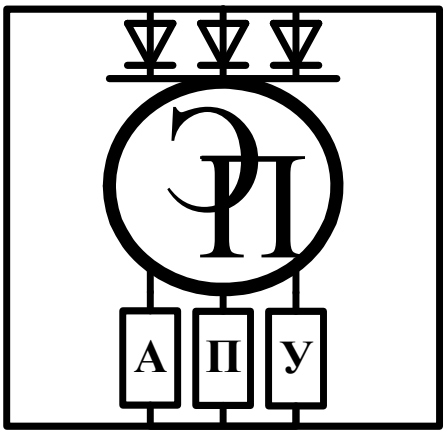
ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и АПУ»

СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Методические рекомендации к практическим занятиям для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» дневной формы обучения



Могилев 2018

УДК 658.012.011.56: 621.865.8
ББК 32.965: 32.816
С 40

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой ЭП и АПУ «7» февраля 2018 г., протокол № 7

Составитель ст. преподаватель А. В. Янкович

Рецензент С. В. Болотов

Методические рекомендации предназначены к практическим занятиям для студентов направления подготовки 15 03 06 «Мехатроника и робототехника» дневной формы обучения.

Учебно-методическое издание

СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Ответственный за выпуск Г. С. Леневский

Технический редактор А. А. Подошевко

Компьютерная вёрстка Н. П. Полевничая

Подписано в печать . Формат 60×84 /16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 46 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилёв.

Содержание

1 Практическая работа № 1. Разработка программ управления дискретной автоматикой

Цель практической работы: получение теоретических сведений о системах логического управления и практических навыков проектирования систем логического управления (СЛУ).

Краткие теоретические сведения

Системы логического управления (СЛУ) бывают комбинационными и последовательными. Последовательные системы называются также событийно-управляемыми автоматами.

В комбинационных СЛУ выходные сигналы формируются только при определённых комбинациях входных логических сигналов, принимающих значения 0 или 1. В последовательных СЛУ выходные сигналы зависят не только от комбинации входных, но и от последовательности их поступления во времени, что обеспечивается наличием элементов памяти.

В настоящее время в промышленных системах автоматики СЛУ реализуются с использованием программируемых логических контроллеров (ПЛК).

Для описания работы СЛУ используется математический аппарат двузначной (булевой) алгебры логики.

Основным понятием алгебры логики является логическая функция. Логическая функция выражает зависимость выходной переменной от значения входных переменных. Все переменные могут принимать только два значения Ложь или Истина (FALSE или TRUE), которые условно обозначают 0 или 1. Значение выходной переменной определяется входными переменными и связывающими их логическими действиями.

Все действия над переменными в бинарной алгебре выполняются с помощью следующих основных операций, которые наглядно иллюстрируются соответствующими релейно-контактными схемами.

В настоящее время релейная автоматика почти не используется, но, имея навыки построения контактных СЛУ и используя современный язык LD в среде CoDeSys, можно легко эти схемы перевести в программу для ПЛК. Язык LD и был в свое время разработан для инженеров, знающих релейные автоматы, но не владеющих навыками программирования на языках высокого уровня.

Логическое умножение.

Логическое умножение (конъюнкция, функция «И») равнозначно последовательному соединению контактов. Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.1). Очевидно, что только в одном случае результатом логического умножения станет единица, т. е. лампа Л «сработает», если замкнуть контакты и a, и b (рисунок 1.1). Из этой словесной формулы союз И перешел в обозначение функции (как синоним логическому умножению) и в название

бесконтактного элемента (рисунок 1.2). Написание: Л = а · б.

На выходе такого элемента появится сигнал (потенциал), если на его входах a и b одновременно действуют единичные сигналы (рисунок 1.3).

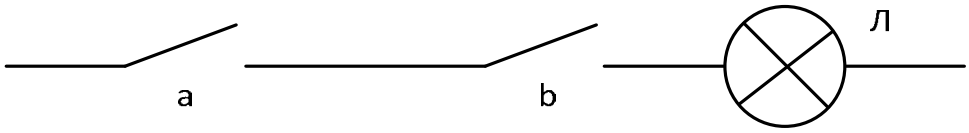


Рисунок 1.1 – Релейно-контактный вариант реализации

Таблица 1.1

a	b	Л
0	0	0
0	1	0
1	0	0
1	1	1

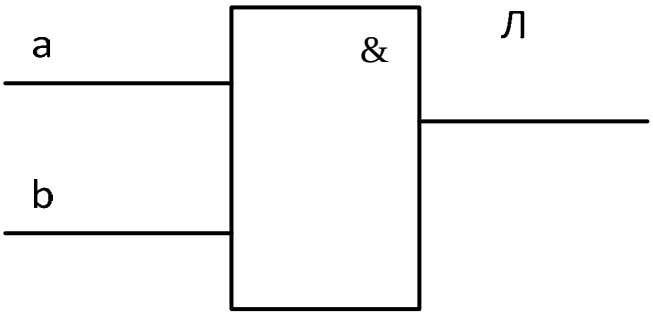


Рисунок 1.2 – Обозначение бесконтактного элемента

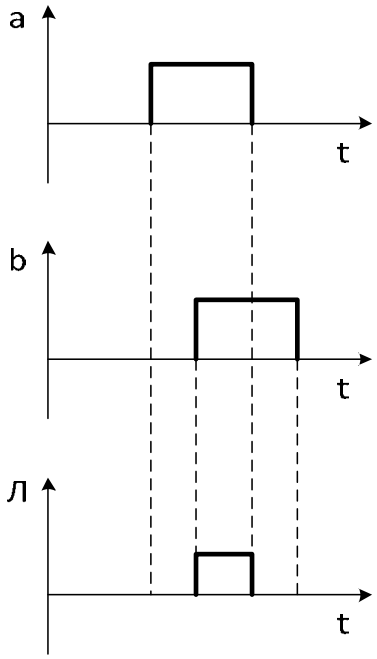


Рисунок 1.3 – Временные диаграммы

Применяются и другие обозначения операции логического умножения:

$$a \cdot b = a \wedge b = a \,\&\, b.$$

Логическое сложение.

Результат логического сложения (дизъюнкции, операции «ИЛИ») X = a + b легко установить из анализа схемы с параллельным соединением контактов (рисунок 1.4). Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.2). Обозначение бесконтактного элемента и временные диаграммы приведены на рисунках 1.5 и 1.6.

Очевидно, что катушка реле X получит питание, если замкнуть контакты или a, или b, или оба вместе.

Вместо знака «+» иногда употребляют «∨»: a + b = a ∨ b.

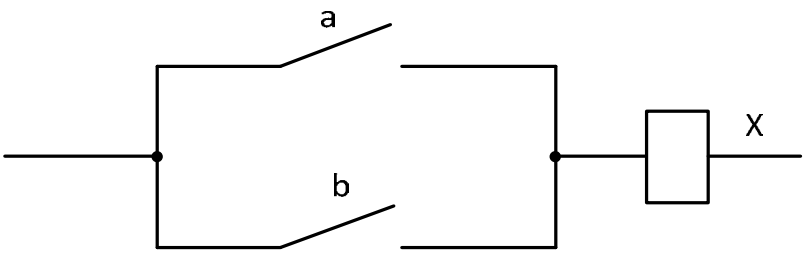


Рисунок 1.4 – Релейно-контактный вариант реализации

Таблица 1.2

a	b	X
0	0	0
0	1	1
1	0	1
1	1	1

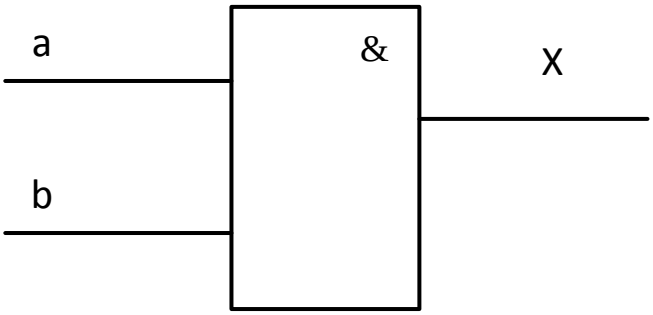


Рисунок 1.5 – Обозначение бесконтактного элемента

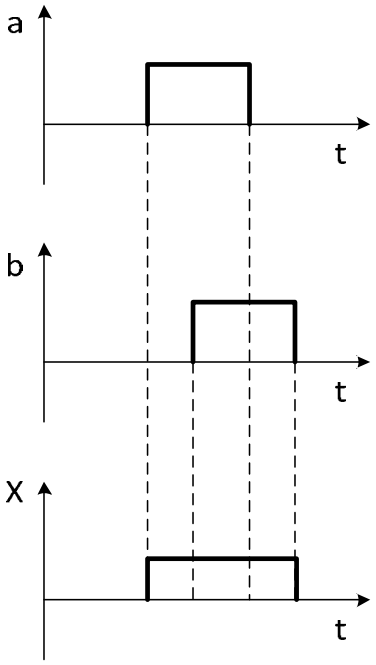


Рисунок 1.6 – Временные диаграммы

Логическое отрицание.

Логическое отрицание (инверсия, операция «НЕ») Л = /a означает, что значение логической функции Л противоположно или неравносильно значению переменной a. В данном примере лампа горит (Л = 1), если контакт a разомкнут (a = 0), и лампа гаснет (Л = 0), если контакт замкнуть (a = 1) (см. рисунок 1.7).

Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.3). Обозначение бесконтактного элемента и временные диаграммы приведены на рисунках 1.8 и 1.9.

Логическое сложение по модулю 2. Функция «ИСКЛЮЧАЮЩЕЕ ИЛИ».

Релейная схема, реализующая функцию «ИСКЛЮЧАЮЩЕЕ ИЛИ», представлена на рисунке 1.10.

Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.4). Обозначение бесконтактного элемента и временные диаграммы приведены на рисунках 1.11 и 1.12.

Обозначение операции:

$$Л = a \cdot \bar{b} + \bar{a} \cdot b.$$

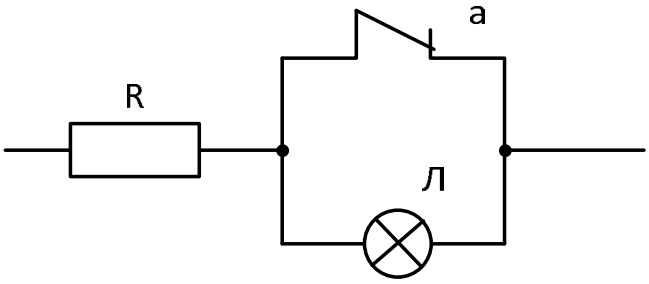


Рисунок 1.7 – Релейно-контактный вариант реализации

Таблица 1.3

a	Л
0	1
1	0

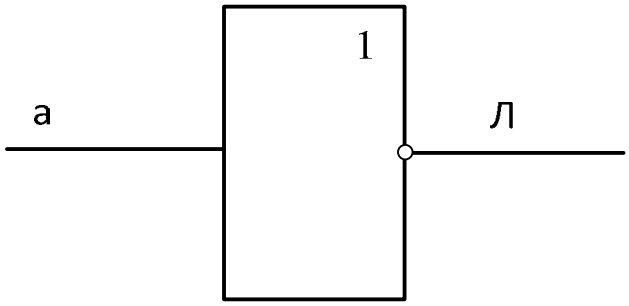


Рисунок 1.8 – Обозначение бесконтактного элемента

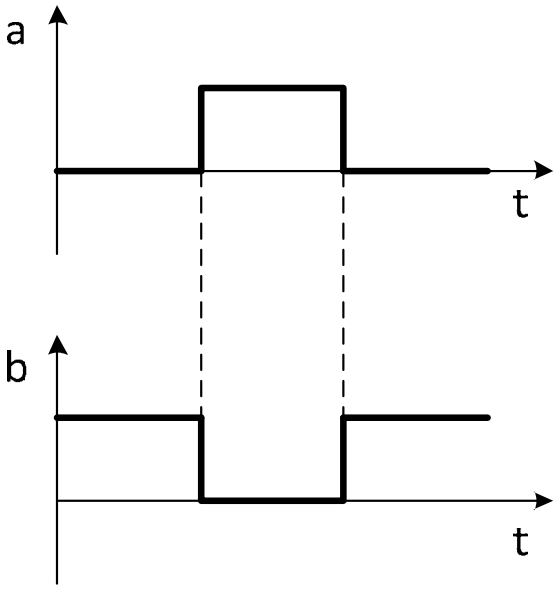


Рисунок 1.9 – Временные диаграммы

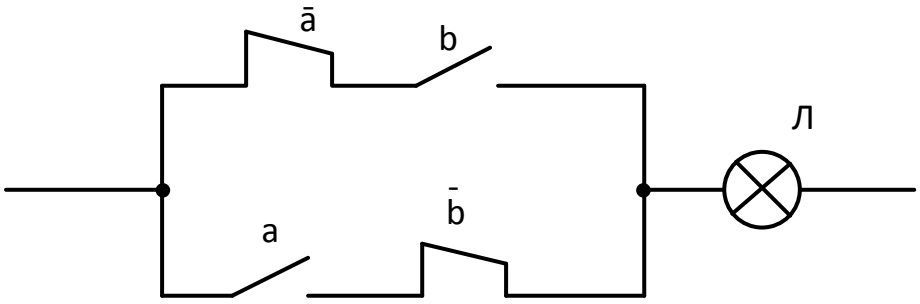


Рисунок 1.10 – Релейно-контактный вариант реализации

Инверсия конъюкции. Функция «И – НЕ».

Релейная схема, реализующая функцию «И – НЕ», представлена на рисунке 1.13.

Таблица 1.4

a	b	Л
0	0	0
0	1	1
1	0	1
1	1	0

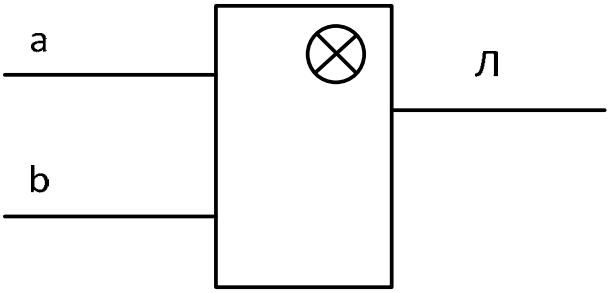


Рисунок 1.11 – Обозначение бесконтактного элемента

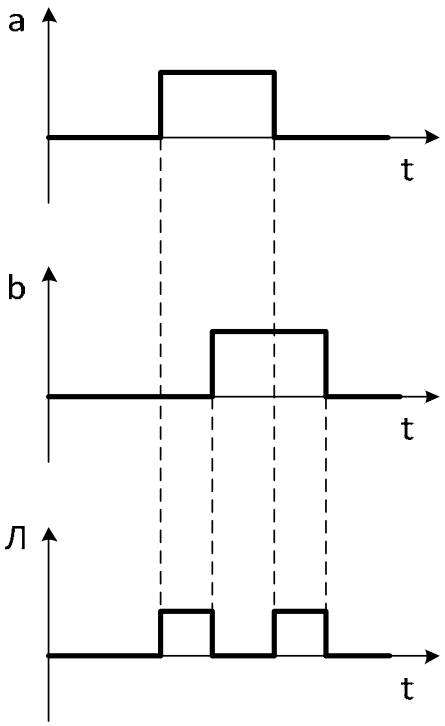


Рисунок 1.12 – Временные диаграммы

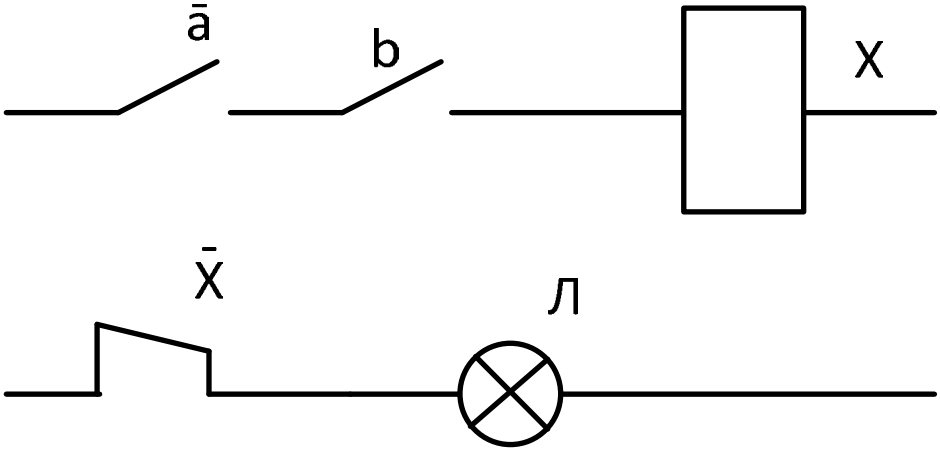


Рисунок 1.13 – Релейно-контактный вариант реализации

Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.5). Обозначение бесконтактного элемента и временные диаграммы приведены на рисунках 1.14 и 1.15.

