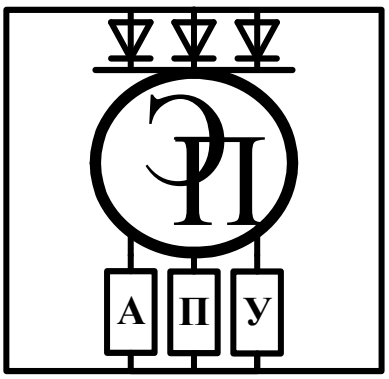
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и автоматизация промышленных установок»

СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Методические рекомендации к практическим занятиям для студентов специальности 1-53 01 01 «Автоматизация технологических процессов и производств (по направлениям)» очной и заочной форм обучения



Могилев 2023

УДК 658.012.011.56
ББК 30.2-5-05
С40

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и АПУ» «14» февраля 2023 г., протокол № 6

Составитель ст. преподаватель А. В. Янкович

Рецензент канд. техн. наук, доц. С. В. Болотов

Методические рекомендации к практическим занятиям для студентов специальности 1-53 01 01 «Автоматизация технологических процессов и производств (по направлениям)» очной и заочной форм обучения.

Учебное издание

СИСТЕМЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Ответственный за выпуск	А. С. Коваль
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 81 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Практическая работа № 1. Разработка программ управления дискретной автоматикой

Цель практической работы: получение студентами теоретических сведений 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 системах 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 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 практически 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 владеющими 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. Очевидно, что только в одном случае результатом логического умножения станет единица, т. е. лампа Л «сработает», если замкнуть контакты и *a* и *b*. Из этой словесной формулы союз «И» перешел в обозначение функции (как

синоним логическому умножению) и в название бесконтактного элемента. Написание: $Л = a \cdot b$.

Релейно-контактный вариант реализации представлен на рисунке 1.1.

Таблица состояний представлена на рисунке 1.2.

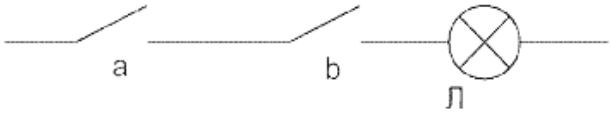


Рисунок 1.1 – Релейно-контактный вариант реализации

a	b	Л
0	0	0
0	1	0
1	0	0
1	1	1

Рисунок 1.2 – Таблица состояний

Обозначение бесконтактного элемента представлено на рисунке 1.3.

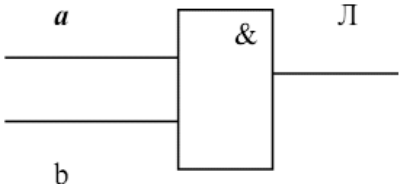


Рисунок 1.3 – Обозначение бесконтактного элемента

Временные диаграммы представлены на рисунке 1.4.

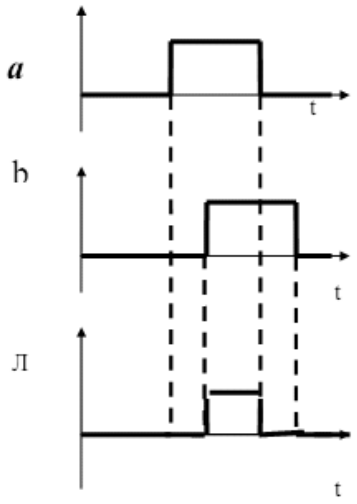


Рисунок 1.4 – Временные диаграммы

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

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.

Релейно-контактный вариант реализации представлен на рисунке 1.5.

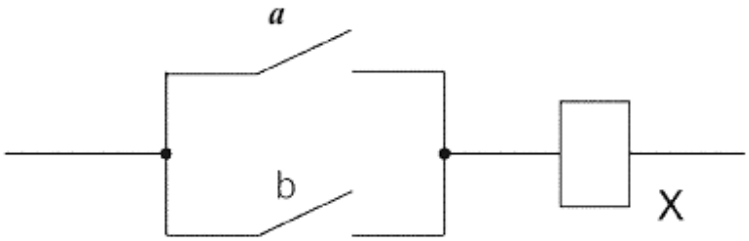


Рисунок 1.5 – Релейно-контактный вариант реализации

Таблица состояний представлена на рисунке 1.6.

Обозначение бесконтактного элемента представлено на рисунке 1.7.

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

Рисунок 1.6 – Таблица состояний

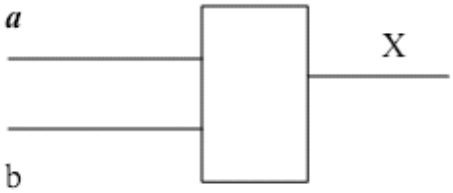


Рисунок 1.7 – Обозначение бесконтактного элемента

Временные диаграммы представлены на рисунке 1.8.

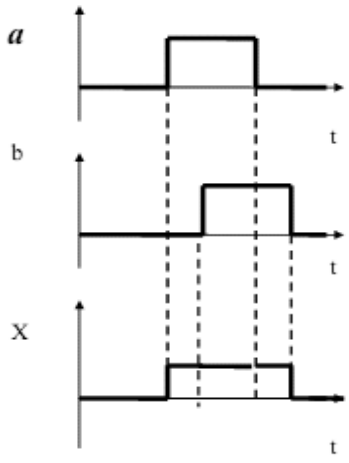


Рисунок 1.8 – Временные диаграммы

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

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

Логическое отрицание.

Логическое отрицание (инверсия, операция «НЕ»), $Л = \overline{a}$ означающая, что значение логической функции $Л$ противоположно или неравносильно значению переменной a. В нашем примере лампа горит ($Л = 1$), если контакт a разомкнут ($a=0$), и лампа гаснет ($Л = 0$), если контакт замкнуть ($a = 1$).

Релейно-контактный вариант реализации представлен на рисунке 1.9.

