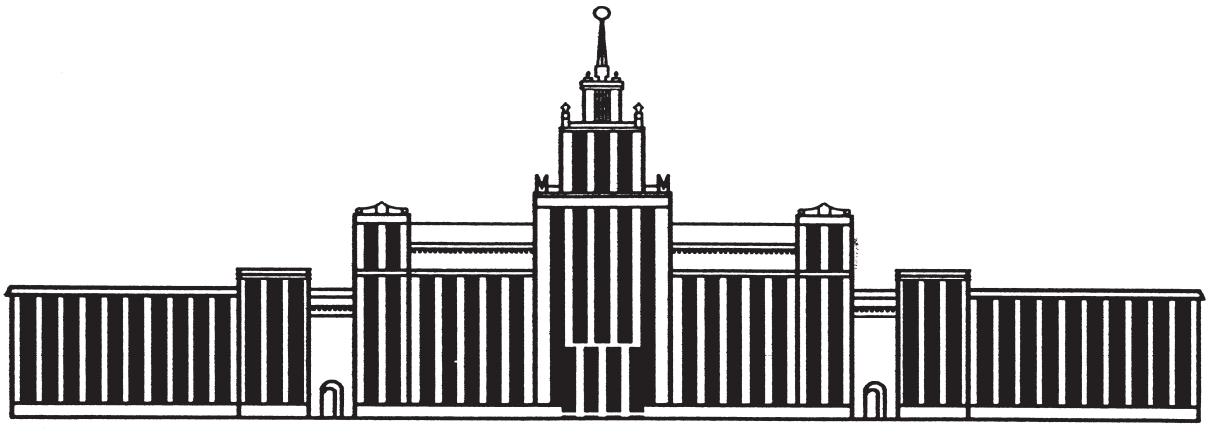
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

681.5(07)
К143

Казаринов Л.С., Канашев Е.А.

ПРОГРАММИРОВАНИЕ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Учебное пособие

Челябинск
2018

Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра «Автоматика и управление»

681.5(07)
К143

Казаринов Л.С., Канашев Е.А.

ПРОГРАММИРОВАНИЕ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Учебное пособие

Челябинск
Издательский центр ЮУрГУ
2018

УДК 681.518.3(075.8)
ББК У9(2)441.2.я7
К143

Одобрено
учебно-методической комиссией
высшей школы электроники и компьютерных наук

Рецензенты:
А.Е. Гудилин, О.С. Логунова

Казаринов, Л.С.

К143 Программирование систем реального времени: учебное пособие / Л.С. Казаринов, Е.А. Канашев. – Челябинск: Издательский центр ЮУрГУ, 2018. – 40 с.

В пособии представлены задания и методические указания по выполнению лабораторных работ, ориентированные на реализацию программного обеспечения систем управления, работающих в режиме реального времени.

Учебное пособие предназначено для студентов очной формы обучения специальности «Управление и информатика в технических системах», обучающихся по программе бакалавриата на основе проектного подхода для высшей инженерной школы (элитной подготовки).

УДК 681.518.3 (075.8)
ББК У9(2)441.2.я7

© Издательский центр ЮУрГУ, 2018

ОГЛАВЛЕНИЕ

1. Задания для практических работ (программирование ПЛК SIMATIC S7)
Задание 1. Создание простого логического устройства 4
Задание 2. Синтез логического устройства, заданного таблицей истинности, на языке LAD 4
Задание 3. Синтез логического устройства, заданного таблицей истинности, на языке FBD 5
Задание 4. Синтез управляющего устройства 6
Задание 5. Лампы и кнопки 7
Задание 6. Печь сопротивлений 7
Задание 7. Штамповочный пресс 8
Задание 8. Генератор импульсов 8
Задание 9. Бегущий огонь 8
Задание 10. Генератор импульсов №2 8
Задание 11. Бегущий огонь с реверсом 9
Задание 12. Бегущий огонь с накоплением 9
Задание 13. Светофор 9
Задание 14. Реализация системы управления технологическим процессом 9
Задание 15. Управления шаговым двигателем 12
2. Методические указания
Общие указания к заданиям 1–15 13
Особые указания к заданиям 5–8 22
Особые указания к заданиям 9–13 23
Дополнительные указания к заданию 13 27
Особые указания к заданиям 14–15 27
Пример программирования контроллера SIMATIC S7–1200 32
Дополнительные указания к заданию 15 35
Методические указания по работе с симулятором ПЛК Siemens 36
Указания по загрузке программы в контроллер и запуску программы 38

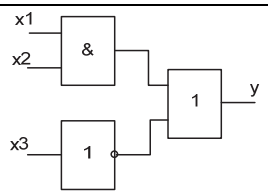
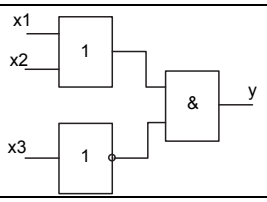
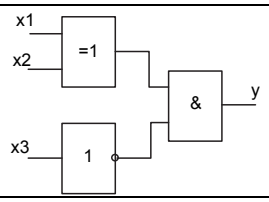
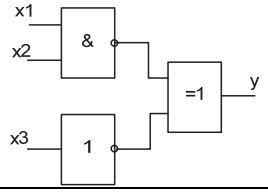
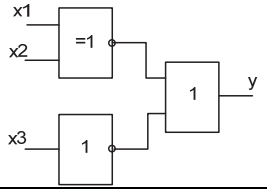
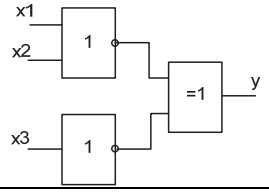
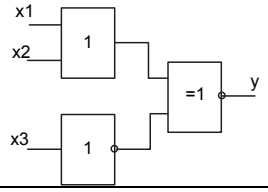
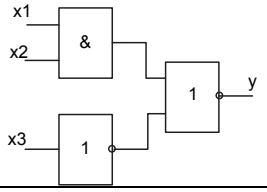
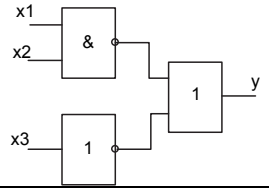
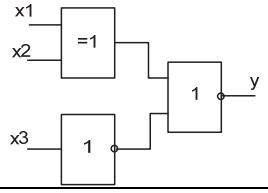
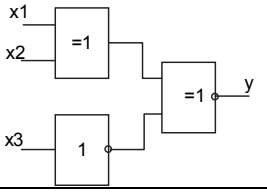
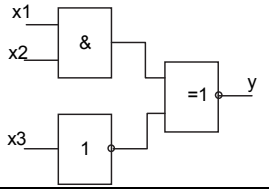
1. ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ РАБОТ (ПРОГРАММИРОВАНИЕ ПЛК SIMATIC S7)

Задание 1. Создание простого логического устройства

Напишите программу для контроллера SIMATIC S7, реализующую логическую функцию в соответствии с вариантом, указанным в табл. 1.

Таблица 1

Варианты заданий

Вариант	1	2	3
Схема			
Вариант	4	5	6
Схема			
Вариант	7	8	9
Схема			
Вариант	10	11	12
Схема			

Задание 2. Синтез логического устройства, заданного таблицей истинности, на языке LAD

Реализовать на языке LAD логическое устройство, заданное таблицей истинности. Входные сигналы указаны в табл. 2, а выходные представлены в табл. 3. Входные сигналы для всех вариантов одинаковые.

Таблица 2

Входные сигналы

X1	X2	X3	X4	X1	X2	X3	X4	X1	X2	X3	X4	X1	X2	X3	X4
0	0	0	0	0	0	1	1	1	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0	1	0	0	1	1	1	0	1
0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	0
0	1	1	1	0	1	1	0	1	0	1	1	1	1	1	1

Таблица 3

Выходные сигналы

Вариант																	
	1	Y1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0
		Y2	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1
	2	Y1	1	0	0	1	0	0	0	1	0	0	0	1	0	1	1	0
		Y2	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0
	3	Y1	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0
		Y2	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
	4	Y1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	1	1
		Y2	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1
	5	Y1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1
		Y2	1	1	0	0	0	0	0	0	1	0	1	0	1	1	1	0
	6	Y1	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1
		Y2	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0
	7	Y1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1
		Y2	0	0	1	1	0	1	1	0	1	0	0	1	0	0	0	1
	8	Y1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
		Y2	1	1	0	0	0	0	1	1	0	0	0	1	1	1	1	1
	9	Y1	0	1	1	0	1	1	0	0	1	0	1	0	0	0	0	0
		Y2	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1
	10	Y1	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1
		Y2	0	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1
	11	Y1	0	0	1	1	1	0	1	1	0	1	0	0	1	1	0	1
		Y2	1	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
	12	Y1	0	0	1	1	0	1	0	1	0	0	1	0	0	1	1	1
		Y2	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	1

Задание 3. Синтез логического устройства, заданного таблицей истинности, на языке FBD

Реализовать на языке FBD логическое устройство, заданное таблицей истинности. Входные сигналы аналогичны входным сигналам предыдущего задания (см. табл. 2), а выходные представлены в табл. 4.

Таблица 4

Выходные сигналы

Вариант																	
	1	Y1	0	0	1	1	0	1	0	1	0	0	1	0	0	1	1	1
		Y2	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	1
	2	Y1	1	0	0	1	0	0	0	1	0	0	0	1	0	1	1	0
		Y2	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0
	3	Y1	0	1	1	0	1	1	0	0	1	0	1	0	0	0	0	0
		Y2	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1
	4	Y1	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1
		Y2	0	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1
	5	Y1	0	0	0	1	0	0	0	1	0	1	1	1	0	0	1	1
		Y2	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1
	6	Y1	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	1
		Y2	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0
	7	Y1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1
		Y2	0	0	1	1	0	1	1	0	1	0	0	1	0	0	0	1
	8	Y1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
		Y2	1	1	0	0	0	0	1	1	0	0	0	1	1	1	1	1
	9	Y1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	0
		Y2	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1
	10	Y1	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0
		Y2	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
	11	Y1	0	0	1	1	1	0	1	1	0	1	0	0	1	1	0	1
		Y2	1	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
	12	Y1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1
		Y2	1	1	0	0	0	0	0	0	1	0	1	0	1	1	1	0

Задание 4. Синтез управляющего устройства

Используя язык релейно-контактных схем напишите программы для управления каждым из представленных технологических объектов.

Объект №1. Автоматизировать линию сортировки изделий. По конвейеру движутся низкие изделия, но редко могут встречаться и высокие. По ходу конвейера установлены две фотоголовки. Первая по ходу конвейера фотоголовка настроена на низкие изделия (нижняя фотоголовка), вторая – на высокие изделия (верхняя фотоголовка). При прохождении низкой детали перекрывается только нижняя фотоголовка, а при высоких – вначале нижняя, а затем верхняя фотоголовки. На выходе конвейера детали ударяются о сортировочную планку, и высокие изделия поступают в правый накопитель (левое положение планки), низкие – в левый (правое положение планки). Планка перебрасывается пневмоприводом. Для переброски планки влево или

вправо достаточно хотя бы кратковременно подать команду на ее перемещение влево или вправо. Включение системы в работу осуществляется нажатием кнопки "Пуск".

Объект № 2. Организовать управление тремя лампами в зависимости от нажатий управляющей кнопки следующим образом: первая лампа включается и выключается через одно нажатие кнопки, вторая – через каждые два нажатия кнопки, третья – через каждые три нажатия кнопки. Алгоритм реализовать, не используя счетчики.

Объект № 3. Реализовать управление двумя вентиляторами. Для управления первым вентилятором служит кнопка «Пуск/Стоп» – при первом нажатии вентилятор включается, при последующем – выключается. Для управления вторым вентилятором служит кнопка «Дубль» – при её нажатии вентилятор копирует режим первого (включен или выключен).

Объект № 4. Автоматизировать освещение складских помещений, состоящих из пяти комнат. В каждой комнате установлен датчик движения, выдающий дискретный сигнал при обнаружении движения в комнате. Осветители установлены в каждой комнате, каждый осветитель управляется своим каналом управления. Принцип управления следующий: свет в комнате включается при поступлении сигнала с соответствующего датчика движения и при включенном свете в одной из соседней комнат. Для крайних комнат последнее условие не обязательно.

Задание 5. Лампы и кнопки

Организовать управление тремя лампами в зависимости от нажатий управляющей кнопки следующим образом: первая лампа включается и выключается через одно нажатие кнопки, вторая – через каждые два нажатия кнопки, третья – через каждые три нажатия кнопки. По возможности организовать различные способы управления ламп.

Задание 6. Печь сопротивлений

Печь сопротивлений содержит три нагревательных элемента (НЭ), каждый из которых питается от собственного тиристорного преобразователя соответственно ТП1…ТП3. Сигнал "1" на входе ТП соответствует максимально допустимому току НЭ, сигнал "0" – минимально допустимому току НЭ. Синтезировать схему, обеспечивающую алгоритм работы группы ТП, представленный в табл. 5.

Таблица 5

Алгоритм работы группы ТП

Номер импульса	–	1	2	3	4	5	6	7	8	9	…
Состояние ТП1	0	1	0	0	1	1	0	1	0	0	…
Состояние ТП2	0	0	1	0	0	1	1	0	1	0	
Состояние ТП3	0	0	0	1	1	0	1	0	0	1	

Начало работы осуществляется после нажатия на кнопку "Пуск". Цикл управления печью сопротивления состоит из 6 тактов. Переход с одного такта на другой осуществляется при поступлении очередного импульса от управляющего сигнала. Циклы управления непрерывно следуют друг за другом.

Задание 7. Штамповочный пресс

Автоматизировать перемещение пуансона штамповочного пресса. При поступлении управляющего сигнала пуансон из крайнего верхнего положения П0 движется вниз, при своем движении до положения П1 входит в матрицу, выбивая из ленты деталь необходимой формы, и возвращается в положение П0. Чисто механическим устройством лента перемещается на один шаг и после нового управляющего сигнала пуансон совершает очередное движение. Лента рассчитана на изготовление 10 деталей, поэтому через 10 ходов пуансона формируется сигнал "конец цикла".

Задание 8. Генератор импульсов

В контроллере программным путем необходимо реализовать генератор импульсов. Время наличия импульса 1 с, время его отсутствия 2 с. При нажатии на кнопку "Пуск" начинается счет импульсов. При прохождении 10 импульсов загорается лампа Л1, при прохождении следующих 10 импульсов – лампа Л2, аналогично включаются лампы Л3 и Л4. После загорания лампы Л4 (т.е. после прохождения 40 импульсов) счет прекращается и все лампы горят до нажатия на кнопку "Стоп". Генератор импульсов реализовать программным путем.

Задание 9. Бегущий огонь

Организовать бегущий огонь на 8 выходах. По умолчанию время горения каждой лампы – по 1 секунде. При нажатии кнопки «Быстро» бегущий огонь убыстряется в 2 раза. По нажатию кнопки «Реверс» происходит реверс бегущего огня в противоположную сторону.

Задание 10. Генератор импульсов №2

В контроллере программным путем необходимо реализовать два генератора импульсов с использованием таймеров. Время наличия первого импульса 1 с, время его отсутствия 2 с. Во время отсутствия первого импульса работает второй генератор с частотой 2 Гц и скважностью 0,5. При нажатии на кнопку "Пуск" начинается генерация импульсов. По нажатию кнопки «Стоп» генераторы начинают вырабатывать импульсы синхронно частотой 0,5 Гц.

Задание 11. Бегущий огонь с реверсом

Организовать бегущий огонь на всех имеющихся выходах ПЛК (14). По умолчанию время горения каждой лампы – по 1 секунде. Огонь «добежав» до последнего выхода «бежит» обратно до первого. Далее начинается новый цикл.

Задание 12. Бегущий огонь с накоплением

Организовать бегущий огонь на 8 выходах с накоплением, т.е. «добежав» до последнего выхода, «огонь бежит» обратно, но последний выход остается гореть. «Добежав» до первого выхода, «огонь бежит» к 7 выходу, а первый выход остается гореть. Таким образом, будет происходить зажигание светодиодов с краев к центру, до момента пока все светодиоды не будут гореть.

Задание 13. Светофор

Реализовать управление «светофором». Сначала горит красный цвет, затем через 5 секунд зажигается еще и желтый. Через 3 секунды красный с желтым тухнут, загорается зеленый цвет. Зеленый горит 5 секунд, затем начинает моргать с частотой 1 Гц в течение 3 секунд. Далее начинается новый цикл. Светофоры работают в противофазе.

Задание 14. Реализация системы управления технологическим процессом

Используя язык релейно-контактных схем напишите программы для управления каждым из представленных технологических объектов.

