Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра автоматизации производственных процессов

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ В ИНСТРУМЕНТАЛЬНОЙ СИСТЕМЕ CoDeSys

Методические указания
к выполнению лабораторной работы
по дисциплине «Технические средства автоматизации»
для студентов очной и заочной форм обучения направления
220700.62 «Автоматизация технологических процессов и производств»

Курган 2014

Кафедра автоматизации производственных процессов

Дисциплина: «Технические средства автоматизации»

Составил: канд. техн. наук, доцент Н.Б.Сбродов

Утверждены на заседании кафедры «14» ноября 2013 г.

Рекомендованы методическим советом университета в рамках проекта «Инженерные кадры Зауралья» «22» ноября 2013 г.

ВВЕДЕНИЕ

Программируемые логические контроллеры и микроконтроллеры являются в современном производстве одним из основных средств автоматизации технологических процессов в различных отраслях промышленности. На рынке средств автоматизации предлагаются сотни различных моделей контроллеров, различающихся техническими характеристиками, функциональными возможностями, средствами программирования, стоимостью и т.д.

В этих условиях специалисту, занимающемуся проектированием, наладкой и эксплуатацией систем управления технологическими объектами важно знать архитектуру и технические параметры контроллеров, принципы и средства разработки прикладного программного обеспечения.

Цель лабораторной работы – изучения основных приемов работы в инструментальной системе CoDeSys по стандарту Международной электротехнической комиссии (МЭК) 61131-3 и приобретение практических навыков программирования контроллера модели ПЛК100 фирмы ОВЕН.

1 ОБЩАЯ ХАРАКТЕРИСТИКА КОНТРОЛЛЕРА МОДЕЛИ ПЛК100

Программируемый логический контроллер ПЛК100 (рисунок 1) предназначен для создания систем автоматизированного управления технологическим оборудованием в различных областях промышленности [1].

Основные технические характеристики контроллера приведены в таблице 1.

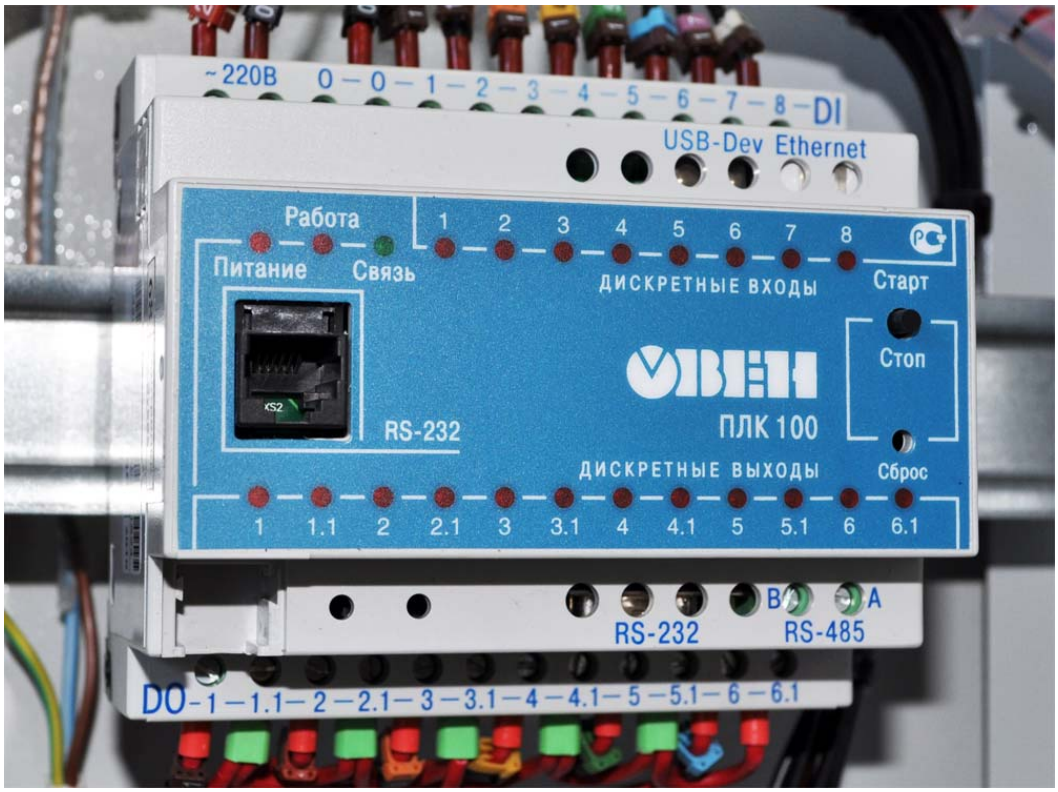


Рисунок 1 – Контроллер ОВЕН ПЛК100

Таблица 1 – Основные технические характеристики контроллера ПЛК100

Наименование	Модель контроллера	
	ПЛК100-24	ПЛК100-220
Напряжение питания	18... 29 В постоянного тока (номинальное напряжение 24 В)	90... 264 В переменного тока (номинальное напряжение 220 В) частотой 47... 63 Гц
Количество дискретных входов	8	
Количество дискретных выходов	6 э/м реле или 6 сдвоенных транзисторных ключей (всего 12 выходных сигналов)	
Максимальная частота сигнала, подаваемого на дискретный вход	1 кГц при программной обработке 10 кГц при применении аппаратного счетчика и обработчика энкодера	
Количество дискретных выходов	6 э/м реле или 6 сдвоенных транзисторных ключей (всего 12 выходных сигналов)	
Гальваническая развязка дискретных входов	есть, групповая	
Гальваническая развязка дискретных выходов	есть, индивидуальная	
Центральный процессор	32-х разрядный RISC-процессор 200 МГц на базе ядра ARM9	
Объем оперативной памяти	8 Мбайт	
Объем энергонезависимой памяти хранения ядра CoDeSys, программ и архивов	4 Мбайт*	
Время выполнения цикла ПЛК	Минимальное 250 мкс (нефиксированное), типовое от 1 мс	
Интерфейсы	Ethernet 100 Base-T RS-232 - 2 канала RS-485 USB 2.0 -Device	
Протоколы	ОВЕН ModBus-RTU, ModBus-ASCII DCON ModBus-TCP GateWay (протокол CoDeSys)	
Среда программирования	CoDeSys 2.3.8.1 (и старше)	
Интерфейс для программирования и отладки	RS-232 USB-Device Ethernet	
Конструктивное исполнение	Унифицированный корпус для крепления на DIN-рейку (ширина 35 мм), длина 105 мм (6U), шаг клемм 7,5 мм	

**Для хранения файлов и архивов используется Flash-память, специализированная файловая система; доступный для пользователя объем 3 Мбайт.

2 МЕТОДИКА ПРОГРАММИРОВАНИЯ КОНТРОЛЛЕРА В ПРОГРАММНОЙ СРЕДЕ CoDeSys

2.1 Назначение и возможности комплекса CoDeSys

Программный комплекс CoDeSys (Controllers Development System) фирмы 3S (Smart Software Solutions) представляет проектировщику удобную среду для программирования контроллеров на языках МЭК [2 - 5].

CoDeSys позволяет использовать языки: IL, ST, LD, SFC, FBD и CFC. В данной работе будем использовать только графический язык LD.

Текстовые редакторы CoDeSys производят автоматическое объявление переменных, тип которых задается в диалоговом окне, и другие действия.

Графический редактор автоматически выполняет расстановку компонентов схемы (контактов, катушек реле, таймеров и пр.) и трассировку их соединений; нумерацию цепей; масштабирование изображения, что позволяет увидеть всю LD - диаграмму (программу) или какую-то её часть и выделять цветом активные цепи.

Встроенные эмулятор и элементы визуализации дают возможность выполнять отладку проекта без самих аппаратных средств.

Интерфейс инструментальной системы и базовые команды графического языка лестничных диаграмм LD (Ladder Diagram) подробно рассмотрены в методических указаниях [6].

1.3 Запуск программного комплекса CoDeSys

Произведем первый запуск среды CoDeSys. В окне **Target Settings** напротив строки **Configuration** выбираем тип логического контроллера **PLC 100.R-L,** поскольку именно он используется в лабораторной работе, и нажимаем ОК. В появившемся окне **Новый программный компонент (POU)** (рисунок 2) выбираем тип **POU - Программа** и язык, на котором будет осуществляться написание программы - **LD**. Имя программы оставляем без изменения. Подтверждаем выбор нажатием на кнопку **ОК**. После выполнения всех вышеописанных действий откроется главное окно (рисунок 3) среды CoDeSys.

