

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

**Лабораторный практикум по
программируемым логическим
контроллерам**

Электронные методические указания
для студентов по выполнению лабораторной работы

Самара

2010

УДК 629.7.064

Составители: **Игонин Андрей Александрович,**
Крючков Александр Николаевич,
Илюхин Владимир Николаевич,
Гимадиев Асгат Гатъатович

Рецензент: канд. техн. наук, доц. Свербилов В. Я.

Лабораторный практикум предназначен для выполнения лабораторных работ по дисциплинам: «Мехатроника», «АСУТП», «Электроавтоматика» студентами 5 курса специальности 150802.65 «Гидравлические машины, гидроприводы и гидропневмоавтоматика» и для магистрантов дневного отделения СГАУ, обучающихся в рамках магистерской программы «Мехатронные пневмогидравлические агрегаты и системы» по направлению 160700.68 «Двигатели летательных аппаратов».

Методические указания разработаны на кафедре автоматических систем энергетических установок.

© Самарский государственный
аэрокосмический университет, 2010

Введение

В наше время программируемые логические контроллеры (ПЛК) в промышленности распространены не менее, чем персональные компьютеры в быту и офисе. Они точно так же смогли стать универсальным инструментом для решения многих задач; особенно широко ПЛК применяются в задачах автоматизации производства и автоматического управления техническими объектами.

Для современного инженера, работающего в сфере машиностроения, а особенно в сфере автоматизации и автоматике, подготовка в вопросах эксплуатации и программирования ПЛК, а также построения систем на их базе является неотъемлемой частью высшего инженерного образования.

С одной стороны, современный уровень развития ПЛК позволяет достаточно эффективно применять его в решении производственных задач инженерам, не обладающим высокой квалификацией инженера-программиста помимо своей основной «механической» специальности, но с другой стороны, качество решения задач автоматизации возрастает с повышением квалификации инженера, и мастерское владение таким мощным инструментом, как микропроцессорная система вообще и ПЛК в частности, существенно увеличивает ценность специалиста на рынке труда. В современных условиях интенсивного восстановления производственных мощностей такие специалисты всегда востребованы.

Учебное пособие представляет собой руководство по выполнению лабораторных работ по курсу «Электроавтоматика» и содержит подробные описания решения типовых задач, демонстрирующих различные аспекты программирования ПЛК. В конце некоторых лабораторных работ приводятся дополнительные задачи без решений, которые по усмотрению преподавателя тоже могут быть рассмотрены на занятиях. Данное пособие предназначено для использования вместе с лекционным курсом.

1. Настройка CoDeSys. Новый проект в среде «CoDeSys». Построение таблиц истинности логических операций.

1.1. Общие сведения, установка среды программирования.

Цель работы: научиться первоначальным настройкам среды программирования и контроллера и элементарным приемам работы. Запустить первую программу.

Программирование ПЛК осуществляется с помощью персонального компьютера, который соединяется с ПЛК посредством кабеля, подключаемого к последовательному порту (на ПК обозначен, как COM), сетевому разъему ПК (так называемый *Ethernet*), или USB-порту.

Для программирования ПЛК используется специальный пакет программ, в который входят:

- компилятор,
- загрузчик программ,
- среда программирования (редактор программ с функциями вызова компилятора, загрузчика, отладчика и справочной системы, а также с возможностью использования набора стандартных библиотек процедур и некоторыми дополнительными возможностями, изучаемыми в последующих лабораторных работах),
- справочная система,
- набор стандартных библиотек подпрограмм для ПЛК.

В данных работах будут использованы ПЛК «ОВЕН-150» и среда программирования «CoDeSys». Данная среда программирования является распространяемой свободно (в том числе и для коммерческого использования) и используется для программирования не только ПЛК «ОВЕН», но и многих других типов ПЛК.

Для начала требуется установить «CoDeSys». Дистрибутив можно загрузить с сайта www.oven.ru; установка

ничем не отличается от установки других приложений. Основные экраны установки показаны на рисунках 1...3.

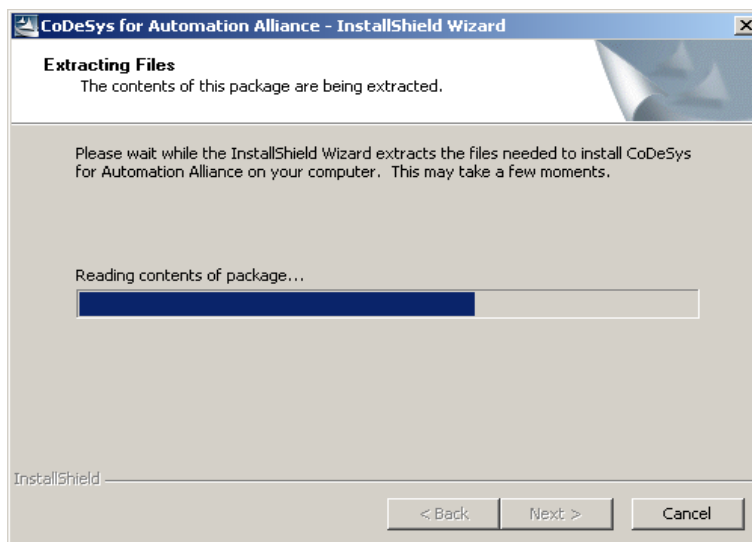


Рисунок 1. Установка «CoDeSys»: распаковка.

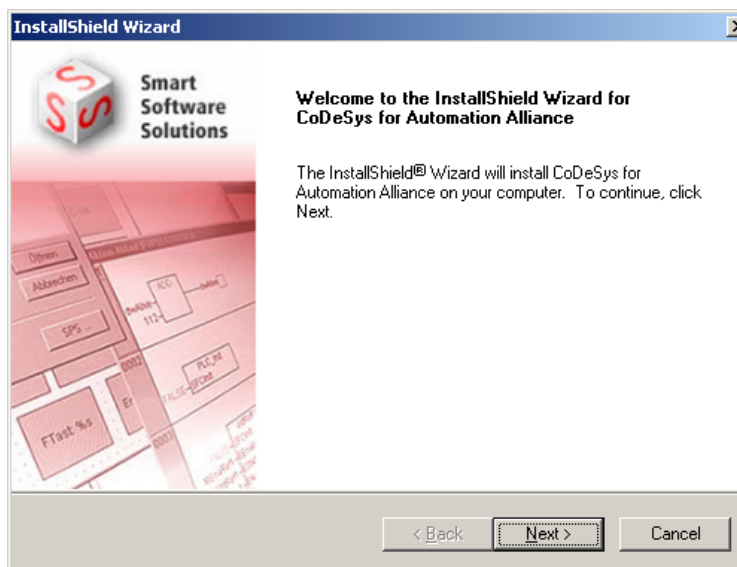


Рисунок 2. Установка «CoDeSys»: начало установки.

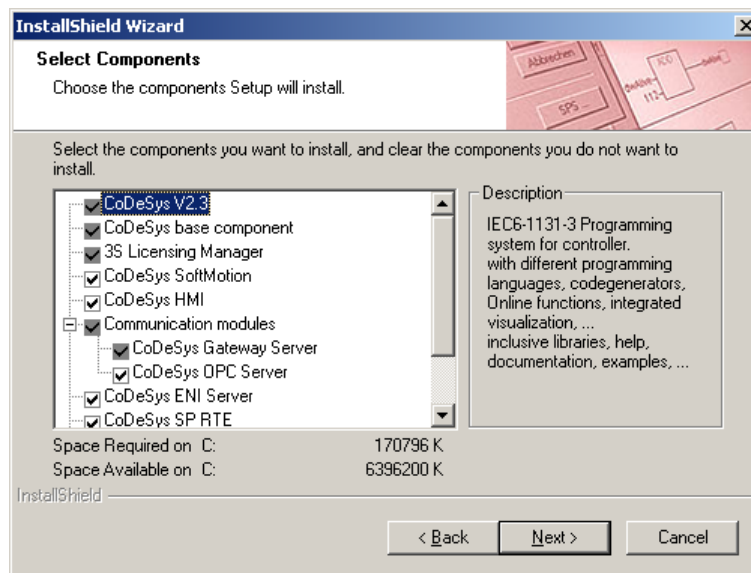


Рисунок 3. Установка «CoDeSys»: выбор компонентов, стандартный набор.

После установки пакета программ необходимо перезагрузить компьютер, некоторые компоненты системы «CoDeSys» начинают работать в процессе запуска операционной системы «Windows». Для использования «CoDeSys» наиболее подходят операционные системы «Windows 2000» SP4, «Windows XP» SP2, или SP3.

Дальнейшая настройка пакета сводится к установке так называемого профиля целевой платформы — *target-файла*. В этом файле содержится информация для компилятора о том, как правильно создавать загрузочные модули для контроллера определенного вида. Для каждого вида контроллеров, программируемых с помощью «CoDeSys» должен существовать свой *target-файл*.

В комплекте «CoDeSys» существует специальная программа установки *target-файлов*. Как ее запускать видно из рисунка 4. Вид основного окна этой программы представлен на рисунке 5.

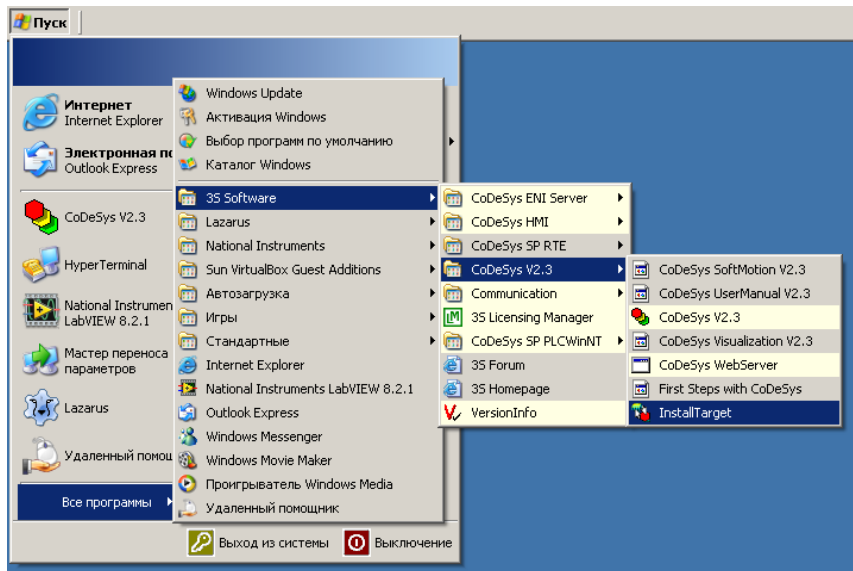


Рисунок 4. Установка target-файла: запуск программы.

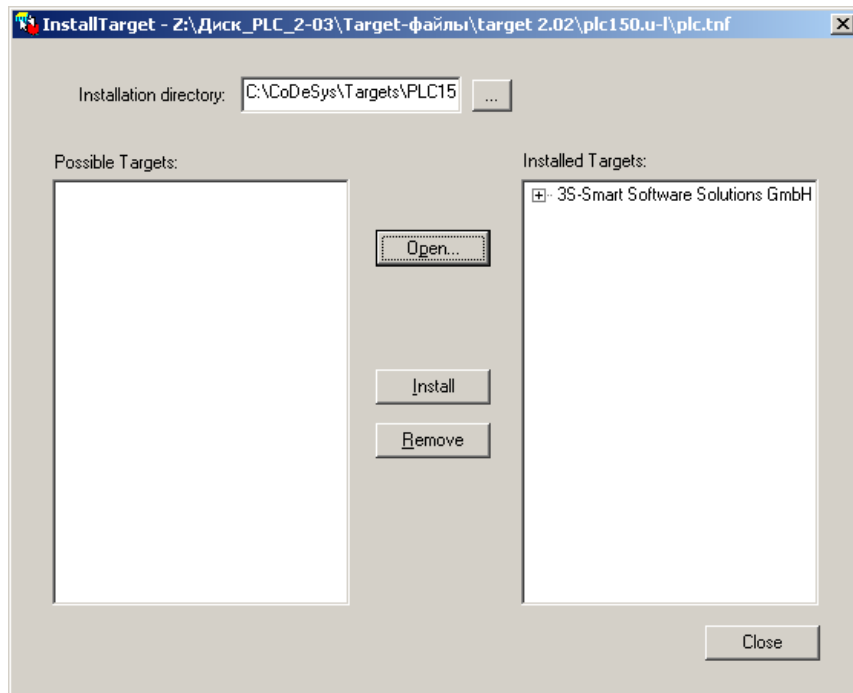


Рисунок 5. Установка target-файла: вид основного окна.

Путь к каталогу, в который будут устанавливаться target-файлы, указан в верхней строке, обозначенный, как Installation directory, можно вписать его от руки, или, нажав на кнопку «...» справа, выбрать в стандартном диалоговом окне задания пути.

Основное пространство окна поделено на три части: слева расположены возможные виды целевых устройств для установки (информация о них содержится в target-файлах), справа — уже установленные виды целевых устройств, для которых можно создавать проекты на «CoDeSys», а в центре — кнопки открытия target-файлов и установки устройств. Для открытия нового target-файла нужно нажать кнопку «Open», появится стандартное диалоговое окно открытия файлов. Далее требуется перейти в папку (на диск) с устанавливаемыми target-файлами, найти нужный файл и открыть его — при этом в левой части окна появится информация об видах ПЛК, содержащихся в этом файле. Эта информация представлена в иерархическом виде, пользователю требуется открыть нужный элемент иерархического дерева и выбрать требуемое название целевой платформы ПЛК, после этого нужно нажать на кнопку «Install». Название появится в аналогичном списке уже установленных целевых платформ, расположенном справа. В нашем случае выбирается и устанавливается целевая платформа «PLC150.U-L».

Если выбрать уже установленную целевую платформу ПЛК в списке справа, затем нажать кнопку «Remove», платформа удалится из списка установленных, и после этого для нее будет нельзя создавать проекты, для возобновления такой возможности нужно будет данную платформу повторно устанавливать.

1.2. Настройка связи компьютера с ПЛК, новый проект в «CoDeSys»

ПЛК «ОВЕН» может подключаться к компьютеру тремя способами — с помощью COM-порта (интерфейс RS-232), с помощью интерфейса локальной вычислительной сети

«Ethernet» на основе витой пары, либо с помощью USB-интерфейса (последнее имеют не все модели, поэтому здесь не рассматривается).

Настройка подключения происходит после создания нового проекта в среде «CoDeSys». Проект содержит программы, в последствии выполняемые контроллером и решающие задачи пользователя. Сперва нужно запустить «CoDeSys» (см. рисунок 6). После запуска на экране появится основное окно системы, в котором может открыться приложение с которым работал предыдущий пользователь; в таком случае нужно выбрать пункт «Заккрыть» в подменю «Файл» главного меню. Создать новый проект можно нажав на крайнюю левую кнопку панели инструментов (под главным меню), или выбрав «Файл» — «Новый».

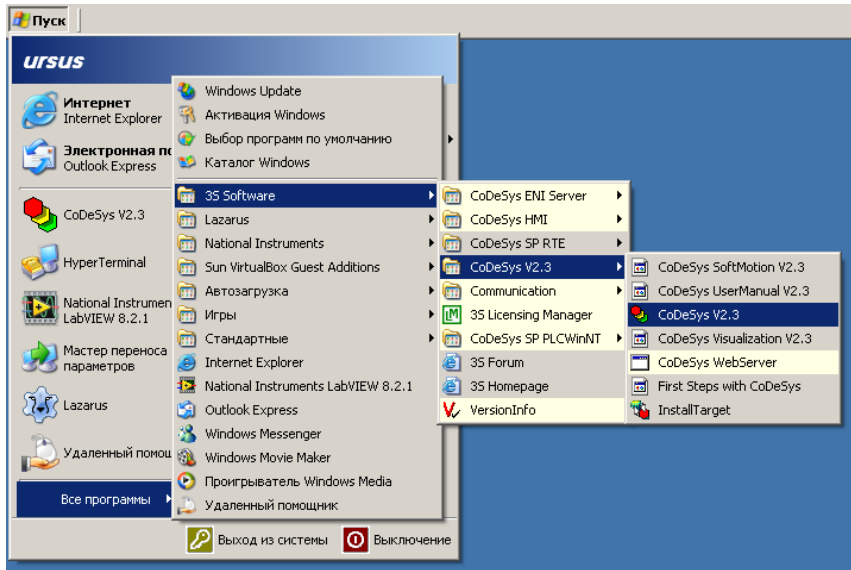


Рисунок 6. Запуск «CoDeSys» из главного меню.

После этого появится окно выбора целевой платформы для создания проекта, где нужно выбрать, для какого вида контроллеров будет создаваться проект. В числе видов контроллеров должен быть и тот, что был (если был) установлен по описанию, данному в п. Вид окна показан на рисунке 7.

После выбора целевой платформы (здесь это

«PLC150.U-L») и подтверждения выбора кнопкой «Ok» в текущем окне на экране появится новое окно, в котором будут содержаться основные параметры и настройки выбранной платформы ПЛК (адреса сегментов памяти, тактовая частота процессора, тип процессора, количества входов и выходов, значения некоторых системных переменных), некоторые из которых пользователь может изменить. В нашем случае менять ничего не требуется, нужно только нажать кнопку «Ok».

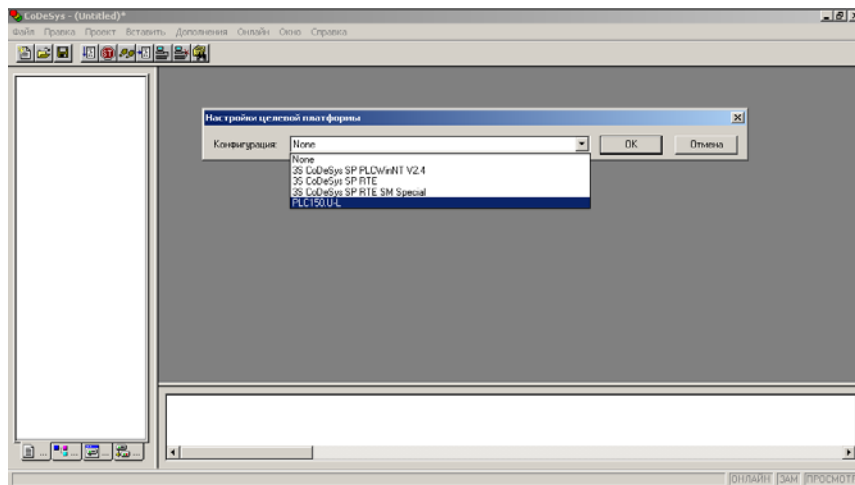


Рисунок 7: Выбор целевой платформы для нового проекта «CoDeSys».

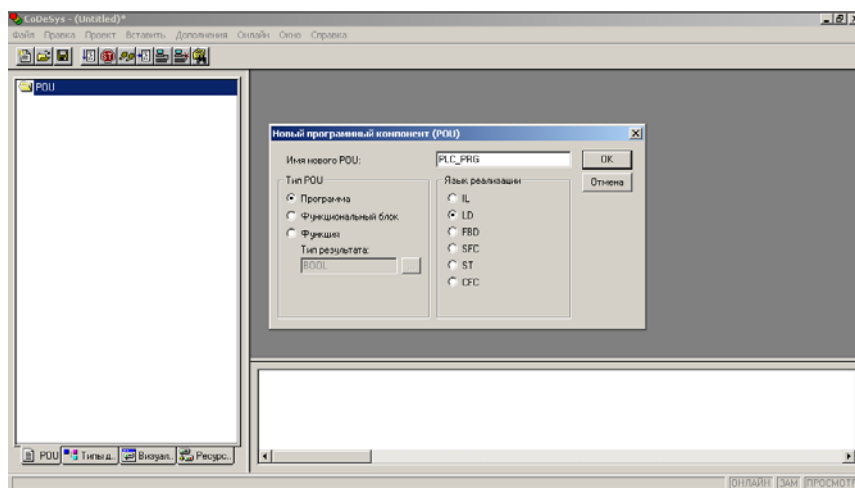


Рисунок 8. Создание основной программы в новом проекте

Далее система предложит создать модуль основной программы (см. рисунок 8) Тип ROU — «Программа», Язык программирования — «LD». В принципе, для начала можно создать любую программную единицу на любом языке.