Обозначение операции:

$$Л = \bar{a} \cdot \bar{b}.$$

Инверсия дизъюнкции. Функция «ИЛИ – НЕ».

Релейная схема, реализующая функцию «И – НЕ», представлена на рисунке 1.16.

Таблица 1.5

a	b	Л
0	0	1
0	1	1
1	0	1
1	1	0

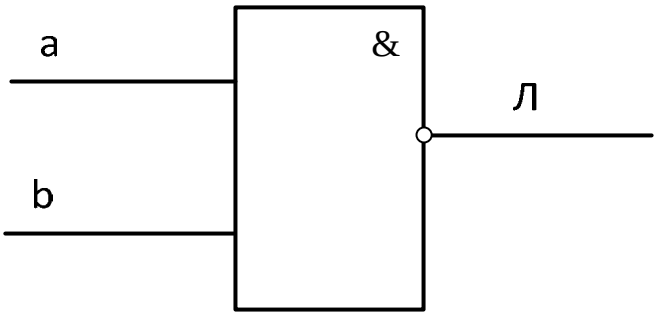


Рисунок 1.14 – Обозначение бесконтактного элемента

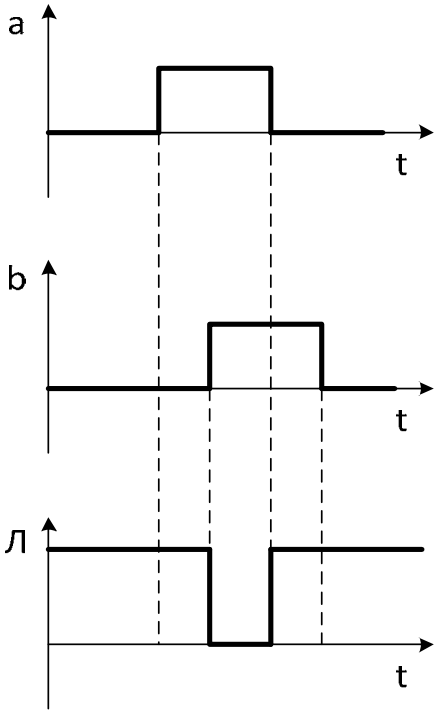


Рисунок 1.15 – Временные диаграммы

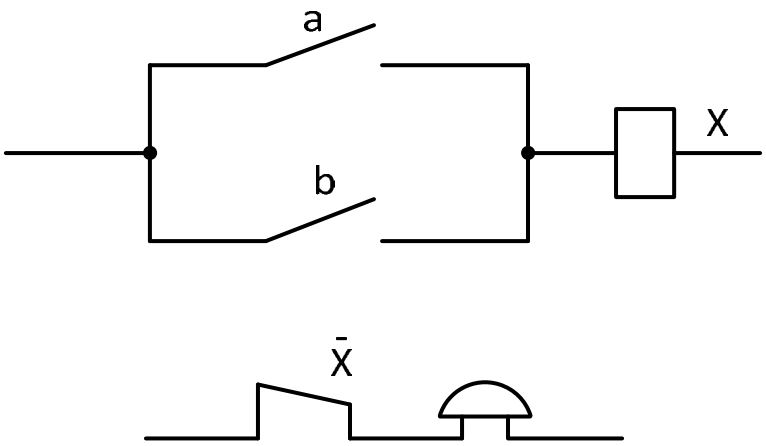


Рисунок 1.16 – Релейно-контактный вариант реализации

Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний (таблица 1.6). Обозначение бесконтактного элемента и временные диаграммы приведены на рисунках 1.17 и 1.18.

Обозначение операции:

$$З = \bar{a} + \bar{b}.$$

Указанные логические операции справедливы и для большего числа переменных. Возрастет при этом лишь количество возможных комбинаций, т. е. число строк в таблице состояний.

Таблица 1.6

a	b	З
0	0	0
0	1	0
1	0	0
1	1	0

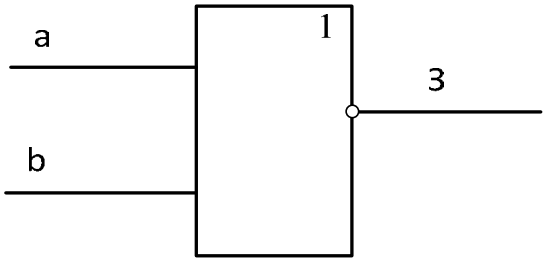


Рисунок 1.17 – Обозначение бесконтактного элемента

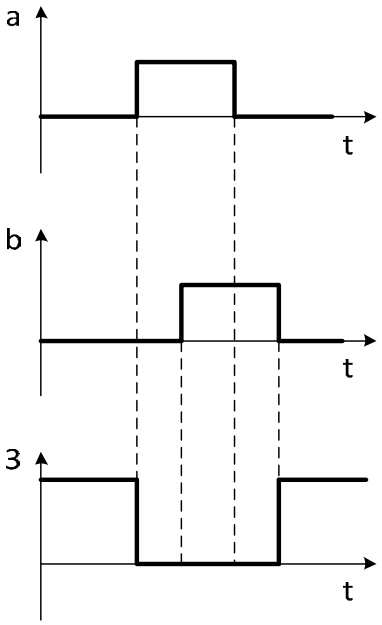


Рисунок 1.18 – Временные диаграммы

Знак «=», который в обычной алгебре является знаком равенства, в данном применении выражает равносильность связываемых логических операций, т. к. сами функции лишены количественной меры и могут принимать лишь два качественных состояния: 0 или 1.

Как и в обычной алгебре, в алгебре логики действуют законы:

– переместительный (коммутативный)

а) относительно логического умножения: a b = b a;

б) относительно логического сложения: a + b = b + a;

– сочетательный (ассоциативный)

а) относительно логического умножения: (a b)c = (a c)b;

б) относительно логического сложения: (a + b) + c = a + (b + c);

– распределительный (дистрибутивный)

а) относительно логического умножения: (a + b)c = a c + b c;

б) относительно логического сложения: a b + c = (a + c)(b + c).

Пример проектирования СЛУ

Система логического управления содержит три приёмных элемента A, B, C и два исполнительных элемента X, Y.

Алгоритм работы системы

Для элемента X (в данном примере – маломощный электродвигатель постоянного тока, включаемый пускателем КМ):

1) элемент X срабатывает, если срабатывают A,B, но не срабатывает C;

2) элемент X срабатывает, если срабатывают A,C, но не срабатывает B;

3) элемент X срабатывает, если срабатывает A, но не срабатывают B и C.
Для элемента Y (в данном примере – сигнальная лампа HL):
1) Y срабатывает, если срабатывает A, но не срабатывают B и C;
2) Y срабатывает, если срабатывает B, но не срабатывают A и C.

Решение

Составляем основной элемент синтеза – таблицу состояния.

В таблице 1.7 рассматриваем все возможные комбинации состояний приемных элементов. Так как приемных элементов три, то возможное число комбинаций состояния равно восьми, т. е. имеем восемь строк в таблице состояний. Состояние исполнительных элементов записываем в соответствии с алгоритмом: если элемент срабатывает – ставится 1, в противном случае – 0.

Таблица 1.7 – Состояние входов и выходов

Номер состояния	A	B	C	X	Y
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	1
4	0	1	1	0	0
5	1	0	0	1	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	0	0

Перейдем к составлению логической функции. Для этого составим частные условия срабатывания для элемента X:

$$X_1 = a \cdot \overline{b} \cdot \overline{c};$$
$$X_2 = a \cdot \overline{b} \cdot c;$$
$$X_3 = a \cdot b \cdot \overline{c}.$$

Общие условия срабатывания запишем как дизъюнкцию частных условий срабатывания. Это означает, что элемент X сработает, если будет выполнено или одно частное условие срабатывания, или все частные условия, или их комбинация.

$$X = X_1 + X_2 + X_3 = a \cdot \overline{b} \cdot \overline{c} + a \cdot \overline{b} \cdot c + a \cdot b \cdot \overline{c} = a \cdot \overline{b}\left(\underbrace{c + \overline{c}}_{1}\right) + a \cdot b \cdot \overline{c} =$$
$$= a \cdot \overline{b} + a \cdot b \cdot \overline{c} = a\left(\overline{b} + b \cdot \overline{c}\right) = a\left(\overline{b} + \overline{c}\right).$$

Аналогично составляем логическую функцию для элемента Y.

$$Y_1 = \bar{a} \cdot b \cdot \bar{c};$$

$$Y_2 = a \cdot \bar{b} \cdot \bar{c};$$

$$Y = Y_1 + Y_2 = \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} = \bar{c}\left(\bar{a} \cdot b + a \cdot \bar{b}\right).$$

Релейно-контактные варианты проектируемой системы логического управления имеют следующий вид (рисунок 1.19).

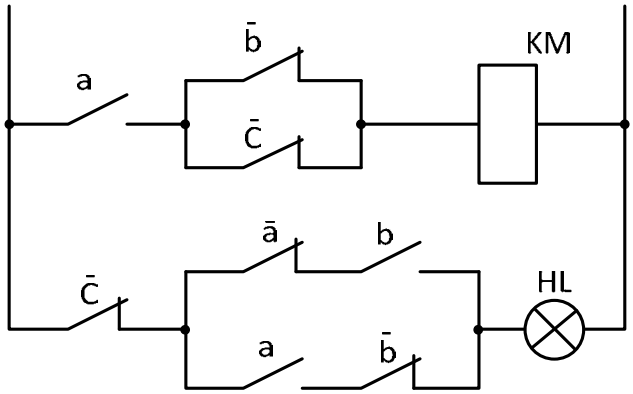


Рисунок 1.19 – Релейно-контакторная схема

Основные сведения по 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++).

Для создания проекта необходимо запустить программную среду CoDeSys. В окне Target Settings напротив строки Configuration выбрать тип логического контроллера PLC и нажать OK. В появившемся окне New POU (рисунок 1.20) выбрать тип POU Программа (Program) и язык, на котором будет осуществляться написание программы LD (релейные диаграммы). Имя программы можно оставить без изменения. Подтвердить выбор нажатием на кнопку OK.

После выполнения всех вышеописанных действий откроется главное окно (рисунок 1.21) среды CoDeSys.

Оно состоит из следующих элементов (в окне они расположены сверху вниз):

– меню;

– панель инструментов. На ней находятся кнопки для быстрого вызова команд меню;

– организатор объектов, имеющий вкладки POU, Data types, Visualizations и Resources;

– разделитель Организатора объектов и рабочей области CoDeSys;

– рабочая область, в которой находится редактор;
– окно сообщений;
– строка статуса, содержащая информацию о текущем состоянии проекта.

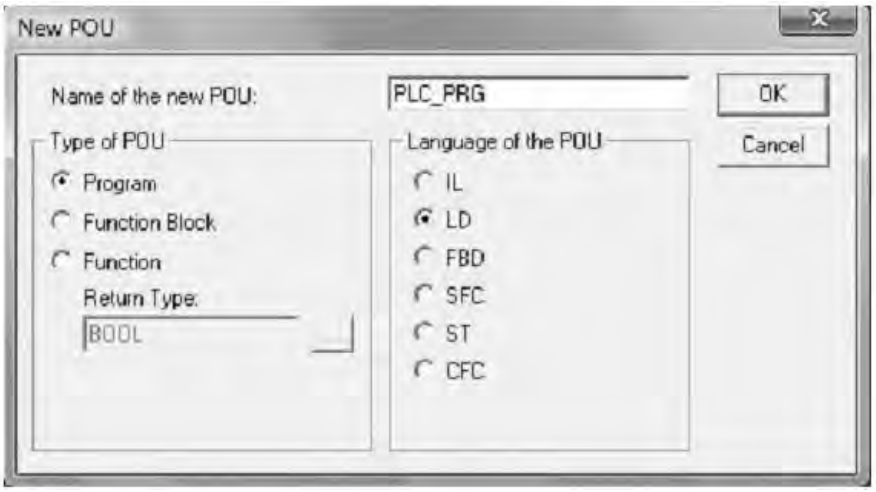


Рисунок 1.20 – Выбор языка программирования и задание имени программы

Рисунок 1.21 – Главное окно среды CoDeSys

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

Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню. Вызванная с помощью кнопки на панели инструментов команда автоматически выполнится в активном окне, как только кнопка будет отпущена.

Если поместить указатель мышки на кнопку панели инструментов, то через небольшой промежуток времени можно увидеть название этой кнопки в подсказке.

Кнопки на панели инструментов различны для разных редакторов CoDeSys.

Получить информацию относительно назначения этих кнопок можно в описании редакторов.

Организатор объектов всегда находится в левой части главного окна CoDeSys. В нижней части организатора объектов расположены вкладки POUs, Data types (типы данных), Visualizations (визуализации) и Resources (ресурсы). Переключаться между соответствующими объектами можно с помощью мышки, нажимая на нужную вкладку.

Язык LD – графический язык, в котором программа выглядит, как релейная схема в стандарте промышленной автоматизации. Две вертикальные линии слева и справа образуют линии питания. Между ними располагаются контактные цепи в виде горизонтальных линий по аналогии с обычными электрическими цепями релейной автоматики (по общему виду программы и дано название языка программирования). Слева по линии располагаются контакты (соответствуют входным переменным логического типа и дискретным входам). Справа – катушки реле (соответствуют выходным переменным логического типа и дискретным выходам).

Каждому контакту соответствует логическая переменная. Если переменная имеет значение ИСТИНА, то контакт считается замкнутым, если ЛОЖЬ – разомкнутым.

Каждой катушке также соответствует логическая переменная. Если контактная цепь от левой линии до катушки состоит из замкнутых контактов, то реле считается включенным и соответствующей переменной присваивается значение ИСТИНА, иначе реле выключается и соответствующей переменной присваивается значение ЛОЖЬ. Если катушка инверсная (обозначается символом (/)), тогда в соответствующую логическую переменную копируется инверсное значение.