Его можно разделить на различные области (в окне они расположены сверху вниз):

- меню (рисунок 4);
- панель инструментов, которая содержит кнопки для быстрого вызова команд меню (рисунок 5);
- организатор объектов, имеющий вкладки «**POU», «Типы данных»**, **«Визуализации»** и **«Ресурсы»**;
- разделитель организатора объектов и рабочей области CoDeSys;
- рабочая область, в которой находится редактор;
- окно сообщений;
- строка статуса, содержащая информацию о текущем состоянии проекта.

Меню находится в верхней части главного окна. Оно содержит все

команды CoDeSys.

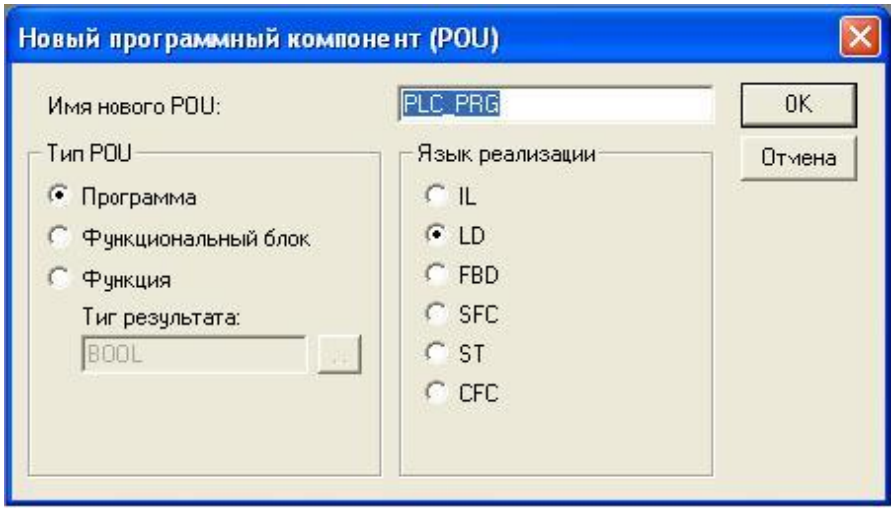


Рисунок 2 – Выбор языка программирования и задание имени программы

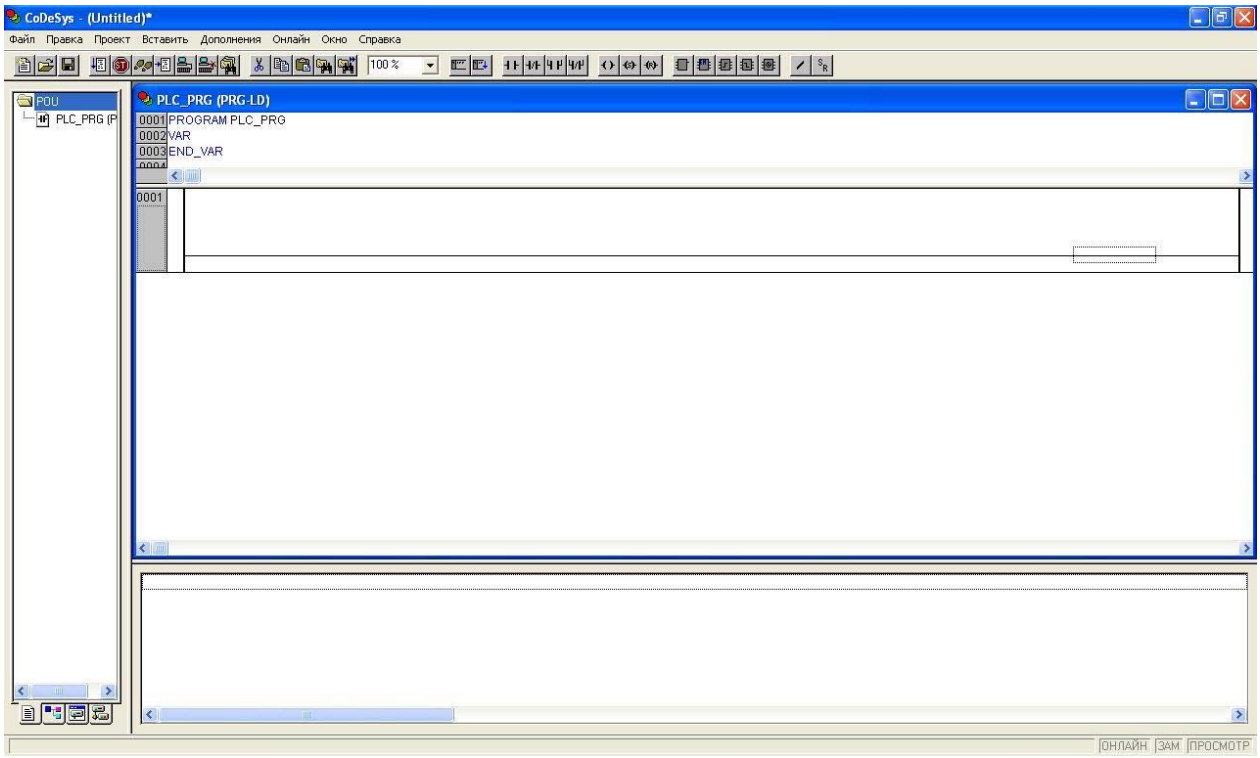


Рисунок 3 – Главное меню CoDeSys

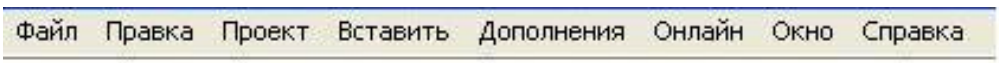


Рисунок 4 – Меню среды

Рисунок 5 – Панель инструментов

Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню. Вызванная с помощью кнопки на панели инструментов команда автоматически выполняется в активном окне. Получить информацию относительно назначения этих кнопок при разработке LD-программ можно в методических указаниях [6].

Организатор объектов (рисунок 5) всегда находится в левой части главного окна CoDeSys. В нижней части организатора объектов находятся вкладки «**POU**», **«Типы данных»**, **«Визуализации»** и **«Ресурсы»**. Переключаться между соответствующими объектами можно с помощью мышки, нажимая на нужную вкладку.

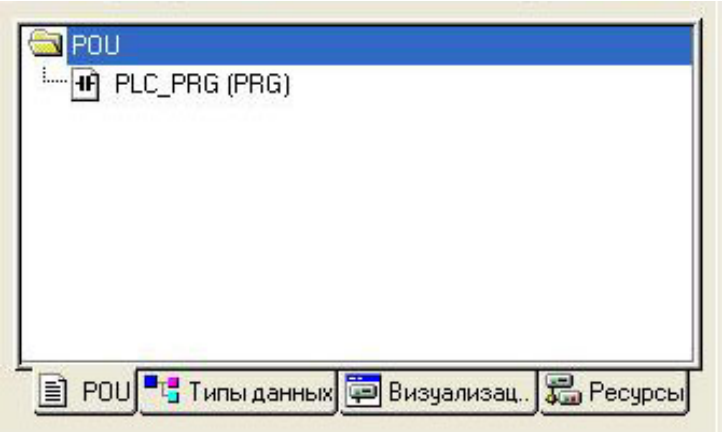


Рисунок 5 – Организатор объектов

1.5 Окно сообщений

Окно сообщений отделено от рабочей области разделителем. Именно в этом окне появляются сообщения компилятора, результаты поиска и список перекрестных ссылок.

При двойном щелчке левой клавишей мыши или при нажатии клавиши Enter на сообщение будет открыт объект, к которому относится данное сообщение.

Далее сокращенно операции с кнопками мыши будем записывать так: 1ЛКМ, если одно нажатие на левую клавишу мышки, 2ЛКМ - если два нажатия; 1 ПКМ, если один щелчок правой кнопкой.

1.6 Статусная строка

Статусная строка находится в нижней части главного окна CoDeSys и предоставляет информацию о проекте и командах меню.