Теперь можно настраивать связь с контроллером. Выбрать в главном меню «Онлайн» — «Параметры связи...», в результате появится окно (см. рисунок 9).

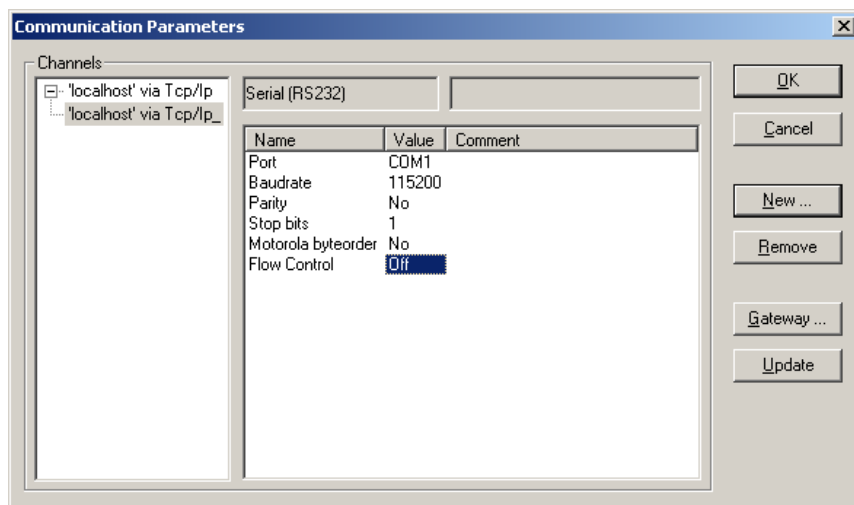


Рисунок 9: Окно настройки связи с ПЛК.

На рисунке 9 показаны уже существующие настройки. Если таковых не имеется, и в иерархическом дереве слева есть только одна строка «'localhost' via Tcp/Ip», то нужно создать новое подключение. Для этого надо нажать кнопку «New...», в правой части окна и в появившемся окне (см. рисунок 10) выбрать вид соединения с контроллером (в нашем случае — «Serial (RS232)»), затем нажать кнопку «OK». Средняя часть окна настроек связи примет вид примерно как на рисунке 9. Список параметров будет тот же, а на рисунке 9 показаны значения параметров, верные для подключения контроллера по умолчанию. Проверить связь с контроллером будет можно после создания первой простой программы. Перед загрузкой программы в контроллер нужно проверить его подключение к компьютеру с помощью кабеля (соединение интерфейса RS-232 на лицевой панели ПЛК с

СОМ-портом ПК).

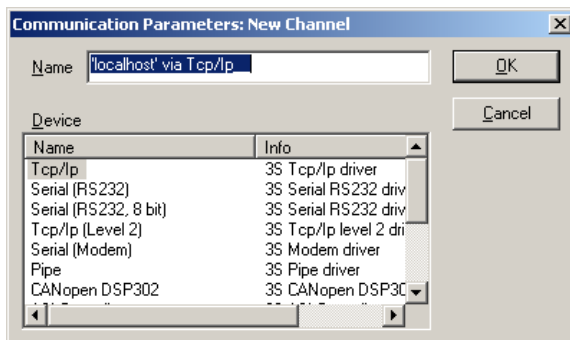


Рисунок 10. Выбор соединения компьютера с ПЛК.

Подробнее о работе интерфейсов связи и их параметрах и характеристиках можно узнать из лекционного курса «Электроавтоматика».

1.3. Первая программа на ПЛК. Таблицы истинности логических операций.

После создания нового проекта и создания в нем новой основной программы, вид основного окна среды программирования будет аналогичен рисунку 11. Создадим первую программу для ПЛК.

Программы для ПЛК создаются на языках программирования, стандартизированных МЭК в документе [IEC61131-3]. Самым простым и старым языком программирования МЭК-стандарта является язык LD — *ladder diagram* — лестничная диаграмма, лестничная логика, в советском стандарте называвшийся языком релейно-контактных схем (РКС). На нем и будем писать первую программу для ПЛК.

Язык LD — графический язык, в котором программа выглядит, как релейная схема в стандарте промышленной автоматизации. Вертикальная линия слева — «провод с высоким потенциалом», вертикальная линия справа — «нулевой провод». Между ними располагаются контактные цепи в виде горизонтальных линий (по общему виду

программы и дано название языка программирования). Слева по линии располагаются «коммутаторы электрического тока» (соответствуют входным переменным логического типа и дискретным входам). Справа — «потребители электрического тока» (соответствуют выходным переменным логического типа и дискретным выходам). Предполагается, что «коммутаторы» можно соединять в любом порядке и последовательности, определяя логику работы цепи, а потребители — только параллельно, так как в реальном подобном устройстве они в любом случае замыкали бы цепь и включались бы одновременно.

Цепь из двух последовательно соединенных коммутаторов соответствует логической операции «И», а цепь из двух параллельно соединенных коммутаторов соответствует логической операции «ИЛИ». Операции «НЕ» соответствует нормально замкнутая кнопка, которая размыкает цепь (фактически, дает логический ноль) при нажатии (подачи логической единицы) и наоборот. Подробные сведения о языке LD можно взять из лекционного курса.

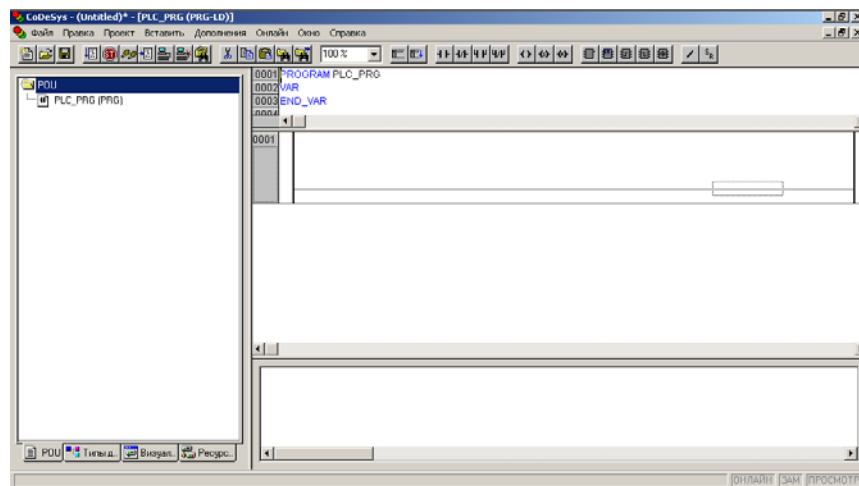


Рисунок 11. Новая программа, рабочее поле для программирования на LD.

Составим программу, с помощью которой можно построить таблицы истинности основных логических операций, которые зададим с помощью лестничной логики.

Во-первых надо знать, как правильно в программе использовать дискретные входы ПЛК. Внизу вертикальной панели в левой части основного окна системы «CoDeSys» есть четыре закладки, каждая из которых соответствует определенной части структуры проекта на «CoDeSys», структура отображается на самой панели. По четвертой (самой правой) закладке отображаются параметры конфигурации проекта, в том числе и конфигурации самого ПЛК (см. рисунок 12). Она и потребуется, нужно выбрать ее, далее выбрать в появившемся иерархическом дереве пункт «Конфигурация ПЛК».

Во-вторых надо поставить задачу, решаемую с помощью ПЛК. Есть три логических операции: «И», «ИЛИ» и «НЕ». Максимальное число операндов у элементарных операций — два; выход у каждой операции один. Таким образом, чтобы задать операнды, надо задействовать два дискретных входа, а чтобы одновременно видеть результаты всех трех операций надо задействовать три дискретных выхода.

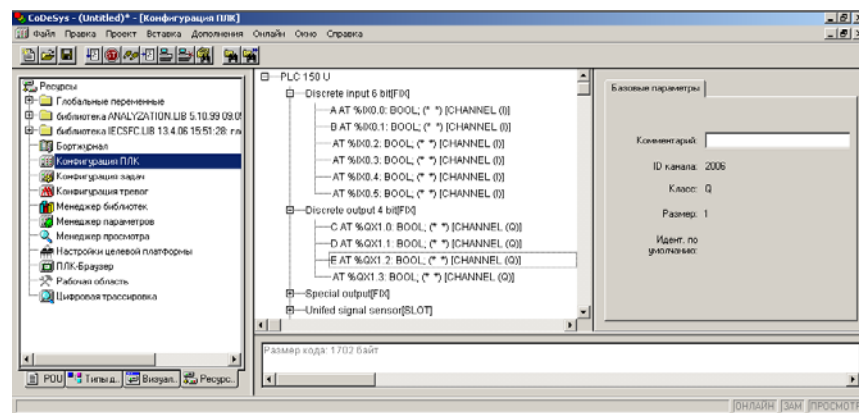


Рисунок 12. Конфигурирование входов и выходов контроллера в системе CoDeSys.

Входы и выходы в МЭК-языках могут обозначаться по именам и по адресам. Пример обозначения входа по адресу: «%IX0.0», пример обозначения выхода: «%QX1.0». Здесь «I» и «Q» обозначают, что это, соответственно, вход и выход, «X» — то, что его емкость составляет один двоичный разряд (один бит), то есть, он может хранить либо нуль, либо единицу

(бывают входы и выходы на 8, 16, или даже 32 бита, которые хранят числа). Первая цифра (слева от точки) — это адрес ячейки (группы бит, байта, двухбайтового слова, структуры, и т. д.), где хранится значение, вторая цифра (справа от точки) — номер значения в ячейке, в данном случае номер бита в байте (нумеруется от 0). По умолчанию дискретные входы (рисунок 15, верхняя часть лицевой панели) 1, 2,.. 6 соответствуют адресам %IX0.0, %IX0.1,.. %IX0.5, а дискретные выходы (рисунок 15, нижняя часть лицевой панели) 1, 2, 3, 4 соответствуют %QX1.0, %QX1.1, %QX1.2, %QX1.3. Эти адреса будем использовать в процессе всей дальнейшей работы.

Адресам можно поставить в соответствие имена. Один из способов это сделать показан на рисунке 12: на конфигурационном экране в дереве имеющихся дискретных входов нужно сделать двойной щелчок «мыши» на слове «АТ», в результате появится поле для ввода имени, и после ввода имени (например «А»), вся конструкция будет выглядеть, как «А АТ %IX0.0...», это будет обозначать, что переменная *A* связана со входом %IX0.0. Таким образом, зададим переменные *A* и *B*, соответствующие входам %IX0.0 и %IX0.1 и переменные *C*, *D* и *E*, соответствующие выходам %QX1.0, %QX1.1 и %QX1.2.

Переходим к написанию непосредственно программы. В переменную *C* будет заноситься результат операции «И», в переменную *D* — результат операции «ИЛИ» (участвуют переменные *A* и *B*), а в переменную *E* — результат операции «НЕ», произведенной над переменной *A*.

Вид программы для решения данной задачи приведен на рисунке 13. Стоит немного рассказать о работе с редактором LD. Чтобы на новой строке поставить первый элемент, необходимо сделать клик «мышью» в правой части строки, при этом на ней появится пунктирный прямоугольник. Далее следует нажать кнопку на панели под главным меню, на которой изображен требуемый элемент, при этом «коммутатор» будет расположен в левой части строки, а «потребитель» — в правой части. Чтобы создать параллельное соединение «коммутаторов» необходимо

выделить один или несколько «коммутаторов» параллельно которым строится ветвление, (последнее делается «мышью» при зажатой клавише «Shift» клавиатуры), после чего нажимается соответствующая кнопка на панели инструментов под главным меню (третья, или четвертая слева в группе кнопок «коммутаторов»). Чтобы добавить еще один «потребитель» необходимо сделать щелчок «мышью» по существующему «потребителю» и нажать соответствующую кнопку на панели инструментов под главным меню.

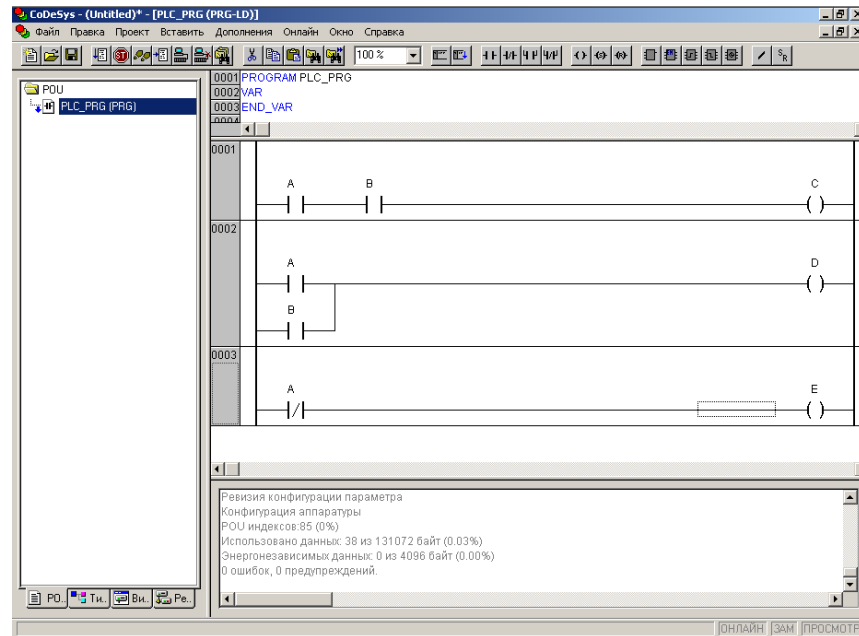


Рисунок 13. Программа для построения таблиц истинности логических элементов в редакторе среды «CoDeSys».

Удалить элемент можно выделив его щелчком левой кнопки «мыши» и нажав клавишу «Delete». Добавлять в программу новые строки можно комбинацией клавишей «Ctrl-T», либо пунктами выпадающего по правой кнопке меню «Цепь (перед)» и «Цепь (после)».

После создания программы, ее необходимо перевести на машинный язык — скомпилировать. Вызов компилятора осуществляется из основного окна «CoDeSys», из главного меню: «Проект» — «Компилировать всё». После выбора этого

пункта в нижней панели окна (см. рисунок 13; панель служебных сообщений, если ее не было, то она появится) через некоторое время появится надпись «0 ошибок, 0 предупреждений». Компиляция успешна, если в ней нет (то есть, ноль) ошибок. Предупреждения — это указания на не ошибочные, но сомнительные, либо бессмысленные места в программе (например, переменная значение знакового типа присваивается переменной беззнакового типа, или переменная объявлена, но нигде не использована), при наличии предупреждений программа, возможно, будет работать некорректно, либо из текста программы можно удалить определенные вещи, совершенно ей не вредя. При наличии ошибок, их необходимо исправлять, выше по тексту на панели служебных сообщений (там же, где указано количество ошибок и предупреждений) находятся сообщения об ошибках, которые указывают на ошибочные строки программы.

После компиляции программу можно загружать в ПЛК и запускать. В случае отсутствия в ПЛК программы, или наличия в нем другой программы, загрузка программы в ПЛК происходит автоматически после выбора пункта «Подключить» подменю «Онлайн» главного меню (см. рисунок 14). Запуск на исполнение и остановка программы в ПЛК осуществляется одним нажатием кнопки «Старт/стоп» на передней панели ПЛК, либо выбором соответствующего пунктов «Старт» и «Стоп» подменю «Онлайн» главного меню.

После загрузки и запуска программы можно проверить ее работоспособность. Как было задумано, тумблерами, подсоединенными к первому и второму дискретным входам ПЛК (сверху, см. рисунок 15) задаются исходные данные для логических операций (в программе обозначены, как A и B). Первый дискретный выход (в программе обозначен, как C) соответствует результату логического умножения A и B , второй дискретный выход (обозначен, как D) соответствует результату логического сложения A и B , а третий дискретный выход соответствует результату инверсии A . Устанавливая тумблеры в разных комбинациях, можно, наблюдая

результаты, составить таблицы истинности для логических операций.

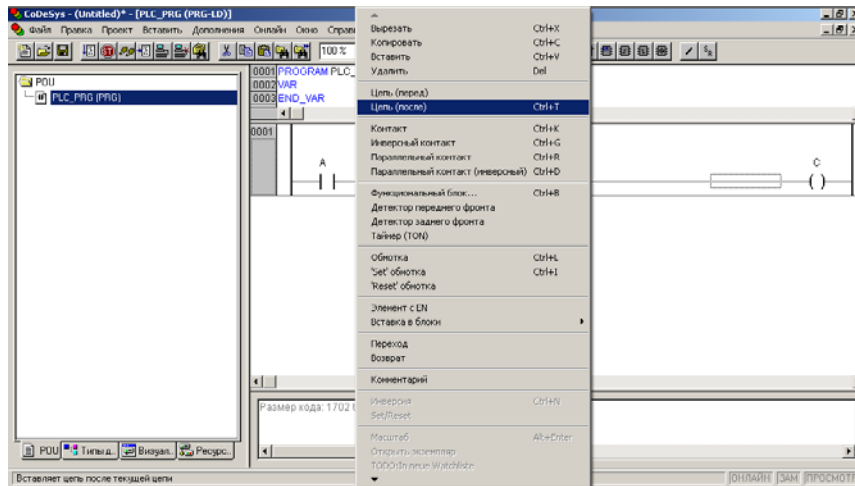


Рисунок 14. Подменю взаимодействия с ПЛК в главном меню.

На рисунке 15 видно, что при первом тумблере, (соответствует переменной A) задающем логический нуль, и втором тумблере (соответствует переменной B), задающем логическую единицу, результат логического умножения (переменная C — первый дискретный выход) будет нулевым, результат логического сложения (переменная D — второй дискретный выход) — будет равен единице, также единице будет равна переменная E в которую программно заносится инвертированная переменная A .

С помощью запрограммированного таким образом ПЛК можно построить таблицы истинности основных логических операций.

Итак, в процессе постановки данной задачи и реализации ее решения был получен опыт работы с программируемым логическим контроллером и усвоены и описаны основные приемы работы со средой разработчика и основные принципы программирования на языке Ladder Diagram стандарта МЭК.