Предполагается, что контакты можно соединять в любом порядке и последовательности, определяя логику работы цепи, а катушки – только параллельно, как в релейных схемах подобных устройств.

Цепь из двух последовательно соединенных контактов соответствует логической операции «И», а цепь из двух параллельно соединенных – «ИЛИ». Операции «НЕ» соответствует нормально замкнутый контакт, которая размыкает цепь (фактически дает логический нуль) при включении (подаче логической единицы) и наоборот.

Кроме последовательного и параллельного соединения нормально разомкнутых или замкнутых контактов и катушек, язык LD позволяет:

– включать фиксируемые Set / Reset-катушки;
– осуществлять переходы по цепям;
– включать в цепи функциональные блоки;
– управлять работой блоков по входам EN;
– записывать комментарии.

Наиболее важные команды редактора расположены на панели инструментов или в контекстном меню, которое вызывается правой кнопкой мыши или сочетанием клавиш <Ctrl> + <F10>.

После открытия главного окна среды CoDeSys на экране монитора поя-

вится первая цепь проектируемой системы. Это горизонтальная линия между двумя вертикальными шинами, которую надо заполнить необходимыми элементами: контактами, функциональными блоками FB, катушками реле. Элементы схемы можно в процессе программирования перемещать по цепи с помощью перетаскивания его мышью (drag & drop) или менять их наименования.

Порядок ввода, редактирования и отладки управляющих программ на языке LD в системе CoDeSys подробно описан в Руководстве пользователя по программированию ПЛК в CoDeSys 2.3.

Порядок проведения практической работы

1 Изучить основные положения алгебры логики, основные логические элементы и их реализацию в релейной автоматике.

2 Изучить порядок работы в системе программирования CoDeSys.

3 Изучить основные правила составления управляющих программ на языке LD в системе CoDeSys.

4 Изучить порядок ввода редактирования и отладки управляющих программ на языке LD в системе CoDeSys.

5 Изучить пример разработки СЛУ на релейных элементах.

6 Составить управляющую программу, реализующую СЛУ.

7 Проверить работу управляющей программы в режиме эмуляции.

8 Записать программу в память контроллера и проверить ее выполнение.

Содержание отчета

Отчет по работе должен содержать следующее.

1 Титульный лист установленного образца.

2 Цель работы.

3 Список инструкций языка программирования LD, реализующие логические функции.

4 Листинг программы.

5 Вывод по практической работе.

2 Практическая работа № 2. Разработка программ управления, реализующих временные функции

Цель практической работы: получение теоретических сведений о реализации временных функций и практических навыков проектирования систем управления с временными функциями.

Краткие теоретические сведения

Существует набор типовых функциональных блоков, реализующий функции, часто используемые в программировании ПЛК. Это счетчики, таймеры,

триггеры, детекторы фронтов и т. д. В стандартной библиотеке подпрограмм Standart.lib, входящей в комплект CoDeSys, к таким блокам относятся:

– таймеры (TON, TOF, TP);
– счетчики (CTU, CTD, CTUD);
– триггеры (RS, SR);
– детекторы фронтов (R_TRIG, F_TRIG).

Эти функциональные блоки могут использоваться в программах на языке LD совместно с типовыми элементами этого языка.

Триггеры.

RS- и SR-триггеры отличаются лишь реакцией сигналов на оба входа: SET и RESET. Напоминаем, что для классического RS-триггера такое состояние входов является запрещенным, т. к. приводит к неоднозначности выходного сигнала.

Для исследования этих триггеров достаточно набрать лишь два фрагмента, в которых участвуют указанные функциональные блоки. На рисунке 2.1 изображены фрагменты схем с RS-и SR-триггерами.

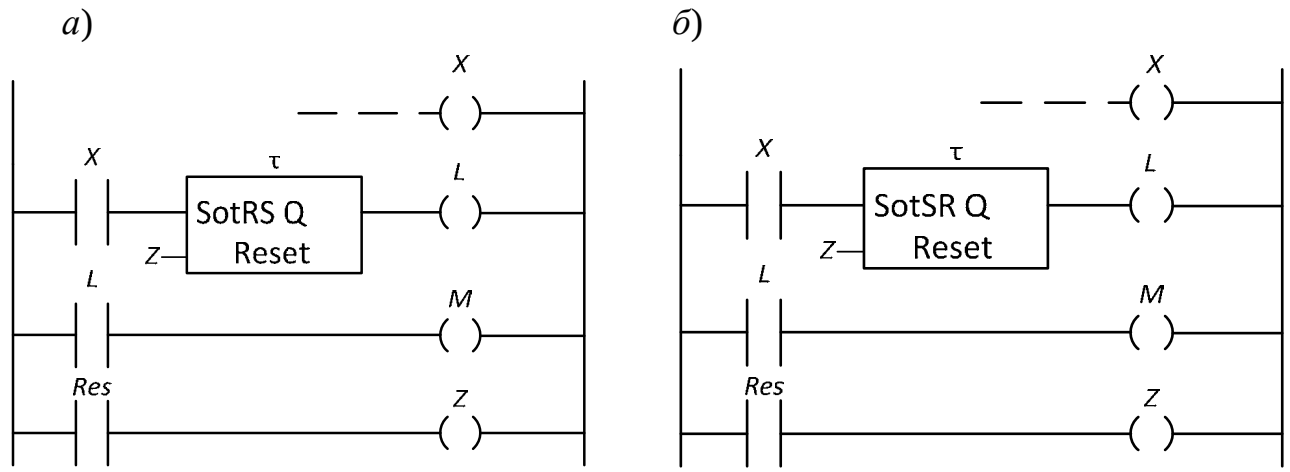


Рисунок 2.1 – Фрагменты схем с RS- и SR-триггерами

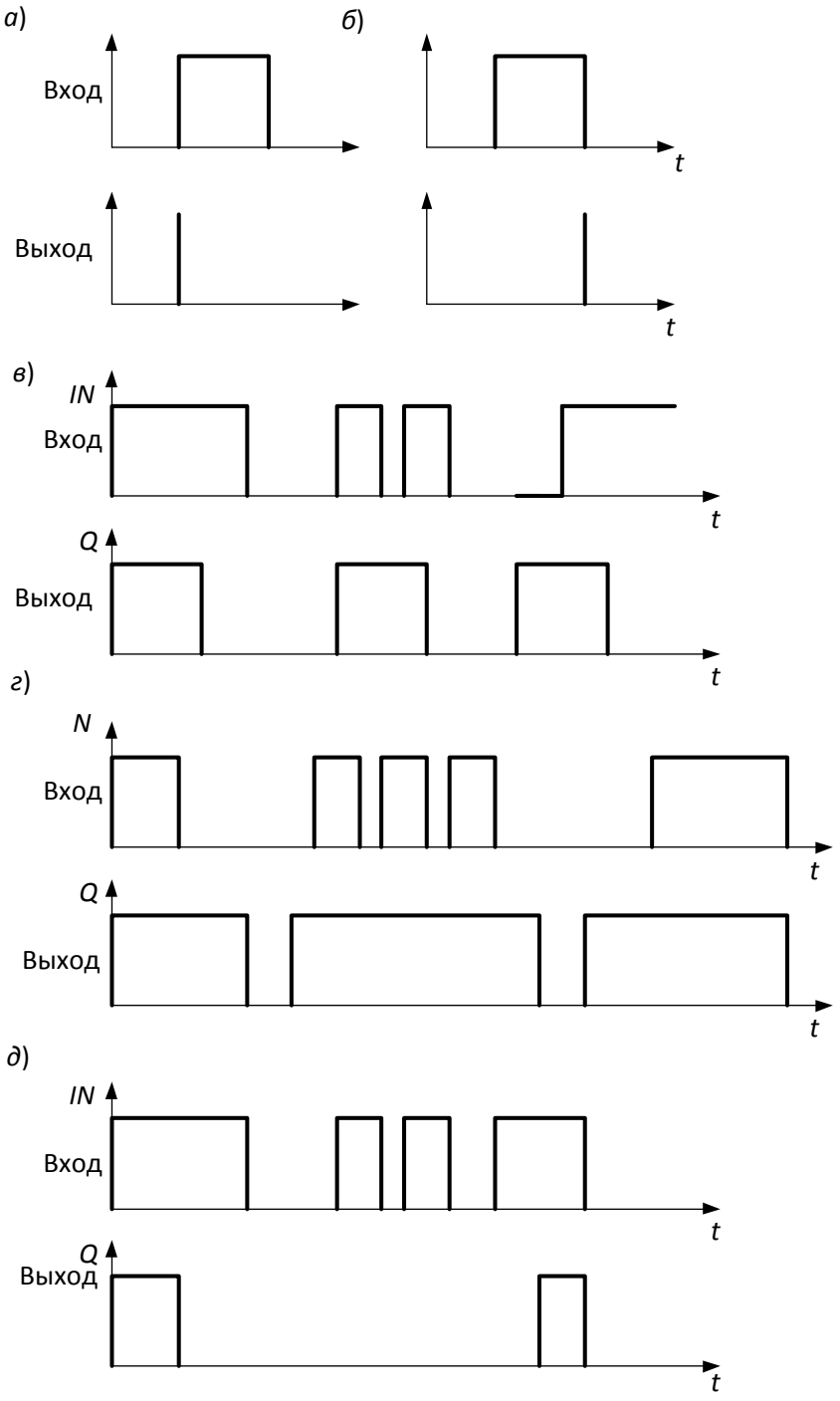
При замыкании контакта X подается сигнал на вход SET каждого из триггеров (см. рисунок 2.1), что вызывает срабатывание реле L, и через замыкающий контакт L этого реле приходит сигнал на обмотку реле M, включающего, например, электрическую машину. Последующие размыкания или замыкания не меняют состояния выходов этих триггеров, и катушки L и M остаются включенными. Для отключения реле M необходимо кратковременно нажать кнопку Res.

Если одновременно замкнутся контакты X и Res, то в RS-триггере преобладающим будет сигнал на отключение L и, соответственно, M, а в SR-триггере – на включение этих реле.

Следует отметить, что входу RESET этих триггеров необходимо присвоить идентификатор (имя) того реле, которое будет обеспечивать сброс. В данном примере в третьей цепи каждого фрагмента (см. рисунок 2.1) стоит реле с именем «Z». Поэтому на входах RESET поставлен тот же символ – «Z».

Детекторы импульсов.

Функциональные блоки R_TRIG и F_TRIG являются детекторами импульсов. Первый из них генерирует одиночный импульс по переднему фронту, а другой – по заднему спаду входного сигнала. Временные диаграммы входных и выходных сигналов показаны на рисунке 2.2.



а – R_TRIG; б – F_TRIG; *в* – TP; *г* – TOF; *д* – TON

Рисунок 2.2 – Временные диаграммы детекторов импульсов и таймеров

Таймеры.

Три типа таймеров находят широкое применение:

1) TP-таймер или генератор одиночного импульса заданной длительности;
2) TOF-таймер с задержкой выключения;
3) TON-таймер с задержкой включения.

У этих таймеров есть вход IN для логических сигналов, вход PT для установки требуемых временных параметров и логический выход Q.

Работу этих таймеров поясняют временные диаграммы (см. рисунок 2.2).

Из этих диаграмм следует, что TP-таймер запускается мгновенно передним фронтом входного сигнала и в течение времени уставки действия выходного сигнала не реагирует на новые импульсы, поступающие на вход IN (см. рисунок 2.2, *в*).

TOF-таймер также срабатывает по фронту входа IN. Выход Q сбрасывается после спада входного сигнала с задержкой времени уставки от установленной по входу PT. Пауза между входными сигналами должна быть не меньше времени задержки (см. рисунок 2.2, *г*).

TON таймер срабатывает по переднему фронту входа IN, но сигнал на выходе Q появится с задержкой TB, установленной по входу PT.

Таймер не реагирует на импульсы продолжительностью менее значения TB (рисунок 2.2, *д*).

При включении любого таймера в цепь многоступенчатой схемы (рисунок 2.3) программа запрашивает имя этого функционального блока (вопросительные знаки над таймером) и временную уставку по входу PT (вопросительные знаки у входа PT).

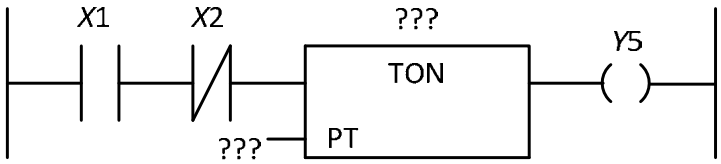


Рисунок 2.3 – Фрагмент схемы с вставленным таймером

Щелкнув 1ЛКМ по верхним ???, присваиваем имя. Например, N1. Щелкнув по ??? у хода PT, нажав, и не отпуская Shift, нажать T, затем #, отпустить Shift, нажать требуемое значение задержки (например 15), единицу времени (например S – т. е. секунды) и Enter. Этот фрагмент будет выглядеть, как показано на рисунке 2.4.

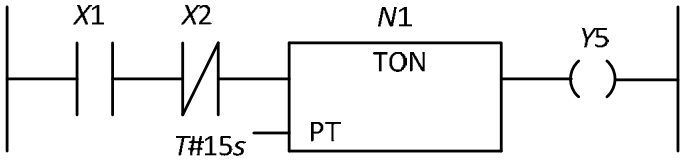


Рисунок 2.4 – Фрагмент схемы после записи уставок

Временные уставки задают в миллисекундах (mS), секундах (S), минутах (m) или часах (h). Уставка может быть дробной. К примеру, T#1.5m.

Счетчики.

CTU – инкрементный, CTD – декрементный и CTUD – инкрементный/декрементный счетчики.

Простейшая схема с CTU-счетчиком показана на рисунке 2.5.

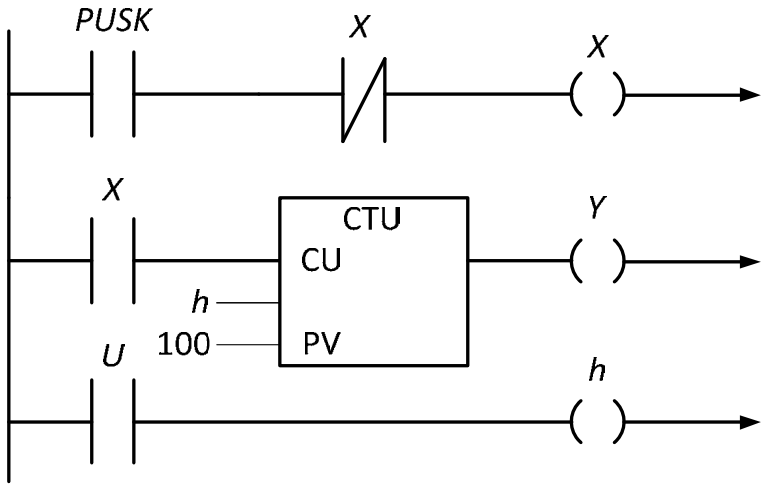


Рисунок 2.5 – Схема с CTU-счетчиком

После переноса счетчика в цепь многоступенчатой схемы появятся ??? над блоком, на входе – RESET и PV.

Знаки ??? над блоком заменяем именем. Знаки ??? перед PV запрашивают значение уставки, т. е. требуемое количество импульсов, вызывающее срабатывание счетчика, при котором выход Q перейдет в TRUE (при условии, что на входе RESET был сигнал FALSE).

Вход RESET знаками ??? запрашивает имя логического элемента, от которого должен поступить сигнал TRUE, останавливающий счет, обнуляющий выход CV и устанавливающий на выходе Q FALSE.

