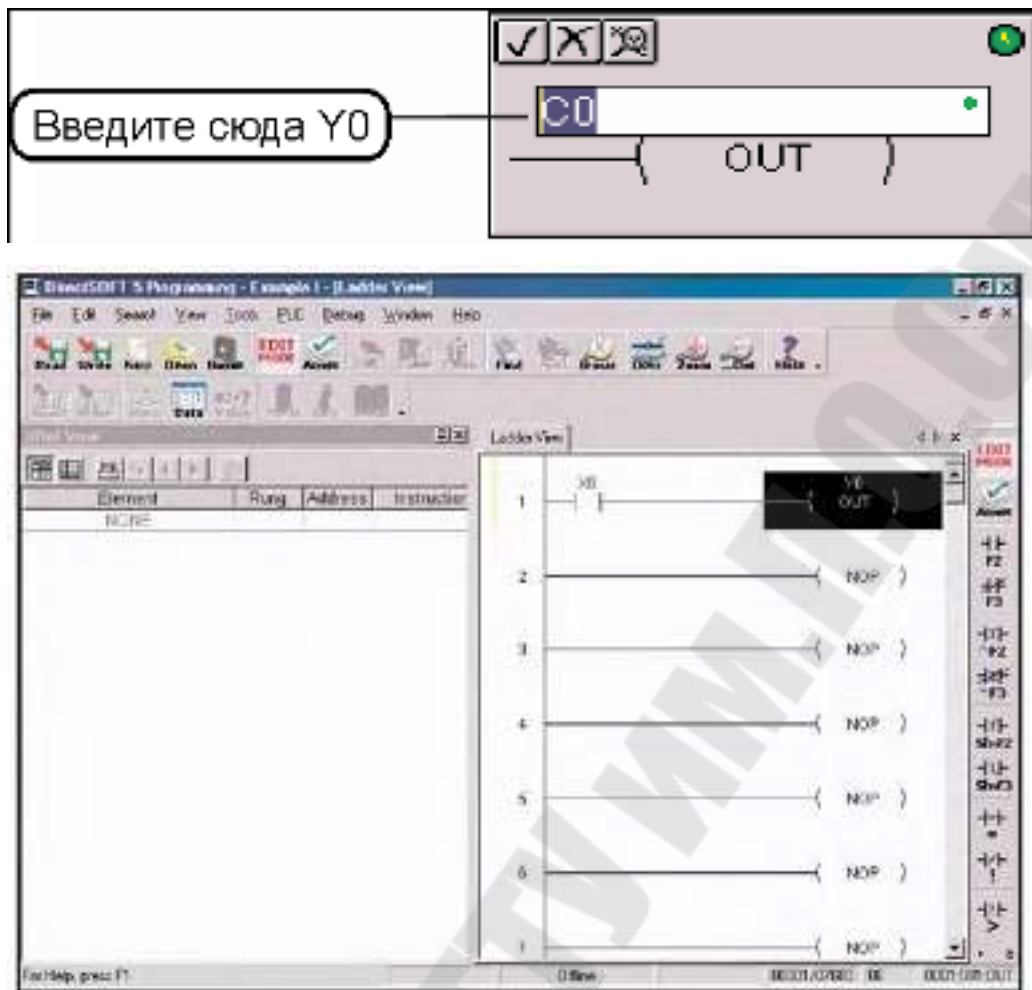


Рисунок 1.22 – Окончательный вид запрограммированного Действия

Ступенька 1 запрограммирована. Для того чтобы эту ступеньку программы можно было загрузить в ПЛК, необходимо добавить еще одну недостающую ступеньку с обмоткой END Coil (Конечная обмотка), которой должны заканчиваться все программы.

Для программирования этой ступеньки поместите курсор на NOP (No Operation – Нет команды) в конце ступеньки 2 и щелкните по кнопке Browse Coils (Просмотр обмоток) на панели инструментов программирования. Появится окно Instruction Browser (Просмотр команд). На этот раз выберите позицию Program Control (Управление программой) в наборе команд Coil Class (Группа обмоток). Затем выберите позицию END (Конец) в наборе команд Coils (Обмотки). Щелкните по ОК, затем Enter.

Для того чтобы загрузить программу в ПЛК, ее сначала нужно принять. Как видно на рисунке наверху, есть две кнопки Ассерпт (Принять). Щелкните по любой из этих кнопок для того, чтобы скомпилировать программу. Если ступеньки программы принимаются компилятором без ошибок, то желтая полоса изменится на зеленую, кнопки Ассерпт станут бледно-серыми и в окне перекрестных ссылок (Cross Reference – XRef) появятся два элемента, которые и были запрограммированы.

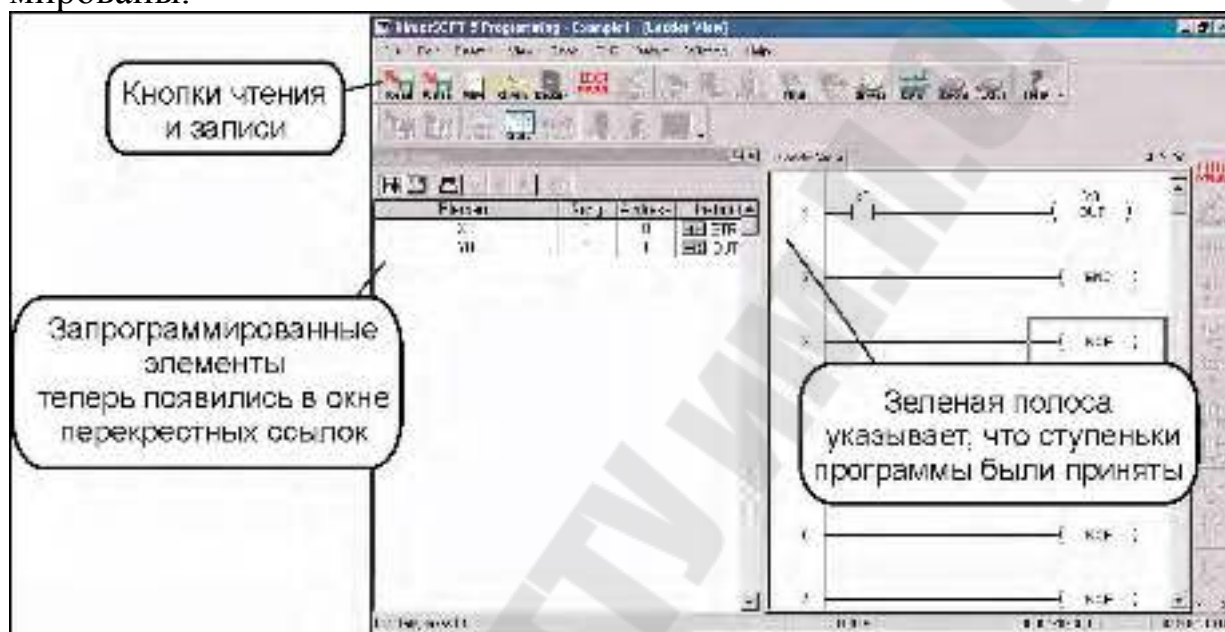


Рисунок 1.23 – Окно записи/чтения готового проекта

Обратите внимание, что две кнопки Read (Прочитать) и Write (Записать), расположенные слева на панели инструментов Offline, теперь включены и не бледно-серые. Программу теперь можно сохранить на диск компьютера. Для записи программы на диск щелкните по кнопке Write (Записать). Совсем не обязательно сохранять программу на диске, если вы хотите загрузить ее в ПЛК, однако, сохранять программу после ее редактирования – это хорошая привычка. Иногда может быть сделана ошибка в программе и нужно вернуть программу в состояние, в котором она была до ошибки. Если ошибка была сделана, и вы хотите восстановить программу, щелкните по кнопке Read (Прочитать). На экране появится ранее сохраненная версия вашей программы.

Если программа сохраняется нажатием кнопки Write (только на диск), то сохраняется только сама лестничная программа. Если же вы

отредактировали программу и включили в нее всю сопутствующую документацию, то необходимо сохранить все, что вы сделали. Это осуществляется выбором в главном меню File > Save Project > to disk. Можно также щелкнуть по Backup (Резервное копирование), чтобы выполнить то же самое действие, при этом будет сохранена резервная копия файла.

Для того чтобы загрузить программу в ПЛК, необходимо создать коммуникационный канал (канал связи).

Соедините кабелем программирования последовательный порт компьютера с последовательным портом ПЛК. Включите ПЛК и убедитесь, что переключатель контроллера RUN/TERM/STOP находится в положении TERM. Щелкните по PLC (ПЛК) в главном меню, затем в выпадающем меню выберите Connect (Соединить), появится диалоговое окно Select Link (Выберите тип канала связи). Поскольку пока еще нет ни одного канала для выбора, щелкните по Add (Добавить). Появится диалоговое окно помощника создания канала связи, в котором пере-числены коммуникационные порты. Выберите порт, который вы будете использовать (обычно COM1), и щелкните по Next (Далее).



Рисунок 1.24 – Последовательность действий по созданию канала связи

В следующем окне представлен список семейств ПЛК (PLC Families). Выберите семейство ПЛК, щелкнув по нужной строчке списка. Если вы не уверены в том, какое семейство выбрать, но знаете, какой коммуникационный протокол будет использовать, выберите «Not Sure» (Не уверен). Если используется ПЛК, совместимый с кон-

троллерами DirectLOGIC, то помощник создания канала связи попытается определить тип ПЛК автоматически. Когда закончите выбор, щелкните по Next (Далее).

На этом шаге нужно выбрать или DirectNET, или K-Sequence. Предположим, что вы выбрали семейство контроллеров DirectLOGIC (только не DL305), то тогда по умолчанию будет подсвечиваться K-Sequence. Протокол обмена данными KSequence позволяет производить операции записи в индивидуальные дискретные каналы ввода/вывода и в управляющие реле. Протокол DirectNET эти операции выполнить не может. (См. Приложение А, в котором приведен перечень протоколов обмена данными для контроллеров DirectLOGIC и совместимых с ними ПЛК. Если вашему ПЛК задан узловой адрес, отличный от 1, то введите этот адрес сейчас. Щелкните по Next (Далее). Если помощник в создании коммуникационного канала установил связь с ПЛК, то в следующем окне вам будет предложено ввести уникальное имя канала связи и, если нужно, описание этого канала. Описание канала связи может содержать до 32 символов. Введите имя канала связи и его описание, затем щелкните по Finish (Готово), появится диалоговое окно Select Link (Выберите канал связи), в списке коммуникационных каналов которого будет содержаться и имя созданного канала связи.



Рисунок 1.25 – Окно идентификации канала связи

Поскольку только что созданный канал связи единственный имеющийся канал в списке окна Select Link (Выберите канал связи), щелкните по Select (Выбрать) для запуска процесса соединения с ПЛК.

DirectSOFT 5 автоматически сравнит открытую в настоящий момент программу с программой, которая сохранена в ПЛК. Появится диалоговое окно Online/Offline Differences (Отличия программ в компьютере и в ПЛК), в котором спрашивается, какую копию лестничной программы следует вывести на экран компьютера: копию из ПЛК или копию из компьютера. Поскольку в настоящий момент мы имеем дело с новой программой, выберите кнопку Use Disk (Воспользоваться программой на диске компьютера). Кнопку Use Disk следует использовать всякий раз, когда на компьютере, который соединен с ПЛК каналом связи, вы сделали изменения в программе и собираетесь загрузить ее в ПЛК. Если нажать кнопку Details (Подробности), то бок о бок друг с другом появятся отличающиеся ступеньки программ в ПЛК и в компьютере, например, такие, как показанные ниже в окне Compare Programs (Сравнение программ). Выбор нужной программы можно также сделать в этом диалоговом окне.

После нажатия на кнопку Use Disk (Воспользоваться программой на диске компьютера) окно программирования будет выглядеть несколько иначе. Обратите внимание на то, что рисунки (иконки) на кнопках панели инструментов Online уже не бледно-серые. Индикаторы внизу окна программирования подсказывают вам, что ПЛК исправен, компьютер связан с ПЛК каналом передачи данных и ПЛК находится в режиме программирования (Program Mode). На данном этапе программа ещё не записана в ПЛК. Также обратите внимание, что две самые левые кнопки на панели инструментов Online (ReadP (Прочитать программу) и WriteP (Записать программу) подсвечены. Для записи программы в ПЛК выберите кнопку WriteP. Появится всплывающая индикаторная панель, которая позволяет видеть, что программа записывается в ПЛК.

После того как программа будет записана в ПЛК, всё что останется сделать это перевести ПЛК в режим выполнения программы (RUN Mode). Щелкните по кнопке Mode (Режим) на панели инструментов Online. Появится диалоговое окно PLC Modes (Режимы ПЛК). Щелкните по RUN, затем ОК и ПЛК перейдет в режим выполнения программы (RUN Mode). Обратите внимание на зелёный индикатор под

окном программы. Он показывает, что ПЛК сейчас находится в режиме выполнения программы.

1.5. Основы программирования на RLL

1.5.1. Программа на RLL

Программы на RLL Plus не столько пишутся, сколько рисуются в виде схем Релейно-Лестничной Логике (Relay Ladder Logic -RLL), которые ближе конструктору-электрику, нежели программисту. Программирование можно отождествить с проектированием релейно-контактной схемы, да и терминология в описании команд языка RLL Plus используется соответствующая реле, контакт, шина питания. Разработчики RLL Plus и его графического представления предполагали, что этим языком легко смогут пользоваться конструкторы и наладчики, чтобы легко и быстро создавать системы управления, пользуясь привычными понятиями и на основе собственных схем. Опыт программирования на RLL Plus позволяет утверждать, что язык RLL Plus при умелом пользовании и знании специфики работы процессора достаточно гибок и позволяет реализовывать красивые решения достаточно сложных логических задач. И для успешного освоения этого языка совсем не обязательно проникаться «аппаратной» терминологией любая программа на RLL Plus при внимательном рассмотрении удивительно напоминает хорошо знакомые всем профессионалам Паскаль и Си! Рассмотрим фрагмент программы на PLL Plus:



Этот фрагмент называется ветвью программы или ступенью. Каждая ступень состоит из условной (проверка состояния бита C20) и исполняемой (установка бита Y2 и записи числа 7 в ячейку V1500) частей. На Паскале это выглядело бы так:

```
IF C20 = True THEN  
BEGIN  
Y2 := 1 (* или Y2=True *);
```



```
V1500 := 7;  
END;
```

Таким образом, вся программа на RLL Plus представляет собой набор операторов IF THEN с условиями разной сложности. И никаких там контактов с обмотками и цепей питания! Правда, если в Паскале условия могут быть сколь угодно сложные, лишь бы программист в них не запутался, то в RLL Plus существует ограничение: процессор DL имеет 8-уровневый логический стек, в который записывается результат проверки условия (0 или 1), а при выполнении логических операций OR, AND, XOR он продвигается на один уровень выше. На практике такого набора условий, который вызвал бы переполнение логического стека, еще встречать не приходилось. Довольно часто встречаются ситуации, когда нужна только исполнительная часть ветви. Условие в этом случае тоже задавать необходимо, но это должно быть условие, которое выполняется всегда. В RLL Plus для этого предназначено реле SP1, которое всегда установлено.

Язык RLL Plus позволяет оперировать как отдельными битами, так и словами, а также двойными словами, причем все операции со словом (двойным словом) можно выполнять только в аккумуляторе. Ячейку памяти можно лишь инкрементировать/декрементировать, но зато она может участвовать в условной части.