Объект №1. Перекладыватель служит для перемещения листов со стола на постоянно вращающийся рольганг. Перекладыватель имеет подъемник листов, установленный на тележке. Подъемник представляет собой поперечину, перемещающуюся по вертикальным стойкам сверху вниз и обратно. Поперечина имеет на конце электромагнит для притягивания листов.В исходном положении перекладыватель стоит в положении П0 над столом, а его подъемник в крайнем верхнем положении КВ. При поступлении листа на стол подается сигнал наличия листа, подъемник включается для движения вниз. При касании листа подъемник останавливается и включается электромагнит. Через 2 с включается подъемник на подъем, достигает положения КВ и останавливается. Включается тележка и перемещается в положение П1 над рольгангом, где останавливается и электромагнит отключается. Лист падает на рольганг. Через 2 с тележка движется в положение П0. Цикл повторяется при поступлении очередного сигнала наличия листа. Включение системы в работу – нажатием кнопки «Пуск».

Объект №2. В контроллере программным путем необходимо реализовать генератор импульсов. Время наличия импульса 1 с, время его отсутствия 2 с. При нажатии на кнопку «Пуск» начинается счет импульсов. При прохождении 10 импульсов загорается лампа Л1, при прохождении следующих 10 импульсов

– лампа Л2, аналогично включаются лампы Л3 и Л4. После загорания лампы Л4 (т.е. после прохождения 40 импульсов) счет прекращается и все лампы горят до нажатия на кнопку «Стоп». Генератор импульсов реализовать программным путем.

Объект №3. Автоматизировать перемещение пунсона штампованного пресса. Рабочий заправляет металлическую ленту (заготовку) и нажимает кнопку «Пуск». Пуансон из крайнего верхнего положения П0 движется вниз, при своем движении до положения П1 входит в матрицу, выбивая из ленты деталь необходимой формы, и возвращается в положение П0. Чисто механическим устройством лента перемещается на один шаг и через 2 с пуансон совершает очередное движение. Лента рассчитана на изготовление 10 деталей, поэтому через 10 ходов пуансона формируется сигнал «Конец цикла». При подаче питания на систему автоматизации пуансон из любой точки должен прийти в положение П0.

Объект №4. На программируемом контроллере реализовать систему управления толкателем с кривошипно-шатунным механизмом с реверсивным электродвигателем. Толкатель имеет два фиксированных положения П0 и П1. При подаче питания на систему толкатель из любого положения движется вперед медленно до положения П0. При появлении кратковременного сигнала «Пуск» толкатель из положения П0 медленно движется до положения П1, стоит там в течение 2 с, а затем электродвигатель включается для движения назад быстро и возвращается в положение П0, где формируется сигнал «Конец цикла». Цикл повторяется при нажатии кнопки «Пуск». При нажатии кнопки «Стоп» толкатель из любого положения на быстрой скорости возвращается в положение П0.

Объект №5.. Автоматизировать управление крышкой котла (типа большой кастрюли). При кратковременном нажатии кнопки «Открыть» крышка поднимается до крайнего верхнего положения, в котором поворачивается, открывая доступ к содержимому котла. При кратковременном нажатии на кнопку «Закрыть» крышка поворачивается в положение точно над котлом и затем опускается, закрывая котел. Для исключения ударов предусмотреть переход на пониженную скорость перед касанием крышкой котла при его закрывании.

Объект №6. Автоматизировать управление дверью (воротами) въезда/выезда гаража. Нормальное (исходное) положение двери закрытое. При кратковременном нажатии кнопки «Открыть» включается звонок и через 5 с включается привод двери на открывание. При полном открытии двери привод двери отключается сразу, а звонок звенит еще 2 с. Закрывание дверей – аналогично: при кратковременном нажатии кнопки «Закрыть» включается звонок, через 5 с включается привод двери на закрывание. При полном закрытии двери привод двери отключается, а звонок звенит еще 2 с.

Объект №7. Манипулятор (рука со схватом) служит для подачи заготовок из накопителя в пресс. Рука манипулятора может перемещаться вперед/назад из крайнего заднего КЗ в крайнее переднее КП положение и назад, а также

поворачиваться из крайнего правого положения КПР в крайнее левое положение КЛ и обратно. Схват имеет электромагнитный механизм зажима/разжима заготовок. Исходное положение манипулятора перед накопителем (в положениях КЗ и КПр, схват под действием пружин зажат). При поступлении запроса со станка и наличии заготовки в накопителе включается электромагнит разжима схвата. Рука идет в крайнее переднее положение КП, через 2 с схват зажимает заготовку (электромагнит схвата отключается) и рука возвращается в положение КЗ. Рука поворачивается в положение КЛ и выдвигается вперед до КП. Схват разжимается (электромагнит схвата включается) и через 2 с рука идет в положение КЗ, затем поворачивается в положение КПР (исходное положение руки). Если есть запрос со станка, а накопитель пуст, то включается сигнальная лампа «Нет детали» и кратковременный звуковой сигнал (в течение 5 с). Световой сигнал снимается кнопкой «Сброс». Включение системы в работу – нажатием кнопки «Пуск».

Объект №8. Автоматизировать процесс безупорной остановки длинномерного металла на рольганге Р1. На рольганг Р1 металл передается с предыдущего рольганга Р0, наличие металла на котором контролируется датчиком Д0. Вдоль рольганга Р1 расположены датчики, контролирующие наличие металла последовательно в положениях П1, П2 и П3. Нормальное положение рольганга Р1 – неподвижное. При наличии сигнала с датчика Д0 и отсутствии металла на рольганге Р1, рольганг Р1 включается и металл перемещается по рольгангу Р1. При достижении передним концом металла положения П1 происходит снижение скорости Р1 до «ползучей» скорости, а в положении П2 привод рольганга Р1 отключается. Если по каким-либо причинам металл достиг положения П3, то привод рольганга Р1 реверсируется и работает на пониженной скорости, пока металл не уйдет из положения П3. При остановке переднего конца металла между положениями П2 и П3 формируется сигнал разрешения работы механизмов уборки металла с Р1. Включение системы в работу – при нажатии на кнопку «Пуск».

Объект №9. Автоматизировать работу толкателя с кривошипно-шатунным механизмом с нереверсивным электродвигателем. При нажатии кнопки «Пуск» кривошип из положения $\alpha = 0^0$ движется до положения $\alpha = 120^0$. В этом положении происходит переход на пониженную (малую) скорость, кривошип идет до положения $\alpha = 180^0$ и останавливается. Через 3с электродвигатель автоматически вновь включается и на повышенной скорости шатун идет в положение $\alpha = 0^0$. Стоит в нем 3 с, включается и движется к положению $\alpha = 120^0$ и на пониженной скорости до $\alpha = 180^0$ и т.д. После 5 толканий толкатель останавливается в исходном положении (кривошип – в положении $\alpha = 0^0$) и формируется сигнал «Конец цикла». Очередной цикл из пяти толканий – после очередного нажатия на кнопку «Пуск». При нажатии на кнопку «Стоп» толкатель из любого положения на повышенной скорости движется в положение $\alpha = 0^0$ и останавливается.

Объект №10. Тележка движется по кольцевому пути, на котором есть 3 фиксированных положения П0, П1 и П2. Привод тележки – от нереверсивного

электродвигателя. При подаче питания из любого положения тележка должна идти в положение П0. При нажатии на кнопку «Пуск» тележка без задержки движется в положение П1, стоит там в течение 3 с, затем движется в П2, стоит там в течение 4 с, движется в положение П0, стоит там в течение 5 с, движется в П1, стоит там в течение 3 с и т.д. То есть тележка непрерывно движется по кольцевому пути с остановками в фиксированных положениях. При нажатии на кнопку «Стоп» тележка должна без остановки идти в П0 и там ожидать очередного нажатия кнопки «Пуск».

Задание 15. Управления шаговым двигателем

Используя язык функциональных блоков напишите программы для управления шаговым двигателем (ШД).

Условие 1. При каждом включении первого тумблера скорость двигателя увеличивается, а при каждом включении второго тумблера скорость уменьшается, третий тумблер останавливает двигатель.

Условие 2. Реализовать разгон и торможение ДПТ по рампе. Разгон и торможение по 5 секунд, а поддержание скорости 10 секунд. Моменты начала разгона, выход на установившуюся скорость, начало и окончание торможения индицировать светодиодами.

Условие 3. Используя сигналы с оптических датчиков, реагирующих на прорези в диске, установленном на валу ШД, считать количество сделанных оборотов. По сигналу с первого тумблера ШД делает 5 оборотов в одну сторону, при сигнале со 2-го тумблера 5 оборотов в другую сторону.

Условие 4. Реализовать разгон и торможение ШД по рампе. Разгон и торможение – 5 секунд, поддержание скорости – 10 секунд. Моменты начала разгона, выход на установившуюся скорость, начало и окончание торможения индицировать светодиодами.

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Общие указания к заданиям 1–15

Для подготовки программ при проведении лабораторных работ служит программный пакет *SIMATIC TIA PORTAL*. В частности для программирования контроллеров используется программное обеспечение *STEP*7 *Basic*, встроенное в SIMATIC TIA PORTAL. Для запуска программы необходимо на рабочем столе *Windows* дважды щелкнуть курсором мыши по соответствующему ярлыку. После чего откроется мастер *SIMATIC TIA PORTAL* (рис. 1), с помощью которого можно создать новый проект или открыть уже созданный проект. Всю рабочую область программы можно условно разделить на три части:

1) левая – главное окно проекта;
2) центральная – выбор действия;
3) правая – рабочая область.

В зависимости от состояния проекта те или иные пункты будут активны (подсвечены) или нет. Структура проекта используется для надлежащего хранения и размещения всех данных и программ. Для создания нового проекта необходимо нажать «*Create new project*».

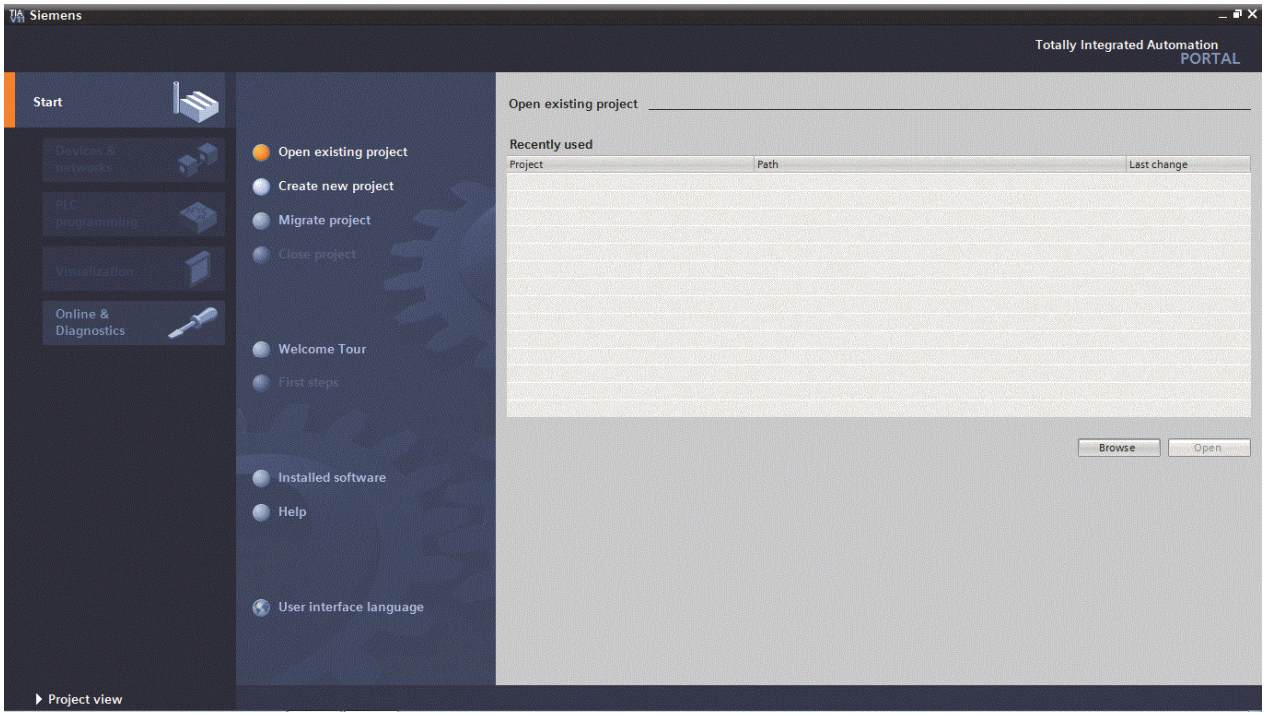


Рис. 1. Стартовое окно *TIA PORTAL*

В открывшемся диалоговом окне можно выбрать название проекта, его расположение, указать автора и сделать какие-то комментарии (рис. 2), после чего необходимо нажать «*Create*».

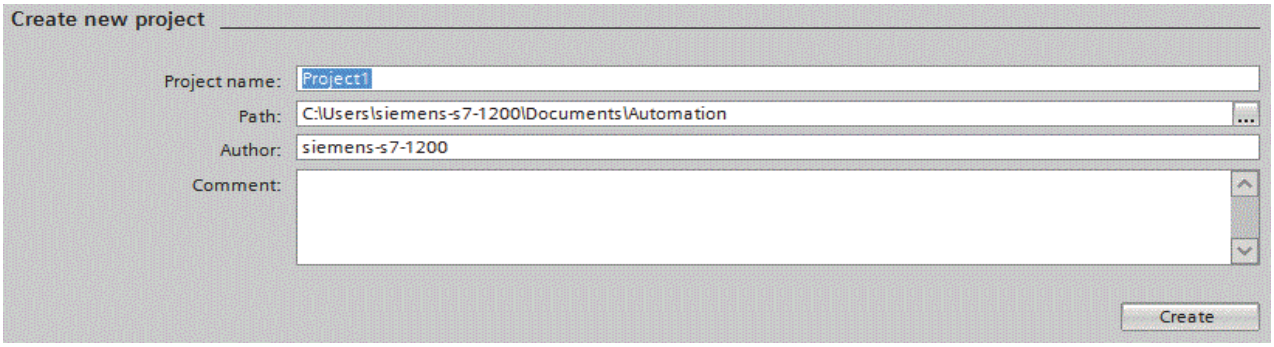


Рис. 2. Окно редактирования свойств проекта

После этого откроется следующее окно (рис. 3). Здесь в главном окне проекта необходимо выбрать «*Device & networks*» («Устройства и сети»).

После чего в появившемся окне выбора действия необходимо выбрать указатель «*Add new device*» («Добавить новое устройство»). Рабочая область проекта изменит свой вид. В этом окне необходимо выбрать иконку «*PLC*», далее раскрыть закладку «*CPU* 1214 *DC\DC\DC*» и там дважды щелкнуть левой клавишей мыши по коду устройства 6*ES*7 1214-1*AG*31-0*XB*0 (рис. 4). После этого окно проекта примет следующий вид (рис. 5).