В схеме (см. рисунок 2.5) для счетчика присвоено имя W, для входа RESET – имя реле h и принята уставка PV=100.

Само реле h находится в третьей цепи и управляется кнопкой U.

Первая цепь содержит кнопку PUSK. По каждому фронту сигнала, поступающему на вход CU, значение выхода CV возрастает на 1, и, как только их сумма достигнет значения PV, счет останавливается.

На других языках программирования ПЛК с выхода CV можно снимать информацию о количестве поступивших импульсов для последующей обработки с помощью операторов и функций.

CTD-счетчик отличается от CTU тем, что каждый входной импульс уменьшает значение счетчика на 1. Когда счетчик достигнет нуля, выход Q устанавливается в TRUE.

CTUD-инкрементный/декрементный счетчик имеет как накопительный вход CU (как в CTU), так и вычитающий CD (как в CTD).

Информация о работе данных блоков содержится в Руководстве пользователя по программированию ПЛК в CoDeSys 2.3 и в Справочной системе

комплекса программирования CoDeSys.

Далее приведены примеры программ на языке LD, составленные с использованием этих функциональных блоков.

Пример 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 0 и уменьшает на 1 значение этой – при наличии отрицательного фронта на входе 1. Общий вид программы на языке LD представлен на рисунке 2.6. Для регистрации фронтов использованы детекторы фронтов R_TRIG и F_TRIG, для работы с целой переменной – 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.

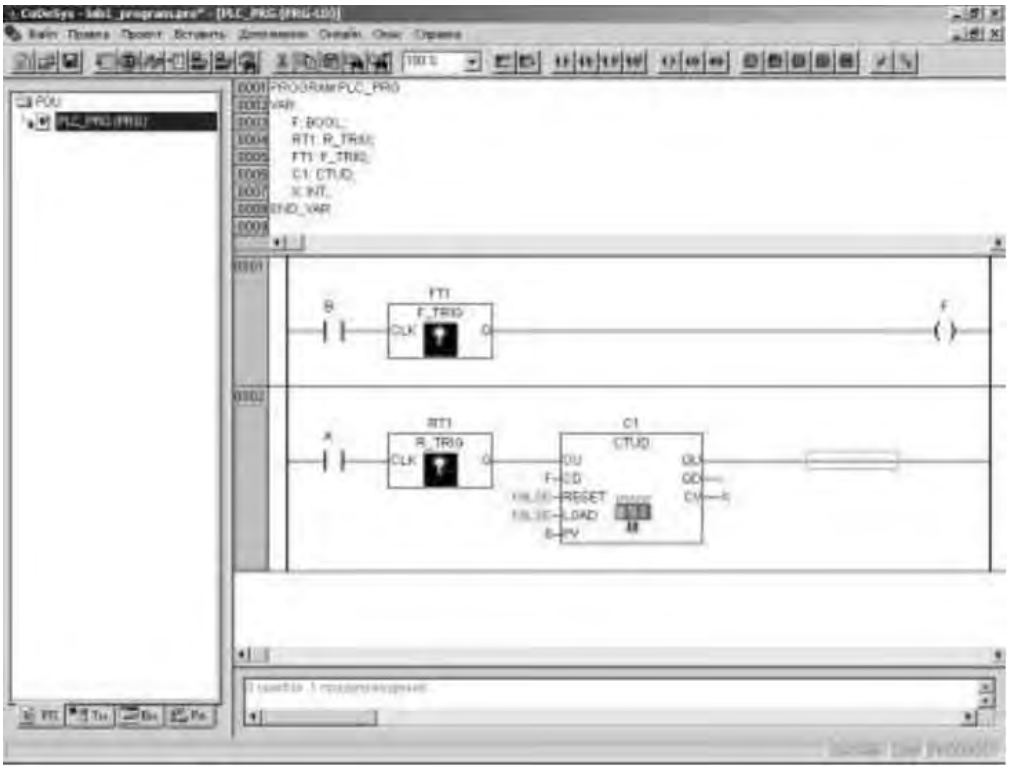


Рисунок 2.6 – Программа демонстрации работы детекторов фронтов и реверсивного счетчика

С первого дискретного входа значение сигнала подается на вход блока R_TRIG, объявленного как переменная RT1, со второго – на F_TRIG, объявленного, как переменная FT1. Выход FT1 связан с переменной F, которая

далее подана на вход CD (уменьшение на единицу) счетчика. Выход RT1 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.

Поскольку счетчик CTUD используется не полностью, при компиляции данный пример сгенерирует одно предупреждение, но, несмотря на это, пример работает нормально. Тестирование примера просто: если нажимается кнопка, подключенная к первому входу, переменная X увеличивается на единицу, если нажимается и отпускается кнопка, подключенная ко второму входу, 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 2.7.

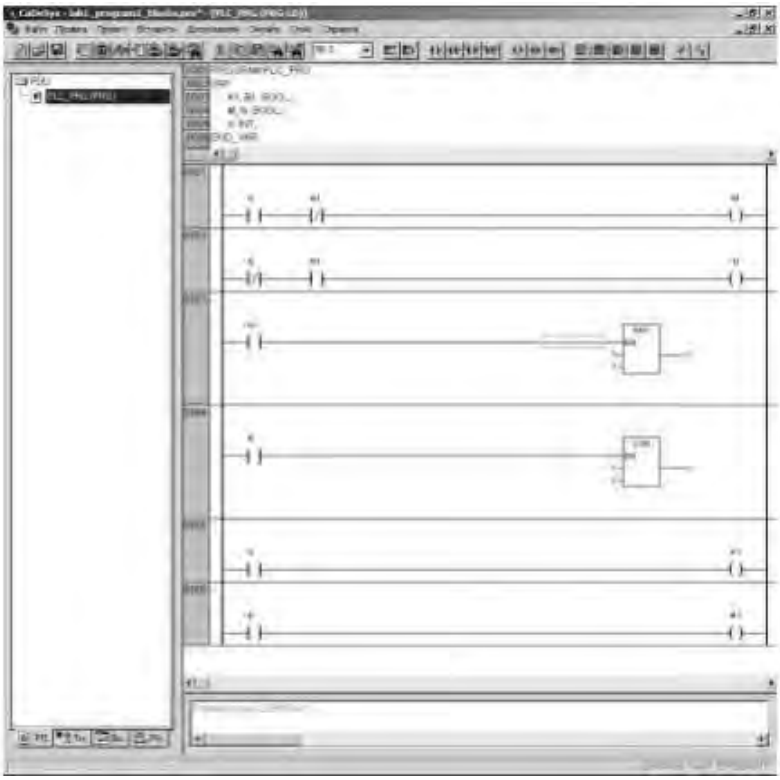


Рисунок 2.7 – Программа демонстрации работы детекторов фронтов и реверсивного счетчика

Следует обратить внимание на арифметические операции. 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 с входом EN.

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 цикле программы как значения, сохраненные в прошлом цикле; так происходит каждый цикл.

Первые две строки программы представляют собой именно детектирование сигнала, единовременную проверку его значения в прошлом и настоящем шагах.

Пример 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 включатель 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 2.8.

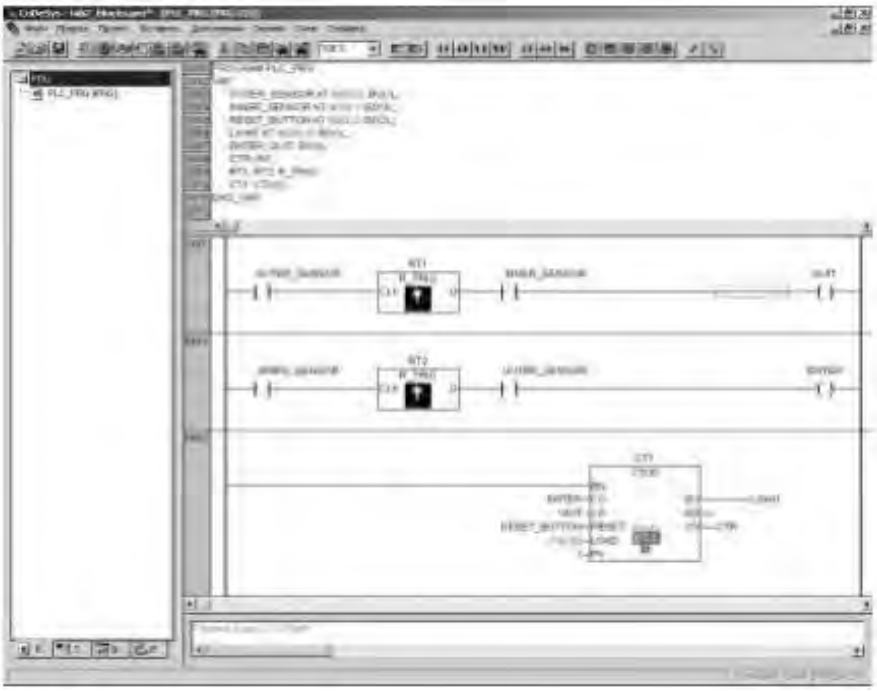


Рисунок 2.8 – Решение задачи об автоматическом включении света, сделанное с помощью типовых функциональных блоков

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.

Если значение счетчика больше или равно 1, следует включить свет, если нет – выключить. Предположим, что возможна ситуация, когда человек, находясь в комнате, хочет выключить свет; для этого необходимо иметь кнопку принудительного гашения света, которую следует 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 выхода из комнаты и входа в нее соответственно. LIGHT – переменная, связанная с реле включения света, CTR – переменная счетчика вошедших в комнату.

На рисунке 2.9 изображено решение той же задачи, но без применения типовых функциональных блоков.

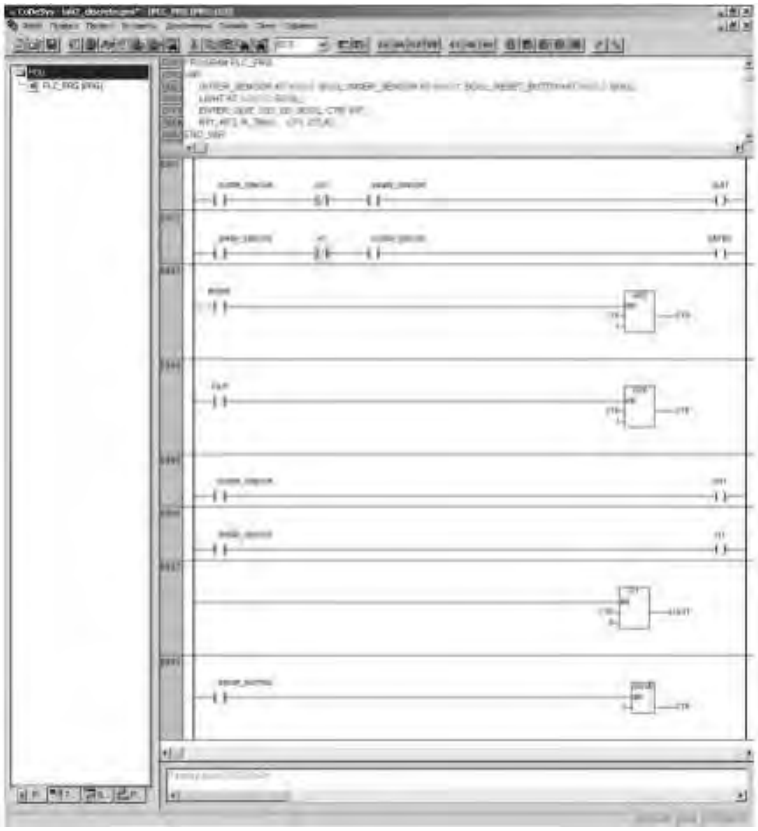


Рисунок 2.9 – Решение задачи об автоматическом включении света без типовых функциональных блоков

Переменные IS1 и OS1 хранят значения переменных INNER_SENSOR и OUTER_SENSOR за предыдущий цикл.

Функция GT – сравнение двух чисел на входе, и если «верхнее» больше, чем «нижнее», то 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.

Пример 3 – Программный генератор периодических импульсов.

Требуется создать программный генератор прямоугольных импульсов с постоянной скважностью и задаваемым периодом. Для построения генератора будут использоваться таймеры, работа которых описана во встроенной справочной системе CoDeSys.

Один из вариантов программы показан на рисунке 2.10. Принцип функционирования генератора приведен ниже.

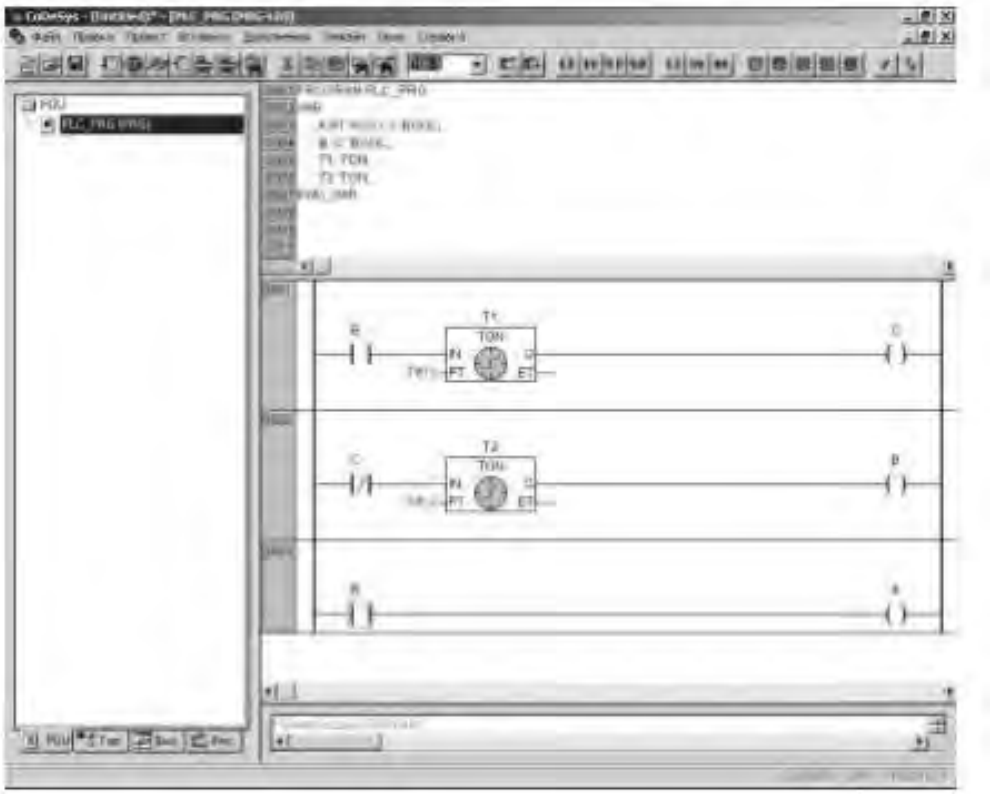


Рисунок 2.10 – Программа генератора прямоугольных импульсов на языке LD

Допустим, непосредственно после первого запуска программы переменные B и C равны логическому нулю, в результате таймер T1 не запускается, а таймер T2 запускается, т. к. 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 с) устанавливается в единицу переменная B, стоящая на выходе таймера T2, при этом запускается таймер T1. Таймер T1 через заданное время (тоже 1 с) 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, т. к. 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 С не меняет своих значений, а переменная В – периодически устанавливается и сбрасывается, т. к. установка и сброс переменной С происходят с интервалом времени в один цикл, а установка и сброс переменной В – с заданными промежутками времени.