1.5.2. Основные операции и команды

Логические команды

Оператор END

Любая программа в DL должна заканчиваться операцией END. Ни одна команда, записанная после END не будет обрабатываться, если только на нее нет передачи управления из основной программы, например, если она не принадлежит какой-либо подпрограмме. Графически оператор END представляется обмоткой реле, прямо соединенной с шиной питания:



Простая ступень, нормально разомкнутый контакт

Обычно условную часть ступени начинает контакт реле. Простейшая ступень нашего примера состоит из нормально разомкнутого контакта Реле X0, подающего питание на обмотку Реле Y0. Условная логическая цепь состоит из единственной команды STR. Состояние входного Реле X0 передается выходному Y0 командой OUT:



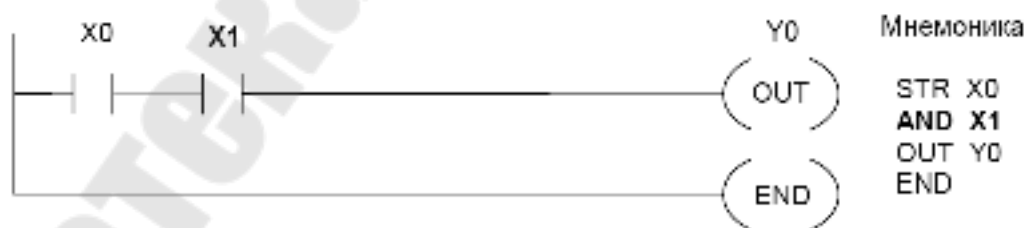
Простая ступень, нормально замкнутый контакт

Нормально замкнутый контакт реле также часто начинает ступень. Если нужно добиться срабатывания выходного Реле Y0, при отпуске входного X0, то условная часть простой ступени должна состоять из команды STRN, а исполнительная – из команды OUT:



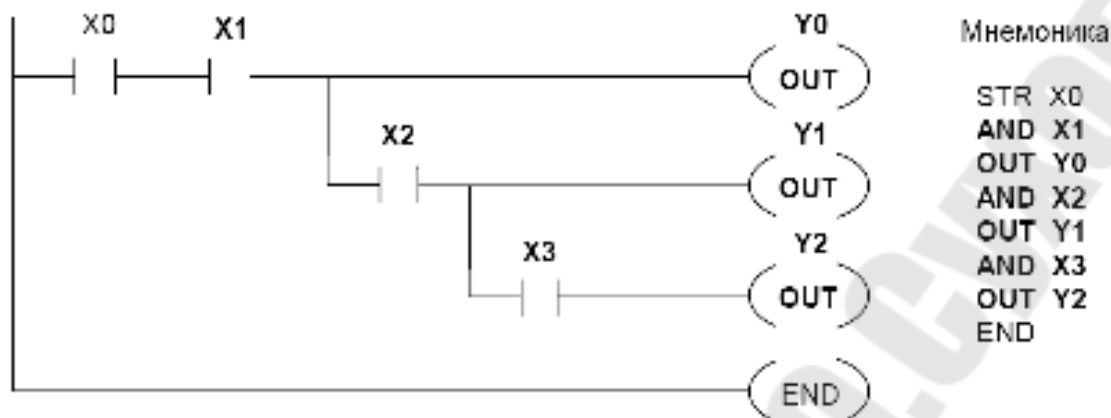
Последовательные контакты

Команда AND последовательно соединяет два (или более) контакта. Например, Реле Y0 «под током», пока «под током» оба Реле - X0 и X1:



Выходы с промежуточных точек условной цепи

Иногда необходимо «подать питание» на какие-либо исполнительные реле с промежуточных точек условной логической цепи контактов. Следующий пример показывает, как это можно сделать, используя команду AND. Реле Y0 срабатывает, как в предыдущем примере; Реле Y1 - когда вдобавок работает Реле X2, а Реле Y2 - когда срабатывает еще Реле X3.



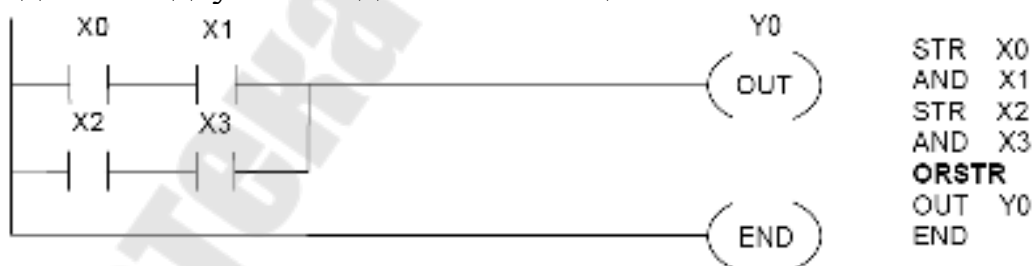
Параллельные элементы

Команда OR соединяет контакты условной части ступени параллельно. В следующем примере Реле Y0 сработает, если "под током" хотя бы одно из Реле - X0, X1.



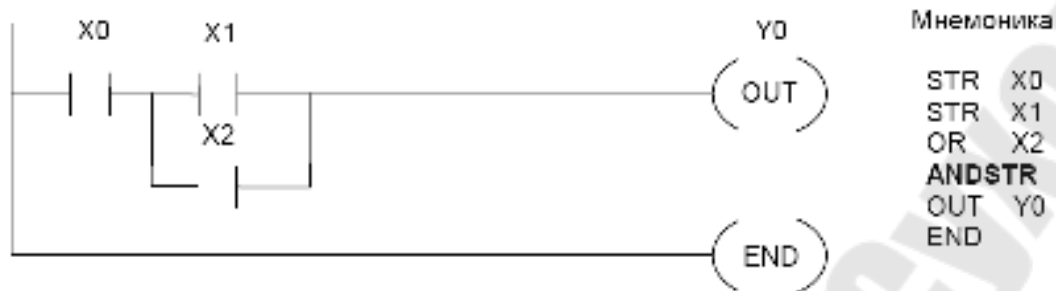
Параллельное соединение последовательных цепей

Команда ORSTR позволяет параллельно соединять несколько последовательных цепей. На следующем примере показано параллельное соединение двух последовательных цепей: X0-X1 и X2-X3.



Последовательное соединение параллельных цепей

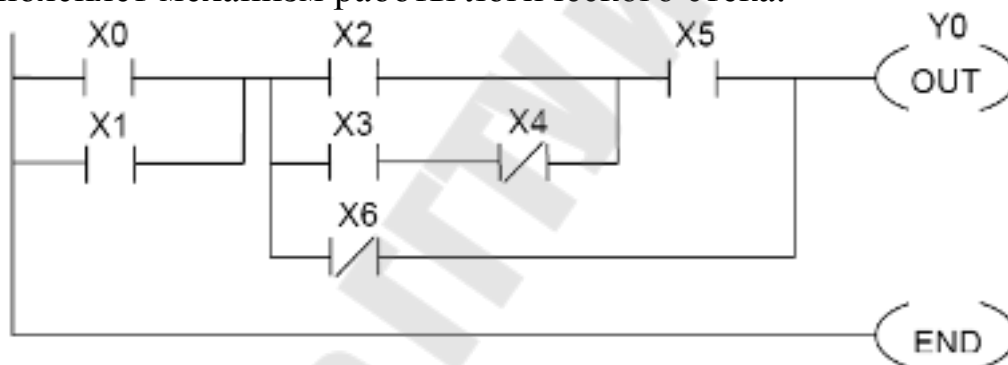
Если необходимо последовательно соединить несколько параллельных цепей, то эту операцию позволяет сделать команда ANDSTR. Ниже показано последовательное соединение контакта реле X0 и параллельно соединенных контактов реле X1 и X2.



Сложные цепи

RLL допускает достаточно сложные комбинации последовательных и параллельных соединений в релейно-контактных цепях условной части ступени, например:

В DL440 существует ограничение на число элементов, включаемых в условную часть логической ступени. Это ограничение связано с тем, что DL440 использует 8-уровневый логический стек в формировании условия для исполнительской части ступени. Следующий параграф поясняет механизм работы логического стека.



Логический стек - это 8-битовый регистр сдвига, прямо недоступный пользователю, бит n ($n=1,2,3...8$) которого образует n -ный уровень логического стека. Бит 1 этого регистра является вершиной логического стека, а бит 8 - дном. Все условные логические команды служат для изменения состояния логического стека.

1	2	3	4
STR r	OR r	AND r	ORSTR
STRN r	ORN r	ANDN r	ANDSTR

Как видно из таблицы, первые три пары логических команд имеют в качестве второго операнда состояние Реле r , которое участвует в операции прямо или инверсно (в последнем случае обозначение команды имеет суффикс N).

STR STRN

Первая пара логических команд STR/STRN г загружает на вершину логического стека прямое/инверсное состояние Реле г. При этом прежнее содержимое вершины стека (1-й уровень) "вытесняется" на 2-й уровень, а прежнее содержимое этого уровня - на 3-й, и т.д. "Вытесненное" содержимое 8-го уровня ("дно" стека) теряется. Другими словами, команды STR г и STRN г «проталкивают» в логический стек прямое или инверсное состояние Реле г.

OR ORN AND ANDN

Логические команды OR/ORN г или AND/ANDN г производят логическое сложение (OR) или логическое умножение (AND) содержимого вершины логического стека с прямым/ инверсным состоянием Реле г. Результат помещается на вершину логического стека, содержимое остальных его уровней не изменяется.

ORSTR ANDSTR

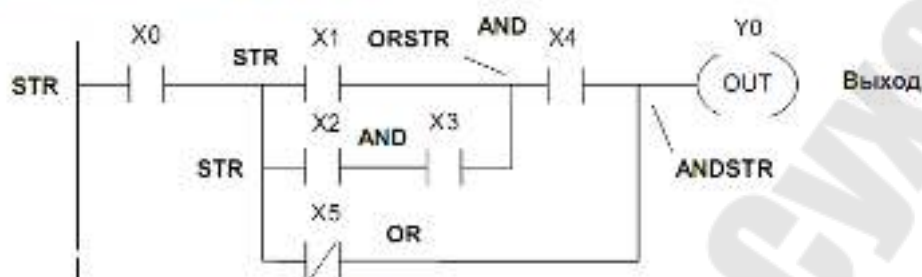
В логических командах ORSTR/ANDSTR операнды не указываются. Эти команды сначала производят «выталкивание» стека - перемещение содержимого n-ного ($n=2,3...8$) уровня логического стека на уровень n-1 («дно» стека обнуляется). «Вытолкнутое» содержимое вершины логического стека логически складывается/умножается с новым ее содержимым, поступившим со 2-го уровня, и результат остается на вершине стека.

Работу логического стека поясняет следующий пример.

Из примера видно, каким образом продвигаются данные в логическом стеке. Хотя логическая цепь контактов реле X была достаточно сложной, «проталкивание» в логический стек «достало» только до 3-го уровня. Понятно, что переполнение этого стека возможно только при очень сложной логической цепи, насыщенной последовательно-параллельными ветвями.

Работу логического стека поясняет следующий пример:

STR X0
 STR X1
 STR X2
 AND X3
 ORSTR
 AND X4
 ORN X5
 ANDSTR
 OUT Y0



Команда	Уровень	Содержимое уровня после выполнения команды
STR X0	1 (вершина)	X0
	...	-
	8 (дно)	-
STR X1	1 (вершина)	X1
	2	X0
	...	-
	8 (дно)	-
STR X2	1 (вершина)	X2
	2	X1
	3	X0
	...	-
	8 (дно)	-
AND X3	1 (вершина)	X2 AND X3
	2	X1
	3	X0
	...	-
	8 (дно)	-
ORSTR	1 (вершина)	(X2 AND X3) OR X1
	2	X0
	...	-
	8 (дно)	-
AND X4	1 (вершина)	((X2 AND X3) OR X1) AND X4
	2	X0
	...	-
	8 (дно)	-
ORN X5	1 (вершина)	(((X2 AND X3) OR X1) AND X4) OR (NOT X5)
	2	X0
	...	-
	8 (дно)	-
ANDSTR	1 (вершина)	((((X2 AND X3) OR X1) AND X4) OR (NOT X5)) AND X0
	...	-
	8 (дно)	-
OUT Y0		Содержимое вершины логического стека копируется в Реле Y0.

Команды изменения состояний реле

Прежде чем рассматривать команды, изменяющие состояния реле, подчеркнем, что их выполнение зависит от состояния, которое к моменту выполнения имеет логический стек, в частности его вершина, а также от некоторых других условий, например от активности последнего встретившегося мастер-реле или стадии, в которой находится команда. Вот список команд управления реле в DL:

OUT r	OROUT r	PD r	SET $r [r']$	RST $r [r']$	PAUSE $y [y']$
$\neg(\text{OUT } r)$	$\neg(\text{OROUT } r)$	$\neg(\text{PD } r)$	$\neg(\text{SET } r [r'])$	$\neg(\text{RST } r [r'])$	$\neg(\text{PAUSE } y [y'])$

OUT

Команда OUT r копирует состояние вершины логического стека в Реле r . В неактивной стадии или мастер-линии, которые эквивалентны обнулению вершины логического стека, команда OUT r приводит к сбросу Реле r .

OROUT

Команда OROUT r специально спроектирована для повторений в нескольких ступенях программы и позволяет сделать состояние Реле r зависимым от нескольких ступеней так, как-будто их условная логика соединена в параллельные ветви. Команда производит логическое сложение состояний Реле r и вершины логического стека, а результат помещает в Реле r .

PD

Команда PD r установит Реле r , если состояние вершины логического стека в момент выполнения инструкции в текущем Рабочем цикле DL440 равно «1», а в предыдущем было равно «0». В начале следующего цикла DL440 автоматически сбросит Реле r .

SET RST

Команды SET $r [r']$ и RST $r [r']$ устанавливают реле r [или цепочку реле от r до r'] в состояние «1» или «0» соответственно, если состояние вершины логического стека в момент выполнения команды равно «1»; иначе состояние указанных реле не меняется.

PAUSE

Команда PAUSE $y [y']$ останавливает вывод из реле y [или из цепочки реле от y до y'] в

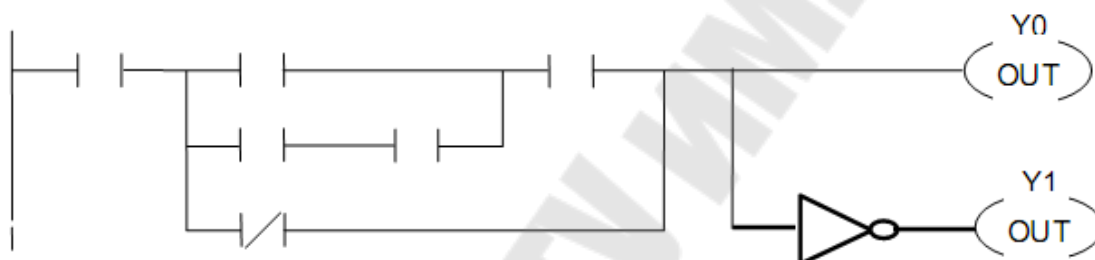
модули вывода, пока состояние вершины логического стека в момент выполнения команды равно «1», т.е. пока замкнута условная логическая цепь контактов. Состояния

соответствующих выходов модулей вывода все это время находятся в отключенном состоянии.

При PAUSE возможны любые действия над реле у, результат которых тем не менее не изменит пассивного состояния соответствующего выхода в модуле вывода.