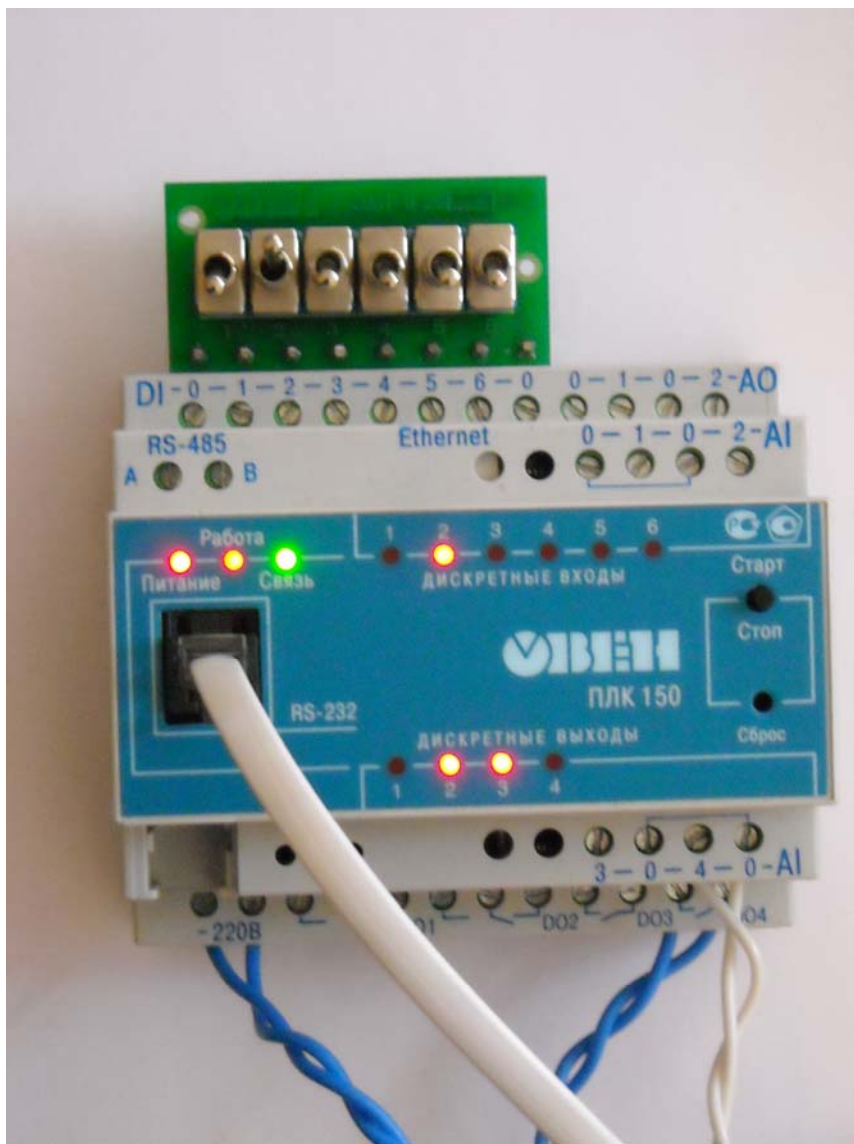


Рисунок 15. Внешний вид лицевой панели контроллера с запущенной программой

2. Программирование на языке LD. Таймеры, счетчики и детекторы фронтов.

2.1. Общие сведения о программе и программировании для ПЛК.

Цель работы: освоить основные принципы программирования на языке LD, в том числе, использование типовых функциональных блоков: таймеров, счетчиков, детекторов фронтов.

В предыдущей лабораторной работе была составлена простая программа на простом в освоении языке программирования стандарта МЭК — Ladder Diagram, при этом рассматривались основные приемы работы со средой без подробностей о том, каким образом программа выполняется, как устроены внутри более сложные проекты, и какие еще дополнительные средства существуют для повышения удобства разработки приложений.

Известно, что микропроцессор в работе выполняет заданную последовательность машинных команд; последовательность обычно имеет начало и конец и кодирует в себе выполнение процессором определенного действия.

В случае, когда ПЛК управляет устройством, требуется, чтобы ПЛК работал до тех пор, пока устройство не будет остановлено по команде оператора, то есть фактически требуется чтобы программа без команды останова, поданной извне, могла выполняться бесконечно долго.

Из курса информатики известно, что основные алгоритмические структуры бывают трех видов: последовательные (линейные) условия (ветвления) и циклы. Последовательная структура представляет собой простую цепочку команд, которые выполняются процессором подряд от начала и до конца, при ветвлении одни команды выполняются при истинности определенного утверждения, другие — при его ложности, а когда выполняется цикл — определенная последовательность команд постоянно повторяется, пока не будет истинным (или ложным) определенное утверждение. Таким образом, можно сделать

вывод, что при определенных условиях циклы могут выполняться бесконечно, и можно предположить, что в основе программ для ПЛК должна лежать циклическая алгоритмическая структура.

Так оно и есть: цикл выполнения программы в ПЛК реализован непосредственно в его встроенном программном обеспечении, и пользователю не приходится заботиться о непрерывности выполнения программы. Проект для ПЛК описывает внутреннее алгоритмическое содержимое этого внешнего цикла, а с помощью определенных настроек проекта можно задать особенности циклического выполнения программы: например, выполняется ли один оборот цикла за строго определенное время, или следующий цикла начинается непосредственно сразу после окончания текущего.

Особенности организации циклического исполнения программы на ПЛК можно узнать из лекционного курса.

Данные (числа, символы, строки, логические значения, и т. д.) при программировании на языках стандарта МЭК хранятся в переменных, как и при программировании на многих других языках. Для использования переменные объявляются в программе (в верхней части экрана, между словами VAR и END_VAR). Возможные типы переменных и правила задания значений приведены в лекционном материале.

В системе CoDeSys имеется два вида переменных: адресные, или безадресные. Адресные переменные связаны со строго определенной ячейкой в памяти ПЛК, положение которой задается в виде специального обозначения. При объявлении безадресных переменных адрес не указывается, и им отводится место в первой подходящей по размеру свободной области так называемой динамической памяти. Адресные переменные предназначены для того, чтобы связывать их со входами, выходами, статическими ячейками памяти, которые могут быть использованы для передачи данных по протоколам связи (см. лекции), а безадресные — для внутреннего использования. Адресные переменные более доступны для внешних устройств, в то время как с безадресными переменными проще работать программисту.

Далее для ввода и вывода будут использоваться адресные, а для обработки данных — безадресные переменные. Безадресные переменные наравне со всеми остальными доступны в визуализациях (см. стр. 45).

Адресные переменные можно объявить двумя способами, один из которых продемонстрирован в предыдущей лабораторной работе (см. стр. 17). Второй способ заключается в том, что переменную, связанную с выходом можно объявить вместе со всеми остальными (внутренними) переменными в верхней части экрана программы (рисунок 19, переменные *OUTER_SENSOR*, *INNER_SENSOR*, *RESET_BUTTON* и *LIGHT*). В данной работе (и во многих последующих) будет избран именно второй способ объявления переменных, так как при его использовании все задействованные имена переменных видны одновременно, это удобно при программировании.

Проект в системе «CoDeSys» состоит из отдельных программ (основной из которых является программа PLC_PRG), функций и функциональных блоков. Подробнее это будет рассматриваться на стр. 38, в одной из следующих лабораторных работ. Здесь будет нужна только та информация, что функциональный блок имеет свой набор переменных, который существует вместе с переменными использующей его программной единицы, например, если использовать функциональный блок в основной программе, то значения всех входов и выходов функционального блока не будут теряться, пока выполняется программа. У каждого функционального блока свой набор переменных, и поэтому функциональный блок сам перед использованием объявляется как переменная. Как это происходит будет видно далее, в процессе выполнения работы.

Существует набор типовых функциональных блоков, реализующий функции, часто используемые в программировании ПЛК. Это счетчики, таймеры и детекторы фронтов. В случае ПЛК «ОВЕН» они находятся в стандартной библиотеке подпрограмм, входящей в комплект CoDeSys, который можно загрузить с официального сайта компании «ОВЕН». К этим блокам относятся *таймеры* (*TON*,

TOF), счетчики(*CTU, CTD, CTUD*), детекторы фронтов (*R_TRIG, F_TRIG*). Далее будут рассмотрены задачи на использование этих функциональных блоков. Информация о работе данных блоков присутствует в файле помощи программного пакета «CoDeSys», а также они рассматриваются в лекционном материале.

2.2. Задача 1. Демонстрация работы реверсивного счетчика и детекторов фронтов.

Создать на языке LD программу, которая увеличивает на 1 значение целой переменной при наличии положительного фронта на дискретном входе *%IX0.0* и уменьшает на 1 значение этой переменной при наличии отрицательного фронта на входе *%IX0.1*.

Общий вид программы на языке LD представлен на рисунке 16. Для регистрации фронтов использованы детекторы фронтов *R_TRIG* и *F_TRIG*, для работы с целой переменной используется реверсивный счетчик *CTUD*.

На вход *CLK* детектора фронтов подается дискретный сигнал: информация с дискретного входа, значение логической переменной, или логического выражения. Выход *Q* детектора фронта устанавливается в единицу в том случае, если входное значение блока изменилось по сравнению со значением в предыдущем цикле, единичное значение сохраняется в течение одного цикла. *R_TRIG* выдает единицу, когда ноль на входе сменяется единицей, *F_TRIG* выдает единицу, когда единица на входе сменяется нулем.

Переменные *A* и *B* связаны с дискретными входами точно так же, как в предыдущей задаче. С первого дискретного входа значение сигнала подается на вход блока *R_TRIG*, объявленного как переменная *RTI*, со второго — на *F_TRIG*, объявленный, как переменная *FTI*. Выход *FTI* связан с переменной *F*, которая далее подана на вход *CD* (уменьшение на единицу) счетчика. Выход *RTI* подан напрямую на вход *CU* (увеличение на единицу) счетчика. Переменная *X*, объявленная, как целое число, связана со

счетным выходом счетчика *CV*. Выходы сброса счетчика на ноль (*RESET*) и загрузки в него начального значения (*LOAD*) в данном примере не используются и на них подается логический ноль — логическая константа «ложь» - *FALSE*.

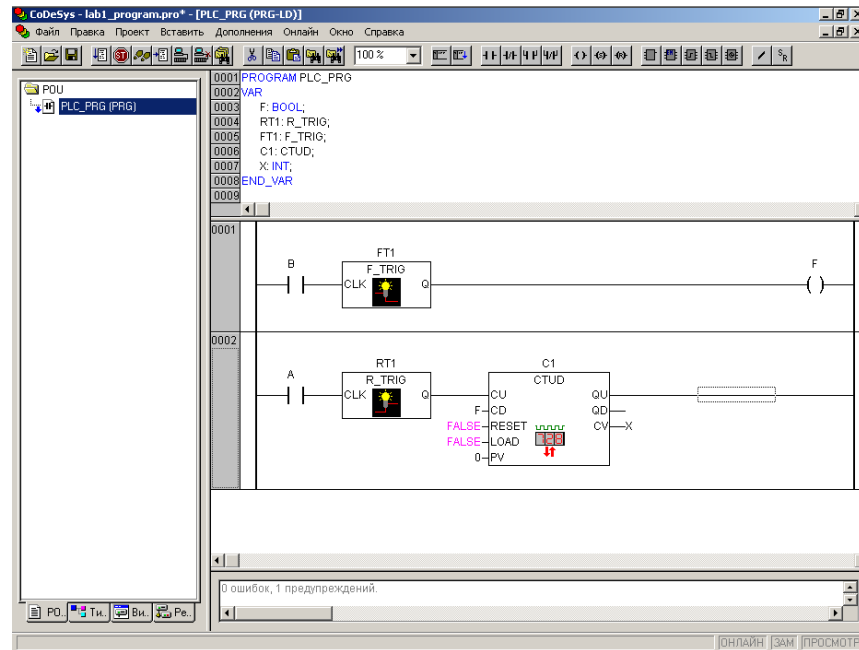


Рисунок 16. Программа демонстрация работы детекторов фронтов и реверсивного счетчика.

Поскольку счетчик *CTUD* используется не полностью, при компиляции данный пример сгенерирует одно предупреждение, но, несмотря на это, пример работает нормально. Тестирование примера просто: если первый слева тумблер переводится из нижнего положения в верхнее, переменная *X* увеличивается на единицу, если второй слева тумблер переводится из верхнего положения в нижнее, переменная *X* уменьшается на единицу. Следует обратить внимание, что детекторы фронтов, счетчики и таймеры не являются базовыми (то есть, непредставимыми простыми операторами) программными единицами. Все функциональные блоки можно реализовать программно с помощью базовых операций. В доказательство этого составим программу для того же самого примера без применения

счетчиков и детекторов фронтов. Программа показана на рисунке 17.

Стоит сказать об арифметических операциях. Они реализованы в виде стандартных функций ADD (сложение) и SUB (вычитание). При значении логической единицы на входе EN блок работает, не работает при логическом нуле на входе EN. Во встроенной справке приведен перечень всех операций, осуществляемых с помощью стандартных функций со входом EN.

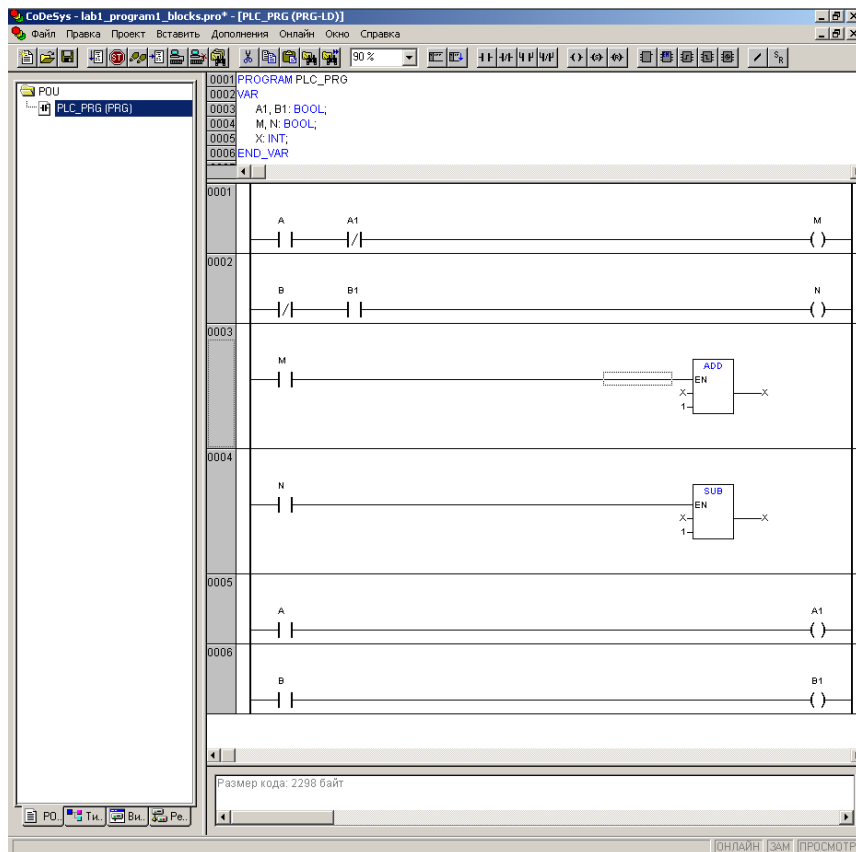


Рисунок 17. Программа демонстрация работы детекторов фронтов и переменных машины

Детектор фронта в этом примере реализован следующим образом: объявлены две дополнительные переменные, по одной на каждый детектируемый сигнал, в

самом конце программы, после использования текущих значений сигналов, они сохраняются в объявленные переменные, и значения переменных используются в следующем обороте цикла программы как значения, сохраненные в прошлом шаге, и так происходит каждый цикл. Первые две строки программы представляют собой именно детектирование сигнала, единовременную проверку его значения в прошлом и настоящем шагах.

2.3. Задача 2. Управление освещением в комнате.

Условие: есть комната, в двери стоят два датчика регистрации пересечения линии: снаружи и внутри комнаты, они подсоединены к ПЛК. Также к ПЛК подсоединен выключатель комнатного освещения, есть возможность использовать еще одну кнопку. Требуется составить программу, которая управляет автоматическим включением и выключением света в комнате.

Программа, являющаяся решением задачи показана на рисунке 18.

Если человек входит в комнату, то он пересекает сначала наружный датчик, потом внутренний, и в момент пересечения внутреннего датчика внешний датчик уже регистрирует присутствие человека в дверях. Процесс выхода из комнаты относительно датчиков происходит так же, только датчики следует поменять местами. Таким образом, по переднему фронту одного датчика в сочетании с уже сработавшим другим получим короткий импульс, обозначающий вход, или выход одного человека. Далее требуется реализовать счет людей, это можно сделать с помощью реверсивного счетчика. Также, если значение счетчика больше, или равно 1, следует включить свет, если нет, то выключить. Предположим, что возможна ситуация, когда человек, находясь в комнате, может выйти из нее не через дверь, а, например, выпрыгнуть в окно. Тогда свет останется гореть и в том случае, если все вышли. Для этого стоит использовать кнопку принудительного гашения света, которую следует расположить с наружной

стороны двери, кнопку соединить со сбросом счетчика.

Приведем назначение переменных. *OUTER_SENSOR* и *INNER_SENSOR* — переменные, связанные с внутренним и наружным датчиком пересечения линии. Устанавливаются, если линия пересечена посторонним объектом и сбрасываются, если пересечения нет. *RESET_BUTTON* кнопка гашения света. *QUIT* и *ENTER* — внутренние переменные, устанавливающиеся в единицу в моменты соответственно выхода из комнаты и входа в нее. *LIGHT* — переменная, связанная с реле включения света, *CTR* — переменная счетчика вошедших в комнату.

На рисунке 19 изображено решение той же задачи, но без применения типовых функциональных блоков.

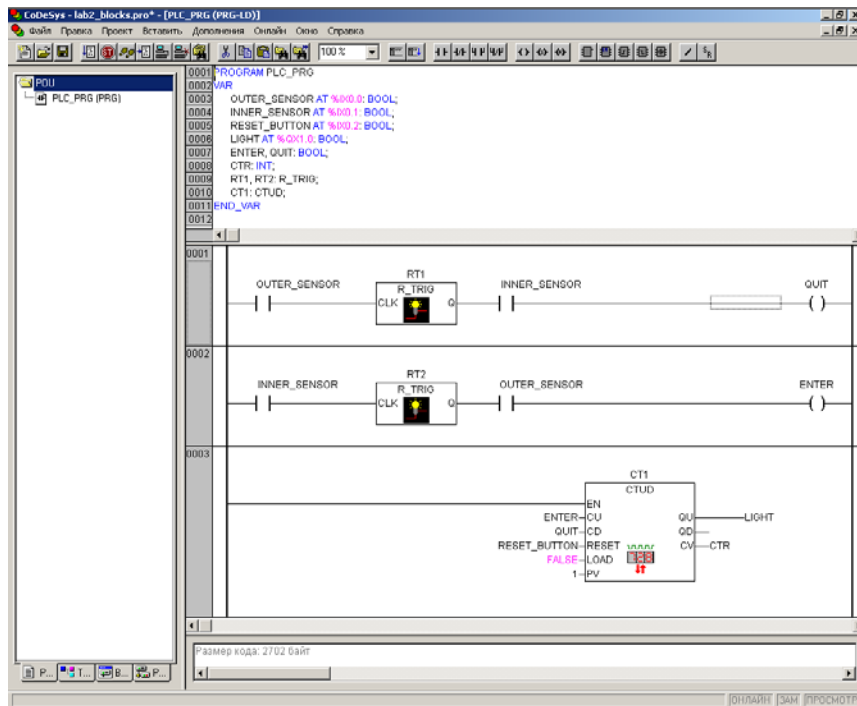


Рисунок 18. Решение задачи об автоматическом включении света, сделанное с помощью типовых функциональных блоков.

Переменные *ISI* и *OSI* хранят значения переменных *INNER_SENSOR* и *OUTER_SENSOR* за предыдущий цикл. Функция *GT* — сравнение двух чисел на входе, и, если

«верхнее» больше, чем «нижнее», функция возвращает логическую единицу. MOVE — пересылка значения. Слева указывается его источник, а справа — приемник, значение может быть любого типа, переменные источника и приемника должны быть одного и того же, или совместимых типов. Подробнее про совместимость типов можно узнать из лекционного материала.