Порядок проведения практической работы

1 Изучить работу типовых блоков, реализующих функции таймеров, счетчиков, триггеров, детекторов фронтов и т. д., входящих в библиотеку Standart.lib системы CoDeSys .

2 Изучить примеры применения типовых функциональных блоков в управляющих программах на языке LD.

3 Проверить работу программ, приведенных в примерах, в режиме эмуляции.

4 Записать программы в память контроллера и проверить их выполнение.

5 Составить отчет по работе.

При выполнении работы необходимо дополнительно использовать Руководство пользователя по программированию ПЛК в CoDeSys 2.3 и Справочную систему комплекса программирования CoDeSys.

Содержание отчета

Отчет должен содержать следующее.

1 Титульный лист установленного образца.

2 Цель работы.

3 Графическое изображение типовых функциональных блоков: таймеров, счетчиков, триггеров, детекторов фронтов и временные диаграммы, поясняющие их работу.

4 Блок-схему алгоритма управления.

5 Список управляющих кнопок и исполнительных устройств с указанием адресов входов-выходов, к которым они подключены.

6 Листинги программ.

7 Вывод по работе.

3 Практическая работа № 3. Разработка программ управления, реализующих функции регулирования

Цель практической работы: получение теоретических сведений о системах регулирования и практических навыков проектирования систем регулирования.

Краткие теоретические сведения

ПИД-регулирование является наиболее точным и эффективным методом поддержания контролируемой величины на заданном уровне. На рисунке 3.1 приведена функциональная схема ПИД-регулятора. Основное назначение регулятора – формирование управляющего сигнала Y, задающего выходную мощность исполнительного механизма (ИМ) и направленного на уменьшение рассогласования E или отклонения текущего значения регулируемой величины T от величины уставки $T_{уст}$.

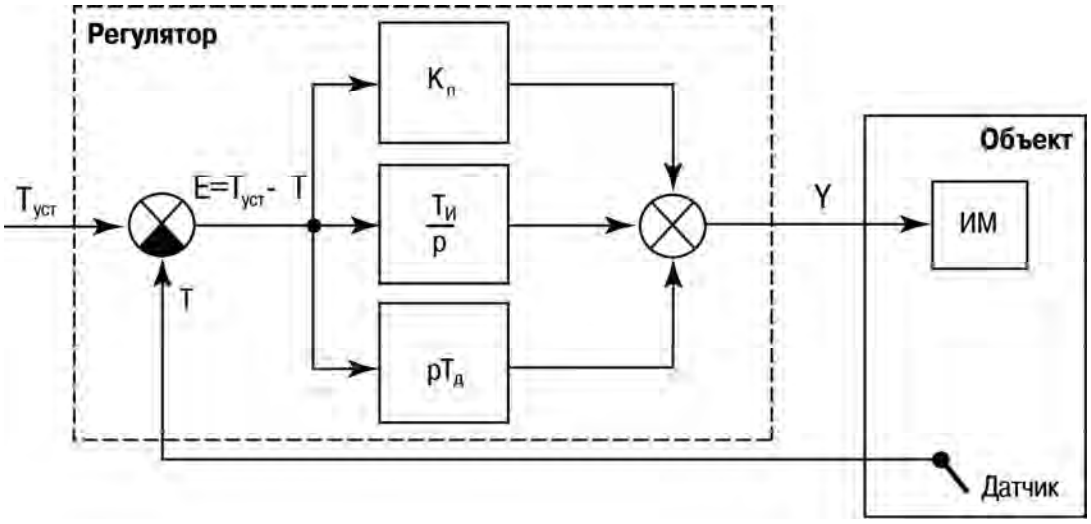


Рисунок 3.1 – Схема ПИД-регулятора

ПИД-регулятор состоит из трех основных частей: пропорциональной, интегральной и дифференциальной. Пропорциональная составляющая зависит от рассогласования Ei и отвечает за реакцию на мгновенную ошибку регулирования.

Интегральная составляющая содержит в себе накопленную ошибку регулирования и позволяет добиться максимальной скорости достижения заданного значения.

Дифференциальная составляющая зависит от скорости изменения рассогласования и позволяет улучшить качество переходного процесса.

В системе программирования CoDeSys в библиотеке UTIL.lib реализован стандартный блок ПИД-регулятора, который показан на рисунке 3.2.

Функциональный блок реализует ПИД-закон регулирования:

$$Y = Y\_OFFSET + KP\left(e(t) + \frac{1}{TN}\int_0^{TN} e(t) + TV\frac{de(t)}{dt}\right).$$

где Y_OFFSET – стационарное значение;
KP – коэффициент передачи;
TN – постоянная интегрирования (ms);
TV – постоянная дифференцирования (ms);
e(t) – сигнал ошибки (SET_POINT-ACTUAL).

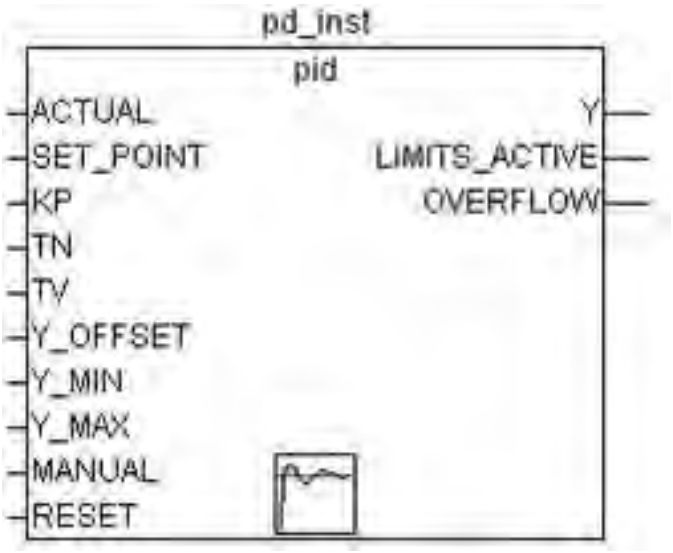


Рисунок 3.2 – ПИД-регулятор в CoDeSys

Входы ACTUAL, SET_POINT, KP, Y_OFFSET, Y_MIN Y_MAX типа REAL.
Входы TN и TV типа DWORD, RESET и MANUAL типа BOOL.
Выходы Y – REAL, LIMITS_ACTVE и OVERFLOW типа BOOL.

Значение выхода Y ограничено Y_MIN и Y_MAX. При достижении Y границ ограничения, выход LIMITS_ACTVE, (BOOL) принимает значение TRUE. Если ограничение выхода не требуется, Y_MIN и Y_MAX должны быть равны 0.

Неправильная настройка регулятора может вызвать неограниченный рост интегральной составляющей. Для обнаружения такой ситуации предназначен выход OVERFLOW. При переполнении он принимает значение TRUE, одновременно останавливается работа регулятора. Для его включения необходимо использовать рестарт.

Пример реализации двухпозиционного и ПИД-регулятора. Список переменных (рисунок 3.3) включает в себя параметры двух регуляторов и входы-выходы системы управления.

Программа управления на языке FBD, состоящая из двух функциональных блоков, представлена на рисунке 3.4.

В алгоритме ПИД-регулятора реализовано ограничение по мощности регулирования (0…100 %).

Двухпозиционный регулятор реализован в виде функционального блока, который может быть реализован на различных языках.

Область объявления переменных для функционального блока представлена на рисунке 3.5, ST – на рисунках 3.5 и 3.6.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003 (*параметры ПИД-регулятора*)
0004     T:REAL:=20;(*ustavka*)
0005     ti:REAL:=1;(*integr.postyannaya*)
0006     td:REAL:=0;(*dif.postyannaya*)
0007     Xp:REAL:=10;(*polosa proporcionalnosty*)
0008     C1:REAL:=0;(*nizhnaya ustavka komparatora*)
0009     C2:REAL:=20;(*verhnaya ustavka komparatora*)
0010
0011 (*параметры для ПИД-регулятора*)
0012     a12:REAL:=0.1;(*zona nechustvitelnosty*)
0013     a13:REAL:=100;(*ogranicheniy vyh. moshnosty, %*)
0014     a14:BOOL;(*type isp. mechanizma: 0 - nagrevatel; 1 - ohladitel;*)
0015
0016 (*параметры для двухпозиционного регулятора*)
0017     a21:WORD:=0;(*type logiki: 0- vykl; 1 - prym.gysterezis;2 - obr.gysterezis; 3- P-logika; 4 - U-logika*)
0018     a29:BOOL;(*sostoyaniye VU2 pri neispravnosty: 0 - otkl;1 - vkl 100%*)
0019
0020 (*рабочие параметры*)
0021     dat:REAL;(*signal datchika temperature*)
0022     vu2:BOOL;(*signal vu2*)
0023     pvu1:REAL;(*vyh. moshnost signal vu1*)
0024     reg1: reg_2pos_cfc;
0025 END_VAR
0026 VAR
0027     pidreg: PID;
0028 END_VAR
0029 VAR
0030     reg2: reg_2pos;
0031 END_VAR
```

Рисунок 3.3 – Область объявления переменных проекта

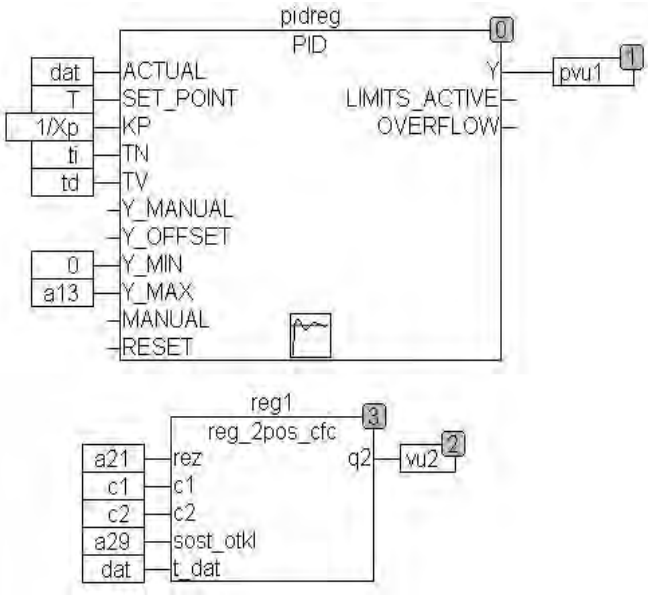


Рисунок 3.4 – Программа регулятора в CoDeSys

```
0001 FUNCTION_BLOCK reg_2pos
0002 VAR_INPUT
0003     rez:INT;(*rezhim regulyatora*)
0004     c1,c2:REAL;(*predely komparatora, c2-verhniy, c1 - nizhniy*)
0005     sost_otkl:BOOL;(*состояние при отключении*)
0006     t_dat:REAL;(*сигнал с датчика*)
0007 END_VAR
0008 VAR_OUTPUT
0009     q2:BOOL;(*сигнал 2 выход*)
0010 END_VAR
0011 VAR
0012
0013 END_VAR
0014
```

Рисунок 3.5 – Область объявления переменных двухпозиционного регулятора