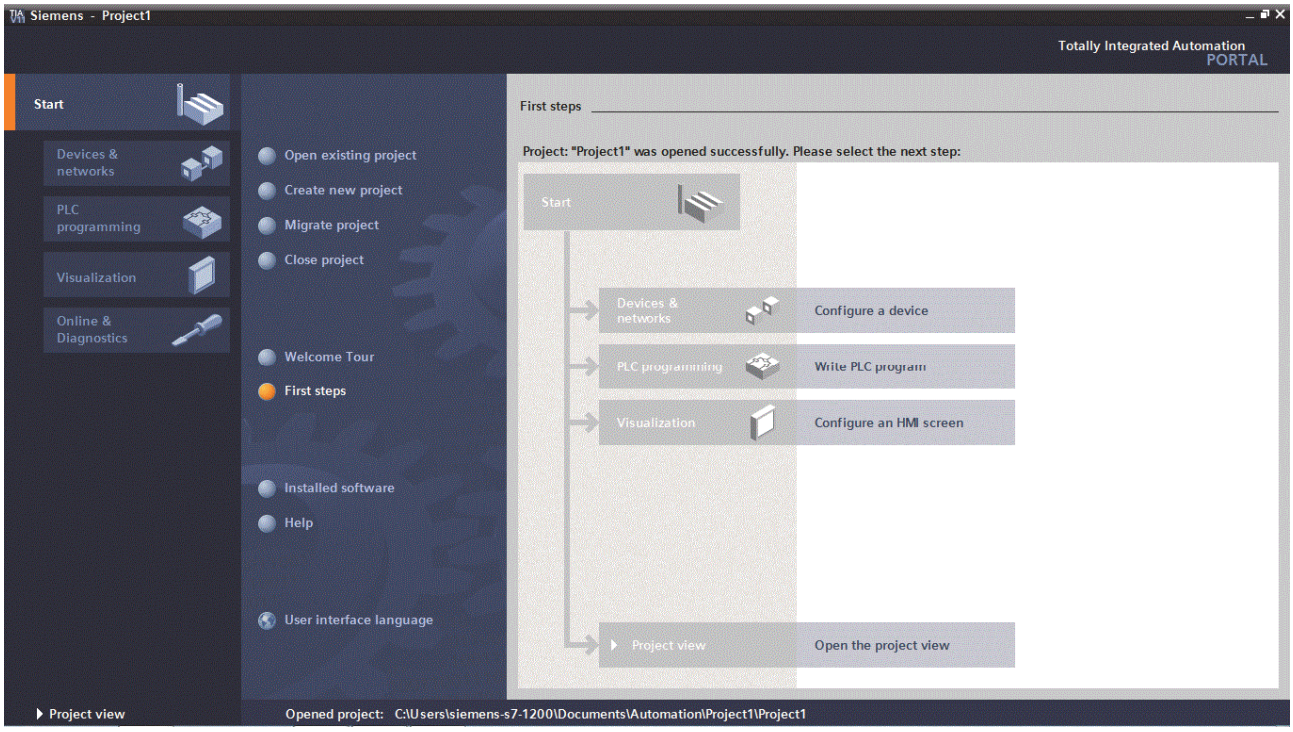


Рис. 3. Окно выбора дальнейших действий с проектом

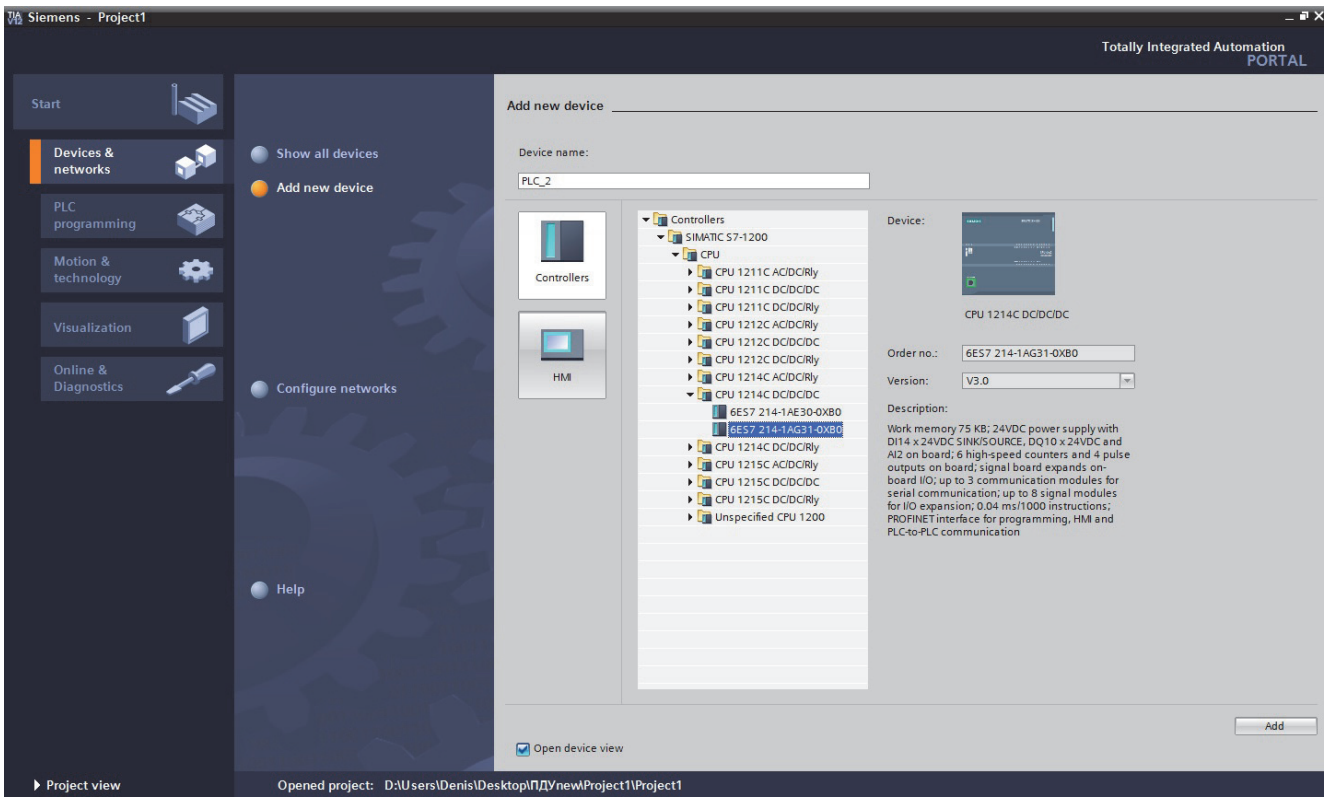


Рис. 4. Окно выбора нового оборудования

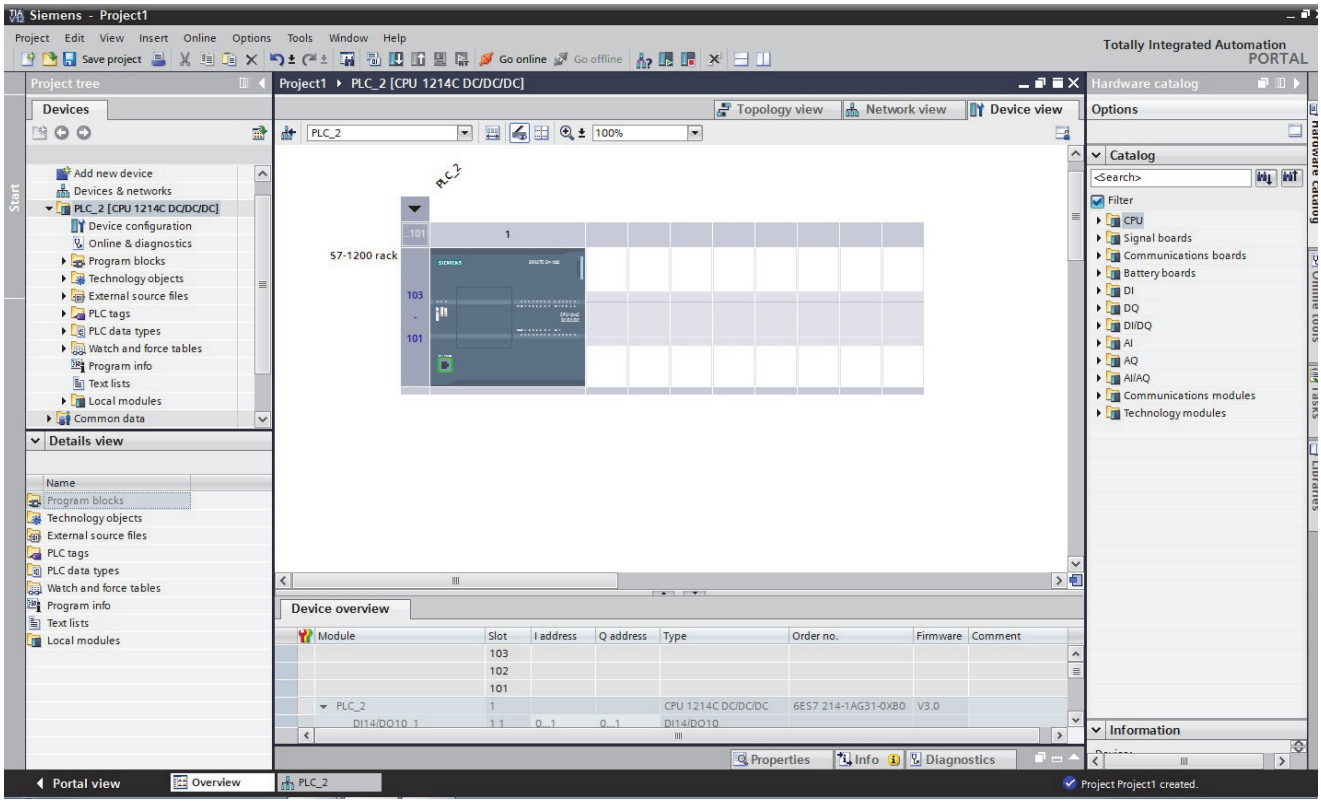


Рис. 5. Основное окно проекта

Для конфигурирования контроллера необходимо в дереве проекта выделить пункт «*PLC*_1 [*CPU* 1214 *DC\DC\DC*]», далее в меню «*Online*» выбрать пункт «*Accessible devices*». При подключенном к сети *PROFINET* контроллере

начнется процесс детектирования оборудования, о чем будет сигнализировать открывшееся окно, которое после поиска сетевых устройств примет следующий вид (рис. 6).

Необходимо отметить, что контроллер уже имеет заранее заданный IP-адрес, который будет использоваться в дальнейшем. При необходимости изменения IP-адреса необходимо воспользоваться специальным программным обеспечением (Primary Setup Tool или S7–1200 Tool).

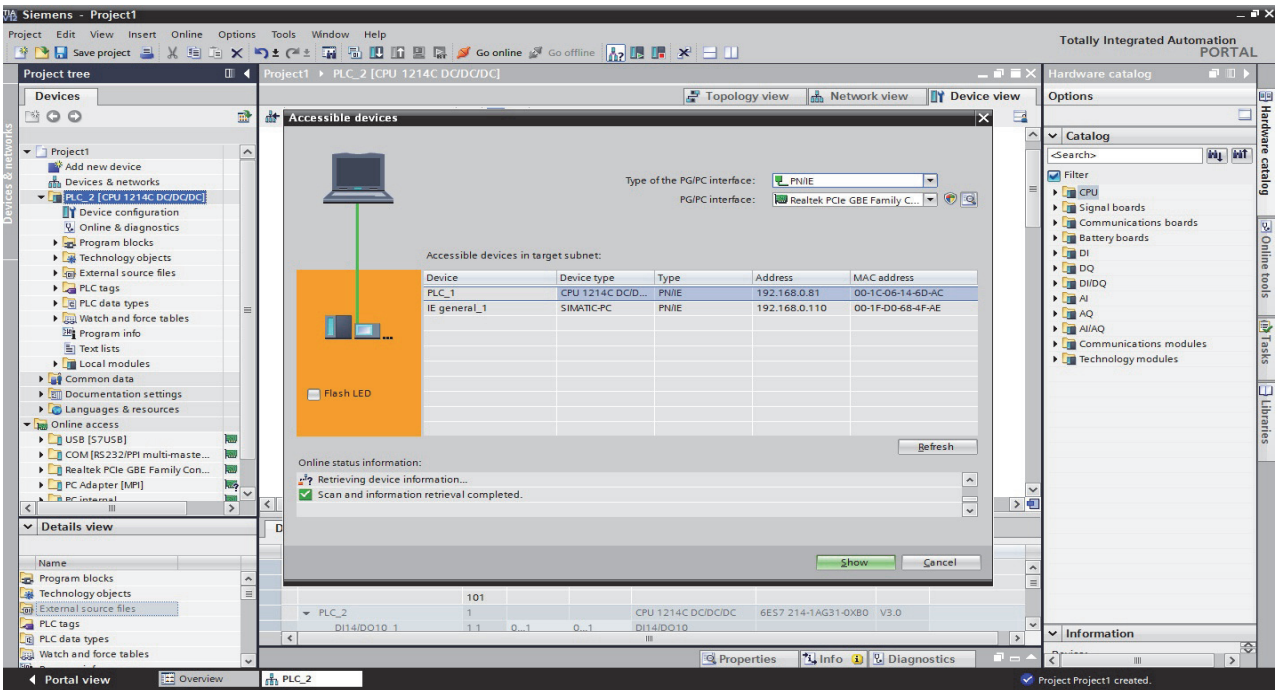


Рис. 6. Поиск сетевых устройств

Необходимо открыть «*Device configuration*» и зайти в настройки «*Ethernet addresses*» (рис. 7). Далее нужно внести изменения в адрес контроллера в проекте соответствующий реальному адресу контроллера.

После конфигурирования необходимо в соответствии с поставленной задачей разработать алгоритм автоматизации управления объектом либо в виде логических уравнений, либо в виде схемы алгоритма. Входным, выходным сигналам и внутренним переменным (меркерам) присваиваются проектировщиком адреса в соответствии с конфигурацией контроллера.

Основная программа формируется в организационном блоке *Main*. Для доступа к Main необходимо в дереве проекта в ветке *PLC*_1 выбрать пункт *Program blocks*, в котором выбрать *Main* [*OB*1], дважды щелкнув по нему правой клавишей мыши. После этого открывается окно программирования (рис. 8).

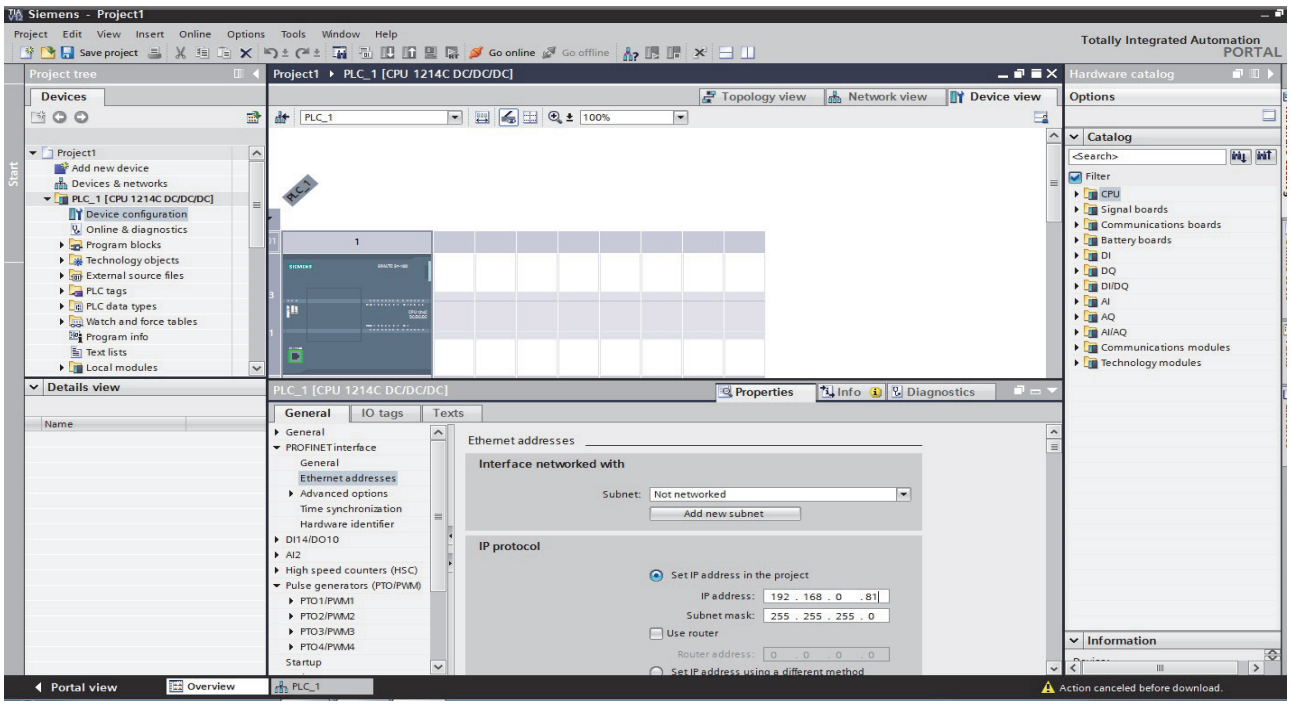


Рис. 7. Настройка IP адреса

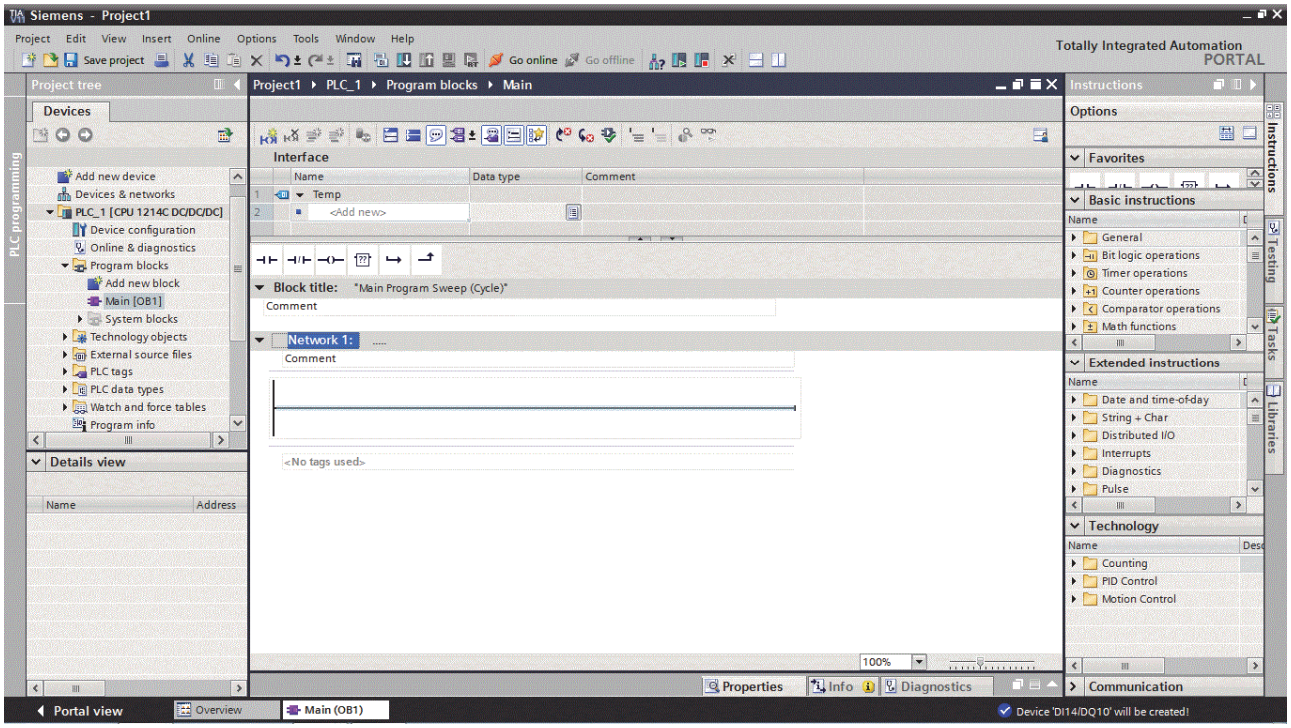


Рис. 8. Окно программирования

Программирование контроллера осуществляется на одном из трех языков программирования: *LAD/STL/FBD*. Язык *LAD* (язык лестничных диаграмм) удобен для инженеров-электриков, язык *STL* (язык операторов) ориентирован для пользователей из мира компьютерных технологий (мнемокода) и, наконец, язык *FBD* (язык функциональных блоков) ориентирован на инженеров-схемотехников. ПО *STEP7 Basic* позволяет в любое время изменить язык программирования. Для этого необходимо выбрать нужной программный блок

OB и, нажав правой клавишей мышки выбрать пункт «*Switch programming language*», в котором необходимо выбрать желаемый язык программирования.

