Министерство науки и высшего образования Российской Федерации
ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра «Мехатроника и автоматизация»

Басков С.Н.

Система программирования ПЛК CoDeSys

Методические указания для выполнения практических работ

Челябинск, 2018

CoDeSys предоставляет программисту удобную среду для программирования контроллеров на языках стандарта МЭК 61131-3. Используемые редакторы и отладочные средства базируются на широко известных и хорошо себя зарекомендовавших принципах, знакомых по другим популярным средам профессионального программирования (например, Visual C++).

В пособии описан порядок выполнения работ, сформулированы требования к отчётным материалам, даны контрольные вопросы, позволяющие закрепить изучаемый материал.

Предназначен для магистров, обучающихся по направлению 27.04.04 «Управление в технических системах».

ВВЕДЕНИЕ

ОСОБЕННОСТИ УСТРОЙСТВА РЕАЛЬНОГО ПЛК

Устройство ПЛК. Аппаратно ПЛК является вычислительной машиной, поэтому архитектура его процессорного ядра практически не отличается от архитектуры компьютера. Отличия заключены в составе периферийного оборудования, отсутствуют видеоплата, средства ручного ввода и дисковая подсистема. Вместо них ПЛК имеет блоки входов и выходов.

Конструктивно контроллеры подразделяют на моноблочные, модульные и распределенные. Моноблочные, или одноплатные, ПЛК имеют фиксированный набор входов-выходов. В модульных контроллерах модули входов-выходов устанавливаются в разном составе и количестве в зависимости от требуемой конфигурации. Так достигается минимальная аппаратная избыточность. В распределенных системах модули или даже отдельные входы-выходы, образующие единую систему управления, могут быть разнесены на значительные расстояния.

Входы-выходы

В современных ПЛК широко используют *аналоговые и дискретные входы и выходы.* Дискретный входной сигнал может принимать только два состояния – логического нуля и логической единицы. Так, наличие тока (или напряжения) в цепи входа считается обычно логической единицей. Отсутствие тока (напряжения) означает логический 0. Датчиками, формирующими такой сигнал, являются кнопки ручного управления, концевые датчики, датчики движения, контактные термометры и многие др.

Дискретный выходной сигнал также имеет два состояния – включен и выключен. Сфера применения бинарных выходов очевидна:

электромагнитные реле, силовые пускатели, электромагнитные клапаны, световые сигнализаторы.

Аналоговый или непрерывный сигнал отражает уровень напряжения или тока, соответствующий некоторой физической величине в каждый момент времени. Этот уровень может относиться к температуре, давлению, весу, положению, скорости, частоте.

Аналоговые входы контроллеров могут иметь различные параметры и возможности. Так, к их параметрам относятся: разрядность АЦП, диапазон входного сигнала, время и метод преобразования, несимметричный или дифференциальный вход, уровень шума и нелинейность, возможность автоматической калибровки, программная или аппаратная регулировка коэффициента усиления, фильтрация.

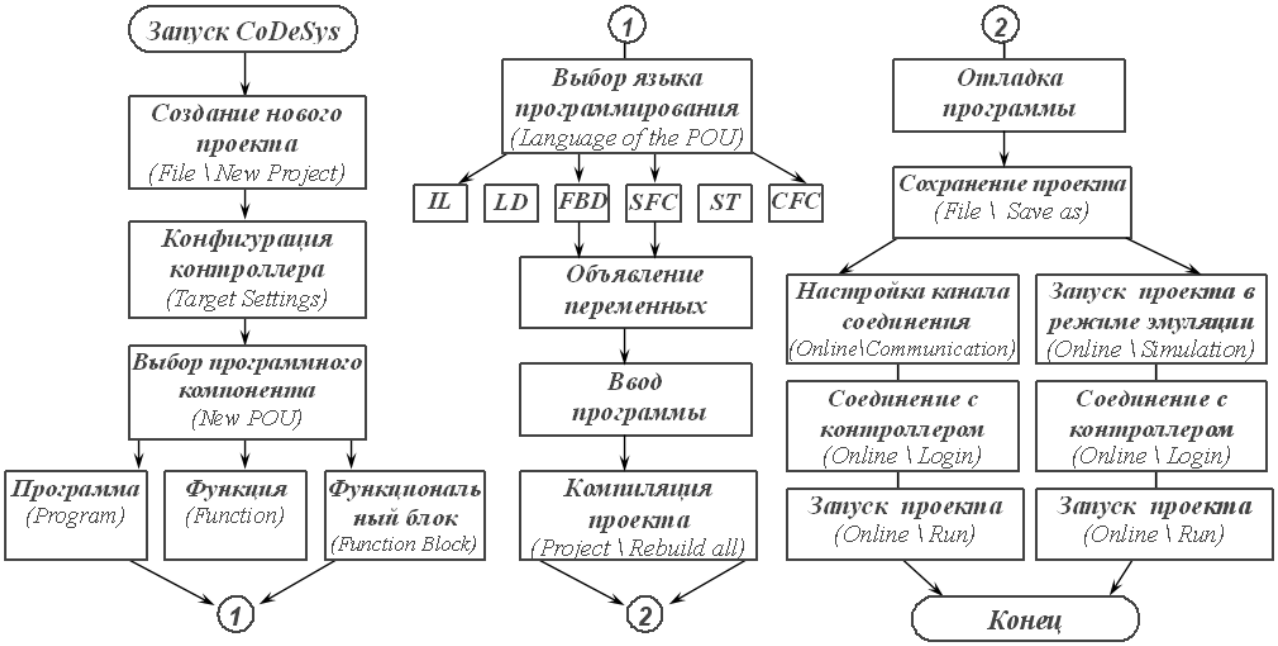
Помимо «классических» дискретных и аналоговых входов-выходов многие ПЛК имеют специализированные входы-выходы.

Среда разработки CoDeSys

CoDeSys – это современный инструмент для программирования контроллеров (CoDeSys образуется от слов Controllers Development System).

CoDeSys предоставляет программисту удобную среду для программирования контроллеров на языках стандарта МЭК 61131-3. Используемые редакторы и отладочные средства базируются на широко известных и хорошо себя зарекомендовавших принципах, знакомых по другим популярным средам профессионального программирования (такие, как Visual C++). Этапы разработки проекта показаны на рис. 1.

Этапы разработки проекта



Р и с. 1. Этапы разработки проекта

Ввод конфигурации центрального процессора ПЛК

Запуск CoDeSys. CoDeSys запускается как большинство Windows приложений.

Пуск → Программы → Moeller Software → XSoft V2.3.3 → XSoft V2.3.3.

Создание нового проекта. Чтобы начать новый проект, нужно выполнить команду New (Новый проект) из меню File.

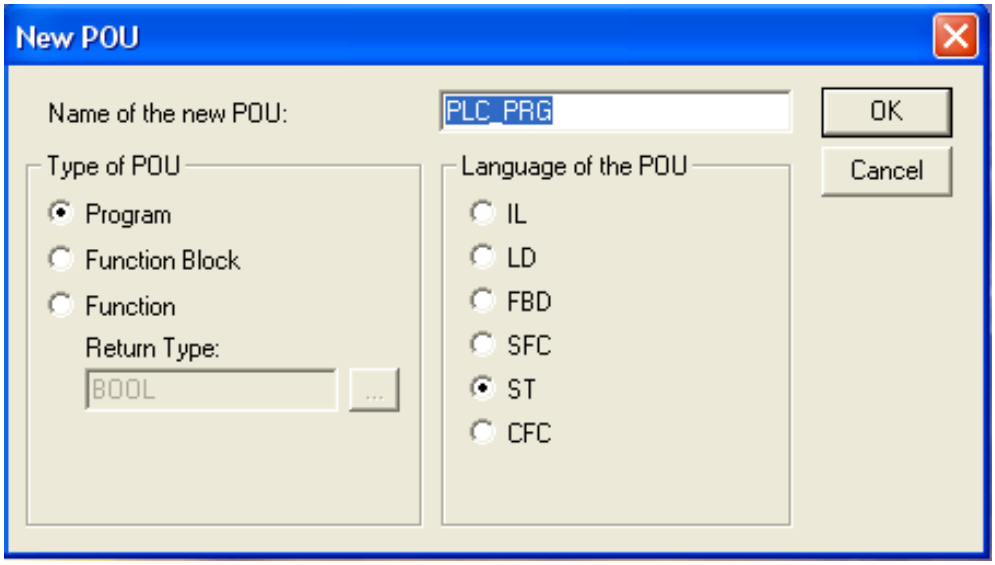
Проект включает следующие объекты: POU, типы данных, визуализации, ресурсы, библиотеки. Каждый проект сохраняется в отдельном файле.

Конфигурация контроллера. При этом сразу откроется окно Target Settings (конфигурирование ПЛК). В данном окне нужно выбрать марку модуля центрального процессора (CPU) ПЛК в соответствии с аппаратными средствами своего контроллера.

Выбор типа программного компонента и языка программирования

Выбор программного компонента. Следующее диалоговое окно (рис. 2) определяет тип первого программного компонента (New POU).

Типов POU в МЭК 61131-3 всего 3: функция (FUNCTION), функциональный блок (FUNCTION BLOCK) и программа (PROGRAM).



Р и с. 2. Окно выбора нового программного компонента

Функция. Результат работы функции всегда однозначно определяется значениями ее параметров без учета предыстории работы. Функция не может иметь внутренней памяти и возвращает единственное значение (оно может быть составным). Функцию можно использовать в выражениях, например: Y := Sin(X) – 1. Функция не имеет возможности запомнить этапы своего выполнения (нет памяти) между вызовами, поэтому она всегда выполняется синхронно.

Функциональный блок представляет собой подпрограмму, объединенную с собственной структурой данных.

Как и при объявлении структуры, создание нового функционального блока создает только новый тип. Для практического использования необходимо объявить экземпляр соответствующего функционального блока.

Функциональный блок может иметь произвольное число входных и выходных переменных, доступных извне (интерфейс компонента). Внутренние переменные не доступны другим компонентам (к отладчику это не относится). Объявление дополнительного экземпляра функционального блока приводит к выделению памяти в области данных. Исполняемый код для разных экземпляров один и тот же.

Программа в CoDeSys аналогична функциональному блоку, но не имеет экземпляров. Программа – это глобальный объект. Традиционно программы применяются в CoDeSys для очень крупных программных модулей в многозадачных проектах.

Языки программирования

Список Инструкций (IL). Простой машинно-независимый ассемблер.

Структурированный текст (ST) Высокоуровневый 'Паскаль-подобный' язык.

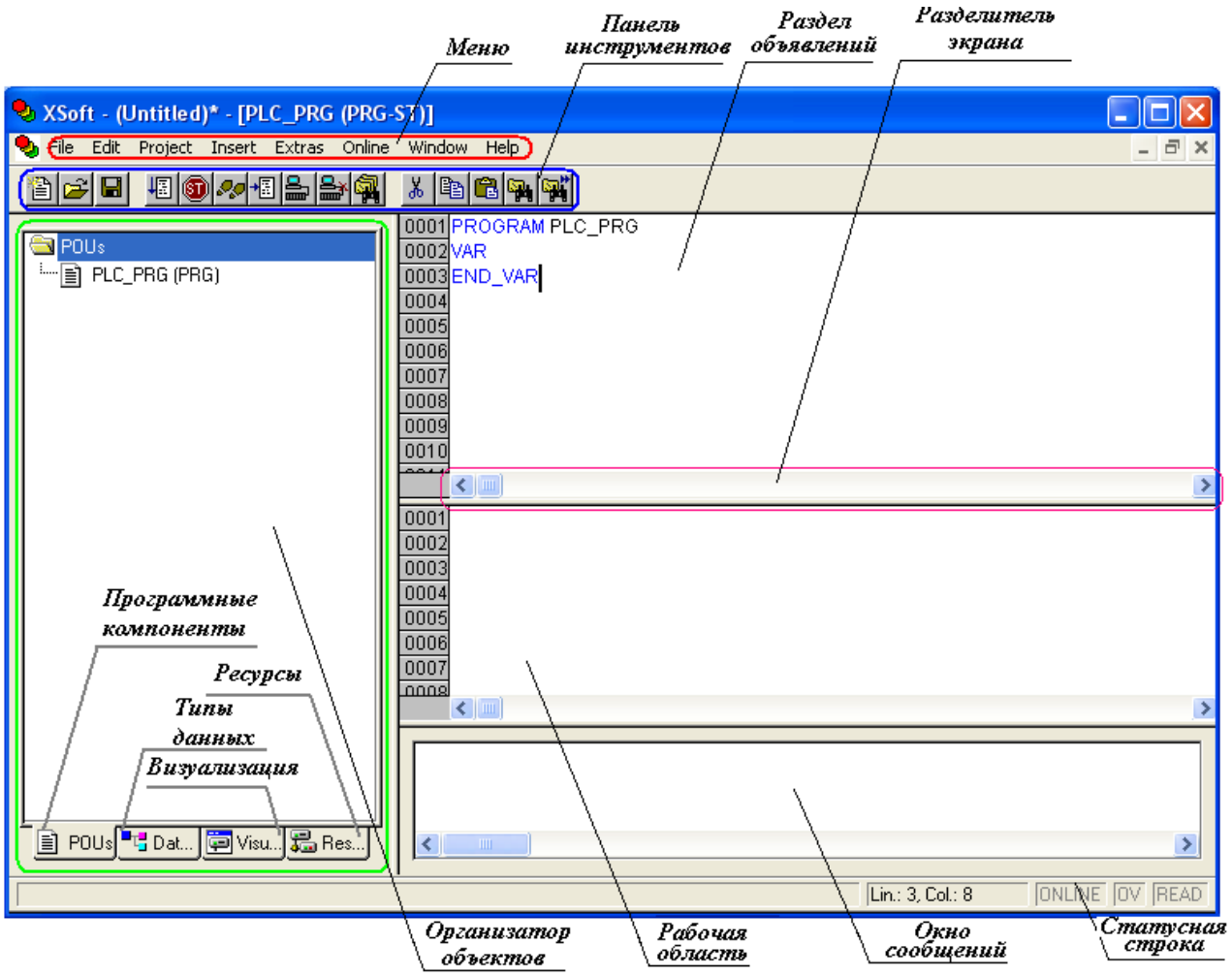
Функциональные блоковые диаграммы (FBD). Графический язык описания логических и аналоговых вычислений в очень простой и выразительной форме. CoDeSys автоматизирует составление FBD диаграмм, самостоятельно размещая программные компоненты и соединения.

Релейно-контактные схемы (LD). Графический язык, описывающий логику работы программы в форме соединения контактов и обмоток реле. Как и в FBD, редактор LD автоматически размещает и проводит соединения компонентов схемы.

Последовательные функциональные схемы (SFC). Графический язык, ориентированный на описание взаимосвязанных состояний и действий системы. CoDeSys поддерживает все без исключения типы действий, предусмотренные стандартом.

Непрерывные функциональные схемы (CFC). Редактор CFC аналогичен FBD, но в отличие от него не разделяет диаграмму на цепи, а оперирует со свободно размещаемыми компонентами. Диаграммы могут иметь обратные связи и настраиваемый порядок выполнения.

Знакомство с элементами главного окна (рис.3).



Р и с. 3. Элементы главного окна

Меню находится в верхней части главного окна. Оно содержит все команды CoDeSys.

Панель инструментов. Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню. Вызванная с помощью кнопки на панели инструментов команда автоматически выполняется в активном окне.

Кнопки на панели инструментов различны для разных редакторов CoDeSys. Получить информацию относительно назначения этих кнопок можно в описании редакторов.

Разделить экрана – это граница между двумя непересекающимися окнами. В CoDeSys есть следующие разделители: между организатором объектов и рабочей областью, между разделом объявлений и разделом кода POU, между рабочей областью и окном сообщений.

Вы можете перемещать разделители с помощью мышки, нажав ее левую кнопку. Разделитель сохраняет свое положение даже при изменении размеров окна.

Рабочая область находится в правой части главного окна CoDeSys. Все редакторы, а также менеджер библиотек открываются именно в этой области. Имя открытого объекта находится в заголовке окна. Именно в этой части главного окна вводится основной текст программы на языке, выбранном ранее.

Окно сообщений отделено от рабочей области разделителем. Именно в этом окне появляются сообщения компилятора, результаты поиска и список перекрестных ссылок. При двойном щелчке мышкой или при нажатии клавиши <Enter > на сообщении будет открыт объект, к которому относиться данное сообщение. С помощью команд “Edit” ”Next error” и ”Edit ” ” Previous error ” можно быстро перемещаться между сообщениями об ошибках.

Статусная строка находится в нижней части главного окна CoDeSys и предоставляет информацию о проекте и командах меню. При выборе пункта меню его описание появляется в левой части строки статуса. Если вы работаете в режиме online, то надпись Online в строке статуса выделяется черным цветом. В ином случае надпись серая.

С помощью статусной строки в режиме online можно определить, в каком состоянии находится программа: SIM – в режиме эмуляции, RUN – программа запущена, BP – установлена точка останова, FORCE – происходит фиксация переменных.

В визуализации в статусной строке выводятся координаты курсора X и Y, которые отсчитываются относительно верхнего левого угла окна. Если указатель мыши находится на элементе или над элементом производятся какие-либо действия, то указывается номер этого элемента. При вставке элемента в строке статуса указывается его название (например, Rectangle).

Контекстное меню. Быстрый вызов: <Shift>+<F10> Вместо того, чтобы использовать главное меню для вызова команд, можно вос-

пользоваться контекстным меню. Это меню, вызываемое правой кнопкой мыши, содержит наиболее часто используемые команды.

Объявление переменных

Раздел объявлений (рис.4). Локальные переменные POU объявляются в разделе объявлений редактора программного компонента. Такими переменными могут быть входные и выходные переменные, переменные, одновременно являющиеся входными и выходными, локальные переменные, сохраняемые переменные и константы.

Объявление переменных

Имена переменных не должны содержать пробелов и специальных символов, должны объявляться только один раз и не должны совпадать с зарезервированными словами. Регистр букв в имени переменной не имеет значения, т.е. переменные Var1, VAR1 и var1 не различаются. В именах переменных допустим знак подчеркивания. Переменные A_BCD и AB_CD считаются разными. Идентификатор не должен содержать подряд более одного символа подчеркивания. Длина идентификатора не ограничена, все символы являются значимыми.

Все переменные и типы данных можно инициализировать. Для этого используется оператор ":=". Переменные простейших типов инициализируются константами. По умолчанию все переменные инициализируются нулем.

Пример:

var1:INT:=12; (* Переменная типа INT, инициализируемая числом 12*)

Выделение цветом

Все редакторы выделяют переменные соответствующим цветом при их объявлении или использовании. Это позволяет быстро находить синтаксические ошибки в программе. Если вы случайно забыли закрыть комментарий или ошиблись, вводя зарезервированное слово, то вы сразу заметите это.

Используются следующие цвета:

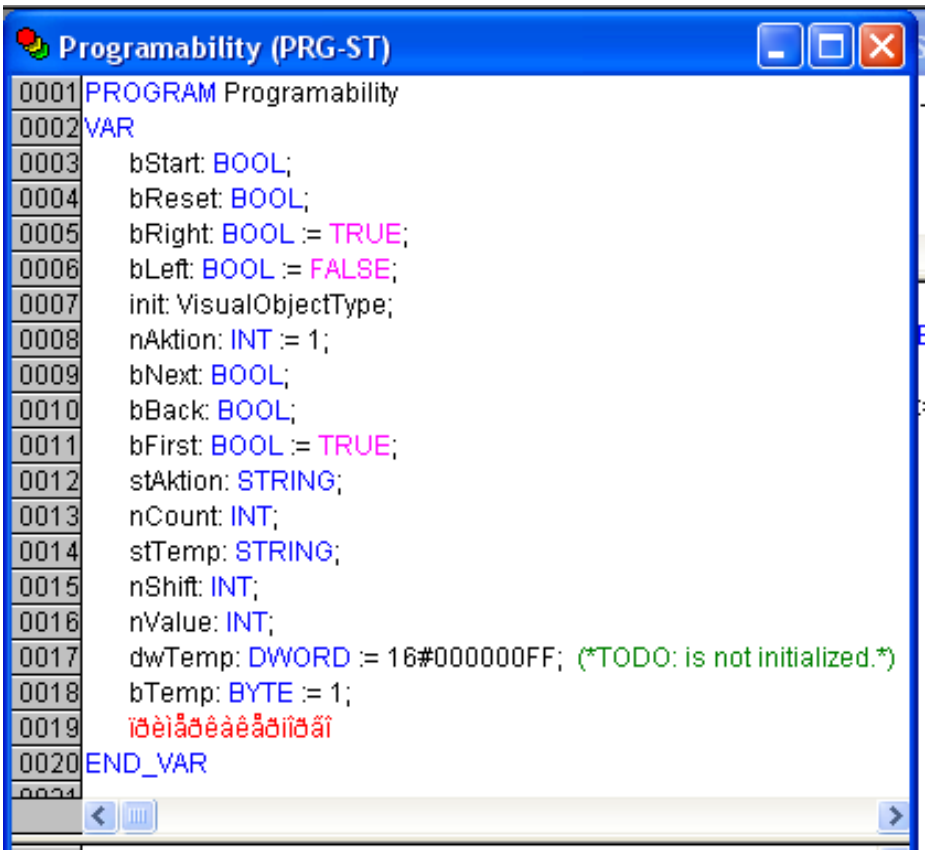
Синий – Ключевые слова

Зеленый – Комментарии в текстовых редакторах

Розовый – Специальные константы (например, TRUE/FALSE, T#3s,%IX0.0)

Красный – Ошибки ввода (например, неверные временные константы, ключевые слова, записанные строчными буквами)

Черный – Переменные, константы, операторы.



Р и с. 4. Раздел объявлений

Дополнительные возможности и ресурсы, используемые для создания проекта

Обзор ресурсов (рис.5)

Во вкладке Resources Организатора объектов находятся объекты, предназначенные для настройки и управления проектом и распределением переменных:

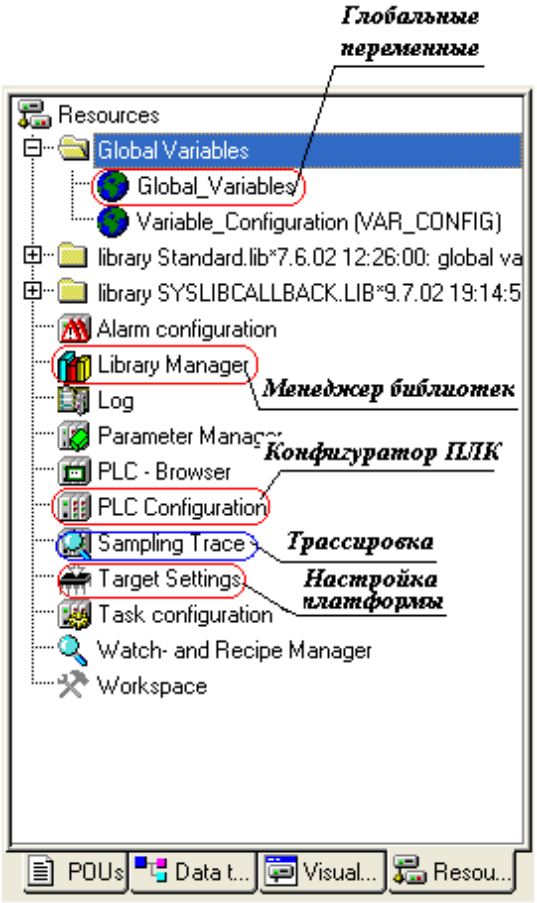
Global Variables – описания глобальных переменных, которые используются в проекте. Здесь же находятся глобальные переменные, описанные в библиотеках.

Library Manager – управление библиотеками, включенными в проект.

PLC Configuration – создание описания конфигурации аппаратных средств.

Target settings – выбор аппаратной платформы и настройка ее специфических параметров.

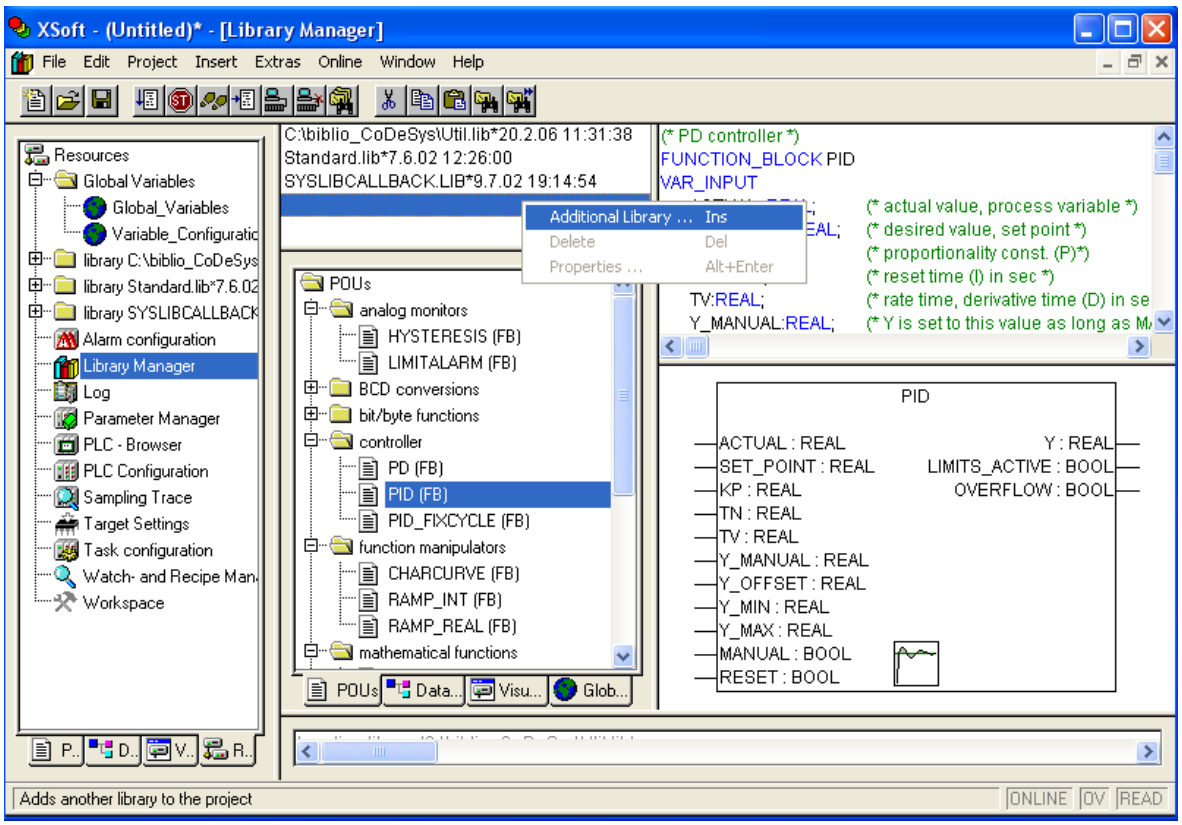
Sampling Trace – графическое осциллографирование значений переменных.



Р и с. 5. Обзор ресурсов

Менеджер библиотек (Library Manager)

Менеджер библиотек (рис.6) содержит список всех библиотек, которые связаны с проектом. POU, типы данных и глобальные переменные библиотек можно использовать так же, как и определенные пользователем POU типы данных и глобальные переменные.



Р и с. 6. Менеджер библиотек

Использование менеджера библиотек

Окно менеджера библиотек разделено на 3 или 4 области. Список библиотек, соединенных с проектом, находится в левой верхней области. Ниже, в зависимости от выбранной вкладки, показаны переменные POU, типы данных или глобальные переменные выделенной библиотеки.

Если выбрать POU, то в правой верхней части экрана появится раздел объявлений этого POU, а в нижней части – графическое изображение в форме блока с входами и выходами. При выборе типов данных и глобальных переменных в правой части окна выводится их объявление.

Стандартная библиотека

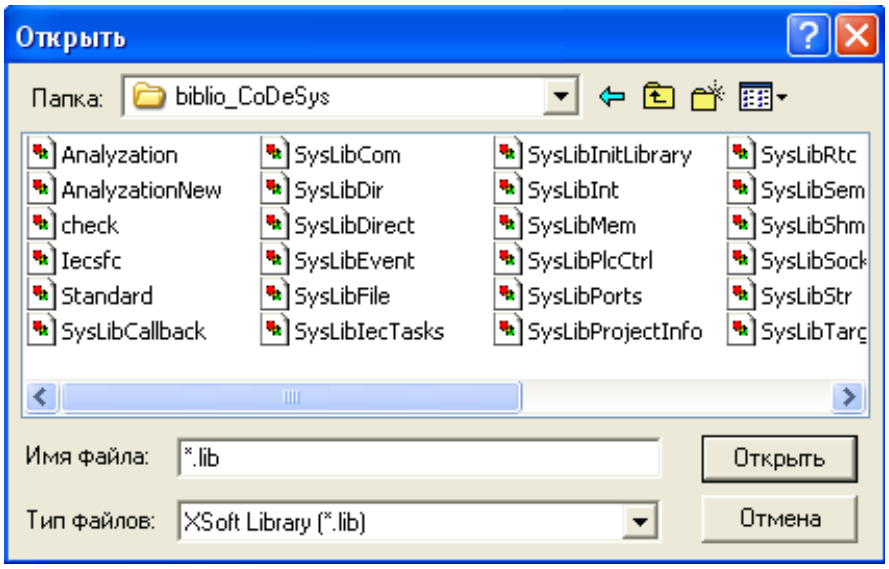
Библиотека "standard.lib" доступна всегда. Она содержит все функции и функциональные блоки, требуемые стандартом МЭК 61131-3. Разница между стандартными функциями и операторами заключается в том, что операторы признаются неявно системой про-

граммирования, а стандартные POU должны быть присоединены к проекту (standard.lib).

"Insert" "Additional Library"

Этой командой можно присоединять библиотеку к проекту. В открывшемся диалоговом окне выберите нужную библиотеку с расширением "*.lib". Название библиотеки появится в Менеджере библиотек, и ее объектами можно будет пользоваться как определенными пользователем объектами.

Пути поиска библиотек (рис. 7) зависят от состава директорий, определенных в опциях проекта. Если вы присоединяете библиотеку из другой директории, то библиотека будет добавлена в форме полного имени файла. Например: вы присоединяете библиотеку standard.lib из директории "D:\codesys\libraries\standard".



Р и с. 7. Путь поиска библиотек

Удаление библиотеки

Удаление библиотеки из проекта в Менеджере библиотек происходит по команде "Edit" "Delete".