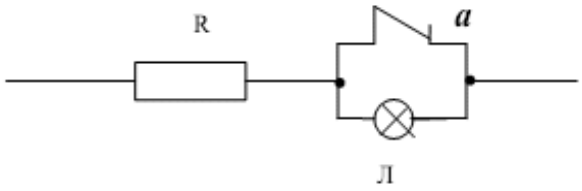


Рисунок 1.9 – Релейно-контактный вариант реализации

Таблица состояний представлена на рисунке 1.10.

Обозначение бесконтактного элемента представлено на рисунке 1.11.

a	Л
0	1
1	0

Рисунок 1.10 – Таблица состояний

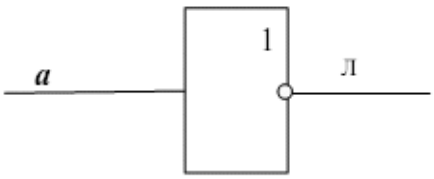


Рисунок 1.11 – Обозначение бесконтактного элемента

Временные диаграммы представлены на рисунке 1.12.

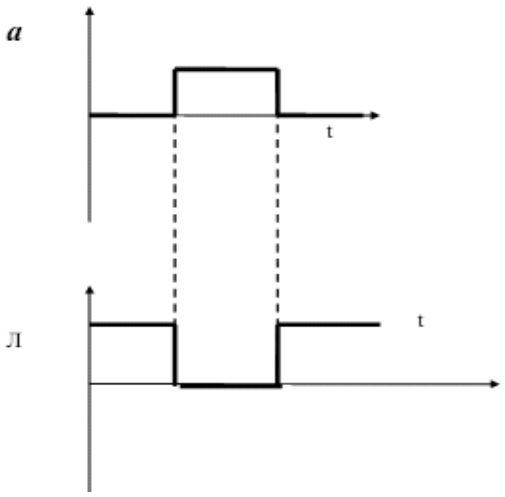


Рисунок 1.12 – Временные диаграммы

Логическое сложение по модулю 2.

Функция «**ИСКЛЮЧАЮЩЕЕ ИЛИ**»: $Л = a \cdot \bar{b} + \bar{a} \cdot b$.

Релейно-контактный вариант реализации представлен на рисунке 1.13.

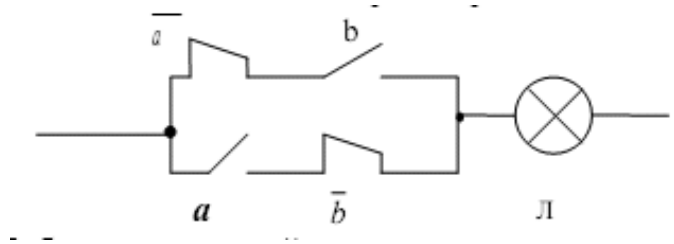


Рисунок 1.13 – Релейно-контактный вариант реализации

Таблица состояний представлена на рисунке 1.14.

Обозначение бесконтактного элемента представлено на рисунке 1.15.

a	b	Л
0	0	0
0	1	1
1	0	1
1	1	0

Рисунок 1.14 – Таблица состояний

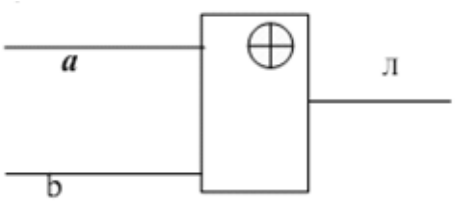


Рисунок 1.15 – Обозначение бесконтактного элемента

Временные диаграммы представлены на рисунке 1.16.

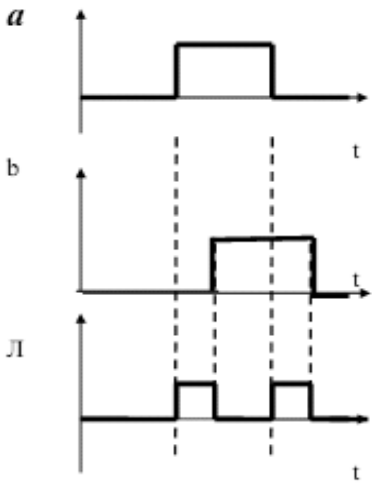


Рисунок 1.16 – Временные диаграммы

Инверсия конъюнкции.

Функция «**И–НЕ**»: $Л = \overline{a \cdot b}$.

Релейно-контактный вариант реализации представлен на рисунке 1.17.

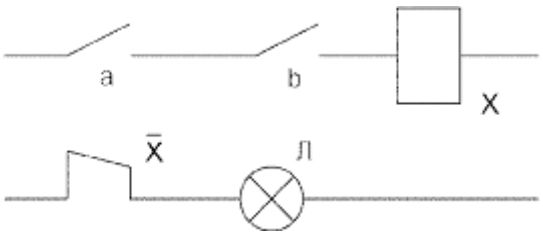


Рисунок 1.17 – Релейно-контактный вариант реализации

Таблица состояний представлена на рисунке 1.18.
Обозначение бесконтактного элемента представлено на рисунке 1.19.

a	b	Л
0	0	1
0	1	1
1	0	1
1	1	0

Рисунок 1.18 – Таблица состояний

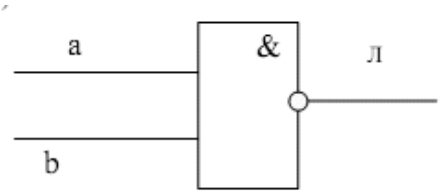


Рисунок 1.19 – Обозначение бесконтактного элемента

Временные диаграммы представлены на рисунке 1.20.

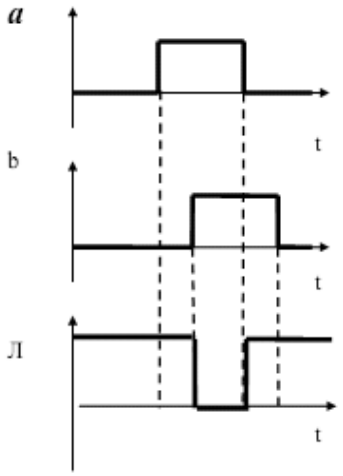


Рисунок 1.20 – Временные диаграммы

Инверсия дизъюнкции.