Ниже рассматривается пример программирования на языке *LAD*. При программировании лестничными диаграммами (*LAD*) в виде релейно-контактной схемы программа разделяется на сегменты. Каждый сегмент представляет собой отдельную цепь, по которой может протекать ток. Шина питания находится слева (вертикальная линия).

По умолчанию в блоке *Main* уже есть пустой шаблон сегмента. При начале работы в нем автоматически создается второй сегмент. Таким образом, нет необходимости специально создавать новый сегмент. Для принудительного создания нового сегмента на панели инструментов есть соответствующая кнопка –

Рассмотрим ввод программы для реализации простого уравнения:

$$Q0.0 = \left(I0.1 \cdot \overline{I0.3}\right) + I0.2$$

1) Установить в токовую цепь (сегмент) реле (катушку) (рис. 9). Для этого нажать на кнопку – *Coil* [Катушка]

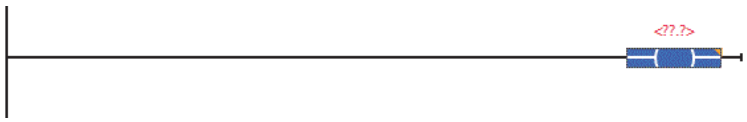


Рис. 9. Катушка

2) Щёлкнуть по тому месту (непосредственно по линии), куда нужно установить замыкающий (нормально открытый) контакт (рис. 10).

Рис. 10. Выбор места установки

3) Вставить замыкающий контакт, нажав на кнопку – *Normally Open Contact* [Нормально открытый контакт] (рис. 11).

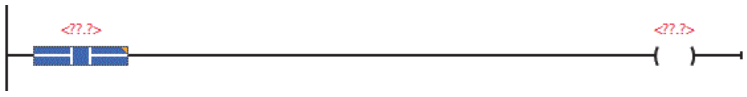


Рис. 11. Нормально открытый контакт

4) Создать параллельную ветвь для замыкающего контакта. Для этого щелкнуть по месту, от которого будет ответвление (в нашем случае – шина питания) и нажать на кнопку – *Open Branch* [Создать ветвление] (рис. 12).

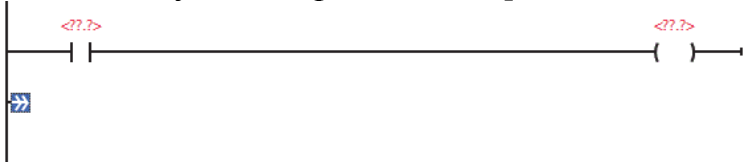


Рис. 12. Параллельная ветвь

5) Вставить нормально открытый контакт и завершите параллельную ветвь, нажав на кнопку – *Close Branch* [Завершить ветвление] (рис. 13).

Рис. 13. Завершение ветви

6) Выделить место, куда необходимо поставить нормально замкнутый (размыкающий) контакт (рис. 14).

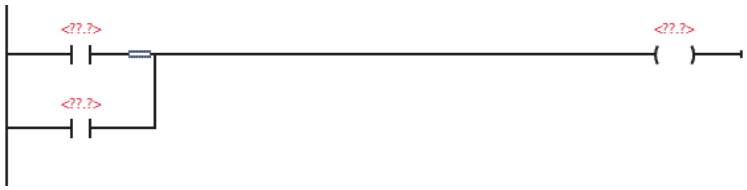


Рис. 14. Выбор места

7) Вставить размыкающий контакт, нажав на кнопку – *Normally Closed Contact* [Нормально закрытый контакт] (рис. 15).

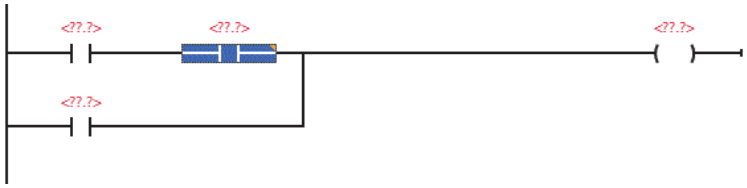


Рис. 15. Установка нормально закрытого контакта

8) Теперь написать адреса контактов. Для этого щёлкнуть по знакам ??.? над контактом и ввести адрес (ввод адреса завершается нажатием клавиши *ENTER*) (рис. 16).

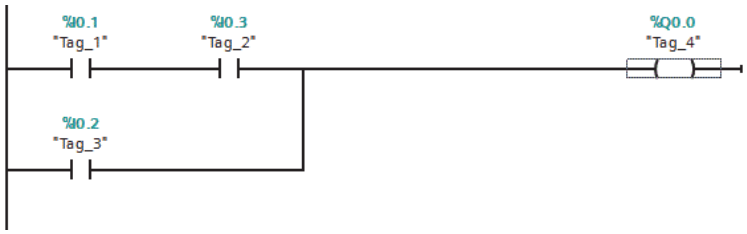


Рис. 16. Адреса контактов

В среде *STEP*7 удобно пользоваться символической адресацией, когда любому адресу может быть присвоено удобное для пользователя символическое имя (тэг).

Для открытия редактора символов необходимо в дереве проекта выбрать пункт «*PLC tags*» и в нем выбрать закладку «*Show all tags*». В открывшемся окне можно создавать новые и редактировать уже существующие тэги.

По умолчанию при вводе адресов контактов и катушек присваиваются тэги по умолчанию от «*Tag*_1» и т.д. (рис. 17).

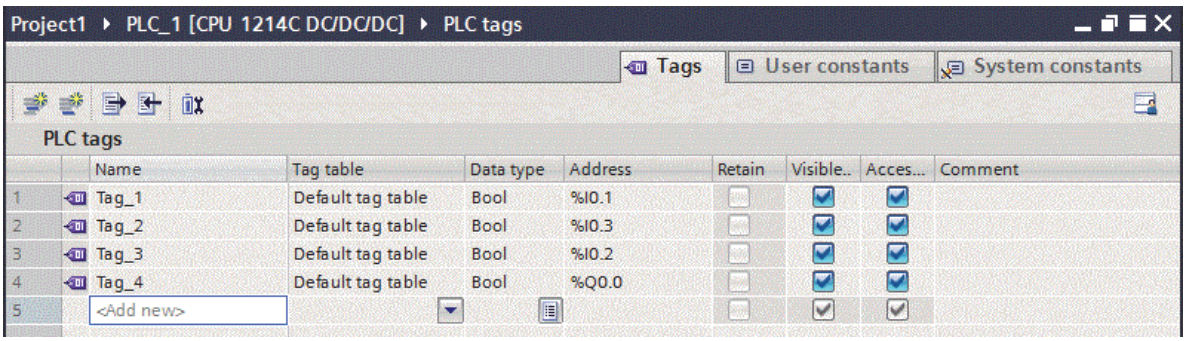


Рис. 17. Окно редактора символов

Далее необходимо переименовать *I*0.1 – из «*Tag*_1» в «*Key*1 [Кнопка 1]», *I*0.3 – из «*Tag*_2» в «*Key*2 [Кнопка 2]», *I*0.2 – из «*Tag*_3» в «*Switch* [Тумблер]», *Q*0.0 – из «*Tag*_4» в «*Lamp* [Лампа]».

Для этого необходимо в столбце символов (*Symbol*) вместо «*Tag*_1» вставить «*Key*1». Для этого необходимо выполнить следующие операции:

– в столбце символов (*Symbol*) вставить «*Key*1»;

– в строке адресов (*Address*) вставить «*I*1.0» (автоматически в столбце типа данных появится *BOOL* – тип данных «бит»;

– в столбце для комментариев (*Comment*) вставить «Кнопка1».

Для остальных адресов символы вводятся аналогично (рис. 18).

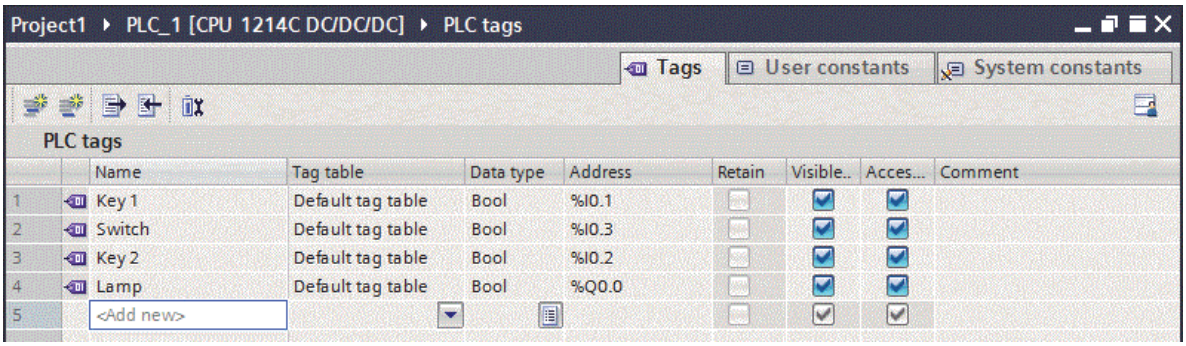


Рис. 18. Окно редактора символов с исправленными символами

Далее необходимо закрыть окно *PLC tags*. Теперь программа будет выглядеть следующим образом (рис. 19).

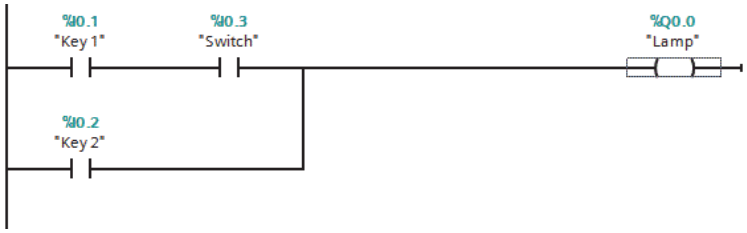


Рис. 19. Тэги в теле программы

Для удаления какого-либо элемента программы необходимо его выделить и выбрать: *Edit→Delete*. Для отмены предыдущего действия необходимо выбрать: *Edit→Undo*.

В случае некорректного ввода адреса или символического имени программа высвечивает некорректное обозначение красным цветом.

Далее рассматривается вставка функции на примере создания следующей программы: если кратковременно нажать на кнопку *Key*1, то зажжётся лампа *Lamp* (адрес *Q*0.0), если кратковременно нажать на кнопку *Key*2, лампа *Lamp* (адрес *Q*0.2) погаснет. Для этого воспользуемся функцией *SR* триггера:

1) в новом сегменте нажать кнопку *Program Elements* – появится окно списка функций (рис. 20);

2) в окне списка функций выбрать: *SR* (*SR* триггер) (рис. 21);

3) над функцией *SR* триггера (рис. 22) вместо символов ″??.?″ вставьте символическое имя выхода;

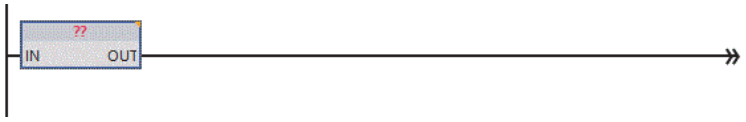


Рис. 20. Функция

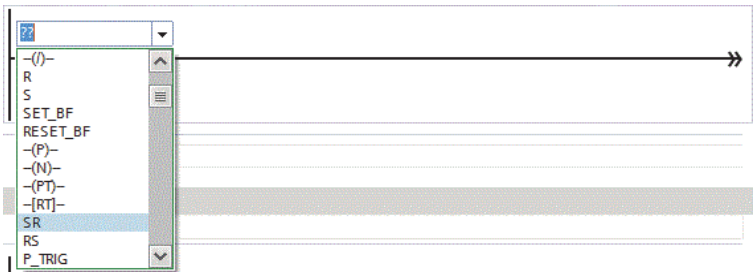


Рис. 21. Окно списка функций

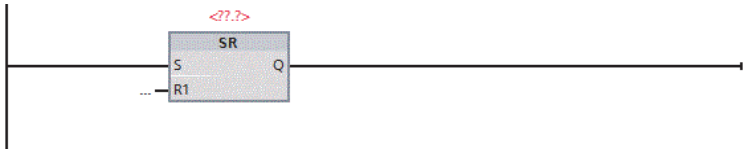


Рис. 22. Триггер

4) на входную линию функции *S* поставьте нормально открытый контакт *Key*1, а на место входа *R* – *Key*2 (рис. 23).

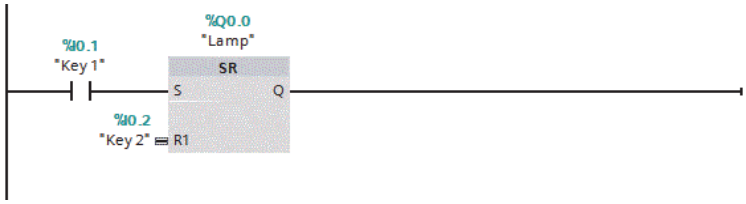


Рис. 23. Указание сигналов S/R триггера

Для удаления функции необходимо её выделить нажать клавишу *Delete* либо подвести к ней курсор, нажать на правую клавишу мыши и из выпадающего меню выбрать *Delete*.

Особые указания к заданиям 5–8

Для решения задач этой группы необходимо использовать счетчики контроллера.

Счетчики имеют собственную область памяти в *CPU*. Для каждого счетчика резервируется слово длиной от 3 до 12 Кб.

Каждый счетчик использует структуру, хранящуюся в блоке данных, для сохранения данных счета. При этом назначается блок данных, когда в программе устанавливается счетчик. Эти команды используют программные счетчики, максимальная скорость счета которых ограничена скоростью исполнения *OB*, в котором они находятся. *OB*, в котором размещены эти команды, должен исполняться достаточно часто, чтобы обнаруживать все изменения входов *CU* или *CD*.

Когда в функциональном блоке устанавливаются команды счета, можно выбрать вариант многоэкземплярного блока данных. Имена структур счетчиков могут быть различными у различных структур данных, но данные счетчиков находятся в одном единственном блоке данных и не требуют отдельного блока данных для каждого счетчика. Это уменьшает время обработки и объем памяти данных, необходимой для счетчиков. Между структурами данных счетчиков в совместно используемом многоэкземплярном блоке данных нет взаимодействия.

В контроллере реализованы 3 вида счетчиков (рис. 24):

1) на сложение (прямой счет);
2) на вычитание (обратный счет);
3) реверсивный (прямой/обратный счет).

На рисунке 24 представлены блоки всех 3-х видов счетчиков, при этом используются следующие обозначения: “*IEC_Counter*_0_*DB*” – имя счетчика, *CTU*, *CTD*, *CTUD* – тип счетчика, *CU* – вход на сложение, *CD* – вход на вычитание, *PV* – задание (уставка) счетчика, *R* – вход сброса, *Q* – выход (состояние) счетчика, *CV* – текущее значение счетчика в двоичном формате.

Уставка счетчика задается либо константой в формате *C*#<значение>, либо из области памяти (*I*, *Q*, *M*, *D*). Нарастающим фронтом сигнала на входе *LD* в счетчик заносится уставка с входа *PV*. При нарастающем сигнале на входе *CU* значение счетчика увеличивается на «1». При нарастающем фронте сигнала на входе *CD* значение счетчика уменьшается на «1», если значение счетчика больше, чем «0». При нарастающем фронте сигнала на входе *R* счетчик сбрасывается (в счетчик записывается значение «0»).