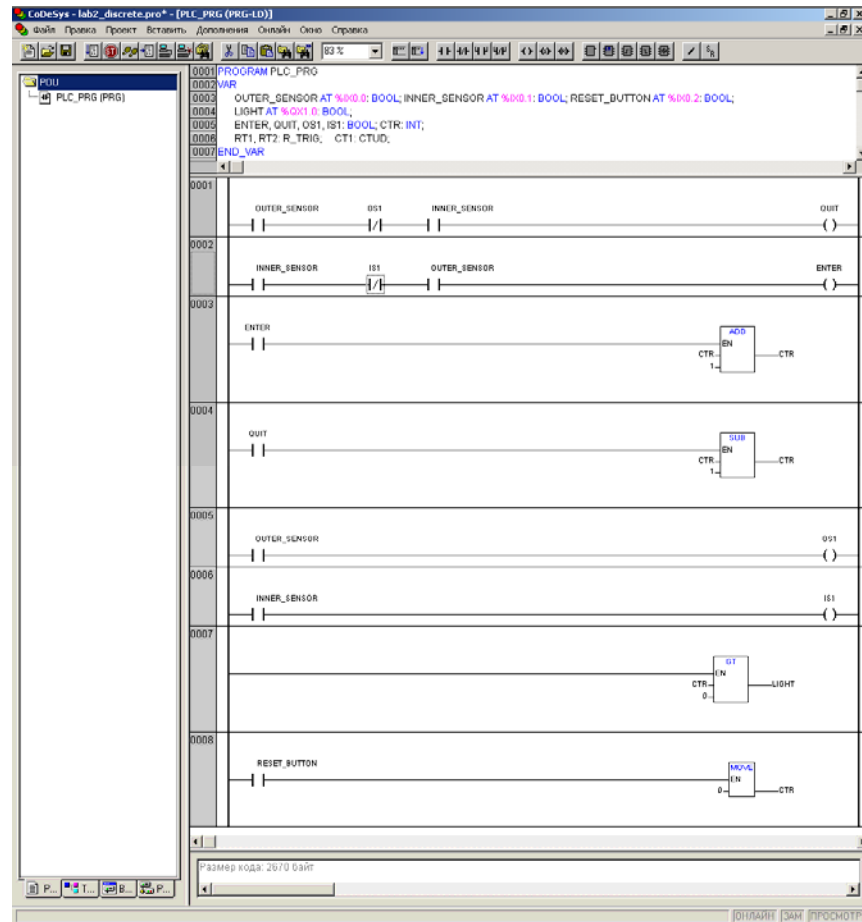


Рисунок 19. Решение задачи об автоматическом включении света, сделанное без типовых функциональных блоков.

2.4. Задача 3. Программный генератор периодических импульсов.

Создать программный генератор прямоугольных импульсов с постоянной скважностью и задаваемым периодом, использовать таймеры.

Такая программа генератора будет использована и в других лабораторных работах. Работа таймера описана во встроенной справке CoDeSys.

Внешний вид программы показан на рисунке 20. Принцип функционирования генератора приведен ниже.

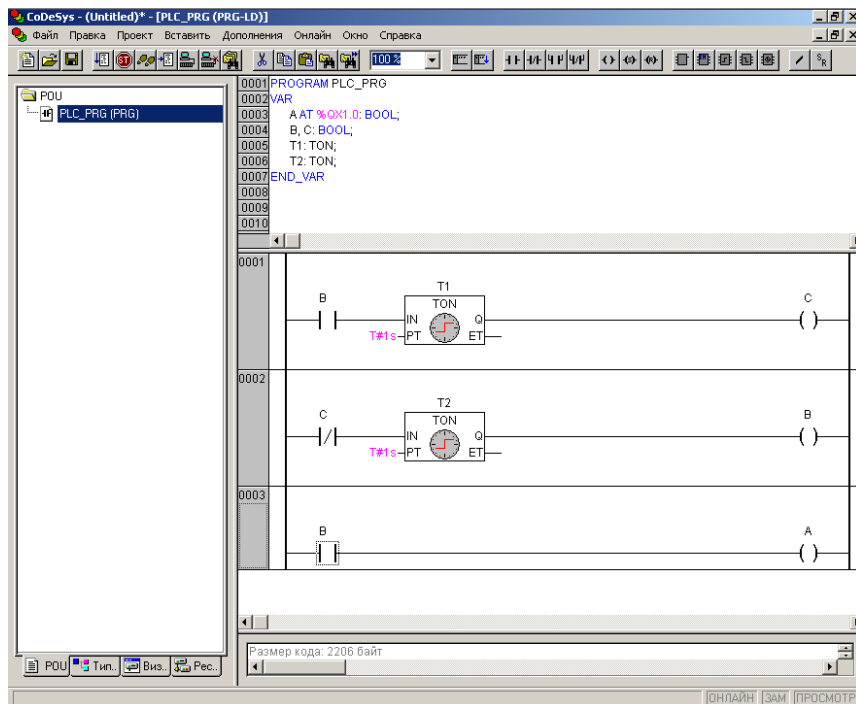


Рисунок 20. Программа генератора прямоугольных импульсов на языке LD.

Допустим, непосредственно после первого запуска программы, переменные B и C равны логическому нулю, в результате таймер $T1$ не запускается, а таймер $T2$ — запускается, так как на его входе находится инвертированная переменная C , которой таймер по истечении времени передал

значение логической единицы. После заданного времени (на таймере T2 в данном примере время равно 1 сек.) устанавливается в единицу переменная *B*, стоящая на выходе таймера T2, при этом запускается таймер T1. Таймер T1 через заданное время (тоже 1 сек.) выдаст на выходе логическую единицу, которая будет присвоена переменной *C*. При этом таймер T2 перестает держать на выходе логическую единицу, и ноль сразу же записывается в переменную *B*, при этом выключается и первый таймер, так как с его входа пропадает логическая единица, в виде переменной *B*. После выключения первого таймера переменная *C* сбрасывается и снова запускается таймер T2, на входе которого появляется логическая единица в виде инвертированной переменной *C*. Цикл повторяется.

Третья строка программы предназначена только для того, чтобы выводить значение переменной на дискретный выход.

После запуска программы можно увидеть, когда контроллер подключен к компьютеру, что переменная *C* не меняет своих значений, а переменная *B* – периодически устанавливается и сбрасывается, так как, установка и сброс переменной *C* происходит с интервалом времени в один цикл, а установка и сброс переменной *B* – с заданными промежутками времени.

3. Основные возможности языков ST, CFC и FBD.

3.1. Особенности построения программ на языках ST, CFC и FBD.

Цель работы: освоить основные приемы программирования на основных МЭК-языках.

Любая система промышленного программирования построена так, что любую программную единицу, созданную для ПЛК с ее помощью можно записать на любом языке стандарта МЭК, и разница состоит лишь в удобстве ее прочтения.

Каждый язык программирования стандарта МЭК подходит для определенного круга задач. Язык LD, знакомство с которым имело место на предыдущих лабораторных работах, являясь наиболее простым для понимания, в то же время применим для узкого круга задач машинной логики и простого дискретного управления, в то время, как организовать вычисления с его помощью не представляется особенно удобным.

Язык ST похож на многие популярные языки программирования высокого уровня, такие, как BASIC, или Pascal. В языке имеются все стандартные структуры, необходимые программисту: условия и циклы. На нем удобно организовывать вычисления, обработку строк, но сложные логические конструкции на нем выглядят гораздо более громоздко и непонятно, чем на LD. Также, если нужно описать на языке ST некоторую структуру, хорошо представимую в виде схемы, состоящей из блоков, соединенных линиями, то легкость прочтения и наглядность такого описания также будет невысока.

Язык CFC — полностью графический, в нем очень удобно оперировать с функциональными блоками. Приложения, которые представимы в виде схем и состоят из функциональных блоков, выглядят в такой системе наглядно и интуитивно понятны, но также, сложные вычисления с

большим количеством элементарных действий приводят к быстрому разрастанию «собираемой схемы», уменьшению удобства ее прочтения и уменьшению эффективности данного применения языка. Несмотря на то, что язык не является стандартным языком системы МЭК, при решении задач, для которых характерны «схемотехнические» представления данных он применяется довольно часто. Еще две особенности его реализации в системе CoDeSys – автоматическая трассировка связей между блоками схемы и возможность обработки обратных связей внутри схемы (с задержкой на цикл). Также на языке CFC сложно управлять порядком выполнения функций и функциональных блоков, так как в случае устройства, собранного по схеме и непрерывно работающего, очередность работы каждого блока отсутствует, как понятие: управлять алгоритмической последовательностью выполнения блоков можно посредством связей (связанные блоки исполняются от входа к выходу всей цепи), разновидностью такого управления является условное управление выполнением функциональных блоков с помощью сигналов *EN* и *ENO*, которые можно добавить к любому блоку с помощью кнопки на панели инструментов под главным меню, об этом подробнее в следующих работах, хотя это уже применялось в языке LD (см. рисунок 19).

Язык FBD является промежуточным между LD и CFC. Схема разбивается на смысловые фрагменты, каждый из которых помещается в строку, выглядящую наподобие строки языка LD, при этом взаимное расположение блоков не так свободно, как в CFC. В отличие от языка CFC, данный язык принят как стандарт. Его можно рассматривать как попытку объединить определенность очередности выполнения операторов, присущую языку LD с наглядностью CFC. Язык мало применим, применяется в отсутствие языка, подобного CFC на других системах. Язык FBD более популярен среди ПЛК Siemens, в частности Siemens S7-200 в среде Step 7 MicroWin.

3.2. Решение на языках ST, CFC и FBD задачи об управлении включением света.

Решение одной из задач предыдущей лабораторной работы на языке CFC представлено на рисунке 21.

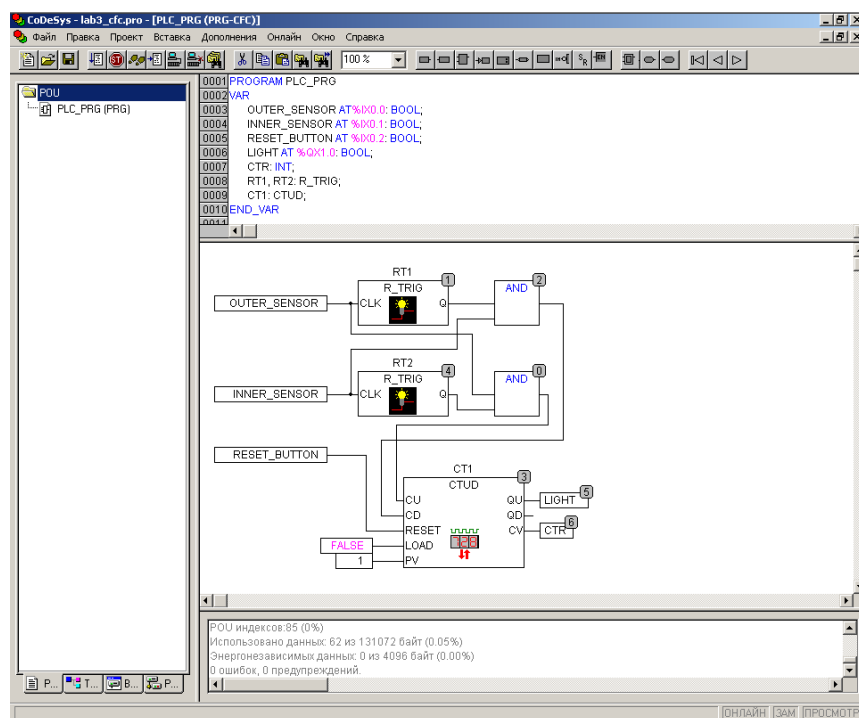


Рисунок 21. Решение задачи об автоматическом включении света на языке CFC.

Видно, что программа занимает мало места, все представления компактны и понятны. Видны детекторы фронтов RT1 и RT2, логические элементы и блок двунаправленного счетчика, к которому присоединены управляющие сигналы. В верхней части экрана объявлены все переменные, для их объявления используются конструкции, использующие синтаксис языка ST.

Программа на языке FBD для решения той же самой задачи представлена на рисунке 22. При той же компактности представления информации видно, в какой последовательности исполняется программа. Но наглядность

структуры уменьшена, и требуется дополнительное время, чтобы понять, каким образом к счетчику CT1 подключены выходы логических элементов, в то время, как в программе на CFC в принципе отсутствует нужда в переменных *QUIT* и *ENTER*.

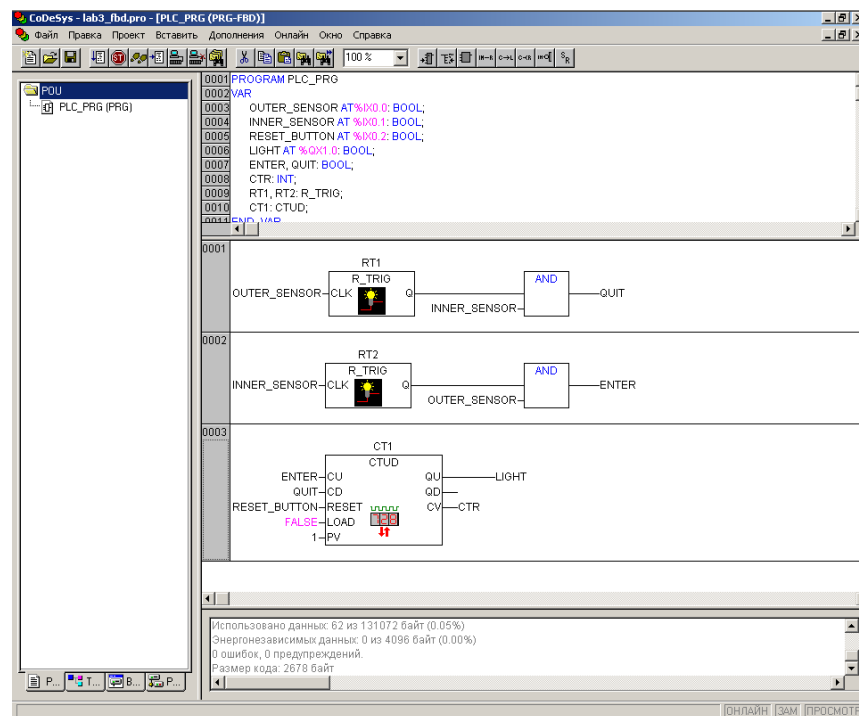
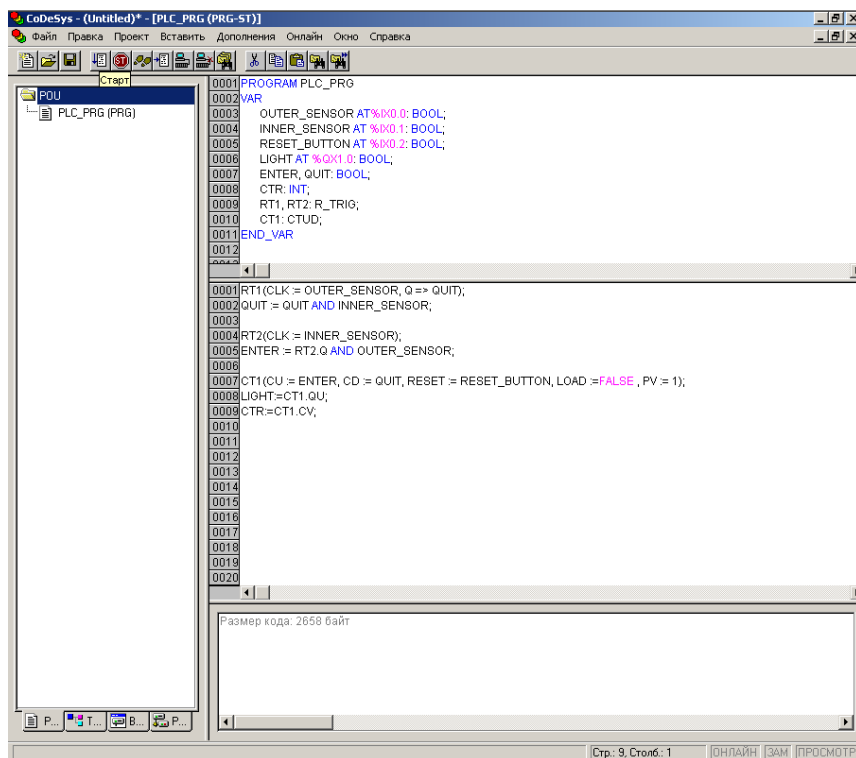


Рисунок 22. Решение задачи об автоматическом включении света на языке CFC

Если скомпилировать и запустить две этих программы, результат их работы будет абсолютно одинаков.

Программа на ST представлена на рисунке 23. На самом деле, данная программа не является хорошим примером демонстрации возможностей языка ST в плане организации программы, но на этом примере показано, что, с одной стороны, наглядность задач, описываемых схемами текстовым языком, описывающим последовательности действий, невысока, а, с другой стороны, возможность вызова функциональных блоков в ST имеется и может быть записана достаточно аккуратно, поэтому избегать функциональных

блоков при программировании на ST не стоит. Это не последний случай использования функциональных блоков с языком ST.



```
0001 PROGRAM PLC_PRG
0002 VAR
0003   OUTER_SENSOR AT %IX0.0: BOOL;
0004   INNER_SENSOR AT %IX0.1: BOOL;
0005   RESET_BUTTON AT %IX0.2: BOOL;
0006   LIGHT AT %QX1.0: BOOL;
0007   ENTER, QUIT: BOOL;
0008   CTR: INT;
0009   RT1, RT2: R_TRIG;
0010   CT1: CTUD;
0011 END_VAR
0012
0001 RT1(CLK = OUTER_SENSOR, Q => QUIT);
0002 QUIT := QUIT AND INNER_SENSOR;
0003
0004 RT2(CLK = INNER_SENSOR);
0005 ENTER = RT2.Q AND OUTER_SENSOR;
0006
0007 CT1(CU := ENTER, CD = QUIT, RESET = RESET_BUTTON, LOAD = FALSE, PV = 1);
0008 LIGHT := CT1.Q;
0009 CTR := CT1.CV;
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
```

Размер кода: 2658 байт

Стр.: 9, Столб.: 1 | ОНЛАЙН | ЗАМ | ПРОСМОТР

Рисунок 23. Решение на языке ST задачи об автоматическом управлении включением света.

Отладочные визуальные возможности графических языков наглядны, и их легко использовать в процессе исполнения программы на контроллере, подключенном к компьютеру с запущенной средой разработки «CoDeSys». Текстовые языки более неудобны в плане отображения значений переменных в реальном времени, но в среде «CoDeSys» такая возможность есть, и ее реализация показана на рисунке 24. Поле с текстом программы разделяется вертикально на две части, при этом в левой части отображается текст, а в правой — для каждой строки текста переменные, встречающиеся в тексте и их значения. Состояние логического нуля обозначается белыми буквами на

черном фоне, а логической единицы — белыми буквами на синем.