Если вы поместили указатель на пункт меню, то в строке статуса появляется его краткое описание.

Если работаете в режиме **Онлайн**, то надпись **Онлайн** в строке статуса выделяется черным цветом. В ином случае надпись серая. С помощью статусной строки в режиме **Онлайн** можно определить, в каком состоянии находится программа. Например, **«Эмул.»** - в режиме эмуляции.

Статусную строку можно убрать либо включить (в меню **«Проект»**, **«Опции…»**, **«Рабочий стол»**).

1.7 Контекстное меню

Альтернативой использования главного меню для вызова команд может стать контекстное меню (рисунок 6). Это меню, вызываемое 1ПКМ на рабочей области, содержит наиболее часто используемые команды.

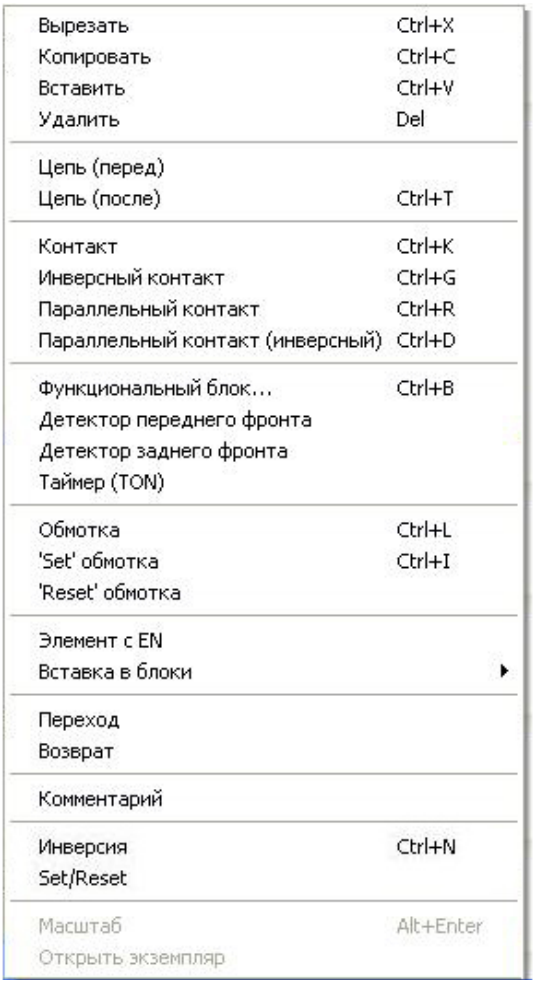


Рисунок 6 – Контекстное меню программы CoDeSys

2.2 Программирование временных интервалов

2.2.1 Основные типы таймеров в CoDeSys

Большинство технологических процессов по своей сути разбиты на временные интервалы, чередующиеся события. Для формирования данных временных интервалов и фиксации событий в программах управления ПЛК применяют таймеры [4].

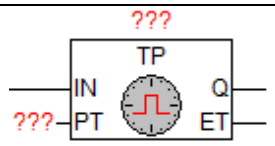
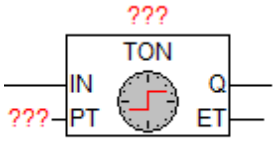
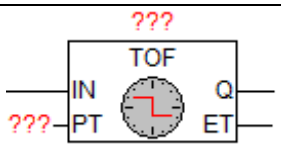
В CoDeSys применяются в основном три типа таймеров: TP-таймер – таймер одиночного импульса, TON-таймер – таймер с задержкой включения,

TOF-таймер – таймер с задержкой выключения (таблица 2). Указанные таймеры могут входить в состав LD-программы в виде функциональных блоков.

У всех таймеров есть вход IN для логических сигналов, вход PT для задания требуемых временных параметров (уставки), логический выход Q и выход ET текущего значения времени.

Работу этих таймеров поясняют временные диаграммы входных и выходных сигналов (рисунок 7).

Таблица 2 – Типы таймеров

Обозначение таймера в LD-программе	Наименование таймера
	**TP-таймер** или таймер (генератор) одиночного импульса с заданной по входу PT длительностью
	**TON-таймер** или таймер с задержкой включения
	**TOF-таймер** или таймер с задержкой выключения

Из этих диаграмм следует, что TP-таймер запускается мгновенно передним фронтов входного сигнала и в течение времени $T_{им}$ действия выходного сигнала не реагирует на новые импульсы, поступающие на вход IN (рисунок 7 а).

TON-таймер срабатывает по переднему фронту входа IN, но сигнал на выходе Q появится с задержкой $T_{вк}$, установленной по входу PT. TON-таймер не реагирует на импульсы продолжительностью менее значения $T_{вк}$. После того, как TON-таймер отрабатывает заданную выдержку времени (уставку), задний фронт входного сигнала IN сбрасывает выход Q таймера (рисунок 7 б).

TOF-таймер запускается по заднему фронту входа IN. При этом выход Q сбрасывается после спада входного сигнала с задержкой времени T_3, установленной по входу PT. Пауза между входными сигналами должна быть не меньше времени задержки (рисунок 7 в).

2.2.2 Пример LD программы с таймером

На рисунке 8 показаны временные диаграммы изменения значения двух входных сигналов SQ1, SQ2 и двух выходных сигналов KM1, KM2.

Оба выходных сигнала KM1, KM2 равны логическому 0 в следующие

интервалы времени: от 0 до T1, в течение которого входные сигналы SQ1 и SQ2 имеют высокий уровень (логическая 1), от T1 до T2, в течение которого сигнал SQ1 имеет низкий уровень (логический 0), а сигнал SQ2 – высокий уровень (логическая 1), от T2 до T3 – оба входных сигнала имеют низкий уровень. В момент времени T3 SQ=1, SQ=0 запускается таймер с задержкой включения, равной 15с. По истечении указанного времени формируются новые значения выходных сигналов КМ1=1, КМ2=1. Выключение выходов происходит, в момент перехода в 0 входа SQ1.