Функция «**ИЛИ**–**НЕ**»: $3 = \overline{a + b}$.
Релейно-контактный вариант реализации представлен на рисунке 1.21.
Таблица состояний представлена на рисунке 1.22.
Обозначение бесконтактного элемента представлено на рисунке 1.23.
Временные диаграммы представлены на рисунке 1.24.

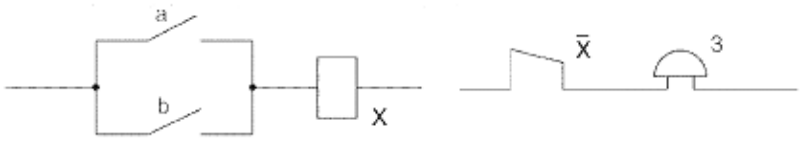


Рисунок 1.21 – Релейно-контактный вариант реализации

a	b	3
0	0	1
0	1	0
1	0	0
1	1	0

Рисунок 1.22 – Таблица состояний

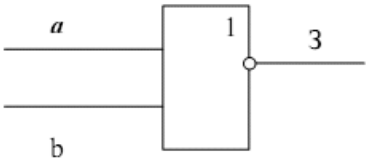


Рисунок 1.23 – Обозначение бесконтактного элемента

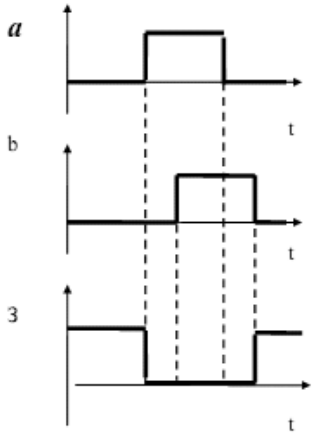


Рисунок 1.24 – Временные диаграммы

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

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 следующие законы.

1 Переместительный (коммутативный):

а) относительно логического умножения: $ab = ba$;

б) относительно логического сложения: $a + b = b + a$.

2 Сочетательный (ассоциативный):

а) относительно логического умножения: $(ab)c = (ac)b$;

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

3 Распределительный (дистрибутивный):

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

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

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 нашем 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 нашем 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 (рисунок 1.25).

В таблице состояния 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.

	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

Рисунок 1.25 – Таблица состояния

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*.

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

Релейно-контактные варианты проектируемой системы логического управления имеют вид, представленный на рисунке 1.26.

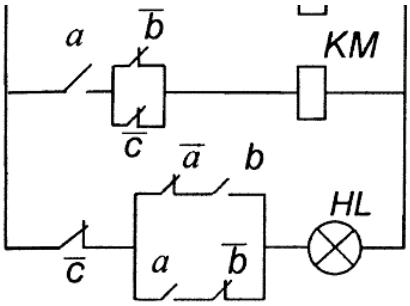


Рисунок 1.26 – Вид релейно-контактного варианта проектируемой системы логического управления

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 1.27) выбрать тип POU Программа (Program) и язык, на котором будет осущест вляться написание программы LD (релейные диаграммы). Имя программы можно оставить без изменения. Подтвердить выбор нажатием на кнопку OK.

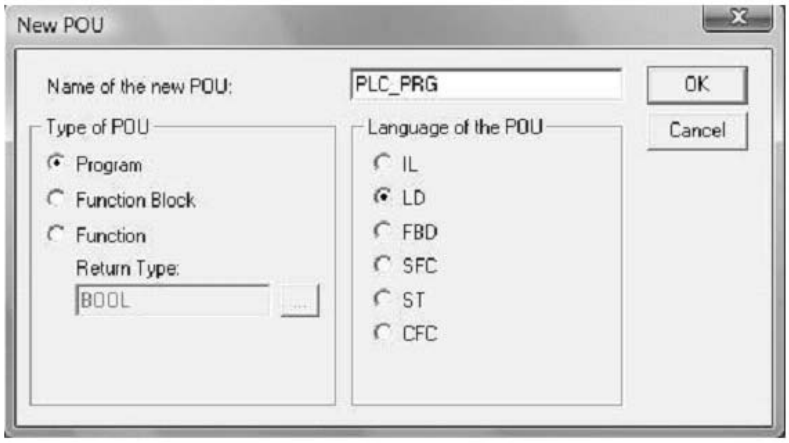


Рисунок 1.27 – Выбор языка программирования и задание имени программы

После выполнения всех вышеописанных действий откроется главное окно (рисунок 1.28) среды CoDeSys.

Рисунок 1.28 – Главное окно среды CoDeSys

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

1 Меню.

2 Панель инструментов. На ней находятся кнопки для быстрого вызова команд меню.

3 Организатор объектов, имеющий вкладки POU, Data types, Visualizations и Resources.

4 Разделитель Организатора объектов и рабочей области CoDeSys.

5 Рабочая область, в которой находится редактор.

6 Окно сообщений.

7 Строка статуса, содержащая информацию о текущем состоянии проекта.

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 выполняется в активном окне.

Команда выполнится, как только нажатая на панели инструментов кнопка будет отпущена. Если вы поместите указатель мышки на кнопку панели инструментов, то через небольшой промежуток времени увидите 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 находятся 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 переменная имеет значение ЛОЖЬ, то контакт считается 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 – логической операции «ИЛИ». Операции «НЕ» соответствует нормально замкнутый контакт, который размыкает цепь (фактически, дает логический нуль) при включении (подачи 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 появится 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.

9 Составить отчет по работе.

Содержание отчета

Отчет по работе должен содержать следующее.

1 Титульный лист установленного образца.

2 Цель работы.

3 Список инструкций языка программирования LD, реализующих логические функции.

4 Листинг программы.

5 Вывод по практической работе.

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

Цель практической работы: получение студентами теоретических сведений 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.

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 состояние выходов этих триггеров, и катушки L и M

остаются включенными. Для отключения реле M необходимо кратковременно нажать кнопку Res.