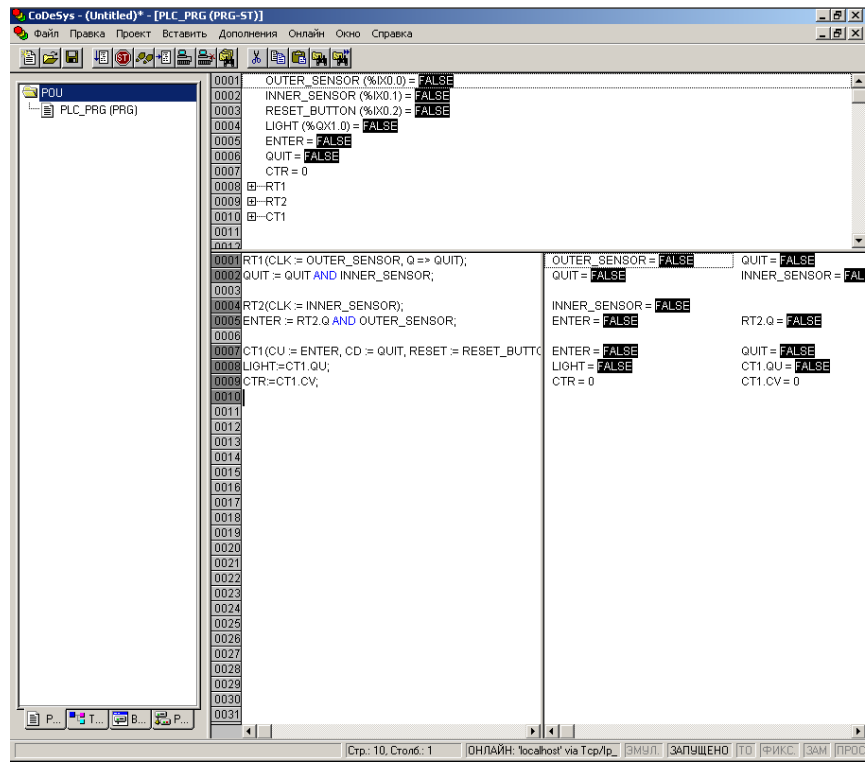


Рисунок 24. Экран с программой на языке ST при подключенном к системе контроллере и запущенной программе.

4. Программные единицы: функции, программы и функциональные блоки, создание структуры приложения.

4.1. Структура приложений в среде «CoDeSys».

Цель работы: научиться работать со сложными проектами и использовать ROU для их структурирования.

Чем больше объем текста программы, тем труднее понять, как он работает. Потребность в структурировании появилась сразу же, как только стали появляться объемные программные проекты. Так появились языки программирования с возможностью структурирования программы. В языках стандарта МЭК такие возможности также есть. Программный проект состоит из отдельных программных единиц, которые называются ROU — program organisation unit — блок для организации программы.

ROU бывают трех типов — функции, функциональные блоки и программы.

Функции возвращают одно значение, в качестве входных параметров могут иметь несколько значений, и те переменные, которые используются внутри функции, сохраняют свои значения только в процессе выполнения функции, иными словами, *функция живет только пока выполняется*.

Функциональные блоки фактически являются объединением программного кода и группы входных и выходных переменных. Переменные функционального блока не теряют свои значения всё время исполнения программы. У функционального блока может быть несколько входных и несколько выходных параметров. Он объявляется как переменная, при этом название ROU фактически является «типом» функционального блока, а имена переменных, объявленных с этим типом являются именами экземпляров данного типа функционального блока; у каждого экземпляра свой набор значений переменных, к ним можно обращаться

из программы, обозначая сначала имя блока, а далее, через точку, имя внутренней переменной блока. Функциональные блоки были уже использованы ранее, например счетчик СТ1 типа СТUD (см. рисунок 22). Также, обращение к переменным функционального блока можно увидеть в тексте программы на рисунке 23.

Программы редко используются во множественном числе. Когда создается проект, создается одна программа, которая исполняется по умолчанию, она называется PLC_PRG, это имело место еще в первой лабораторной работе. Тем не менее, когда контроллер одновременно решает несколько различных по смыслу задач, имеет смысл создать несколько программ, которые будут вызываться из PLC_PRG, при этом основная программа может не нести никаких функций, кроме вызова других программ, а между вызываемыми программами разделить решаемые контроллером задачи.

У программы нет входов и выходов, все переменные программы — ее внутренние. Для обмена данными между программами в проекте системы CoDeSys можно создавать глобальные переменные.

4.2. Пример проектирования структуры приложения и его реализации.

Чтобы понять, как правильно поделить задачу на функции, функциональные блоки и программы, следует описать предполагаемое ее решение сначала с максимальным обобщением, а затем всё более и более детально. Проанализируем решение задачи.

Задача. С четырех конвейеров с не на много отличающейся скоростью сходят подшипники. Различие в скорости обусловлено особенностями конвейеров. Подшипники одинакового качества, с одинаковыми характеристиками. Задача — подсчитать их все. Концевой выключатель для фиксации прохождения подшипника через определенное сечение конвейера имеется на каждом конвейере. Также подшипники на конвейере могут

выбраковываться. Это происходит до места их подсчета, при этом подшипник снимается с конвейера, его место ничем не занимает. Требуется написать программу, моделирующую конвейер с выбраковкой и осуществляющую подсчет подшипников.

Анализируем структуру решения. Что должно быть? Четыре конвейера. Счетчик, суммирующий все четыре количества. Нужно ли считать количества по каждому конвейеру? Чтобы ответить на последний вопрос, ответим на другой вопрос: «Как считаются подшипники на конвейере?» Установлен датчик, который срабатывает при прохождении подшипника через некоторое сечение, например, пара фотодатчик-источник света. Датчик дает дискретный импульс. Если логически сложить все четыре сигнала, то может получиться, что два импульса, поступившие в один момент времени, совпадут, и фактически получится один импульс, который приведет к тому, что несколько подшипников посчитаются как один. Таким образом, нужно накопительно считать подшипники на каждом конвейере, затем складывать все результаты, получая при этом действительную сумму накопившихся подшипников на момент её (суммы) подсчета. Многократный подсчет суммы позволит судить о том, как количество подшипников увеличивается. Более того, можно будет узнать долю каждой конвейерной линии в производстве подшипников.

Итак, перечисляем блоки точнее:

- четыре конвейера с выбраковкой, отличающиеся только скоростью;
- четыре счетчика, считающих подшипники;
- итоговый сумматор, который в каждый момент показывает сумму всех четырех счетчиков.

Как реализовать счетчики и считать сумму, рассматривалось в предыдущих лабораторных работах. Следует уточнить, что же представляет собой модель конвейера. Движущиеся на конвейере подшипники, попадающие в зону движения датчика, видны контроллеру, как периодические дискретные импульсы, таким образом, сигнал с конвейера можно моделировать с помощью

генератора прямоугольных импульсов, рассмотренного в предыдущей лабораторной работе. Отбраковку можно моделировать вручную, с помощью внешнего дискретного сигнала (тумблера на дискретном входе), который, устанавливаясь в единицу, будет обозначать отсутствие подшипника на месте, при этом импульс с конвейера на счетчик не будет поступать. Это можно реализовать с помощью логической операции. Функциональный блок модели конвейера удобнее сделать, используя язык LD, программа представлена на рисунке 25. Входной параметр конвейера — постоянная времени генератора импульсов; от скорости работы линии зависит частота следования импульсов

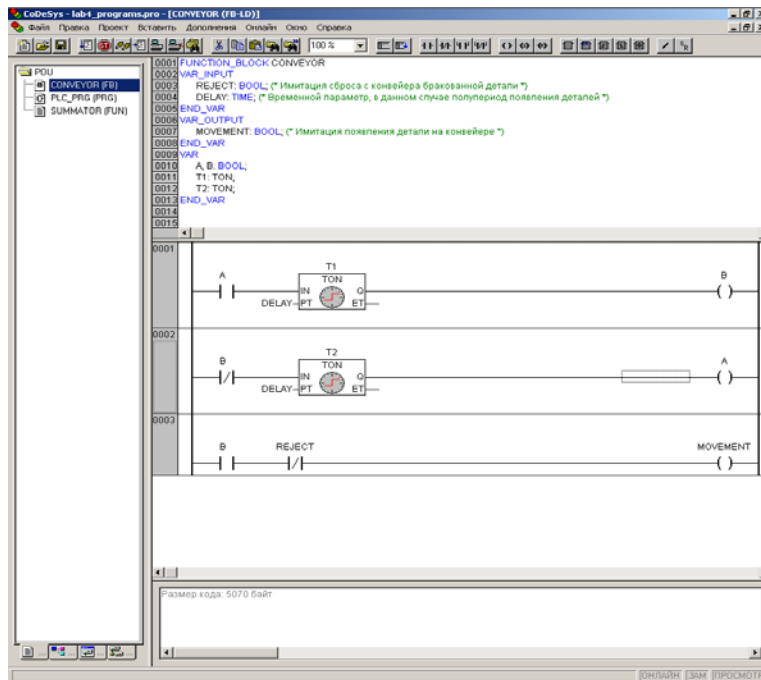


Рисунок 25. Функциональный блок модели конвейера для решения задачи. Реализован на языке LD.

То действие, которое делает сумматор — постоянный многократный подсчет мгновенного значения суммы произведенных подшипников, в «CoDeSys» реализуется в виде функции, и здесь функциональный блок не нужен. Дело в том, что функциональные блоки нужны там, где результат

зависит не только от текущих значений параметров, но и от предыдущих, так как значения параметров функциональных блоков сохраняются в любой момент работы программы. А в данной задаче сумматор только складывает уже подсчитанные значения сумм деталей по четырем конвейерам. Реализация сумматора изображена на рисунке 26. Так как это арифметические операции, их проще написать на языке ST.

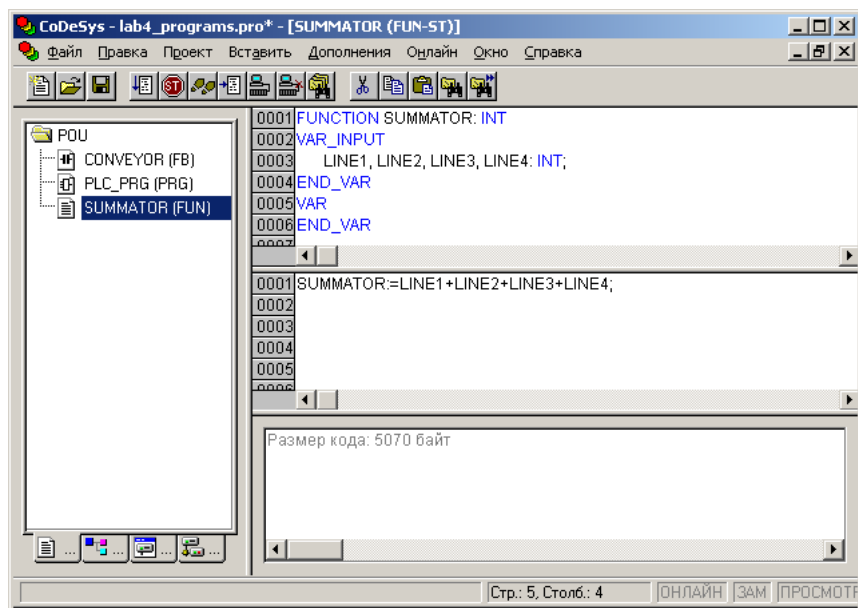


Рисунок 26. Функциональный блок модели конвейера для решения задачи.
Вариант на языке ST

Само приложение имеет достаточно простую структуру, тем более, что его удобно представлять в графическом виде. Четыре конвейера, сигналы с них обрабатываются детекторами фронтов, так как счетчик должен увеличиваться с каждым импульсом строго на единицу, это возможно, когда в подаваемом на счетчик сигнале от импульса остается только признак его переднего фронта, длящийся один цикл. Выходы счетчиков подаются на четыре входа функции сумматора, которая считает общее количество произведенных подшипников. Вид программы приведен на рисунке 27.

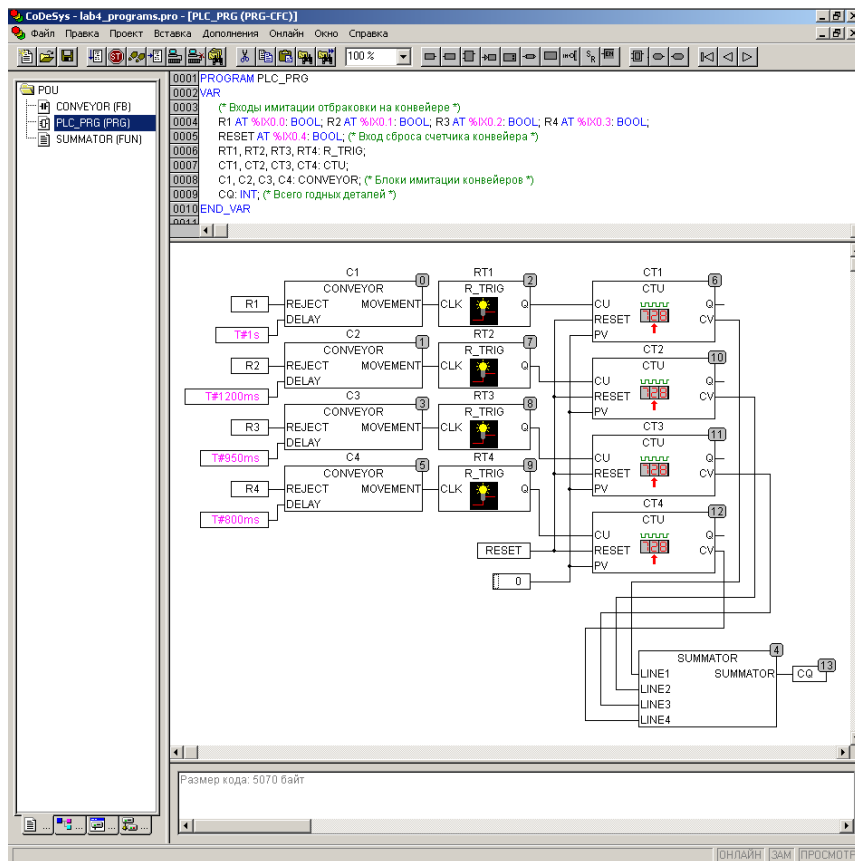


Рисунок 37. Основная программа на языке СFC

Если программа составлена верно, то в режиме исполнения значение CQ будет постепенно увеличиваться, каждую секунду на 4...6 единиц.

Чтобы поставить блок CONVEYOR, или функцию SUMMATOR, нужно поставить функциональный блок соответствующей кнопкой (в панели инструментов под главным меню третья справа от указателя масштаба) и в верхней строке блока после двойного клика «мышью» по ней с клавиатуры ввести название блока (функции). После нажатия на «Enter» блок примет вид, в котором его надо использовать: появятся названия входных и выходных параметров с точками для соединений.

В данной программе также показано, как правильно оформлять пользовательские комментарии в проектах.

Комментарии заключены в скобки: «(*)» и «(*)» и выделены зеленым цветом. Скобки вводятся пользователем, область комментариев подсвечивается сразу, комментарий может быть как односимвольным, так и многострочным, или даже многостраничным.

Таким образом, в данной лабораторной работе разработан первый в этом курсе сложный проект, состоящий из нескольких программных единиц.

5. Система визуализаций в CoDeSys.

5.1. SCADA-системы и визуализации.

Цель работы: научиться создавать простые пользовательские интерфейсы.

Обеспечение надежного, удобного и понятного человеко-машинного интерфейса — одна из главных задач проектирования АСУТП.

В современных АСУТП системы человеко-машинного интерфейса представлены панелями управления и SCADA-системами.

Под панелью управления подразумевается устройство визуального отображения информации об объекте управления в текстовом или графическом виде, подключенное к управляющему объектом ПЛК и имеющие органы управления (кнопки, сенсорная поверхность экрана, и т. п.), доступные для пользователя, с закрепляемыми за ними функциями управления объектом. Простейшие панели, отображающие только текстовую информацию здесь не рассматриваются.

SCADA-система — компьютер, с набором специального программного обеспечения, подключенный к ПЛК объекта управления, который отображает и регистрирует данные о нем и позволяет пользователю осуществлять управление объектом.

На SCADA-системах информация об объекте отображается в виде, понятном пользователю. Это могут быть не только значения параметров и интерфейсные элементы включения-выключения и установки значения параметров, но и схемы, графические изображения объектов управления и их частей, а состояние их может отображаться с применением элементов анимации. Часто применяются обновляющиеся графики значений параметров по оси времени (тренды) с информацией, сохраняемой в базе данных, или текстовых файлах.

Для разработки таких приложений применяются специальные среды разработки. Их инструментарий схож: набор элементов ввода информации и графических

примитивов со свойствами (цветами контура и заливки, реакцией на нажатие «мышью», текстом надписи, и т. д.), которые можно связать с переменными, значения которых получают с ПЛК, а также отправляют на ПЛК, если они были введены. Также вместе со SCADA-системой поставляют библиотеки стандартных элементов, используемых на изображениях: краны, трубопроводы, емкости, элементы конструкций, арматуры, элементы электросхем, и т. д.

В проекты CoDeSys можно добавлять элементы визуализации. Они используются преимущественно в отладочных целях, но тем не менее дают представление о том, как создаются приложения для SCADA-систем.

5.2. Создание визуализации в CoDeSys.

В качестве задачи, для которой будет создаваться визуализация, рассмотрим задачу из предыдущей лабораторной работы. Задача будет усложнена: требуется посчитать не только общее количество деталей, но и процент брака, и количество бракованных деталей, причем, сделать это для каждого конвейера. Конвейеры должны иметь возможность остановки. Сбрасывать значения счетчиков не нужно.

Изменения претерпит основной экран данной задачи (он представлен на рисунке 28) и модель конвейера, в которую добавится функция остановки и выдача сигнала о бракованном изделии, модель показана на рисунке 29. Даже в программу добавится функция подсчета процента брака — PF. Она изображена на рисунке 30. В функции условный оператор применен с целью обойти вычислительную ошибку деления на ноль, возникающую в начальный момент запуска конвейеров («всего деталей — ноль»).

Перейдем к созданию визуализации. В проекте CoDeSys визуализацией называют одно окно с элементами отображения, отображающее информацию. Для начала в левой нижней части экрана, под вертикальной панелью нужно выбрать третью вкладку («Визуализации»), при этом панель с названиями ROU сменится на панель с названиями визуализаций. Так как визуализаций еще не создавали, на

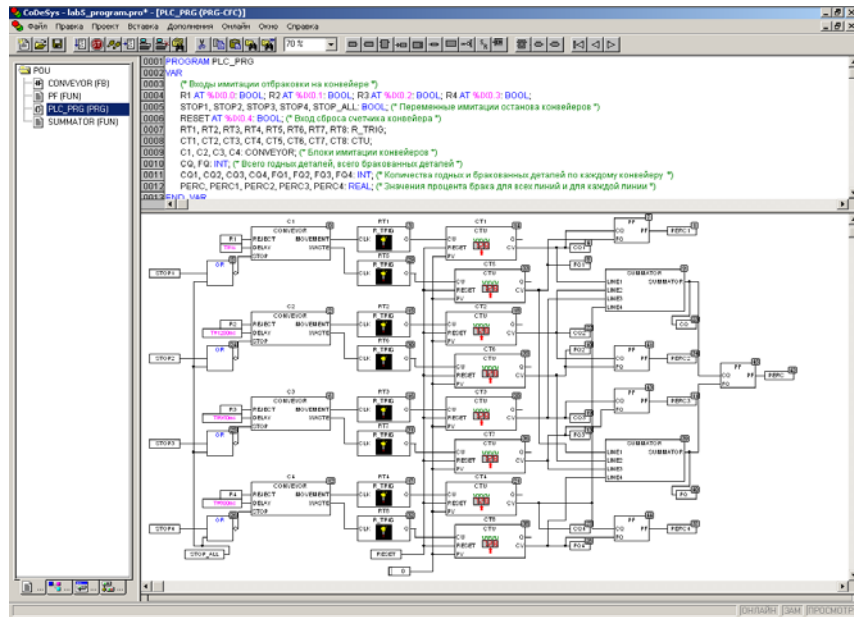


Рисунок 28. Видоизмененная основная программа задачи о конвейере на языке CFC.