Для реверсивного счетчика, если оба счетных входа имеют нарастающий фронт, то значение счетчика остается неизменным.

Для всех типов счетчиков выход *Q* равен «0», если значение счетчика равно «0», и устанавливается в «1», когда значение счетчика отлично от «0».

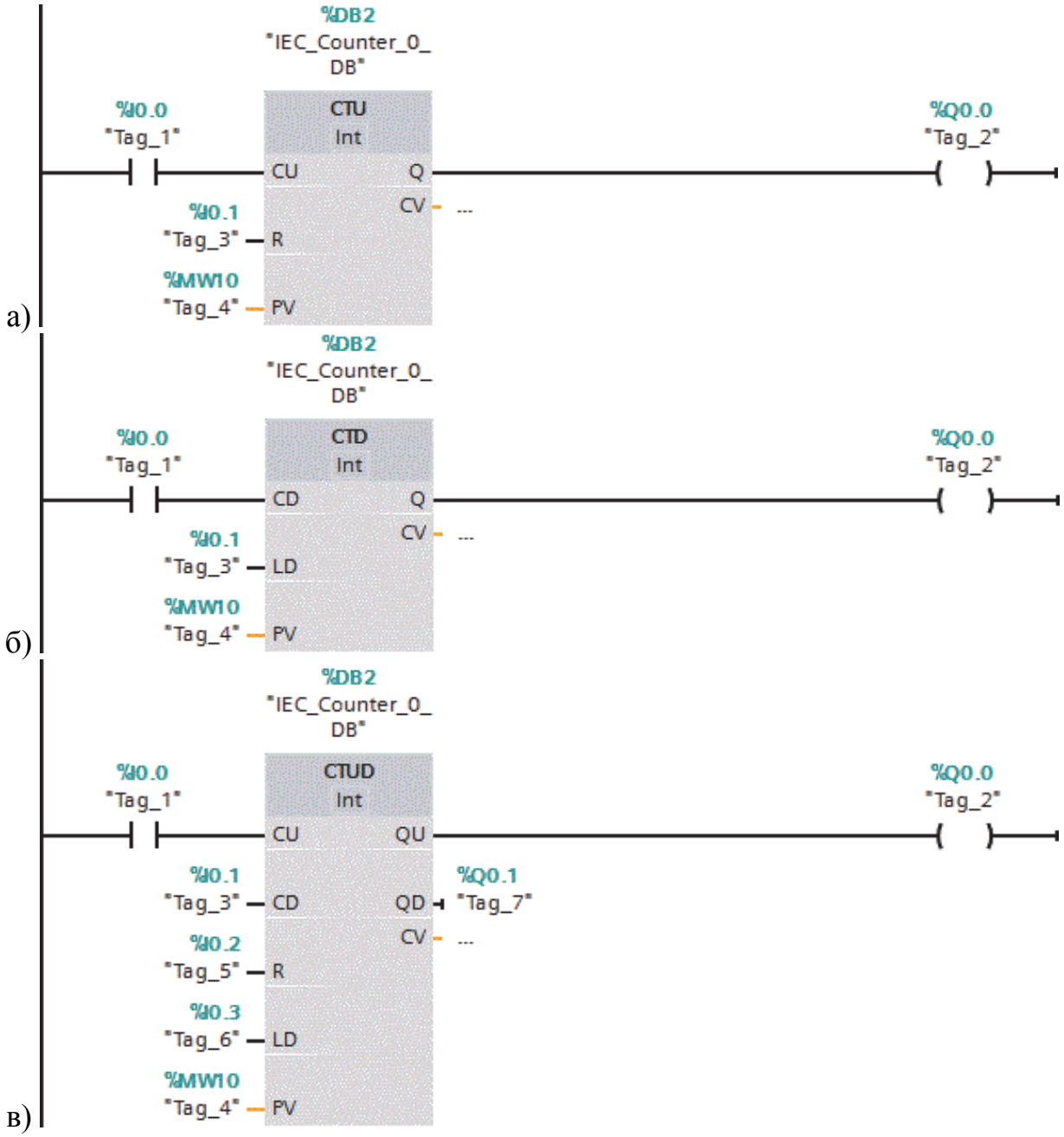


Рис. 24. Счетчики: а) на сложение, б) на вычитание, в) реверсивный

Особые указания к заданиям 9-13

В данной группе задач необходимо воспользоваться таймера, имеющимися в распоряжении ПЛК.

Таймеры имеют собственную область памяти в *CPU*. Для каждого таймера резервируется 16 Кб памяти контроллера. Для задания уставки времени используется идентификатор *T#* совместно с данными типа *TIME* (32 бита).

Таким образом, уставку времени можно задать в *BCD* коде с использованием следующего синтаксиса:

T#aaD_bbH_ccM_ddS_eeeMS,

где *aa* = сутки, *bb* = часы, *cc* = минуты, *dd* = секунды и *eee* = миллисекунды.

Диапазон уставки времени от 0 до 2147483647 мс, что соответствует 24*d*_20*h*_31*m*_23*s*_647*ms*.

Каждый таймер использует структуру, хранящуюся в блоке данных, для сохранения данных о времени. При этом назначается блок данных, когда устанавливается таймерная команда.

Когда в функциональном блоке устанавливаются таймерные команды, можно выбрать вариант многоэкземплярного блока данных. Имена структур таймеров могут быть различными у различных структур данных, но данные таймеров находятся в одном единственном блоке данных и не требуют отдельного блока данных для каждого таймера. Это уменьшает время обработки и объем памяти данных, необходимой для управления таймерами. Между структурами данных таймеров в совместно используемом многоэкземплярном блоке данных нет взаимодействия.

В режиме *RUN* текущее значение таймера уменьшается на одну единицу через интервал времени, установленный базой времени, до тех пор, пока значение времени не станет равным нулю, что соответствует срабатыванию таймера.

В *STEP* 7 *Basic* реализуется 4 разновидности таймеров. В лестничных диаграммах блоки всех разновидностей обозначаются одинаково и различаются только обозначением типа таймера.

Ниже на рис. 25–28 приведены для каждого типа таймера варианты программирования и временные диаграммы их работы, из которых без дополнительных комментариев ясны особенности их работы.

Блок таймера имеет имеет следующие входы и выходы:

– *IN* – вход запуска;
– *PT* – уставка (заданное время);
– *R* – вход сброса;
– *Q* – выход (состояние таймера);
– *ET* – оставшееся время в двоичном формате.

Также возможен сброс таймеров подачей команды *R* (*RESET*) по адресу таймера (рис. 29).

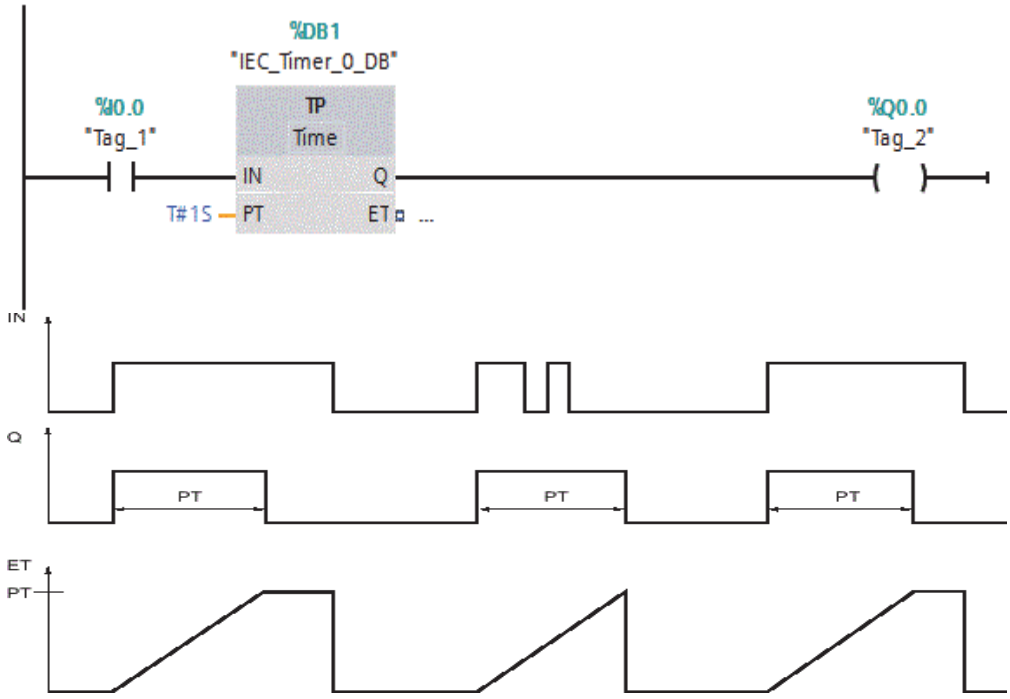


Рис. 25. Таймер формирователь импульса

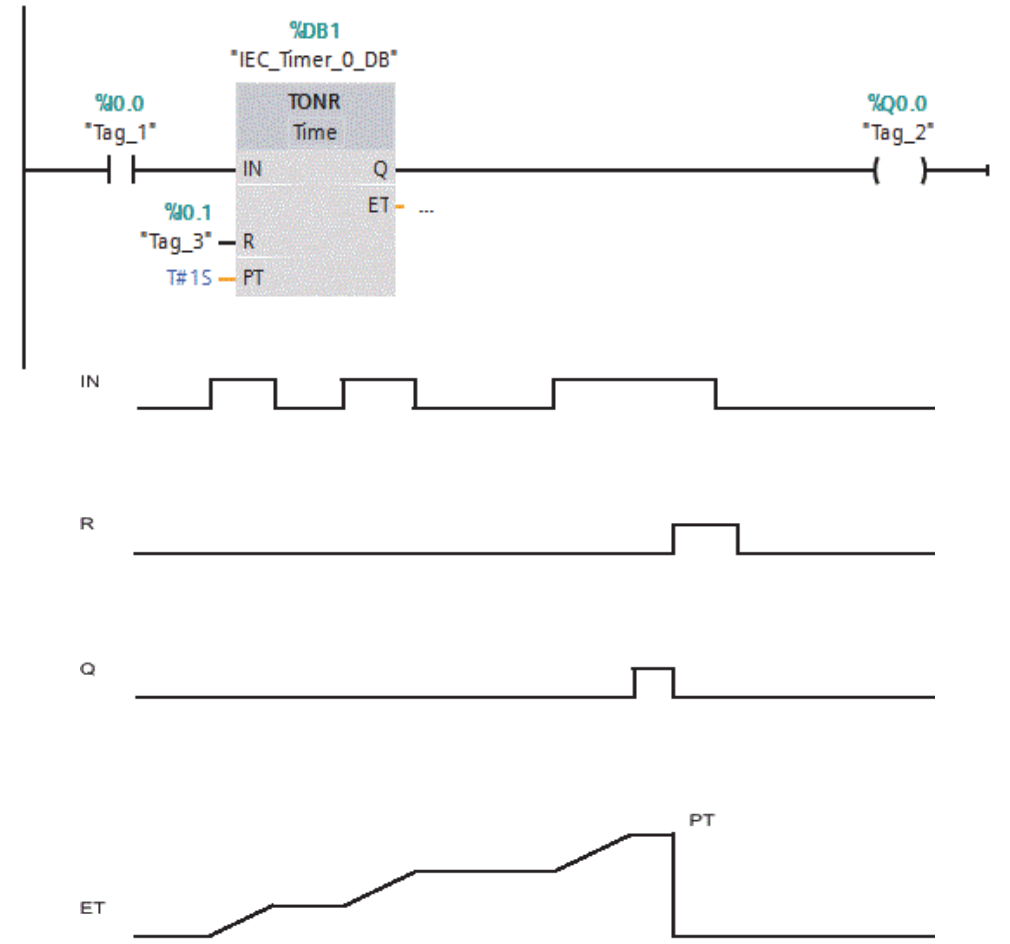


Рис. 26. Накапливающий таймер формирователь задержки включения

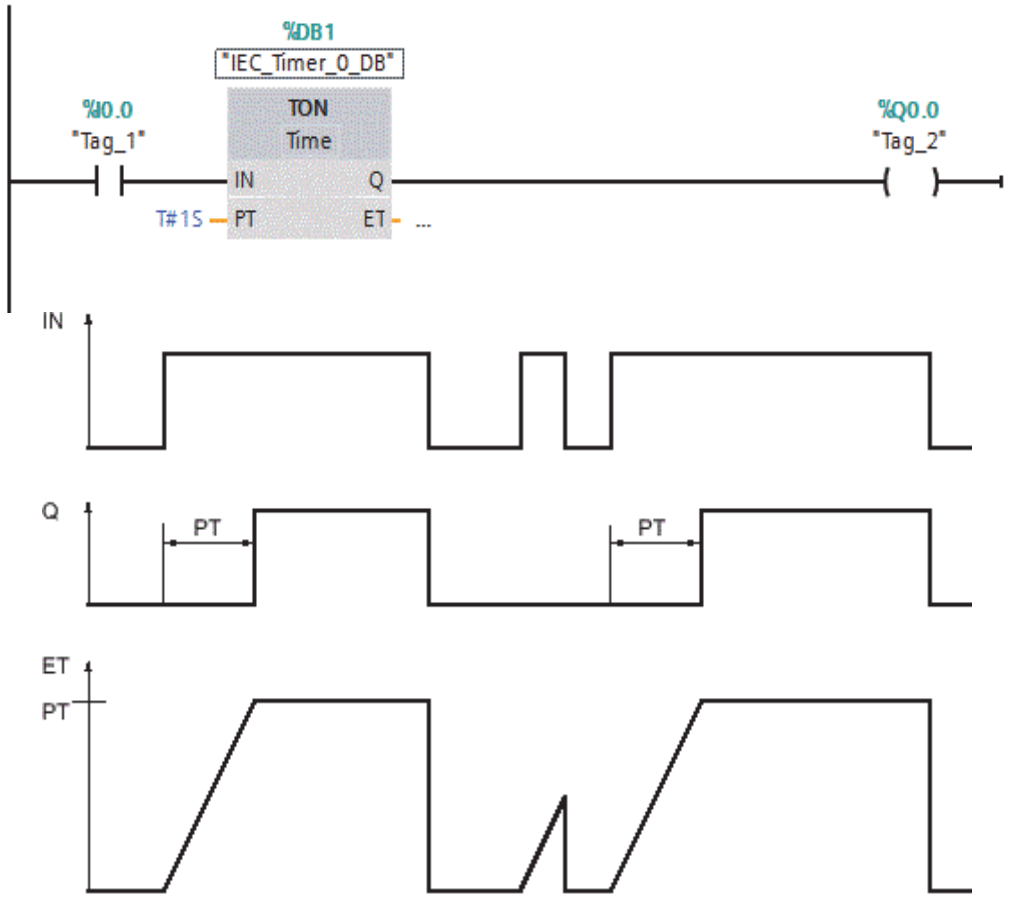


Рис. 27. Таймер формирователь задержки включения

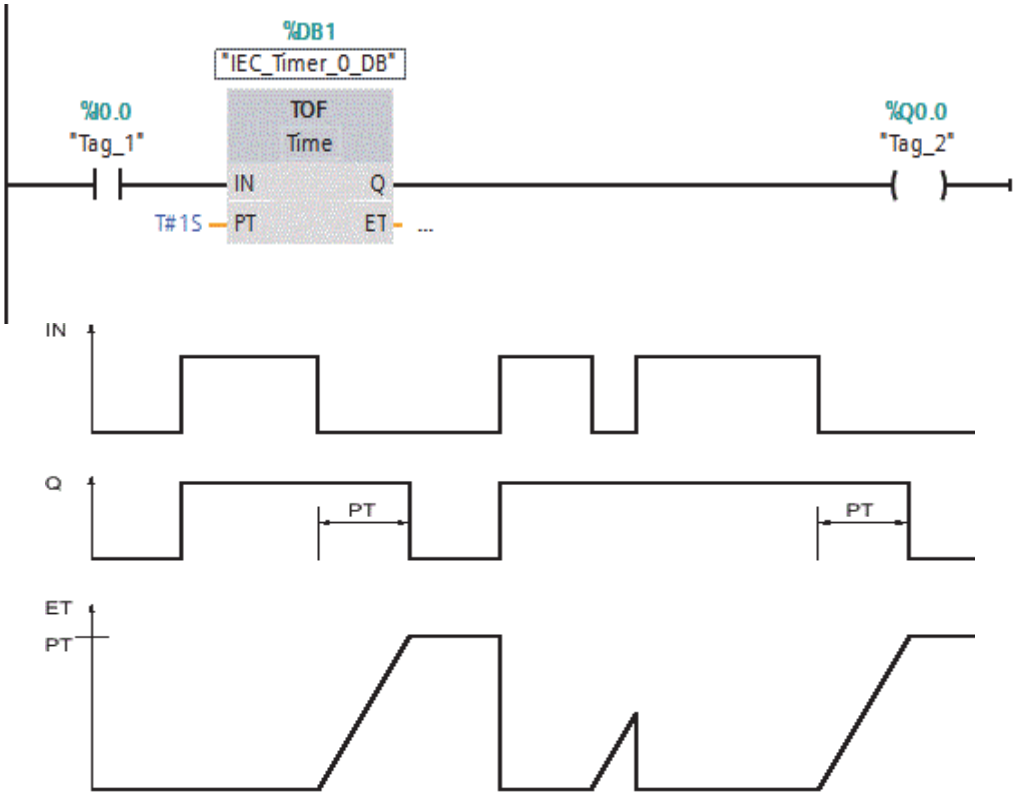


Рис. 28. Таймер формирователь задержки выключения

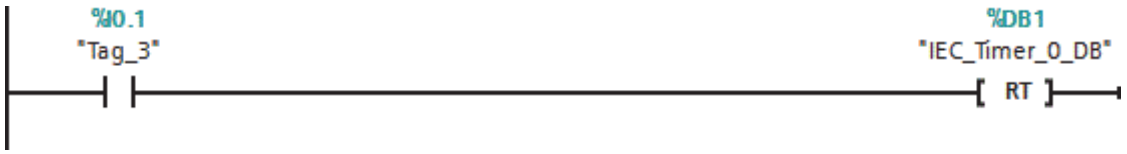


Рис. 29. Сброс таймера

Дополнительные указания к заданию 13

Модель перекрестка (рис. 30) представляет собой изображение двух пересекающихся дорог, где установлены 2 светофора. Каждый их светофоров продублирован. Светофор представляет собой 3 светодиода разных цветов, соответствующих реальному светофору. Каждому светодиоду соответствует клемма. Для того чтобы светодиод загорелся, необходимо на соответствующую клемму подать логическую единицу.