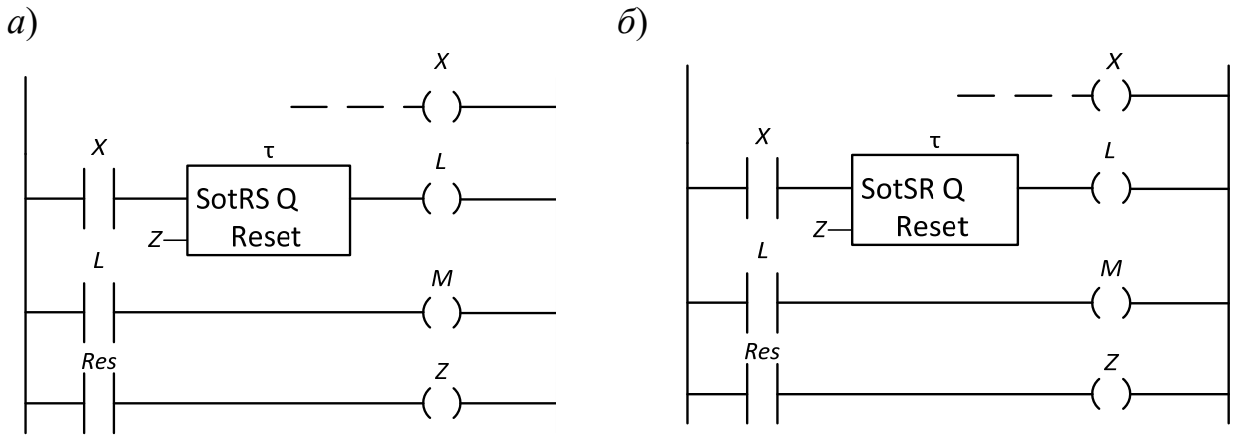


Рисунок 2.1 – Фрагменты схем с RS- и SR-триггерами

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.

Необходимо напомнить, что входу RESET этих триггеров необходимо присвоить идентификатор (имя) того реле, которое будет обеспечивать сброс. В нашем 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.

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.

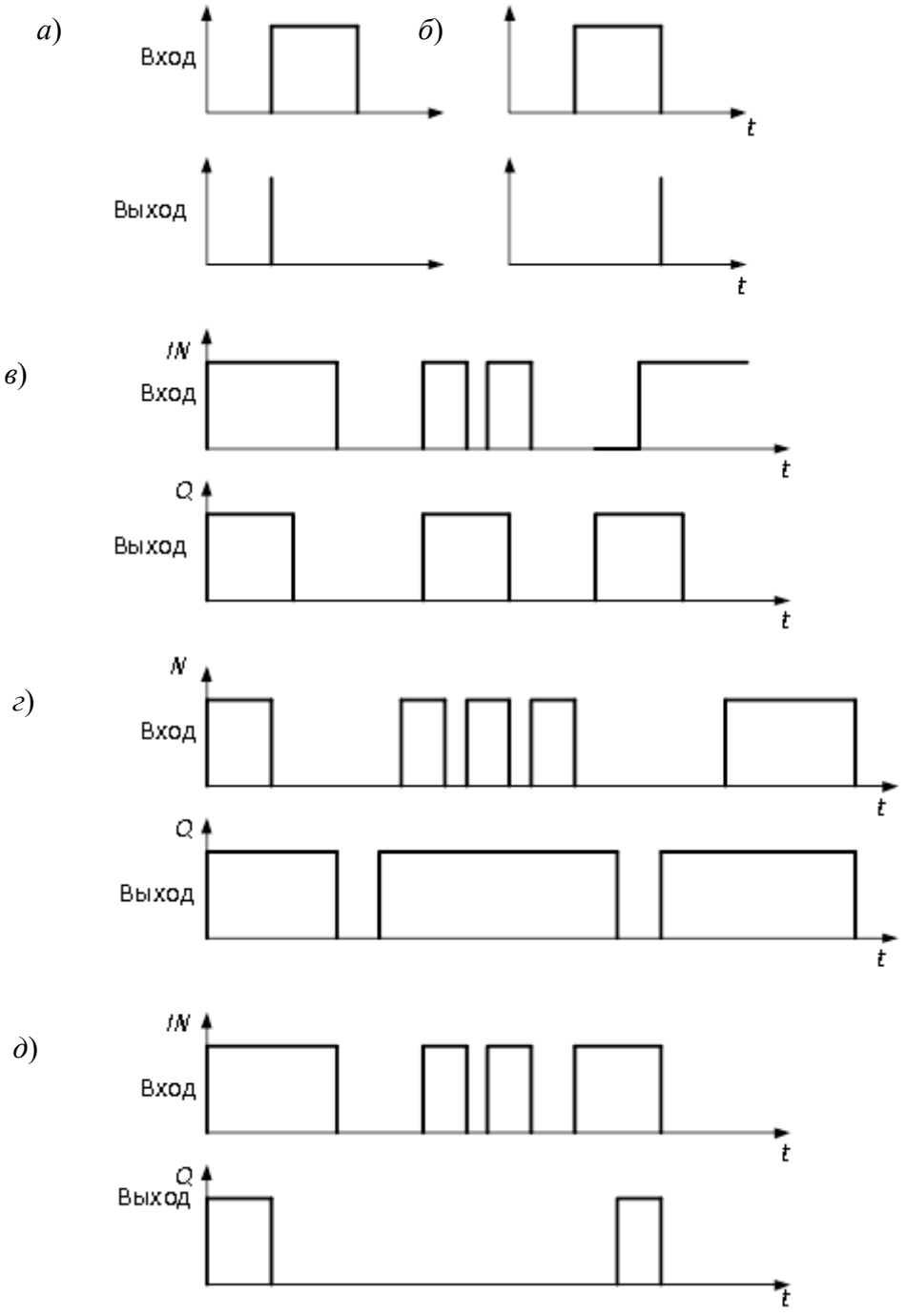
Работу этих таймеров поясняют временные диаграммы (см. рисунок 2.2).