Команда инверсии NOT

В некоторых операциях требуется изменить состояние вершины логического стека на противоположное, например, чтобы включить одно реле (Y0) и выключить другое (Y1) при замыкании какой-либо сложной условной логической цепи:




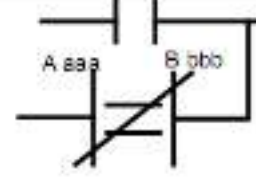
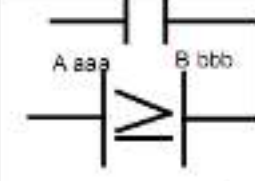
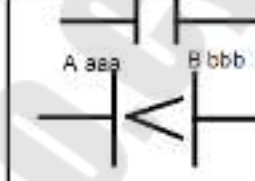
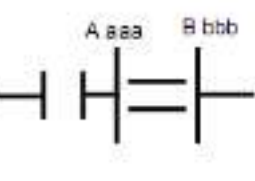


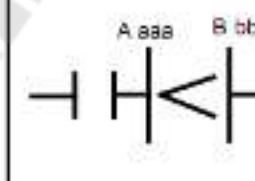
Логика сравнений

DL440 позволяет включить в логическую условную цепь результат сравнения двух 16-разрядных целых чисел в виде контакта реле. Команды сравнения охватывают все возможные соотношения между двумя числами:

равно	не равно	Больше или равно	меньше
A aaa B bbb — = —	A aaa B bbb — / —	A aaa B bbb — ≥ —	A aaa B bbb — < —
STRE Aaaa Bbbb	STRNE Aaaa Bbbb	STR Aaaa Bbbb	STRN Aaaa Bbbb

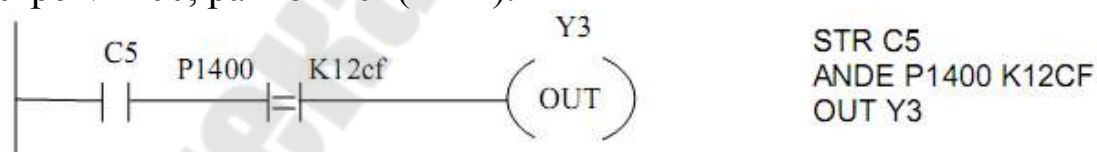
В таблице показаны видеографические изображения команд сравнения, которые использует DirectSOFT, а также их мнемонические обозначения. В качестве операндов сравнения (Aaaa и Bbbb) могут выступать 16-разрядные целые числа из всех V-регистров, например, текущие значения Таймеров и Счетчиков. При этом можно использовать либо прямые ссылки (универсальные и спе-

циальные) на регистры, либо указатели (Prrr) на них. В качестве операнда Bbbb может выступать еще шестнадцатиричная (HEX) константа (ссылка - Kkkk). Приведенная таблица не исчерпывает всех команд сравнения и должна быть дополнена командами, позволяющими соединять указанные контакты параллельно или последовательно:

			
ORE Aaaa Bbbb	ORNE Aaaa Bbbb	OR Aaaa Bbbb	ORN Aaaa Bbbb
			
ANDE Aaaa Bbbb	ANDNE Aaaa Bbbb	AND Aaaa Bbbb	ANDN Aaaa Bbbb

Заметим, что первые четыре команды сравнения (первая таблица) загружают результат сравнения на вершину логического стека, «проталкивая» ее содержимое на уровень 2 и т.д. Последние 8 команд сравнения помещают результат сравнения на вершину логического стека, не изменяя содержимого остальных уровней.

В примере, изображенном ниже, реле Y3 устанавливается, если реле C5 «под током» и если число из регистра, HEX-адрес которого в регистре V1400, равно 12cf (HEX).



Заметим, кстати, что двоичное (HEX) число из регистра V1400, будучи записанным в восьмеричной системе счисления, должно давать допустимый номер V-регистра, иначе устанавливается специальное реле SP71.

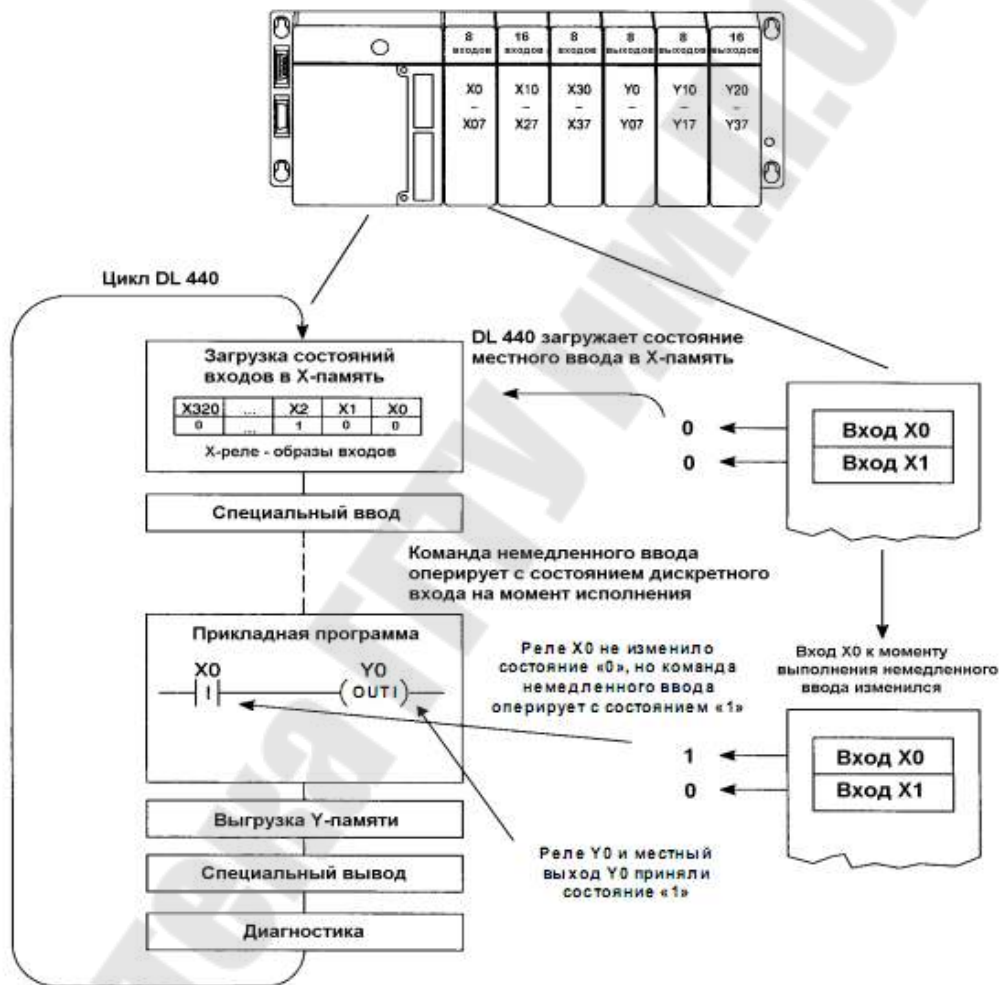
Немедленный ввод/вывод

Цикл обновления Местного Ввода/Вывода DL440 обычно составляет миллисекунды. Фактически это длительность Рабочего Цикла, в начале которого система проводит Местный Ввод, а после прикладной программы - Местный Вывод. Длительность Рабо-

чего Цикла непостоянна и зависит от многих факторов. Если какие-либо входы/выходы требуют стробирования через строго определенный интервал времени,

то обычно применяют прерывание по таймеру (INT O17) и команды немедленного ввода и немедленного вывода, которые имеются в наборе DL440.

Команды немедленного ввода/вывода осуществляют Местный Ввод и Местный Вывод непосредственно в момент исполнения, но выполняются медленнее, чем аналогичные системные операции DL440.



ТАЙМЕРЫ

Таймеры задают длительности событий и являются счетчиками внутрисистемных тактовых импульсов. В DL440 Таймеры бывают двух типов - простые и накапливающие. Оба типа Таймеров могут работать в медленном или быстром вариантах. Медленные Тай-

меры работают от тактовых импульсов частотой 10Гц, быстрые - от тактовых импульсов частотой 100Гц:

Простые Таймеры		Накапливающие Таймеры	
Медленный вариант	Быстрый вариант	Медленный вариант	Быстрый вариант
TMR	TMRF	TMRA	TMRAF

DL440 поддерживает до 256 простых Таймеров.

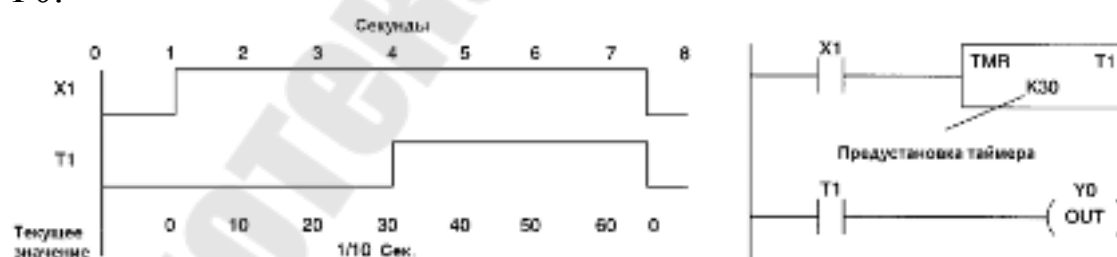
Простой Таймер с номером $nnn = 000 \div 3778$ - это объект, с которым связаны три параметра:

1) текущее значение $TAnnn$ от 0000 до 9999 в BCD-формате (в памяти ЦП эти данные располагаются в 256 регистрах $V0 \div V377 = TA0 \div TA377$);

2) предустановка от 0000 до 9999 в BCD-формате, заданная константой, переменной V-памяти или указателем на нее;

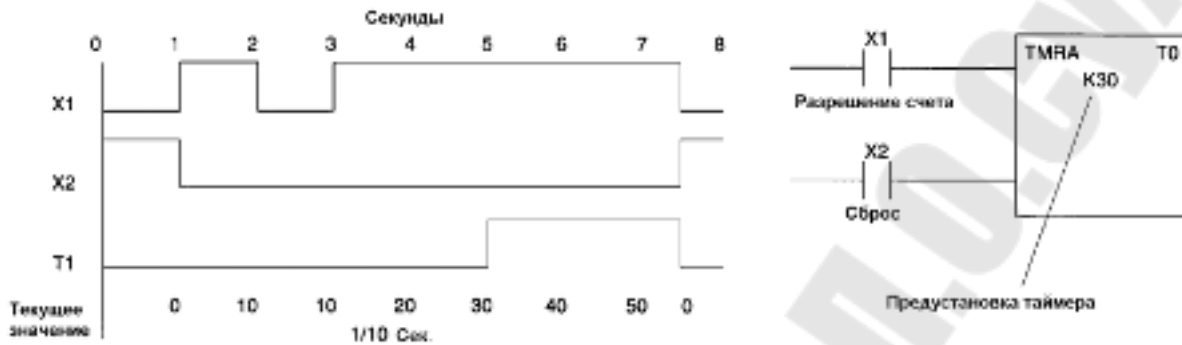
3) реле $Tnnn$, устанавливающееся в «1», когда текущее значение $TAnnn$ достигнет предустановки (в V-памяти эти реле занимают 16 регистров $V41100 \div V41117 = VT0 \div VT360$).

DirectSOFT изображает простой Таймер в виде исполнительного блока с одной входной линией. Пока эта линия получает питание (выполняется условие вхождения в блок) к текущему значению Таймера добавляется «1» каждые 100мс (TMR) или 10мс (TMRF). Когда текущее значение Таймера достигнет величины 9999, он останавливает счет. Если условие вхождения в блок не выполняется, текущее значение ТА и реле Т обнуляются. Обнуление Таймера можно провести также другим способом, обнуляя реле Т или регистр ТА. В приведенном примере Таймер 1 отсчитывает 3 секунды после установки реле X1 и устанавливает реле T1, что вызывает установку реле Y0:



Накапливающий Таймер $Tnnn$ подобен простому, но имеет два входа и под текущее значение занимает 2 регистра - $TAnnn$ и $TA(nnn+1)$, емкостью до 99999999 в BCD-формате. Предустановка также задается 8-разрядным десятичным числом, являющемся константой или занимающем 2 V-регистра. В качестве флага Таймера nnn выступает реле $Tnnn$, а реле $T(nnn+1)$ не используется. Вход разрешения счета (в DirectSOFT верхний), если не получает питания,

останавливает Таймер, не обнуляя его. Подача питания на вход сброса обнуляет текущее значение Таймера и реле Т. Ниже показан пример, где реле X1 разрешает накопление текущего значения Таймера T0 (в регистровой паре TA0, TA1), а реле X2 обнуляет его и реле T0:



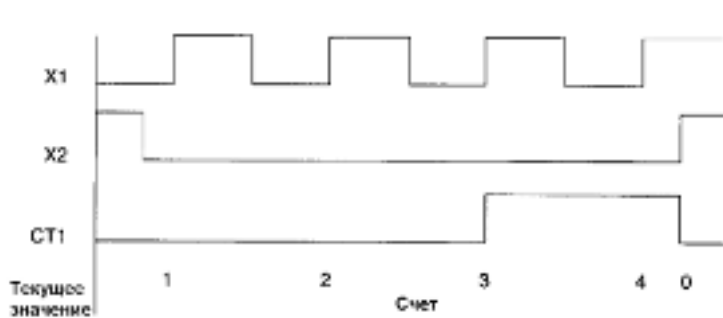
СЧЕТЧИКИ

Если Таймеры подсчитывают внутрисистемные тактовые импульсы, то Счетчики - моменты замыкания определенных условных релейно-контактной цепей. В DL440 Счетчики бывают трех типов - простые (CNT), реверсивные (UDC) и Счетчики стадий (SGCNT).

DL440 организует до 128 простых Счетчиков. Счетчик с порядковым номером $nnn = 000 \div 1778$ - это объект, с которым так же, как с Таймером, связаны в памяти ЦП три величины:

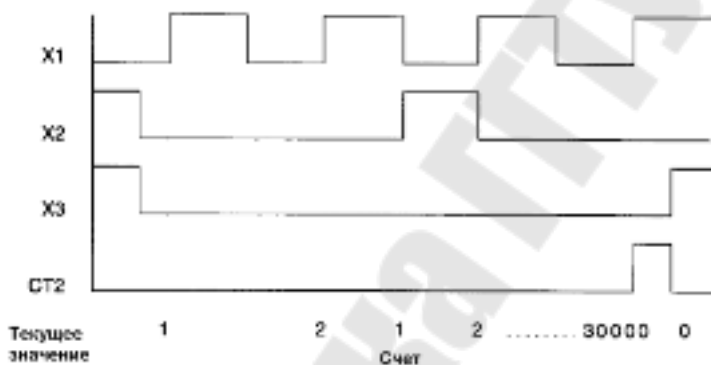
- 1) текущее значение $CTAnnn$ от 0000 до 9999 в BCD-формате (в памяти ЦП эти данные занимают 128 регистров $V1000 \div V1177 = STA0 \div STA177$);
- 2) предустановка от 0000 до 9999 в BCD-формате, заданная константой, переменной V-памяти или указателем на нее;
- 3) реле $CTnnn$, активизирующееся, когда текущее значение достигнет предустановки (в V-памяти эти реле размещаются в восьми регистрах $V41140 \div V41147 = VCT0 \div VCT360$)

DirectSOFT изображает простой Счетчик исполнительным блоком с двумя входными линиями. В момент замыкания условной цепи контактов верхней линии простой Счетчик увеличивает текущее значение на «1». Как и Таймер, простой счетчик останавливает счет на числе 9999 (BCD). Подача питания на нижнюю линию обнуляет текущее значение $CTAnnn$ и реле $CTnnn$:



В показанном примере Счетчик CNT CT1 подсчитывает срабатывания реле X1. После третьего замыкания контактов реле X1 устанавливается реле CT1. Установка реле X2 сбрасывает текущее значение СТА1 и реле CT1.

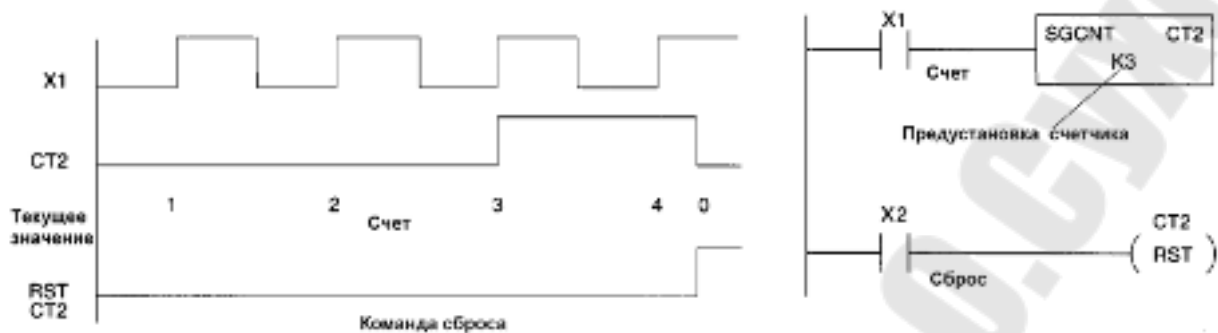
Реверсивный счетчик имеет три входные линии управления счетом. Подача питания на верхнюю линию увеличивает на «1», на среднюю линию - уменьшает на «1», а на нижнюю линию обнуляет текущее значение СТА_{nnn}. Текущее значение реверсивного Счетчика (UDC) СТ_{nnn} занимает регистровую пару СТА_{nnn}-СТА_(nnn+1). При счете на увеличение достижение максимального значения (99999999 в BCD-формате) останавливает счет. То же при достижении нуля в счете на уменьшение.