```
0001 CASE rez OF
0002 (*регулятор выключен*)
0003 0:IF sost_otkl =0 THEN q2:=0;
0004 ELSE q2:=1;
0005 END_IF;
0006
0007 (*прямой гистерезис*)
0008 1: IF t_dat<c1 THEN q2:=1;
0009 END_IF;
0010 IF t_dat>c2 THEN q2:=0;
0011 END_IF;
0012
0013 (*обратный гистерезис*)
0014 2: IF t_dat>c2 THEN q2:=1;
0015 END_IF;
0016 IF t_dat<c1 THEN q2:=0;
0017 END_IF;
0018
0019 (*П-логика*)
0020 3: IF (t_dat>c1) AND (t_dat<c2) THEN q2:=1;
0021 ELSE q2:=0;
0022 END_IF;
0023
0024 (*U-логика*)
0025 4:IF (t_dat>c1) AND (t_dat<c2) THEN q2:=0;
0026 ELSE q2:=1;
0027 END_IF;
0028 END_CASE;
```

Рисунок 3.6 – Подпрограмма двухпозиционного регулятора на языке ST

Программа на языке ST и CFC представлена на рисунках 3.6 и 3.7 соответственно.

Порядок проведения практической работы

1 Изучить работу типовых блоков, реализующих функции регулирования, входящих в библиотеку Util.lib системы CoDeSys .

2 Изучить примеры применения типовых функциональных блоков в управляющих программах.

3 Проверить работу программ, приведенных в примерах, в режиме эмуляции.

4 Записать программы в память контроллера и проверить их выполнение.

5 Составить отчет по работе.

При выполнении работы необходимо дополнительно использовать Руководство пользователя по программированию ПЛК в CoDeSys 2.3 и Справочную систему комплекса программирования CoDeSys.

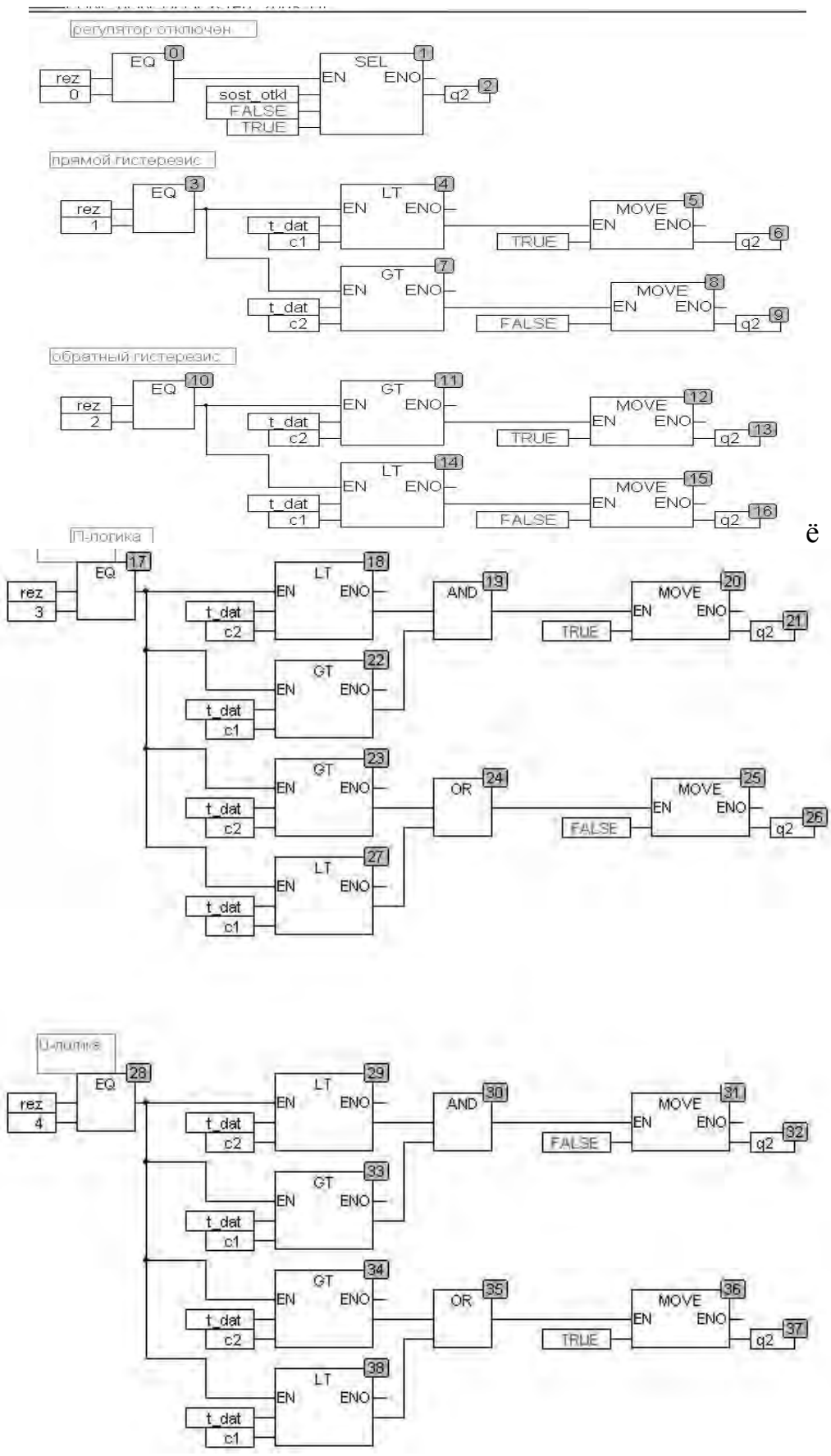


Рисунок 3.7 – Подпрограмма двухпозиционного регулятора на языке CFC

Содержание отчета

Отчет должен содержать следующее.

1 Титульный лист установленного образца.

2 Цель работы.

3 Графическое изображение типовых функциональных блоков, реализующих функции регулирования.

4 Блок схему алгоритма регулирования.

5 Листинги программ.

6 Вывод по работе.

4 Практическая работа № 4. Разработка программ управления, реализующих сетевые функции

Цель практической работы: получение теоретических сведений о распределенных системах управления и практических навыков проектирования распределенных систем управления

Краткие теоретические сведения

Часто на практике в системе управления необходимо организовать обмен информацией между различным оборудованием. К такому оборудованию относятся программируемые контроллеры, преобразователи частоты, локальные регуляторы, устройства удаленного ввода-вывода и т. д.

Для организации связи между подобными устройствами наиболее рациональным решением является создание промышленной сети на основе последовательного интерфейса RS-485 и коммуникационного протокола Modbus.

Интерфейс RS-485 подразумевает использование исключительно топологии «шина». Число устройств на шине не должно превышать 32-х. На практике это значение может быть увеличено до 247 устройств при использовании повторителей интерфейса (после каждых 32-х устройств), но нужно учитывать, что так как опрос всех устройств происходит последовательно, время одного полного цикла опроса может значительно увеличиться.

На первом и последнем устройстве шины должен быть установлен согласующий резистор (терминатор) с сопротивлением 120 Ом.

Для линий связи RS-485 необходимо использовать экранированный кабель с витой парой, предназначенный для промышленного интерфейса RS-485 с волновым сопротивлением 120 Ом (например, КИПЭВ). Экран кабеля должен быть соединен с функциональной землей только в одной точке.

Протокол Modbus предусматривает наличие в сети только одного master-устройства, которое отсылает и принимает запросы подчиненных slave-устройств. Slave-устройства не могут являться инициаторами обмена. При этом, как правило, у всех подчиненных устройств есть сетевой адрес, по которому к

нему обращается мастер. При формировании команды мастер указывает, какому устройству предназначена команда, и отсылает команду в сеть. Если в сети имеется несколько устройств, то команда от мастера принимается всеми устройствами, но ответ формирует только то устройство, которому данная команда предназначена. В одной сети может быть только один мастер и несколько подчиненных устройств, при этом адреса подчиненных устройств должны быть различны.

Запрос master-устройства к slave-устройству содержит:

– адрес slave-устройства;

– код функции, применяемый к slave-устройству;

– данные – адрес первого регистра и их количество (в случае записи – также их значения);

– контрольную сумму.

Ответ slave-устройства имеет схожую структуру.

При запросе master-устройство обращается к одной из областей памяти slave-устройства с помощью определенной функции. Область памяти характеризуется типом хранящихся в ней значений (биты/регистры) и типом доступа (только чтение/чтение и запись).

Стандарт Modbus определяет четыре области памяти (таблица 4.1).

Таблица 4.1 – Области данных протокола Modbus

Область данных	Обозначение	Тип данных	Тип доступа
Coils (Регистры флагов)	0x	BOOL	Чтение/запись
Discrete Inputs (Дискретные входы)	1x	BOOL	Только чтение
Input Registers (Регистры ввода)	3x	WORD	Только чтение
Holding Registers (Регистры хранения)	4x	WORD	Чтение/запись

Каждая область памяти состоит из определенного (зависящего от конкретного устройства) количества ячеек. Каждая ячейка имеет уникальный адрес. Для конфигурируемых устройств (таких как ТРМ, ПЧВ и т. д.) производитель предоставляет карту регистров, в которой содержится информация о соответствии параметров устройства и их адресов.

Для программируемых устройств пользователь формирует такую карту самостоятельно с помощью среды разработки.

Существуют устройства, в которых сочетаются оба рассмотренных случая – у их карты регистров есть фиксированная часть, которую пользователь может дополнить в соответствии со своей задачей (очевидно, что адреса ячеек при этом не должны пересекаться).

В некоторых устройствах области памяти наложены друг на друга (например, 0x и 4x), т. е. пользователь сможет обращаться разными функциями к одним и тем же ячейкам памяти.

Функция определяет операцию (чтение/запись) и область памяти, с которой эта операция будет произведена. В таблице 4.2 приведен список наиболее часто используемых функций.

Таблица 4.2 – Основные функции протокола Modbus

Код функции	Имя функции	Выполняемая команда
1 (0x01)	Read Coil Status	Чтение значений из нескольких регистров флагов
2 (0x02)	Read Discrete Inputs	Чтение значений из нескольких дискретных входов
3 (0x03)	Read Holding Registers	Чтение значений из нескольких регистров хранения
4 (0x04)	Read Input Registers	Чтение значений из нескольких регистров ввода
5 (0x05) l	Force Single Coi	Запись значения в один регистр флага
6 (0x06)	Preset Single Register	Запись значения в один регистр хранения
15 (0x0F)	Force Multiple Coils	Запись значений в несколько регистров флагов
16 (0x10)	Preset Multiple Registers	Запись значений в несколько регистров флагов

В примере в качестве мастера рассматривается программируемый контроллер ОВЕН ПЛК.

Из задач контроллера можно выделить несколько основных:

– открытие и настройка порта (задание скорости обмена, стоп-битов, битов четности и номера порта);

– формирование запроса (программа контроллера должна в соответствии с протоколом обмена сформировать определенную последовательность байт или символов);

– отправка команды в сеть;

– получение ответа;

– расшифровка ответа (проверка правильности ответа, например, подсчет контрольной суммы или проверка числа байт в ответе, определение адреса устройства, от которого пришел ответ, и выборка нужных данных из ответа);

– реализация тайм-аутов (реализация времени ожидания от устройств и реализация задержки ответа).

В самом простом случае общение между мастером сети и устройством может выглядеть следующим образом (рисунок 4.1).

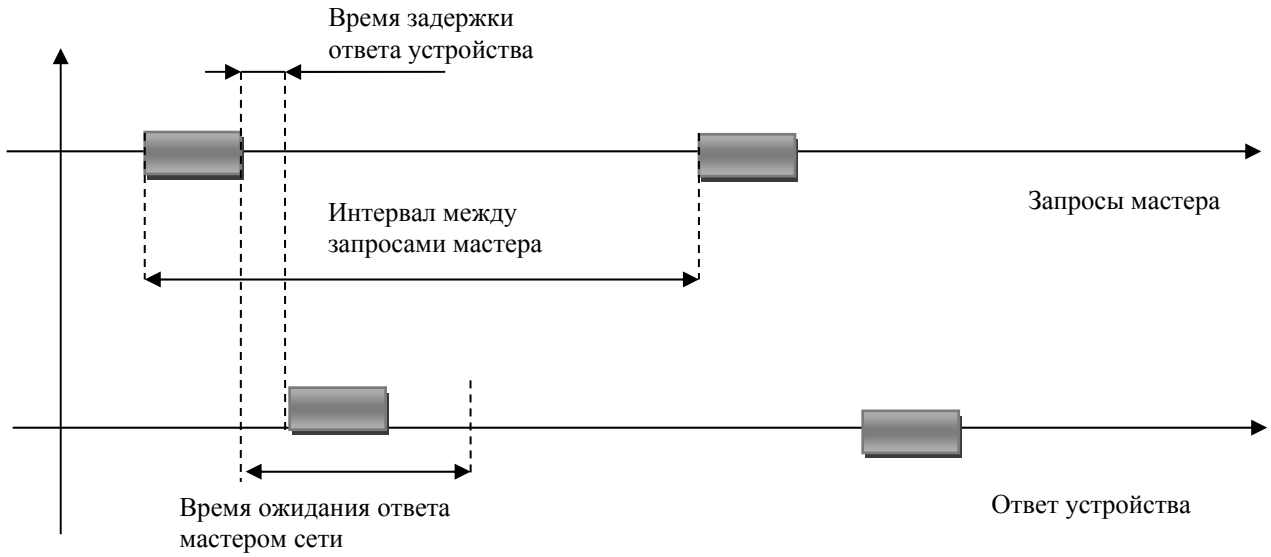


Рисунок 4.1 – Временные диаграммы