Из этих диаграмм следует, что TP-таймер запускается мгновенно передним фронтов входного сигнала и в течение времени *T* 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 *T* 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 (см. рисунок 2.2, *д*).



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

Рисунок 2.2 – Временные диаграммы детекторов импульсов и таймеров

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).

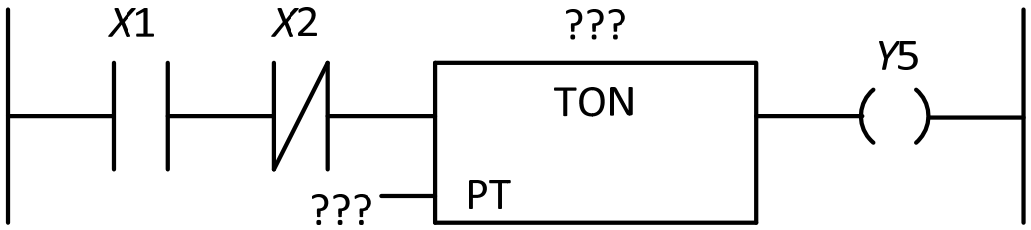


Рисунок 2.3 – Фрагмент схемы со вставленным таймером

Щелкнув левой клавишей мыши по верхним знакам ???, присваиваем имя. Например, N1. Щелкнув по знакам ??? у хода PT, нажать Shift, и не отпуская 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 .

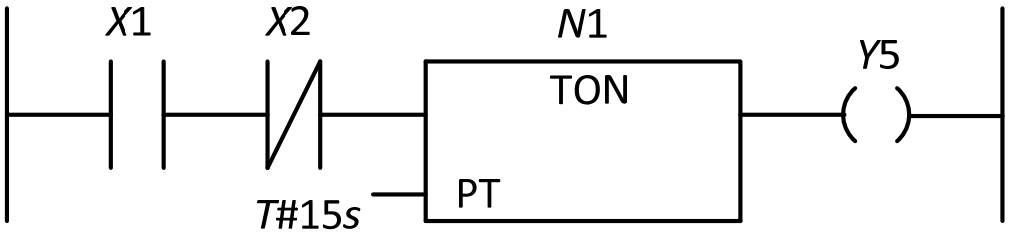


Рисунок 2.4 – Фрагмент схемы после записи уставок

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 счетчик, CTD – декрементный; CTUD – инкрементный/декрементный.

Простейшая схема с CTU-счетчиком показана на рисунке 2.5.

После переноса счетчика в цепь многоступенчатой схемы появятся знаки ??? 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.

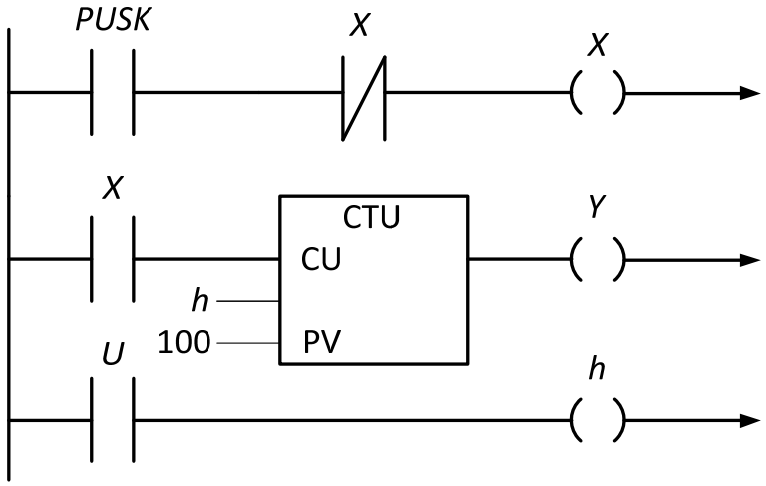


Рисунок 2.5 – Схема с CTU-счетчиком

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 единицу 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 CTU тем, что каждый входной импульс уменьшает значение счетчика на единицу. 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 единицу значение целой переменной при наличии положительного фронта на дискретном входе 0 и уменьшает на единицу значение этой переменной 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 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.