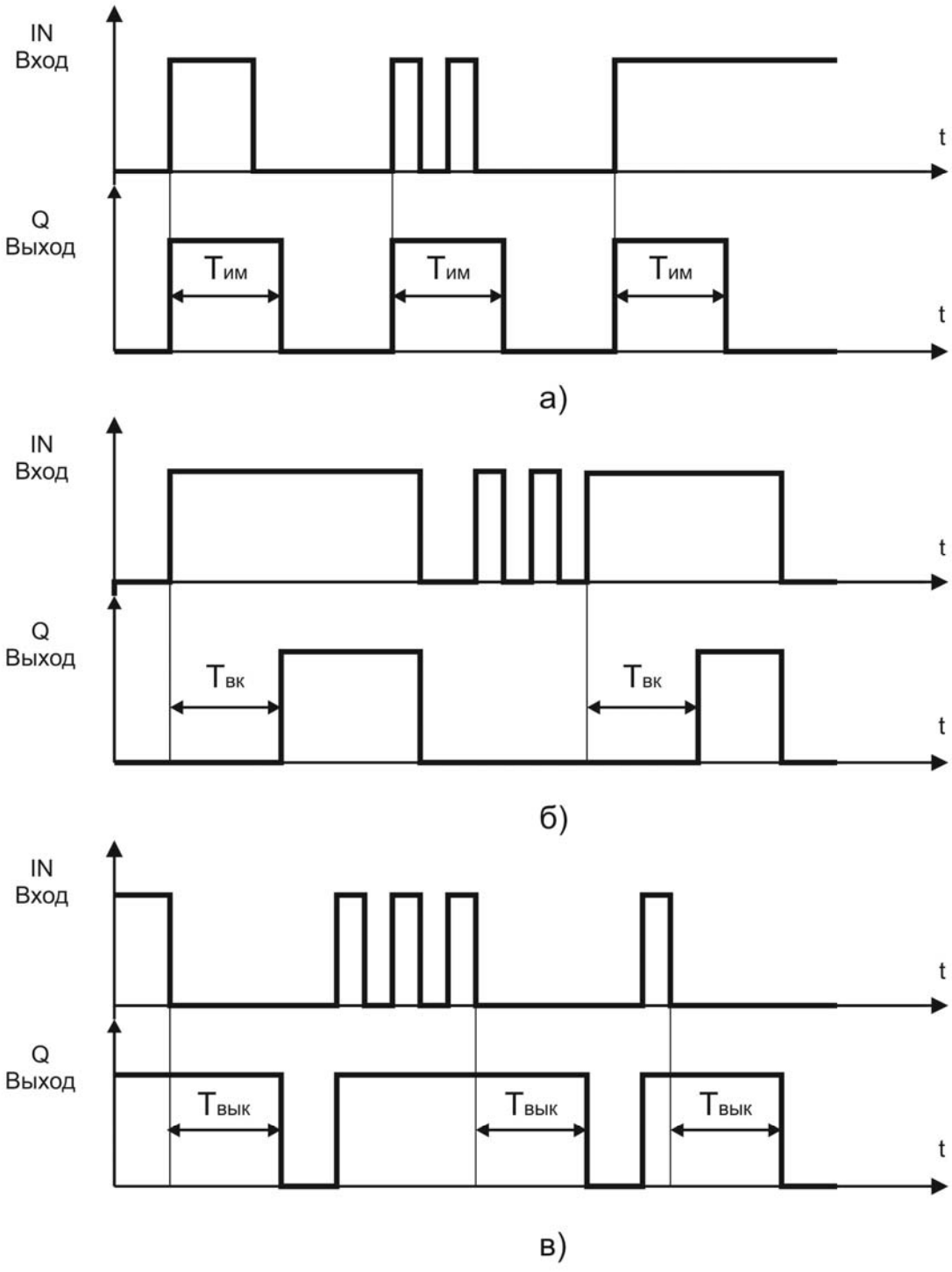


Рисунок 7 – Временные диаграммы таймеров

Для ввода в состав LD-программы таймеров необходимо, чтобы в проекте была подключена соответствующая библиотека функциональных блоков. Обычно в начале работы над проектом такое подключение отсутствует. Чтобы его реализовать необходимо в организаторе объектов 1ЛКМ выбрать вкладку **Ресурсы**. Щелкаем 2ЛКМ по строке **Менеджер библиотек**. В левой верхней части открывшегося окна **Менеджера библиотек** щелкаем 1ПКМ и в открывшемся окне (рисунок 9) выбираем 1ЛКМ строку **Добавить библиотеку**.

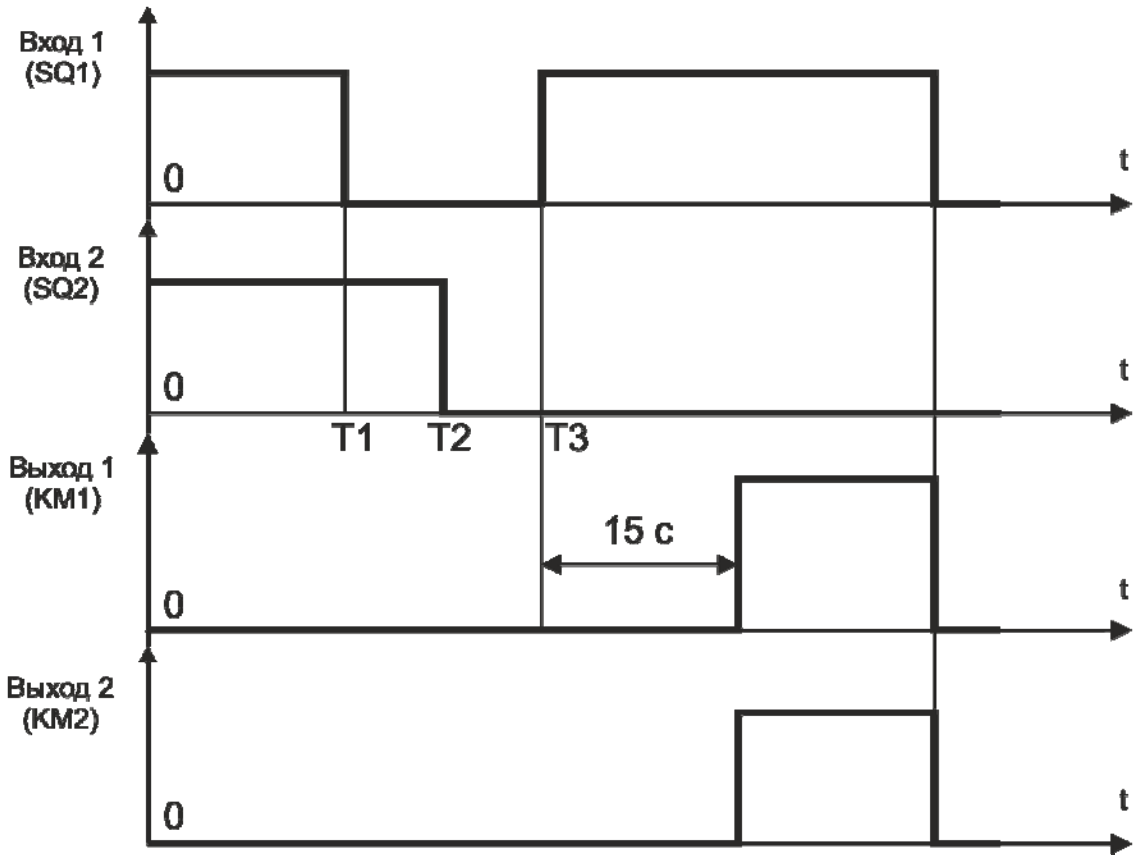


Рисунок 8 – Временные диаграммы

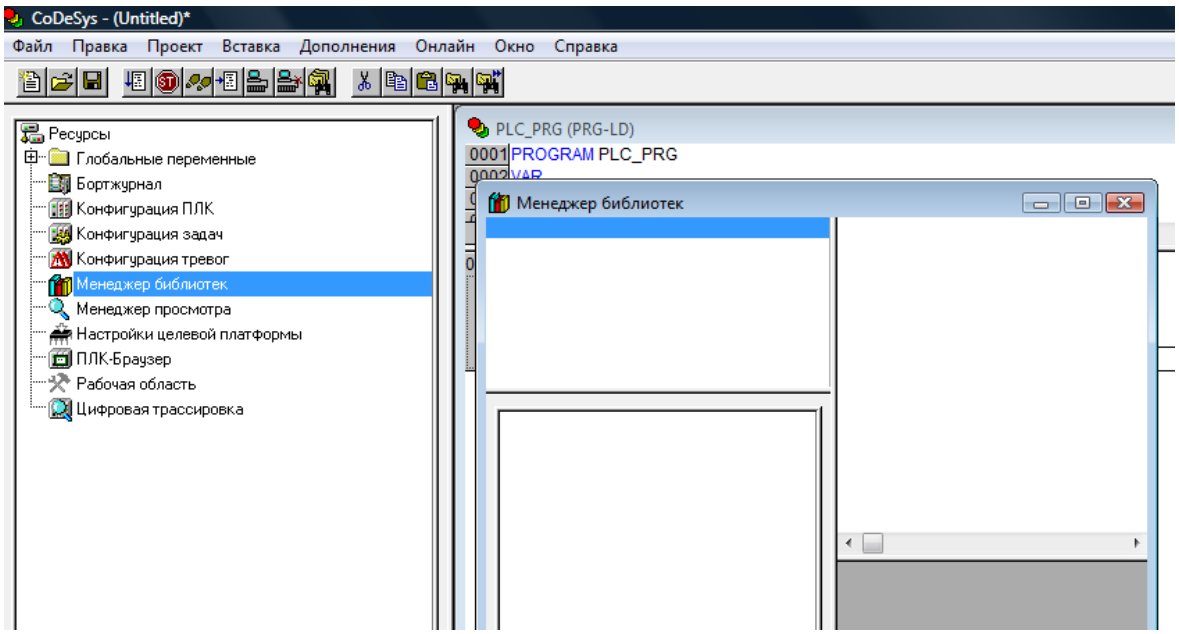


Рисунок 9 – Добавление библиотеки в проект

В окне **Открыть** выбираем и открываем файл **Standard.lib** (рисунок 10). Библиотека стандартных функциональных блоков загружена в проект. С помощью вкладки **POU** организатора объектов можно вернуться в проектируемую программу.