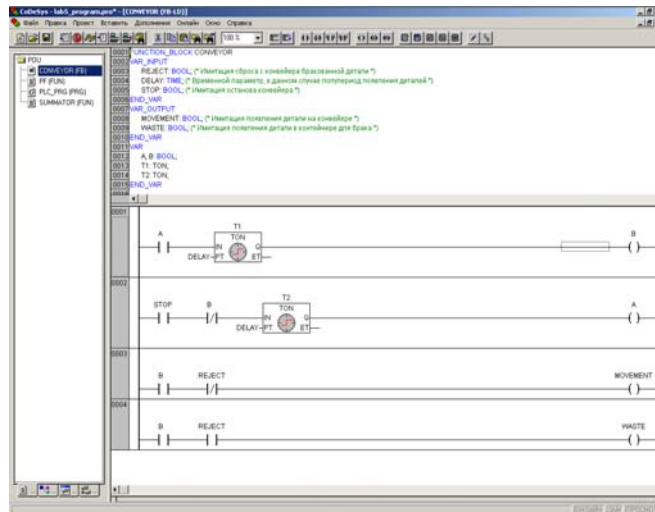


Рисунок 29. Видоизмененная основная программа задачи о конвейере на языке CFC.

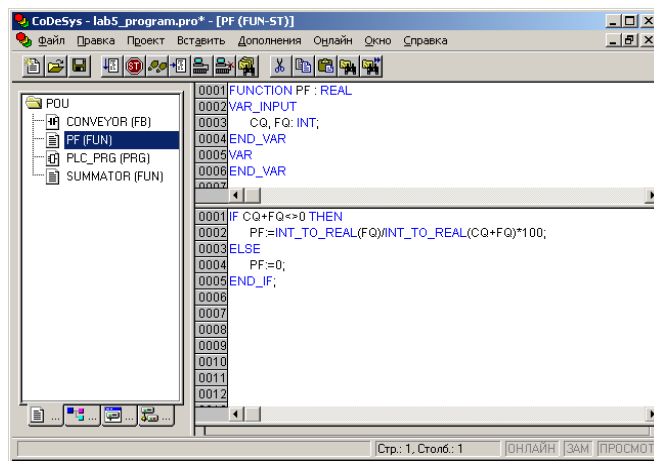


Рисунок 30. Видоизмененная основная программа задачи о конвейере на языке CFC.

Чтобы создать новый экран визуализации, необходимо щелкнуть правой кнопкой «мыши» на открывшейся панели слева и в появившемся выпадающем меню выбрать пункт «Добавить объект», как показано на рисунке 31. При этом справа появится пустое поле для размещения на нем графических объектов и, справа от индикатора текущего значения масштаба изображения, под главным меню, панель управления. На рисунке 32 можно увидеть экран с уже размещенными элементами пользовательского интерфейса для измененной задачи о конвейере. Размещение элементов на рабочем поле мало отличается от рисования графических примитивов в «CorelDRAW!», или «Inkscape», поэтому отдельно заострять внимания на самом процессе рисования не стоит.

Для удобства рисования поле размечено сеткой, и графические примитивы выравниваются по узлам сетки (режим «прилипания»). При необходимости отображение сетки и выравнивание можно отключить. При работе программы сетка исчезает.

При создании интерфейса пользователя для данной задачи необходимо знать, как организовать вывод числа на экран, как улучшить отображение текстовых меток, а также, как настроить работу интерфейсной кнопки.

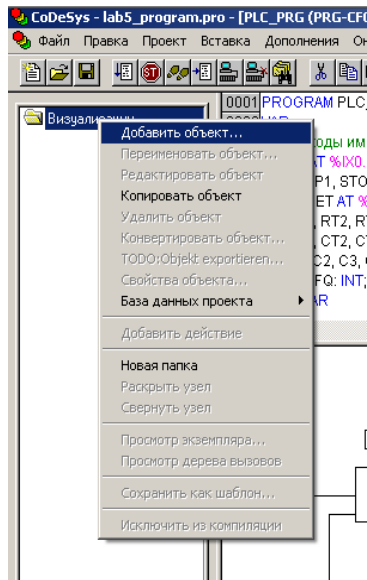


Рисунок 31. Создание новой визуализации.

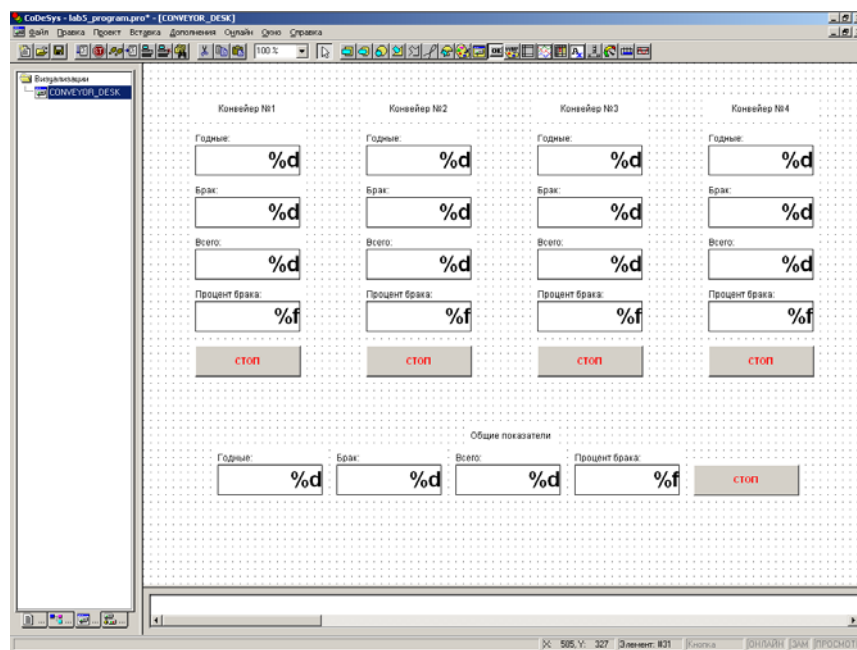


Рисунок 32. Создание новой визуализации.

У визуализаций CoDeSys нет специального примитива «текст», или «надпись», но у каждого графического примитива есть текстовая метка, у которой можно менять шрифт, размер, цвет, а также присваивать ей в виде строки шаблон, похожий на строковой шаблон в операторах вывода языка C, а также делать метку элементом не вывода, а ввода текста и чисел. На рисунке 33 изображена замена атрибутов цвета для прямоугольника с надписью с целью «спрятать» рамку вокруг надписи: ее цвет заменяется на цвет фона визуализации. На рисунке 34 изображена замена атрибутов шрифта. Для того, чтобы поменять значения атрибутов, цвет, шрифт, связать параметры примитива с переменными, и т. д., необходимо выделить элемент левой кнопкой «мыши», затем нажать на нем правую кнопку «мыши» и в появившемся меню выбрать пункт «Настройки». В результате на экране появится окно «Конфигурирование элемента...», в котором и будут перечислены все параметры и настройки выбранного элемента визуализации с их текущими значениями.

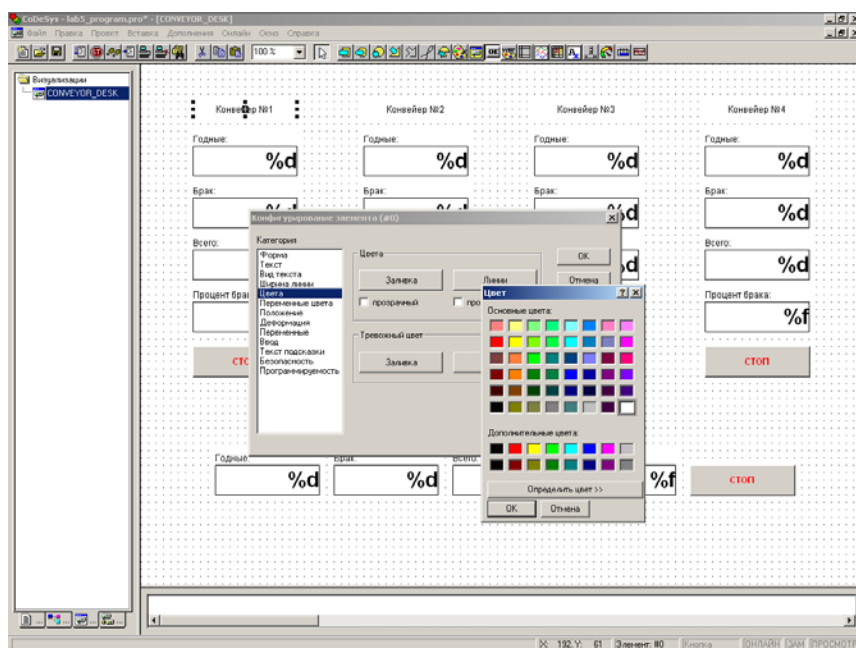


Рисунок 33. Придание рамке вокруг текста «невидимости» посредством настроек цвета.

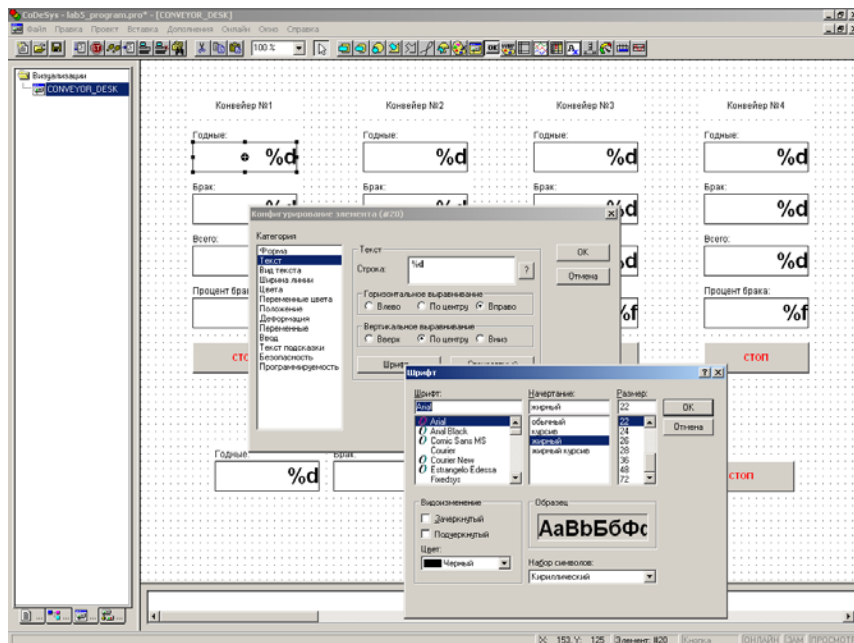


Рисунок 34. Изменение свойств шрифта надписи.

Чтобы поменять цвет линии прямоугольника, в таком окне (рисунок 33) необходимо выбрать в списке слева «Цвета», затем нажать кнопку «Линии» в группе «Цвета» и в появившемся диалоге выбора цветов выбрать нужный цвет. Далее нажать в каждом открытом окне «Ок» для подтверждения изменения параметров.

Чтобы поменять шрифт, или его размер, или способ его вывода, в таком окне (рисунок 34) необходимо выбрать в списке слева «Текст», затем нажать кнопку «Шрифт...» в группе «Текст», и в появившемся стандартном диалоге изменения свойств шрифта внести необходимые изменения. Далее нажать в каждом открытом окне «Ок» для подтверждения изменения параметров.

Вывод чисел в визуализациях осуществляется посредством тех же текстовых меток на графических примитивах. Для вывода числа, как и значения переменной любого типа, необходимо вместо текста (вместе с текстом) ввести символьный шаблон для ввода: там, где нужно вставить целое число, нужно поставить символы `%d` (см.

рисунок 35), а на месте числа с плавающей запятой необходимо ввести **%f**. Далее в том же окне необходимо выбрать в списке слева пункт «Переменные» и в поле «Вывод» занести имя той переменной, которая будет выводиться на текстовой метке выбранного и конфигурируемого примитива. Имя переменной записывается вместе с именем той программной единицы, в которой она была объявлена: сначала пишется имя ROU, потом через точку имя переменной, например так, как показано на рисунке 36. После окончания конфигурирования примитива нажать кнопку «Ок» для сохранения внесенных изменений.

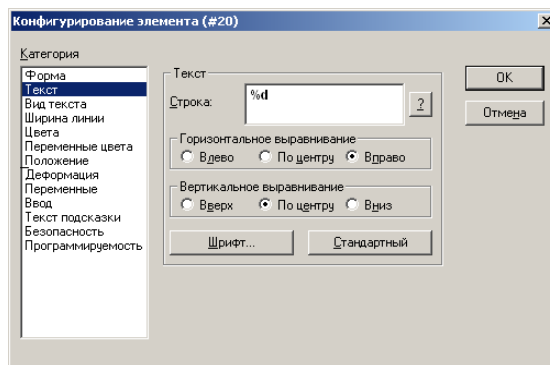


Рисунок 35. Значение строки надписи в виде шаблона для отображения числа.

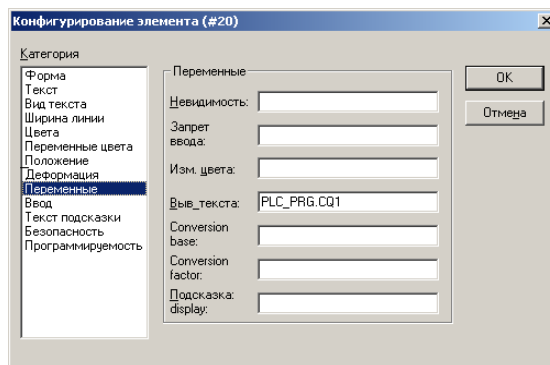


Рисунок 36. Привязка переменной к выводу числа вывода числа.

Здесь не раскрывается всех возможностей конфигурирования примитивов в визуализациях, а также

возможностей создания шаблонов для вывода значений переменных, эти данные частично даются в лекционном курсе, а полную информацию можно найти в [инструкция по CoDeSys].

В курсе не рассматриваются в подробностях особенности организации ограниченного доступа к приложениям визуализации CoDeSys, хотя такая возможность присутствует, и при конфигурировании переменных для отладочной визуализации необходимо обеспечить полный доступ всех групп пользователей к элементам отображения и управления, то есть проверить, чтобы в категории «Безопасность» радиокнопки групп «0»...«7» находились в положении «Полный доступ», и можно применить этот выбор ко всем визуальным элементам (см. рисунок 37).

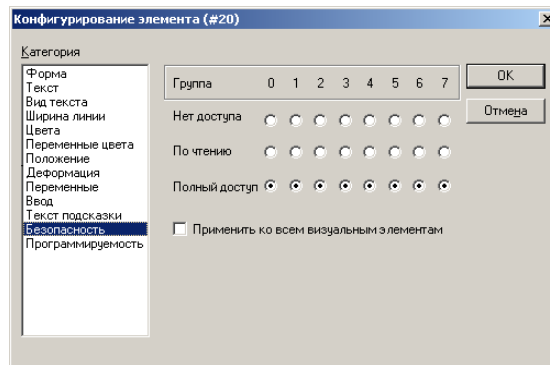


Рисунок 37. Ограничение доступа к объекту.

Конфигурирование кнопки происходит следующим образом: в окне «Конфигурирование элемента...», выведенном для визуального элемента-кнопки выбирается вкладка «Ввод». Существует два режима работы кнопки: смена состояния по нажатию и смена состояния во время нажатия. Первое можно описать словами «нажал — включил, еще раз нажал — выключил», а второе — «нажал — замкнул, отпустил — разомкнул». Чтобы включить первый режим работы кнопки, необходимо поставить отметку возле надписи «Переменная переключения» и ввести имя переменной в поле ввода, располагающееся справа от надписи, чтобы включить второй режим, необходимо то же самое проделать с отметкой

и полем для ввода «Переменная-кнопка». Ввод переменной производится точно так же, как и для вывода текста: сначала имя программы, в которой переменная объявлена, затем точка, а следом имя переменной (см. рисунок 38).

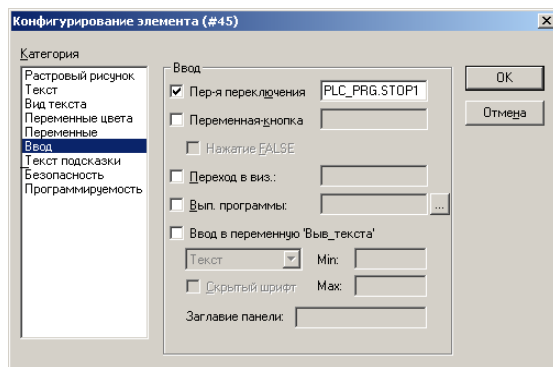


Рисунок 38. Привязка переменной к кнопке.

После того, как программа полностью создана и безошибочно скомпилирована, визуализация закончена, и все ее элементы настроены, можно приступать к запуску программы. Переход между программой и визуализациями осуществляется с помощью закладок левой боковой панели в нижней части экрана. Работая визуализация показана на рисунке 39, примерно так же выглядят и разрабатываются приложения визуализации для реальных промышленных АСУТП.

В качестве замечания по работе стоит сказать о том, что данное упражнение содержит в себе *модель-имитатор* конвейера. Если применять такую программу на настоящем конвейере, то необходимость в такой программе отпадает. И одним из способов проверки работоспособность приложения при отсутствующем объекте является создание подобных имитаторов.

В промышленных SCADA-системах, не связанных со средой разработки программ для ПЛК, как это сделано в CoDeSys, кроме возможностей настроить поведение объекта в зависимости от значений переменных почти везде присутствует встроенный язык программирования, позволяющий задавать более сложное поведение SCADA-

приложения.

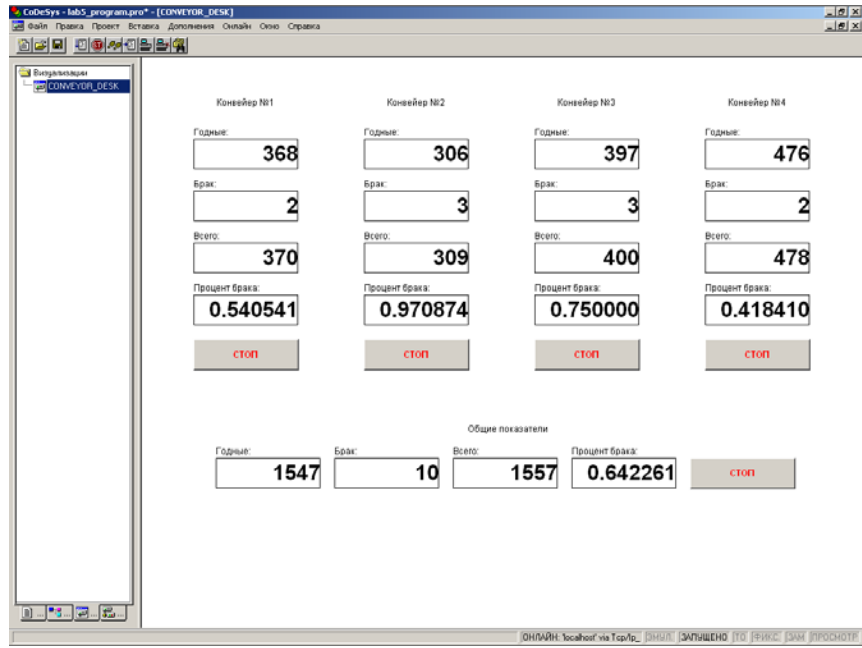


Рисунок 39. Визуализация в работе.