В примере показан Счетчик UDC CT2, подсчитывающий разницу в количестве срабатываний реле X1 и реле X2. Когда она (разница) достигает 30000, устанавливается реле CT2. Срабатывание реле X3 сбрасывает текущее значение СТА2-СТА3 и реле CT2.

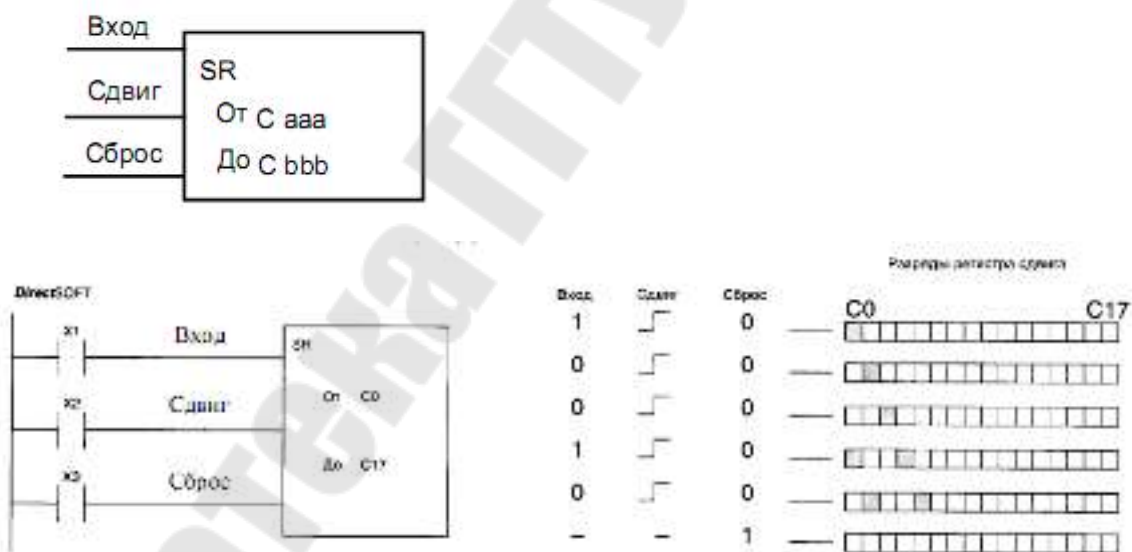
Счетчик стадий SGCNT полностью аналогичен простому CNT, но не имеет входа сброса. Счетчик стадий СТ_{nnn} можно сбросить либо через реле СТ_{nnn}, либо обнулением регистра СТА_{nnn}. Этот тип Счетчиков будет особенно полезен при программировании на RLL-PLUS.

Ниже показана работа Счетчика стадий SGCNT CT2 в варианте, полностью эквивалентном примеру с простым Счетчиком CT1, который был показан на предыдущей странице.



РЕГИСТРЫ СДВИГА

В тех случаях, когда необходимо стробировать последовательные состояния определенной логической цепи в моменты, когда замыкается другая логическая цепь, DL440 позволяет организовать регистр сдвига в С-области памяти ЦП. В сдвиге может участвовать цепочка из любого количества (кратного 8) С-реле, имеющих в ЦП, а сам сдвиг может быть организован как в сторону увеличения порядковых номеров С-реле, так и в сторону уменьшения.



В приведенном примере при каждом срабатывании реле X2 (переход от состояния «0» к состоянию «1») инструкция SR произведет следующие действия: состояние реле C16 запишется в реле C17 (прежнее состояние реле C17 теряется), состояние реле C15 запишется в реле C16 и т.д. Состояние реле C0 запишется в реле C0, а состояние реле X1 запишется в реле C0. Установленное в со-

стояние «1» реле X3 будет удерживать в состоянии «0» все реле цепочки от C0 до C17.

АККУМУЛЯТОР

Аккумулятор ЦП DL - 32-битовый регистр, который используется при обработке чисел и кодов. При этом используются два формата представления чисел:

- двоичный код на все сочетания (HEX-формат);
- упакованный двоично-десятичный (BCD-формат).

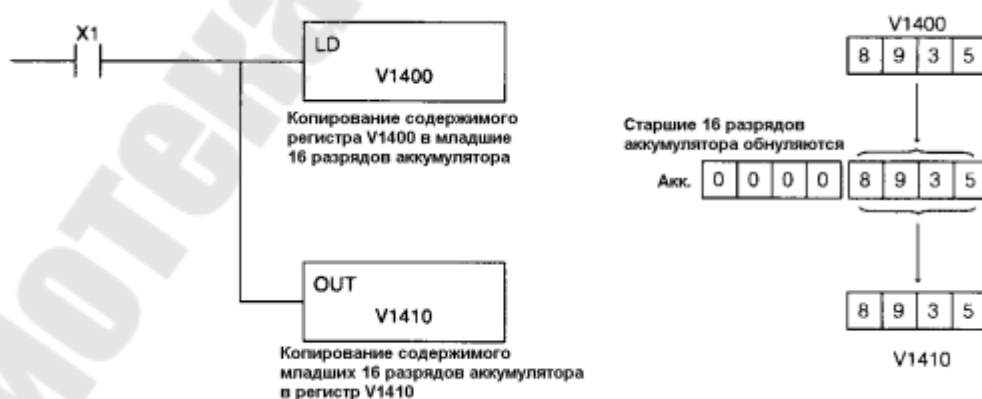
Программы DL в основном работают с 16-разрядными данными. Этому отвечает структура V-памяти ЦП, которая разбита на 16-разрядные регистры.

Но DL может работать также и с 32-разрядными данными, используя в памяти ЦП регистровые пары – два смежных V-регистра, адресуемые по регистру с меньшим номером. При 16-разрядных операциях представление данных в BCD-формате ограничивается 4-разрядными десятичными числами, а в 32-разрядных операциях - 8-разрядными.

Загрузка данных в аккумулятор и выгрузка из него

Пожалуй самыми распространенными командами в любой программе являются команды загрузки аккумулятора числами из V-памяти ЦП (команда LD) и команды выгрузки чисел из аккумулятора в V-память (OUT).

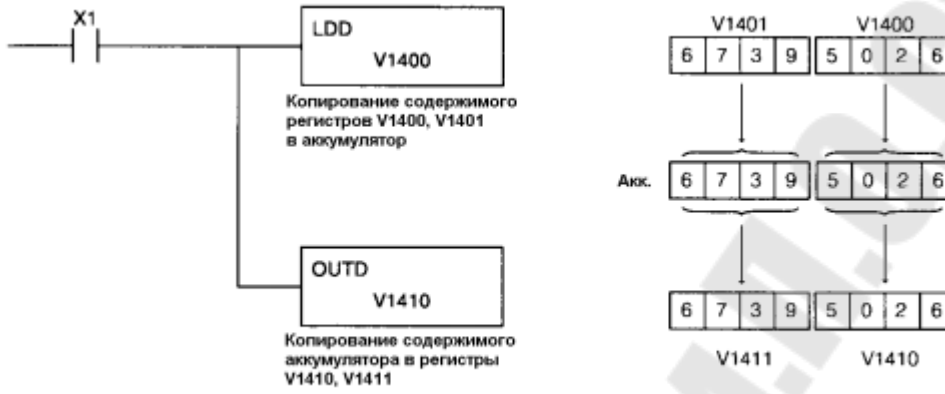
Нижеследующий пример показывает, как число из регистра V1400 пересылается через аккумулятор в регистр V1410 (разумеется, содержимое регистра V1400 при этом не изменяется):



Аккумулятор имеет возможность обмениваться данными с регистровыми парами, т.к. емкость аккумулятора в 2 раза больше ем-

кости V-регистра. Для этого в DL440 имеются команды загрузки и выгрузки 32-битовых чисел (LDD и OUTD).

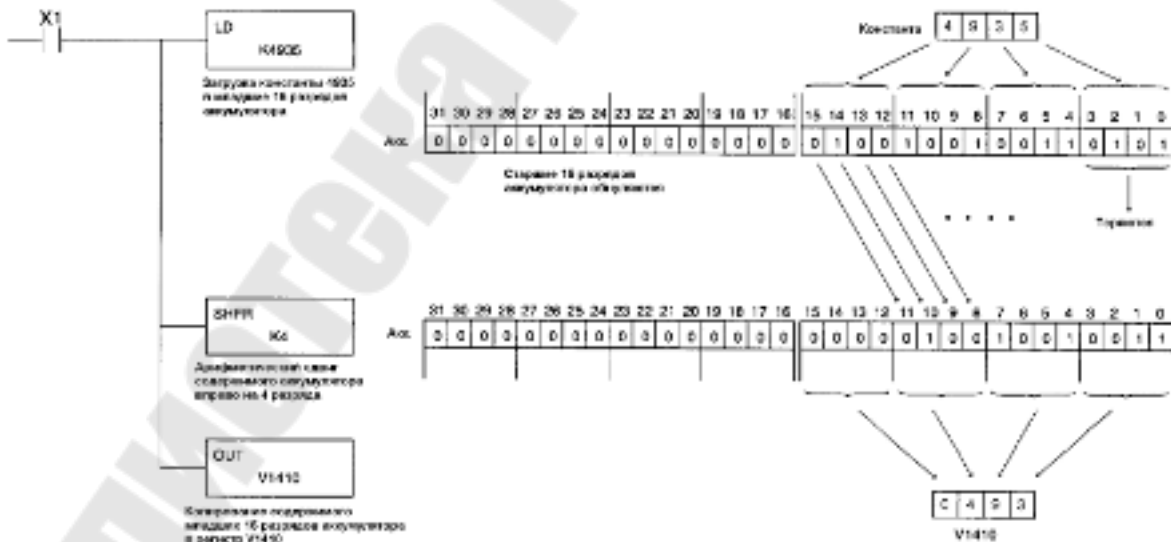
Например, если необходимо скопировать данные, из регистров V1400, V1401 в регистры V1410, V1411 соответственно, то наиболее эффективный путь следующий:



Обработка данных в аккумуляторе

Команды, которые обрабатывают данные, используют содержимое аккумулятора, помещая в него результат обработки (исходное содержимое аккумулятора теряется).

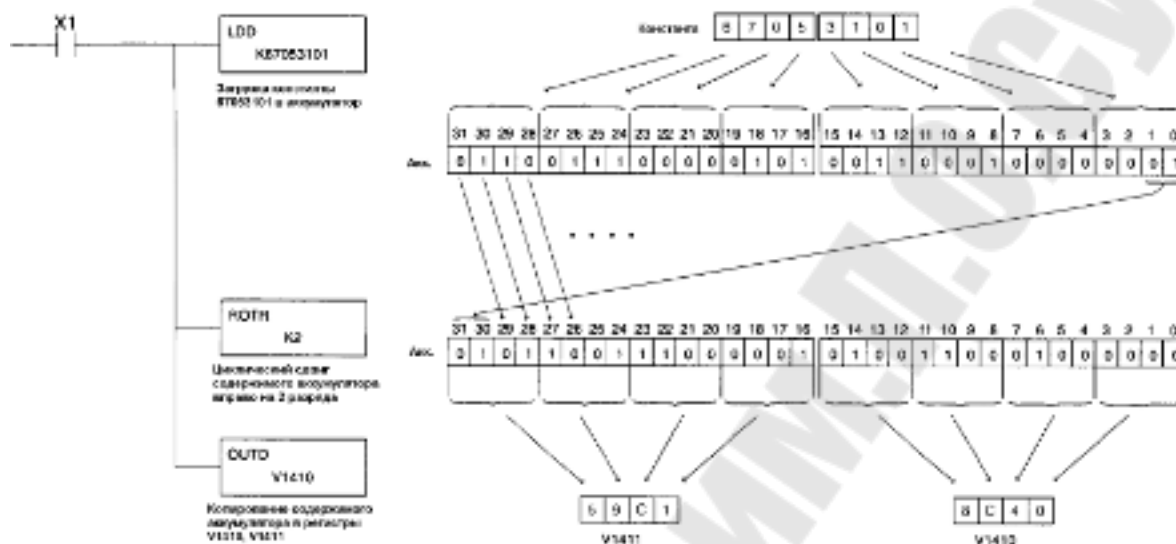
В нижеследующем примере BCD-константа 4935, загруженная в аккумулятор, сдвигается вправо на 4 бита (на один десятичный разряд) арифметически (с потерей младшего десятичного разряда исходного числа), и результат выводится в регистр V1410.



Иногда требуется обработка 32-разрядных данных. В этом случае используется емкость всего аккумулятора, два последовательных V-

регистра памяти (регистровая пара) или 32-разрядные константы (8-значные двоично-десятичные константы).

Например, BCD-константа 67053101 сдвигается в аккумуляторе циклически вправо на два бита и результат выводится в регистровую пару V1410, V1411.



Стек аккумулятора

Команды, требующие более одного параметра, используют аккумуляторный стек. Стек аккумулятора проявляется там, где выполняется подряд более одной команды загрузки типа LD (без использования команды выгрузки OUT). Первая после OUT (или начала цикла DL440) команда загрузки помещает данные в аккумулятор на место его прежнего содержимого. Каждая последующая инструкция загрузки, кроме того, сначала «проталкивает» содержимое аккумулятора в 8-уровневый стек, который образован восемью регистровыми парами.

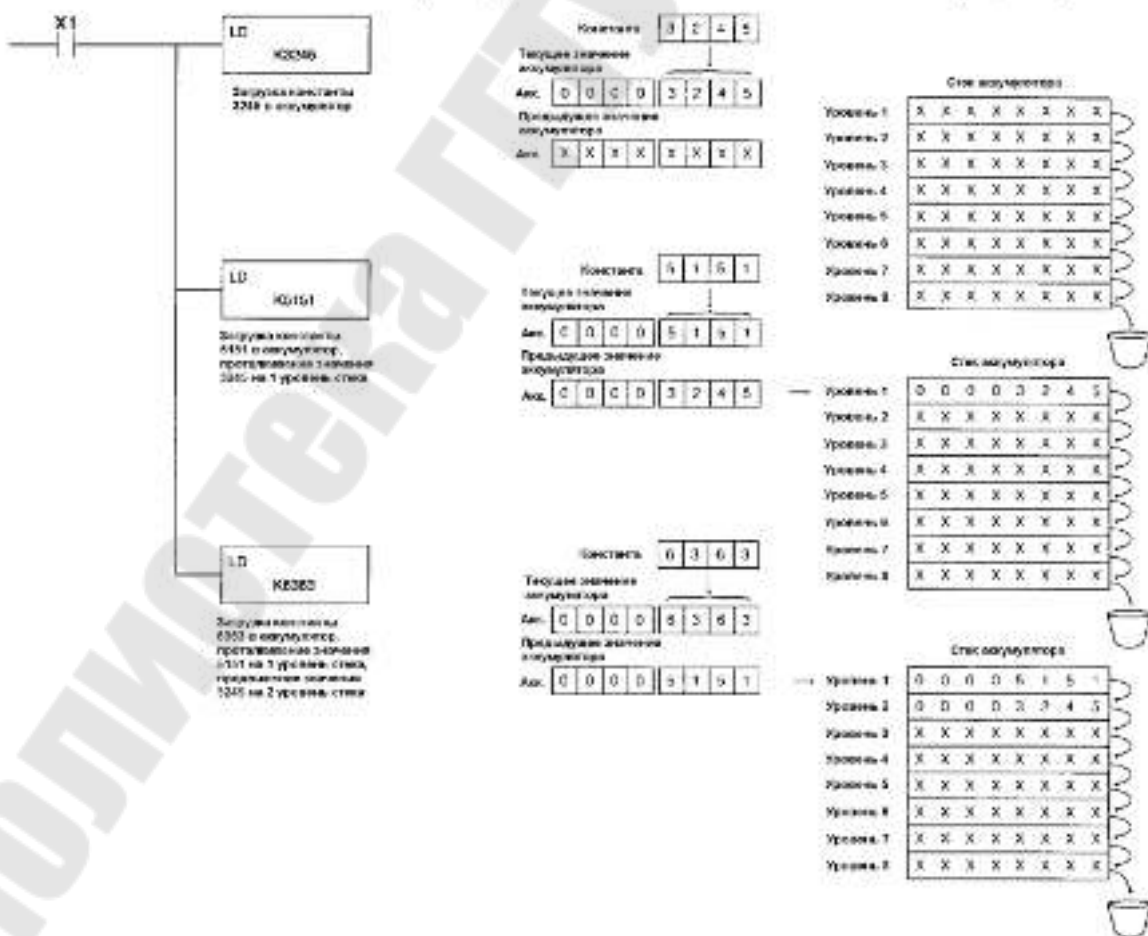
«Проталкивание» в стек происходит следующим образом. Содержимое 7-го уровня стека (регистровая пара V716, V717) копируется в 8-й уровень (регистровая пара V720, V721), прежнее содержимое которого теряется. Содержимое 6-го уровня стека (регистровая пара V714, V715) копируется в 7-й уровень (регистровая пара V716, V717) и т.д. Наконец, содержимое аккумулятора (регистровая пара V700, V701) копируется в 1-й уровень стека (регистровая пара V702, V703).