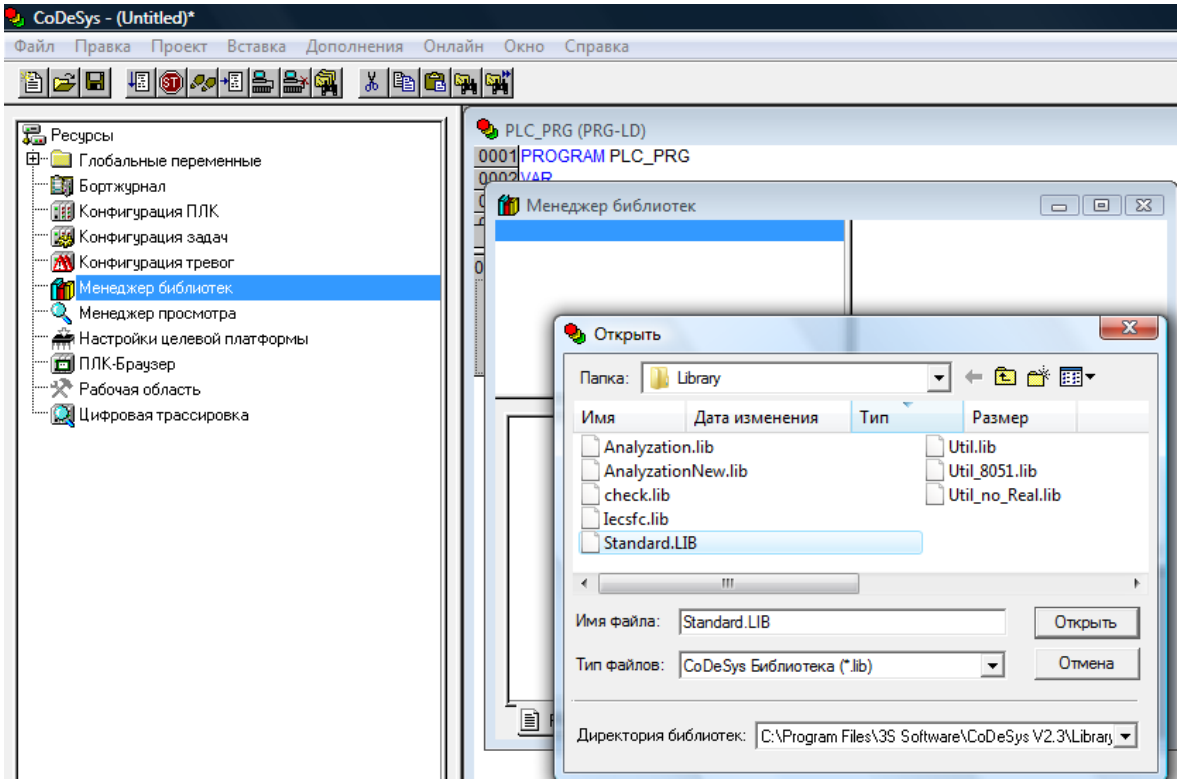


Рисунок 10 – Выбор библиотеки

Для включения в цепь LD-программы таймеров можно щелкнуть 1JIKM по кнопке  панели инструментов. Откроется окно. Подводим курсор к кнопке **Timer**, щелкаем 1ЛКМ. Открывается папка с перечислением функциональных блоков FB. Подводим курсор к строке с нужным таймером; щелкаем 1JIKM. Затем направляем курсор на ОК и нажимаем 1JIKM. Таймер появился в цепи. TON-таймер можно сразу включить в цепь, щелкнув 1JIKM по кнопке  панели инструментов.

При включении любого таймера в цепь многоступенчатой схемы программа запрашивает кроме имени этого функционального блока (вопросительные знаки над таймером) временную уставку по входу PT (вопросительные знаки у входа PT). Временные уставки задают в миллисекундах (mS), секундах (S), минутах (m) или часах (h).

Щелкнув 1ЛКМ по верхним ???, присваиваем имя, например Timer. Щелкнув по ??? у входа PT, нажав и не отпуская клавишу **Shift,** нажать T, затем #. Отпустить **Shift,** набрать требуемое значение задержки - 15, единицу времени (S - т. е. секунды) и **Enter.**

Программа будет выглядеть, как показано на рисунке 11.

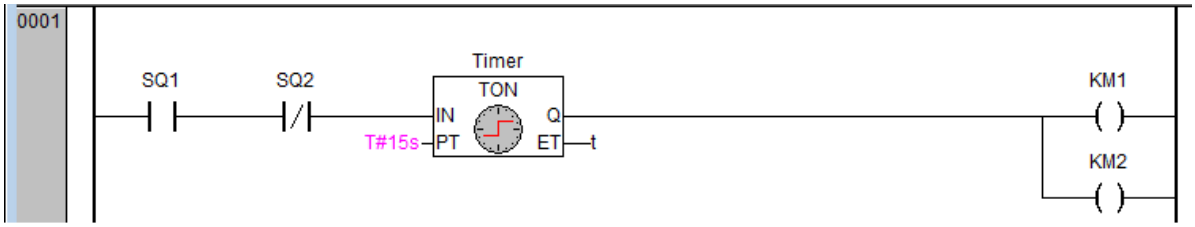


Рисунок 11 – Программа с TON-таймером

На выходе ET можно получит информацию о текущем значении уставки PT с момента запуска таймера. С этой целью при написании программы наводим курсор на выход ET, щелкаем 1JIKM. Появляется мерцающий курсор. Задаем идентификатор этого выхода, например, t. Нажимаем Enter. Открывается окно объявления переменной. Подводим курсор к кнопке (в конце строки с типом переменной BOOL), щелкаем 1JIKM. Открывается окно **Input assistant** (рисунок 12). Щелкаем 1ЛКМ по строке TIME, потом ОК.

В режиме эмуляции можно проверить работу данной программы. Задав значения входов SQ1=TRUE, SQ2=FALSE, запускаем TON-таймер. При этом на выходе ET таймера можно наблюдать в режиме эмуляции изменение уставки PT. При отработке таймером заданной уставки значение выхода таймера Q= TRUE, что обеспечивает включение выходов (KM1=TRUE, KM2=TRUE). Если изменить состояние входов SQ1=TRUE, SQ2=FALSE, то значение входа таймера IN= FALSE, что в соответствии с временной диаграммой работы TON-таймера (рисунок 7 б) и исходной временной диаграммой (рисунок 8) сбрасывает значение выхода таймера Q и выключает выходы KM1 и KM2.

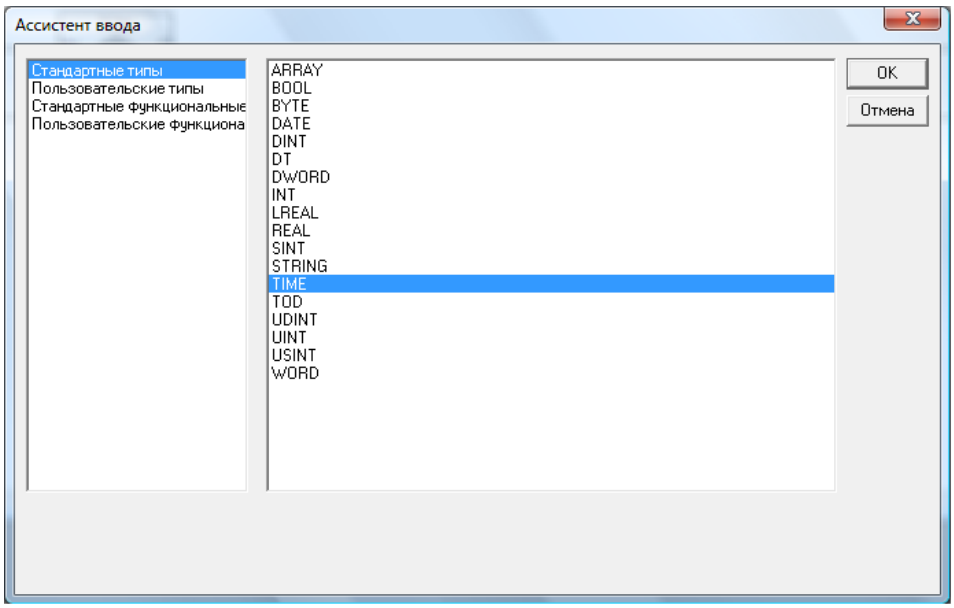


Рисунок 12 – Окно **Input assistant**

Теперь после срабатывания таймера на выходе ET можно наблюдать в режиме эмуляции изменение уставки PT.