6. Создание ПИД-регулятора на ПЛК и регулирование температуры.

6.1. Теоретические основы ПИД-регулирования.

Цель работы: научиться работать с вычислениями и программировать алгоритмы регулирования.

Для регулирования многих технологических процессов достаточно регуляторов первого, или второго порядка. Самыми распространенными видами регулирования являются следующие:

- *пропорциональное регулирование*, когда регулирующее воздействие пропорционально сигналу на входе регулятора; обычно это сигнал ошибки между регулируемой величиной и ее заданным значением,
- *интегральное регулирование*, когда регулирующее воздействие пропорционально определенному интегралу входного сигнала по времени, взятому от начала отсчета до текущего момента,
- *дифференциальное регулирование*, когда регулирующее воздействие пропорционально производной входного сигнала по времени,
- *ПИД-регулирование (пропорционально-интегрально-дифференциальное)*, когда и сам входной сигнал, и его производная, и его определенный интеграл присутствуют в процессе регулирования с определенными долями, определяемыми коэффициентами; сюда отнесем и ПИ-регулирование, особенность которого заключается лишь в том, что коэффициент при производной входного сигнала равен нулю.

В промышленности для реализации пропорционально-дифференциально-интегрального закона регулирования широко распространены универсальные ПИД-регуляторы. Такие приборы представляют собой электронные устройства с аналоговой, либо цифровой (микропроцессорной) схемой, в

последнее время гораздо более распространены вторые.

Также к числу микропроцессорных устройств относятся и ПЛК, на котором также можно реализовать алгоритм ПИД-регулирования, что в настоящее время также находит частое применение.

Микропроцессорные устройства, в отличие от аналоговых, работают с дискретными данными и закон ПИД-регулирования в них реализован несколько иначе.

Структурная схема непрерывного ПИД-регулятора представлена на рисунке 40. Закон ПИД-регулирования для непрерывных систем можно записать уравнением:

$$K_P S_{\text{ex}}(t) + K_I \int_0^t S_{\text{ex}}(\tau) d\tau + K_D \frac{dS_{\text{ex}}(t)}{dt} = S_{\text{вых}}(t) \quad (1)$$

здесь $S_{\text{ex}}(t) = e(t) + e_0$, где $e(t)$ — регулируемая величина, e_0 — требуемое значение величины; фактически $S_{\text{ex}}(t)$ — ошибка регулирования.

Запишем уравнение для дискретного ПИД-регулятора и найдем, как вычислить коэффициенты дискретного ПИД-регулятора, исходя из коэффициентов непрерывного ПИД-регулятора.

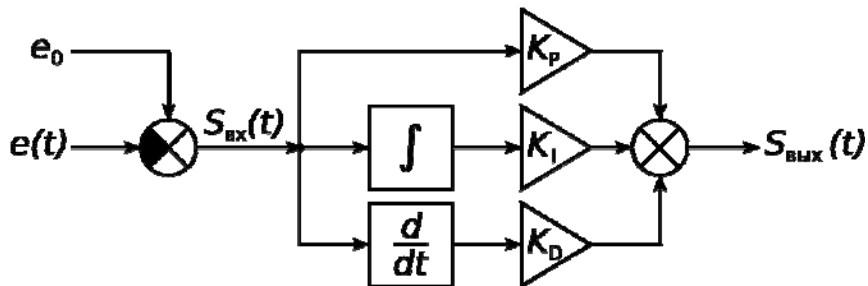


Рисунок 40. Непрерывный ПИД-регулятор.

Перепишем то же уравнение, предположив дискретность процессов по времени: если две соседние точки на оси времени будут отстоять друг от друга не на бесконечно малую величину, а на некоторое значение T , называемое периодом дискретизации, то в уравнении (1) интеграл превратится в сумму, а производная — в отношение разностей. Уравнение при этом примет вид:

$$K_P S_{\text{ex}}[nT] + K_I \sum_{i=0}^n [S_{\text{ex}}[iT]] + K_D \frac{S_{\text{ex}}[nT] - S_{\text{ex}}[(n-1)T]}{T} = S_{\text{вых}}[t] \quad (2)$$

Далее вынесем за знак суммы и из знаменателя дроби период дискретизации, соединив его с соответствующими коэффициентами и перепишем функцию ошибки в дискретном виде, независимо от непрерывного времени (без T под знаком функции). Уравнение примет следующий вид:

$$\begin{aligned} K_P S_{\text{ex}}[n] + K_I T \sum_{i=0}^n S_{\text{ex}}[i] + \frac{K_D}{T} S_{\text{ex}}[n] - S_{\text{ex}}[n-1] &= \\ &= K_{P,\text{дискр.}} S_{\text{ex}}[n] + K_{I,\text{дискр.}} \sum_{i=0}^n S_{\text{ex}}[i] + \\ &+ K_{D,\text{дискр.}} S_{\text{ex}}[n] - S_{\text{ex}}[n-1] = S_{\text{вых}}[n] \end{aligned} \quad (3)$$

Таким образом, соотношения между коэффициентами ПИД-регуляторов дискретного и непрерывного можно описать следующим образом:

$$K_{P,\text{дискр.}} = K_P; \quad K_{I,\text{дискр.}} = K_I T; \quad \frac{K_{D,\text{дискр.}}}{T} = K_D. \quad (4)$$

Структурная схема дискретного ПИД-регулятора показана на рисунке 41.

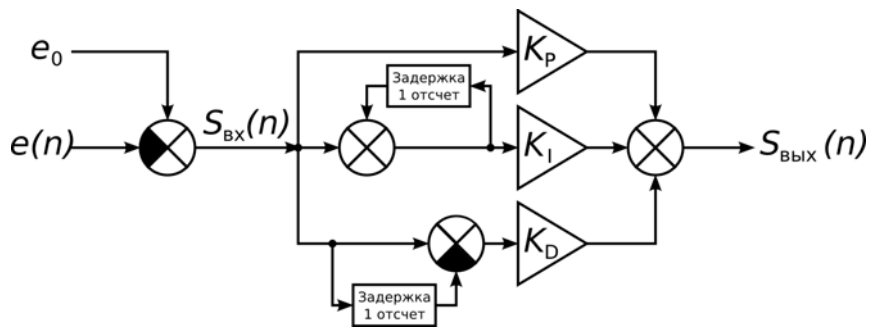


Рисунок 41. Структурная схема дискретного ПИД-регулятора

Кроме аппаратных реализаций ПИД-регуляторы многократно запрограммированы во всевозможных библиотеках подпрограмм и функциональных блоков под

различные типы ПЛК и промышленных ЭВМ. Тем не менее, задача, решаемая в данной лабораторной работе, состоит в проектировании и программировании дискретного ПИД-регулятора и его использовании в системе регулирования температуры, это поможет понять их принципы работы и способы применения.

6.2. Программирование ПИД-регулятора на ПЛК.

Задача. Разработать систему регулирования температуры на базе ПЛК «ОВЕН» для имитатора термокамеры, имеющегося в учебном комплекте. Не использовать ПИД-регулятор в библиотеке компонентов, разработать регулятор самостоятельно. Создать визуализацию, позволяющую видеть текущее и заданное значения температуры и график изменения температуры за последние несколько минут, настраивать коэффициенты ПИД-регулятора, а также вручную выключать и включать нагревательный элемент.

До этой лабораторной работы рассматривались примеры, задействующие только дискретные входы и выходы; здесь же возникла необходимость использовать аналоговый вход, и для этого нужно его настроить. Для настройки входов и выходов, как было указано на стр. 17, необходимо внизу левой панели окна CoDeSys выбрать четвертую слева закладку «Ресурсы», затем в иерархическом дереве выбрать «Конфигурация ПЛК». Далее на рабочем поле CoDeSys появится иерархическое дерево устройств, имеющихся в ПЛК с переменными, доступными, как входы, или выходы.

Четырем аналоговым входам ПЛК по умолчанию соответствуют пункты «Unified signal sensor», обозначающие стандартные входы по напряжению. Термосопротивление подключено к четвертому аналоговому входу, в списке он самый последний, и вход необходимо перенастроить для работы с термосопротивлением. Для этого на последнем пункте «Unified signal sensor» необходимо нажать правую

кнопку «мышь» и в появившемся меню выбрать «Заменить элемент», затем «RTD-sensor» (см рисунок 42).

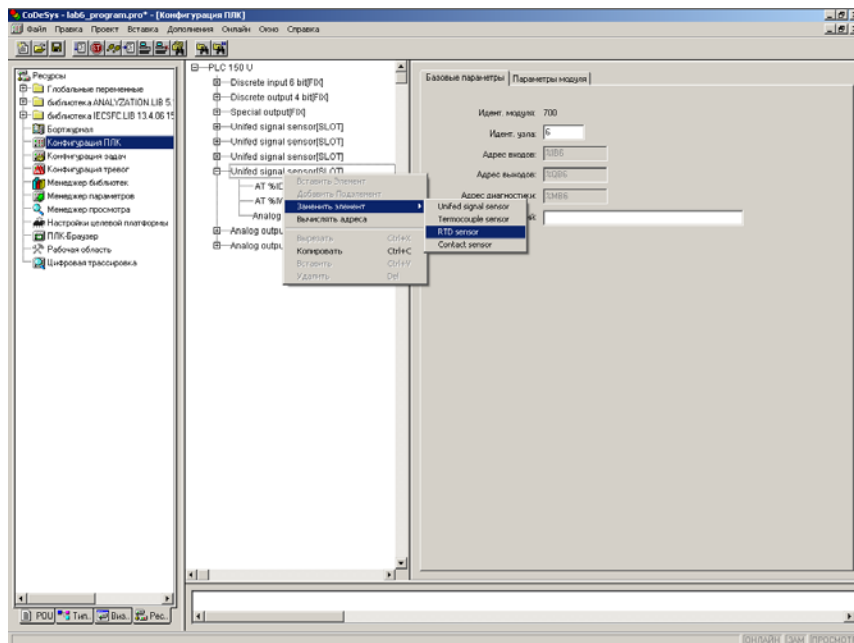


Рисунок 42. Настройка аналогового входа для подключения термосопротивления (вместо Unified signal sensor выбрать RTD sensor).

После этого надпись «Unified signal sensor» в дереве устройств сменится на надпись «RTD-sensor», и справа откроются настройки аналогового входа для подключения термосопротивления. Те значения настроечных параметров, которые являются корректными, показаны на рисунке 43. Аналоговый вход настроен, переходим к проектированию приложения.

В термокамере присутствует только дискретное управление нагревом, то есть его включение и отключение посредством релейного выхода, но, по условию задачи, нужно обеспечить плавную регулировку нагрева. Каким образом можно решить подобную задачу, имея только дискретный вход?

Время нагрева нагревательного элемента в термокамере с 20 до 100 °С составляет около трех минут, то есть за одну секунду происходит повышение температуры в среднем на

0,33 °С. Нагревание происходит плавно, на небольшом участке почти линейно, остывание происходит медленнее, чем нагревание. Можно сказать, температура термокамеры по отношению к поданному на нагревательный элемент напряжению изменяется по аperiodическому закону.

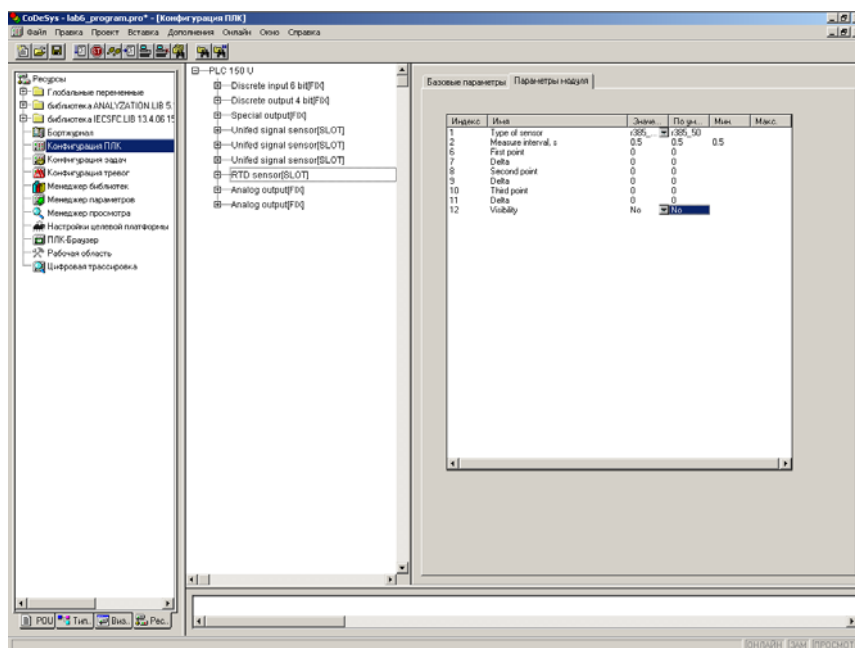


Рисунок 43. Настройка аналогового входа с термосопротивлением.

В таких случаях плавный нагрев можно реализовать с помощью широтно-импульсной модуляции (ШИМ). При этом на нагревательный элемент питание необходимо подавать импульсами определенной длительности и с заданной периодичностью. необходимо, не меняя периода импульсов, устанавливать их длительность в зависимости от требуемой температуры нагрева; при увеличении длительности импульсов температура будет расти, а при уменьшении — падать. Максимальная скорость нагрева достигается при непрерывной работе нагревателя, то есть, когда длительность импульсов равна периоду их следования, а максимальная скорость остывания достигается при полном отключении нагревателя, то есть при длительности импульсов равной нулю. Если в течение некоторого времени подавать импульсы

неизменной длительности на нагреватель, в термокамере устанавливается определенная средняя температура, и если период импульсов значительно меньше времени нагрева или остывания термокамеры, то изменения температуры при подаче импульсов на нагревательный элемент можно считать незначительными. Чем выше частота следования импульсов, тем меньше погрешность регулирования температуры.

Будем считать зависимость установившейся средней температуры от длительности импульсов линейной.

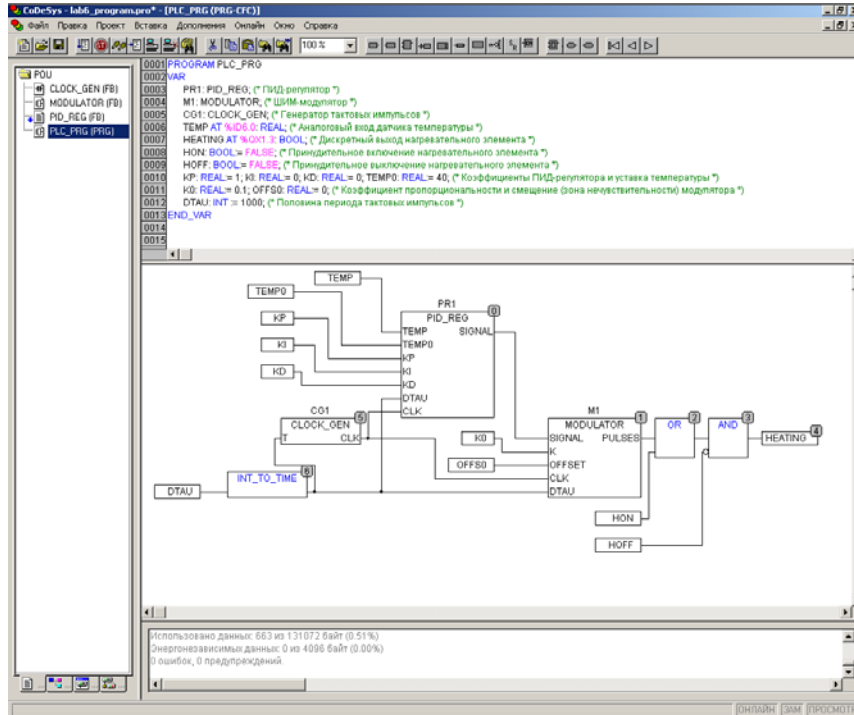


Рисунок 44. Регулировка температуры, основная программа.

ПИД-регулирование — задача, которую удобно представить в виде схемы, поэтому целесообразно использование языка CFC.

На рисунке 44 представлен общий вид программы ПИД-регулирования. В отличие от задачи о конвейере, здесь применим методы проектирования «сверху вниз», то есть от общего к частному. Сперва следует определить общий вид программы и предположить, какие входные и выходные

переменные будут у функциональных блоков.

Основные функции, которые можно выделить в данном регуляторе: собственно, ПИД-регулирование, преобразование выходного сигнала регулятора в частотный сигнал на коммутатор питания нагревательного элемента и задающие генераторы тактовых импульсов для дискретного ПИД-регулятора и для ШИМ-модулятора. Поскольку обрабатываемые сигналы имеют приблизительно одну постоянную времени, целесообразно реализовать один тактовый генератор и для регулятора и для ШИМ-модулятора.

Таким образом, основными функциональными блоками являются ПИД-регулятор, генератор тактовых импульсов и ШИМ-модулятор.

Для ПИД-регулятора необходимыми входными данными будут являться:

- значение регулируемого сигнала и его заданное значение, чтобы найти ошибку регулирования, на основе которой рассчитать воздействие,
- коэффициенты регулирования: пропорциональная, интегральная и дифференциальная составляющие,
- тактовые импульсы,
- период следования тактовых импульсов (для расчета коэффициентов дискретного ПИД-регулятора).

Выходом ПИД-регулятора будет являться регулирующее воздействие, преобразуемое модулятором к нужному виду.

Для модулятора входными сигналами будут являться

- регулирующее воздействие,
- коэффициент пропорциональности и смещение сигнала, для пересчета управляющего сигнала в длину ШИМ-импульсов
- тактовые импульсы с генератора,
- постоянная времени.

Выходом модулятора будут импульсы, управляющие коммутацией питания нагревательного элемента термокамеры.

Для генератора тактовых импульсов входным сигналом будет период тактовых импульсов, а выходным сигналом — тактовые импульсы.

Алгоритм ПИД-регулирования можно реализовать на языке ST, он показан на рисунке 45. Как и требуется, реализован дискретный алгоритм ПИД-регулирования, коэффициенты интегрирующей и дифференцирующей составляющих пересчитываются в соответствии с равенствами (4).

```

FUNCTION_BLOCK PID_REG
VAR_INPUT
    TEMP: TEMP0; (* Входной сигнал и его установочное значение *)
    KP, KI, KD: REAL; (* Коэффициенты ПИД-регулятора *)
    DTAU: TIME; (* полупериод дискретизации *)
    CLK: BOOL; (* тактовый сигнал *)
END_VAR
VAR_OUTPUT
    SIGNAL: REAL; (* Сигнал регулирования *)
END_VAR
VAR
    ERR: REAL := 0; (* Предыдущее значение ошибки *)
    ERR_DERR: REAL; (* Ошибка и ее производная *)
    ERRSUM: REAL := 0; (* Интеграл ошибки *)
    RT: R_TRIG;
END_VAR

RT(CLK) = CLK;
IF RT.Q THEN
    (* Вычисление коэффициентов дискретного ПИД регулятора
    по коэффициентам непрерывного регулятора *)
    KI := KI * (TIME_TO_INT(DTAU) / 500.0);
    KD := KD * (TIME_TO_INT(DTAU) / 500.0);
    ERR := TEMP0 - TEMP;
    ERRSUM = ERRSUM + ERR;
    DERR = ERR - ERR1;
    (* Вычисление значения сигнала регулирования *)
    SIGNAL := KP * ERR + KI * ERRSUM + KD * DERR;
    (* Сохранение текущего значения сигнала в предыдущее *)
    ERR1 = ERR;
END_IF
END_FUNCTION_BLOCK

```

Рисунок 45. ПИД-регулятор, функциональный блок.