Для написания программы контроллера необходимо как минимум общее представление работы с портом. Прежде чем контроллер начнет принимать или отсылать информацию, необходимо открыть и настроить нужный порт. Непосредственная работа с портом – открытие, закрытие, настройка, чтение и запись – реализуется через библиотеку SysLibCom.lib. Для упрощения работы с портом компания ОВЕН поставляет дополнительную библиотеку ComService.lib, которая позволяет открывать и закрывать порт с необходимыми настройками. После открытия и настройки порта контроллер автоматически начинает слушать интерфейс и собирать информацию во входной буфер. Размер буфера составляет ~1,5 Кбайт, поэтому, если контроллер долгое время не считывал информацию и при этом порт был открыт, необходимо очистить входной буфер. Сделать это можно, вызывая функцию чтения порта SysComRead до тех пор, пока данная функция не будет возвращать нулевое значение. После этого можно начинать работу с устройствами.

В первую очередь контроллер должен сформировать необходимую команду, которую нужно послать в порт. Эту функцию может выполнять написанный функциональный блок (удобно в случае использования однотипных стандартизированных команд, но с различными данными), либо можно использовать заранее подготовленный список команд (например, АТ-команды модема). В примерах программ, которые будут рассмотрены далее, используются списки заранее сформированных команд.

После формирования команды ее можно отсылать в порт. Сделать это можно, используя функцию SysComWrite. На входы данной функции подается массив байт и количество байтов, необходимых для передачи. При успешной отправке функция вернет число отправленных байтов. После посыла команды контроллер следует перевести в режим чтения порта. Ответ от устройства может прийти не сразу, а с некоторой задержкой. Связано это с тем, что, во-первых, устройству необходимо получить команду, обработать и подготовить ответ, на это, как правило, тратится незначительное время, во-вторых, в большинстве устройств организованы задержки ответа для реализации тайм-аутов. Если в устройстве правильно введены сетевые настройки и линия связи защищена от помех, то ответ от устройства придет с небольшой задержкой, как правило, это не больше 50 мс. Если линия связи плохая или периодически бывают сильные помехи, которые нарушают обмен, то следует ввести таймер ожидания ответа, чтобы контроллер не ждал ответ до бесконечности, а послал новый запрос. Если помехи проходят часто, то можно организовать несколько повторных запросов одной и той же команды.

После посылки команды контроллер необходимо перевести в режим чтения порта для приема ответа. Делается это функцией SysComRead. По мере прихода данных во входной буфер функция их оттуда берет и передает во входной массив. При этом функция возвращает число принятых байтов. Принимаемую информацию можно анализировать сразу либо собирать в более большой буфер (склеивать по мере приема данных), и далее обрабатывать полностью принятый ответ. В последующих примерах для наглядности ответ от устройства собирается в увеличенный буфер, и только по мере прихода

команды полностью начинается ее обработка.

Данные, которые пишутся в порт, могут представляться в двух различных форматах. Например, рассмотрим работу контроллера с модулем МВА8 по протоколу MODBUS и работу модема ПМ01 посредством АТ-команд. В случае с модулем МВА8 данные читаются и пишутся в двоичном коде, по сути, эта работа с числами. При работе с модемом ПМ01 использовать бинарный код неудобно, т. к. в этом случае данные передаются командами, состоящими из ascii символов.

Для демонстрации работы с портом имеются две программы.

Первая программа считывала с модуля МВА8 значение температуры по протоколу Modbus. Происходило это путем чтения двух регистров:

1) регистр 0 – положение десятичной точки для входа 1;

2) регистр 1 – измерение с входа 1 в целочисленном формате со смещением точки.

Вторая программа предусматривает использование модема.

В первом случае имеются две стандартизированные команды запроса по протоколу MODBUS – это третья функция чтения, которая имеет следующий формат:

Запрос:

Адрес устройства	Номер функции	Адрес регистра	Число регистров	Контрольная сумма
1 байт	1 байт	2 байта	2 байта	2 байта

Ответ:

Адрес устройства	Номер функции	Число байт в ответе (N)	Ответ	Контрольная сумма
1 байт	1 байт	1 байт	N байт	2 байта

Если адрес МВА8 64, то запросы будут выглядеть следующим образом:

Запрос смещения (в HEX):

Адрес устройства	Номер функции	Адрес регистра		Число регистров		Контрольная сумма	
40	3	0	0	0	1	8B	1B

Ответ (в HEX):

Адрес устройства	Номер функции	Число байт в ответе	Ответ		Контрольная сумма	
40	3	2	0	2	5	8A

В данном ответе указано, что смещение равно двум знакам.

Запрос температуры в целочисленном виде со смещенной точкой (в DEC):

Адрес устройства	Номер функции	Адрес регистра		Число регистров		Контрольная сумма	
64	3	0	1	0	1	218	219

Ответ (в HEX):

Адрес устройства	Номер функции	Число байт в ответе	Ответ		Контрольная сумма	
40	3	2	9	9A	02	70

В данном ответе указано, что целочисленное значение температуры равно 2458, учитывая, что смещение равно 2, получаем, что температура с датчика равна 24,58 °С.

В примере работы с модемом команды не стандартизованы. Представляют собой набор символов, удобных для чтения, но все команды имеют разную длину (число символов в запросе) и у каждой команды собственный формат ответа.

Например, самая простая команда, которая используется при работе с модемом – AT, получив ее, модем должен прислать ответ OK, что говорит о том, что связь с модемом настроена верно. А для проверки регистрации модема в сети используется команда AT+CREG?, на которую контроллер даст ответ в формате: +CREG:<n>,<m> .

Исходя из этого, можно сделать вывод, что в зависимости от протокола обмена может понадобиться различная методика обработки данных. В случае с протоколом MODBUS это однотипные команды и ответы, а в случае с AT-командами команды имеют разную длину и разный формат. Поэтому было написано два примера. В первом примере был написан функциональный блок, на вход которого подавалась команда определенной длины, далее эта команда отправлялась в порт, и контроллер ожидал ответа. По мере прихода ответа функциональный блок из ответа выбирал данные и передавал их на свой выход. Если за отведенное время ответа от устройства не было или он был частичным, то на выходе блока выдавалась ошибка. При работе с AT-командами написать один функциональный блок, который мог бы обработать все форматы команд неудобно и не нужно. Проще сделать функциональный блок, на вход которого подается строка и длина ожидаемого ответа. В зависимости от подаваемой на вход блока команды на выходе блока приходят ответы различной длины и формата. После приема данных их можно обработать в программе самого контроллера.

В обеих программах реализованы тайм-ауты ожидания ответа и пауза между последующими посылками, чтобы исключить наложение данных. В случае обрыва связи или прохождения помехи функциональные блоки генерируют код ошибки:

81 – связи нет, за время ожидания данные не пришли вообще;

87 – данные пришли частично, за время ожидания ответа.

Также введена возможность повторного запроса, по умолчанию в программе стоит два повтора. Если данные не приходят или есть ошибки, то контроллер еще два раза попробует получить данные.

Выбор языка ST связан со сложностью написания программы, на других языках данные алгоритмы организовать либо сложно, либо не возможно.

Во вложении имеются две программы, написанные для ПЛК154-У.М, с версией прошивки 2.10.9 и версией Таргет-файла 2.10.

Код программ не привязан к конкретной модели ОВЕН ПЛК, т. к. не

используются физические входы и выходы контроллера. Если необходимо запустить программу на другой модели ОВЕН ПЛК, то следует последовательно выполнить следующие операции:

– открыть вкладку ресурсы;

– найти раздел настройки целевой платформы и щелкнуть по нему 2 раза;

– в открывшемся меню выбрать модель контроллера и нажать кнопку ОК;

– открыть вкладку ресурсы, найти раздел конфигурация контроллера и щелкнуть по нему 2 раза.

В панели инструментов выбрать меню Дополнительно и выполнить команду Стандартная конфигурация.