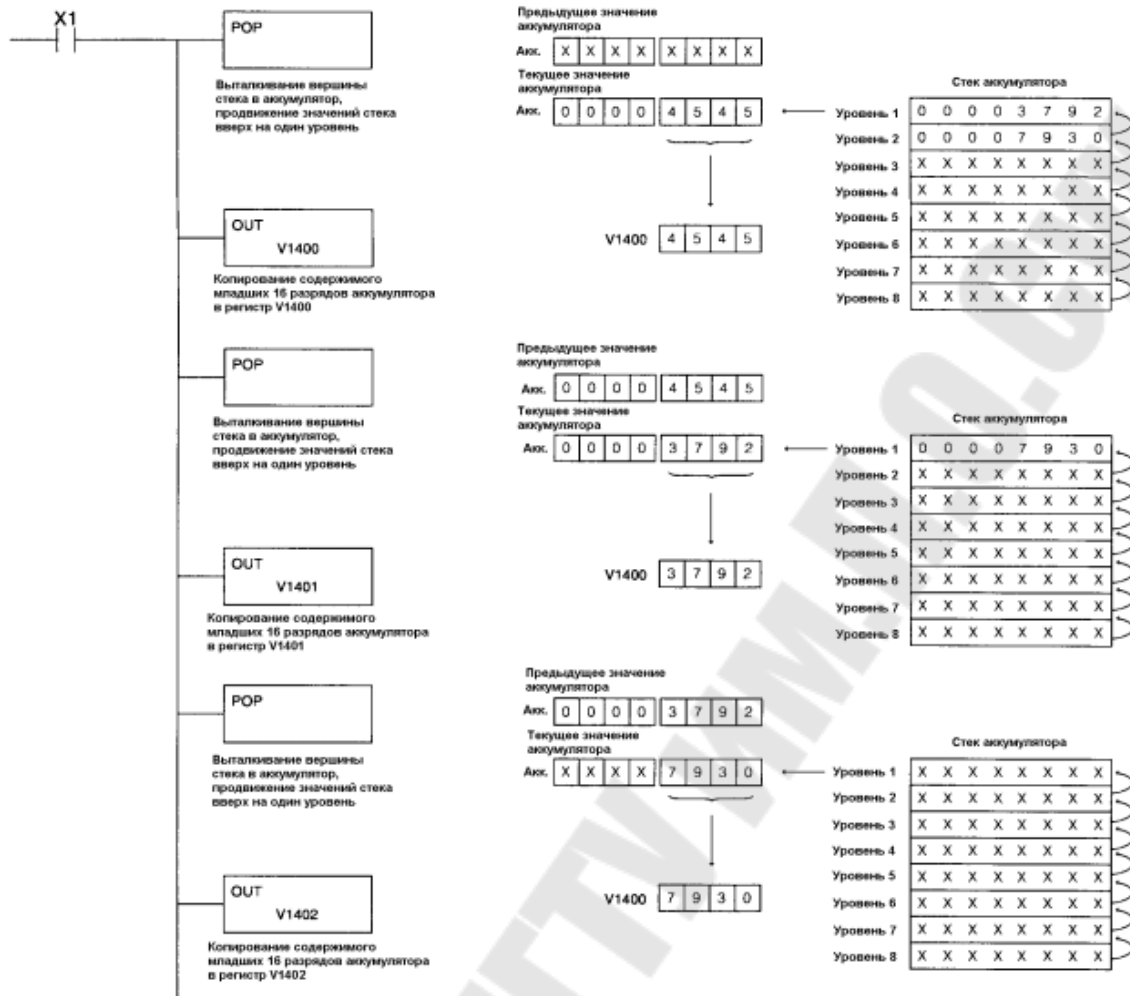
Ниже приводится пример загрузки VCD-константой 3245 второго уровня стека, VCD-константой 5151 первого уровня стека и VCD-константой 6363 аккумулятора.

Команда POP «выталкивает» данные из стека в аккумулятор. Этот процесс идет в направлении, обратном процессу «проталкивания». Когда POP выполняется, значение, которое было в аккумуляторе, замещается данными с уровня 1 стека. Данные уровня n=2,3...8 стека перемещаются на уровень (n-1), а уровень 8 - "дно" стека - обнуляется.

Отображение Аккумулятора и стека аккумулятора в V-памяти ЦП

В случае, когда необходимо прочитать значение, которое было помещено в стек, без выталкивания его в аккумулятор, можно использовать тот факт, что в системной области V-памяти имеются 9 регистровых пар, копирующих аккумулятор и стек аккумулятора, которые могут быть доступны программе. Следует однако помнить, что системная область V-памяти доступна только по чтению.





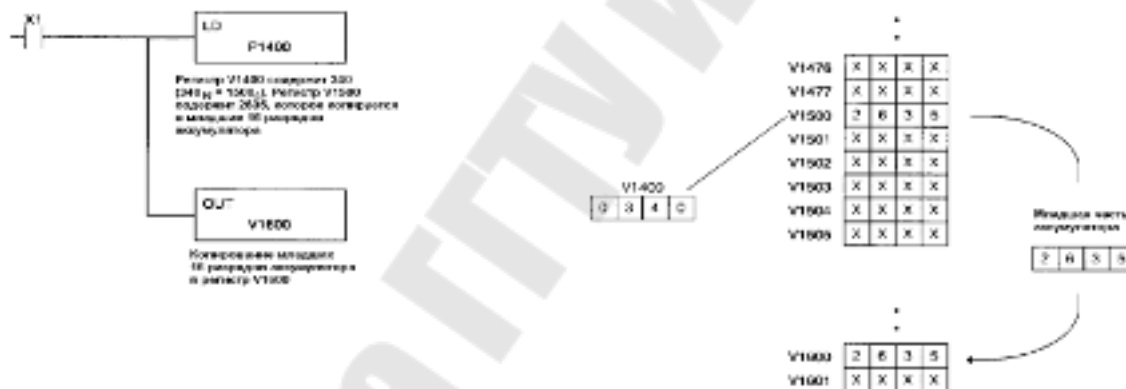
УКАЗАТЕЛИ

Многие команды DL440 могут использовать содержимое какого-либо V-регистра, как номер другого V- регистра, где содержится операнд. В этом случае первый V-регистр называется указателем. Указатели полезны при обращении к массивам операндов.

дов, но могут быть сложны в понимании, если нет опыта работы с указателями (косвенной адресацией).

В DL440 принято нумеровать регистры V-памяти восьмеричными числами. Однако это лишь форма представления номера V-регистра, который в регистре-указателе должен быть представлен двоичным кодом (HEX-формат). Чтобы облегчить перевод восьмеричной формы числа в шестнадцатиричную, в составе DL440 имеется специальная команда LDA 000000 (LOAD ADDRESS) – загрузить адрес 00000 в аккумулятор. LDA попутно выполняет преобразование восьмеричного числа в шестнадцатеричное (HEX).

В следующем примере инструкция LD использует регистр V1400, в котором находится число 340h (в HEX-формате), как указатель на регистр V1500, поскольку HEX-число 340h в восьмеричной записи выглядит как 1500: $340_{16} = 1500_8$. В этом примере регистр V1500 содержит число 2635 в шестнадцатеричной (HEX) форме, которое будет скопировано в регистр V1600, через младшую часть аккумулятора.

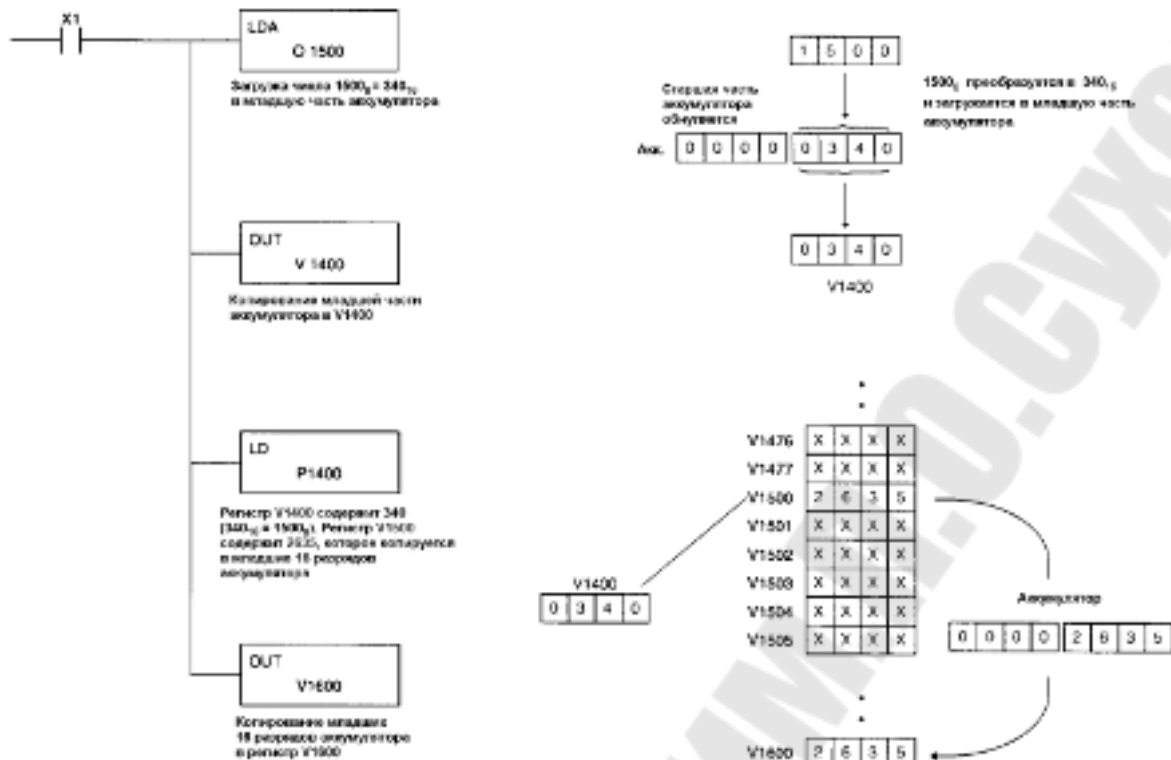


Следующий пример эквивалентен предыдущему с одним отличием. Используется команда LDA, которая автоматически преобразует восьмеричный адрес в шестнадцатеричный эквивалент.

ТАБЛИЧНЫЕ ПРЕОБРАЗОВАНИЯ

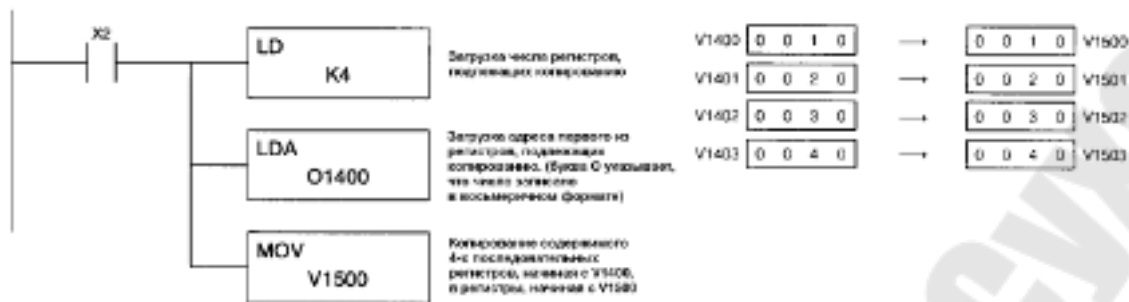
В DL440 можно обрабатывать таблицы данных. Таблица - это массив V-регистров (общим числом до 256), в котором младший по номеру регистр выполняет роль указателя. Здесь сумма содержимого регистра-указателя с его номером дает номер выбранного V-регистра таблицы. Иными словами, регистр-указатель содержит «расстояние» до выбранного V-регистра.

Возможны следующие манипуляции с массивами и таблицами регистров:



- заполнить массив определенным числом;
- найти в массиве заданное число;
- найти в массиве первое число, которое больше заданного;
- скопировать содержимое одного массива в другой;
- скопировать указанное значение из таблицы в заданный регистр с последующим увеличением указателя на 1 (инкрементированием);
- скопировать указанное значение из таблицы в заданный регистр с последующим уменьшением указателя на 1 (декрементированием);
- скопировать значение из заданного регистра в указанное место таблицы с последующим инкрементированием указателя;
- скопировать значение из заданного регистра в указанное место таблицы с последующим декрементированием указателя;
- скопировать содержимое массива в/из картриджа памяти программ.

Пример демонстрирует перемещение данных из группы V-регистров в другую группу V-регистров. Когда перемещаются большие количества данных, табличные инструкции экономят время.



1.6. Использование шагового двигателя в качестве исполнительного механизма

Шаговые двигатели – это электромеханические устройства, преобразующие сигнал управления в угловое (или линейное) перемещение ротора с фиксацией его в заданном положении без устройств обратной связи. Современные шаговые двигатели являются по сути синхронными двигателями без пусковой обмотки на роторе, что объясняется не асинхронным а частотным пуском шагового двигателя. Роторы могут быть возбужденными (активными) и невозбужденными (пассивными).

Существуют три основных типа шаговых двигателей:

1. двигатели с переменным магнитным сопротивлением;
2. двигатели с постоянными магнитами;
3. гибридные двигатели.

В зависимости от конфигурации обмоток двигателя делятся на биполярные и униполярные. Биполярный двигатель имеет одну обмотку в каждой фазе, которая для изменения направления магнитного поля должна переполюсовываться драйвером. Для такого типа двигателя требуется мостовой драйвер, или полумостовой с двухполярным питанием. Всего биполярный двигатель имеет две обмотки и, соответственно, четыре вывода (рис. 1.26а).