Редактор визуализации в CoDeSys

Визуализация предназначена для графического представления объекта управления и непосредственно связана с созданной в CoDeSys программой контроллера. Редактор визуализации CoDeSys

предоставляет набор готовых графических элементов, которые могут быть связаны соответствующим образом с переменными проекта.

В Online режиме представление элементов на экране изменяется в зависимости от значений переменных.

Свойства отдельных элементов визуализации, а также визуализации в целом устанавливаются в соответствующих диалогах конфигурации и диалоге свойств объекта. Здесь определяется начальный вид элементов и выполняется привязка динамических свойств к значениям переменных проекта.

Дополнительные возможности открывает программируемость свойств элементов посредством специальных структурных переменных.

Используя заместители в диалогах конфигурации, вы можете облегчить себе работу и сэкономить время в случаях, когда созданный объект визуализации должен использоваться неоднократно. Например, при визуализации значений нескольких экземпляров одного функционального блока.

Обратите внимание также на возможность определять различные назначения клавиш для отдельных визуализаций.

Конфигурация недостающих модулей, входящих в состав ПЛК

Конфигуратор ПЛК (PLC Configuration)

Объект PLC Configuration (Конфигуратор ПЛК) расположен на вкладке ресурсов Организатора объектов. Конфигурация ПЛК определяет аппаратные средства вашей системы. Здесь задается распределение адресов входов/выходов контроллера, что определяет привязку проекта к аппаратным средствам. На основе описания конфигурации ПЛК CoDeSys проверяет правильность задания МЭК адресов, используемых в программах, на их соответствие фактически имеющимся аппаратным средствам.

Редактор конфигурации CoDeSys позволяет подключать удаленные модули ввода/вывода.

После окончания настройки двоичный образ конфигурации передается в ПЛК.

Конфигурация ПЛК отображается в редакторе в виде дерева. Для редактирования элементов применяются команды меню и специализированные диалоги. В конфигурации присутствуют элементы ввода и/или вывода, каждый из которых может содержать вложенные подэлементы.

Для входов и выходов могут быть назначены символические имена. Прямые МЭК адреса отображаются в конфигурации для каждого символического имени.

Компиляция проекта

Компилятор

Встроенный компилятор создает быстрый машинный код непосредственно из МЭК приложения. Помимо логических переменных, компилятор поддерживает: целые и битовые переменные, длительность, время дня и дату, вычисления в формате с плавающей запятой, строки, массивы, структуры и перечисления.

Компиляция проекта

Откомпилировать проект целиком можно, используя команду меню Project→Rebuilt all, либо клавишей «F11».

Ошибки и предупреждения компилятора

CoDeSys выводит результаты компиляции в специальном окне сообщений, расположенном в нижней части рабочего экрана.

Лучшим результатом является: 0 Error(s), 0 Warning(s). Если при компиляции проекта обнаружены ошибки или предупреждения, то перейдите в окно сообщений и найдите самое первое сообщение об ошибке. Щелчок мышки или нажатие Enter на тексте сообщения приведет к установке курсора на место в тексте вызвавшее проблему.

Каждая ошибка и предупреждение имеет уникальный номер. Клавиша <F1> в окне сообщений открывает соответствующее ошибке окно справочной системы.

Если в процессе ввода программы вам мешает окно сообщений, уберите его с помощью быстрой комбинации клавиш Shift-Esc. При компиляции окно сообщений будет открыто автоматически.

Сохранение проекта

После ввода конфигурации контроллера, а также после разработки программы рекомендуется сохранять результаты работы. Запись проекта на диск выполняется с помощью команды File \ Save as.

При первом сохранении проекта Concept просит ввести имя проекта. Имя проекта должно соответствовать требованиям MS-DOS: длина имени не должна превышать восьми символов, имя должно начинаться с буквы, использование пробелов и других специальных символов (*, ?, : и т.п.) недопустимо. Имя проекта также не должно совпадать с именами секций и переменных, используемых в проекте.

При записи проекта на диск, Concept создаёт несколько файлов с именем проекта и разными расширениями, поэтому рекомендуется для каждого проекта создавать отдельную папку.

Запуск проекта в режиме эмуляции

Режим эмуляции

Для того чтобы режим эмуляции был включен, необходимо выбрать в меню команду Online →Simulation.

При этом напротив команда Simulation в меню появляется галочка.

В режиме эмуляции программа выполняется в ПК. Этот режим используется для тестирования проекта и позволяет отладить прикладное ПО без подключения контроллера. Благодаря этому CoDeSys широко применяется при создании виртуальных лабораторий в вузах.

Состояние режима эмуляции (включен/выключен) сохраняется вместе с проектом.

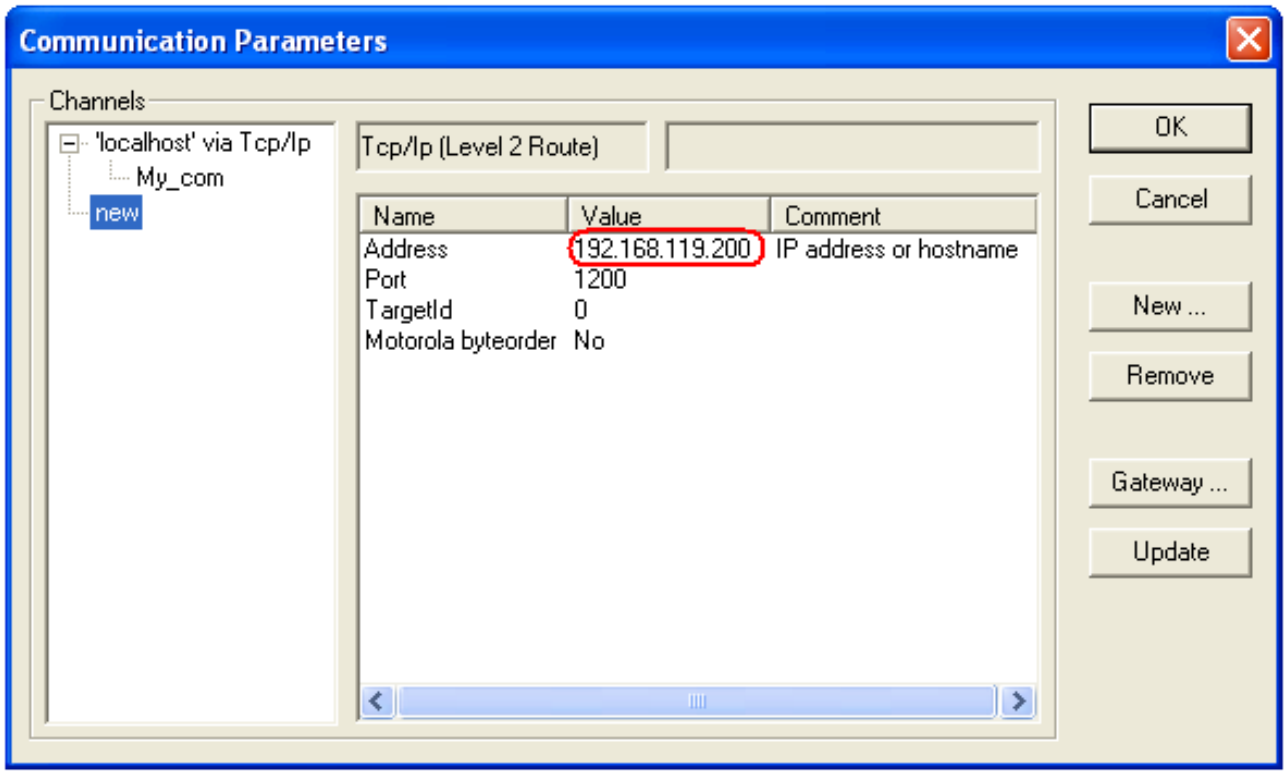
Если режим эмуляции выключен, то программа будет запущена в контроллере. Обмен данными между ПК и ПЛК обычно осуществляется по последовательному интерфейсу.

Настройка канала и соединения (рис. 9)

Если вы впервые подключаете контроллер к CoDeSys, необходимо выполнить определенные настройки.

В меню Online откройте диалог Communication Parameters. Нажмите клавишу New для настройки нового соединения.

Учитывая, что контроллер расположен на другой машине сети, необходимо изменить параметр ‘localhost’ на имя машины или задать соответствующий IP адрес (в нашем случае 192.168.119.200).

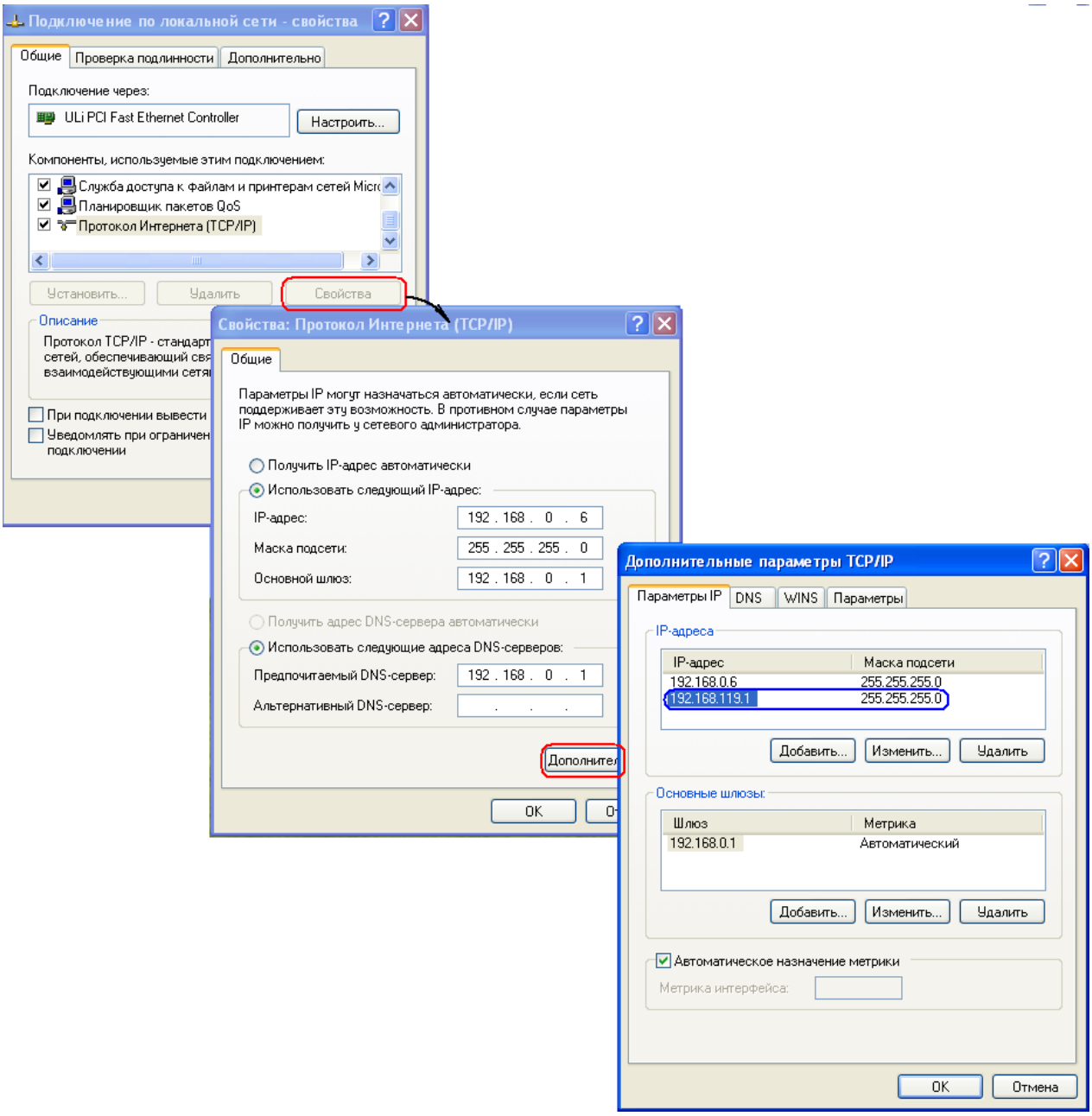


Р и с. 9. Установка соединения со средой программирования по Ethernet (начало)

Соединение с контроллером

При выборе в меню команды Online/Login ваш проект автоматически переходит в режим работы Стоп.

Если проект уже запущен, то для того чтобы перейти в режим Стоп, следует воспользоваться командой Online Stop или соответствующей иконкой на панели управления. Остановка программы выполняется не мгновенно по окончании рабочего цикла ПЛК.



Р и с. 9. Установка соединения со средой программирования по Ethernet (окончание)

Работа в режиме Стоп

В режиме стоп система исполнения CoDeSys опрашивает входы выходы контроллера и производит все обычные вспомогательные действия с единственным исключением – прикладная программа не выполняется.

Не смотря на то, что в режиме стоп программа ни чего не делает, это самый полезный режим при отладке. В режиме стоп все данные переменных «заморожены» и мы можем изучать их сколь угодно долго.

В реальном ПЛК мы имеем возможность доступа к значениям входов и выходов ПЛК. Это дает возможность проверить монтаж оборудования. Так для изучения работы датчиков мы можем формировать внешние воздействия и наблюдать результат в числовом формате или в виде графиков (графическая трассировка). Все это можно делать, «написав» и загрузив в ПЛК пустую программу, которая содержит только конфигурацию входов/выходов (для модульных систем).

Запуск проекта

Выполнение в реальном времени

Дайте команду Online Run (F5). Контроллер начнет выполнять программу так, как это будет происходить в реальных условиях. Отладчик позволяет просматривать и изменять значения переменных, как и в режиме стоп.

Здесь необходимо учитывать следующее: значения переменных считываются из памяти ПЛК с определенным интервалом асинхронно (обычно от 0.1 до 2 с, зависит от настройки системы), т.е. для быстрых процессов мы будем видеть меняющиеся значения, по последовательности изменения которых нельзя судить о последовательности работы программы.

Работа отладчика в CoDeSys практически не влияет на скорость выполнения программы. В 8 разрядных ПЛК (8051 22МГц) мониторинг 20 переменных типа INT увеличивает среднее время рабочего цикла контроллера менее чем на 2%. Работа SCADA системы скажется на ПЛК аналогичным образом. В 32 разрядных контроллерах с вытесняющей многозадачностью такое влияние отсутствует полностью (например, в CoDeSys SP RTE). При желании вы можете написать программу, измеряющую время рабочего цикла ПЛК (используйте функцию Time()) и с помощью графической трассировки детально изучить влияние различных факторов на работу вашего ПЛК.

При выполнении программы в эмуляторе контроллер тактируется по системному таймеру ПК, т.е. мы работаем с ПЛК имеющим время рабочего цикла около 60 мс. Режим эмуляции в CoDeSys включается двумя способами: 1) выбор None в качестве аппаратной платформы; 2) при работе с реальным ПЛК, в меню Online включите Simulation mode.

ТРЕБОВАНИЯ И РЕКОМЕНДАЦИИ ПО ОФОРМЛЕНИЮ ОТЧЁТА

Выполнение практических работ происходит в соответствии с порядком проведения работы. Номера вариантов соответствуют номерам машин или согласовываются с преподавателем. По результатам выполнения работы составляется отчет, оформленный в соответствии с требованиями.

Отчёт о проделанной работе должен содержать следующее.

1. Номер работы, номер варианта, номер задания, дату выполнения задания.
2. Сведения о лицах, выполнявших данную работу.
3. Название работы и цель работы.
4. Все выполненные при подготовке расчеты и таблицы, дополненные соответствующими пояснениями, выводами и комментариями, и отражающие порядок выполнения заданий.
5. Синтезированные функциональные схемы.
6. Функции выхода каждого из синтезированных устройств, заданные в табличной и аналитической форме.
7. Результаты исследования в виде описаний, таблиц, графиков, экспериментально снятых временных диаграмм и т.д. с соответствующими пояснениями.
8. Выводы по работе.

В отчёте должны быть отражены все основные этапы разработки, создания и испытания схемы, соответствующей выполняемому заданию, поэтому следует создавать отчёт параллельно с ходом выполнения задания, а на последнем этапе нужно лишь оформить отчёт.

Для получения зачёта по работе необходимо:

1) оформить и сдать правильно оформленный отчёт;

2) защитить отчёт. В ходе защиты студент должен рассказать об основных этапах выполнения задания и получаемых результатах. Студент должен ответить на любые вопросы преподавателя по подготовке и выполнению данного задания и на любые контрольные вопросы из выполненной работы.

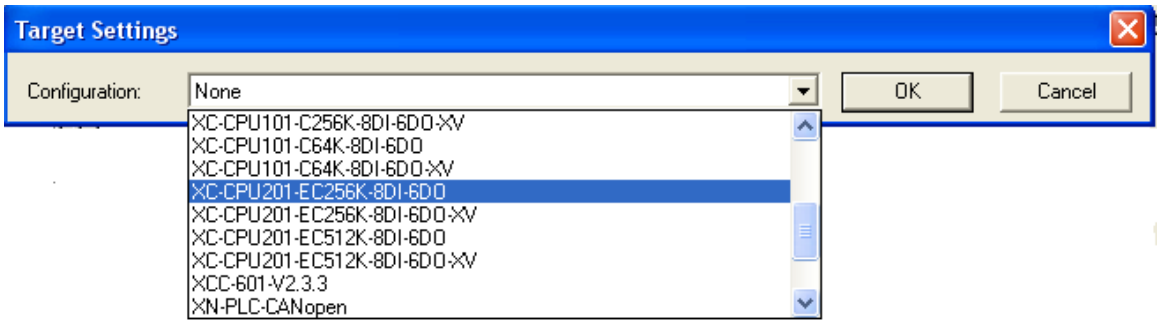
Практическая работа № 1
АППАРАТНАЯ КОНФИГУРАЦИЯ КОНТРОЛЛЕРА

Цель работы – знакомство с интерфейсом среды разработки приложений CoDeSys. Изучение возможностей и особенностей конфигуратора ПЛК. Провести аналогию между реальной аппаратной составляющей ПЛК ее моделью в конфигураторе CoDeSys.

Представление модульного контроллера в среде программирования CoDeSys

Конфигурация Центрального процессора (CPU) ПЛК

При создании нового проекта автоматически открывается окно Target Settings (Конфигурирование ПЛК). В данном окне (рис.1.1) нужно определить конфигурацию центрального процессора ПЛК в соответствии с аппаратными средствами своего контроллера.



Р и с. 1.1. Окно конфигурирования ПЛК

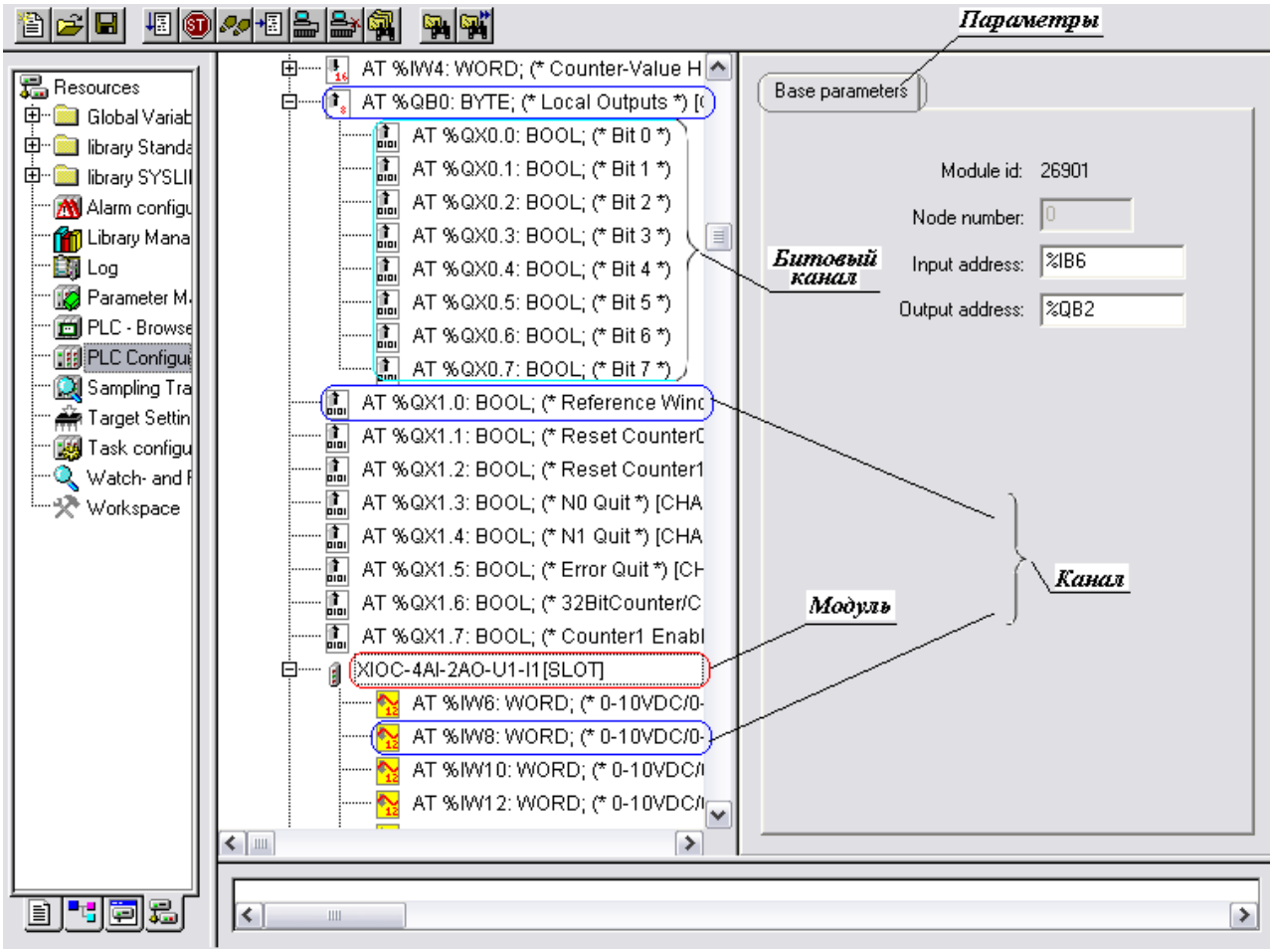
Дальнейшие изменения в конфигурацию ПЛК вносятся непосредственно в PLC Configuration (конфигуратор ПЛК), расположенном во вкладке ресурсов Организатора объектов.

Конфигуратор ПЛК (PLC Configuration)

Конфигурация ПЛК (рис. 1.2) определяет аппаратные средства вашей системы. Здесь задается распределение адресов входов/выходов контроллера, что определяет привязку проекта к аппаратным средствам. На основе описания конфигурации ПЛК CoDeSys проверяет правильность задания МЭК адресов, используемых в программах, на их соответствие фактически имеющимся аппаратным средствам.

Редактор конфигурации CoDeSys позволяет подключать удаленные модули ввода/вывода.

После окончания настройки двоичный образ конфигурации передается в ПЛК.



Р и с. 1.2. Окно конфигуратора ПЛК

Конфигурация ПЛК отображается в редакторе в виде дерева. Для редактирования элементов применяются команды меню и специализированные диалоги. В конфигурации присутствуют элементы ввода и/или вывода, каждый из которых может содержать вложенные подэлементы.

Базовые термины

Конфигуратор ПЛК (PLC Configuration): редактор CoDeSys, в котором определяется состав аппаратных средств и производится настройка определенных параметров ввода-вывода.

Модуль: независимая единица аппаратных средств. Модуль включает набор каналов ввода-вывода. Как и каждый отдельный ка-

нал, модуль может иметь параметры. Каждый тип модуля имеет уникальный идентификатор.

Канал – это собственно данные ввода-вывода. Как правило, модуль имеет фиксированный набор каналов или подмодулей. Каждый канал имеет определенный МЭК тип и адрес. Естественно, для каждого канала выделяется определенное пространство памяти. Каждый канал имеет уникальный в пределах данной конфигурации ПЛК идентификатор.

Битовый канал – идентификатор отдельного бита в многобитном канале.

Плоская модель адресации – модель определения МЭК адресов, без спецификации иерархии модулей.

Все адресное пространство ввода-вывода представляется в виде плоского набора последовательно пронумерованных ячеек памяти. Если включена опция 'Calculate addresses', то при изменении положения модуля адреса его каналов соответствующим образом смещаются. Альтернативой может служить фиксированная адресация. В этом случае для каждого модуля отводится фиксированное адресное окно, которое определяется физическим расположением (номером слота) модуля. Например: %QB0, %IB26, %MW4.

Параметр – атрибут канала или модуля. Значение параметра устанавливается интерактивно до компиляции проекта. Оно передается в ПЛК и влияет на работу аппаратуры.

Работа в редакторе конфигуратора ПЛК

Окно редактора конфигуратора ПЛК разделено на две части. В левой части окна показано дерево конфигурации. Структура и компоненты дерева определяются главным образом файлом конфигурации, но могут быть изменены пользователем CoDeSys. В правом окне показаны доступные в настоящее время диалоги конфигурации в виде одной или нескольких табличных вкладок.

Правая часть окна видна по умолчанию, но может быть скрыта через меню 'Extras' 'Properties'. Верхушка конфигурационного дерева

начинается с корневого элемента, имя которого определено в файле конфигурации.

Ниже вы увидите другие элементы конфигурации: модули различного типа.

Выбор элементов

Для выбора элемента щелкните мышкой по его наименованию либо перемещайтесь по элементам с помощью стрелок на клавиатуре. Выбранный элемент обведен прямоугольником из точек.

Элементы, перед которыми стоит значок "плюс", раскрываются на подэлементы. Для развертывания элемента выберите его и щелкните по нему дважды мышкой или нажмите <Enter>. Для свертывания раскрытого элемента (на месте теперь присутствует знак "минус"), нужно выполнить аналогичные действия.

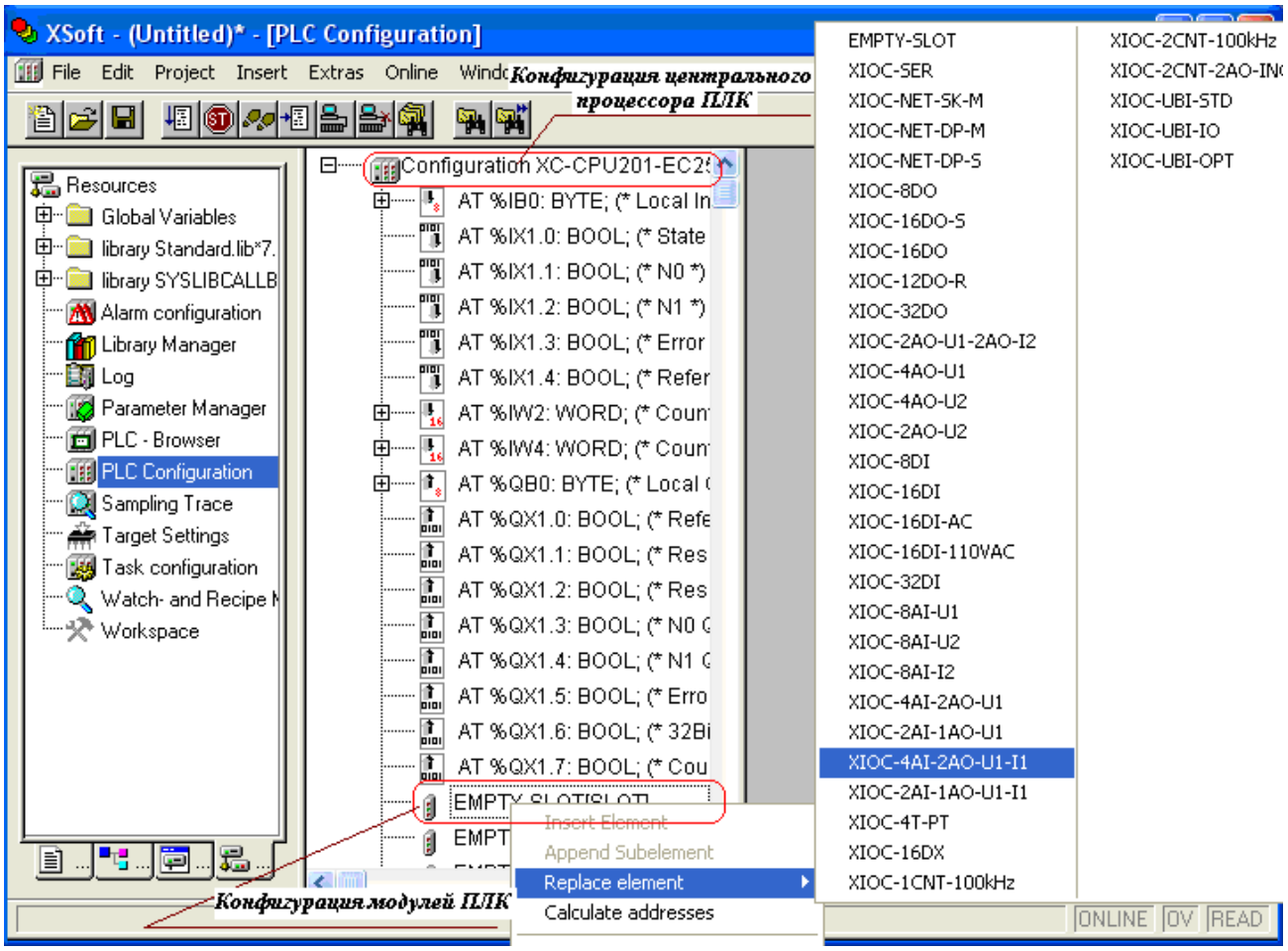
Добавление элементов, 'Insert' 'Insert element', 'Insert' 'Append subelement'

В соответствии с определениями в файле (файлах) конфигурации и файлах описания устройств, считанных, когда проект был открыт, основной состав элементов автоматически помещается в дерево конфигураций. Но если файл конфигурации позволяет, то некоторые дополнительные элементы могут быть добавлены. Для них должны присутствовать файлы описания.

- Команда 'Insert' 'Insert element' вставляет новый элемент перед элементом, выбранным в дереве конфигурации.
- Команда 'Insert' 'Append subelement' добавляет новый подэлемент к выбранному в дереве конфигурации элементу. Подэлемент помещается в последнюю позицию.

Замена/переключение элементов, 'Extras' 'Replace element''

В зависимости от определений в файле конфигурации (рис.1.3) выделенный элемент можно заменить на другой. Аналогичным образом можно переключать каналы элементов на ввод или вывод. Используйте команду 'Extras' 'Replace element'.