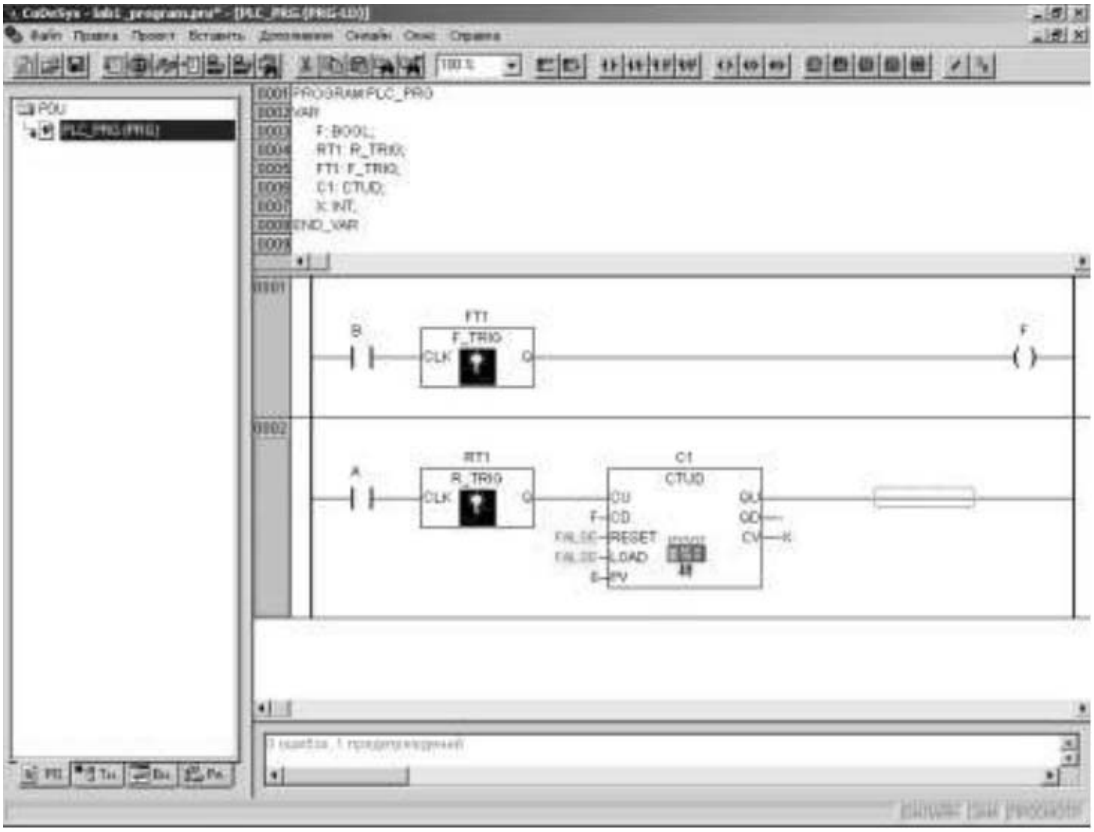


Рисунок 2.6 – Программа демонстрация работы детекторов фронтов и реверсивного счетчика

С первого дискретного входа значение сигнала подается на вход блока R_TRIG, объявленного как переменная RT1, со второго – на F_TRIG, объявленный, 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 (т. е. 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.

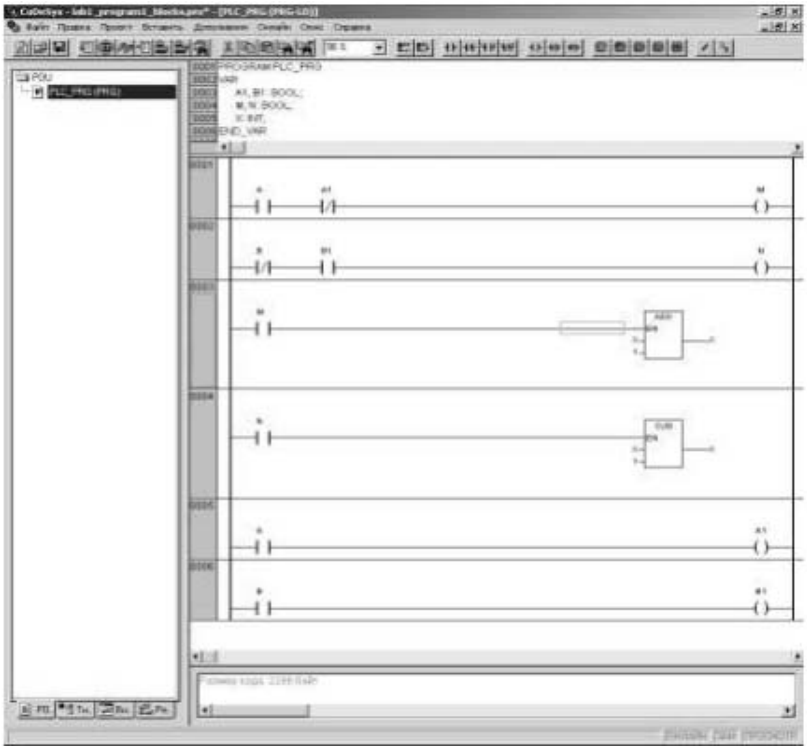


Рисунок 2.7 – Программа демонстрация работы детекторов фронтов и реверсивного счетчика

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 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, и 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.

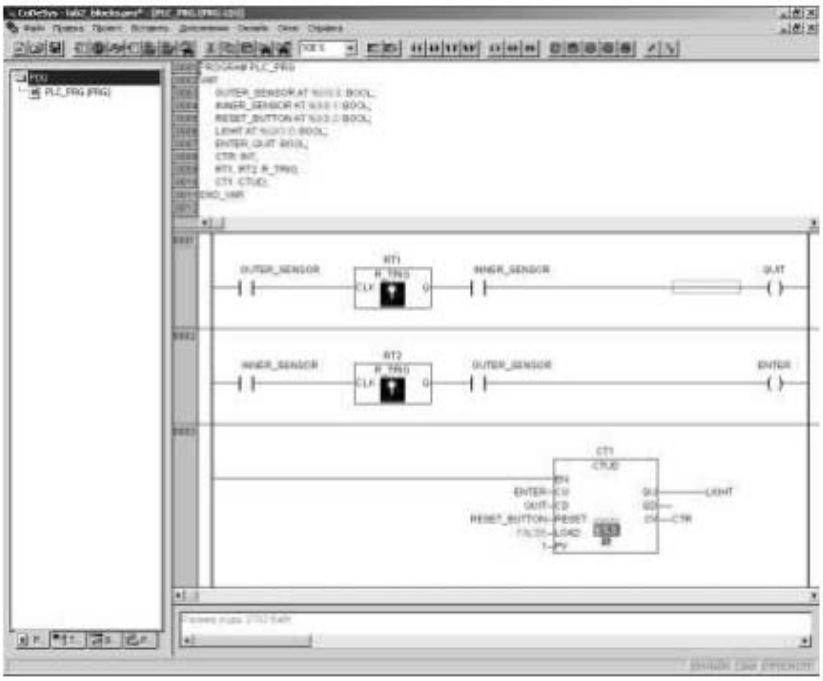


Рисунок 2.8 – Решение задачи об автоматическом включении света, сделанное с помощью типовых функциональных блоков

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 также, 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 единице, следует включить свет, если нет, то 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, соответственно, выхода из комнаты и входа в нее. 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.