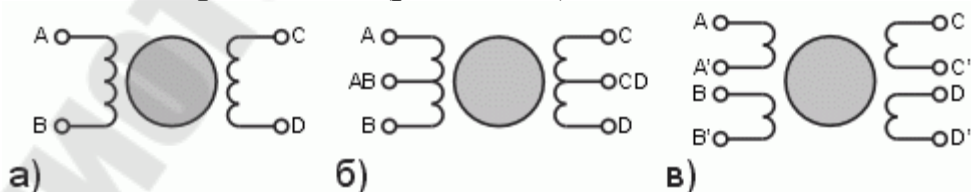


Рисунок 1.26 - Биполярный двигатель (а), униполярный (б) и четырехобмоточный (в)

Униполярный двигатель также имеет одну обмотку в каждой фазе, но от середины обмотки сделан отвод. Это позволяет изменять на-

правление магнитного поля, создаваемого обмоткой, простым переключением половинок обмотки. При этом существенно упрощается схема драйвера. Драйвер должен иметь только 4 простых ключа. Таким образом, в униполярном двигателе используется другой способ изменения направления магнитного поля. Средние выводы обмоток могут быть объединены внутри двигателя, поэтому такой двигатель может иметь 5 или 6 выводов (рис. 1.26б). Иногда униполярные двигатели имеют отдельные 4 обмотки, по этой причине их ошибочно называют 4-х фазными двигателями. Каждая обмотка имеет отдельные выводы, поэтому всего выводов 8 (рис. 1.26в). При соответствующем соединении обмоток такой двигатель можно использовать как униполярный или как биполярный. Униполярный двигатель с двумя обмотками и отводами тоже можно использовать в биполярном режиме, если отводы оставить неподключенными. В любом случае ток обмоток следует выбирать так, чтобы не превысить максимальной рассеиваемой мощности.

Если сравнивать между собой биполярный и униполярный двигатели, то биполярный имеет более высокую удельную мощность. При одних и тех же размерах биполярные двигатели обеспечивают больший момент.

Момент, создаваемый шаговым двигателем, пропорционален величине магнитного поля, создаваемого обмотками статора. Путь для повышения магнитного поля – это увеличение тока или числа витков обмоток. Естественным ограничением при повышении тока обмоток является опасность насыщения железного сердечника. Однако на практике это ограничение действует редко. Гораздо более существенным является ограничение по нагреву двигателя вследствие омических потерь в обмотках. Как раз этот факт и демонстрирует одно из преимуществ биполярных двигателей. В униполярном двигателе в каждый момент времени используется лишь половина обмоток. Другая половина просто занимает место в окне сердечника, что вынуждает делать обмотки проводом меньшего диаметра. В то же время в биполярном двигателе всегда работают все обмотки, т.е. их использование оптимально. В таком двигателе сечение отдельных обмоток вдвое больше, а омическое сопротивление – соответственно вдвое меньше. Это позволяет увеличить ток в корень из двух раз при тех же потерях, что дает выигрыш в моменте примерно 40%. Если же повышенного момента не требуется, униполярный двигатель позволяет уменьшить

габариты или просто работать с меньшими потерями. На практике все же часто применяют униполярные двигатели, так как они требуют значительно более простых схем управления обмотками. Это важно, если драйверы выполнены на дискретных компонентах.

Существует несколько способов управления фазами шагового двигателя.

Первый способ обеспечивается попеременной коммутации фаз, при этом они не перекрываются, в один момент времени включена только одна фаза (рис 1.27а). Этот способ называют "one phase on" full step или wave drive mode. Точки равновесия ротора для каждого шага совпадают с «естественными» точками равновесия ротора у незапитанного двигателя. Недостатком этого способа управления является то, что для биполярного двигателя в один и тот же момент времени используется 50% обмоток, а для униполярного – только 25%. Это означает, что в таком режиме не может быть получен полный момент.

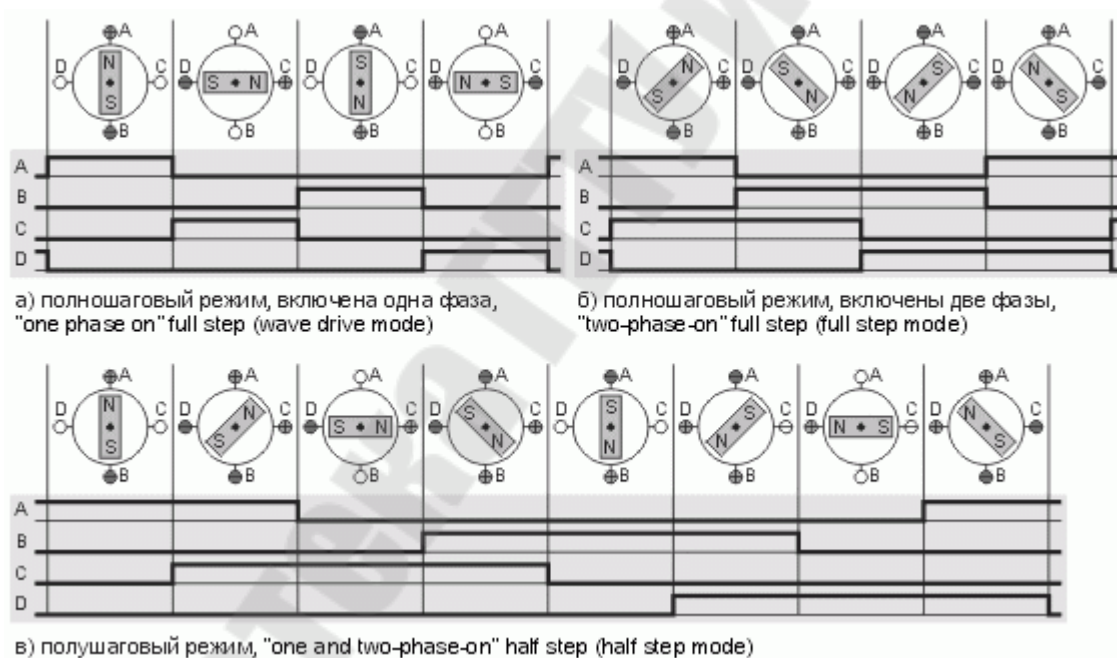


Рисунок 1.27 - Различные способы управления фазами шагового двигателя

Второй способ - управление фазами с перекрытием: две фазы включены в одно и то же время. Его называют "two-phase-on" full step или просто full step mode. При этом способе управления ротор фиксируется в промежуточных позициях между полюсами статора (рис. 1.27б) и обеспечивается примерно на 40% больший момент, чем в

случае одной включенной фазы. Этот способ управления обеспечивает такой же угол шага, как и первый способ, но положение точек равновесия ротора смещено на пол-шага.

Третий способ является комбинацией первых двух и называется полушаговым режимом, "one and two-phase-on" half step или просто half step mode, когда двигатель делает шаг в половину основного. Этот метод управления достаточно распространен, так как двигатель с меньшим шагом стоит дороже и очень заманчиво получить от 100-шагового двигателя 200 шагов на оборот. Каждый второй шаг запитан лишь одна фаза, а в остальных случаях запитаны две (рис. 1.27в). В результате угловое перемещение ротора составляет половину угла шага для первых двух способов управления. Кроме уменьшения размера шага этот способ управления позволяет частично избавиться от явления резонанса. Полушаговый режим обычно не позволяет получить полный момент, хотя наиболее совершенные драйверы реализуют модифицированный полушаговый режим, в котором двигатель обеспечивает практически полный момент, при этом рассеиваемая мощность не превышает номинальной.

Еще один способ управления называется микрошаговым режимом или micro stepping mode. При этом способе управления ток в фазах нужно менять небольшими шагами, обеспечивая таким образом дробление половинного шага на еще меньшие микрошаги. Когда одновременно включены две фазы, но их токи не равны, то положение равновесия ротора будет лежать не в середине шага, а в другом месте, определяемом соотношением токов фаз. Меняя это соотношение, можно обеспечить некоторое количество микрошагов внутри одного шага. Кроме увеличения разрешающей способности, микрошаговый режим имеет и другие преимущества, которые будут описаны ниже. Вместе с тем, для реализации микрошагового режима требуются значительно более сложные драйверы, позволяющие задавать ток в обмотках с необходимой дискретностью. Полушаговый режим является частным случаем микрошагового режима, но он не требует формирования ступенчатого тока питания катушек, поэтому часто реализуется.

1. Одиночные импульсы - самый простой способ. Одновременно подключается только одна катушка. Необходимо 48 пульсов чтобы ротор совершил один полный оборот. Каждый пульс перемещает ротор на 7,5 градусов.

импульс	катушка а1	катушка в1	катушка а2	катушка в2
1	вкл.			
2		вкл.		
3			вкл.	
4				вкл.

2. Двойной импульс - одновременное подключение двух соседних катушек. В этом случае также необходимо 48 пульсов чтобы ротор совершил один полный оборот. Каждый пульс перемещает ротор на 7,5 градусов.

импульс	катушка а1	катушка в1	катушка а2	катушка в2
1	вкл.	вкл.		
2		вкл.	вкл.	
3			вкл.	вкл.
4	вкл.			вкл.

3. Комбинированные импульсы - чередование первого и второго способа. Двигатель нуждается в 96 пульсах, чтобы совершить один оборот. Каждый пульс перемещает ротор приблизительно на 3,75 градуса.

импульс	катушка а1	катушка в1	катушка а2	катушка в2
1	вкл.			
2	вкл.	вкл.		
3		вкл.		
4		вкл.	вкл.	
5			вкл.	
6			вкл.	вкл.
7				вкл.
8	вкл.			вкл.

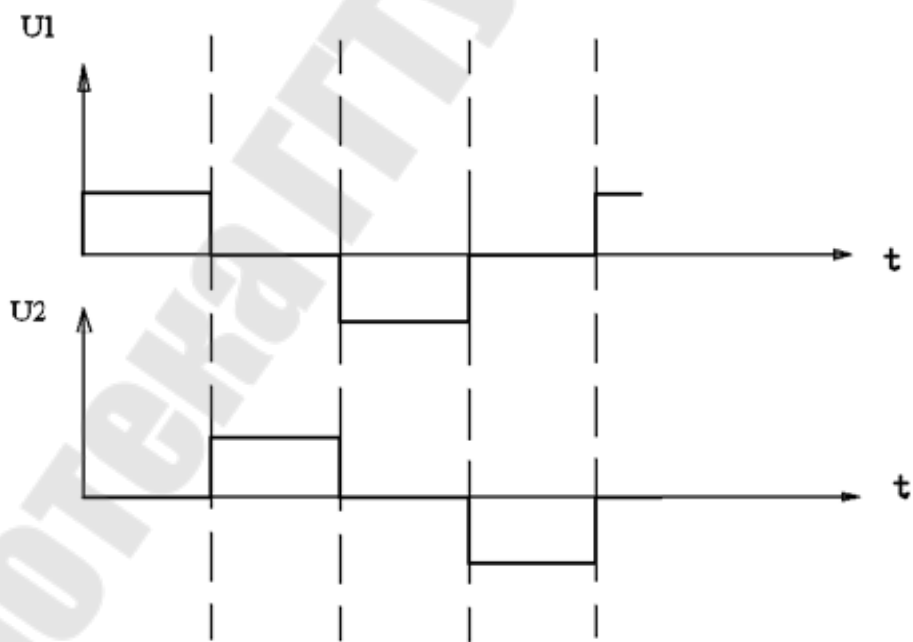
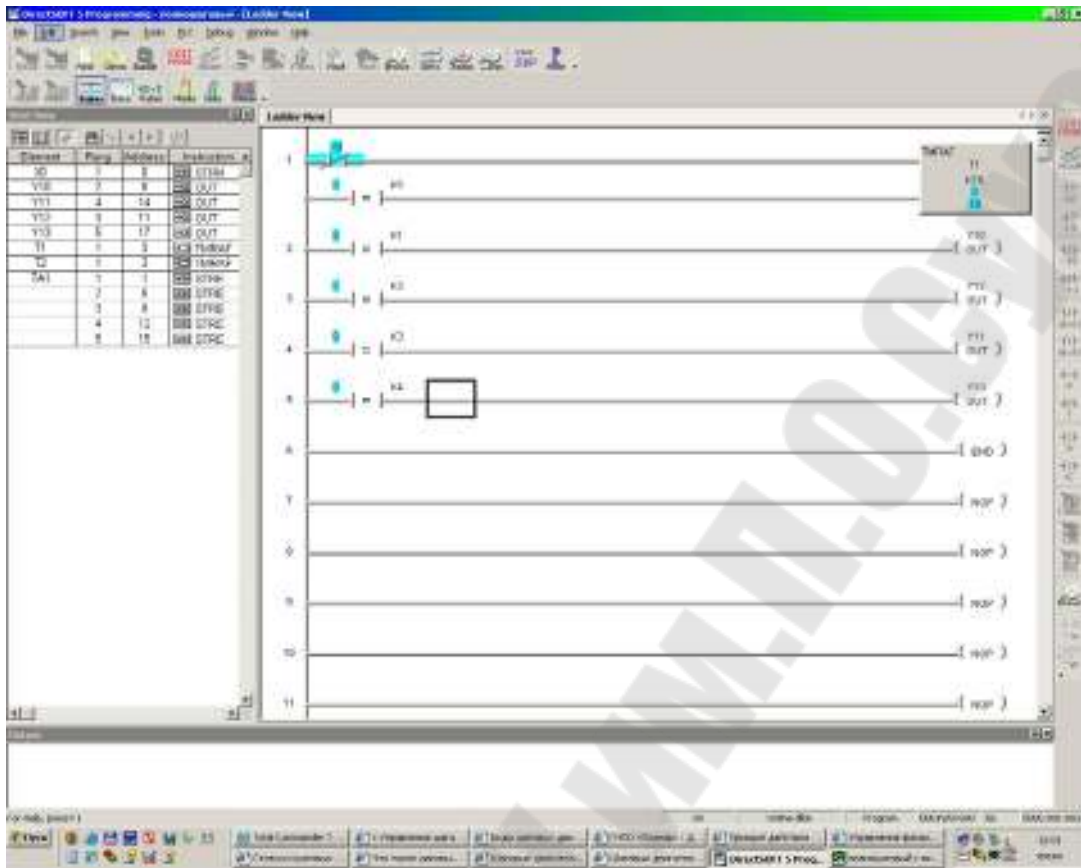


Рисунок 1.28 – Вариант реализации полношагового режима с включением обмоток поочередно