Р и с. 1.3. Конфигурация модулей ПЛК

Адресация входов и выходов контроллера

После расстановки всех модулей по местам, необходимо указать, на какие адреса памяти контроллера будут отображаться каналы модулей ввода-вывода. Назначение адресов памяти каналам модулей ввода-вывода позволяет программе контроллера обращаться считывать значения с входов контроллера и устанавливать нужные значения на его выходах. При выполнении программы, центральный процессор контроллера обращается к памяти как к переменным и использует их в своих вычислениях. В состав контроллера также входит специальное устройство, которое пересылает значения из модулей ввода в память контроллера и из памяти контроллера в модули вывода.

Для контроллеров компании Moeller определены четыре типа адресного пространства, на которые отображаются входные и выходные каналы модулей ввода-вывода контроллера (табл. 1.1).

Адреса каналов модулей задаются в окне конфигурации контроллера. Между адресами каналов одного модуля не может быть разрывов, поэтому разработчик указывает только адрес первого канала. Адреса остальных каналов модуля распределяются автоматически.

Адрес канала состоит из префикса, определяющего область памяти и собственно адреса. Префикс следует указывать в соответствии с типом каналов модуля ввода-вывода (табл. 1.1). Адрес определяет разработчик.

Если нет каких-то особых причин, адреса назначаются последовательно начиная с единицы. Например, для первого модуля дискретного ввода задаётся адрес %IX0.0, программа определяет, что последний адрес для этого модуля – %IX0.7. Тогда для следующего модуля дискретного модуля задаётся адрес %XI1.0 и т.д.

Главное требование при распределении адресов – адреса отображения каналов разных модулей одного типа не должны пересекаться, т.е. на один и тот же адрес не может быть отображено больше одного канала.

Таблица 1.1

Отображение входных и выходных каналов модулей ввода-вывода ПЛК на адресном пространстве

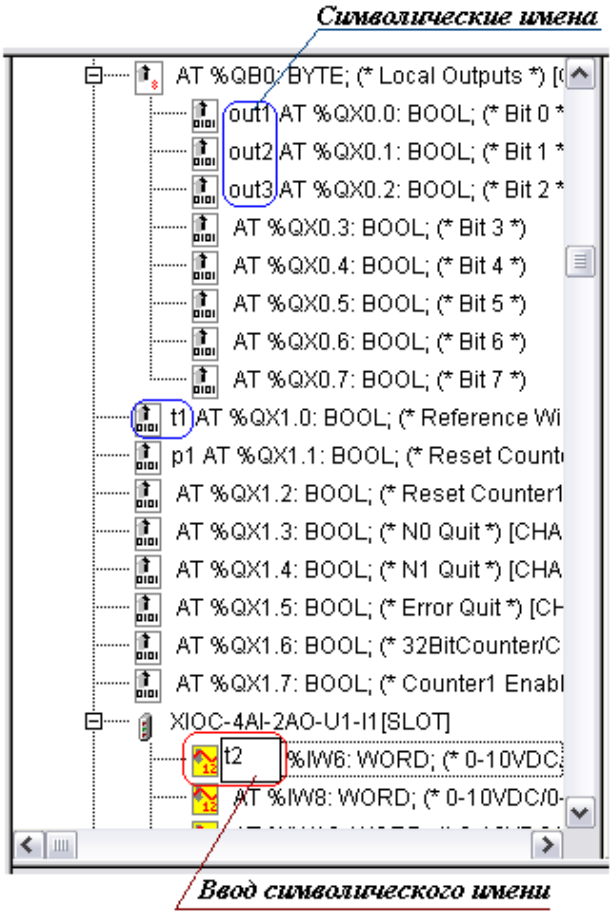
Тип канала	Область памяти	
	Префикс	Пример
Дискретный вход	IX	%IX1.2
Дискретный выход	QX	%QX0.6
Аналоговый вход	IW	%IW6
Аналоговый выход	QW	%QW4

Символические имена

Символические имена для модулей и каналов могут быть заданы в конфигурационном файле. В этом случае они будут отражаться в редакторе конфигурации перед определением прямого МЭК адреса (AT). В конфигурационном файле также определена возможность редактирования и ввода символических имен в редакторе конфигурации ПЛК.

Для ввода символического имени (рис.1.4) выберите необходимый модуль или канал в дереве конфигурации и щелкните мышкой

по текстовому полю перед префиксом прямого МЭК адреса 'AT'. Аналогично вы можете изменить существующее символическое имя двойным щелчком мыши. Символические имена должны удовлетворять общим правилам создания идентификаторов.

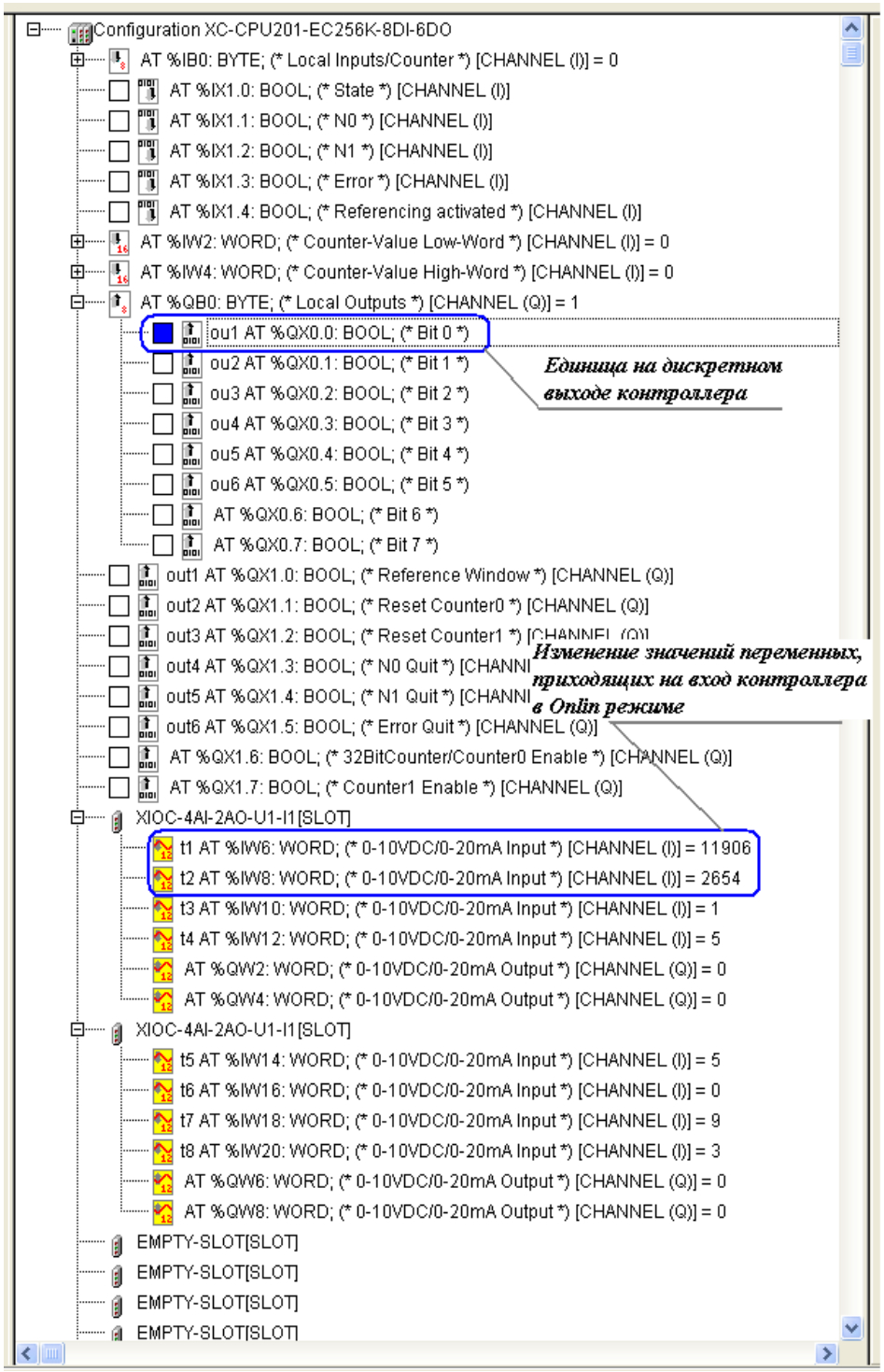


Р и с. 1.4. Ввод символического имени

Конфигурация ПЛК в режиме Online

В режиме Online конфигурация ПЛК (рис.1.5) отображает состояние входов и выходов ПЛК. Если логический вход или выход имеет значение TRUE, то перед его именем в дереве конфигурации отображается маленький прямоугольник, закрашенный голубым цветом. Для других типов переменных в конце строки отображается значение переменной.

Значение логического входа можно изменить щелчком мыши. Для других типов входов щелчок мыши открывает диалог изменения значения. Новое значение записывается в ПЛК сразу же по нажатию кнопки OK.



Р и с. 1.5. Конфигурация ПЛК в Online режиме

Задание для самостоятельного решения

1. В программе CoDeSys нужно создать новый проект (тип POU – программа, язык программирования – CFC), в котором сконфигурировать промышленный контроллер, состоящий из следующих модулей (табл. 1.2).

Таблица 1.2

Конфигурация контроллера согласно номеру варианта

	Номер варианта			
Набор модулей	1	2	3	4
1	Модуль центрального процессора – XC-CPU101-C128K-8DI-6DO	Модуль центрального процессора – XC-CPU201-EC512K-8DI-6DO	Модуль центрального процессора – XC-CPU201-C256K-8DI-6DO	Модуль центрального процессора – XC-CPU201-EC512K-8DI-6DO-XV
2	Модуль аналогового ввода – XIOC-8AI-I2	Модули дискретного вывода – XIOC-16DO	Модуль аналогового ввода – XIOC-4T-PT	Модуль аналогового ввода-вывода – XIOC-2AI-1AO-U1
3	Модули дискретного ввода – XIOC-32DI	Модуль – XIOC-NET-SK-M	Модули дискретного ввода – XIOC-32DI	Модули аналогового вывода – XIOC-2AO-U1-2AO-I2
4	Сигнальный модуль – XIOC-SER	Модули дискретного ввода – XIOC-16DI	Модуль – XIOC-1CNT-100kHz	Модули дискретного ввода-вывода – XIOC-16DX

2. Откомпилировать проект и запустить в режиме эмуляции.

3. Создать новый проект, задав конфигурацию таким образом, чтобы она отображала аппаратные средства нашей системы:

- модуль центрального процессора;
- модули аналогового ввода-вывода.

4. Дать символические имена каналам (с 1 по 4) первого модуля аналогового ввода-вывода.

5. Настроить канал и соединение. Запустить проект в реальном режиме времени (Online).

6. Пронаблюдать за тем, как изменяется значение сигнала на аналоговых входах. Сделать выводы.

СОДЕРЖАНИЕ ОТЧЕТА

1. Ответы на контрольные вопросы.
2. Изображение дерева конфигурации (у каждого варианта свое) в Onlin режиме (подписать все модули, входящие в заданную конфигурацию ПЛК, проанализировать особенности нумерации входов/выходов как аналоговых, так и дискретных).
3. Перечень аппаратных средств, входящих в нашу систему.
4. Изображение дерева конфигурации, соответствующее аппаратным средствам нашей системы.
5. Изображение той же конфигурации, но в реальном режиме времени (после настройки канала и соединения с ПЛК).
6. Анализ изменения сигнала на аналоговых входах ПЛК.

Практическая работа № 2
РЕДАКТОР ВИЗУАЛИЗАЦИИ. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ ОБЪЕКТА УПРАВЛЕНИЯ

Цель работы – изучить возможности редактора визуализации. Смоделировать основные приборы, входящие в наш контур.

Визуализация – краткое введение

Визуализация предназначена для графического представления объекта управления и непосредственно связана с созданной в CoDeSys программой контроллера. Редактор визуализации CoDeSys предоставляет набор готовых графических элементов, которые могут быть связаны соответствующим образом с переменными проекта.

В Online режиме представление элементов на экране изменяется в зависимости от значений переменных.

Простой пример: допустим, уровень заполнения емкости жидкостью, доступен в программе в виде значения некоторой переменной. Мы можем изобразить графический элемент, в виде полосы, которая в зависимости от значения переменной проекта будет изменять свою длину и/или цвет. Рядом мы можем разместить текст, отображающий в виде числа текущий результат измерения и, например, кнопки запуска и остановки программы.

Свойства отдельных элементов визуализации, а также визуализации в целом устанавливаются в соответствующих диалогах конфигурации и диалоге свойств объекта. Здесь определяется начальный вид элементов и выполняется привязка динамических свойств к значениям переменных проекта.

Дополнительные возможности открывает программируемость свойств элементов посредством специальных структурных переменных.

Используя заместители в диалогах конфигурации, вы можете облегчить себе работу и сэкономить время, в случаях, когда созданный объект визуализации должен использоваться неоднократно. Например, при визуализации значений нескольких экземпляров одного функционального блока.

Вполне возможно, что созданная в системе программирования визуализация будет играть роль единственного пользовательского интерфейса для контроля и управления работой ПЛК программы в рабочем режиме. В этом случае ввод данных для программы должен выполняться исключительно посредством элементов визуализации. Такую возможность обеспечивают специальные возможности ввода, задаваемые в процессе конфигурации. Кроме того, предусмотрено создание клавиш быстрого ввода для каждой конкретной визуализации.

Созданная в CoDeSys визуализация может использоваться еще несколькими способами.

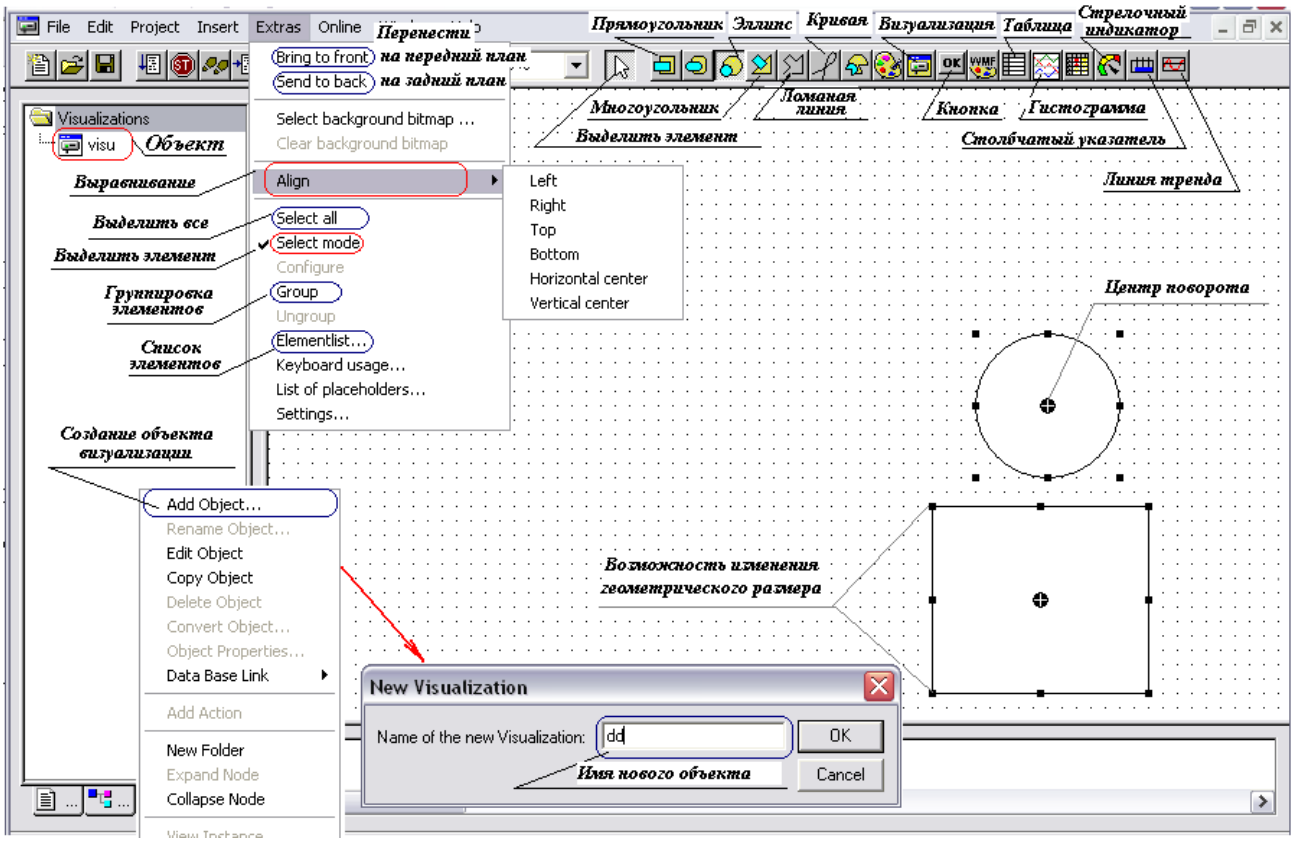
- Программа Win32 CoDeSys HMI отображает формы визуализации на ПК в полноэкранном режиме.
- Web-визуализация отображает данные и предоставляет возможность удаленного управления через Интернет.
- Для контроллеров со встроенным дисплеем доступна целевая визуализация.

Создание объекта визуализации

Объект «визуализация» – это инструмент CoDeSys, расположенный в Организаторе проекта. Он содержит представление и свойства отдельных элементов визуализации проекта. Как и любой другой объект CoDeSys, он имеет определенный набор общих свойств. Один или несколько объектов визуализации могут быть созданы в CoDeSys проекте и связаны друг с другом.

Вставка элементов визуализации

Элемент визуализации – это графический элемент, который используется при построении объекта визуализации. Возможные элементы представлены в виде иконок на панели инструментов CoDeSys. Каждый элемент имеет собственную конфигурацию (набор свойств) (рис. 2.1).



Р и с. 2.1. Окно редактора визуализации. Основные элементы, входящие в его состав

Конфигурирование визуализации

Конфигурироваться могут как отдельные графические элементы визуализации, так и объект визуализация в целом. В зависимости от выбранного элемента в вашем распоряжении оказываются различные диалоги конфигурации. Для вызова соответствующего диалога используйте команду 'Configure' ('Конфигурирование') из меню 'Extras' ('Дополнительно') или аналогичную команду из контекстного меню.

В диалогах конфигурации определяются статические настройки, либо указываются переменные проекта, значения которых определяют соответствующие динамические свойства в режиме online. Кроме

того, свойства элементов визуализации доступны программно через специальную структуру, связанную с каждым элементом.

Конфигурирование элементов визуализации

Техника конфигурирования элемента ('Extras' 'Configure').

Команда меню 'Extras' 'Configure' раскрывает диалог 'Configure element' ('Конфигурирование элемента') конфигурирования выбранного элемента визуализации. Этот же диалог открывается двойным щелчком мыши на элементе.

В левой части диалогового окна представлен список категорий, каждая из которых объединяет некоторый логически взаимосвязанный набор свойств визуализации элемента. Для различных типов элементов доступны разные наборы категории конфигурации.

Ниже приведен и подробно рассмотрен список категорий характерный для элементов:

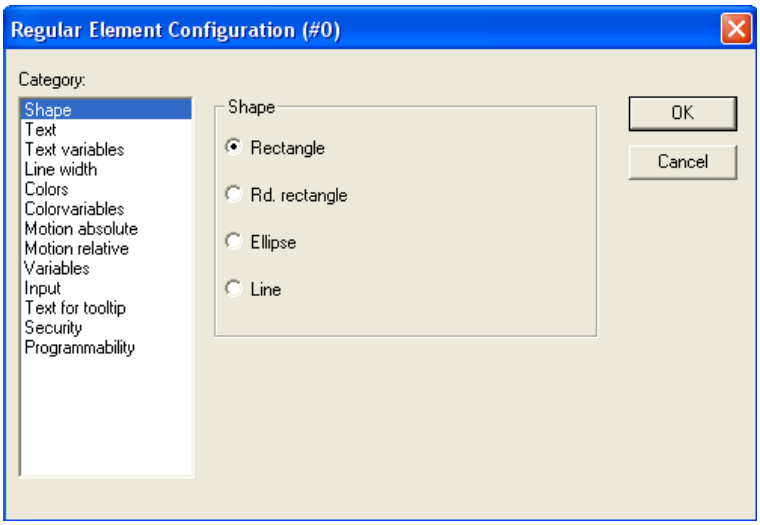
- прямоугольник;
- эллипс;
- линия;
- закругленный прямоугольник.

Shape

В категории Shape (Форма) вы можете изменить форму отображения некоторых элементов: Rectangle (Прямоугольник), Rounded Rectangle (Закругленный прямоугольник), Ellipse (Эллипс), Line (Линия). Форма элемента изменяется в пределах установленных размеров области отображения элемента (рис. 2.2).

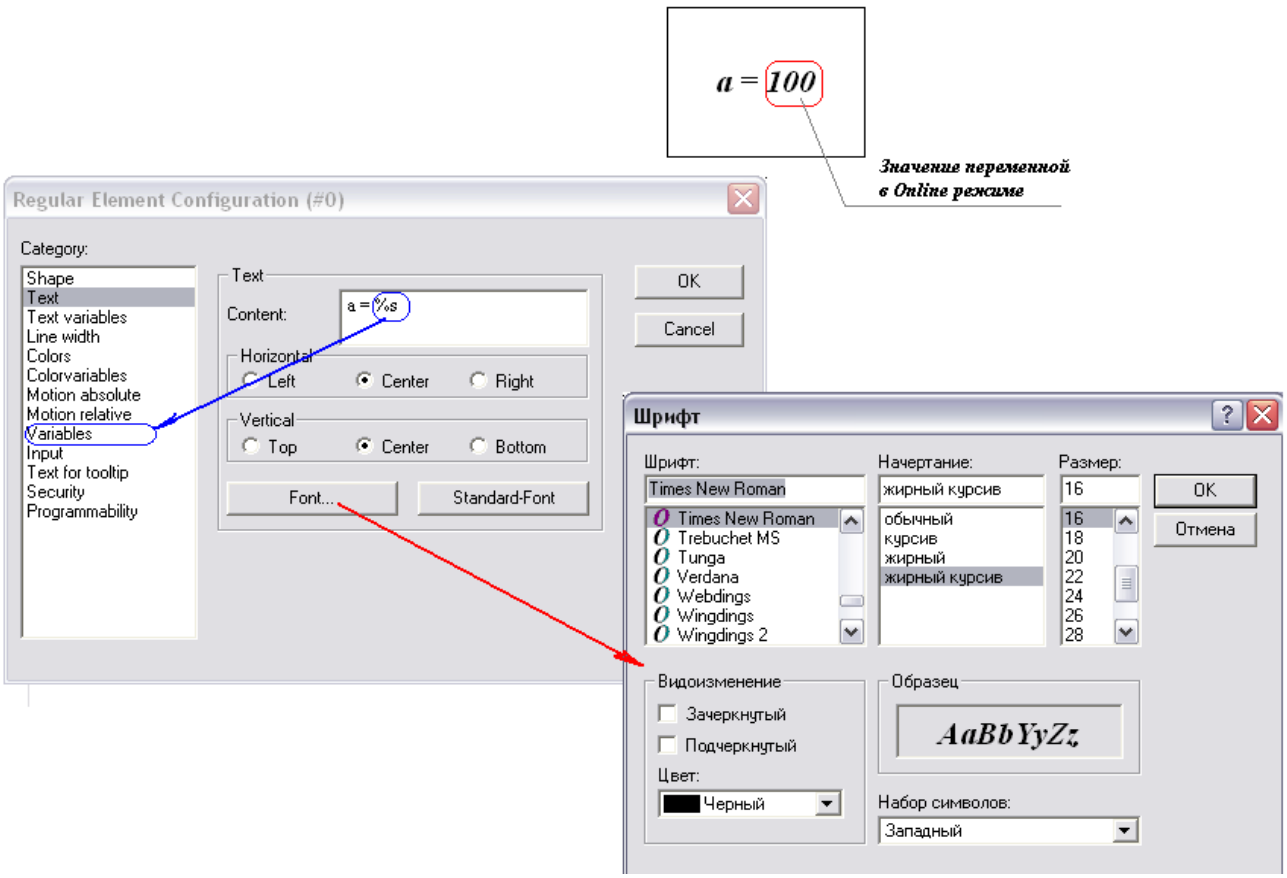
Text

В диалоге конфигурирования элементов визуализации вы можете определить текст элемента в категории Text. Он может быть введен непосредственно, или может быть установлена переменная, которая определяет его динамически. Здесь же задаются основные установки для шрифта и выравнивания.

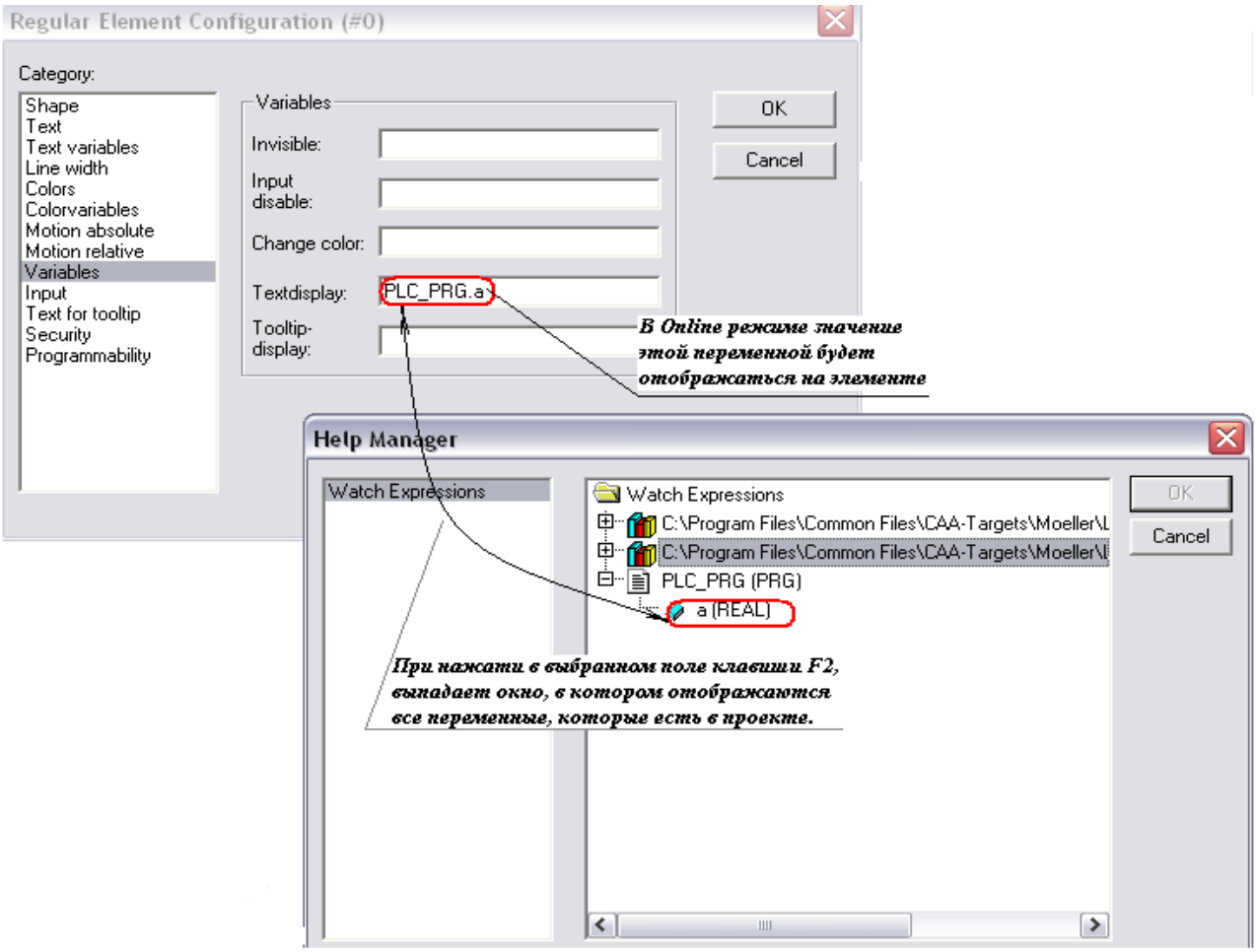


Р и с. 2.2. Окно категории Shape

Введите текст в поле Content (Содержание). Комбинацией клавиш <Crtl>+<Enter> вы можете вставлять перевод строки, <Crtl>+<Tab> – позиции табуляции. В дополнение к вводу чисто текстовых данных вы имеете следующие возможности форматирования (рис.2.3).



Р и с. 2.3. Пошаговое выполнение действий, для того чтобы в Online режиме значение переменной отображалось на элементе (начало)



Р и с. 2.3. Пошаговое выполнение действий, для того чтобы в Online режиме значение переменной отображалось на элементе (окончание)

Если вы вводите в тексте %s, то сюда в режиме online подставляется значение переменной из поля 'Text display' категории ‘Variables’ ('Переменные'). Вы можете использовать форматирование данных, соответствующее функции 'sprintf' стандартной С библиотеки.

Допустимые символы форматирования приведены в табл. 2.1.

Таблица 2.1

Символы форматирования данных

Символы	Параметр/ формат вывода
d, i	Десятичное число
o	Восьмеричное число без знака (без нуля в начале)
x	Шестнадцатеричное число без знака (без 0x в начале)
u	Десятичное число без знака
c	Отдельный символ
s	Строка символов
f	Действительное [-] m.dddddd, причем точность устанавливается

	количеством цифр d (по умолчанию 6)

Соответствующее значение переменной будет отображено в online режиме.