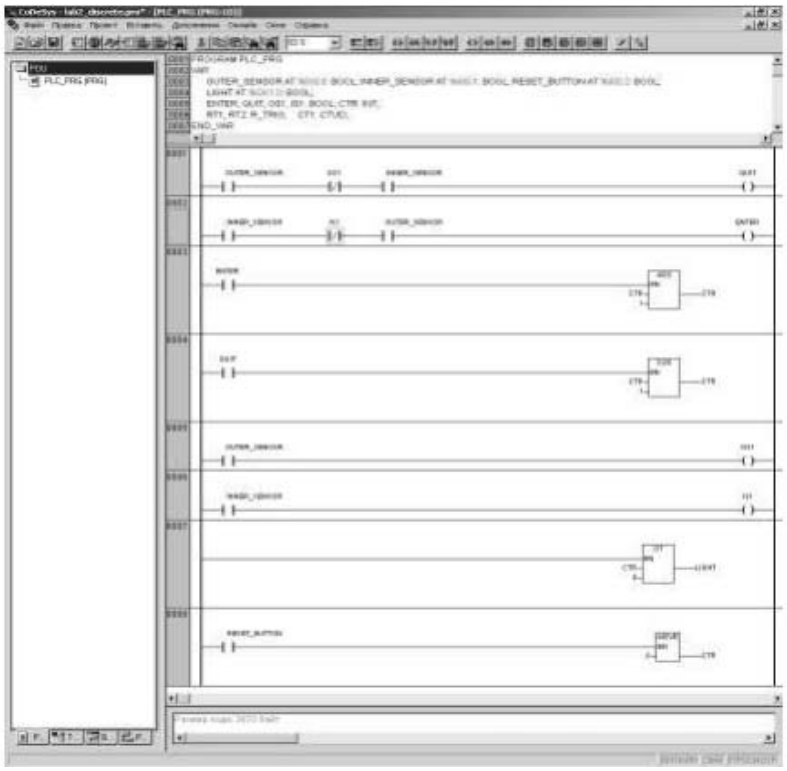


Рисунок 2.9 – Решение задачи об автоматическом включении света без типовых функциональных блоков

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», 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.

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 истечение 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.

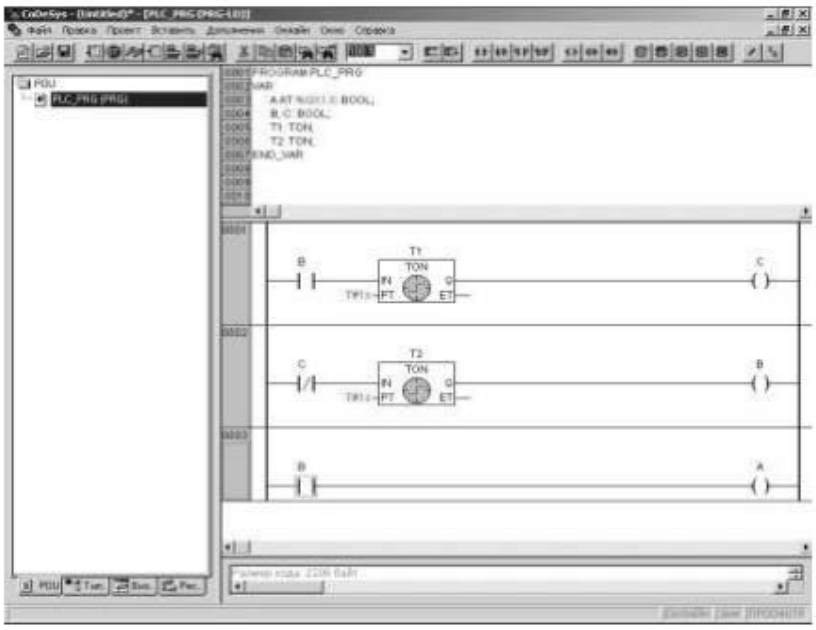


Рисунок 2.10 – Программа генератора прямоугольных импульсов на языке LD

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, т. к. установка и сброс переменной C происходит с интервалом времени в один цикл, а установка и сброс переменной B – 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 CoDeSy.

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 студентами теоретических сведений 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 $T_{уст}$.

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

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

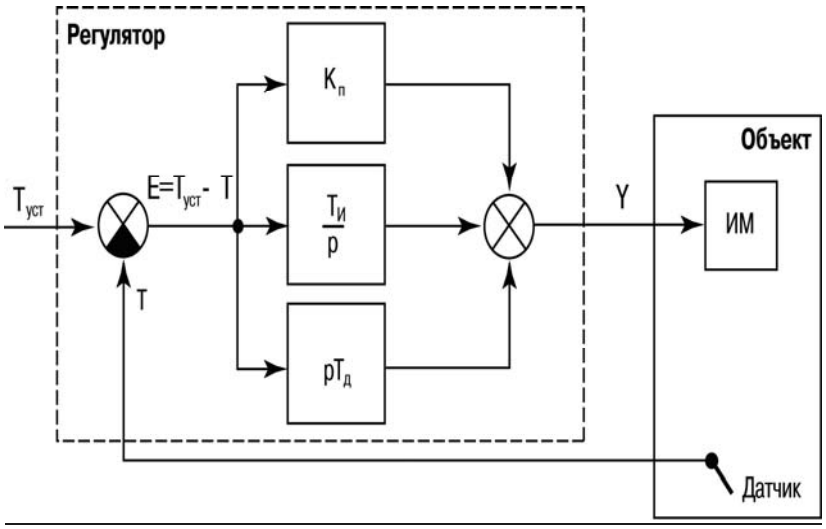


Рисунок 3.1 – Схема ПИД-регулятора

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.

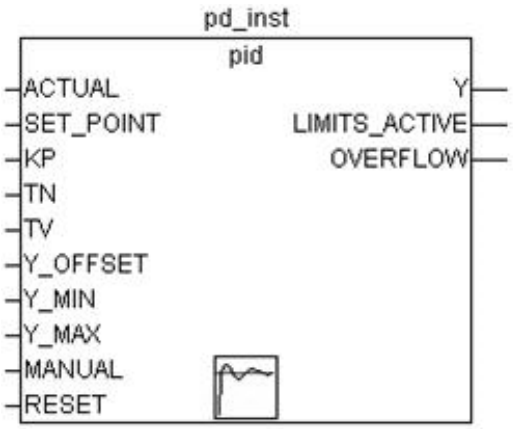


Рисунок 3.2 – ПИД-регулятор в CoDeSys

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, мс;

TV – постоянная дифференцирования, мс;

e(t) – сигнал ошибки (SET_POINT-ACTUAL).