2.2.3 Исследование работы LD программ с таймерами

Наличие таймера не вызывает задержки в прогоне программы.

Необходимо отметить, что есть кажущаяся схожесть электрических РКС и LD-диаграмм. Однако есть принципиальное отличие: в РКС параллельные цепи получают питание одновременно, а в LD «чтение», т. е. сканирование многоступенчатой схемы выполняется последовательно слева направо и сверху вниз. Поэтому, если какое-то реле в цепи изменит свое состояние, то цепи расположенные **ниже** получат новое значение переменной **сразу**, а цепи, находящиеся **выше**, - только в следующем цикле прогона программы.

Создадим две LD-программы (рисунки 13 и 14).

В этих схемах использован детектор импульса R_TRIG, который формирует на выходе Q короткий одиночный импульс по переднему фронту сигнала, поступающего на вход CLK.

Отличие двух программ лишь в расположении S и R катушек реле Y1, т. е. с «позиций» РКС эти схемы идентичны (на пятую цепь пока не обращаем внимания).

Сначала испытаем в режиме эмуляции схему по рисунку 13. «Нажимаем» кнопку Pusk. Сработала катушка реле X. Мгновенно замкнулся его контакт в четвертой цепи. Через детектор переднего фронта F2 пришел очень короткий импульс на R-катушку реле Y1. В следующем цикле замкнулся контакт X в первой цепи и через F1 вызвал срабатывание катушки S реле Y1. Реле сработало и своим контактом во второй цепи включило катушку абстрактной нагрузки Motor.

Повторяем опыт с другой схемой 14 (пока без пятой цепи). После команды

Pusk сработало только реле X.

В чем дело? А дело в том, что после срабатывания X, замыкается его контакт в четвертой цепи и короткий импульс поступает на катушку S (с именем Y1). Настолько короткий, что заметить ее срабатывание не удается.

В том, что срабатывание все-таки имело место легко убедиться, добавив пятую цепь с еще одной катушкой S. (Но с другим, разумеется, именем, например, Y2). Вторую катушку R этого реле не используем.

Прогон программы закончен и начинается новый цикл. Срабатывает контакт X в первой цепи и короткий импульс, формируемый детектором **F1,** поступает в катушку сброса **R** реле **Y1.**

Замыкающий контакт Y1 во второй цепи останется разомкнутым, и катушка реле **Motor,** имитирующего исполнительное устройство некой системы управления, не сработает.

Схема 14 позволяет поставить еще один интересный опыт. Для этого снова запрограммируем LD-диаграмму, заменив R_TRIG **F1** и **F2** на **TP**-таймеры. Сначала временные уставки по входам PT этих таймеров сделаем равными. Например, по $T_1 = T_2$=50 мс.

Переводим схему в режим эмуляции. Производим **Pusk.** Как и следовало ожидать, схема сработает по вышеописанному алгоритму. Действительно, **TP**-таймер при малых временных уставках подобен детектору переднего фронта входного сигнала R_TRIG.

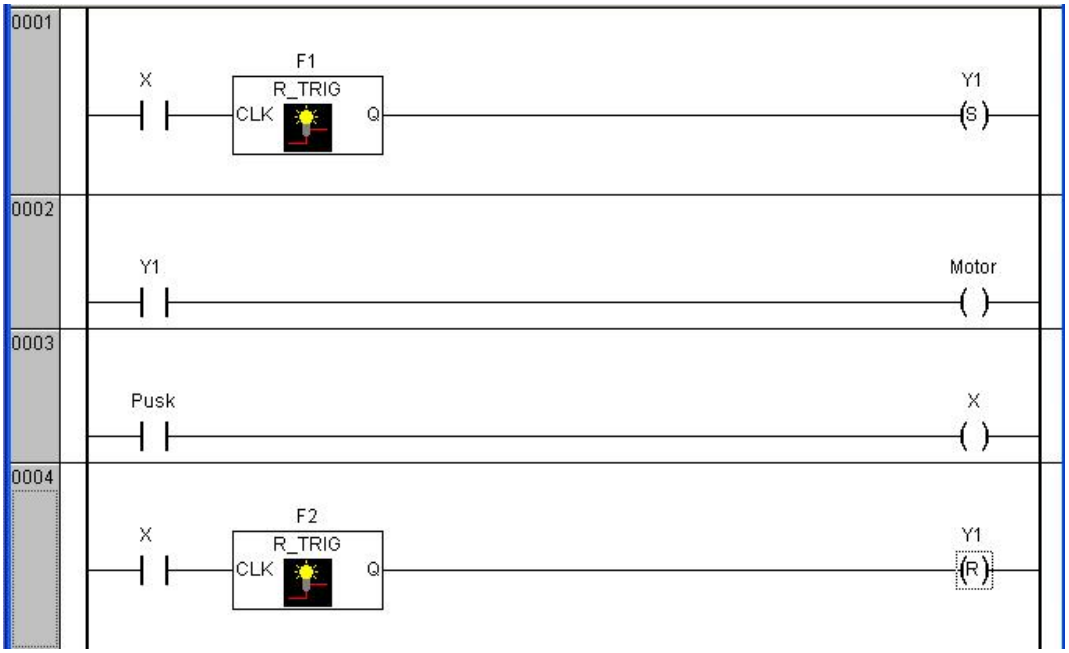


Рисунок 13 – Опытная LD-программа №1

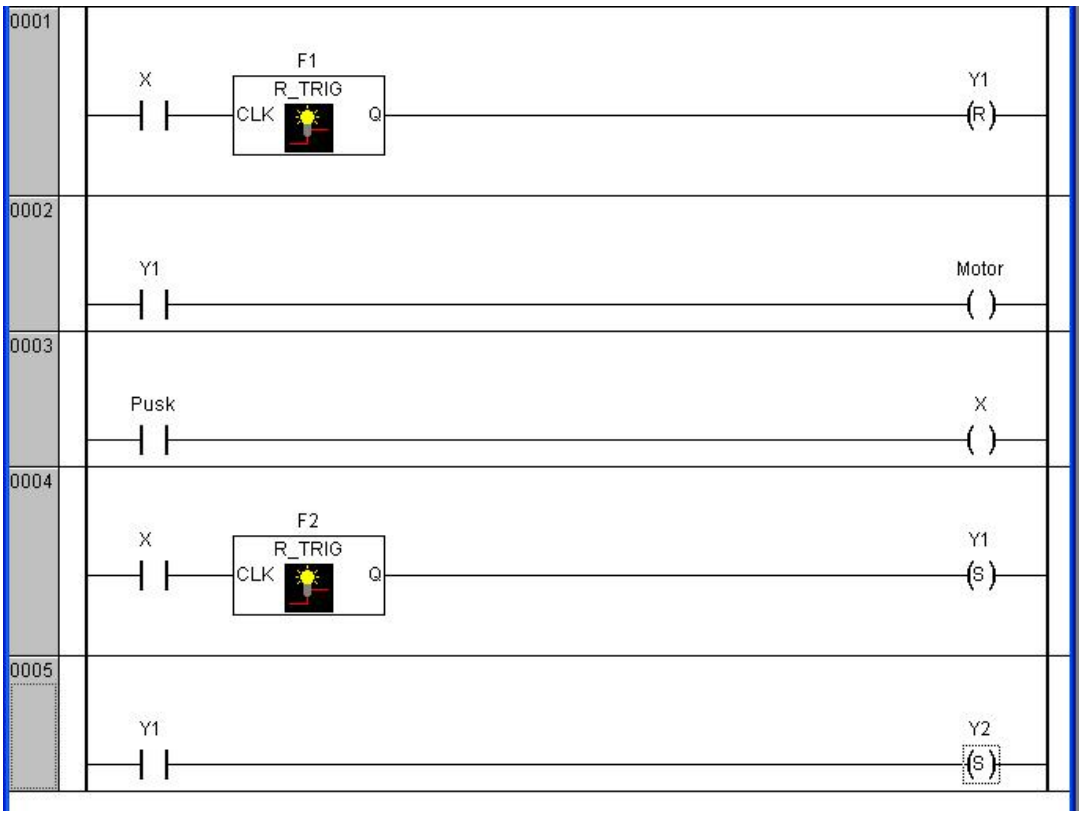


Рисунок 14 – Опытная LD-программа №2

Но в этом случае легко заметить кратковременные срабатывания реле Y1 и нет необходимости в пятой цепи.