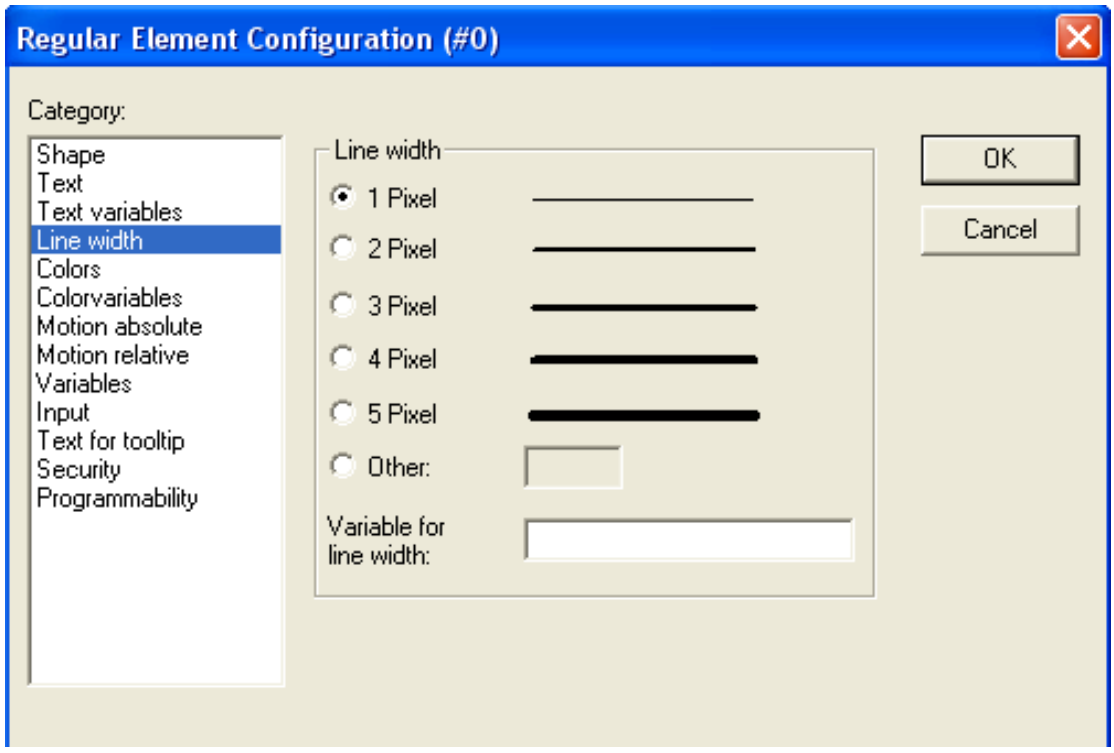
Ввод в поле 'Content' %t, сопровождаемый определенной последовательностью специальных указателей формата, заменяется в online режиме значением системного времени.

Вид текста определяется способом выравнивания: horizontally left (по горизонтали слева), center (по центру) или right (справа) и vertically top (по вертикали сверху), center (центру) или bottom (снизу).

Кнопка Font (шрифт) открывает диалог выбора шрифта. Список шрифтов может быть ограничен в зависимости от выбранной целевой платформы. Выберите желаемый тип шрифта и подтвердите выбор кнопкой OK. Кнопка Standard-Font (Стандартный шрифт) устанавливает тип шрифта, который выбран в опциях проекта ('Project' 'Options' 'Editor'). При его изменении он будет применяться во всех элементах, кроме элементов, для которых шрифт задан явно (с помощью кнопки Font).

Толщина линий (Line width)

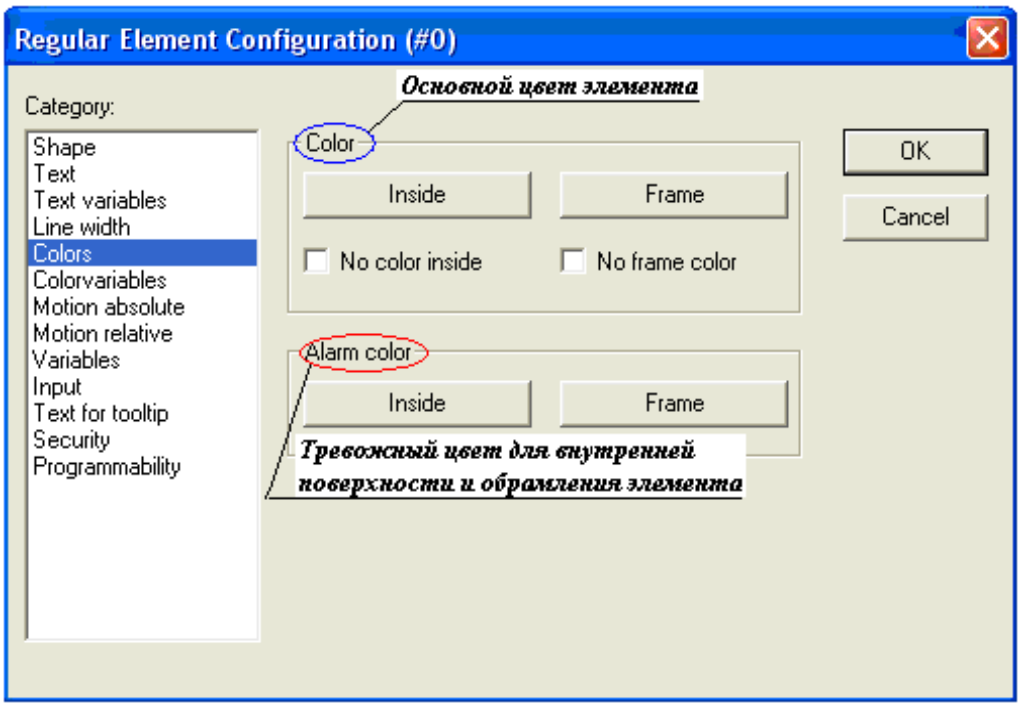
В категории Line width (рис.2.4) вы можете выбирать толщину прорисовки линий элемента. В селекторе выбора заданы толщины от 1 до 5 пикселей. Вы можете задать иное значение вручную в поле Other.



Р и с. 2.4. Окно категории Line width

Цвет (Color)

В категории Color (Цвет) (рис. 2.5) вы можете выбирать основные и тревожные цвета для внутренней поверхности и обрамления вашего элемента. Опции no color inside (нет цвета внутри) и no frame color (нет цвета обрамления) позволяют создание прозрачных элементов.



Р и с. 2.5. Окно категории Color

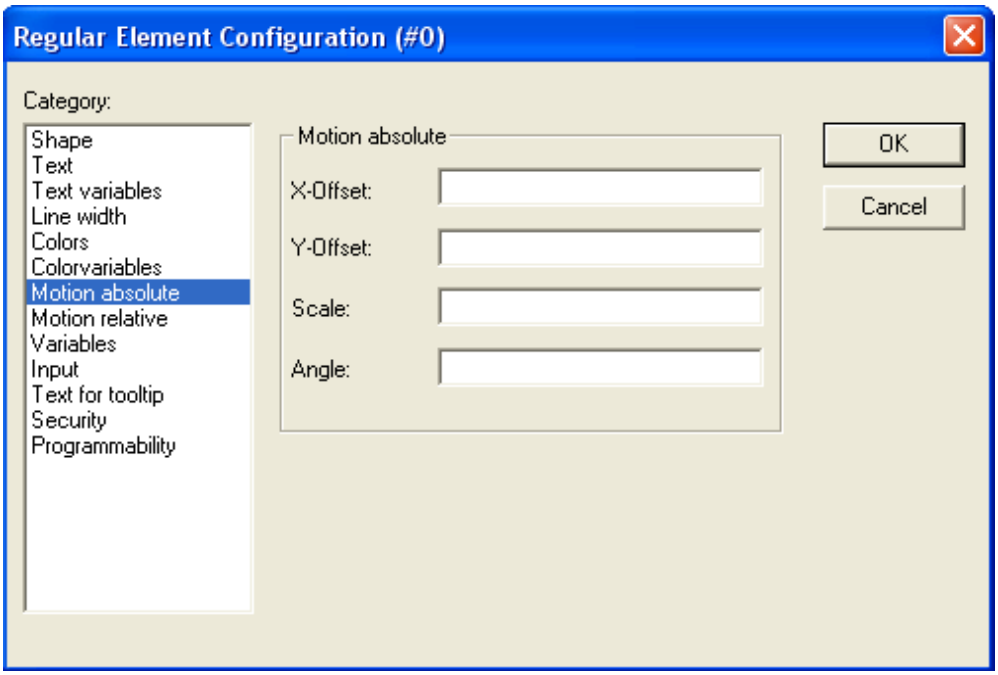
Абсолютное перемещение (Motion absolute)

В категории Motion absolute (Абсолютное перемещение) присутствуют поля X– и Y-Offset (рис. 2.6). В них можно определить переменные, которые задают динамическое смещение элемента по X и Y. Значение переменной в поле Scale (Масштабирование) изменяет линейные размеры элемента. Текущее значение переменной служит масштабным коэффициентом и делится неявно на 1000. Благодаря чему для уменьшения размеров элемента можно применять целочисленные переменные. Изменение размера элемента происходит относительно центра поворота элемента.

Переменная в поле Angle вызывает поворот элемента вокруг его центра поворота. Положительное значение переменной соответствует

направлению по часовой стрелке. Значение угла определяется в градусах. У многоугольников вращается каждая точка, что означает вращение всего многоугольника. Другие элементы вращаются так, чтобы верхняя грань всегда оставалась наверху.

Центр поворота появляется после однократного щелчка мышкой по изображению элемента и представляется как маленький черный круг с белой крестовиной (⊕). Мышью с нажатой левой кнопкой вы можете перемещать центр поворота.



Р и с. 2.6. Окно категории Motion absolute

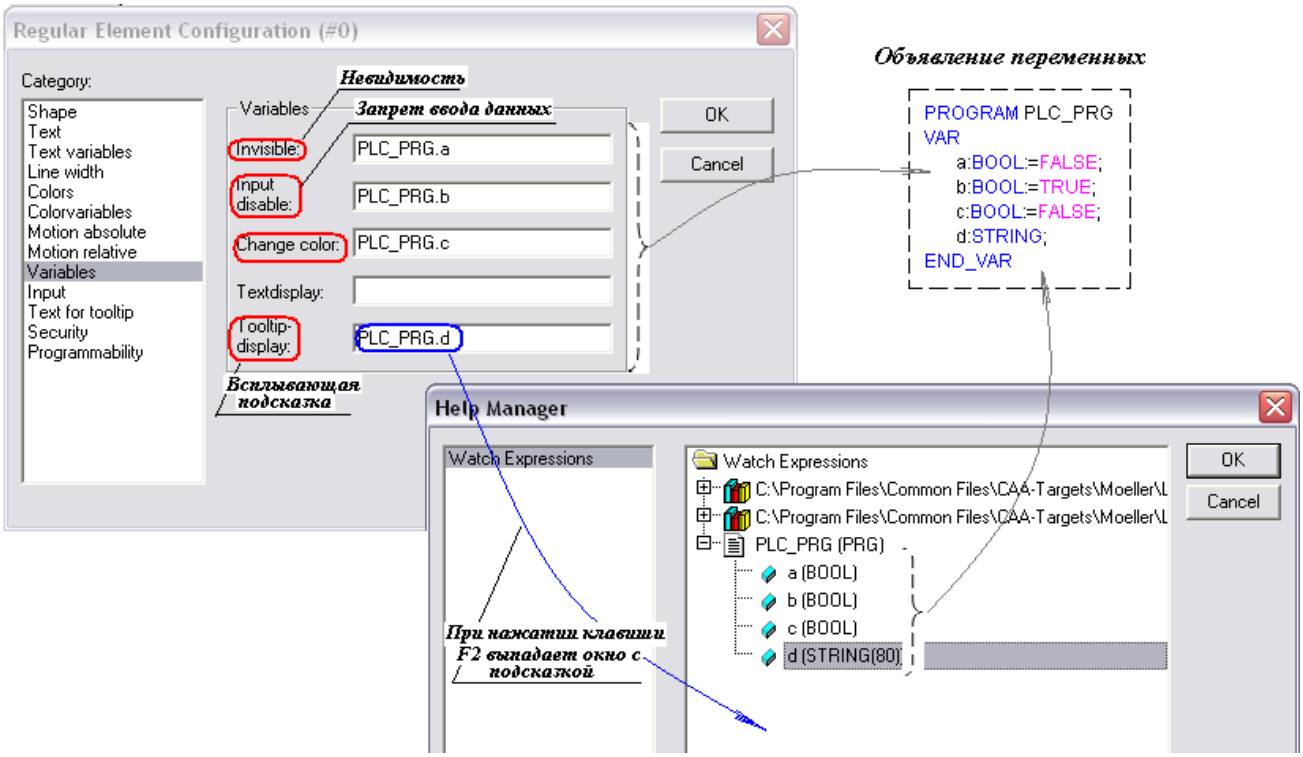
Переменные (Variables)

В категории Variables (Переменные) вы можете указать (как в категориях Textvariables и Color Variables) переменные, описывающие динамическое состояние элемента визуализации (рис. 2.7). Как обычно, при вводе переменных доступен Ассистент ввода данных (<F2>).

Возможности конфигурации

Невидимость. Если введенная здесь булева переменная имеет значение FALSE, то элемент визуализации видим. Если переменная имеет значение TRUE, то элемент невидим.

Запрет ввода данных. Если введенная здесь булева переменная имеет значение TRUE, то все настройки категории 'Input' ('ввод данных') не учитываются.



Р и с. 2.7. Окно конфигурации категории Variables

Изменение цвета. Если введенная здесь булева переменная имеет значение FALSE, то элемент визуализации представляется в его основном цвете. Если переменная является TRUE, то элемент представляется в его тревожном цвете.

Подсказка. Здесь вы можете указать переменную типа STRING, текущее значение которой будет работать как всплывающая подсказка элемента визуализации.

Ввод данных (Input)

Toggle variable (Переменная переключения). Если эта опция активирована, то в режиме online каждый щелчок мышкой на изображении элемента будет изменять значение логической переменной, указанной в поле справа. Значение переменной изменяется при каждом щелчке мыши с TRUE в FALSE и наоборот. Используйте при вводе переменных Ассистент ввода (<F2>).

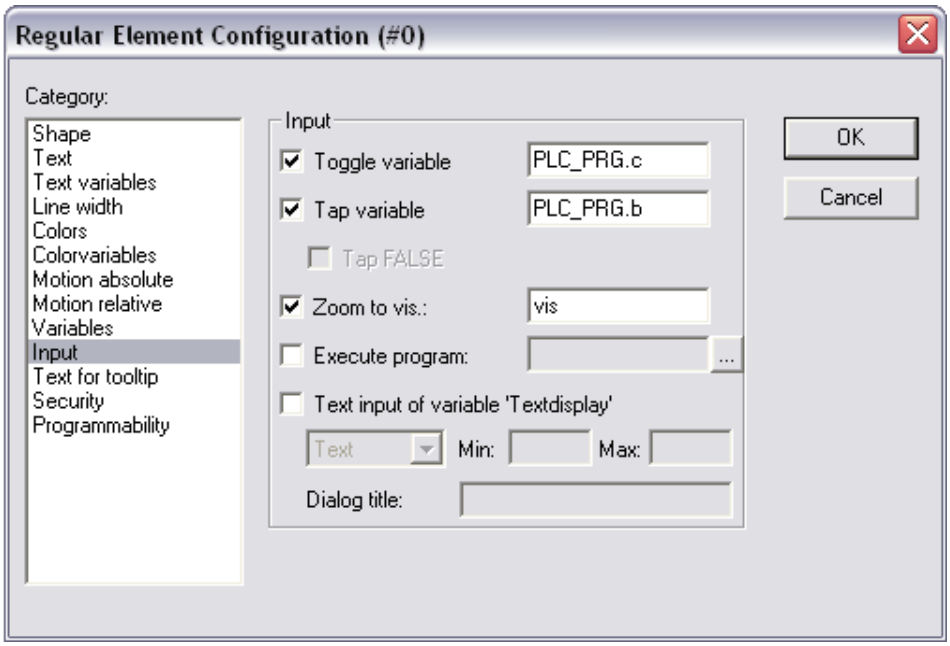
Tap Variable (Переменная-кнопка). Если эта опция активирована, то в режиме online указанная в поле справа логическая переменная будет иметь значение TRUE, пока удерживается клавиша мыши, помещенной на изображение элемента. Если активирована опция Tap FALSE, то значение переменной при нажатии клавиши будет соответствовать FALSE. Как только вы отпустите клавишу, значение переменной возвратится к исходному значению.

Zoom to Vis... Если эта опция активирована, вы можете указывать в следующем поле, к какой визуализации необходимо перейти в режиме online по щелчку мыши на изображении элемента.

Text input of variable 'Textdisplay', (Ввод текста переменной 'Текстовый дисплей'). Если эта опция активирована, в режиме online вы получите возможность вводить в этом элементе визуализации текст, который после нажатия <Enter> преобразуется в значение переменной, заданной в поле Textdisplay категории Variables. Выберите из списка способ ввода данных в online режиме.

Text: (Текст). Будет открыто окно редактирования, в котором значение вводится с клавиатуры.

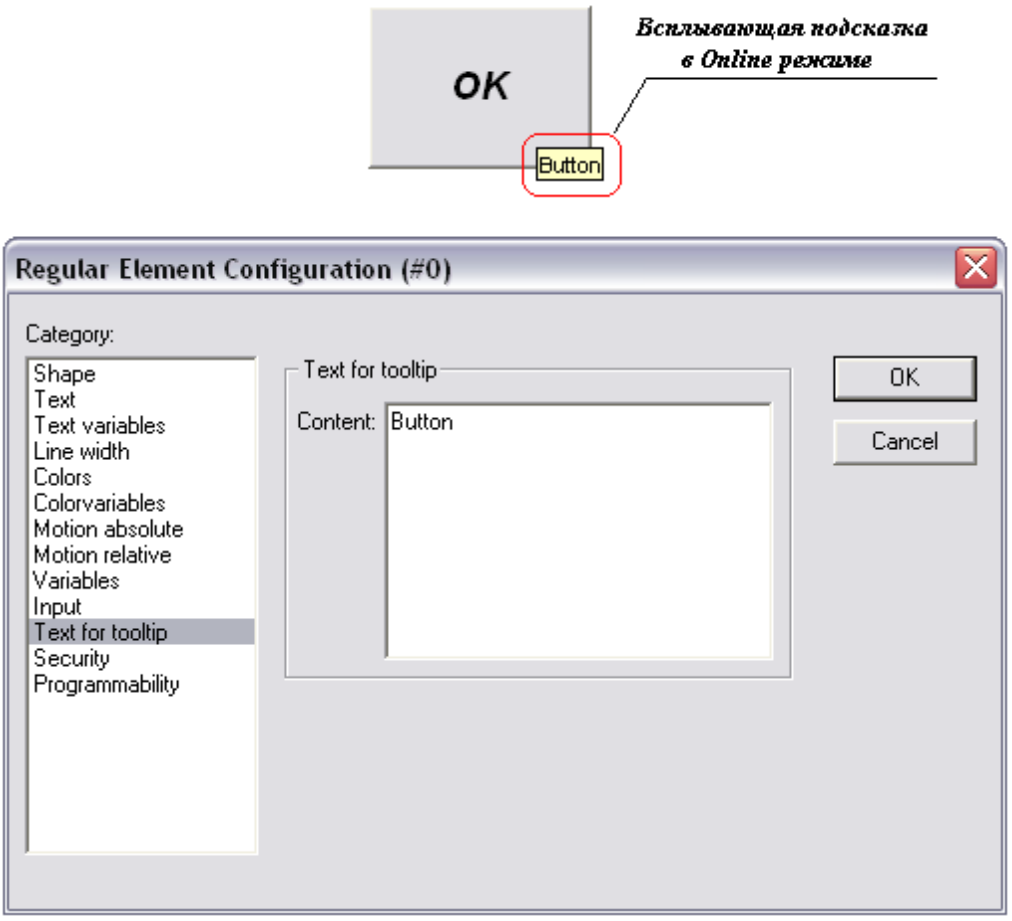
Numpad или Keypad. Будет открыто диалоговое окно с изображением цифрового или алфавитного клавиатурного поля. Ввод значения производится нажатием соответствующих кнопок. Это имеет особое значение при работе с Touch Screen дисплеями. Окно ввода данных показано на рис. 2.8.



Р и с. 2.8. Окно ввода данных

Всплывающая подсказка (ToolTip)

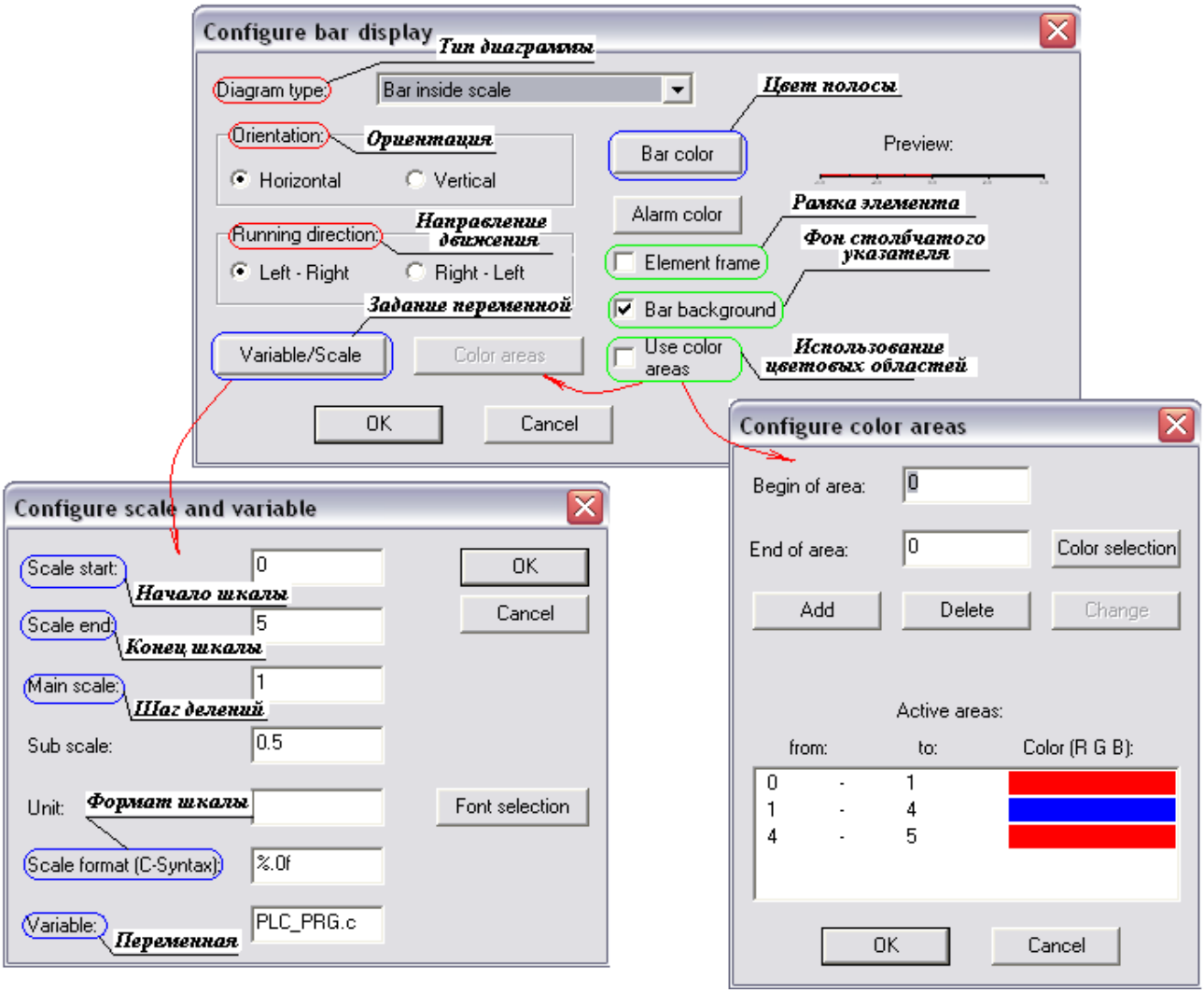
В диалоге Text for Tooltip вы можете ввести текст, который будет появляться во всплывающем текстовом окне при помещении указателя мыши на элемент в режиме online (рис. 2.9). Текст может содержать несколько строк. Используйте комбинацию клавиш <Ctrl> + <Enter> для разрыва строки.



Р и с. 2.9. Всплывающая подсказка в Online режиме
и окно ее конфигурации

Столбчатый указатель (Bar Display)

Данный диалог открывается автоматически, после того как столбчатый указатель добавлен в визуализацию. Предварительный просмотр представляет вид элемента, соответствующий установленным параметрам (рис. 2.10).



Р и с. 2.10. Создание и редактирование столбчатого указателя

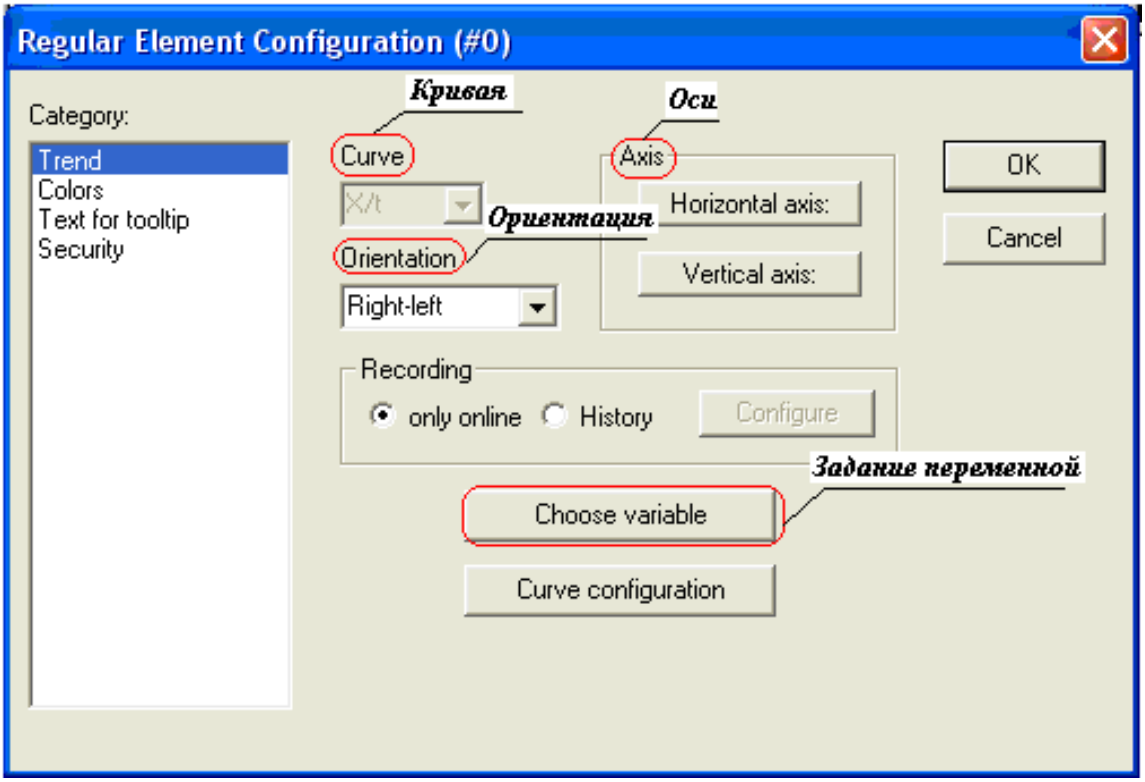
Тренд (Trend)

Тренды служат для записи и отображения хронологического изменения значений переменной в режиме online (рис. 2.11). Они являются аналогом трассировки (Sampling Trace) в визуализации. Интерактивное отображение данных происходит в окне диаграммы. При записи в текстовый файл каждое последующее измерение записывается в очередной строке.

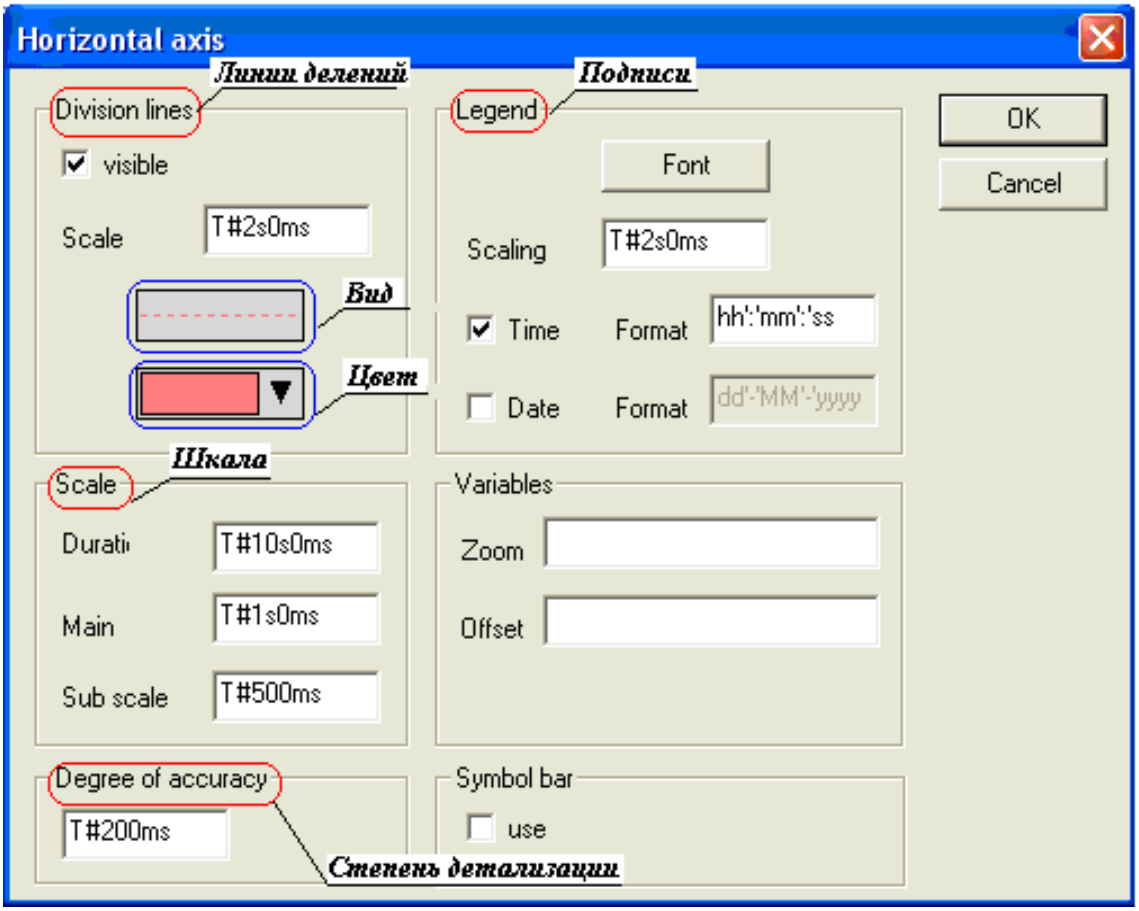
Axis (Оси):

Горизонтальная (horizontal) ось приведена на рис. 2.12.