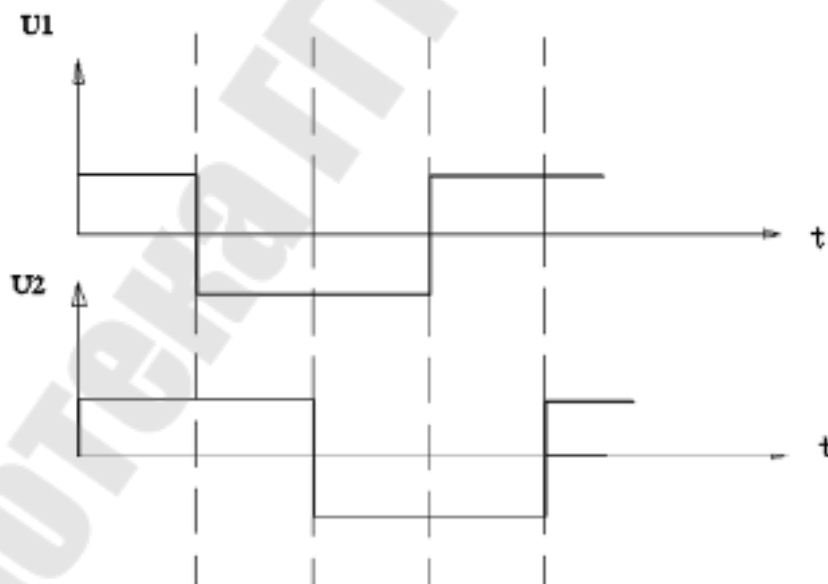
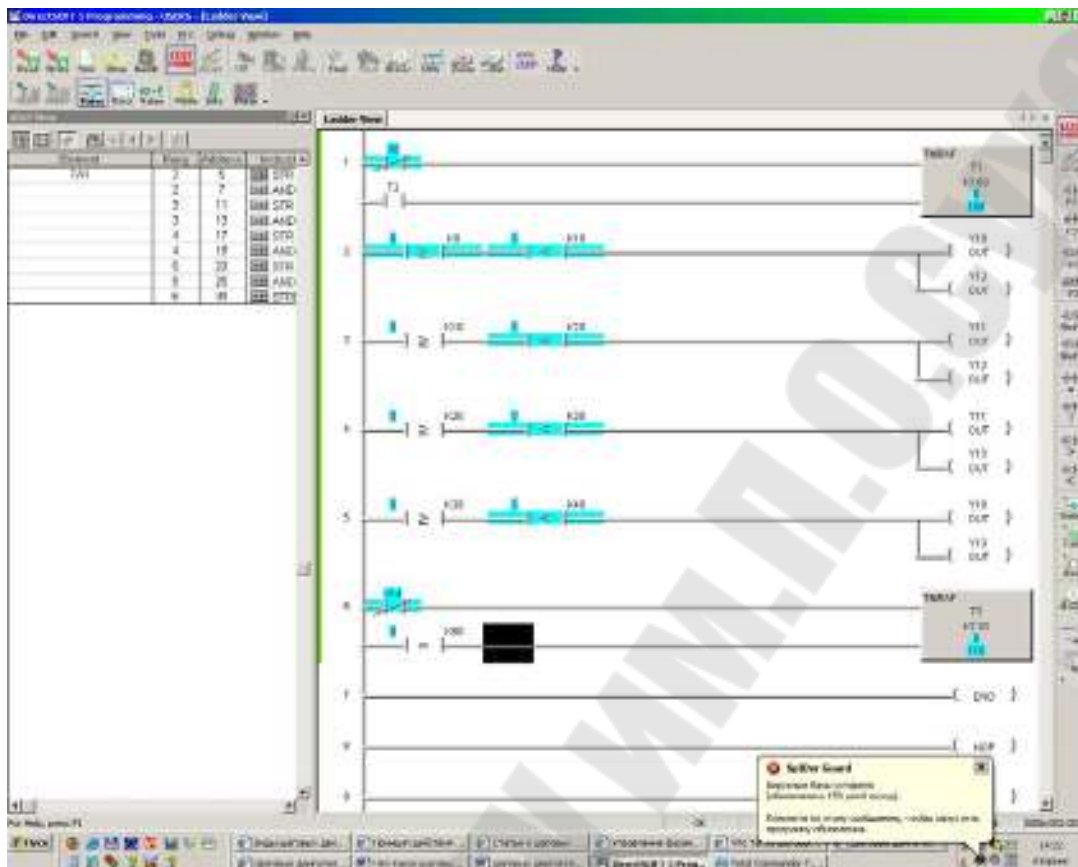


Рисунок 1.29 – Вариант реализации полношагового режима с включением двух обмоток

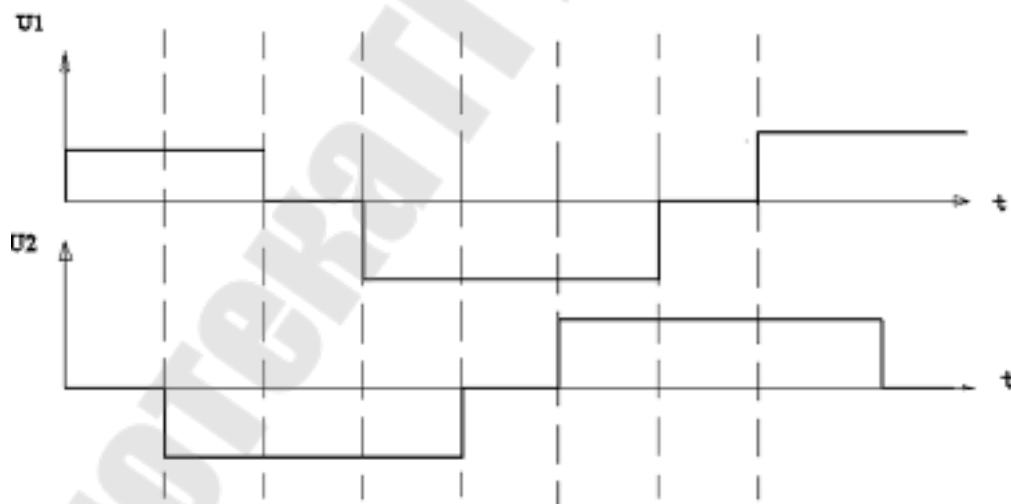
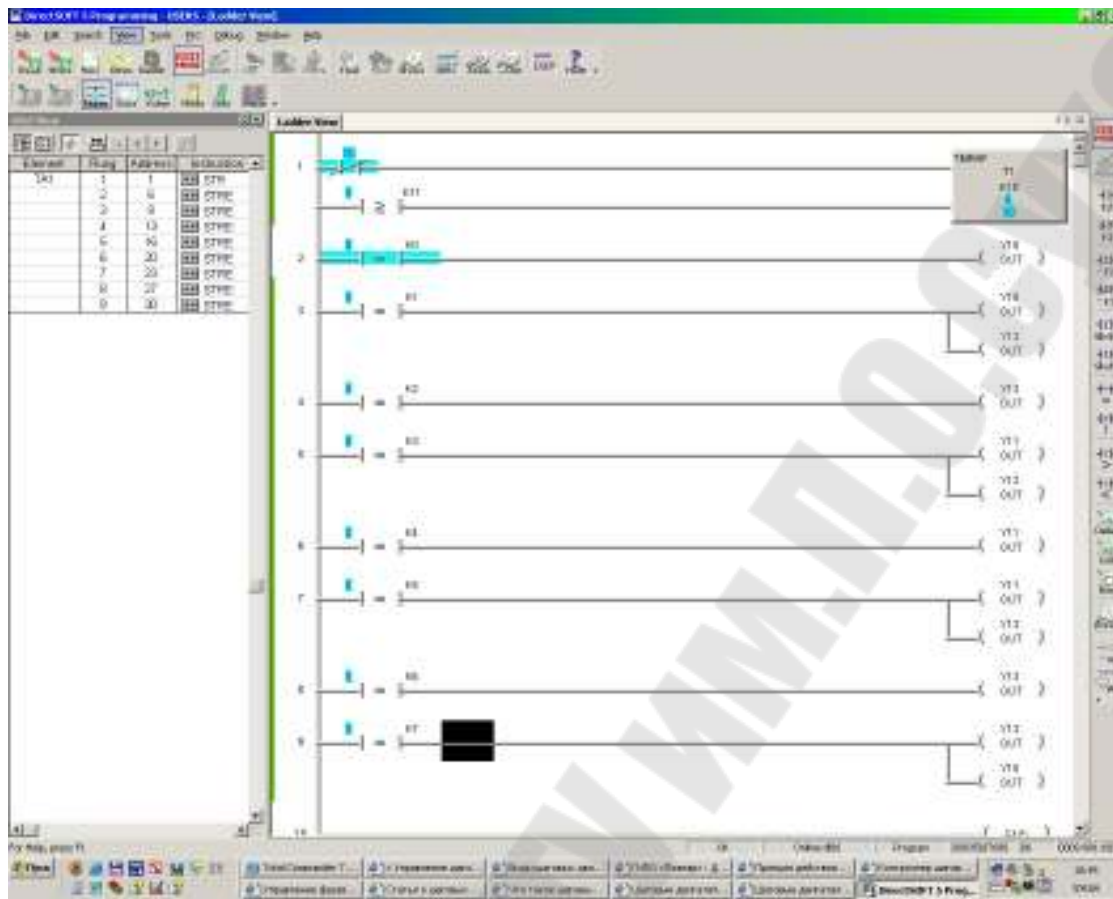


Рисунок 1.30 – Вариант реализации полушагового режима

2. Ход работы

2.1 Создание схемы электрической принципиальной

Собрать схему электрическую принципиальную, представленную на рис. 2.1.

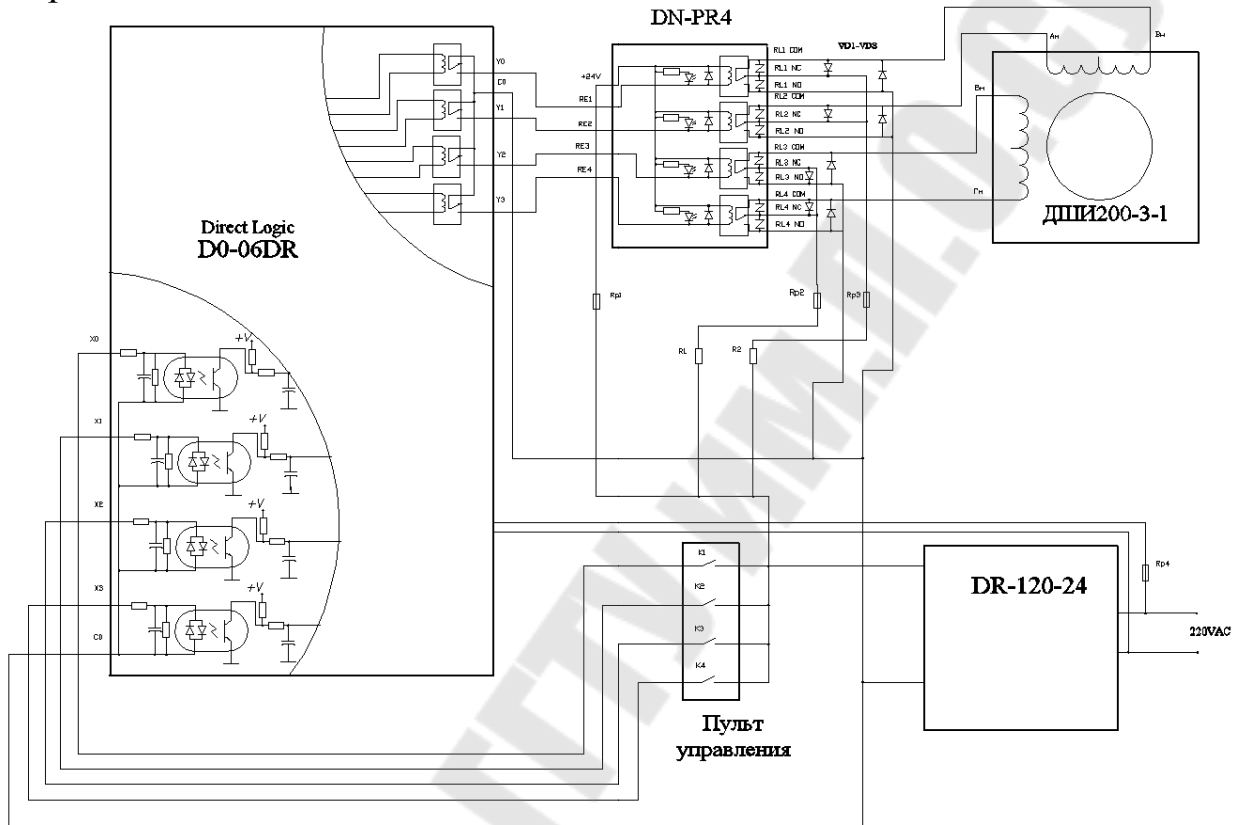


Рисунок 2.1 - Схема электрическая принципиальная

После сборки схемы, дать ее проверить лаборанту или преподавателю.

2.2 Создание проекта автоматического управления шаговым двигателем в среде Direct Soft 32

Многие задачи управления могут быть описаны, как последовательность событий. Управляющая программа должна проверять порядок выполнения событий. Она не только выполняет обычные операции управления, но и учитывает возможные неисправности и критические ситуации. Выполняемые задачи будут следующие:

- 1) Кнопка пуск включена.
- 2) Шаговый двигатель совершает в полношаговом режиме 20 шагов в направлении по часовой стрелке с интервалом тактовых импульсов в 1 секунду.
- 3) Двигатель совершает в полношаговом режиме 80 шагов в направлении по часовой стрелке с интервалом тактовых импульсов 0.1 секунды.
- 4) Шаговый двигатель совершает в полношаговом режиме 160 шагов в направлении по часовой стрелке с интервалом тактовых импульсов 0.01 секунды.
- 5) Шаговый двигатель совершает в полушаговом режиме 32 шага в направлении против часовой стрелки, с интервалом тактовых импульсов 0.1 секунды.
- 6) Переход на начальную стадию.

Схема алгоритма управляющей программы показана на рисунке 2.2.

Создать новый проект в среде Direct Soft 32 согласно разработанного алгоритма.

Видеографическое представление управляющей программы представлено на рис. 2.3.

Скомпилировать программу.

Проверить мнемоническое представление программы, представленное на рис 2.4.

Запустить программу в пошаговом режиме.

Загрузить программу в DL06 и запустить ее в автоматическом режиме.

Продемонстрировать работу стенда преподавателю.