Далее начинаем увеличивать на 5-10 мс уставку второго **TP**-таймера в четвертой цепи. После каждой коррекции повторяем **Pusk.** Опыт продолжаем до тех пор, пока не сработают все цепи. Что же произошло в этом случае?

Как только разница во временных уставках T_1 и T_2 первого и соответственно второго таймера превысит продолжительность цикла сканирования, наблюдается следующая картина.

После команды **Pusk** срабатывает катушка реле X (третья цепь) и своим контактом X в четвертой цепи запускает **TP**-таймер, который в течение времени T_2 будет держать «под напряжением» катушку **S** реле Y1.

Цикл сканирования на этом заканчивается, и начинается новый прогон программы. Срабатывает контакт X в первой цепи, запускается **TP**-таймер, который в течение времени T_1 будет держать «под напряжением» катушку сброса **R** реле **Y1**.

Если период T_2 еще не закончился, то наступает пауза запрещенного состояния для **Y1**: на его катушки **S** и **R** одновременно поступают сигналы, равные логической единице. Но если эта пауза не превышает продолжительность цикла, то заметить на мониторе некорректность предложенной программы нельзя. Рассмотренный пример позволяет оценить продолжительность прогона программы, т. к. минимальная разность времени $T_2 - T_1$ при которой произошло срабатывание катушки **Motor,** будет равна удвоенному интервалу циклу сканирования (но только не самого ПЛК, а Windows эмулятора, привязанного к тактам системного таймера, используемого в работе с ПК).

Продолжим исследование таймеров. Соберем схему генератора с регулируемой длительностью импульса и паузы, как показано на рисунке 15.

Собственно сам генератор собран на двух таймерах, которым присвоены имена **Impulse** и **Pausa,** и реле X**.**

Первая цепь предназначена для пуска и остановки генератора.

Реле X является отдаленным аналогом поляризованного двухобмоточного электромагнитного реле. Поэтому эти «обмотки» в LD расположены в разных цепях, но имеют одинаковый идентификатор **X**.

В первую цепь введен таймер TON, чтобы при исследовании схемы в режиме эмуляции после команды PUSK с экрана монитора исчезло окно меню Online, частично закрывающее многоступенчатую схему, и исследователь мог в течение 5 с «собраться с мыслями».

Таймер с именем «Impulse» определяет длительность импульса, а таймер «Pausa» – длительность паузы. Естественно, можно присвоить и другие имена и уставки по времени.

Теперь можно собрать схему для исследования таймеров (рисунок 16).

Первая цепь служит для запуска уже известного нам генератора, занимающего 2-ю и 3-ю цепи. Кнопки Dl, D2 и D3 для поочередного подключения исследуемых таймеров TON, TOF и TP. Факт срабатывания таймеров можно наблюдать в режиме эмуляции за изменением состояний катушек реле Y1, Y2 и Y3. Временные параметры генератора (T1 – длительность импульса, T2 – паузы) и исследуемых таймеров (T3, T4, T5) нужно изменять.

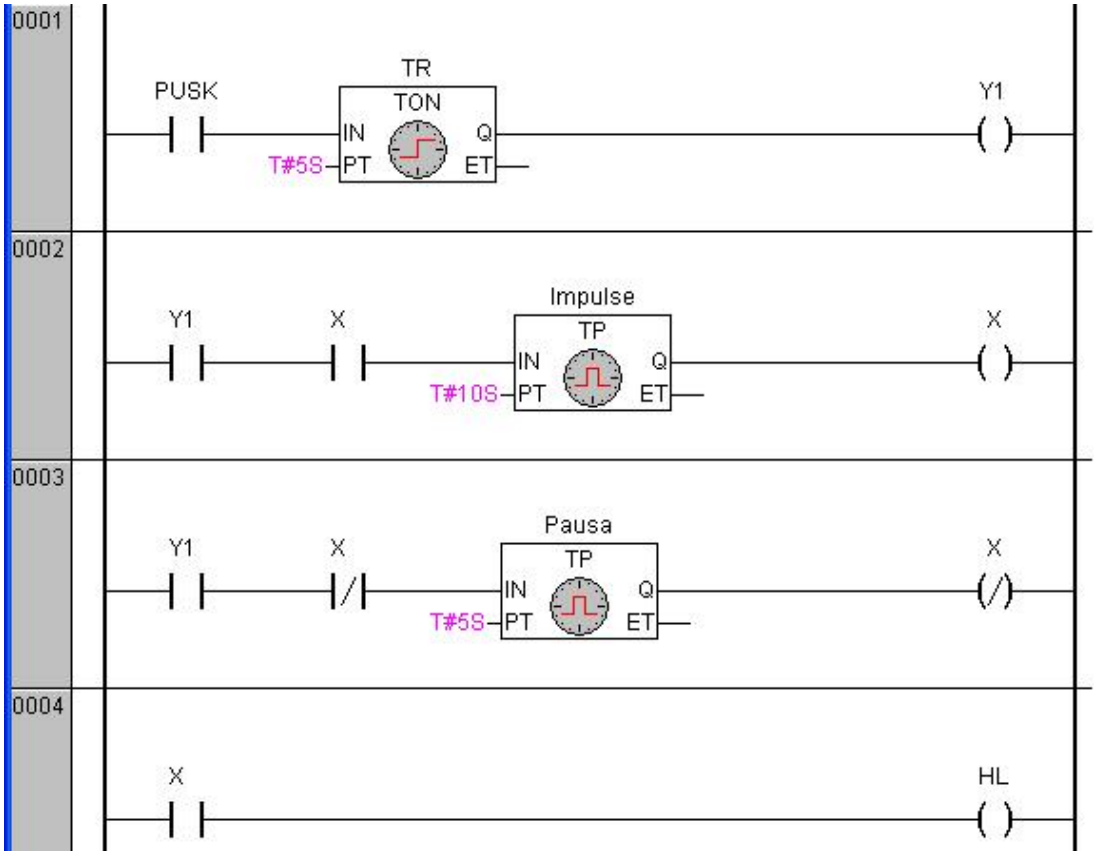


Рисунок 15 – Генератор импульсов

Естественно, временные параметры следует задавать в секундах, чтобы

заметить реакцию катушек Yl, Y2 и Y3.

Например, установим T1 = 5 с, T2 = 3 с, T3 = 8 с. Включаем PUSK и кнопкой D1 таймер TON. Таймер не сработает, т. к. T1 < T3 (рисунок 7).

Устанавливаем T1 = 8 с, T2 = 3 с и T3 = 8 с. Таймер также не сработает, т. к. T1 = T3.

Новая установка: T1 = 10 с; T2 = 3 с; T3 = 8 с. Таймер TON сработает, т. к. T1>T3, и Y1 начнет «мигать» синим цветом, что можно показать на временной диаграмме (рисунок 17).

2.3 Программирование счетчиков

При реализации многих технологических процессов необходимо вести подсчет неких событий. Например, считать количество деталей, перемещаемых конвейером для последующей упаковки в тару. Для решения подобных задач в ПЛК применяются различные виды счетчиков: CTU инкрементный, CTD декрементный и CTUD инкрементный/декрементный счетчики.

Соберем простую схему с CTU-счетчиком (рисунок 18).

После переноса счетчика в цепь многоступенчатой схемы появятся ??? над блоком, на входах **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).