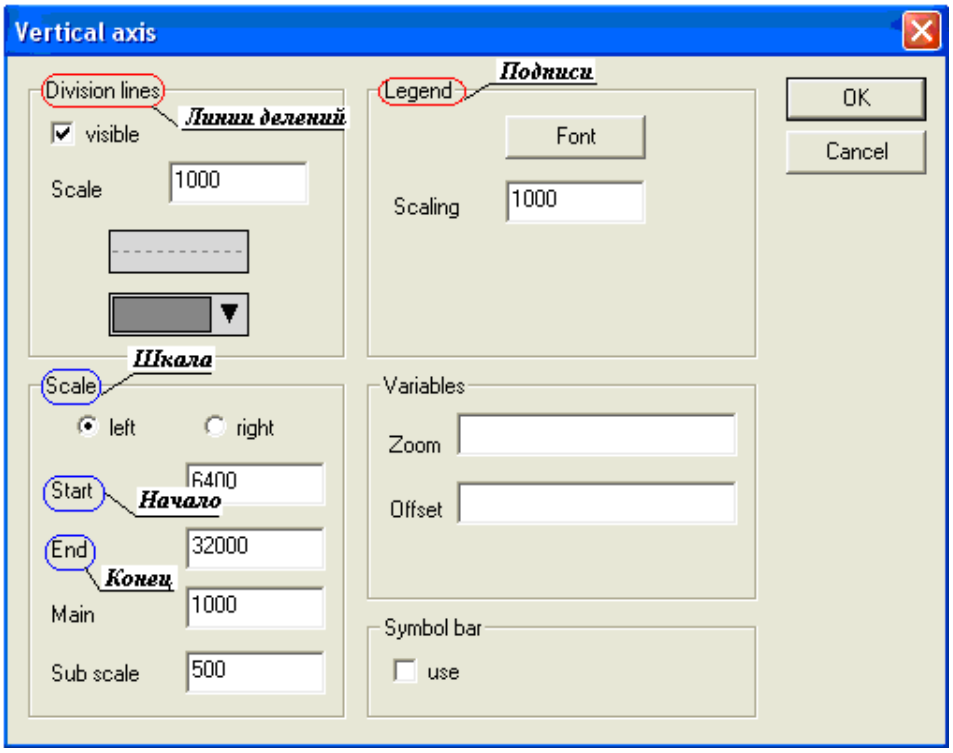
Вертикальная (vertical) ось показана на рис. 2.13.



Р и с. 2.11 Создание линии тренда



Р и с. 2.12. Конфигурация горизонтальной оси



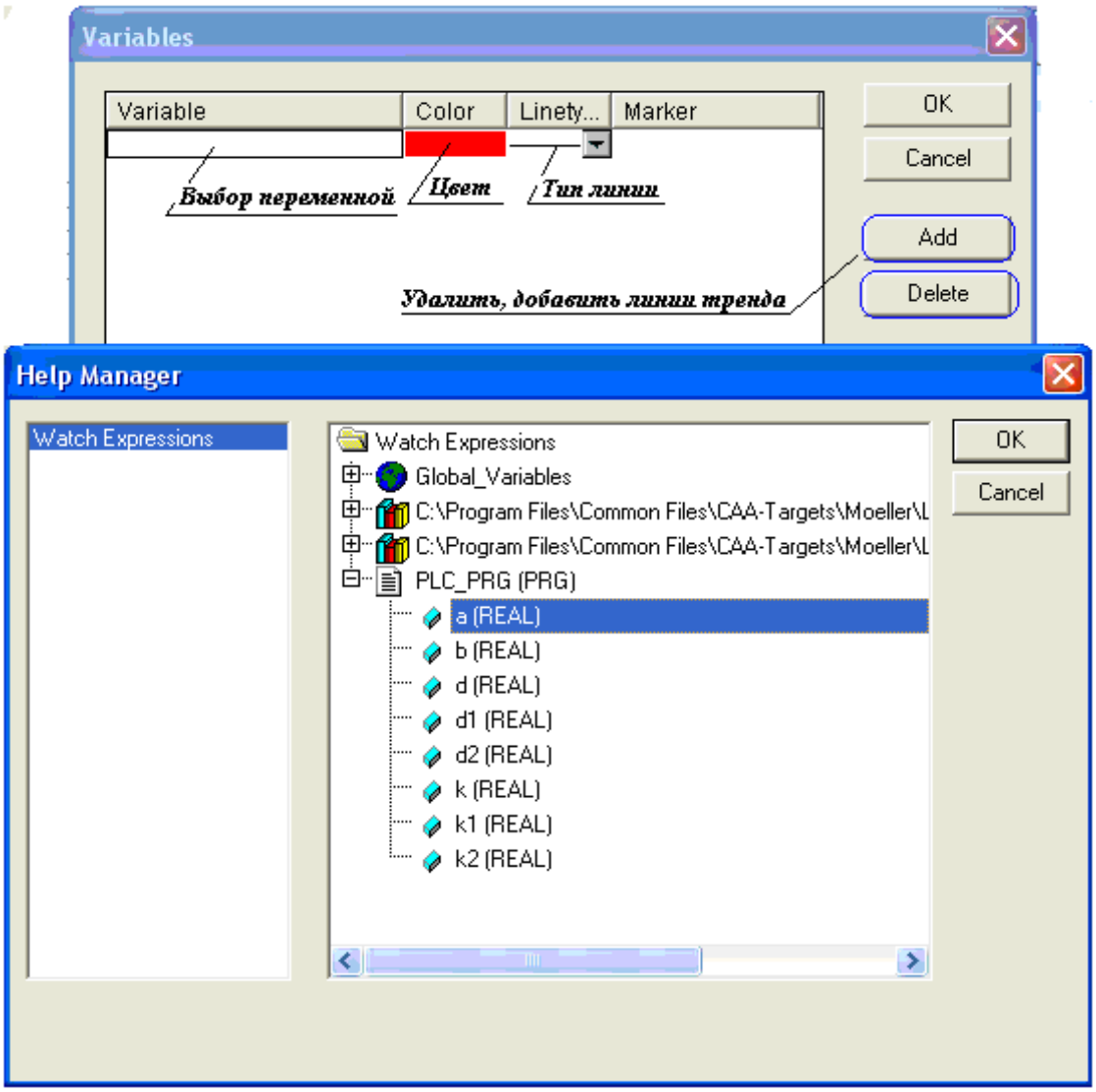
Р и с. 2.13. Конфигурация вертикальной оси

Кнопка *Choose variable* открывает диалог *'Variables' ('Переменные')*. В нем определяется список трассируемых переменных и вид соответствующих им кривых (рис. 2.14). Введите в столбце Variable переменную проекта (щелчком клавиши мыши на поле). Используйте Ассистент ввода <F2> или функцию интеллектуального ввода.

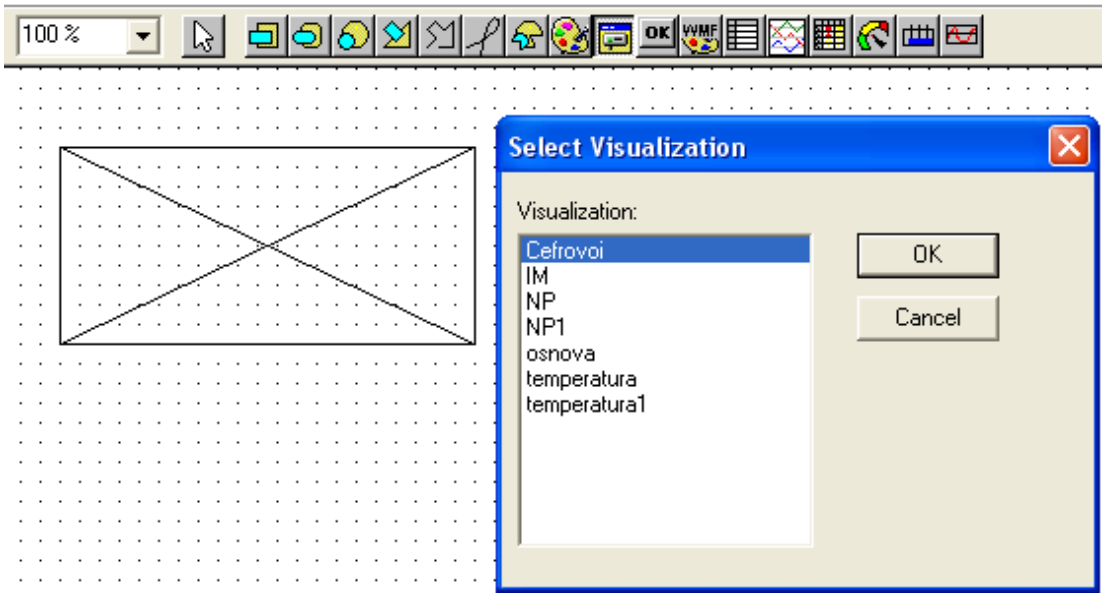
Элемент визуализация (Visualization)

В диалоге конфигурирования категории Visualization определяются свойства визуализации, вставленной как элемент в текущую визуализацию (рис. 2.15). После вставки она обозначается как "reference" ('ссылка').

В поле Visualization (визуализация) указывается имя вставляемой визуализации. Кнопкой открывается диалог, содержащий список всех доступных объектов визуализации.



Р и с. 2.14. Редактирование списка трассируемых переменных и вида соответствующих им кривых



Р и с. 2.15. Добавление элемента визуализации

Задание для самостоятельного решения

1. Создать новый объект визуализации.

2. Смоделировать БРУ, используя возможности редактора визуализации (сочетая различные элементы и их свойства).

• Панель управления смоделированного БРУ должна соответствовать панели “реального” прибора (наличие всех кнопок, цветовых индикаторов, ручек поворота).

• Сигналы с БРУ должны поступать на соответствующие дискретные выходы контроллера (если единичный сигнал подается на нечетный дискретный выход – РО закрывается, если на четный – РО открывается. Каждому из трех объектов соответствует пара дискретных выходов).

Реализация блока ручного управления в редакторе визуализации показана на рис. 2.16.

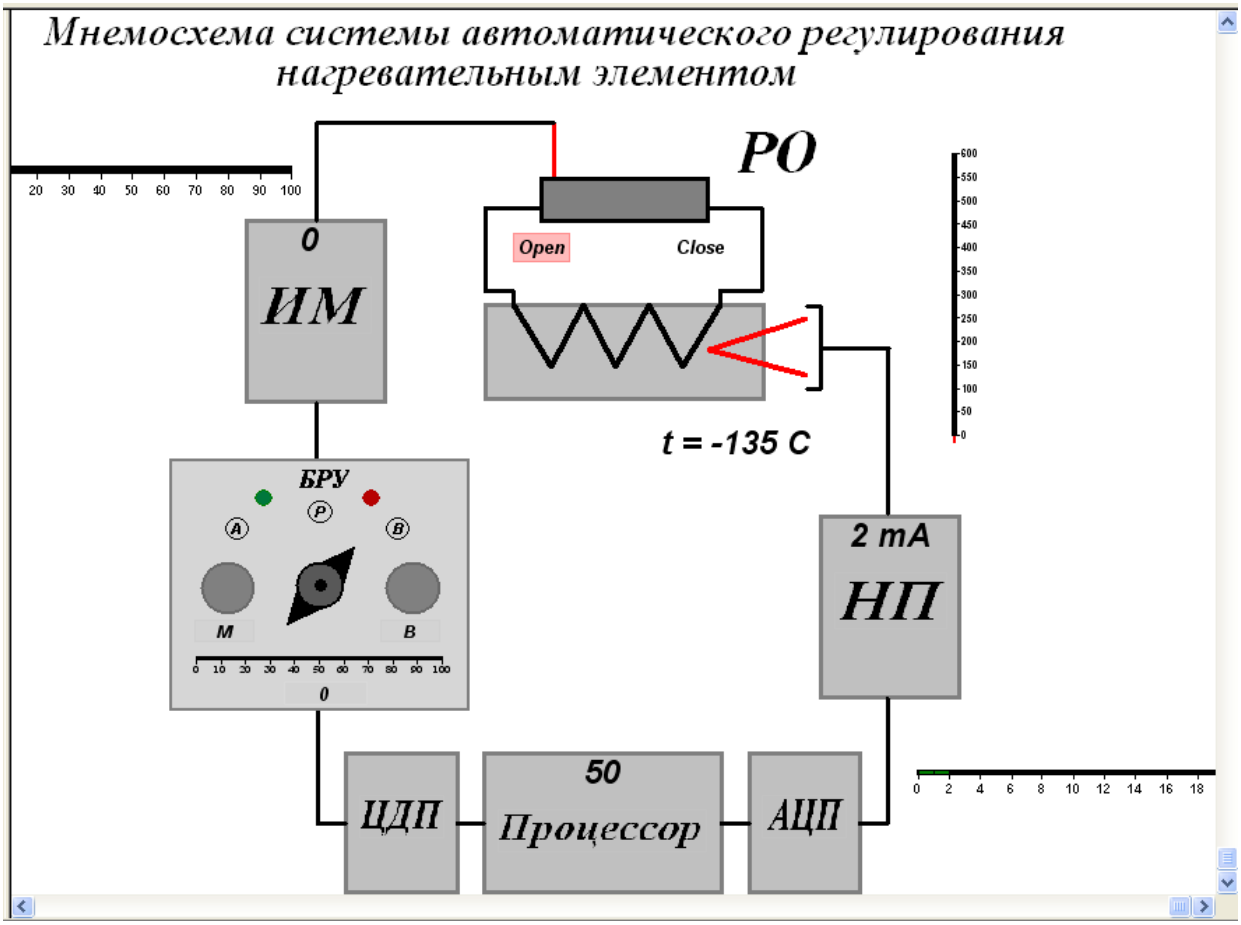
Р и с. 2.16. Реализация блока ручного управления
в редакторе визуализации

3. Добавить еще один объект визуализации. Используя линии тренда, записать и отобразить изменения значений аналоговых сигналов, приходящих на входы контроллера, в online режиме.

4. Смоделировать обобщенную мнемосхему всего контура (в нее должны войти объект регулирующий орган, НП, контроллер, БРУ, ИМ).

- Схематически изобразить каждый из выше перечисленных приборов
- При нажатии на элемент, отожествляющий то или иной прибор (БРУ, контроллер (его входы)), должна открыться соответствующая ему визуализация, созданная вами в предыдущих пунктах (3, 2).

Ниже приведен один из возможных вариантов реализации данного контура, со всеми входящими в него элементами (рис. 2.17).



Р и с. 2.17. Мнемосхема системы автоматического регулирования нагревательным элементом

Практическая работа № 3
ЯЗЫК LD. РАБОТА С ДИСКРЕТНЫМИ СИГНАЛАМИ

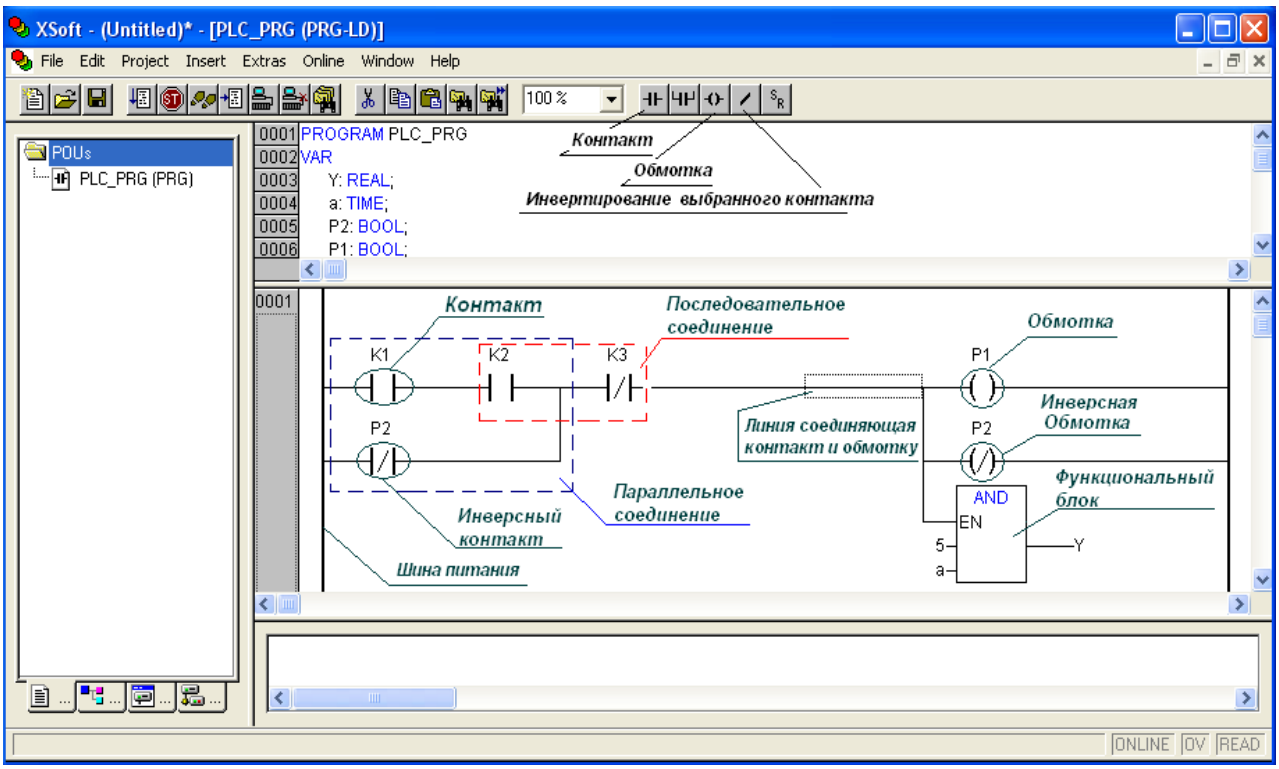
Цель работы – изучение языка релейных диаграмм LD, используемого при обработки дискретных сигналов. Знакомство с особенностями редактора языка LD.

Язык релейных диаграмм (LD)

Язык релейных или релейно-контактных схем (РКС) – графический язык, реализующий структуры электрических цепей (рис. 3.1).

Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи – как в FBD. Кроме того, LD достаточно удобен для управления другими компонентами POU.

Диаграмма LD состоит из ряда цепей. Слева и справа схема ограничена вертикальными линиями – шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями.



Р и с. 3.1. Окно редактора LD. Основные составляющие данного графического языка

Контакт

Контакты обозначаются двумя параллельными линиями и могут иметь состояния "ON" или "OFF". Эти состояния соответствуют значениям ИСТИНА или ЛОЖЬ. Каждому контакту соответствует логическая переменная. Если значение переменной ИСТИНА, то контакт замкнут.

Контакты могут быть соединены параллельно, тогда соединение передает состояние "ON", когда хотя бы одна из ветвей передает "ON".

Если контакты соединены последовательно, то для того чтобы соединение передало "ON", необходимо, чтобы оба контакта передавали "ON". Это соответствует электрической параллельной и последовательной схеме.

Контакт может быть инвертируемым. Такой контакт обозначается с помощью символа |/| и передает состояние "ON", если значение переменной ЛОЖЬ.

Обмотка

В правой части схемы может находиться любое количество обмоток (реле), которые обозначаются круглыми скобками ( ). Они могут соединяться только параллельно. Обмотка передает значение соединения слева направо и копирует его в соответствующую логическую переменную.

В целом цепь может быть либо замкнутой (ON), либо разомкнутой (OFF). Это как раз и отражается на обмотке и соответственно на логической переменной обмотки (ИСТИНА/ЛОЖЬ).

Обмотки также могут быть инверсными. Если обмотка инверсная (обозначается символом (/)), тогда в соответствующую логическую переменную копируется инверсное значение.

Функциональные блоки в LD

Кроме контактов и обмоток, в LD можно использовать функциональные блоки и программы. Они должны иметь логические вход и выход и могут использоваться так же, как контакты.

С помощью основных элементов собирается электрическая цепь. Прохождение сигнала по цепи зависит от состояния выключателей

(замкнут или разомкнут). Если электрический сигнал проходит через катушку реле, то контакт реле замыкается или размыкается в зависимости от типа реле.

Управление контактами осуществляется с помощью значений логических переменных. Адрес или имя управляющей логической переменной указывается на схеме над контактом. Если значение управляющей переменной «ложь» или логический нуль, то контакт находится в «нормальном» состоянии: замыкающий контакт разомкнут, а размыкающий контакт замкнут. Когда значение управляющей переменной примет значение «истина», или логическая единица, замыкающий контакт замкнётся и будет проводить ток, а размыкающий контакт разомкнётся.

«Нормальным» состоянием называют состояние, в котором находится контакт при отсутствии внешних воздействий, поэтому иногда встречаются термины «нормально замкнутый» и «нормально разомкнутый» контакт или контакт реле.

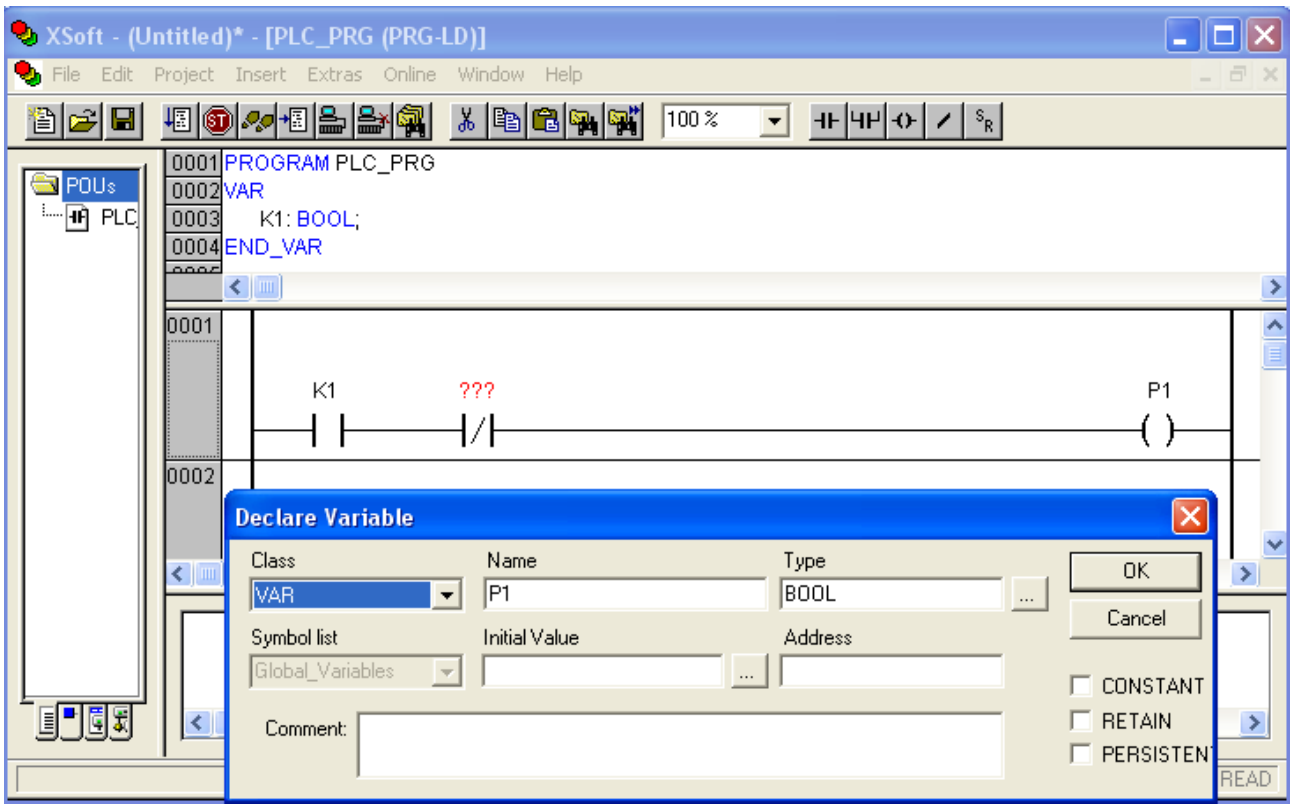
В отличие от контакта контакты реле изменяют состояние логических переменных, указанных на релейной схеме над катушками реле. Если ток не течёт, то замыкающийся (нормально разомкнутый) контакт реле разомкнут и логическая переменная принимает значение «ложь». Размыкающийся (нормально замкнутый) контакт реле в этом случае замкнут, и логическая переменная принимает значение «истина».

При протекании тока через катушку реле значение переменной изменяется на противоположное.

Чтобы назначить логическую переменную контакту или катушке реле, нужно щелкнуть левой клавишей мыши на знак ??? над выбранном элементом. Дать имя переменной – в диалоговом окне Declare Variable подтвердить ее тип, как BOOL. Пример объявления приведен в рис. 3.2.

Редактор LD в режиме Online

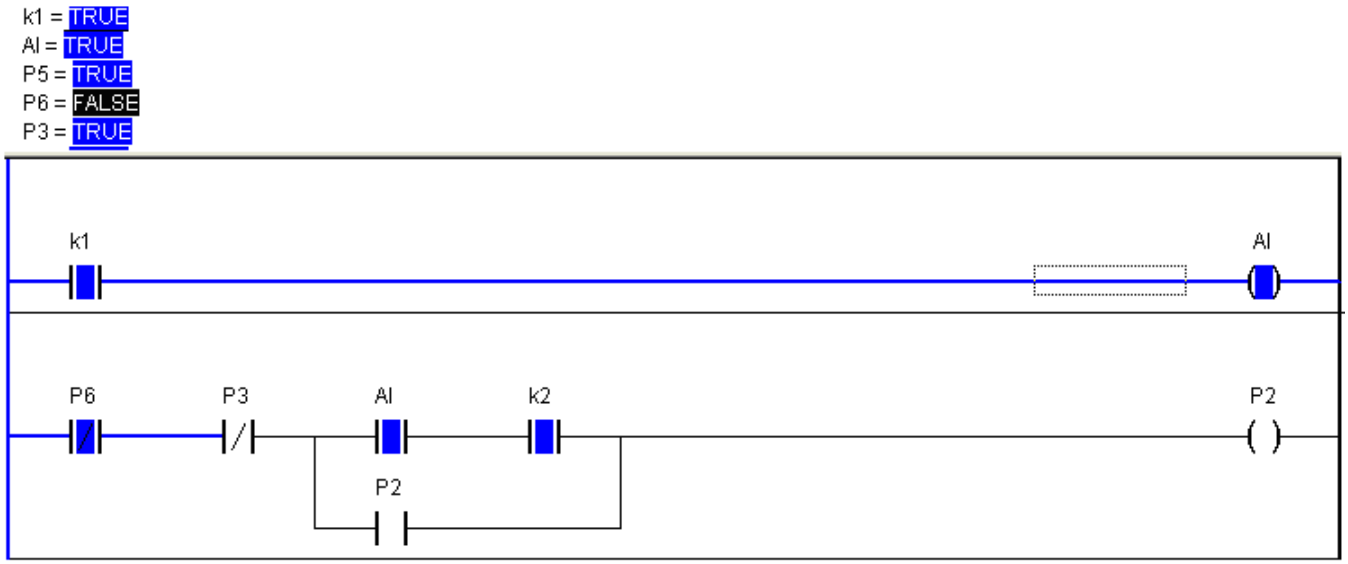
В режиме Online (рис.3.3) контакты и обмотки, которые находятся в состоянии On, изображаются синим цветом. Кроме того, все линии, передающие состояние On, также окрашиваются синим. Указываются значения всех входов и выходов функциональных блоков.



Р и с. 3.2. Объявление переменных в языке LD

В режиме Online можно устанавливать точки останова и выполнять программу по шагам.

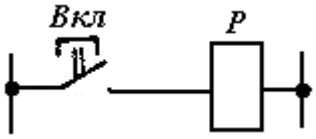
Если вы переместите указатель мыши на переменную, то в подсказке появятся тип, комментарии и адрес этой переменной.



Р и с. 3.3. Работа редактора LD в режиме Online

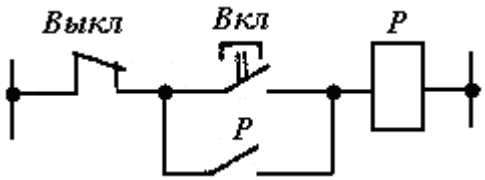
Простейшие схемы включения реле

Схема повторителя (рис.3.4). При замыкании кнопки «Вкл» замыкаются цепи питания катушки реле Р, оно срабатывает. При размыкании кнопки реле обесточивается, т.е. «повторяет» положение контакта (кнопки).



Р и с. 3.4. Схема повторителя

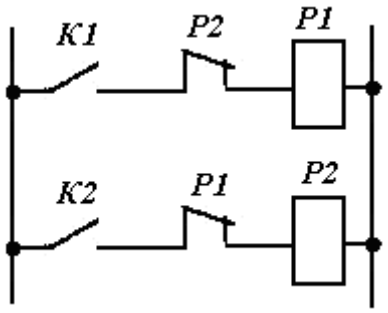
Схема самоблокировки (памяти) (рис.3.5). При замыкании кнопки «Вкл» замыкается цепь питания катушки реле Р, которое срабатывает и своими замыкающими контактами шунтирует кнопку пуска. Таким образом, при размыкании кнопки «Вкл» катушка реле остается замкнутой через свой собственный контакт. Реле заблокировалось во включенном состоянии, оно как бы запомнило кратковременный импульс, полученный при замыкании контактов кнопки «Вкл», включенных последовательно с катушкой реле.



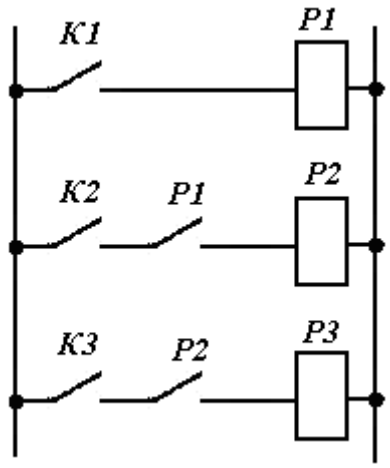
Р и с. 3.5. Схема самоблокировки

Схема взаимной блокировки (рис. 3.6) исключает возможность одновременного срабатывания двух или нескольких реле. Например, нельзя включить одновременно два реле, включающих двигатель для вращения в разные стороны. Для осуществления взаимной блокировки цепь катушки данного реле включается последовательно с контактами тех реле, одновременно с которыми данное реле не должно включаться.

Схема последовательной блокировки обеспечивает только один определенный порядок ее включения (рис.3.7). В ряде случаев нарушение порядка включения аппаратуры может вызвать аварию.