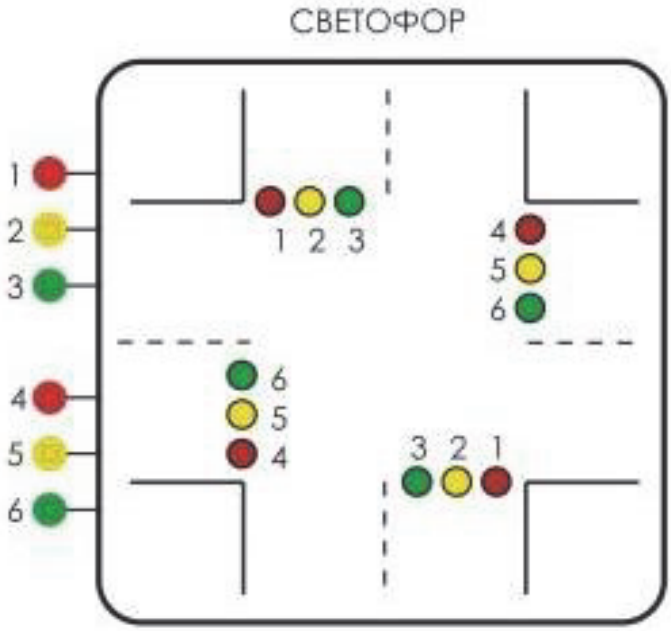


Рис. 30. Модель перекрестка

Особые указания к заданиям 14–15

Язык STEP7 предусматривает возможность упростить программирование, если возникает необходимость в программировании управления однотипными технологическими операциями. Это, например, рассматриваемые ниже испытания двигателей бензинового и дизельного. Испытания в принципе одинаковые, только аппараты (кнопки, датчики и устройства индикации) свои. Тогда централизованно программируется функциональный блок FB, который может многократно вызываться в организационном блоке OB1. Все формальные параметры и статические данные FB сохраняются в отдельном блоке (DB), назначаемом функциональному блоку.

Например, будем программировать функциональный блок FB1, которому дадим символическое имя «Двигатель». При кратковременном появлении

команды «Bkl [Включить]» двигатель должен включиться и сформировать сигнал «Bkl_on [Включен]».

При достижении фактической скорости Fakt заданного значения Zadanie формируется сигнал Est. Двигатель отключается по сигналу «Otkl [Отключить]» или по сигналу «Neispr [Неисправность]». Здесь используется английский шрифт, потому что при программировании FB кириллица не воспринимается.

Прежде всего, необходимо определиться с символьным представлением сигналов и команд (табл. 6). Для организационного блока OB1 в табл. 6 имеется ряд сигналов, которые будут использованы позднее при рассмотрении программирования функций.

Таблица 6

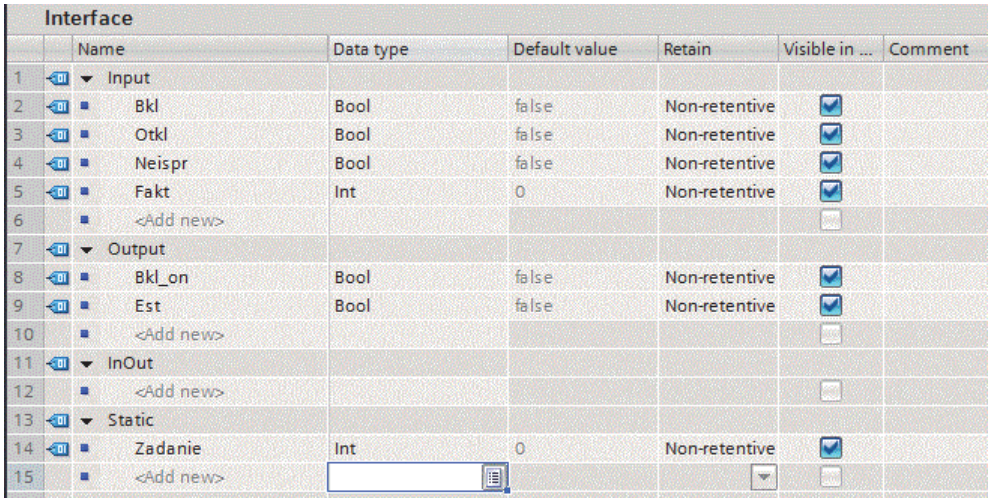
Перечень сигналов

Symbol	Address	Data type	Comment
BD_Bkl	I0.0	BOOL	Включить бензиновый
BD_Otkl	I0.1	BOOL	Отключить бензиновый
BD_Neispr	I0.2	BOOL	Неисправность бензинового
BD_Fakt	MW0	INT	Фактическая скорость бензинового
BD_Bkl_on	Q0.0	BOOL	Бензиновый включен
BD_Est	Q0.1	BOOL	Заданная скорость достигнута
BD_Bent_on	Q0.2	BOOL	Вентилятор бензинового включен
BD_Timer_Bent	0	IEC TIMER	Таймер вентилятора бензинового
DD_Bkl	I1.0	BOOL	Включить дизель
DD_Otkl	I1.1	BOOL	Отключить дизель
DD_Neispr	I1.2	BOOL	Неисправность дизеля
DD_Fakt	MW2	INT	Фактическая скорость дизеля
DD_Bkl_on	Q0.5	BOOL	Дизель включен
DD_Est	Q0.6	BOOL	Заданная скорость достигнута
DD_Bent_on	Q0.7	BOOL	Вентилятор дизеля включен
DD_Timer_Bent	1	IEC TIMER	Таймер вентилятора дизеля

Для создания FB1 в дереве проекта следует выбрать пункт Program blocks, правой кнопкой мыши вызвать всплывающее меню, в нем выбрать Add new block. В появившемся окне выбрать Function block и присвоить FB1 имя «Двигатель». Также в этом окне можно выбрать остальные настройки функционального блока. Одновременно с этим в дереве проекта появится папка FB1.

Для программирования FB1 необходимо дважды щелкнуть на FB1. Открывается окно для программирования. Следует заполнить таблицу

описания переменных (рис. 31). Тип переменной (Bool, Int и т.п.) выбирается с помощью всплывающего меню, используя правую кнопку мыши.



Interface

	Name	Data type	Default value	Retain	Visible in ...	Comment
1	Input					
2	Bkl	Bool	false	Non-retentive	☑	
3	Otkl	Bool	false	Non-retentive	☑	
4	Neispr	Bool	false	Non-retentive	☑	
5	Fakt	Int	0	Non-retentive	☑	
6	<Add new>				☐	
7	Output					
8	Bkl_on	Bool	false	Non-retentive	☑	
9	Est	Bool	false	Non-retentive	☑	
10	<Add new>				☐	
11	InOut					
12	<Add new>				☐	
13	Static					
14	Zadanie	Int	0	Non-retentive	☑	
15	<Add new>				☐	

Рис. 31. Описание переменных функционального блока FB1

Затем программируется включение и отключение двигателя, как было описано выше. Для этого необходимо выделить элемент программы и ввести соответствующие имена из таблицы описания переменных (знак «#» назначается автоматически). Помеченные знаком «#»локальные переменные действительны только в этом блоке.

Набранная программа на языке лестничных диаграмм имеет вид, представленный на рис. 32.

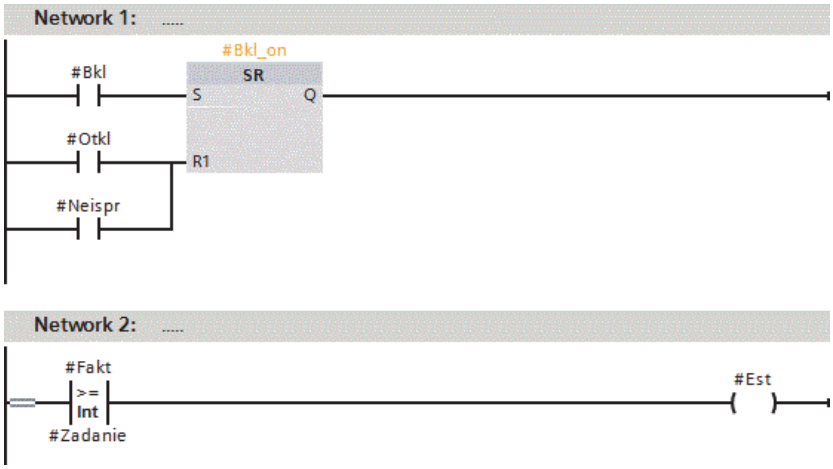


Рис. 32. Программа функционального блока FB1

Чтобы получить возможность вызова организованного блока FB1 в OB1 необходимо сгенерировать соответствующий блок данных. Экземплярный блок данных (DB) всегда ставится в соответствие функциональному блоку.

Рассматриваемый FB1 должен управлять и контролировать работу бензинового и дизельного двигателей. Различные заданные скорости хранятся в отдельных блоках данных.

Для создания блока данных DB необходимо выбрать пункт Program blocks, правой кнопкой мыши вызвать всплывающее меню, в нем выбрать Add new block. В появившемся окне выбрать Data block, задать имя блоку данных DB1 «Бензиновый» и в закладке Type выбрать «Двигатель [FB1]», после чего необходимо выбрать OK. Блок данных DB1 добавляется к проекту. Дважды щелкните на значке DB1, чтобы открыть его.

В открывшемся окне введите значение 1500 для бензинового двигателя в столбец «Start Value» в строке Zadanie. Теперь определена максимальная скорость для этого двигателя. Сохранить DB1 и закройте блок.

Далее так же, как и в случае с DB1, необходимо сгенерировать еще один блок данных DB2 для FB1. Теперь ввести значение 1200 для дизельного двигателя.

На следующем этапе нужно запрограммировать вызов функционального блока в OB1. Это демонстрируется на рис. 33.

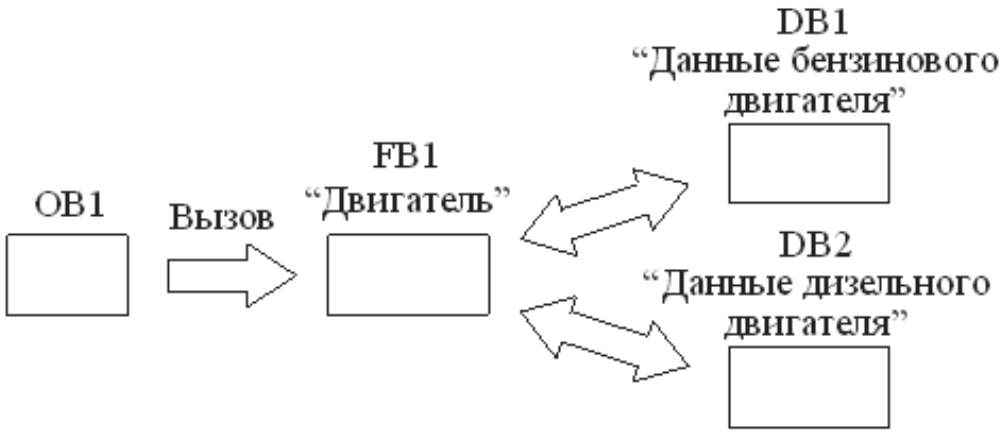


Рис. 33. Вызов блока FB1 в OB1

Для этого необходимо открыть OB1 и из дерева проекта с помощью мышки перетащить функциональный блок FB1. При этом необходимо привязать блок данных «Бензиновый [DB1]» к этому функциональному блоку. Для этого нужно в закладке Name выбрать соответствующий блок данных. После этого нажать OK для подтверждения выбора.

Далее перед каждым из следующих входов (Bkl, Otkl и Neispr) и выходов (Bkl_on, l и Est) необходимо вставить контакты, используя соответствующие символьные имена из табл. 6.

В новом сегменте программы аналогично вызвать еще один функциональный блок «Двигатель [FB1]» совместно с блоком данных «Дизельный [DB2]»

Таким образом программа блока OB1 примет следующий вид (рис. 34).

Далее необходимо сохранить программу и закрыть блок OB1.

Функции, как и функциональные блоки, расположены в иерархии программы ниже организационного блока. Однако в отличие от функционального блока, функции не нужен блок данных. У функций

параметры также перечисляются в таблице описания переменных, но статические данные не разрешаются.

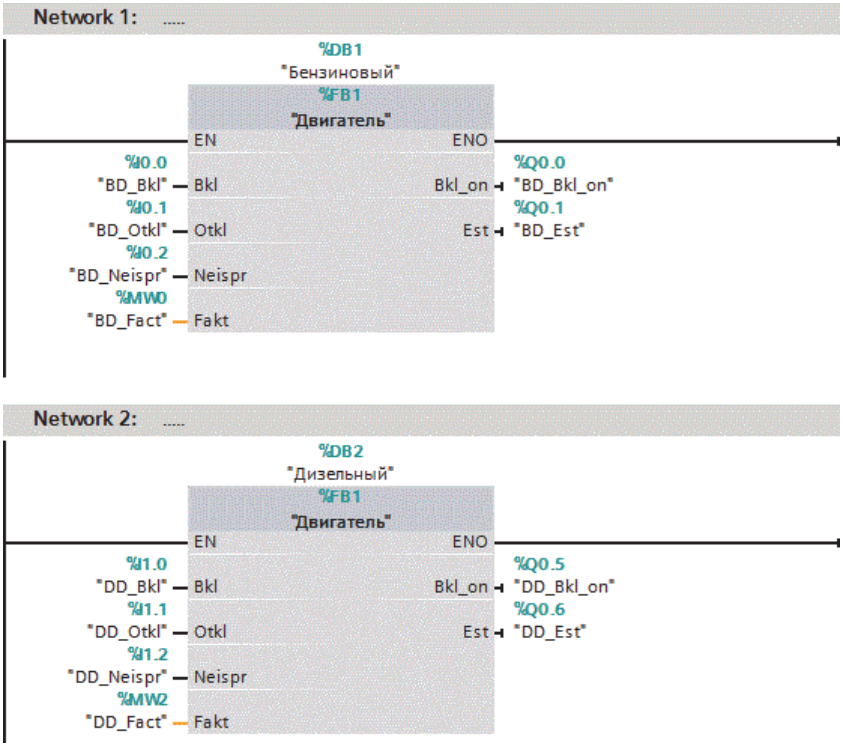


Рис. 34. Программирование вызова функционального блока FB1 в OB1

Для создания функции необходимо выбрать пункт Program blocks, правой кнопкой мыши вызвать всплывающее меню, в нем выбрать Add new block. В появившемся окне выбрать Function FC1, задать ей имя «Вентилятор» и в закладке Language выбрать нужный язык программирования, после чего необходимо выбрать OK. Функция FC1 добавляется к проекту.

В рассматриваемой задаче предусматривается, что каждый двигатель как бензиновый, так и дизель имеет свой вентилятор. Вентилятор включается при работе двигателя и продолжает еще работать 4 с после отключения двигателя. Для управления этими двигателями программируется функция FC1.

Необходимо дважды щелкнуть на значке FC1, чтобы открыть его. Открывается окно для программирования. Далее нужно ввести описание переменных (рис. 35).