Алгоритм модулятора модулятора состоит из формирователя импульсов (см. рисунок 46) в верхней части рисунка и расчета длительности импульса в нижней части рисунка. Также из нижней части рисунка вверх поступают некоторые вспомогательные сигналы для подачи на импульсный выход постоянного высокого или низкого логического уровня. В алгоритме использована модель RS-триггера *RSI* — бистабильная схема, устанавливающая *Q1* в единицу при *SET=1*, *RESET=0* и сбрасывающая *Q1* в ноль при *SET=0*; *RESET=1*.

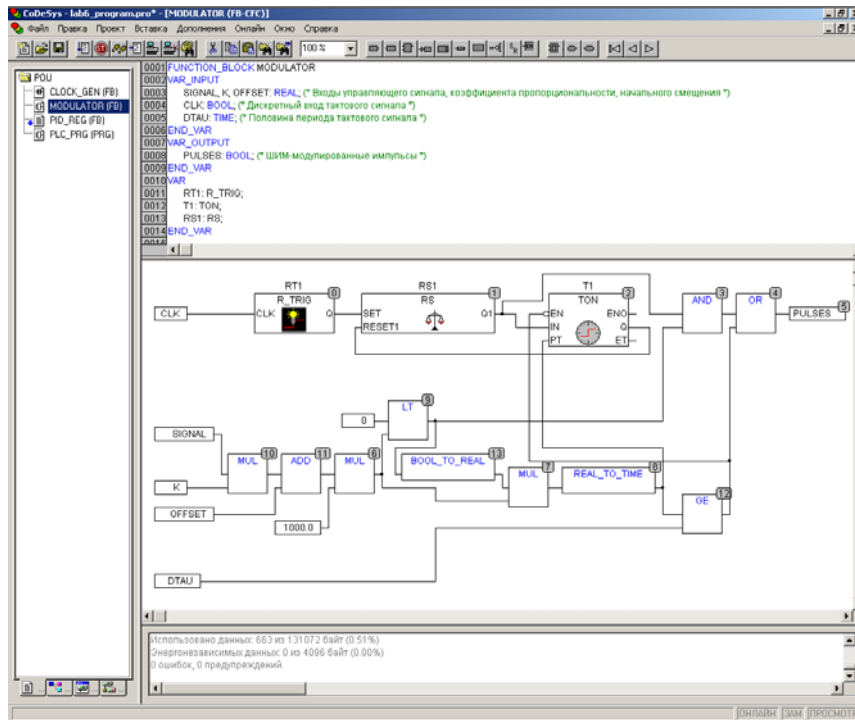


Рисунок 46. Модулятор, функциональный блок.

Генератор тактовых импульсов, представленный на рисунке 47, сокращен до двух строк кода и одной внутренней переменной, поскольку реализован в виде функционального блока, и все его переменные находятся в памяти. Переменная *V*, описанная на рисунке 47, не задействована. Время задается переменной типа TIME.

Общий вид визуализации показан на рисунке 48. Помимо уже применявшихся в текстовых метках шаблонов для вывода целых и вещественных чисел здесь использованы шаблоны для вывода вещественных чисел с заданной точкой. С помощью такого шаблона выводится, например, заданная и фактическая температура в термокамере: **%3.2F** означает следующее: вещественное число, ограниченное тремя знаками до запятой с точностью, ограниченной двумя знаками после запятой (в данном случае — до и после десятичной точки). Также в визуализации для наблюдения температуры за некоторый прошедший период времени применен

графический тренд, с которым связаны значения заданной и фактической температуры.

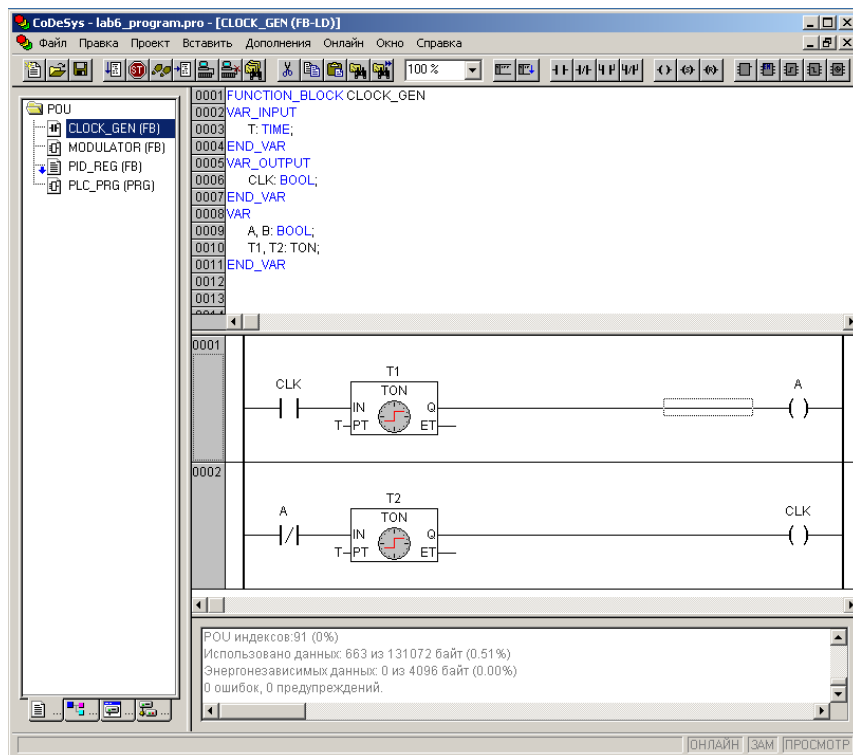


Рисунок 47. Генератор тактовых импульсов, функциональный блок.

Настройка тренда показана на рисунках 48...53. Настройка начинается с внешнего вида осей: масштаб по осям, количество делений, и т. д. Выделив тренд, нажать на нем правую кнопку «мыши», выбрав «настройки». Категория «Тренд», кнопка «Горизонтальная ось». На рисунке 49 показаны следующие заданные значения:

- вертикальные линии сетки нанесены с интервалом в 20 сек.;
- полная длина шкалы по горизонтальной оси равна 5 минут;
- основные (длинные) деления нанесены с интервалом в 10 секунд;
- дополнительные деления нанесены с интервалом в 2

- секунды;
- подписи значений под делениями шкалы следуют каждую минуту;
- новое значение выдается каждые 200 миллисекунд.

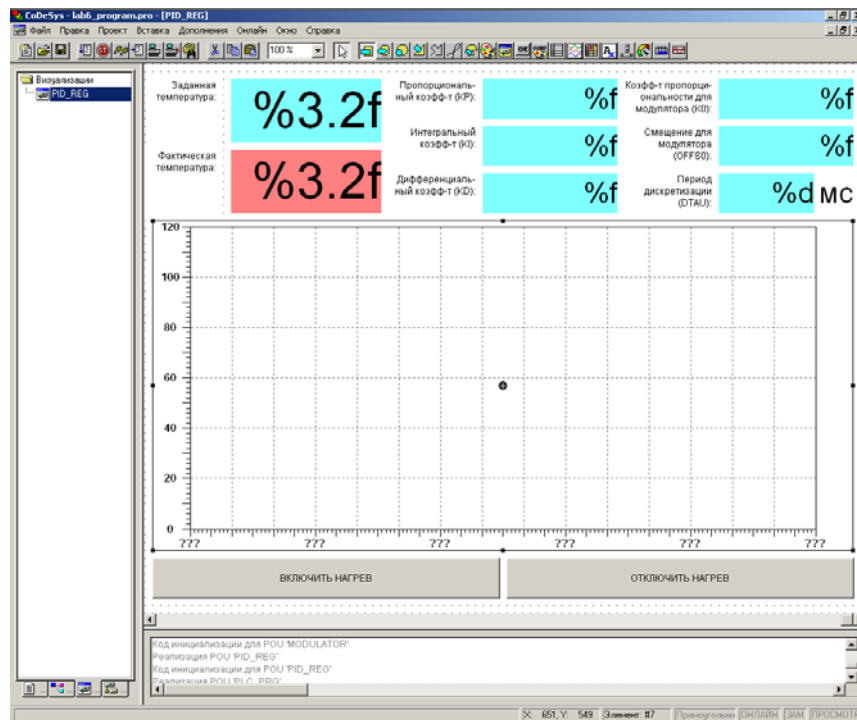


Рисунок 48. Общий вид визуализации.

Задание остальных параметров, таких, как цвет линий сетки, понятно из рисунка 49.

На рисунке 50 показана настройка вертикальной оси тренда. Окно настроек выглядит почти так же, как и окно настроек оси времени с той лишь разницей, что вместо временного интервала в его нижнем левом углу указывается максимальное и минимальное значения отображаемой величины, в данном случае на тренде может отображаться температура от 0 до 120 °С.

Для отображения на тренде необходимо указать переменные, пример настройки показан на рисунке 51. В окне настроек нужно выбрать категорию «Тренд» и нажать кнопку

«Выбрать имя переменной». Появившееся окно отображает список переменных, значения которых выводятся на данном тренде.

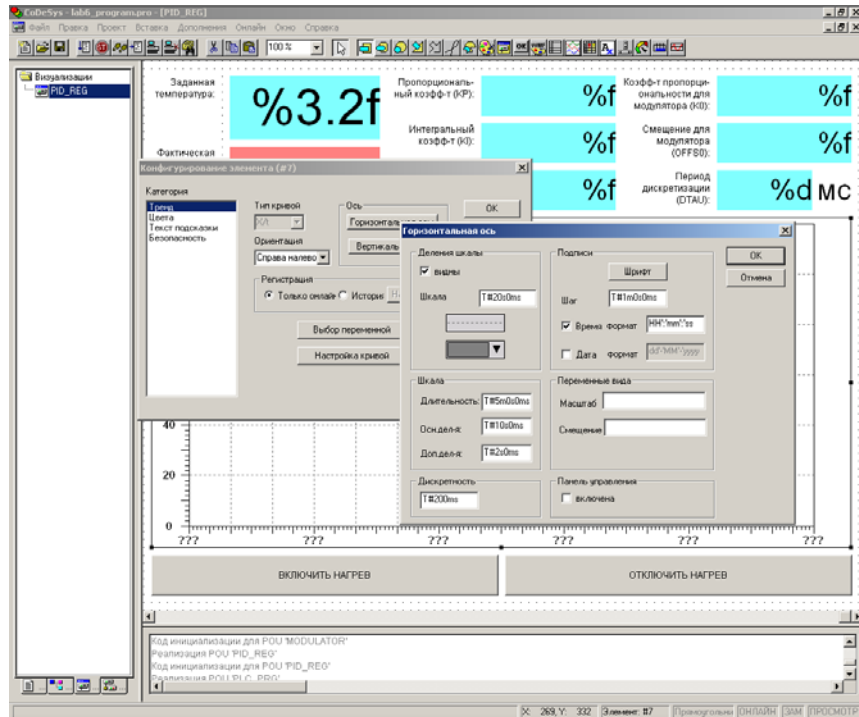


Рисунок 49. Настройка параметров горизонтальной оси тренда.

Переменных может быть несколько, их можно добавлять, убирать, менять внешний вид кривой, цвет, и т. д., такие настройки не сложны и интуитивно понятны. После осуществления всех необходимых настроек нужно подтвердить их верность, нажав кнопку «ОК».

На рисунке 52 изображена настройка текста подсказки, появляющегося при наведении курсора «мыши» на тренд.

На рисунке 53 изображена привязка переменной, включающей нагревательный элемент в ручном режиме к соответствующей кнопке (в окне настройки кнопки).

На рисунке 54 изображен экран работающего регулятора температуры вскоре после включения. Как видно из коэффициентов регулятора, регулирование происходит только по пропорциональному алгоритму.

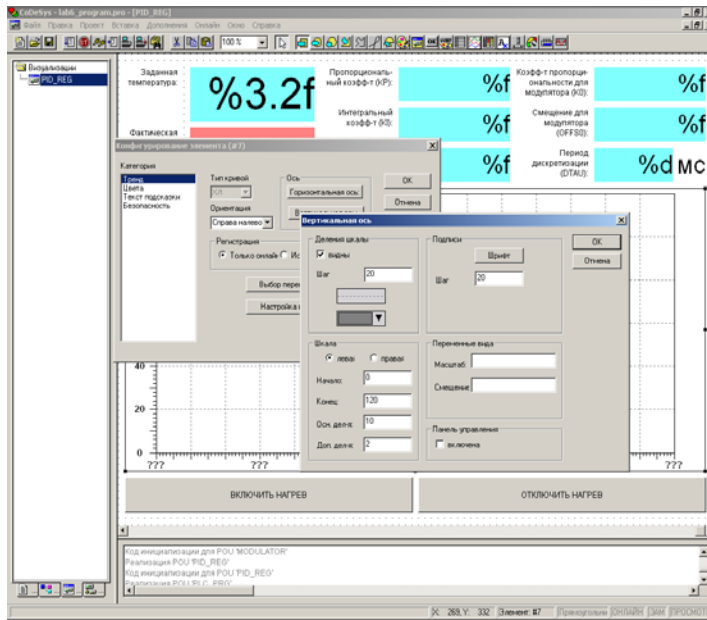


Рисунок 50. Настройка параметров вертикальной оси тренда.

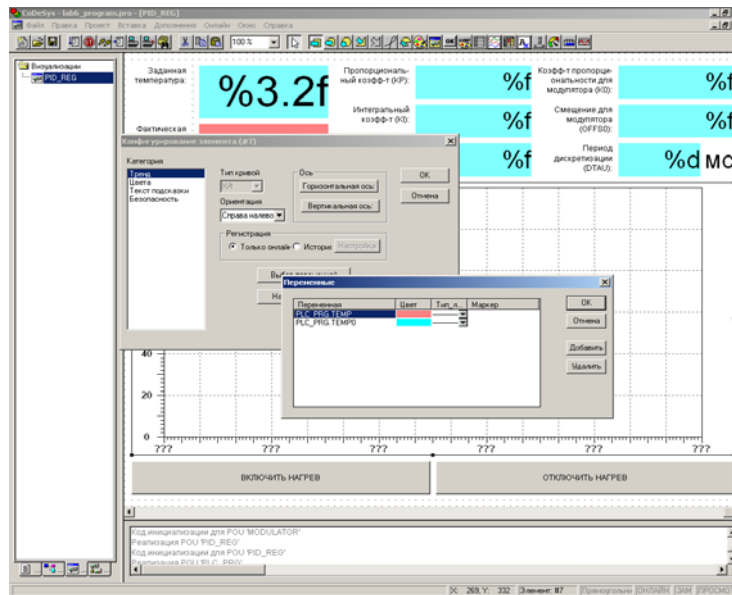


Рисунок 51. Добавление в тренд графиков температуры.

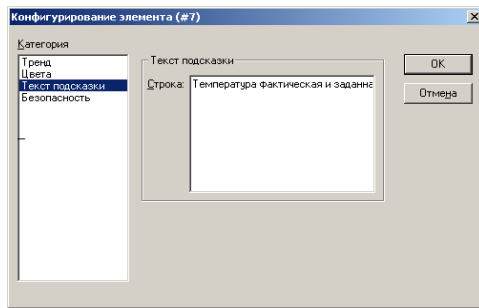


Рисунок 52. Настройка всплывающей подсказки тренда

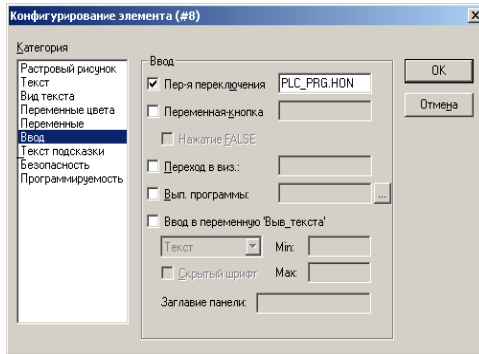


Рисунок 53. Конфигурирование кнопки ручного включения нагревателя.

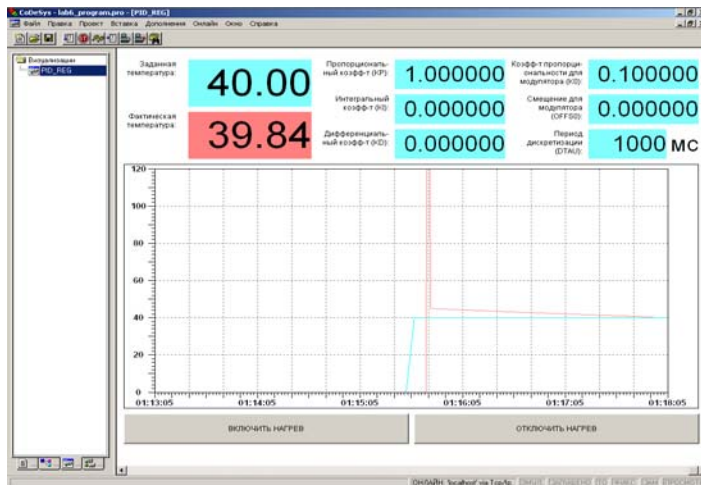


Рисунок 54. Визуализация в работе.

Содержание

Введение.....	5
1.Настройка CoDeSys. Новый проект в среде «CoDeSys».	
Построение таблиц истинности логических операций.	7
1.1. Общие сведения, установка среды программирования.....	7
1.2. Настройка связи компьютера с ПЛК, новый проект в «CoDeSys»	11
1.3. Первая программа на ПЛК. Таблицы истинности логических	
операций.....	15
2.Программирование на языке LD. Таймеры, счетчики и	
детекторы фронтов.	23
2.1. Общие сведения о программе и программировании для ПЛК.	23
2.2. Задача 1. Демонстрация работы реверсивного счетчика и	
детекторов фронтов.	26
2.3. Задача 2. Управление освещением в комнате.....	30
2.4. Задача 3. Программный генератор периодических импульсов.	32
3.Основные возможности языков	
ST, CFC и FBD.	35
3.1. Особенности построения программ на языках ST, CFC и FBD.	35
3.2. Решение на языках ST, CFC и FBD задачи об управлении	
включением света.	37
4.Программные единицы: функции, программы и	
функциональные блоки, создание структуры приложения. ...	41
4.1. Структура приложений в среде «CoDeSys».	41
4.2. Пример проектирования структуры приложения и его реализации.	
.....	42
5.Система визуализаций в CoDeSys.	49
5.1. SCADA-системы и визуализации.	49
5.2. Создание визуализации в CoDeSys.	50
6.Создание ПИД-регулятора на ПЛК и регулирование	
температуры.	60
6.1. Теоретические основы ПИД-регулирования.....	60
6.2. Программирование ПИД-регулятора на ПЛК.....	63
Список использованных источников.	75

Список использованных источников.

1. Густав Олссон, Джангуидо Пиани Цифровые системы автоматизации и управления. - СПб.: Невский Диалект, 2001. — 557 с.: ил.
2. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В. П. Дьяконова. — М.: СОЛОН-Пресс, 2004. — 256 с.: ил. — (Серия «Библиотека инженера»)