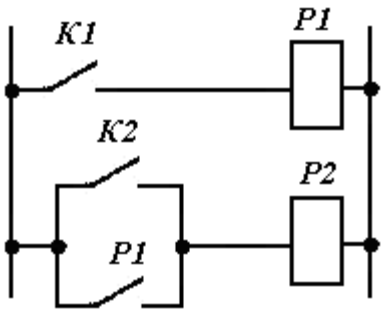


Р и с. 3.6. Схема взаимной блокировки



Р и с. 3.7. Схема последовательной блокировки

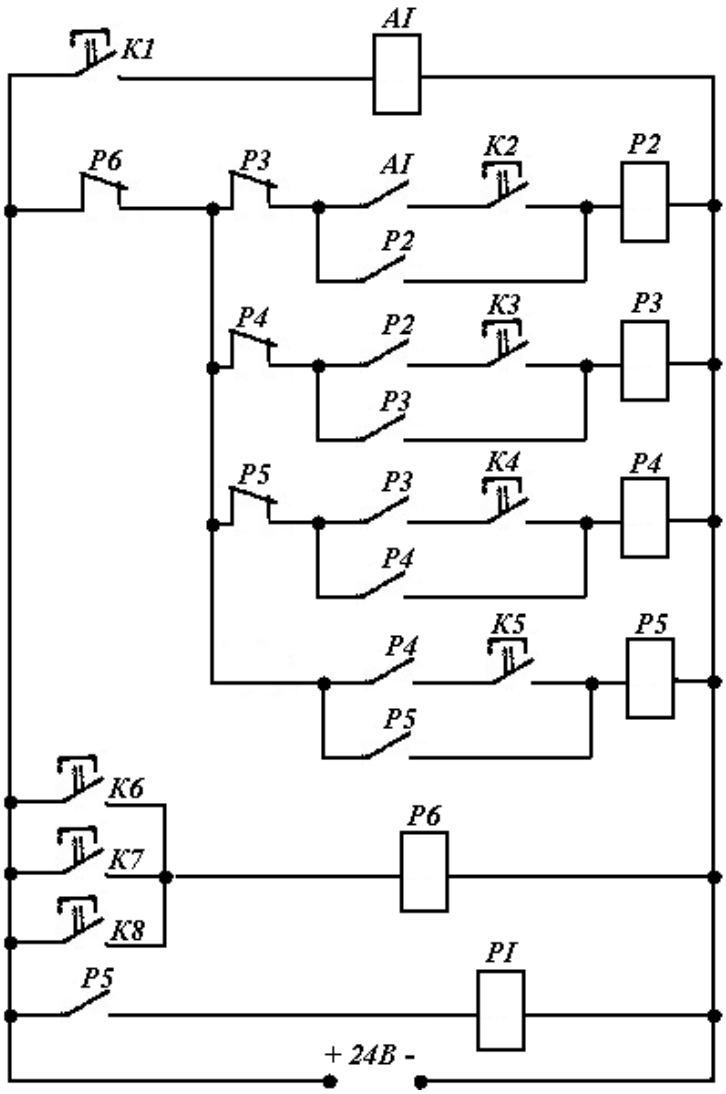
Можно осуществить также *последовательную блокировку выключения*, если необходимо обеспечить строго определенный порядок выключения аппаратуры. Для этого параллельно данному включают контакты тех реле, которые должны быть включены раньше (рис.3.8).



Р и с. 3.8. Схема последовательной блокировки

Рассмотрим возможности данного графического языка программирования на примере (пример демонстрирует замену релейного автомата программной реализацией на ПЛК без переработки алгоритма работы устройства).

Пусть дана некоторая система регулирования, принципиальная электрическая схема которой приведена на рис.3.9.

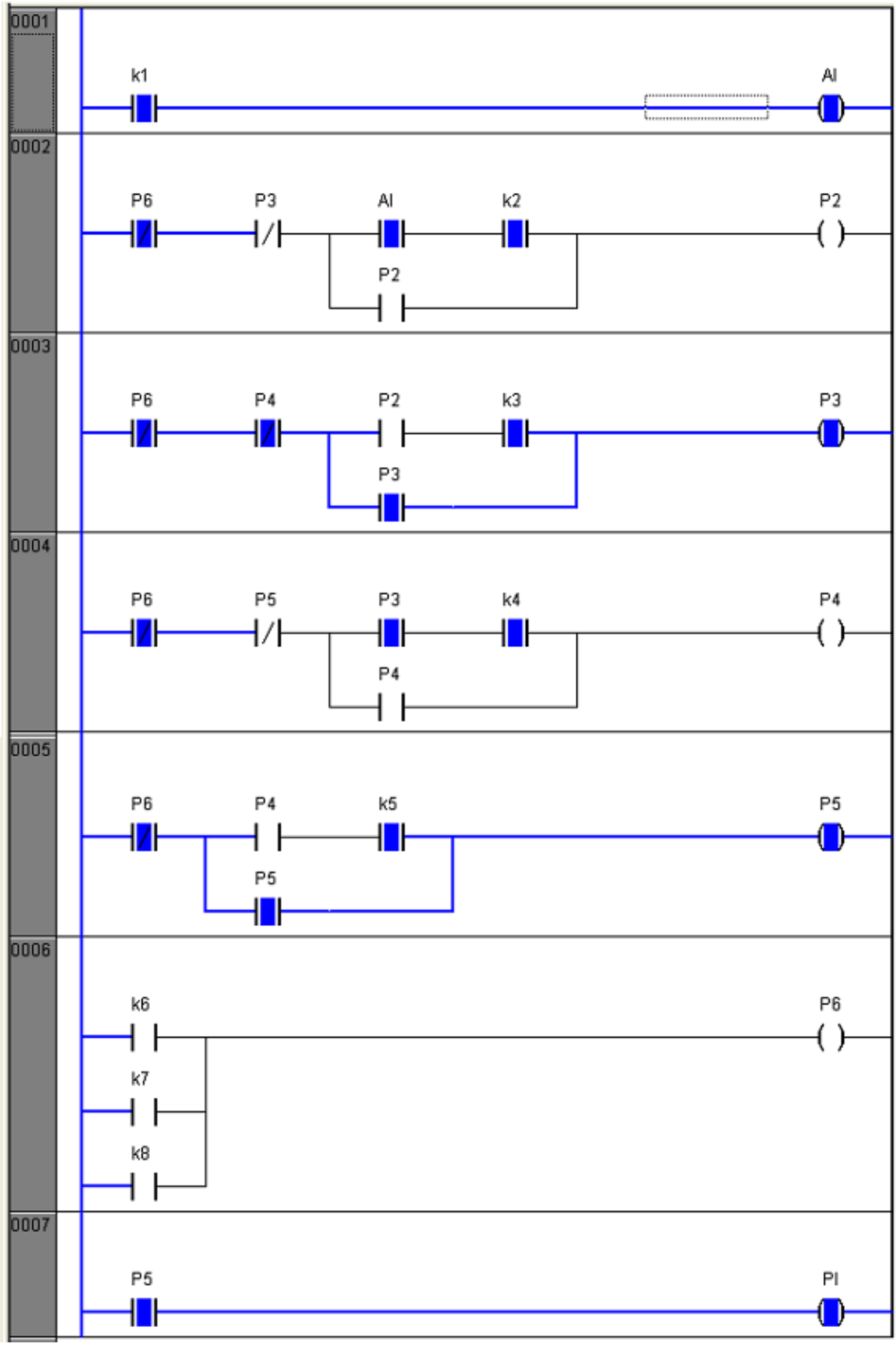


Р и с. 3.9. Принципиальная электрическая схема системы регулирования

Задача состоит в том, чтобы “нажать” на нужные кнопки и в соответствующей последовательности (К1, К2, К3, К4, К5) так, чтобы замкнулся контакт реле Р5 и через реле PI прошел единичный сигнал.

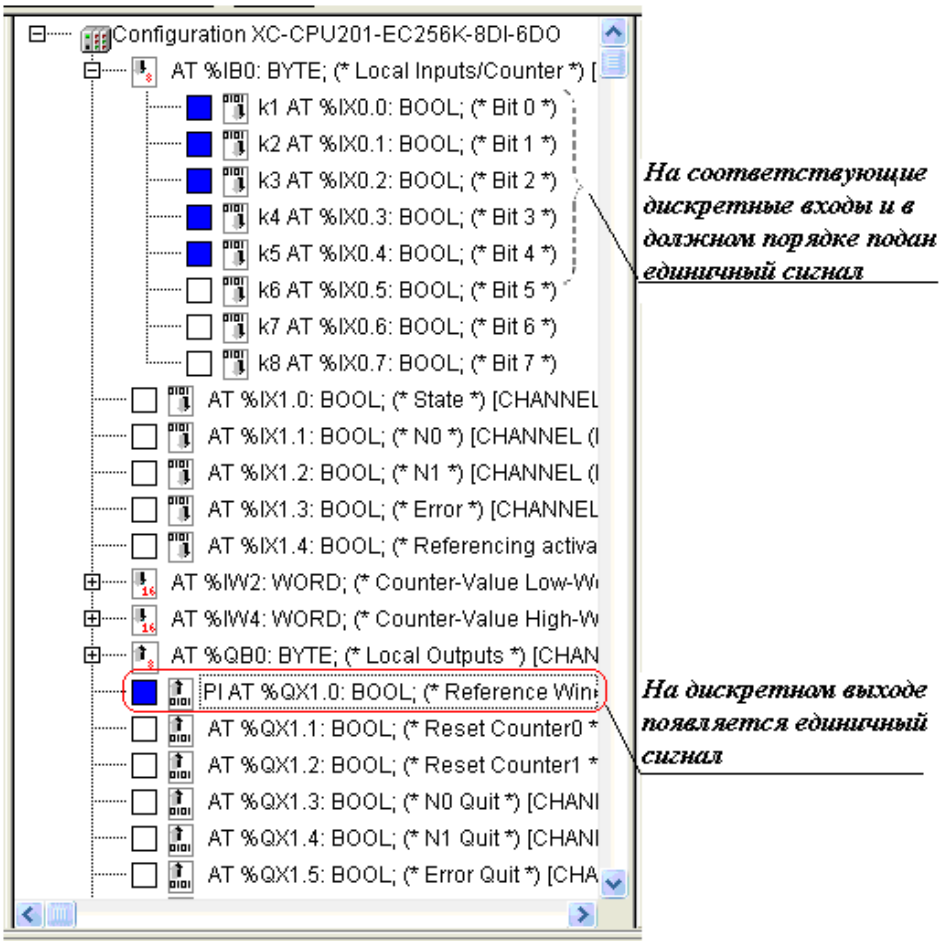
Ошибочное нажатие кнопок К6, К7, К8 сбрасывают сигнал в исходное нулевое состояние.

Реализующая логику управления диаграмма LD показана на рис. 3.10. Легко заметить, что диаграмма практически повторяет принципиальную схему. Единственное отличие в том, что контакты реле Р6 разделены на несколько цепей. Это диктуется правилами построения диаграмм и не влияет на работу схемы.



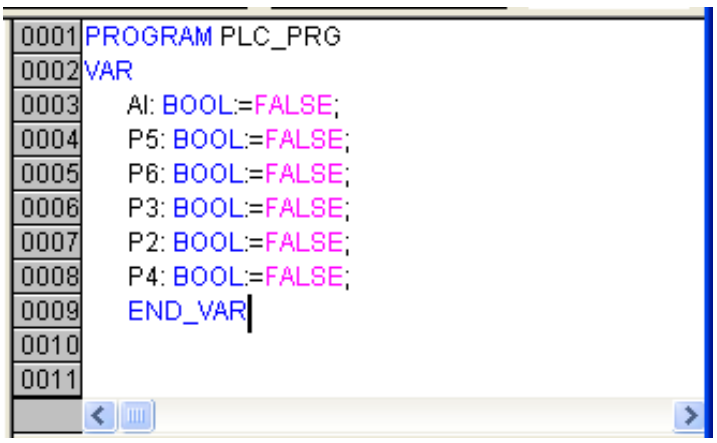
Р и с. 3.10. Диаграмма на языке LD в Online режиме

Для реализации данной системы на контроллере необходим ПЛК имеющий, как минимум, 8 дискретных входов К1 – К8 и 1 выход PI.



Р и с. 3.11. Аппаратные средства, необходимые для реализации поставленной задачи

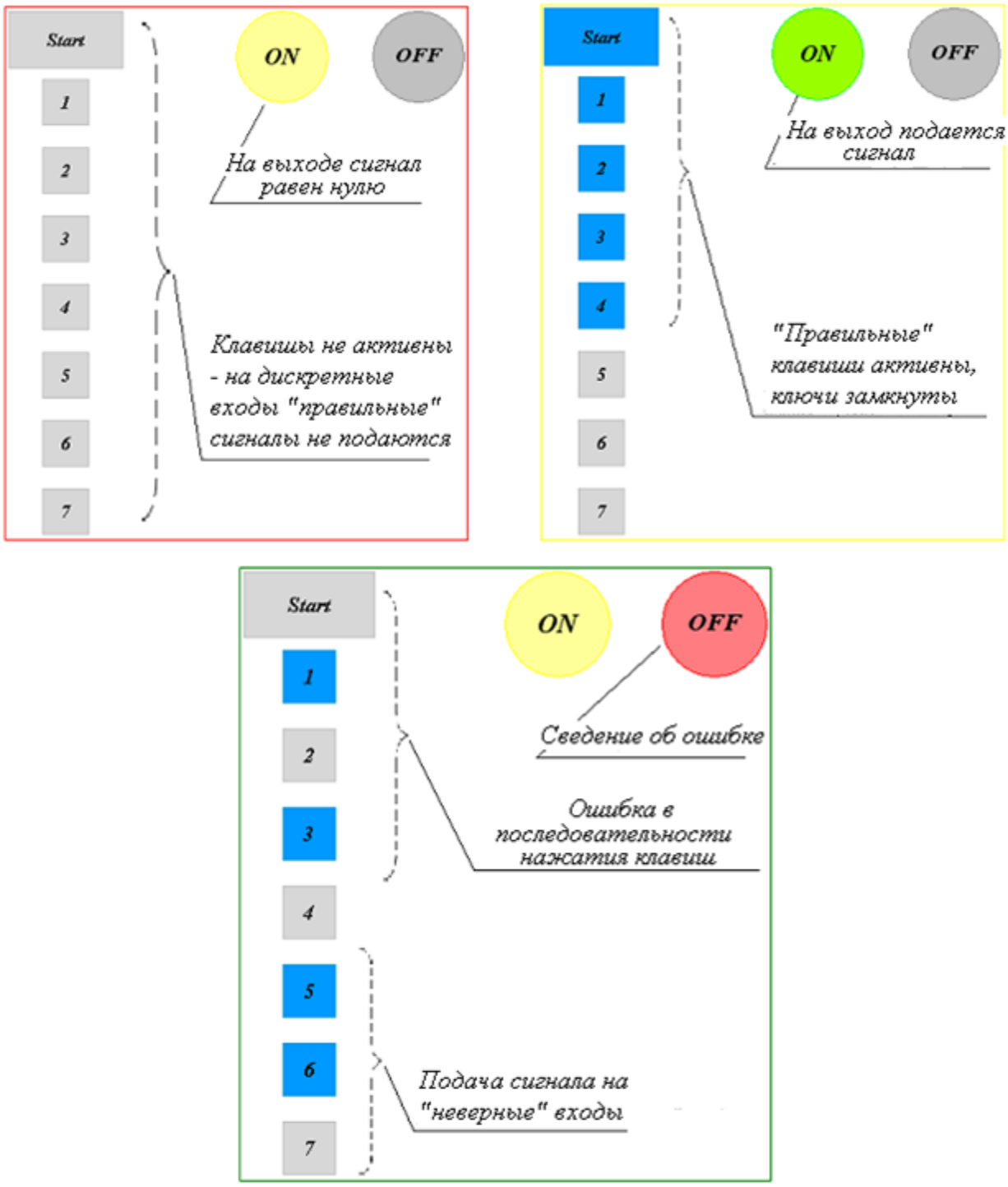
Остальные переменные (AI, P2, P3, P4, P5,P6) помещаются в разделе объявлений (рис.3.12).



Р и с. 3.12. Объявление внутренних переменных проекта

Реализация подачи сигналов на дискретные входы ПЛК

1. В Online режиме щелчком по квадратику напротив нужного вам входа (рис. 3.11)
2. Смоделировать нашу систему в редакторе визуализации (рис.3.13).



Р и с. 3.13. Реализация данной системы регулирования в редакторе визуализации

Задание для самостоятельного решения

1. Реализовать простейшие схемы включения реле на языке LD.

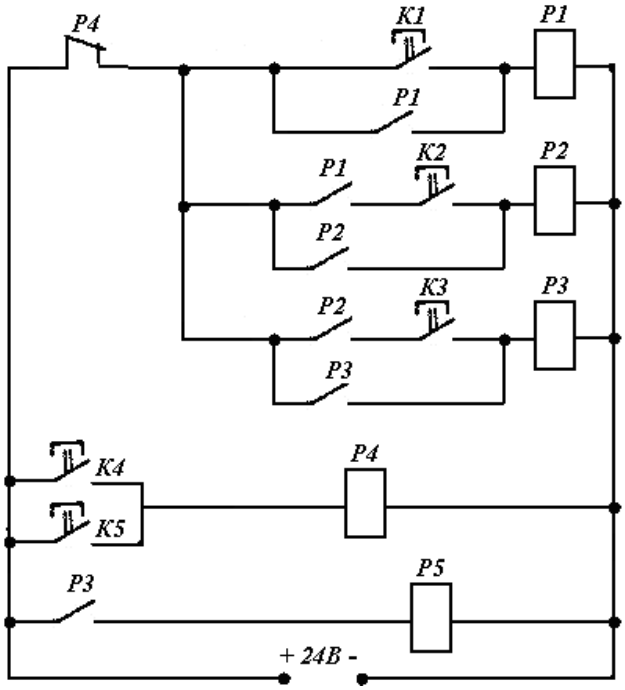
2. Заменить релейный автомат программной реализацией на ПЛК без переработки алгоритма работы устройства.

1. Реализовать диаграмму на языке LD, соответствующую данной принципиальной схеме.

2. Сконфигурировать поставленную задачу (дать символические имена дискретным входам и выходам).

3. Откомпилировать проект, исправить ошибки. Подать необходимые сигналы на входа ПЛК и пронаблюдать за сигналом на выходе.

4. Смоделировать систему в редакторе визуализации (рис.3.14).



Р и с. 3.14. Принципиальная электрическая схема

3. Разработать программу управления электрическим двигателем. Двигатель должен вращаться в двух направлениях. Прежде чем сменить направление вращения на противоположное, двигатель следует остановить. При одновременном нажатии клавиш, отвечающих за вращение двигателя, должна срабатывать защита.

4. В схеме должны присутствовать две кнопки, отвечающие за направление вращения двигателя.

5. Кнопка, при нажатии на которую происходит останов двигателя.

6. Элемент защиты, срабатывающий при одновременном нажатии клавиш, отвечающих за вращение двигателя.

СОДЕРЖАНИЕ ОТЧЕТА

1. Простейшие схемы включения реле на языке LD (их изображения в данном редакторе).
2. Данная принципиальная электрическая схема, соответствующая ей диаграмма на языке LD.
3. Перечень всех переменных, с соответствующими им типами.
4. Изображение конфигурации данной задачи (со всеми задействованными дискретными входами/выходами).
5. Представление данной схемы в редакторе визуализации (в Online режиме).
6. Принципиальная схема управления электрическим двигателем и соответствующая ей диаграмма на языке LD. Реализация данной задачи в редакторе визуализации.

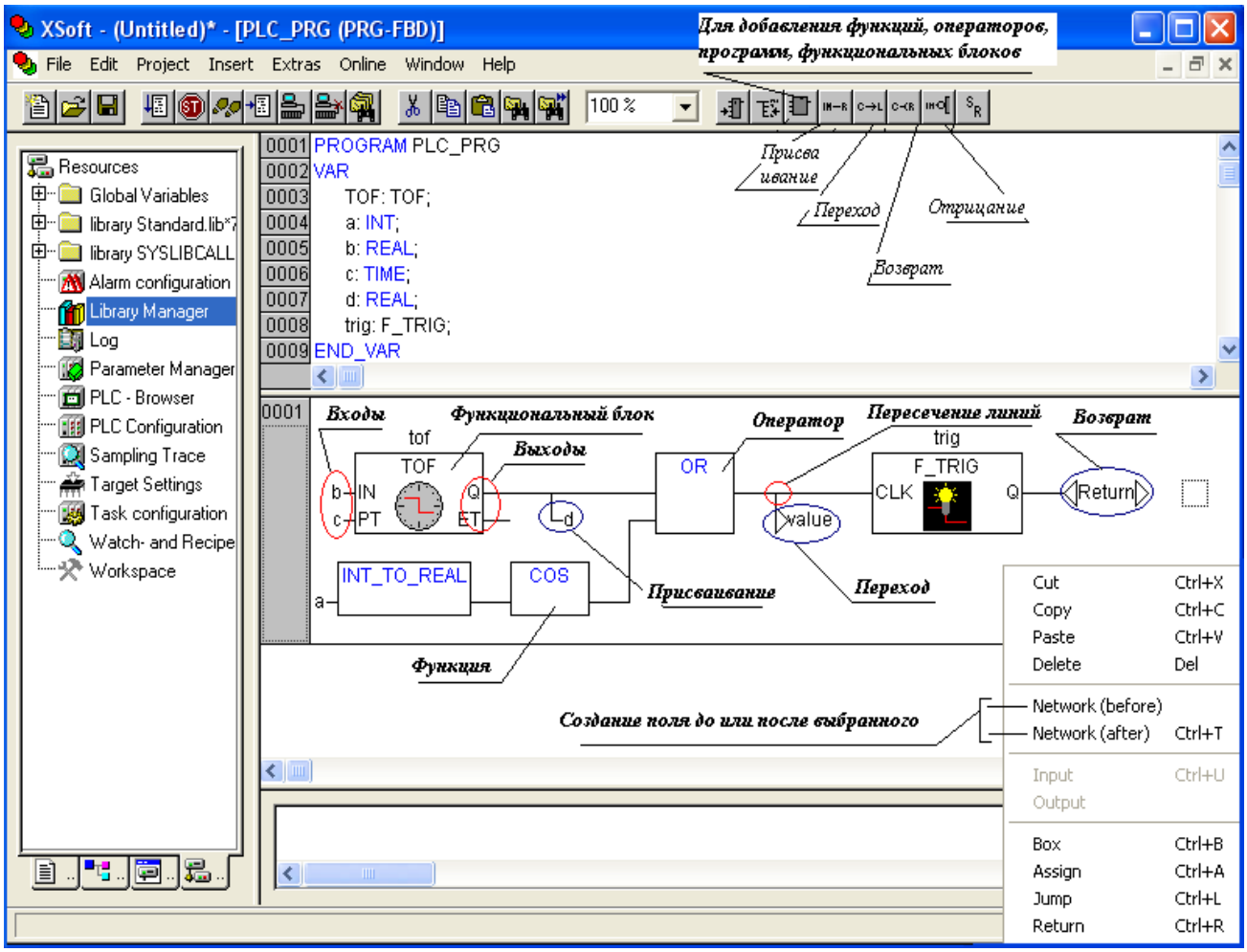
Практическая работа № 4
ЯЗЫКИ FBD И CFC – СХОДСТВА И РАЗЛИЧИЯ. ОСОБЕННОСТИ РАБОТЫ С ФУНКЦИОНАЛЬНЫМИ БЛОКАМИ. БИБЛИОТЕКИ ФУНКЦИОНАЛЬНЫХ БЛОКОВ

Цель работы – изучение графических языков программирования FBD и CFC. Работа с менеджером библиотек. Знакомство с функциональными блоками, которые используются при обработке дискретных и аналоговых сигналов.

Язык функциональных блоковых диаграмм (FBD)

FBD – это графический язык программирования. Он работает с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата (рис. 4.1).

Наиболее важные функции вы можете найти в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl>+<F10>.



Р и с. 4.1. Окно редактора FBD. Основные элементы, входящие в его состав

FBD диаграмма в режиме Online

В режиме Online в редакторе FBD можно устанавливать точки останова. Если в цепи была установлена точка останова, то номер соответствующей цепи станет синим. Выполнение программы останавливается перед цепью, в которой установлена точка останова. В этом случае номер цепи становится красным.

Используя команду “Step in” или ”Step over”, можно последовательно выполнять цепи, останавливаясь после каждой.

На экран выводится текущее значение каждой переменной. Исключение составляет тот случай, когда вход функционального блока – это выражение. Тогда выводится только значение первой переменной в выражении.

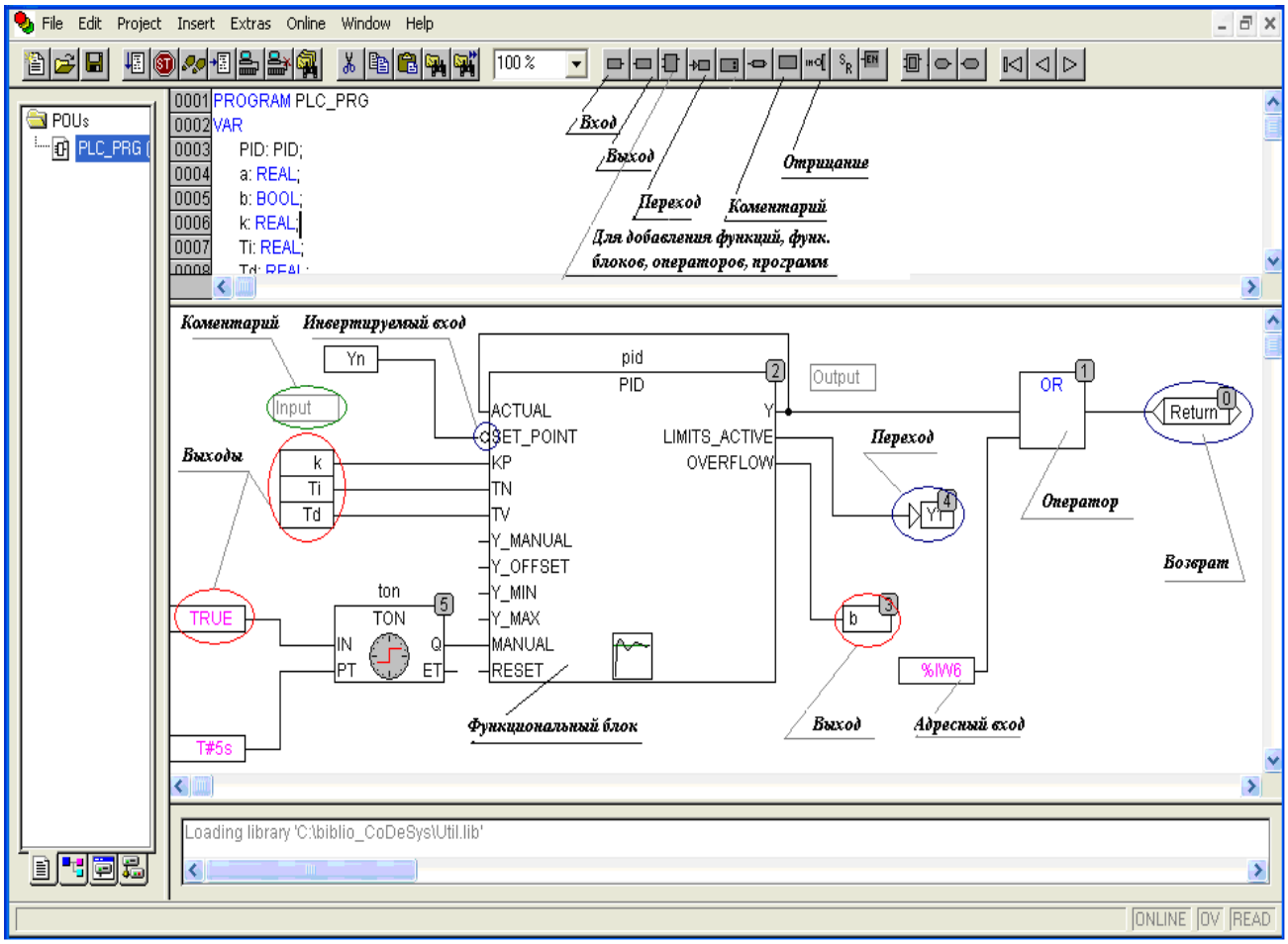
Двойной щелчок мышью по переменной выводит диалоговое окно для ввода нового значения переменной. Если переменная является

логической, то диалоговое окно не выводится, а значение переменной просто переключается. Для записи значения переменных в контроллер используется команда “Online” ”Write values”. После этого переменные снова становятся черными.

В режиме Online, если вы переместите указатель мыши на переменную, то в подсказке появится тип, комментарии и адрес этой переменной.

Язык непрерывных функциональные схемы (CFC)

В отличие от FBD редактор непрерывных функциональных схем не использует цепи, но дает возможность свободно размещать компоненты и соединения, что позволяет создавать обратные связи (рис. 4.2).



Р и с. 4.2. Окно редактора CFC. Основные элементы, входящие в его состав

В этом редакторе нет сетки, и поэтому элементы могут располагаться где угодно. К элементам языка CFC относятся блоки, входы, выходы, возвраты, произвольные переходы, метки и комментарии.

Входы и выходы этих элементов можно соединять, перетаскивая линии соединения мышкой. Эти линии будут перерисовываться автоматически при перемещении элементов. В случае если линия соединения не может быть перерисована, то она становится красной, и как только вы переставите элемент так, чтобы можно было соединить вход и выход линией без пересечений с другими элементами, линия становится нормальной.