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 нулю.

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 ПИД- и двухпозиционного 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.

```
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

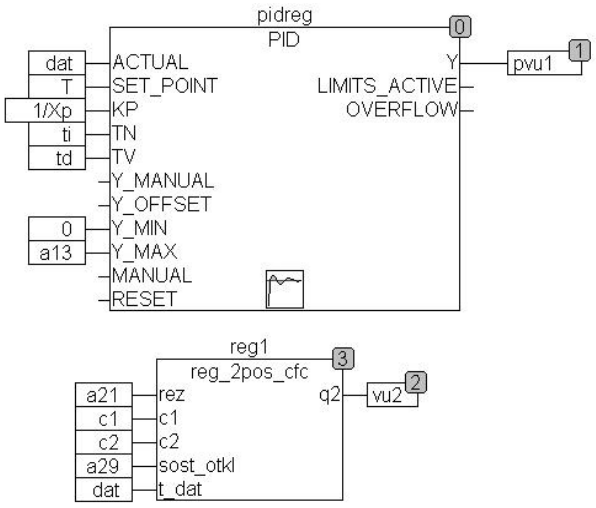


Рисунок 3.4 – Программа регулятора в CoDeSys

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.

```
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

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

```
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

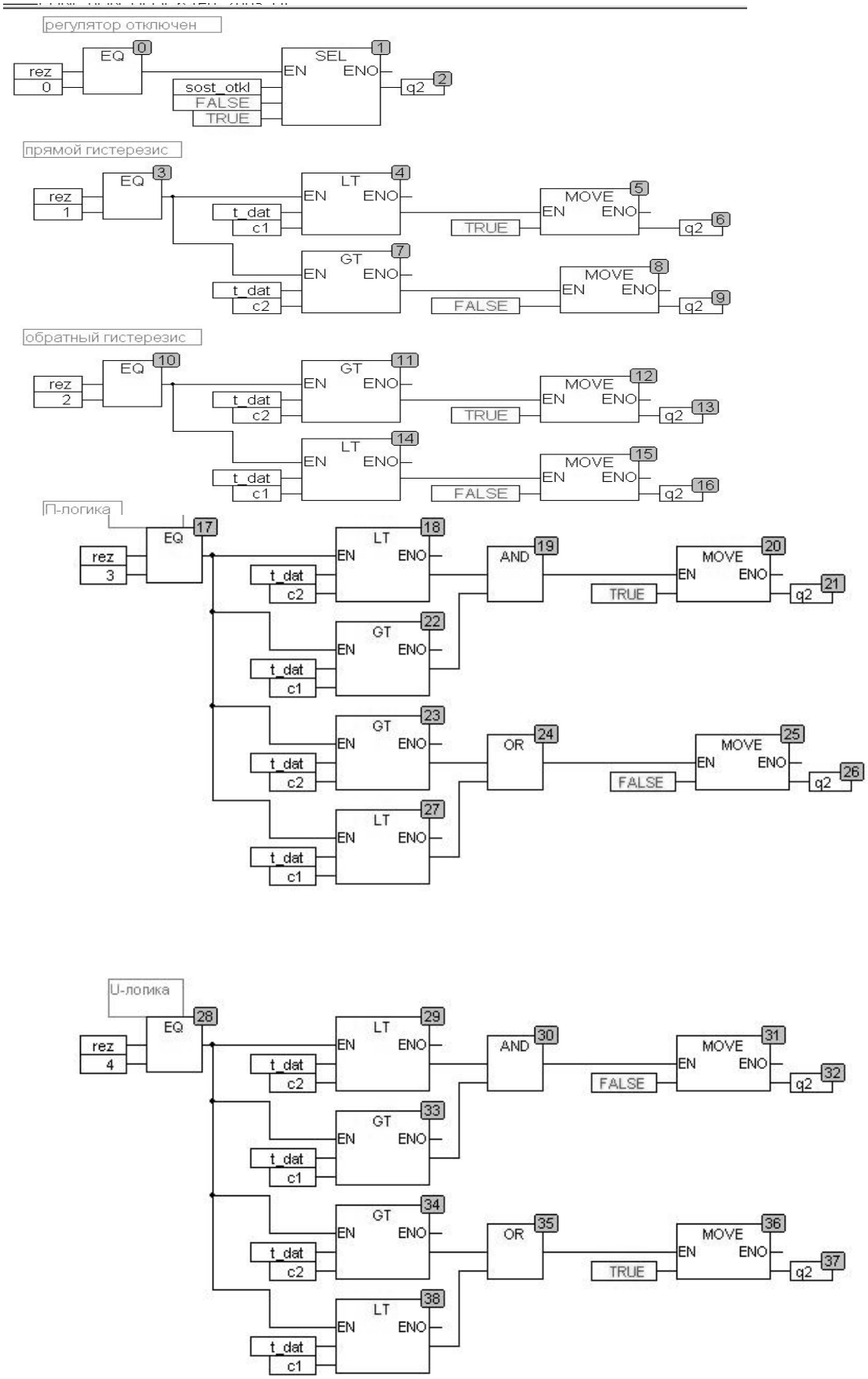


Рисунок 3.7 – Подпрограмма двухпозиционного регулятора на языке CFC

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.

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.

Список литературы

1 **Волков, М. А.** Управление техническими и технологическими системами : учебное пособие / М. А. Волков, А. Ю. Постыляков, Д. В. Исаков. – Москва ; Вологда : Инфра-Инженерия, 2022. – 252 с.

2 **Федоров, Ю. Н.** Справочник инженера по АСУТП: проектирование и разработка : учебное пособие / Ю. Н. Федоров. – 3-е изд., стер. – Москва; Вологда : Инфра-Инженерия, 2022. – 928 с.

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

4 Общие сведения о CoDeSys [Электронный ресурс]. – Режим доступа: http://www.3s-software.ru /publications. – Дата доступа: 16.05.2017.

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