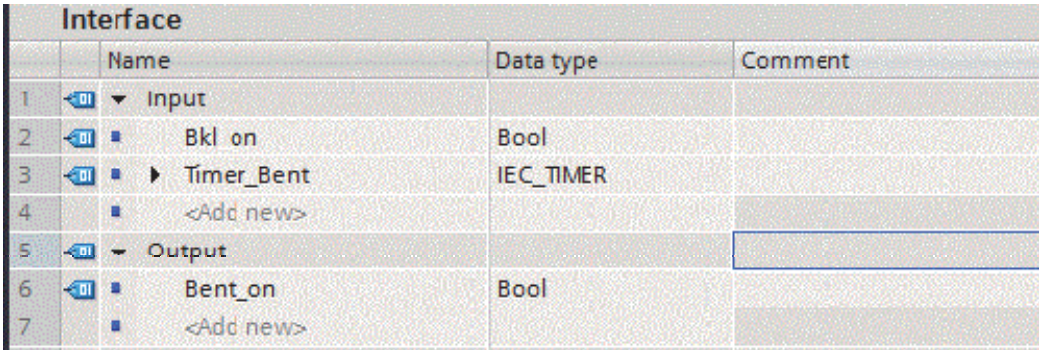


Рис. 35. Описание переменных функций

Затем необходимо выделить сегмент для ввода команд лестничных диаграмм и запрограммировать управление вентилятором. Необходимо выделять вопросительные знаки и вводить соответствующие имена из таблицы описания переменных (знак # назначается автоматически). Набранная программа на языке лестничных диаграмм имеет вид, представленный на рис. 36.

Рис. 36. Программирование функции FC1

Вызов функции FC1 в OB1 выполняется аналогично вызову функционального блока. Все переменные функции снабжаются в OB1 соответствующими адресами бензинового или дизельного двигателя, которые уже отражены в табл. 6.

В блоке OB1 вставить новый сегмент. Затем в дереве проекта необходимо выбрать FC1 и вставить ее в OB1. Выделив вопросительные знаки при вызове FC1, ввести символьные имена из таблицы 6 для бензинового двигателя. Вставьте замыкающий контакт перед «Bkl_on». После этого создать еще один сегмент программы и запрограммировать вызов функции FC1, используя адреса для дизельного двигателя. После этого программа будут иметь вид, представленный на рис. 37. Далее необходимо сохранить блок.

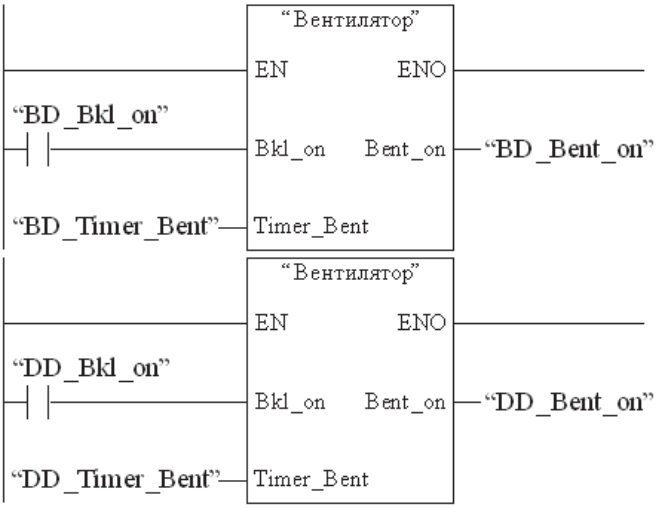


Рис. 37. Программирование вызова функции FC1 в OB1

Пример программирования контроллера SIMATIC S7–1200

В качестве примера представлена программа управления движением тележки. Тележка перемещается по прямолинейному пути. Имеется два фиксированных положения П0 и П1. Исходное положение тележки – П0. При

кратковременном нажатии на кнопку «Пуск» тележка движется вперед до положения П1, стоит в положении П1 5 с и возвращается назад в положение П0. В положении П0 немедленно возникает реверс привода тележки. Тележка вновь движется в положение П1, стоит в нем 5 с и возвращается в П0. В положении П0 вновь возникает команда на движение до положения П1и т.д. после 10 циклов перемещения в положение П1 при возвращении тележки в положение П0 возникает сигнал «Конец цикла».

Для управления движением тележки следует предусмотреть счетчик (СЧ) со счетом до 10, задержку времени (таймер) для формирования задержанного сигнала $П1^{\uparrow}$ и память $Р_П$ о нажатии на кнопку «Пуск»:

$$P_{П} = (Пуск + P_{П}) \cdot \overline{КЦ} .$$

Команда на перемещение тележки вперед (В) возникает в положении П0, если отсутствует сигнал счетчика СЧ, и сохраняется при наличии сигнала памяти пуска $Р_П$ до достижения положения П1. Этому описанию соответствует логическое уравнение

$$B = (П0 \cdot \overline{СЧ} + B) \cdot \overline{П1} \cdot P_{П} .$$

Команда на перемещение тележки назад (Н) возникает через 5 с нахождения в положении П1 и сохраняется при наличии переменной $Р_П$ до достижения положения П0. Тогда логическое уравнение для команды Н имеет вид

$$H = (П1^{\uparrow} + H) \cdot \overline{П0} \cdot P_{П} .$$

Сигнал конца цикла (КЦ) возникает в положении П0 при наличии сигнала счетчика СЧ о завершении 10 циклов перемещения в положение П1и сигнала $Р_П$. Сигнал КЦ сохраняется до очередного нажатия на кнопку “Пуск”. Тогда логическое уравнение для рассматриваемого сигнала имеет вид

$$КЦ = (П0 \cdot СЧ \cdot P_{П} + КЦ) \cdot \overline{Пуск}$$

В ниже приведенной программе (рис. 38–39) использован счетчик на вычитание. При нажатии на кнопку «Пуск» в счетчик вводится уставка 10. Счетчик считает число циклов перемещения в положение П1, при достижении этого положения число в счетчике уменьшается на единицу. Счетчик сбрасывается по сигналу конца цикла КЦ. Признаком окончания цикла является число в счетчике равное нулю, при котором на выходе счетчика «0». Поэтому в уравнениях команд В и КЦ необходимо использовать инверсию сигнала СЧ по отношению к ранее записанным уравнениям

$$B = (П0 \cdot СЧ + B) \cdot \overline{П1} \cdot P_{П}$$

$$КЦ = (П0 \cdot \overline{СЧ} \cdot Р_{П} + КЦ) \cdot \overline{Пуск}$$

Сброс счетчика производится сигналом конца цикла КЦ.
В табл. 7 представлена адресация команд и сигналов для ПЛК S7–1200.

Таблица 7

Адресация команд и сигналов

Команда, сигнал	П0	П1	Пуск	В	Н	КЦ	$Р_П$	СЧ	$П1^{\uparrow}$
Адрес	I0.0	I0.1	I0.2	Q0.0	Q0.1	Q0.2	M0.0	M0.1	M0.2
Символ	П0	П1	Пуск	Вперед	Назад	КЦ	Память	СЧ	Задержка

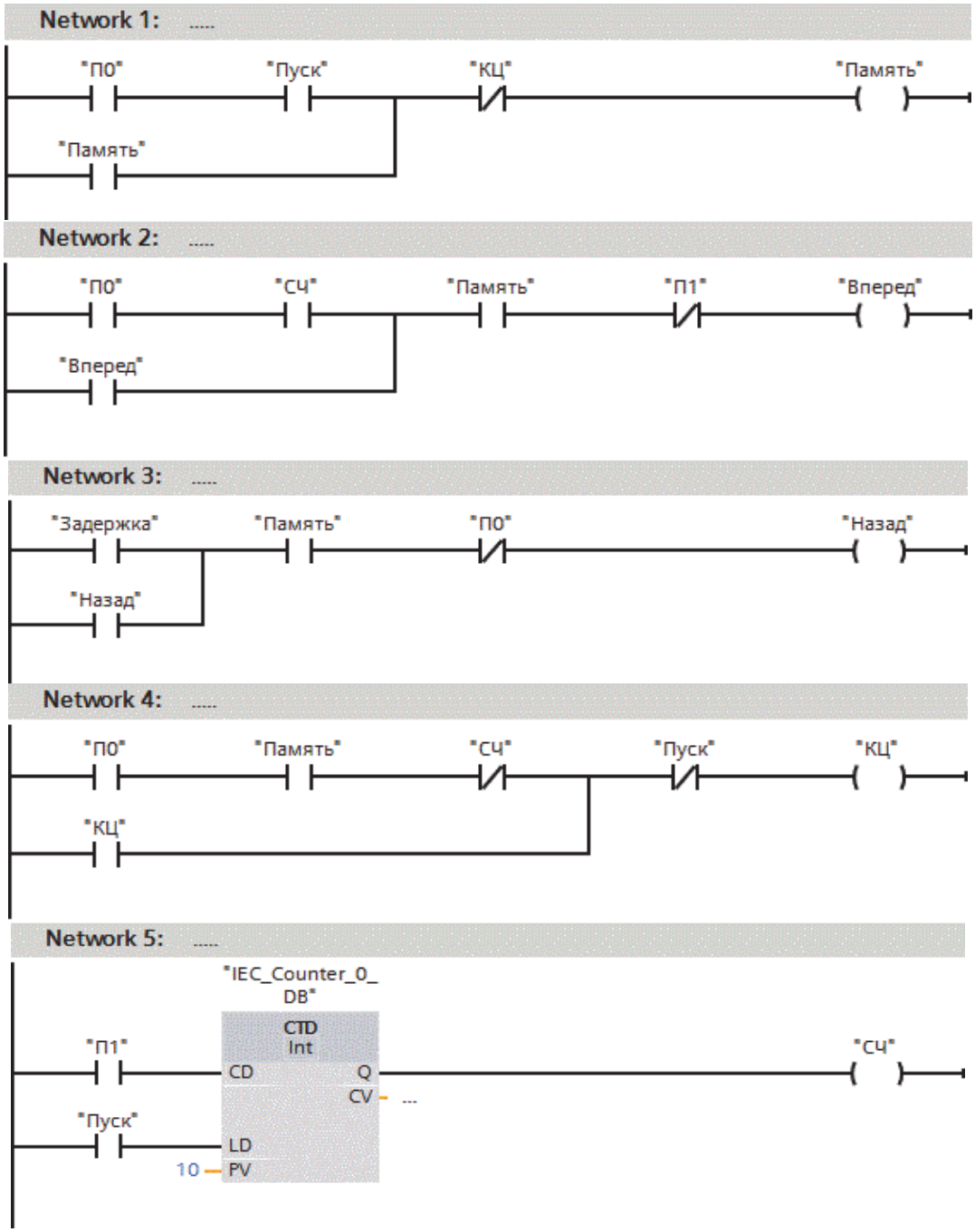


Рис. 38. Программа управления движением тележки

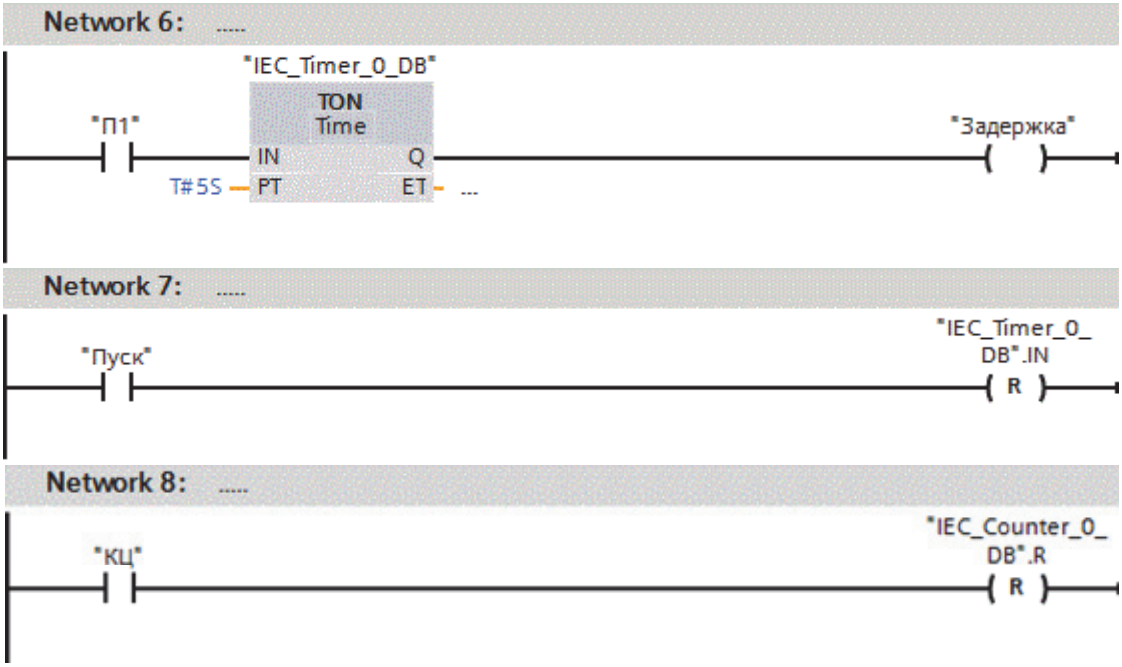


Рис. 39. Программа управления движением тележки

Дополнительные указания к заданию 15.

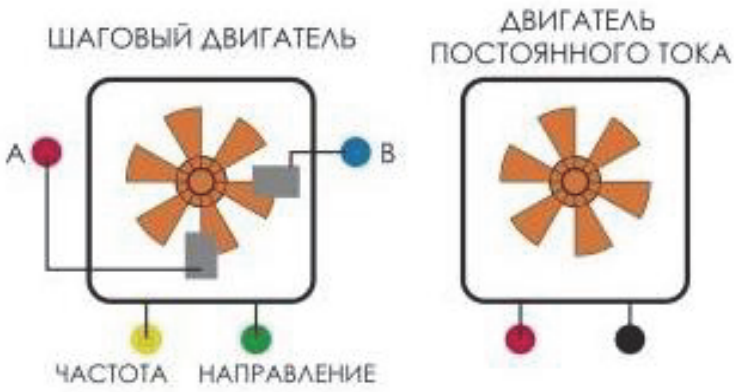


Рис. 40. ШД и ДПТ на лицевой панели стенда

Шаговый двигатель и двигатель постоянного тока, установленные в стенде (рис. 40), имеют собственную систему управления, управляющие входы которой расположены на лицевой панели. Вход «Частота» отвечает за скорость вращения шагового двигателя. На него необходимо подавать импульсы, пропорционально частоте которых электродвигатель будет вращаться. Минимальная длительность импульса должна составлять 1 мс (рис. 41).

Минимальное время между двумя импульсами – также 1 мс. Система управления реагирует только на передний фронт импульса, поэтому скважность входного сигнала (отношение длительности импульса к времени между импульсами) значения не имеет.

Если на вход «Направление» подавать логический ноль, то ШД будет вращаться в одну сторону, если логическую единицу – то в противоположную.

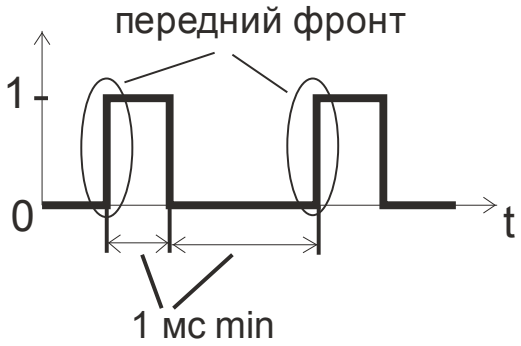


Рис. 41. Требования к управляющим импульсам

На валу шагового двигателя расположен диск с прорезями. Рядом размещаются два оптических датчика, реагирующих на прорези вращающегося диска. Выходы датчиков подключены к клеммам «Датчик А» и «Датчик В». При прохождении прорезью рабочей зоны датчика на клемме появляется сигнал логической единицы. Между собой два датчика расположены таким образом, что при вращении диска сигналы появляются не одновременно, а с определенным промежутком времени.

Двигатель постоянного тока управляется подачей сигнала логической единицы на красную клемму расположенную непосредственно под диском ДПТ, использование черной клеммы не требуется.

Меняя скважность сигнала можно управлять скоростью вращения ДПТ. Таким образом появляется возможность реализовать управление методом широтно-импульсной модуляции.

Методические указания по работе с симулятором ПЛК Siemens

1. Откройте проект в представлении проекта, и в навигационной панели выберите CPU, для которого вы хотите установить соединение.

2. Выберите команду меню "Online> Simulation> Start" или в строке меню щелкните значок "Start simulation" (рис. 42).

3. В диалоговом окне "Extended download to device" выбираете тип интерфейса PG/PC, например "PN/IE". Затем интерфейс "PN/IE" автоматически установится в "PLCSIM".

4. Выберите "Connection to interface/subnet", настроенное в конфигурации устройства, затем нажмите кнопку "Start search" (рис. 43). Примечание: Если для вашего процессора уже было настроеное подключение к интерфейсу PG/PC, аналогичное указанному выше, то онлайн-соединение автоматически устанавливается с этими параметрами.