Моделирование в графическом редакторе CFC

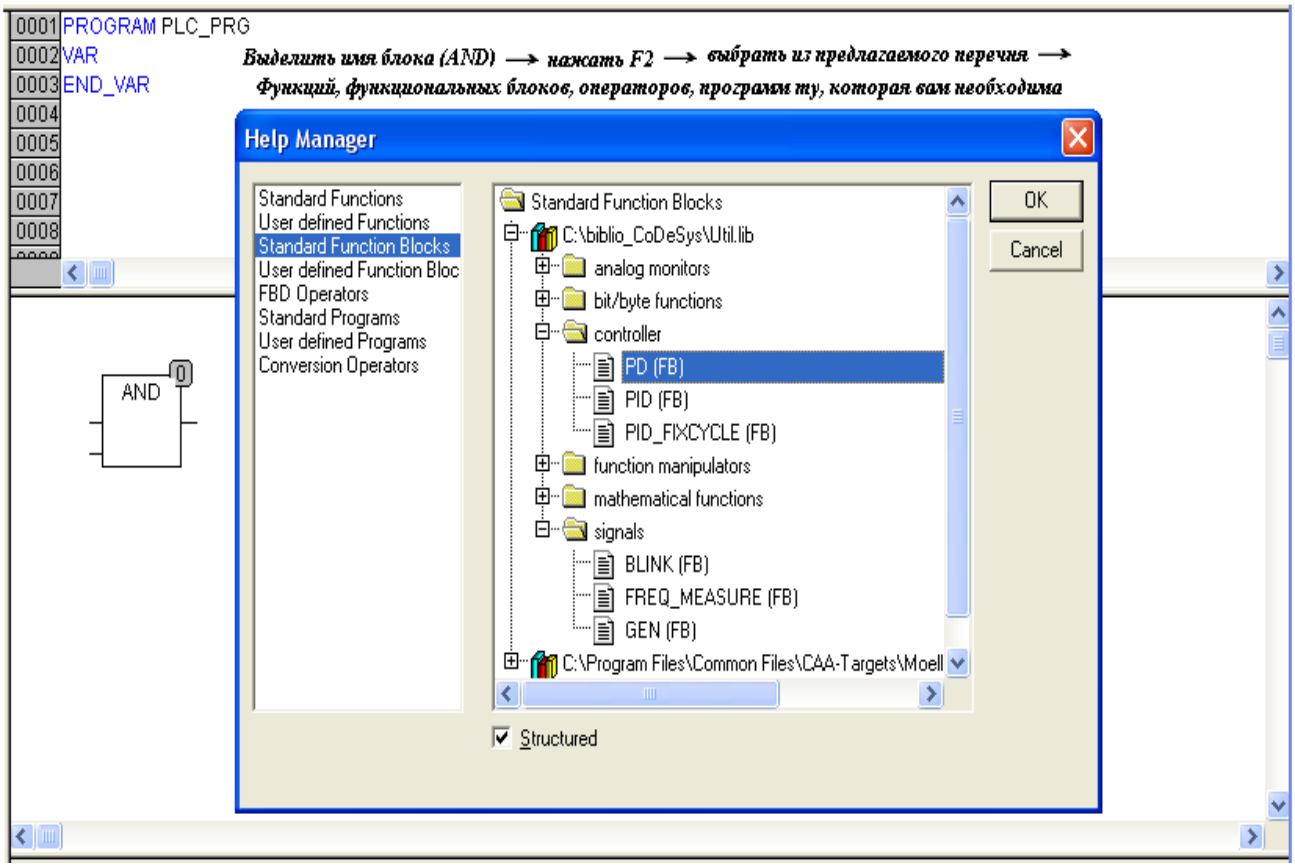
1. Для вставки необходимых операторов, функций, функциональных блоков или программ используем команду “BOX”. Сразу после ее выполнения появляется блок с именем “AND”. Выбрав текстовое поле внутри этого блока, вы можете изменить его на имя любого другого оператора, функции, функционального блока или программы. Если новый блок имеет большее минимальное число входов, то будут добавлены новые входы. Если количество входов нового блока меньше, чем количество входов выбранного блока, то последние входы удаляются.

2. Для просмотра всех операторов, функции, функциональных блоков или программ, входящих в стандартную библиотеку, необходимо нажать клавишу F2 (при этом текстовое поле внутри блока AND должно быть выделено) (рис. 4.3).

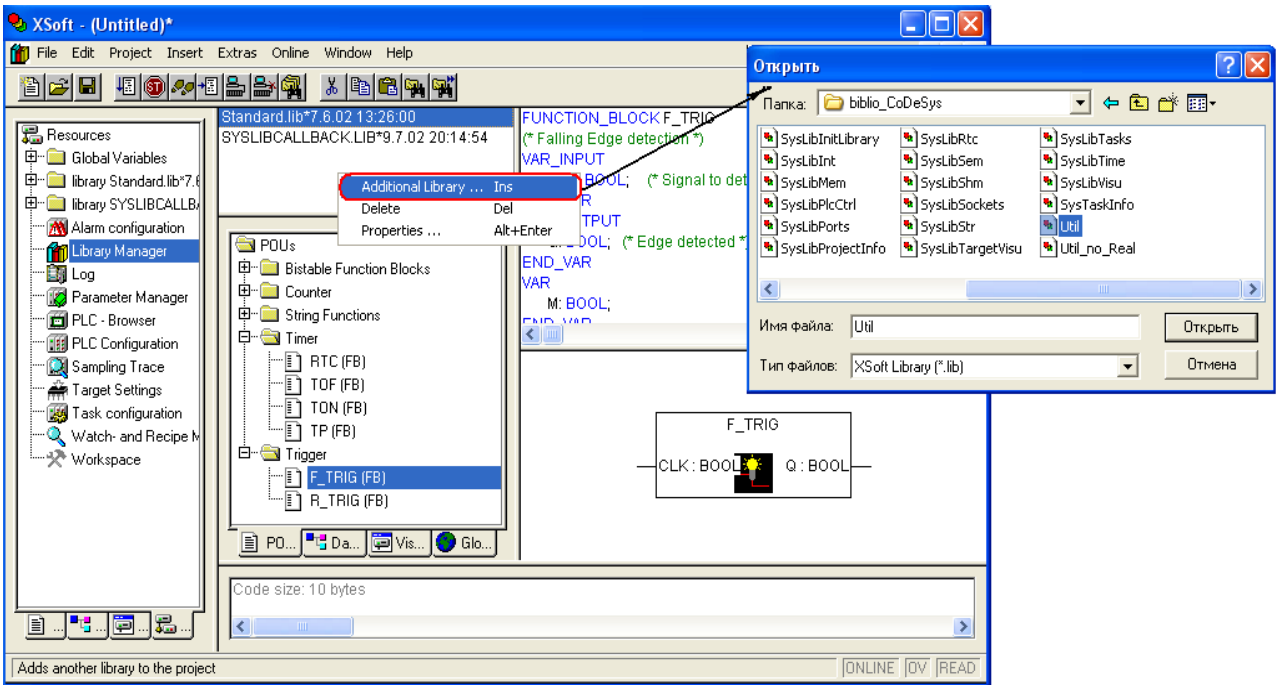
3. Если в предложенном перечне нет необходимого вам элемента, его можно получить путем добавления в список Менеджера библиотек (вкладка ресурсы) недостающую библиотеку (в состав которой входит этот элемент) (рис. 4.4).

4. Затем вставляем необходимое количество входов, выходов (для этого используем соответствующие команды, находящиеся в Панели инструментов).

5. Для того чтобы выбрать элемент, нужно щелкнуть по нему мышкой. Чтобы выбрать больше одного элемента, вы должны нажать клавишу <Shift> и выбирать нужные элементы или, щелкнув мышкой на свободном месте, растягивать получившийся прямоугольник. Команда “Extras” “Select all” сразу выбирает все элементы.



Р и с. 4.3. Выбор необходимого элемента, входящего в стандартную библиотеку



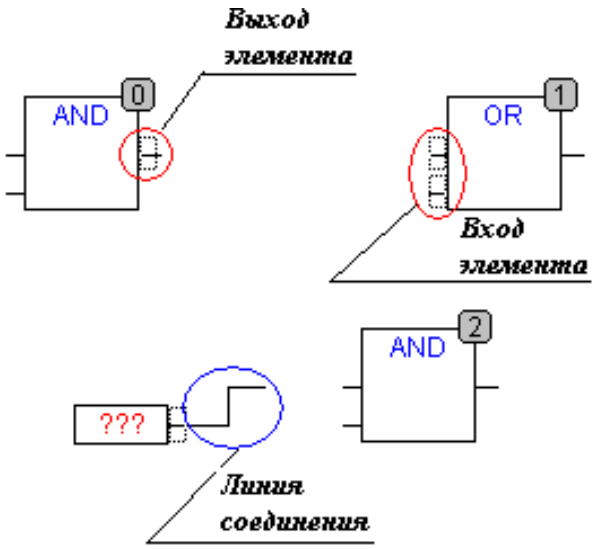
Р и с. 4.4. Добавление недостающей библиотеки в список Менеджера библиотек

6. Для того чтобы перемещать один или несколько элементов можно использовать клавиши перемещения, удерживая при этом клавишу <Shift>. Это можно сделать иначе: выберите элемент и переме-

щайте его, не отпуская левую клавишу мыши. Элементы перемещаются до тех пор, пока они не перекрывают другие элементы или не заходят за пределы экрана. В таких случаях элемент будет перемещен в начальную позицию, и вы услышите сигнал тревоги.

7. Выбранные элементы можно скопировать в буфер с помощью команды ”Edit” ”Copy” и вставить с помощью команды ”Edit” ”Paste”.

8. Вход одного элемента можно соединять с выходом другого. Выход одного элемента может соединяться сразу с несколькими входами других элементов. Для этого необходимо поместить указатель мыши на выход первого элемента, затем нажать левую кнопку мыши и, удерживая ее, перемещать курсор мыши до входа второго элемента. Линия соединения будет создана при перемещении курсора мыши. Этим же методом могут быть соединены вход и выход одного элемента (обратная связь) (рис.4.5).



Р и с. 4.5. Основные составляющие элементов (блоков),
возможности их соединения

9. Есть несколько способов удаления линии, соединяющей вход с выходом элементов. Выберите выход первого элемента или вход второго элемента и нажмите <Delete> или выполните команду “Edit” ”Delete”. Если выход связан с несколькими входами, то будут удалены все соединения. Поместите указатель мыши на вход первого элемента и, удерживая левую клавишу мыши, переместите его на свободную область экрана. Соединение будет удалено, как только вы отпустите кнопку мыши.

10. Автоматическая нумерация элементов схемы в порядке слева направо и сверху вниз. Такой порядок называется топологическим. При этом не имеют значения соединения элементов схемы, а важно лишь расположение элементов.

CFC в режиме Online

Мониторинг. Значения входов и выходов изображаются внутри прямоугольных элементов. Мониторинг констант не производится. Для не логических переменных границы элементов расширяются так, чтобы значения этих переменных были видны. Для логических переменных сами элементы и соединенные с ними линии изображаются синим, если значения переменных TRUE, и остаются черными, если значение переменных FALSE.

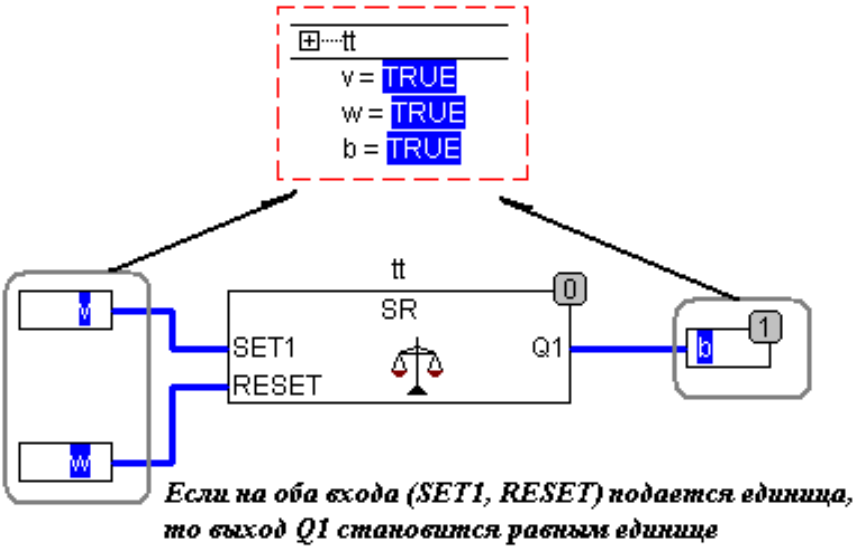
Внутренние логические соединения изображаются синим, если они передают значение TRUE, и черным в противном случае. Значения внутренних нелогических соединений можно увидеть в квадратиках на выходах элементов.

Элементы, входящие в стандартную библиотеку (Standart.lib)

Элементы с двумя устойчивыми состояниями (переключатели)

RS

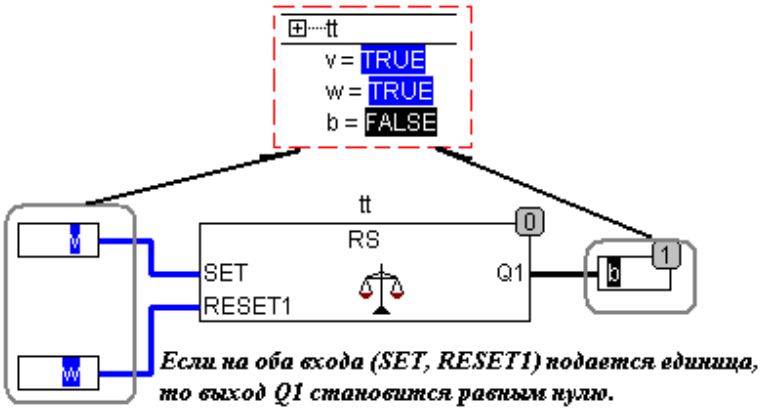
Функциональный блок (рис.4.6) реализует RS-триггер с доминирующим входом R (Reset, сбросить).



Р и с. 4.6. Функциональный блок реализует RS-триггер

SR

Функциональный блок (рис.4.7) реализует триггер с доминирующим входом S (Set, установить).



Р и с. 4.7. Функциональный блок реализует SR-триггер

Счетчики

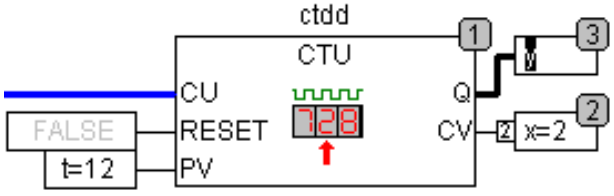
Уменьшающий счётчик (CTD)

Входы CD, LOAD и выход Q типа BOOL, вход PV и выход CV типа WORD.

По каждому фронту на входе CD (переход из FALSE в TRUE) выход CV уменьшается на 1. Когда счетчик достигнет 0, счет останавливается, выход Q переключается в TRUE. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE.

Увеличивающий счётчик (CTU)

Входы CU, RESET и выход Q типа BOOL, вход PV и выход CV типа WORD (рис. 4.8). По каждому фронту на входе CU (переход из FALSE в TRUE) выход CV увеличивается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет значения заданного PV. Счетчик CV сбрасывается в 0 по входу RESET = TRUE.



Р и с. 4.8. Увеличивающий счётчик (CTU)

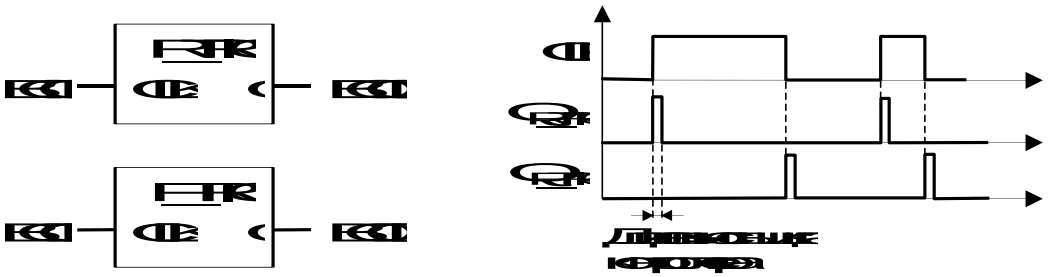
Обнаружение границы (рис.4.9).

В группу входят два блока:

R_TRIG, который реагирует на изменение логического сигнала на входе с «0» на «1»;

F_TRIG, который реагирует на изменение логического сигнала на входе с «1» на «0».

Реакция заключается в появлении логической единицы на выходе блока, которая удерживается в течение одного цикла расчёта программы (цикла контроллера).

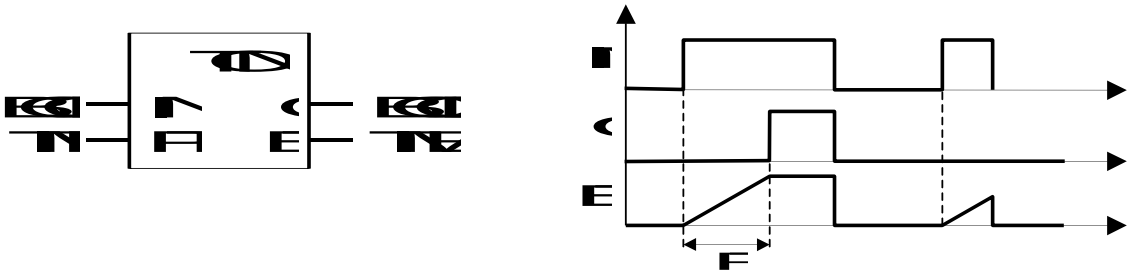


Р и с. 4.9. Обнаружение границ

Таймеры

TON

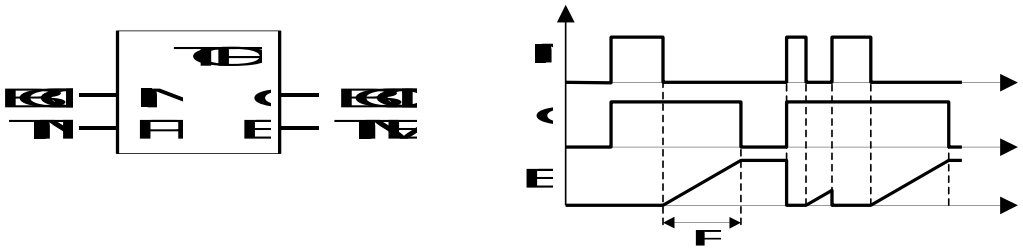
Функциональный блок TON реализует таймер с задержкой включения (рис.4.10). Логическая единица появляется на выходе блока Q спустя заданное время PT с момента появления «1» на входе IN. На выходе ET можно контролировать время, прошедшее с появления «1» на входе IN.



Р и с. 4.10. Функциональный блок, реализующий таймер с задержкой включения

TOF

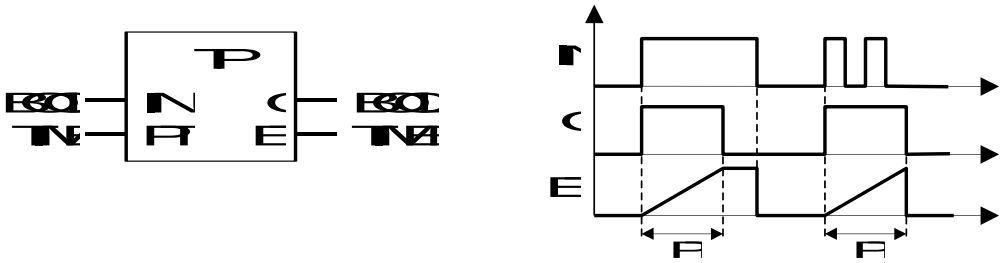
Функциональный блок TOF реализует таймер с задержкой выключения (рис.4.11). Логическая единица появляется на выходе блока Q одновременно с «1» на входе IN и снимается только по прошествии времени PT.



Р и с. 4.11. Функциональный блок, реализующий таймер с задержкой выключения

TP

Функциональный блок служит для генерации импульсов заданной длительности (рис.4.12).



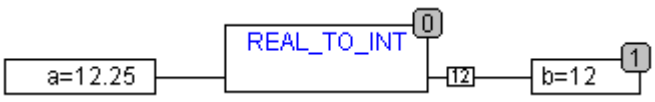
Р и с. 4.12. Генератор импульсов заданной длительности

Преобразователи

Функции, входящие в данную группу, служат для преобразования типов данных в процессе вычислений. Имена функций образуются следующим образом:

<тип «откуда»>_TO_<тип «куда»>,

где <тип «откуда»> – имя типа данных на входе функции; <тип «куда»> – имя типа данных на выходе функции, например, функция REAL_TO_INT преобразует значение типа REAL в значение типа INT, округляя его; функция TIME_TO_REAL преобразует значение типа TIME в значение типа REAL (рис. 4.13).



Р и с. 4.13. Функция, преобразующая типы данных

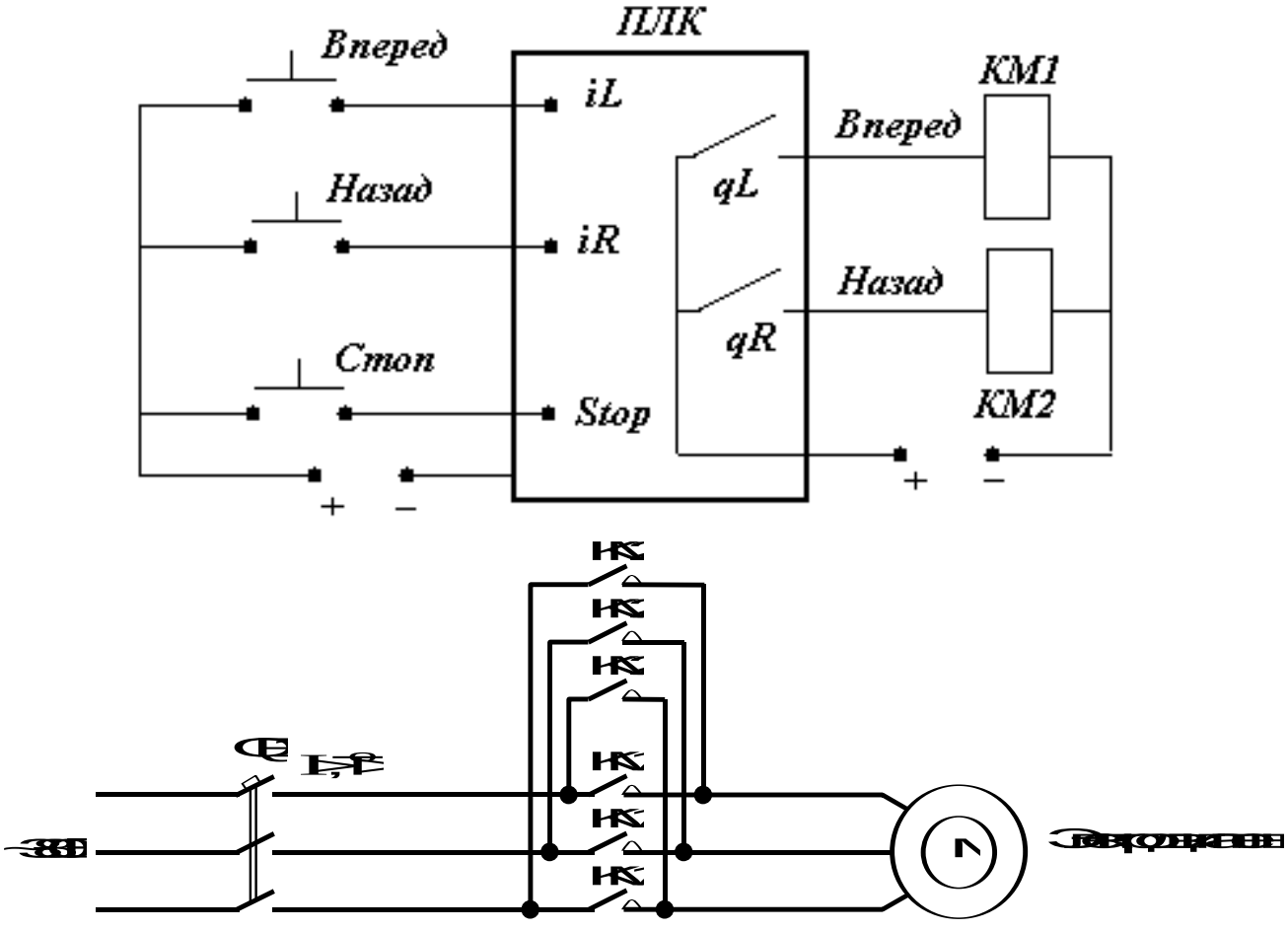
Работа с дискретными сигналами

Рассмотрим способы использования различных функциональных блоков, входящих в стандартную библиотеку, для работы с дискретными сигналами на конкретном примере.

Пример. Требуется разработать программу для контроллера, который реализует систему безопасного включения электродвигателя по командам от оператора.

Для управления вращением электродвигателя вперёд или назад, контроллер коммутирует напряжение на катушку соответствующего магнитного пускателя. Если включить одновременно оба магнитных пускателя, то это приведёт к короткому замыканию источника питания и его выходу из строя. Программа контроллера должна не допускать возникновения аварийной ситуации.

Электрическая схема управления электродвигателем приведена на рис. 4.14.



Р и с. 4.14. Электрическая схема управления электродвигателем:
QF1 – автоматический выключатель; KM1, KM2 – магнитный пускатель

Сформулируем правила функционирования системы управления вращением электродвигателя.

1. Из состояния «Стоп» по кнопке «Вперёд» или «Назад» подаётся напряжение на катушку соответствующего промежуточного реле KM1 или KM2, и двигатель начинает вращаться в заданную сторону.

2. Нажатие кнопки «Стоп» приводит к останову двигателя. Напряжение с катушек промежуточных реле снимается.

3. Если во время вращения двигателя в одну сторону приходит команда на вращение двигателя в другую сторону, то эта команда блокируется, и двигатель продолжает вращаться в том же направлении.

Команды на управление двигателем поступают на входы контроллера в виде кратковременных сигналов. Задача контроллера – удерживать двигатель в состоянии, заданном последней правильной командой до прихода новой правильной (не требующей блокировки) команды, т.е. необходимо «помнить» предыдущее состояние электродвигателя.

Рассматриваемая система имеет три устойчивых состояния: «Вперёд», «Назад» и «Стоп». Для кодирования трёх состояний требуется два элемента памяти. В качестве элементов памяти используются RS-триггеры (переключатели).

Разработка программы контроллера начинается с описания переменных, использующихся в проекте.

В редакторе языка CFC размещаем функциональные блоки, необходимые для реализации алгоритма: два RS-триггера и два блока логического умножения, для реализации функций возбуждения (SET1, SET2).

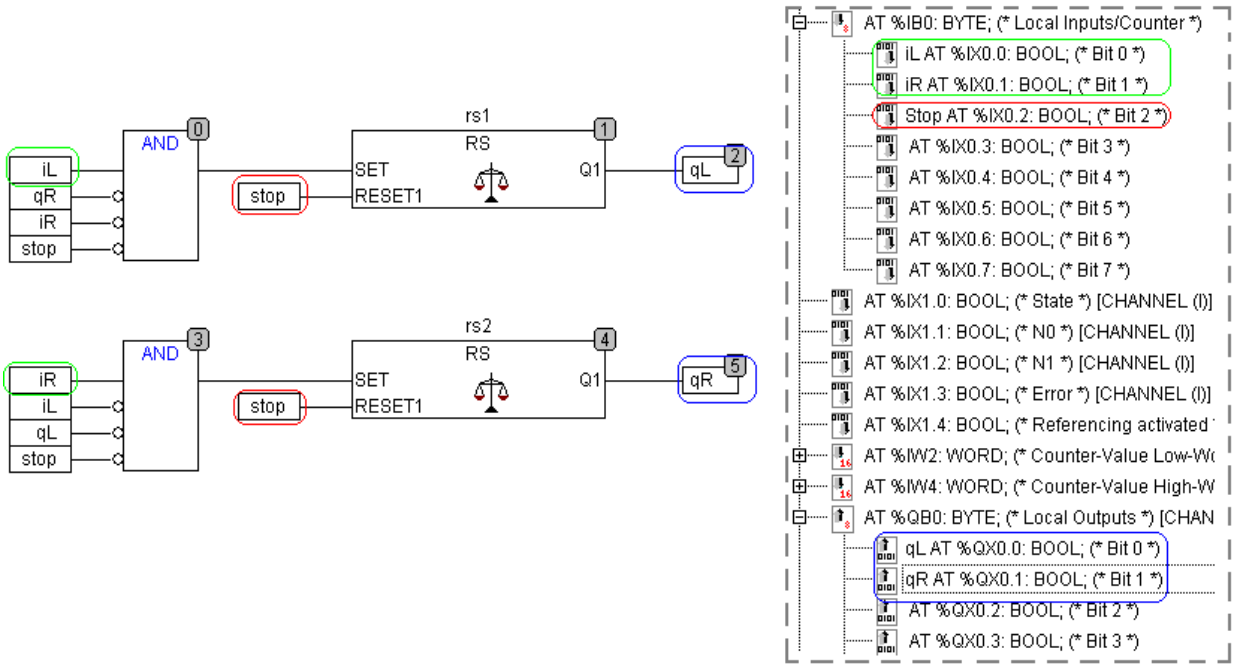
Далее выполняется связывание выходов и входов блоков и, наконец, привязывание переменных. Окончательный вид программы для управления включением электродвигателя представлен на рис. 4.15, на рис. 4.16 дано представление поставленной задачи в редакторе визуализации.

Элементы, входящие в библиотеку Util.lib

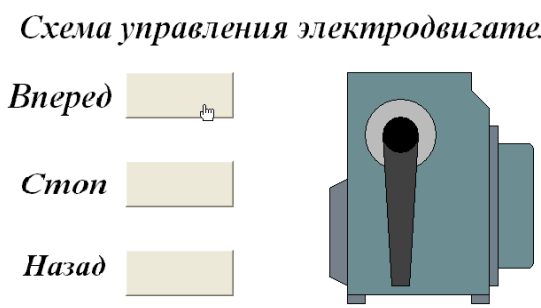
GEN

Функциональный блок “функциональный генератор”

Входы: перечисление MODE предопределенного типа GEN_MODE, BASE типа BOOL, PERIOD типа TIME, CYCLES и AMPLITUDE типа INT и RESET типа BOOL. Выход OUT типа INT.



Р и с. 4.15. Реализация данной схемы в редакторе CFC
с соответствующей конфигурацией входов/выходов ПЛК



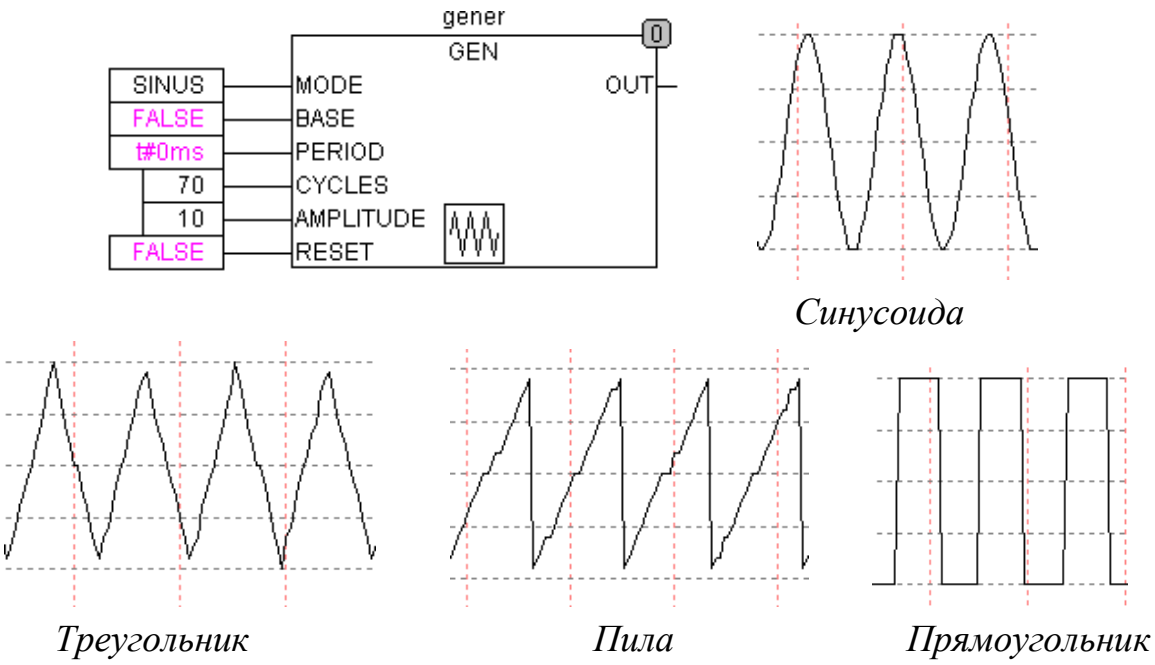
Р и с. 4.16. Представление поставленной задачи
в редакторе визуализации

Вход MODE задает вид генерируемой функции. Перечисление включает следующие значения: TRIANGLE – треугольник, SAWTOOTH_RISE и SAWTOOTH_FALL – пила, RECTANGLE – прямоугольник, SINUS и COSINUS – синусоиды.