Теперь в конфигурации контроллера нужно выставить минимальное время цикла 5 мс.

В панели инструментов выбрать меню проект и выполнить команду «Очистить все».

В панели инструментов выбрать меню проект и выполнить команду «Компилировать все».

Проект готов к загрузке в контроллер.

Порядок проведения практической работы

При выполнении работы необходимо дополнительно использовать Руководство пользователя по программированию ПЛК в CoDeSys 2.3 и Справочную систему комплекса программирования CoDeSys.

Тексты описанных программ приведены в папке проектов системы программирования.

1 Изучить работу типовых блоков, реализующих функции передачи, которые входят в библиотеку SysLibCom.lib системы CoDeSys .

2 Изучить примеры применения типовых функциональных блоков в управляющих программах.

3 Проверить работу программ, приведенных в примерах, в режиме эмуляции.

4 Записать программы в память контроллера и проверить их выполнение.

5 Составить отчет по работе.

Содержание отчета

Отчет должен содержать следующее.

1 Титульный лист установленного образца.

2 Цель работы.

3 Графическое изображение блоков, реализующих передачу данных.

4 Временные диаграммы циклов передачи и приема данных.

5 Листинги программ.

6 Вывод по работе.

5 Практическая работа № 5. Разработка визуализаций в среде CoDeSys

Цель практической работы: изучить принципы создания визуализаций проекта в системе CoDeSys, приобрести навыки создания визуализаций в системе CoDeSys.

Общие сведения

Визуализация представляет собой графическое изображение проектируемой системы, которое может служить пользовательским интерфейсом для контроля и управления работой системы.

Визуализация может исполняться в системе программирования, в отдельном приложении CoDeSys HMI, как Web приложение на сервере или как целевая программа в ПЛК.

Редактор визуализации CoDeSys предоставляет набор готовых графических элементов, которые могут быть связаны соответствующим образом с переменными управляющей программы. Форма и цвет графических элементов могут изменяться при работе программы 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.

На рисунке 5.1 приведен пример визуализации.

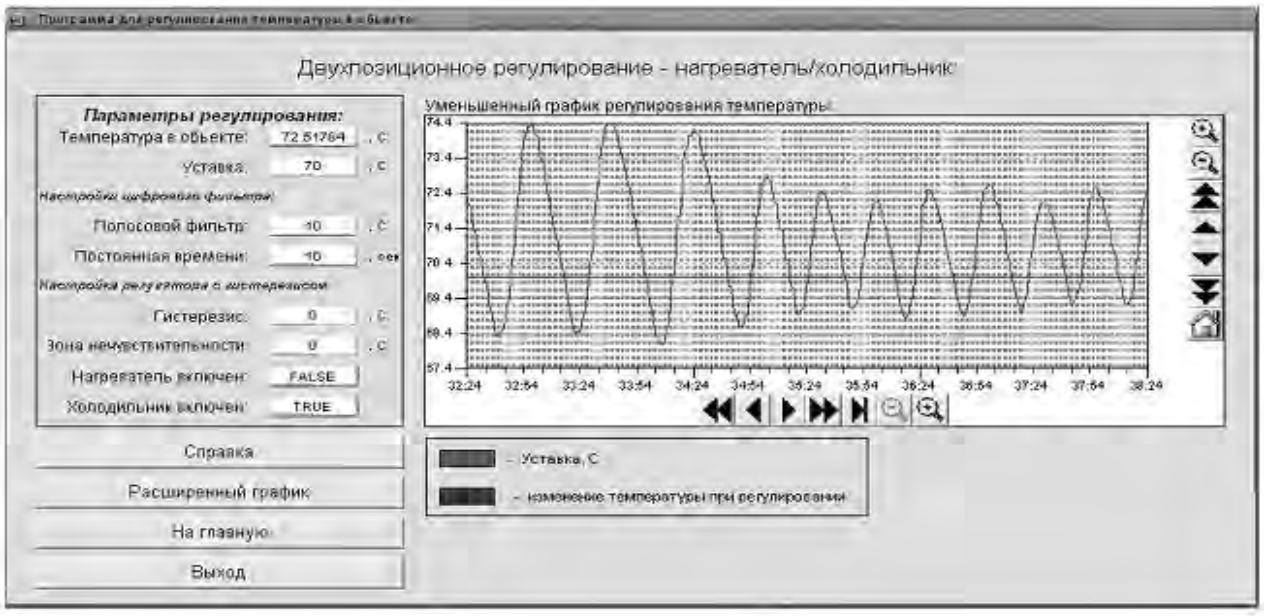


Рисунок 5.1 – Пример визуализации программы контроллера

Создание файла визуализации.

Для создания визуализации нужно перейти на вкладку «Визуализации» организатора объектов, после чего правой кнопкой мыши вызвать контекстное меню в пустом поле и выбрать строку «Добавить объект», после этого появится окно визуализации (рисунок 5.2).

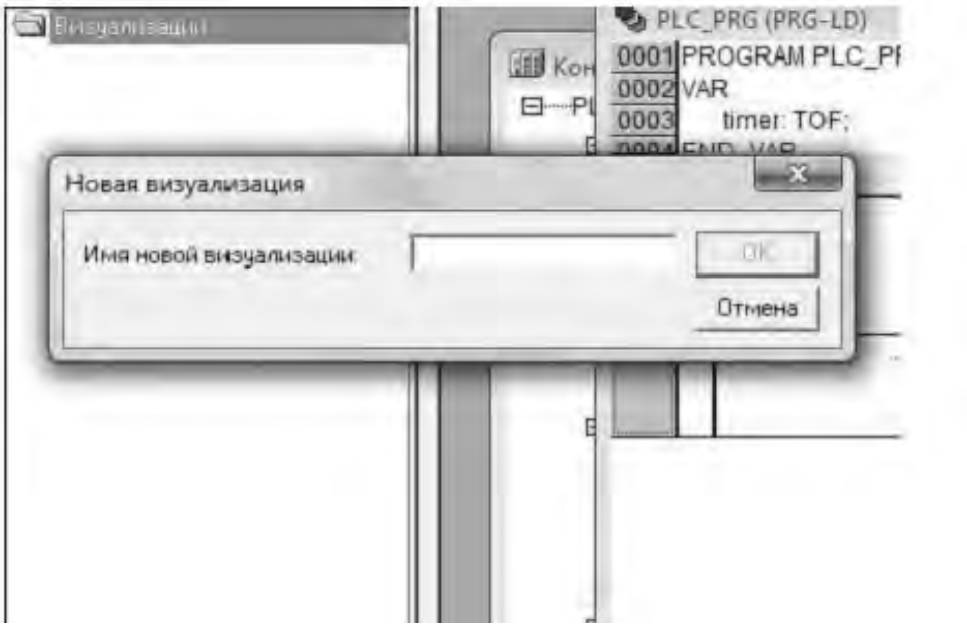


Рисунок 5.2 – Вызов окна визуализации

Элемент визуализации – это графический элемент, который используется при построении объекта визуализации. Возможные элементы представлены в виде иконок на панели инструментов CoDeSys (рисунок 5.3). Каждый элемент имеет собственную конфигурацию (набор свойств). Имеется возможность вставлять в визуализацию различные геометрические формы, а также точечные рисунки, метафайлы, кнопки и существующие визуализации.

Рисунок 5.3 – Панель инструментов редактора визуализации

У каждого элемента визуализации есть свои свойства; вызвать свойства объекта визуализации можно двойным щелчком мыши по объекту либо активизировать правой кнопкой мыши объект в контекстном меню и выбрать строку «Конфигурировать». В открывшемся окне можно задавать необходимые свойства объекта визуализации (рисунок 5.4).

Пример **–** Создание простейшей визуализации.

Создадим программу на языке LD в виде цепочки, состоящей из операторов «Контакт» и «Катушка».

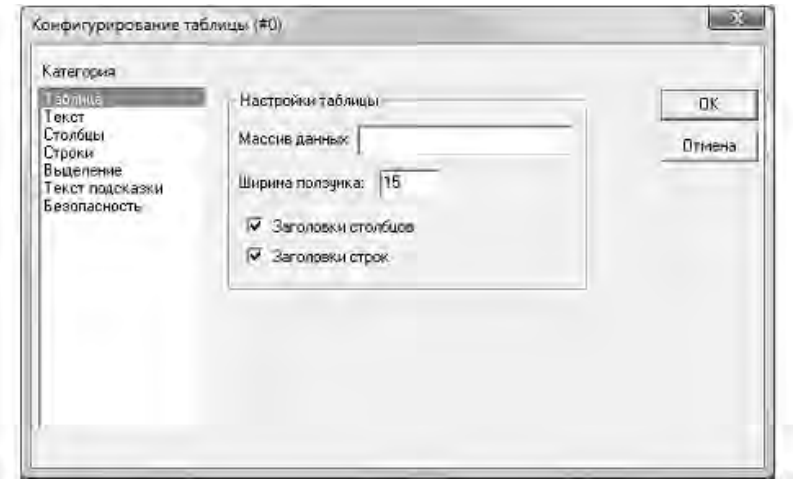


Рисунок 5.4 – Окно конфигурирования

Перейдем на вкладку «Визуализация» и вызовем контекстное меню для добавления новой визуализации (рисунок 5.5).

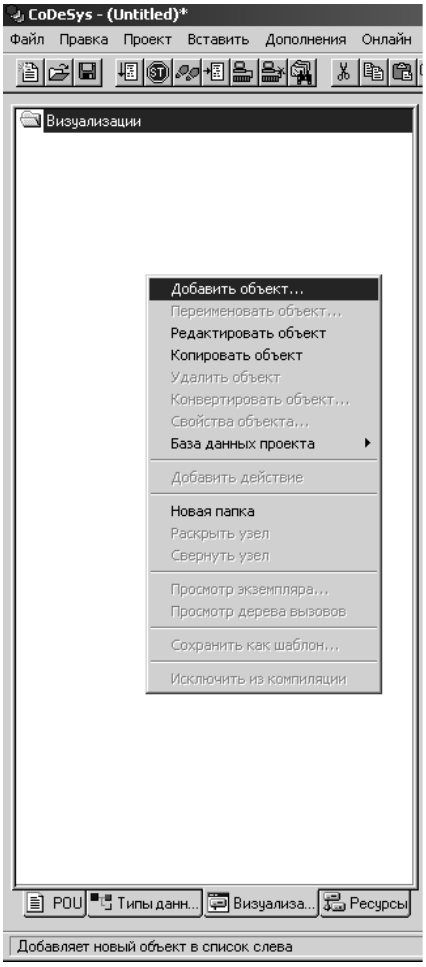


Рисунок 5.5 – Добавление объекта визуализации

В появившемся окне введем имя визуализации (рисунок 5.6).
После ввода имени появляется новое окно для создания визуализации.
Вставляем с помощью панели инструментов элементы «Эллипс» и

«Кнопку», как на рисунке 5.7.

Рисунок 5.6 – Ввод имени визуализации

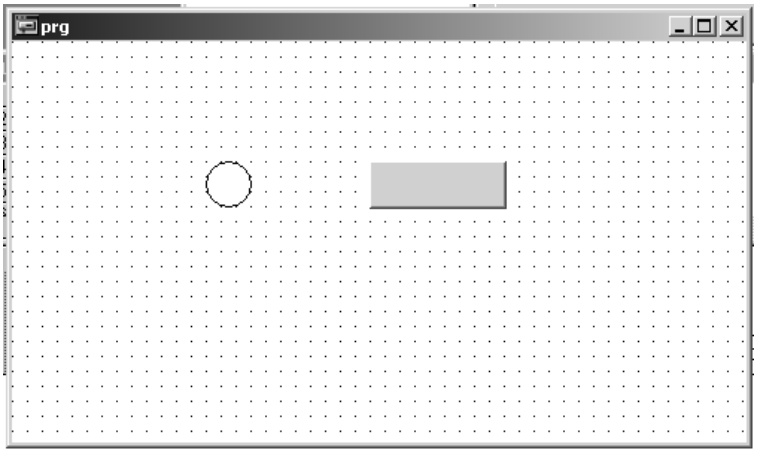


Рисунок 5.7 – Ввод элементов визуализации

Теперь нужно сконфигурировать эти элементы. Пусть необходимо сделать следующую визуализацию: эллипс должен менять цвет, когда срабатывает катушка реле, а при нажатии на кнопку должна изменять значение переменная типа BOOL.

Сначала добавляем переменную C типа BOOL в проект (рисунок 5.8).

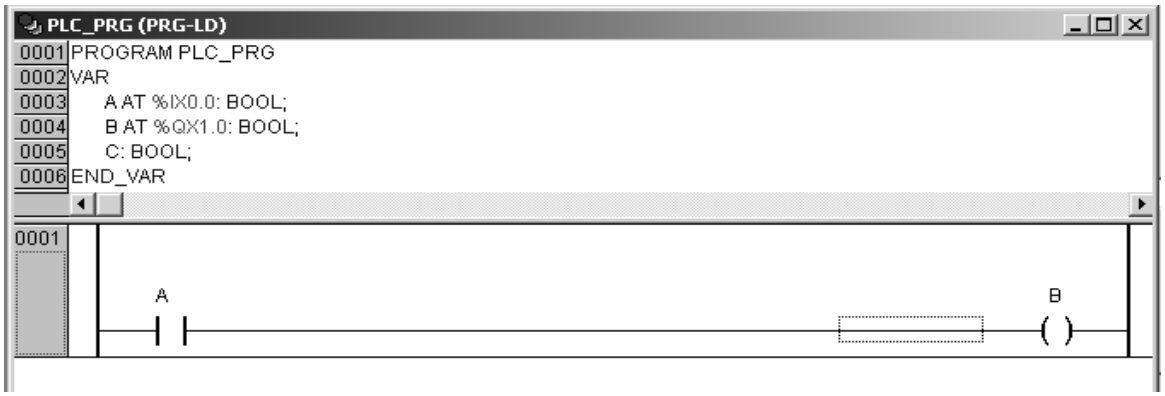


Рисунок 5.8 – Переменные проекта

Затем перейдем в окно редактора визуализации и щелкнем по эллипсу для вызова окна с настройками (рисунок 5.9).

Настроим категории «Цвета» и «Переменные». В категории «Цвета» выберем цвет эллипса в двух состояниях (рисунок 5.10).

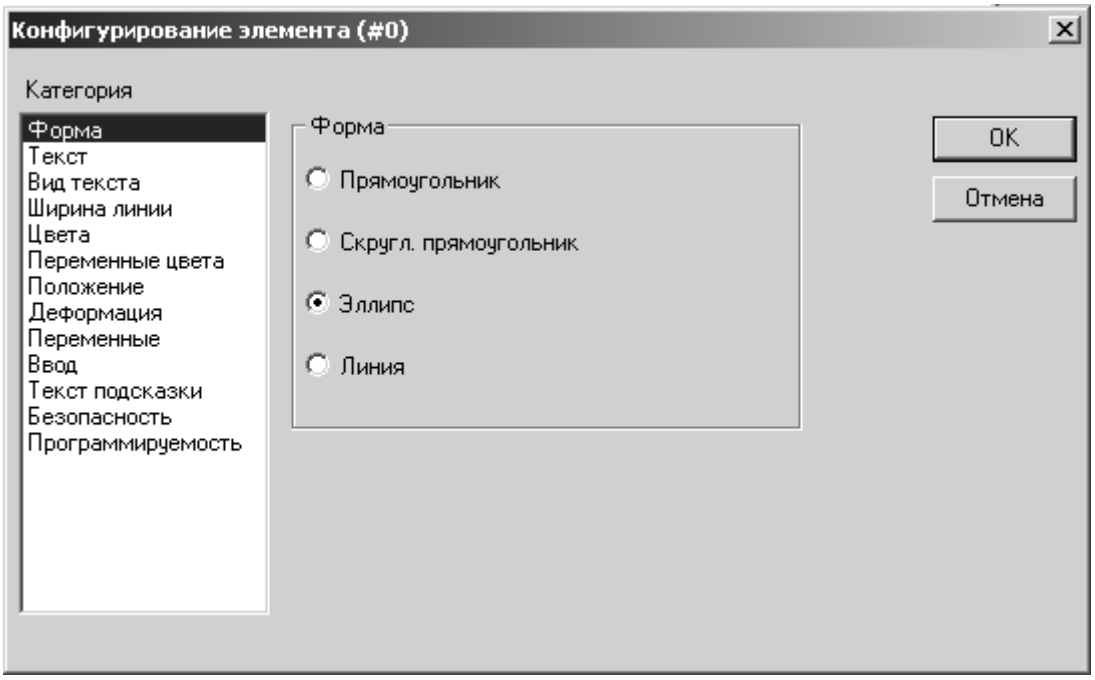


Рисунок 5.9 – Окно конфигурирования эллипса

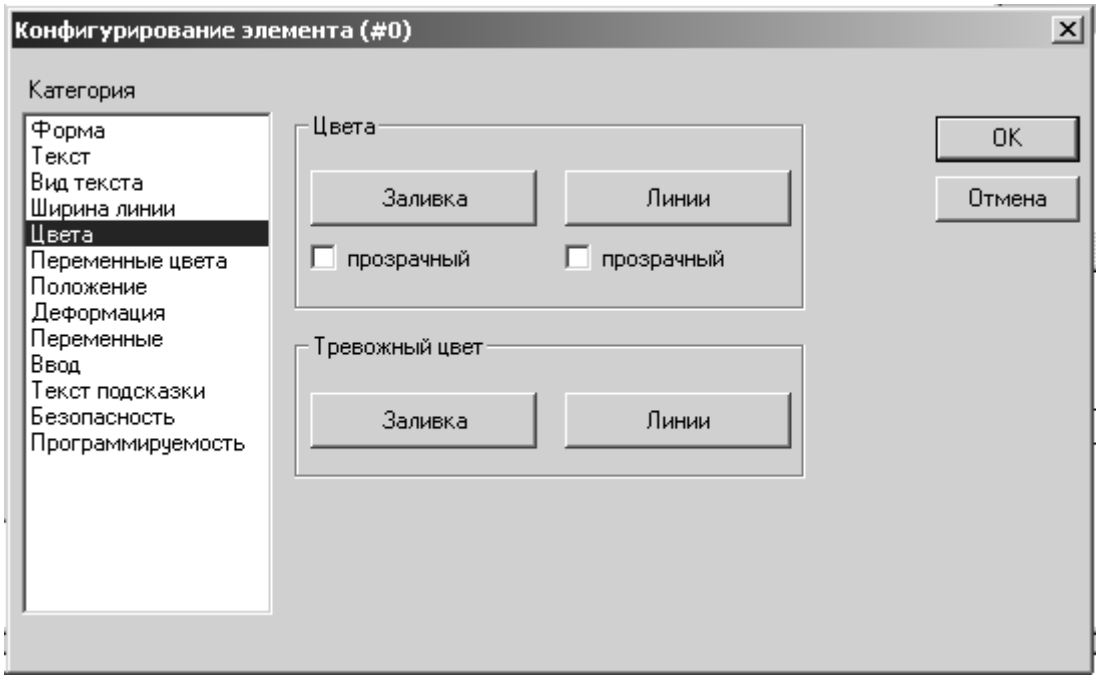


Рисунок 5.10 – Окно конфигурации цвета эллипса

Затем необходимо указать переменную, которая будет изменять цвет; для этого перейдем в категорию «Переменные» и установим курсор в поле «Изм. цвета:».

Для связывания с переменными используем ассистент ввода (клавиша F2), в появившемся окне выберем нужную переменную – переменная B (рисунок 5.11).

После нажатия кнопки OK в окне конфигурации переменных появится переменная для изменения цвета (рисунок 5.12).

Далее вызовем окно конфигурирования кнопки (рисунок 5.13) и выберем категорию «Ввод».

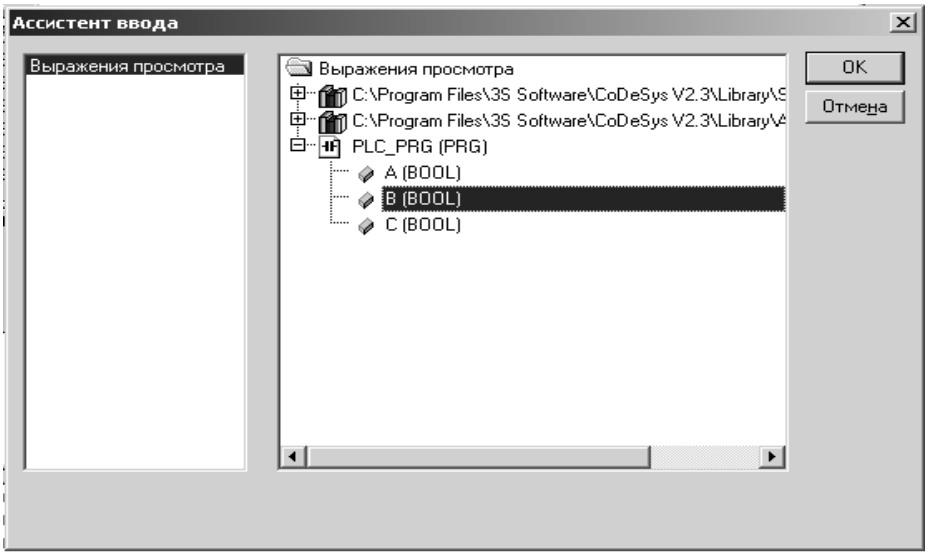


Рисунок 5.11 – Окно ассистента ввода

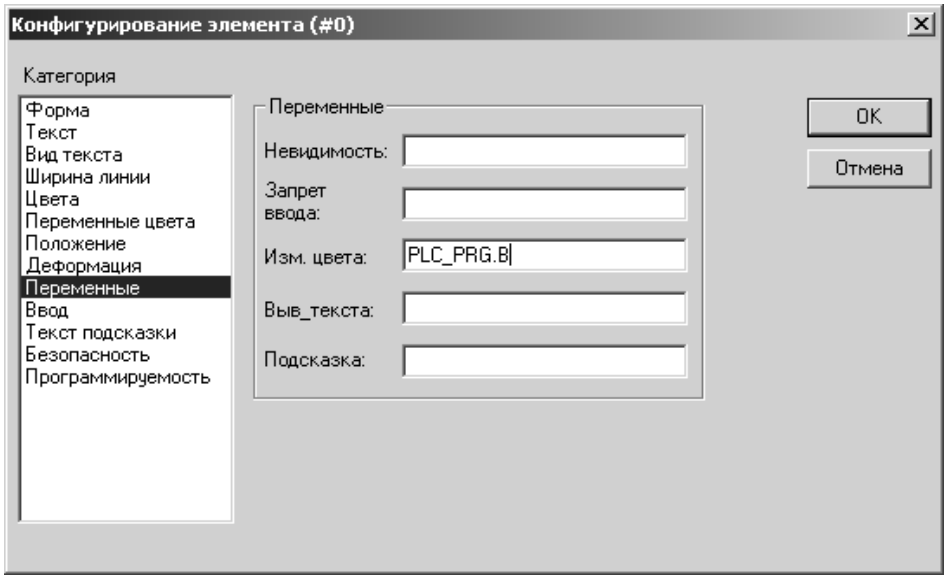


Рисунок 5.12 – Сконфигурированное окно переменных эллипса

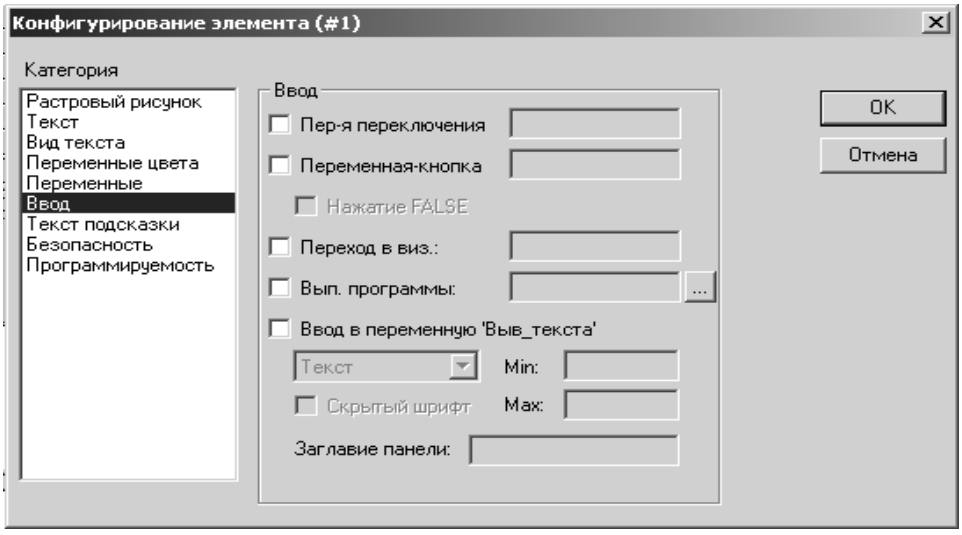


Рисунок 5.13 – Окно конфигурирования переменных кнопки

Поставим галочку «Переменная кнопка» (чтобы получить кнопку с фиксацией, необходимо выбирать «Переменная переключения») и, установив курсор в появившееся поле ввода, вызовем ассистент ввода, чтобы привязать к кнопке переменную C.

Сконфигурированное окно переменных показано на рисунке 5.14.

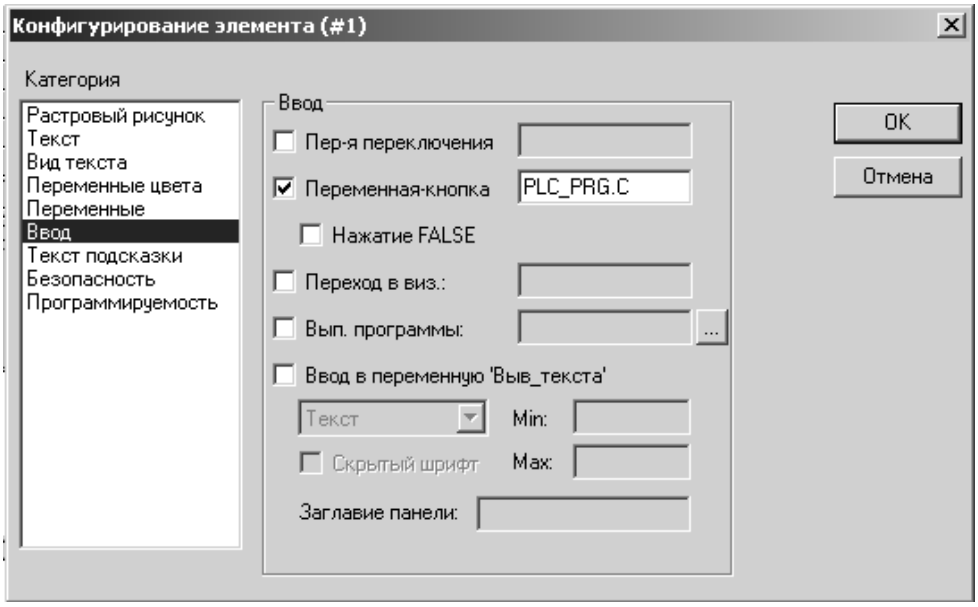


Рисунок 5.14 – Сконфигурированное окно переменных кнопки

Создание визуализации завершено. Можно загрузить программу в ПЛК и проверить работу визуализации.

Порядок проведения практической работы

1 Изучить основные правила создания визуализаций в системе CoDeSys.

2 Изучить состав элементов визуализации в системе программирования CoDeSys.

3 Изучить порядок вставки, позиционирования и конфигурирования элементов визуализации в системе CoDeSys.

4 Изучить порядок конфигурирования объектов визуализации в системе CoDeS.

5 Создать визуализацию для программы управления двигателем.

6 Проверить работу визуализации при выполнении программы.

7 Составить отчет по работе.

При выполнении работы необходимо дополнительно использовать справочную систему среды программирования CoDeSys и дополнение к Руководству пользователя по программированию ПЛК в CoDeSys 2.3. Визуализация CoDeSys.

Содержание отчета

Отчет по работе должен содержать следующее.

1 Титульный лист установленного образца.
2 Цель работы.
3 Описание последовательности создания визуализации.
4 Список элементов визуализации в системы CoDeSys.
5 Вывод по работе.

Список литературы

1 **Петров**, **И. В.** Программируемые контроллеры. Стандартные языки и инструменты / И. В. Петров. – Москва : СОЛОН-Пресс, 2003. – 256 с.

2 Руководство пользователя по программированию ПЛК в CoDeSys 2.3. – Смоленск: Пролог, 2008. – 452 с.

3 **Минаев**, **И. Г.** Программируемые логические контроллеры. Практическое руководство для начинающего инженера / И. Г. Минаев, В. В. Самойленко. – Ставрополь : АРГУС, 2009. – 100 с.

4 Общие сведения о CoDeSys [Электронный ресурс]. – Режим доступа: http://www.3s-software.ru /publications. – Дата доступа: 06.05.2017.

5 Каталог продукции фирмы ОВЕН [Электронный ресурс]. – Режим доступа: http://www.owen.ru/catalog. – Дата доступа: 16.05.2017.