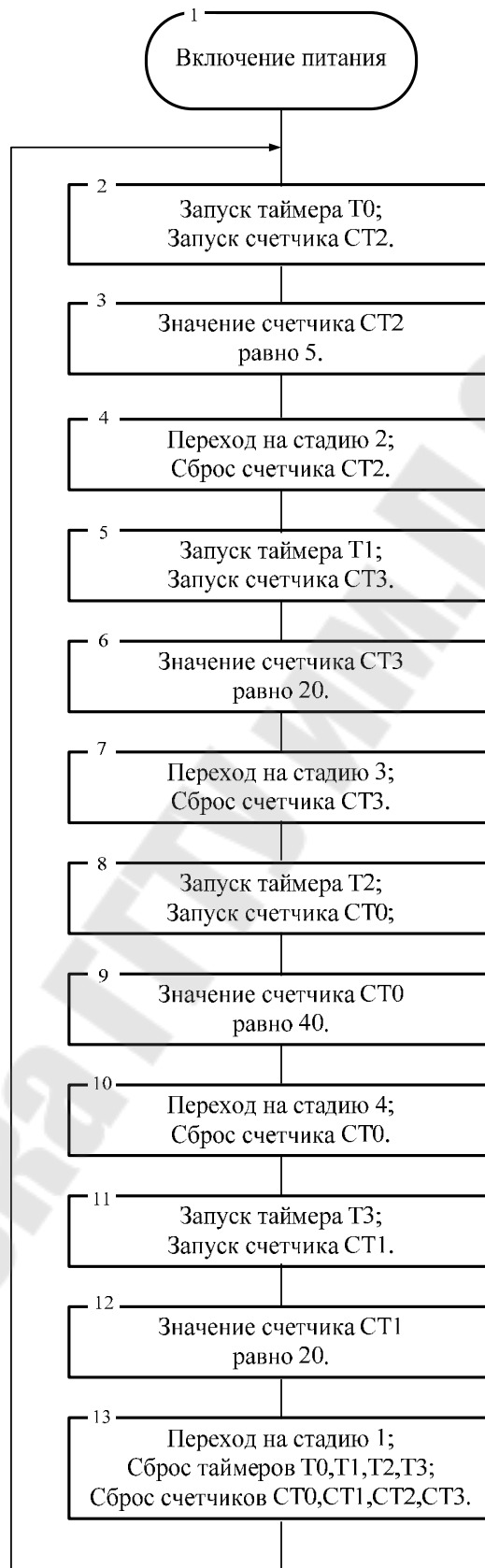
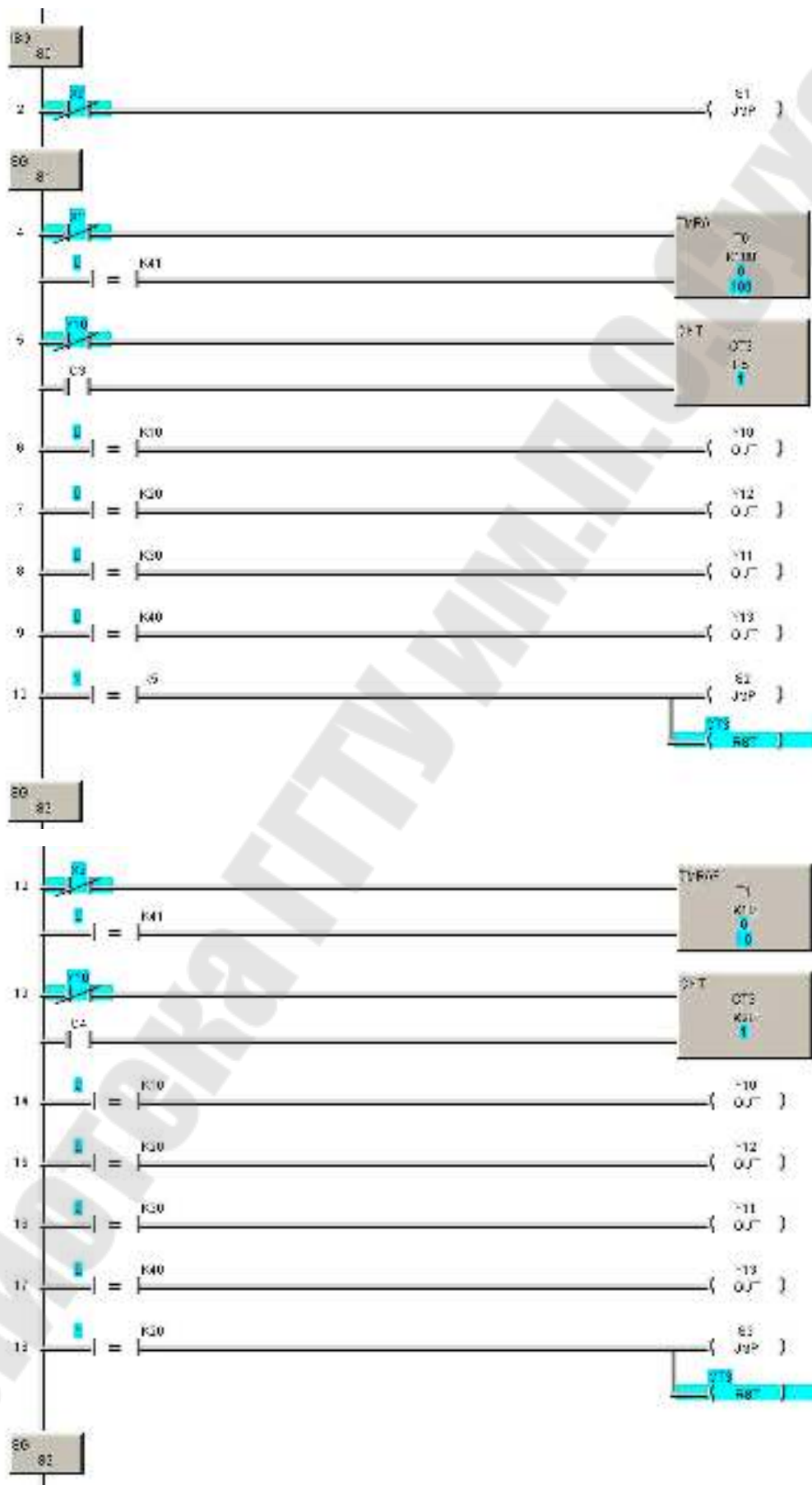
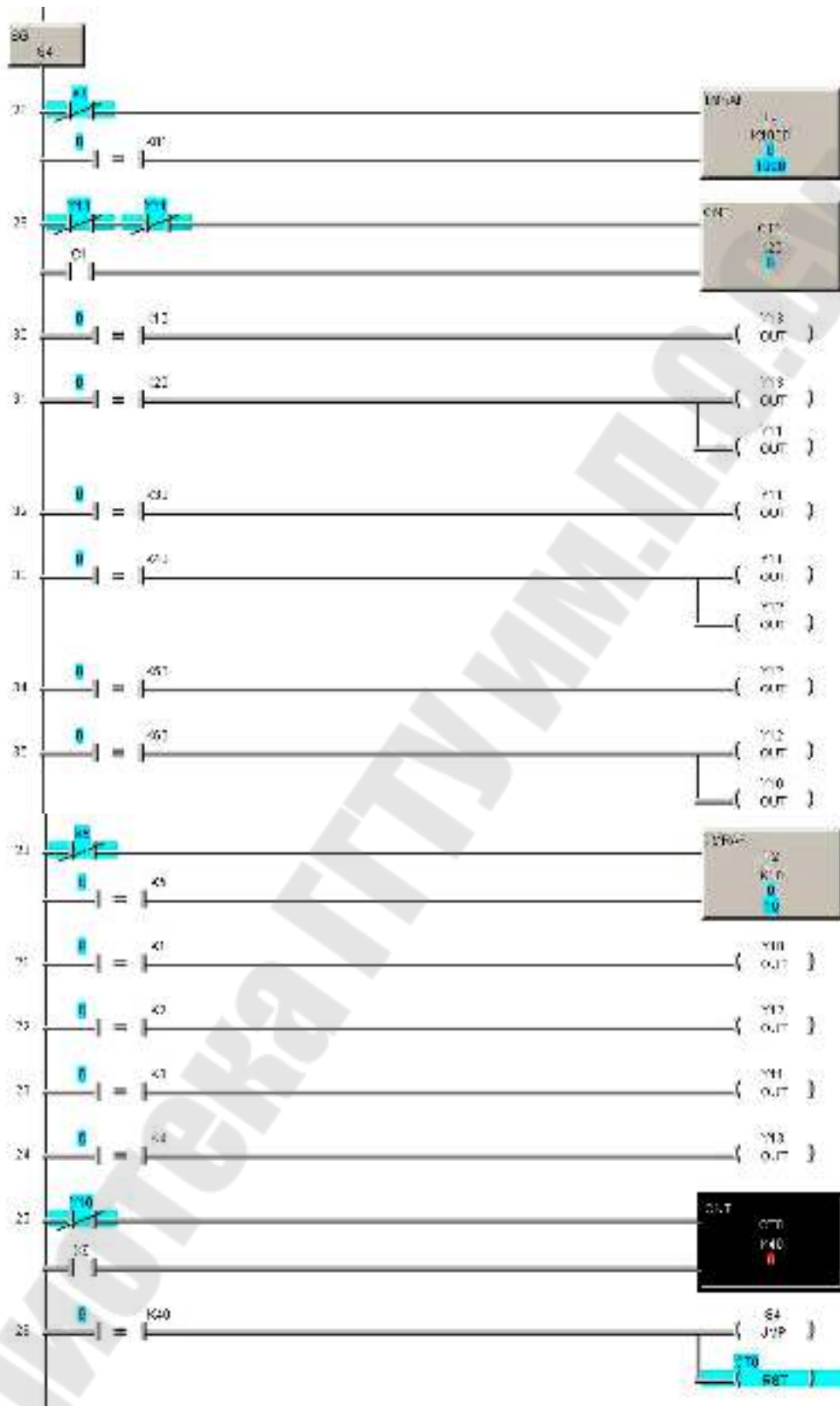


Рисунок 2.2- Алгоритм управляющей программы





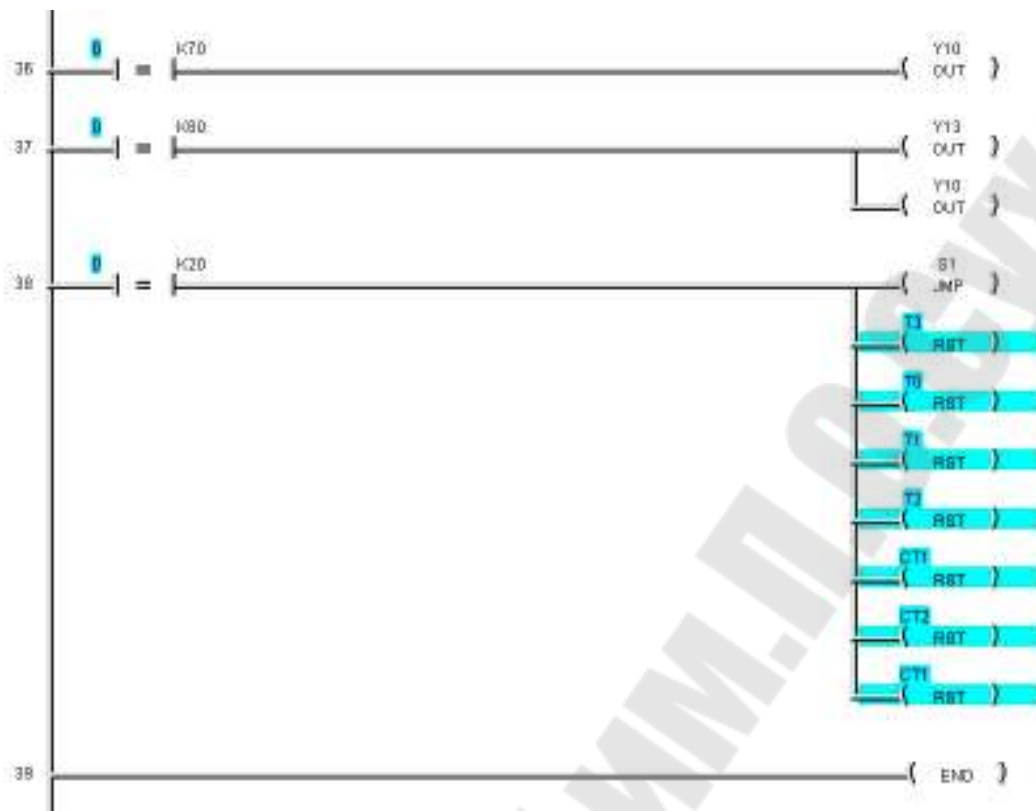


Рисунок 2.3 - Видеографическое представление управляющей программы

0	ISG	S0			
2	STRN	X0			
3	JMP	S1			
4	SG	S1			
6	STRN	X1			
7	STRE	TA0 K41			
9	TMRA	T0 K100			
12	STRN	Y10			
13	STR	C3			
14	CNT	CT3 K5			
17	STRE	TA0 K10			
19	OUT	Y10			
20	STRE	TA0 K20			
22	OUT	Y12	84	CNT	CT0 K40
23	STRE	TA0 K30	87	STRE	CTA0 K40
25	OUT	Y11	89	JMP	S4
26	STRE	TA0 K40	90	RST	CT0
28	OUT	Y13	91	SG	S4
29	STRE	CTA3 K5	93	STRN	X7
31	JMP	S2	94	STRE	TA3 K81
32	RST	CT3	96	TMRAF	T3 K1000
33	SG	S2	99	STRN	Y13
35	STRN	X3	100	ANDN	Y11
36	STRE	TA1 K41	101	STR	C1
38	TMRAF	T1 K10	102	CNT	CT1 K20
41	STRN	Y10	105	STRE	TA3 K10
42	STR	C4	107	OUT	Y13
43	CNT	CT3 K20	108	STRE	TA3 K20
46	STRE	TA1 K10	110	OUT	Y13
48	OUT	Y10	111	OUT	Y11
49	STRE	TA1 K20	112	STRE	TA3 K30
51	OUT	Y12	114	OUT	Y11
52	STRE	TA1 K30	115	STRE	TA3 K40
54	OUT	Y11	117	OUT	Y11
55	STRE	TA1 K40	118	OUT	Y12
57	OUT	Y13	119	STRE	TA3 K50
58	STRE	CTA3 K20	121	OUT	Y12
60	JMP	S3	122	STRE	TA3 K60
61	RST	CT3	124	OUT	Y12
62	SG	S3	125	OUT	Y10
64	STRN	X5	126	STRE	TA3 K70
65	STRE	TA2 K5	128	OUT	Y10
67	TMRAF	T2 K10	129	STRE	TA3 K80
70	STRE	TA2 K1	131	OUT	Y13
72	OUT	Y10	132	OUT	Y10
73	STRE	TA2 K2	133	STRE	CTA1 K20
75	OUT	Y12	135	JMP	S1
76	STRE	TA2 K3	136	RST	T3
78	OUT	Y11	137	RST	T0
79	STRE	TA2 K4	138	RST	T1
81	OUT	Y13	139	RST	T2
82	STRN	Y10	140	RST	CT1
83	STR	X6	141	RST	CT2
			142	RST	CT1
			143	END	

Рисунок 2.4 – Мнемоническое представление управляющей

Программы

3. Контрольные задания

4. По заданию преподавателя выбрать требуемый шаговый двигатель и собрать электрическую цепь на базе ПЛК и модуля гальванической развязки для работы в автономном режиме.

5. Создать проект для управления шаговым двигателем в среде Direct Soft согласно индивидуального задания.

6. В созданном проекте изменить направление движения двигателя на реверс.

7. В созданном проекте уменьшить скорость движения двигателя в два раза.

8. Продемонстрировать работу проектов преподавателю.

4. Содержимое отчета

Отчет должен содержать цель работы, индивидуальное задание, схему подключения шагового двигателя к ПЛК, временные диаграммы работы исполнительного механизма, алгоритм функционирования, программный код DirectSOFT, выводы.

5. Контрольные вопросы

1. Назначение ПЛК.
2. Основные характеристики ПЛК.
3. Системы программирования ПЛК.
4. Схема подключения и работы входов ПЛК
5. Схема подключения и работы выходов ПЛК
6. Схема подключения и работы счетчика ПЛК
7. Схема подключения и работы таймера ПЛК
8. Конфигурирование ПЛК
9. Назначение DirectSOFT.
10. Основные режимы работы DirectSOFT.
11. Каким образом составить программу в среде DirectSOFT.
12. Как записать программу в ПЛК
13. Как считать программу в ПЛК
14. Конфигурирование канала передачи данных ПЭВМ-ПЛК
15. Основные положения RLL.
16. Логические команды RLL.
17. Команды изменения состояния реле.
18. Логика сравнения.

19. Соединения цепей в RLL.
20. Указатели.
21. Табличные преобразования RLL.
22. Принцип работы шаговых двигателей
23. Особенности управления в полношаговом режиме.
24. Особенности управления в полушаговом режиме.
25. Особенности коммутации индуктивных нагрузок при помощи ПЛК

Список литературы

1. Андреев Е. Б., Куцевич Н. А. Синенко О. В. SCADA-системы: взгляд изнутри/ Е. Б. Андреев, Н. А. Куцевич. О. В. Синенко.— М.: издательство РТСофт. 2004.— 176с;
2. Букреев В. Г., Цхе А. В. Основы инструментальной системы разработки АСУ/ В. Г. Букреев, А. В. Цхе.— Томск: издательство ТПУ, 2003 .—127;
3. Локотов А. Что должна уметь система SCADA/ А. Локотов// Современные технологии автоматизации.— 1998.— 3.—с 44;
4. Руководство пользователя. TRACE MODE 6 & T-FACTORY. Быстрый старт. Издание пятое (к релизу 6.03.1). Москва 2006. AdAstrA Research Group. Ltd.-1 63 с.
5. Руководство пользователя. TRACE MODE 6. Издание седьмое (к релизу 6.03.1). Москва 2006. AdAstiA Research Group. Ltd.. том 1-554 с. том 2-598 с.
6. Деменков Н.П. SCADA - системы как инструмент проектирования АСУ ТП. М.: Изд-во МГТУ им. Н.Э. Баумана. 2004. - 328 с.
7. Управляющие вычислительные комплексы : Учеб. пособие/ Под ред. Н.Л. Прохорова. М.: Финансы и статистика. 2003. -352 с.
8. Руководство пользователя контроллера DL06, номер руководства D0-06USER-M-RUS часть 1 перевод ООО «ПЛКСистемы».
9. Программное обеспечение для программирования PC-DSOFT5-M часть 1 перевод ООО «ПЛКСистемы».
10. Системы автоматизированного управления электроприводами: Учеб. пособие / Г.И. Гульков, Ю.Н. Петренко, Е.П. Раткевич О Л. Симоненкова; Под общ. ред. Ю.Н. Петренко. — Мн.: Новое знание, 2004. — 384 с.: ил. 18ВМ 985-475-085-Х.
11. Кенио Т. Шаговые двигатели и их микропроцессорные системы управления: Пер. с англ. - М.: Энергоатомиздат, 1987. - 200 с.: ил.

Ковалев Алексей Викторович
Литвинов Дмитрий Александрович

**УПРАВЛЕНИЕ ПРОМЫШЛЕННЫМИ
ОБЪЕКТАМИ**

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименному курсу
для студентов специальности 1-36 04 02
«Промышленная электроника»
дневной и заочной форм обучения
В двух частях
Часть 1

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 27.03.14.

Рег. № 45Е.
<http://www.gstu.by>