BASE определяет представление единиц периода по времени (BASE=TRUE) или по числу циклов, т.е. по количеству вызовов функционального блока (BASE=FALSE).

Входы PERIOD или CYCLES определяют период выходного сигнала. Вход AMPLITUDE задает амплитуду сигнала.

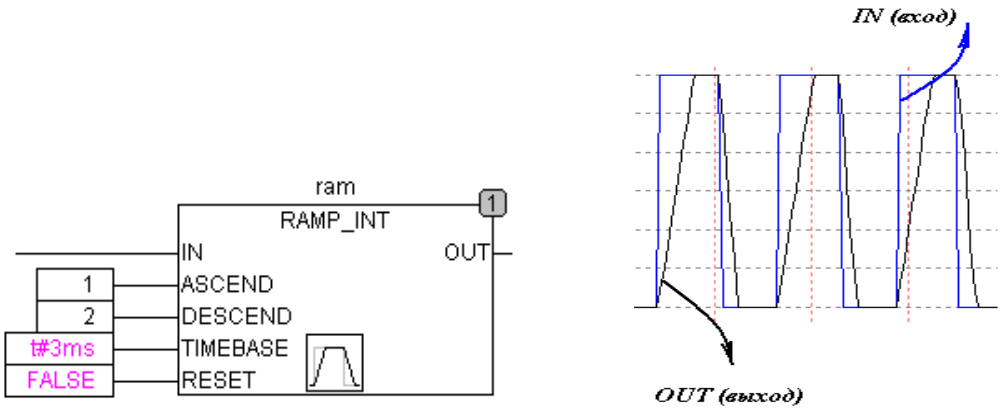
Сброс генератора (рис.4.17) происходит при установке RESET=TRUE.



Р и с. 4.17. Генератор сигналов

RAMP_INT

Функциональный блок RAMP_INT ограничивает скорость нарастания и спада сигнала (рис. 4.18).



Р и с. 4.18. Блок, ограничивающий скорость нарастания и спада сигнала

Три входа имеют тип INT: IN, входные данные, ASCEND и DESCEND, максимальное нарастание и спад за интервал, заданный TIMEBASE типа TIME. Установка двоичного входа RESET в TRUE вызывает сброс RAMP_INT в начальное состояние.

Выход OUT типа INT, выходные данные.

Если TIMEBASE равен t#0s, ASCEND и DESCEND задают ограничение изменения за один цикл (вызов блока) безотносительно времени.

RAMP_REAL

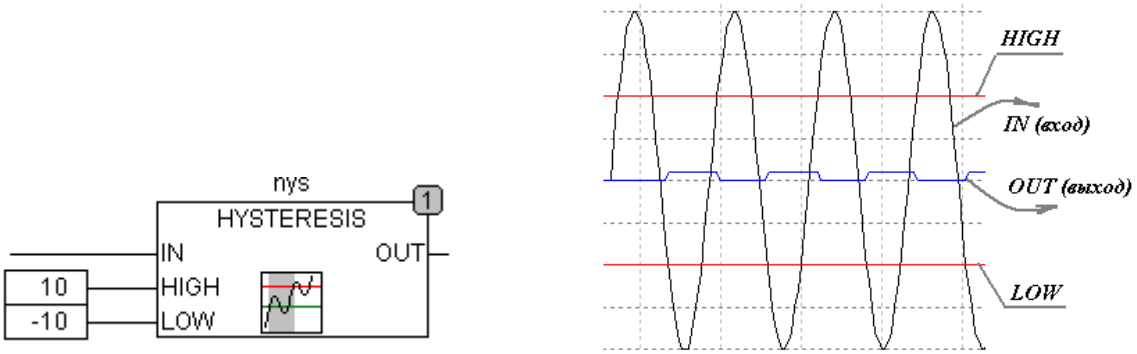
RAMP_REAL аналогичен RAMP_INT, за исключением того, что входы IN, ASCEND, DESCEND и выход OUT типа REAL.

HYSTERESIS

Аналоговый компаратор с гистерезисом (рис. 4.19).

Входы IN, HIGH и LOW типа INT. Выход OUT типа BOOL.

Если вход IN принимает значение, меньшее LOW, выход OUT устанавливается в TRUE. Если вход IN принимает значение, большее HIGH, то выход равен FALSE. В пределах от LOW до HIGH значение выхода не изменяется.



Р и с. 4.19. Аналоговый компаратор с гистерезисом

DERIVATIVE

Функциональный блок выполняет численное дифференцирование (рис. 4.20).

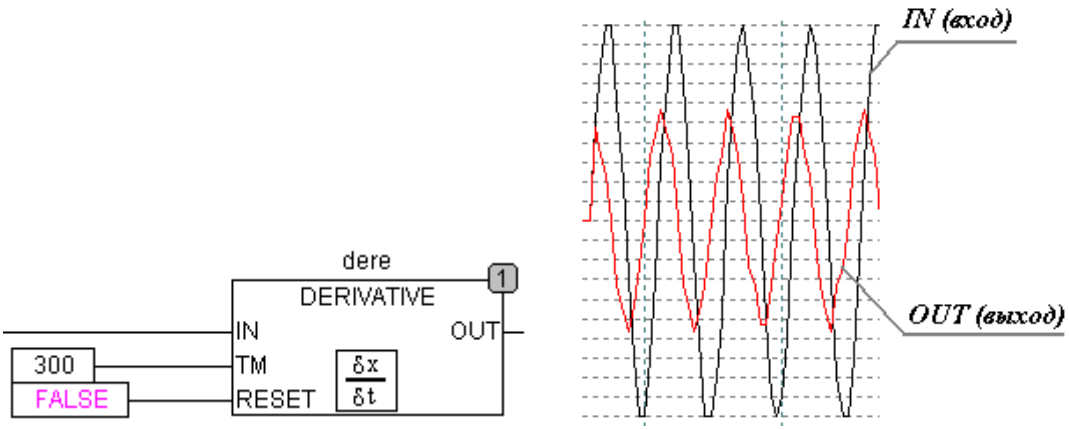
Аналоговый вход IN и выход OUT типа REAL. Вход TM задает время дифференцирования (как правило, в миллисекундах) имеет тип DWORD. В процессе сброса (RESET = TRUE) выход OUT равен нулю.

Алгоритм DERIVATIVE проводит аппроксимацию по четырем точкам, что снижает ошибки при наличии шума во входном сигнале.

INTEGRAL

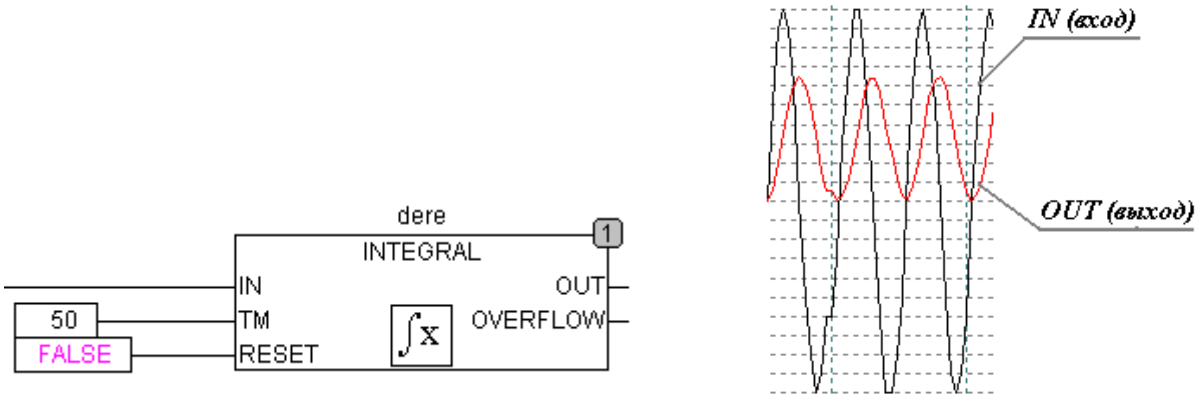
Функциональный блок выполняет численное интегрирование (рис. 4.21).

Аналоговый вход IN типа REAL. Вход TM типа DWORD задает длительность интегрирования (как правило, в миллисекундах). Вход RESET типа BOOL запускает интегрирование при установке в TRUE. Выход OUT типа REAL.



Р и с. 4.20. Дифференциатор

Алгоритм вычисления использует классический двухточечный метод трапеций.



Р и с. 4.21. Интегратор

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

Работа с непрерывными сигналами

1. Используя библиотеку Util.lib, сгенерировать в режиме эмуляции непрерывный сигнал (табл. 4.1).

Таблица 4.1

Вид генерируемой функции согласно варианту

Вид генерируемой функции	Номер варианта		
	1	2	3
	Синус	Косинус	Треугольник

2. Продифференцировать полученные сигналы. Используя аппарат визуализации, посмотреть каким сигнал был изначально, а каким стал после обработки.

3. Добавить блок HYSTERESIS, в зависимости от выбранной амплитуды сигнала установить значения HIGH и LOW. Пронаблюдать, как будет меняться сигнал на выходе данного блока.

4. Сгенерировать прямоугольные импульсы, используя соответствующий функциональный блок, ограничить скорость наростания и спада сигнала.

Работа с дискретными сигналами

5. Разработать программу для управления видеомагнитофоном. Программа должна исходя из текущего состояния магнитофона переводить объект в новое состояние по команде пользователя. Кнопки управления «Play», «Stop», «Forward», «Backward», «Record» должны быть подключены к дискретным входам контроллера. Команда на переход в новое состояние формируется на дискретных выходах контроллера. Выходы контроллера используются для управления переходом видеомагнитофона в одно из состояний: «Play», «Stop», «Forward», «Backward», «Record». Каждое состояние соответствует одному выходу. Переход из одного состояния в другое осуществляется только через состояние «Stop». Программа должна содержать блокировки недопустимых переходов (из «Play» в «Record» и т.п.).

6. Реализовать в редакторе визуализации данную задачу (с наглядным представлением кнопок управления и всех возможных состояний).

СОДЕРЖАНИЕ ОТЧЕТА

1. Изображение соединенных функциональных блоков (в соответствии с требованиями 1-3 пунктов задания) в Online режиме.
2. Три графика, соответствующие выходам функциональных блоков, совмещенных в одном.
3. Изображение соединения двух функциональных блоков (GEN, HYSTERESIS) в реальном режиме времени и графики, отображающие соответствующие сигналы на их выходах.
4. Программа, написанная на языке CFC, для управления магнитофоном и соответствующая ей визуализация.

Практическая работа № 5
РЕАЛИЗАЦИЯ АЛГОРИТМОВ ОБРАБОТКИ АНАЛОГОВЫХ СИГНАЛОВ НА ЯЗЫКЕ ST

Цель работы – знакомство с операторами и операциями языка структурированного текста ST. Изучение принципов работы с аналоговыми сигналами. Знакомство с типами данных среды CoDeSys, библиотеками преобразования типов.

Структурированный текст (ST)

ST представляет собой набор инструкций высокого уровня, которые могут использоваться в условных операторах ("IF…THEN…ELSE") и в циклах (WHILE…DO).

Пример:

```
IF value < 7 THEN
WHILE value < 8 DO
value:=value+1;
END_WHILE;
END_IF;
```

Выражение – это конструкция, возвращающая определенное значение после его вычисления. Выражение состоит из операторов и операндов. Операндом может быть константа, переменная, функциональный блок или другое выражение.

Вычисление выражений выполняется согласно правилам приоритета. Оператор с самым высоким приоритетом выполняется первым, оператор с более низким приоритетом – вторым и т.д., пока не будут выполнены все операторы.

Операторы с одинаковым приоритетом выполняются слева направо.

В табл. 5.1 приведен список ST операторов, расположенных в порядке приоритета.

Ниже приведён список операторов языка ST.

Оператор присваивания

Перед оператором присваивания находится операнд (переменная или адрес), которому присваивается значение выражения, стоящего после оператора присваивания.

Таблица 5.1

Основные операторы языка ST

Оператор	Название	Приоритет
( )	Скобки	1
FUNCNAME (список параметров)	Вызов функции FUNCNAME – имя вызываемой функции	2
–	Смена знака	3
NOT	Отрицание	3
*	Умножение	5
/	Деление	5
MOD	Остаток от деления	5
+	Сложение	6
–	Вычитание	6
<	Меньше	7
>	Больше	7
<=	Меньше или равно	7
>=	Больше или равно	7
=	Равно	8
<>	Не равно	8
&, AND	Поразрядное логическое умножение	9
XOR	Поразрядное исключающее ИЛИ	10
OR	Поразрядное логическое сложение	11 (низший)

Формат оператора

<переменная> := <выражение>

При использовании оператора присваивания необходимо следить за соблюдением следующих требований.

Переменная должна быть выходная (префикс адреса – 0x или 4x) или внутренняя.

Переменная и выражение должны быть одного типа.

Пример:

Var1: = Var2 * 10;

После выполнения этой операции Var1 принимает значение в десять раз большее, чем Var2.

Вызов функционального блока в ST

Функциональный блок вызывается с помощью имени экземпляра функционального блока и списка входных параметров с присваиванием данных в круглых скобках. В следующем примере вызывается

таймер с параметрами IN и PT. Значение выходной переменной Q присваивается переменной A.

Пример:

CMD_TMR (IN: = %IX5, PT: = 300);

A: =CMD_TMR.Q

При использовании функциональных блоков в программе на языке ST должны быть выполнены следующие действия.

1. В разделе описаний секции должен быть описан объект соответствующего типа.

2. В теле программы нужно выполнить вызов объекта с указанием значений или переменных, подаваемых на вход функционального блока.

3. Присвоить значения с выходов объекта переменным проекта.

Оператор условного перехода

IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF

Используя оператор IF, можно проверить условие, и в зависимости от этого условия выполнить какие-либо действия.

Синтаксис:

IF <Boolean_expression1> **THEN**
    <IF_instructions>
    ELSIF <Boolean_expression2> **THEN**
        <ELSIF_instructions1>
            ……
    ELSIF <Boolean_expression n> **THEN**
        <ELSIF_instructions n-1>
    ELSE
        <ELSE_instructions>
END_IF;

Если <Boolean_expression1> возвращает истину, тогда <IF_Instructions> выполняется. В противном случае будут выполняться остальные логические выражения одно за другим, пока одно из них не возвратит истину. Тогда выполняются инструкции, стоящие после этого логического выражения до следующего ELSIF или ELSE.

Если все логические выражения ложны, то выполняются инструкции, стоящие после ELSE.

Пример:

IF temp < 17

THEN heating_on: = TRUE;

ELSE heating_on: = FALSE;

END_IF

В этом примере нагревание (heating) включается, когда температура опустится ниже 17°C, иначе оно останется выключенным.

Оператор выбора

CASE ... OF ... ELSE ... END_CASE

С помощью инструкции CASE можно нескольким различным значениям целочисленной переменной сопоставить различные инструкции.

Синтаксис:

CASE <Var1> **OF**

<Value1>: <Instruction 1>

<Value2>: <Instruction 2>

<Value3, Value4, Value5>: <Instruction 3>

<Value6 .. Value10>: <Instruction 4>

.....

<Value n>: <Instruction n>

ELSE

<ELSE instruction>

END_CASE;

Инструкция CASE выполняется согласно следующим правилам.

• Если переменная <Var1> имеет значение <Value i>, то выполняется инструкция <Instruction i>.

• Если <Var1> не принимает ни одного из указанных значений, то выполняется <ELSE Instruction>.

• Чтобы одна и та же инструкция выполнялась при различных значениях переменной <Var1>, необходимо перечислить эти значения через запятую.

• Чтобы одна и та же инструкция выполнялась для целого диапазона значений, необходимо указать начальное и конечное значения, разделенные двумя точками.

Пример:

```
CASE INT1 OF
    1, 5: BOOL1 : = TRUE;
    BOOL3 : = FALSE;
    2: BOOL2 : = FALSE;
    BOOL3 : = TRUE;
    10. 20: BOOL1: = TRUE;
    BOOL3:= TRUE;
ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Оператор цикла с параметром

FOR ... TO ... BY ... DO ... END_FOR

С помощью FOR можно программировать повторяющиеся процессы.

Синтаксис:

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> BY <Step size> DO
    <Instructions>
END_FOR
```

<Instructions> выполняются, пока счетчик <INT_Var> не больше <END_VALUE>. Это условие проверяется перед выполнением <Instructions>, поэтому раздел <Instructions> не выполняется, если <INIT_VALUE> больше <END_VALUE>.

Всякий раз, когда выполняются <Instructions>, значение <INIT_VALUE>, увеличивается на <Step_size>.

<Step_ size> может принимать любое целое значение. По умолчанию шаг устанавливается равным 1.

Пример:

FOR Counter: =1 **TO** 5 **BY** 1 **DO**

Var1: =Var1*2;

END_FOR;

Erg:=Var1;

В этом примере предполагается, что начальное значение Var1 равно 1. После выполнения цикла эта переменная будет равна 32.

Оператор цикла с предусловием

WHILE ... DO ... END_WHILE

Цикл WHILE может использоваться, как и цикл FOR, с тем лишь различием, что условие выхода определяется логическим выражением. Это означает, цикл выполняется, пока верно заданное условие.

Синтаксис:

WHILE <Boolean expression>

<Instructions>

END_WHILE

Раздел <Instructions> выполняется циклически до тех пор, пока <Boolean_expression> дает TRUE. Если <Boolean_expression> равно FALSE уже при первой итерации, то раздел <Instructions> не будет выполнен ни разу. Если <Boolean_expression> никогда не примет значение FALSE, то раздел <Instructions> будет выполняться бесконечно.

Пример:

WHILE counter<>0 **DO**

Var1: = Var1*2;

counter := counter-1;

END_WHILE

Оператор цикла с постусловием

REPEAT ... UNTIL ... END_REPEAT

Цикл REPEAT отличается от цикла WHILE тем, что первая проверка условия выхода из цикла осуществляется, когда цикл уже выполнился 1 раз. Это означает, что независимо от условия выхода цикл выполняется хотя бы один раз.

Синтаксис:

REPEAT

<Instructions>

UNTIL

<Boolean expression>

END_REPEAT

Раздел <Instructions> выполняется циклически до тех пор, пока <Boolean_expression> дает TRUE. Если <Boolean_expression> равно FALSE уже при первой итерации, то раздел <Instructions> не будет выполнен один раз. Если <Boolean_expression> никогда не примет значение FALSE, то раздел <Instructions> будет выполняться бесконечно.

Оператор RETURN

Оператор RETURN позволяет выйти из POU, например, в зависимости от условия.

Оператор EXIT

Если EXIT встречается в циклах FOR, WHILE, REPEAT, то цикл заканчивает свою работу независимо от значения условия выхода.

Вообще, операторы цикла довольно редко используются в программах промышленных контроллеров, поскольку программа и так выполняется в цикле контроллера. Некорректное использование операторов цикла может привести к бесконечному увеличению длительности цикла контроллера и, как следствие, срабатыванию сторожевого таймера, переводящего ПЛК в СТОП-режим.

main и PLC_PRG

В основе любой контроллерной программы заложен цикл:

```
WHILE(1)
{
 ReadInputs();
 PLC_PRG();
 WriteOutputs();
}
```

В CoDeSys мы видим только PLC_PRG. Сам главный цикл спрятан в системе исполнения. Это наиболее тонкий момент, который нужно очень четко себе представлять. В многозадачных проектах система исполнения создает независимый главный цикл для каждой задачи.

Типы данных CoDeSys

Тип данных определяет род информации и методы ее обработки и хранения, количество выделяемой памяти. Программист может непосредственно использовать элементарные (базовые) типы данных или создавать собственные (пользовательские) типы на их основе.

Элементарные типы данных

Логический (BOOL)

BOOL логический тип данных. Переменная может принимать 2 значения ИСТИНА (TRUE) или ЛОЖЬ (FALSE). Занимает 8 бит памяти, если не задан прямой битовый адрес [10].

Целочисленные

BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, и UDINT – все это целочисленные типы. Они отличаются различным диапазоном сохраняемых данных и, естественно, различными требованиями к памяти. Подробно данные характеристики представлены в табл.5.2.

Таблица 5.2

Целочисленные типы данных

Тип данных	Нижний предел	Верхний предел	Размер памяти
BYTE	0	255	8 Бит
WORD	0	65535	16 Бит
DWORD	0	4294967295	32 Бит
SINT	-128	127	8 Бит
USINT	0	255	8 Бит
INT	-32768	32767	16 Бит
DINT	-2147483648	2147483647	32 Бит
UINT	0	65535	16 Бит
UDINT	0	4294967295	32 Бит

Очевидно, присвоение данных большего типа переменной меньшего типа может приводить к потере информации.

Рациональные

REAL и LREAL данные в формате с плавающей запятой, используются для сохранения рациональных чисел. Для типа REAL необходимо 32 бита памяти и 64 для LREAL.

Диапазон значений REAL от: 1.175494351e-38F до 3.402823466e+38F

Диапазон значений LREAL от: 2.2250738585072014e-308 до 1.7976931348623158e+308

Строки

Строковый тип STRING представляет строки символов. Максимальный размер строки определяет количество резервируемой памяти и указывается при объявлении переменной. Размер задается в круглых или квадратных скобках. Если размер не указан, принимается размер по умолчанию – 80 символов.

Время и дата

TIME представляет длительность интервалов времени в миллисекундах. Максимальное значение для типа TIME : 49d17h2m47s295ms (4194967295 ms).

TOD содержит время суток начиная с 0 часов. Диапазон значений TOD от: 00:00:00 до 23:59:59.999.

DATE содержит календарную дату начиная с 1 января 1970 года. Диапазон значений от: 1970-00-00 до 2106-02-06.

DT содержит время в секундах начиная с 0 часов 1 января 1970 года. Диапазон значений от: 1970-00-00-00:00:00 до 2106-02-06-06:28:15.

Типы TIME, TIME_OF_DAY (сокр. TOD), DATE и DATE_AND_TIME (сокр. DT) сохраняются физически как DWORD.

Преобразования типов данных

Имена функций образуются следующим образом:

<тип «откуда»>_TO_<тип «куда»>,

где <тип «откуда»> – имя типа данных на входе функции, <тип «куда»> – имя типа данных на выходе функции, например, функция REAL_TO_INT преобразует значение типа REAL в значение типа INT, округляя его; функция TIME_TO_REAL преобразует значение типа TIME в значение типа REAL и т.д.

Пример реализации алгоритма первичного преобразования аналогового сигнала на языке ST

Постановка задачи

Необходимо разработать программу первичной обработки измерительной информации, получаемой с датчика температуры, с унифицированным токовым выходам 4…20 мА и линейной зависимостью тока от измеряемой температуры: $I=k\cdot P+b$. Диапазон изменения измеряемой величины, соответствующий диапазону изменения токового сигнала – 0 … 400 С. Изменение сигнала на выходе АЦП 6400…32 000. Диапазон нормального режима работы оборудования: 50…350 С.

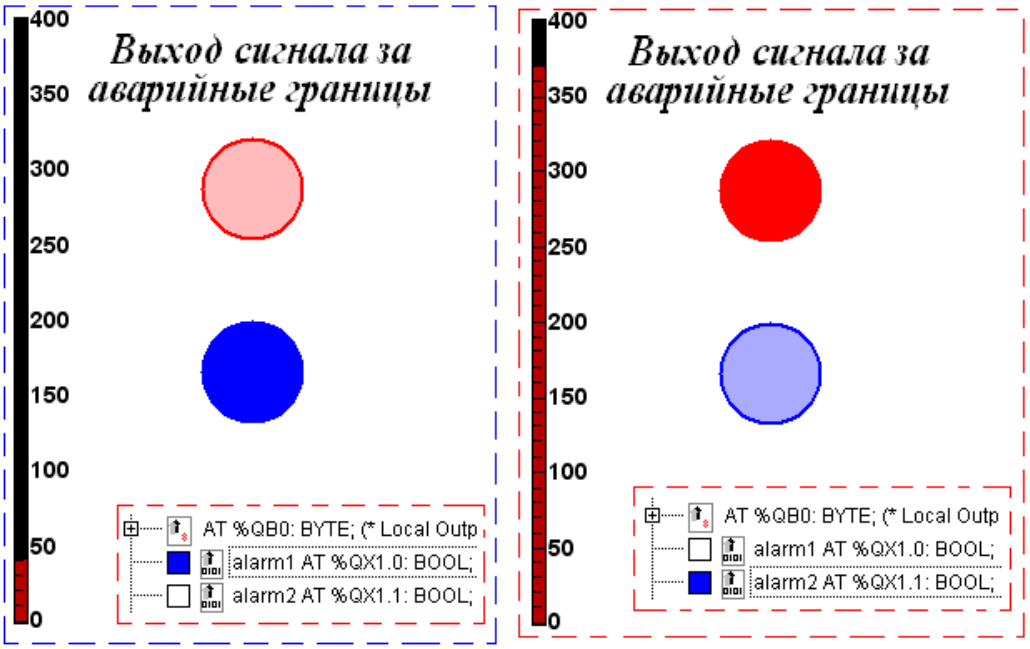
Решение

Коэффициенты линейного преобразования k и b находятся из системы уравнений

Программа на языке ST, реализующая алгоритм линейного преобразования с проверкой аварийных границ приведена на рис.5.1.

Р и с. 5.1. Реализация алгоритма линейного преобразования входного аналогового сигнала на языке ST

Ниже приведена реализация поставленной задачи в редакторе визуализации (рис.5.2).



Р и с. 5.2. Реализация поставленной задачи в редакторе визуализации

Задание для самостоятельного решения

1. Создать новый проект, выбрав в качестве языка программирования – ST. Подобрать конфигурацию, соответствующую нашим техническим средствам.

2. Дать символические имена всем аналоговым входам и дискретным битовым выходам (для написания программы следует использовать один из аналоговых сигналов, поступающих с термопар. Их адреса – %IW6, %IW10, %IW14. Какой из трех объектов рассматривать, следует уточнить у преподавателя).

3. После выбора аналогового входа, следует присвоить его значение некоторой переменной типа REAL (в примере, рассмотренном выше, эта переменная Input). Используя соответствующий БРУ установить ИМ в крайнее левое положение: зафиксировать значение измеряемого сигнала (предварительно дождавшись, когда сигнал примет некоторое установившееся значение), затем провести те же измерения, но ИМ при этом должен быть установлен в крайнем правое положении.

4. Когда ИМ находиться в закрытом положении – температура объекта управления будет равна температуре окружающей среды (примерно 25°С). При 100% открытии ИМ температуру объекта следует измерить, используя тестер и градуировочную таблицу соответствия сигнала в мV – сигналу в 0С.

5. После того как был получен диапазон изменения значения сигнала на выходе АЦП и соответствующий ему диапазон, в котором изменяется температура объекта, вы можете приступить к реализации алгоритма первичного преобразования аналогового сигнала на языке ST.

6. Пусть диапазон нормальной работы оборудования 35…300°С. Если значение измеряемой величины выходит за пределы нормальной работы оборудования, должна сработать аварийная сигнализация.

7. Реализовать поставленную задачу в редакторе визуализации.

8. Зная диапазон изменения цифрового сигнала на выходе АЦП и соответствующий ему диапазон токового сигнал на НП, рассчитать линейные коэффициенты. Смоделировать НП в редакторе визуализации.

СОДЕРЖАНИЕ ОТЧЕТА

1. Изображение дерева конфигурации, соответствующее аппаратным средствам нашей системы, до непосредственного соединения с контроллером (все символические имена должны быть присвоены).
2. Данные с тестера, соответствующие им расчетные значения температур.
3. Программа на языке ST, реализующая алгоритм первичного преобразования аналогового сигнала.
4. Эта же программа, но доработанная с учетом диапазона нормальной работы оборудования.
5. График изменения температуры во времени, средствами редактора визуализации.
6. Программа на языке ST, реализующая НП. График изменения во времени аналогового сигнала, поступающего с НП, средствами редактора визуализации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004.
2. Петров И.В. CoDeSys 3.0 – новый уровень инструментов программирования ПЛК. СТА №2, 2005.
3. Петров И.В. Отладка прикладных ПЛК программ в CoDeSys // Промышленные АСУ и контроллеры. 2006. № 2.
4. Пастушенков Д.В., Петров И.В Программируем временные сложности. Промышленные АСУ и контроллеры. № 7,8,9. М.: НАУЧТЕХИЗДАТ, 2004.
5. Зюбин В.Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. 2005. № 11.
6. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб: Наука, 2000.
7. Альтерман И.З., Шалыто А.А. Формальные методы программирования логических контроллеров // Промышленные АСУ и контроллеры. 2005. № 10.
8. Тэллес М., Хсих Ю. Наука отладки / Пер. с англ. М.: КУДИЦ-ОБРАЗ, 2003.
9. Руководство пользователя по программированию ПЛК в CoDeSys 2.3. 3S – Smart Software Solutions GmbH.
10. Визуализация CoDeSys. Дополнение к руководству пользователя по программированию ПЛК в CoDeSys 2.3 3S – Smart Software Solutions GmbH.
11. Христенсен, Дж. Х. Знакомство со стандартом на языки программирования PLC: IEC 1131-3 (МЭК 1131-3) [Электронный ресурс]/ Дж. Х. Христенсен. [2004]. Режим доступа: http://www.asutp.ru.