5. Когда онлайн-соединение установлено, нажмите кнопку "Load".

6. В открывшемся диалоговом окне "Extended download to device" нажмите "Load preview".

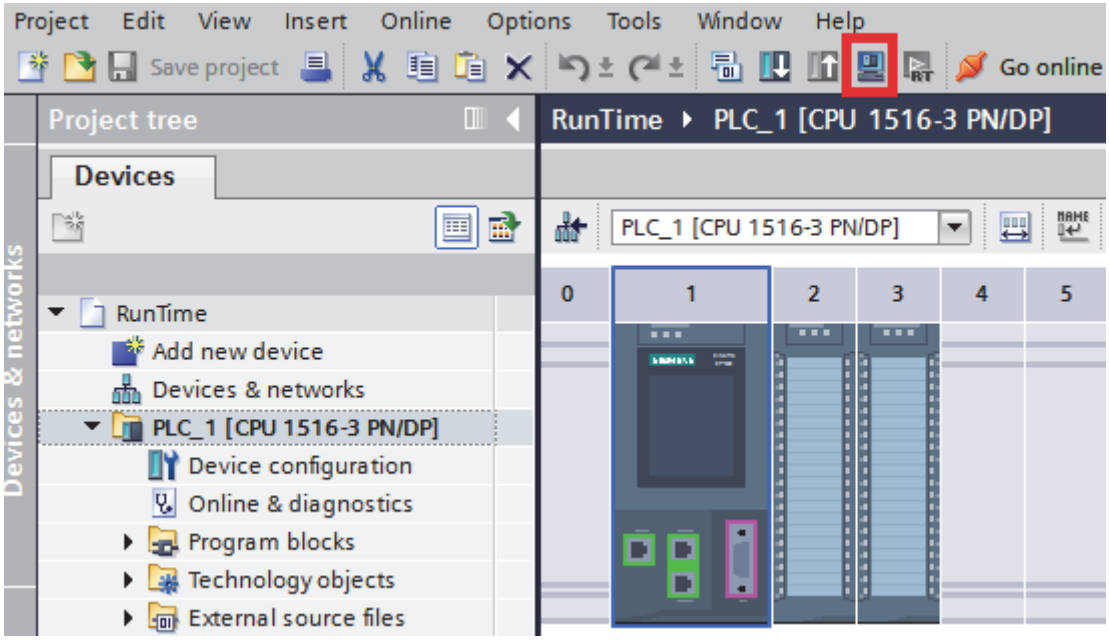


Рис. 42. Окно моделирования в режиме «Онлайн»

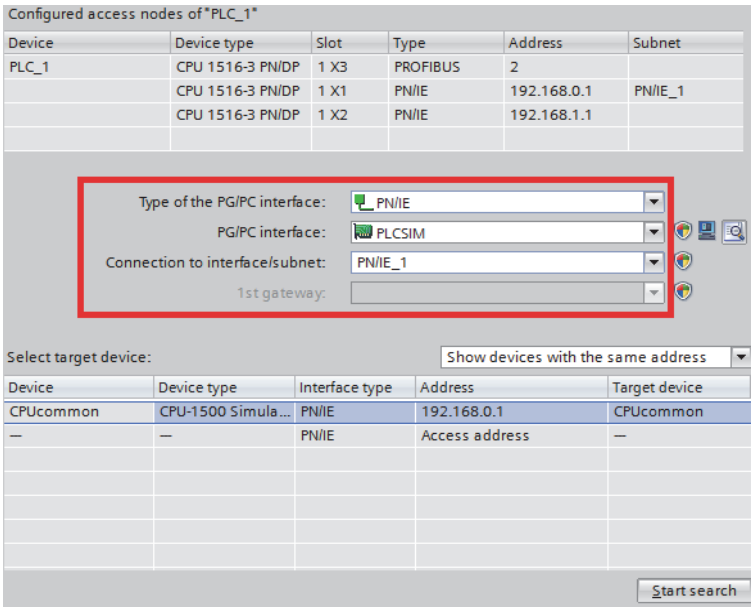


Рис. 43. Окно подключения к интерфейсу / подсети

7. Как и в случае с реальным процессором, в конце открывается диалоговое окно "Load results". Включите функцию "Start all" и нажмите "Finish".

8. Переключитесь в диалог "PLCSIM". С помощью кнопки, обозначенной на рис. 44, в S7–PLCSIM вы можете переключиться с компактного представления на представление проекта.

9. Проверьте свою программу.

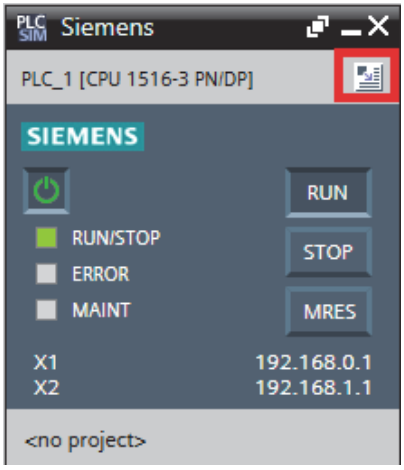


Рис. 44. Кнопка переключения вида представления

Указания по загрузке программы в контроллер и запуску программы

Компиляция и загрузка программы в контроллер осуществляется в следующей последовательности:

В дереве проекта выделить папку контроллера (*PLC*_1). Для компиляции части проекта в выпадающем меню «*Edit*» выбрать пункт «*Compile*». При этом начнется компиляция проекта (рис. 45). В нижней части экрана появится лог компиляции, отображающий ход процесса (рис. 46). По завершению компиляции в логе отобразится итоговая информация. Если предупреждений и ошибок в проекте нет, то все пункты будут отмечены синими галочками. В случае наличия ошибок и предупреждений, все они будут отображены, и двойным кликом левой клавиши мыши можно перейти к сегменту программы, который содержит ошибку или предупреждение и отредактировать этот сегмент.

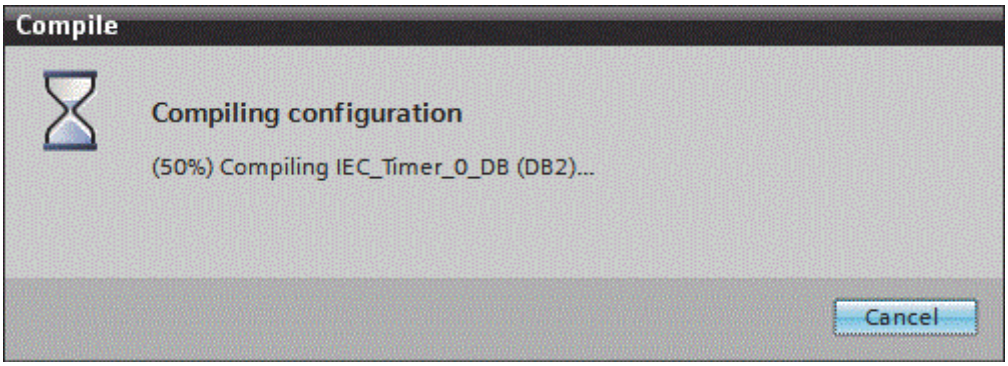


Рис. 45. Отображение процесса компиляции

В дереве проекта выделить папку контроллера (*PLC*_1). В меню «*Online*» выбрать пункт «*Extended download to device* [Расширенная загрузка в устройство]» (рис. 47). В строке «*Type of PG/PC interface*» установить «*PN/IE*», в строке «*PG/PC interface*» – выбрать сетевой адаптер, в строке «*Connection to subnet*» – конкретную локальную сеть. После этого начнется процесс сетевой идентификации контроллера. Далее необходимо выбрать «*Load*», после чего начнется процесс компиляции и загрузки проекта в память ПЛК.

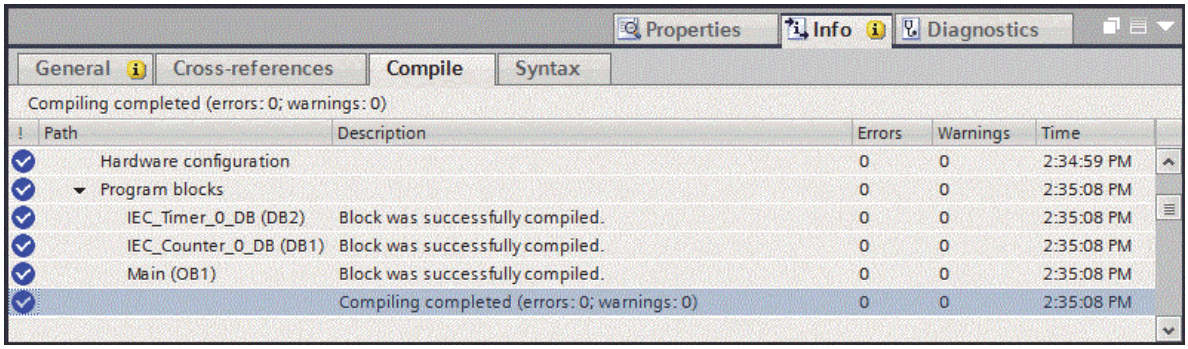


Рис. 46. Лог процесса компиляции

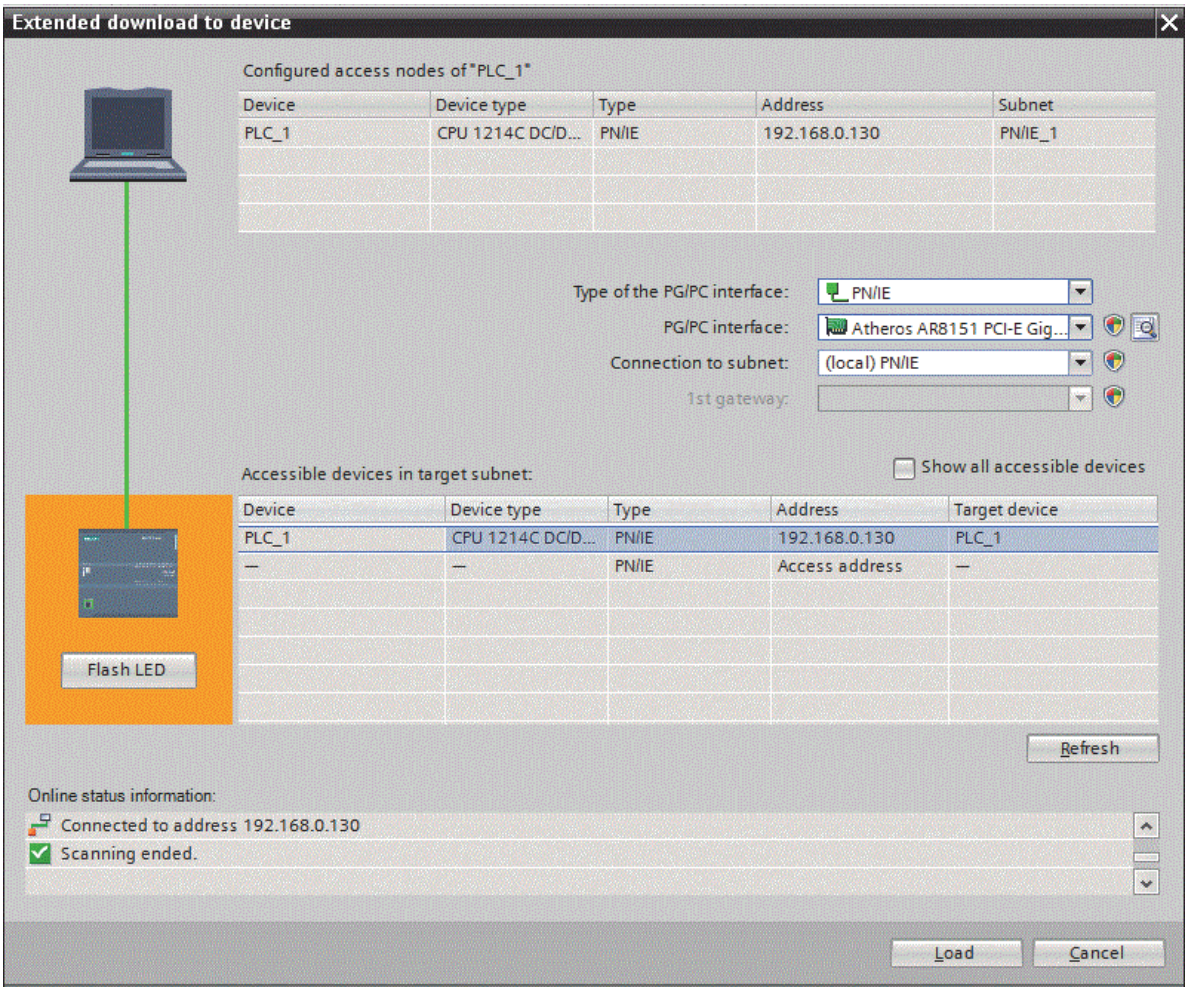


Рис. 47. Диалоговое окно «Расширенная загрузка в устройство»

После завершения процесса компиляции необходимо снова выбрать пункт «*Load*», после чего начнется процесс загрузки в память непосредственно управляющий программ проекта. После этого установить галочку «*Start all*» и нажать «*Finish*», после чего произойдет процесс перезапуска контроллера. В итоге проект полностью загружен в память ПЛК, и контроллер переходит в рабочий режим «*RUN*».

Во время работы контроллера доступен режим мониторинга хода выполнения программы. Для этого необходимо в дереве проекта выбрать

желаемый программный блок, открыть его в окне программирования. Для перехода в режим мониторинга необходимо в панели инструментов окна редактирования проекта выбрать «*Monitoring on/off*» . После этого *STEP*7 *Basic* переходит в режим мониторинга и в окне редактирования проекта будут отображаться состояния контактов и катушек, а также значения счетчиков, таймеров и других функции в реальном времени (рис. 48).

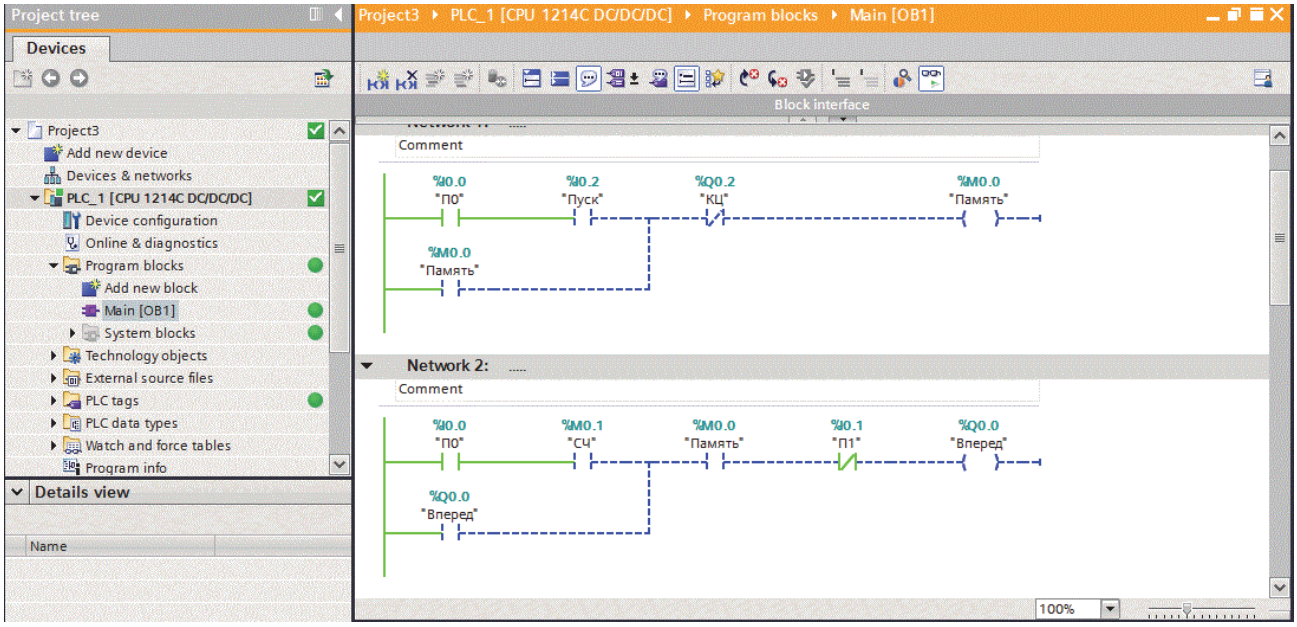


Рис. 48. Режим мониторинга

Учебное издание

Казаринов Лев Сергеевич,
Канашев Евгений Александрович

ПРОГРАММИРОВАНИЕ СИСТЕМ
РЕАЛЬНОГО ВРЕМЕНИ

Учебное пособие

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 20.11.2018. Формат 60×84 1/16. Печать цифровая.
Усл. печ. л. 2,32. Тираж 30 экз. Заказ 463/166.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.