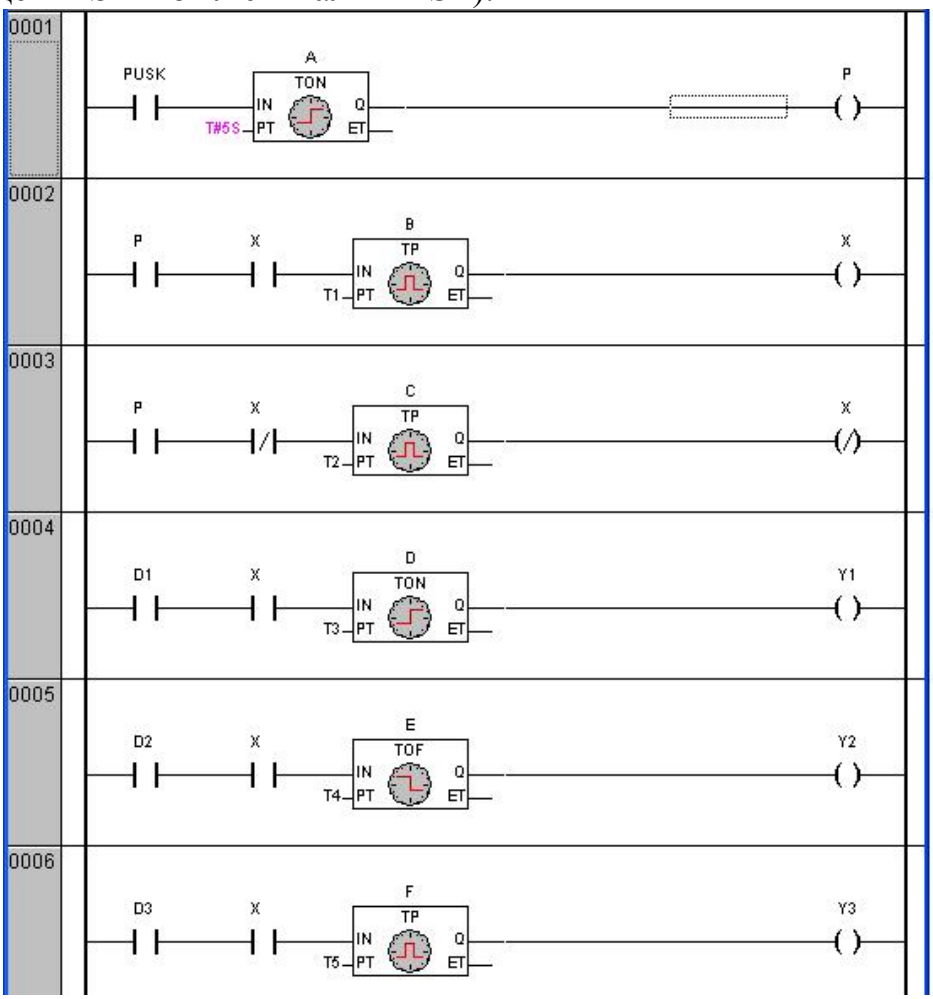


Рисунок 16 –Программа для исследования таймеров

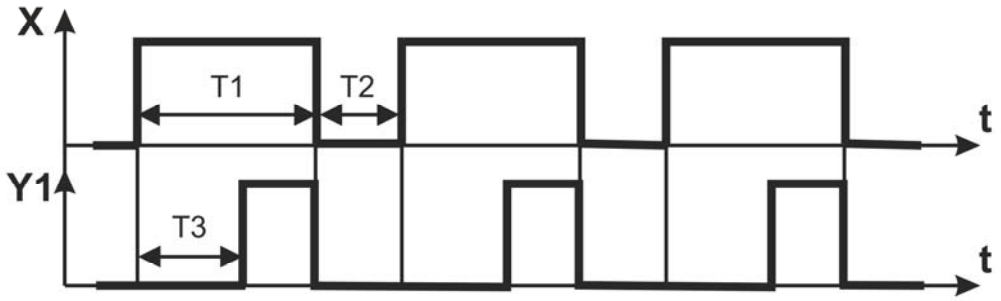


Рисунок 17 – Временные диаграммы работы TON-таймера D

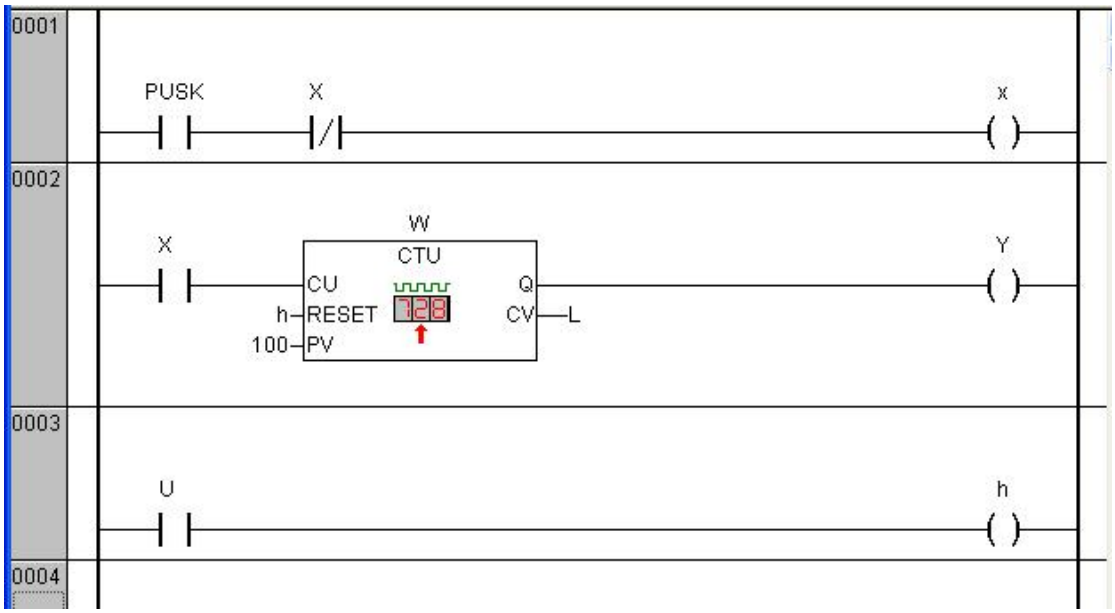


Рисунок 18 – Программа с CTU-счетчиком

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

С выхода CV можно в снимать значение накопленных сигналов. Для этого по полной аналогии с приемами активации выхода ET у таймеров активируем выход CV, присваиваем имя и т. д. Но в окне **Input** assistant (рисунок 12) выбираем тип переменной WORD.

Теперь в режиме эмуляции при запуске этой схемы на выходе CV будет в нарастающем порядке высвечиваться количество поступивших на данный момент импульсов на вход CU.

В программе (рисунок 18) счетчику присвоено имя **W**, входу **RESET** - имя реле **h**, выходу **CV - L** и задана уставка **PV** = 100.

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

Первая цепь содержит кнопку PUSK и генератор импульсов на реле X.

По каждому фронту сигнала, поступающему на вход CU, значение выхода CV возрастает на 1 и как только их сумма достигнет значения PV, выход Q переходит в TRUE, срабатывает реле Y. Но счет не останавливается! Для остановки счета и обнуления CV необходимо нажать кнопку U.

Кстати, эта схема также позволяет оценить время прогона программы. Достаточно секундомером замерить время от момента пуска генератора до момента срабатывания реле Y. Период импульсов генератора равен удвоенной длительности рабочего цикла. Количество же циклов известно и равно уставке PV. Вместо секундомера можно воспользоваться любым таймером с активированным выходом ET и уставкой PT, заведомо превышающей предполагаемое время накопления импульсов, количество которых задано но входу PV. Для этого следует создать дополнительную четвертую цепь (рисунок 19) и в момент срабатывания реле Y снять показания на выходе ET. (Катушка не потребовалась).

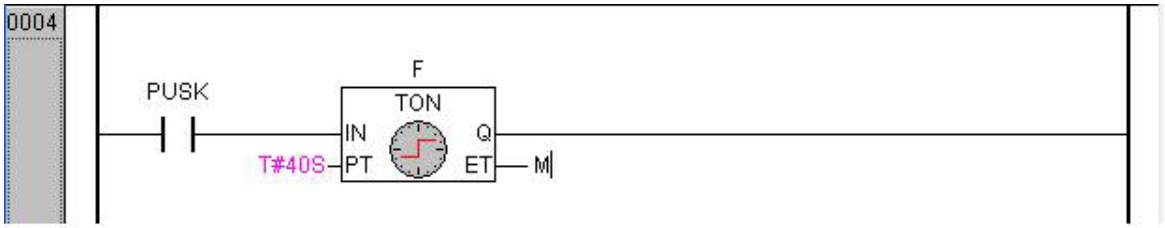


Рисунок 19 – Дополнительная цепь к рисунку 18

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

Важно отметить, что счетчик **CTD** загружается значением уставки, равным **PV,** только когда на входе **LOAD** есть сигнал **TRUE**.

Создайте программу с **CTD**-счетчиком (рисунок 20).

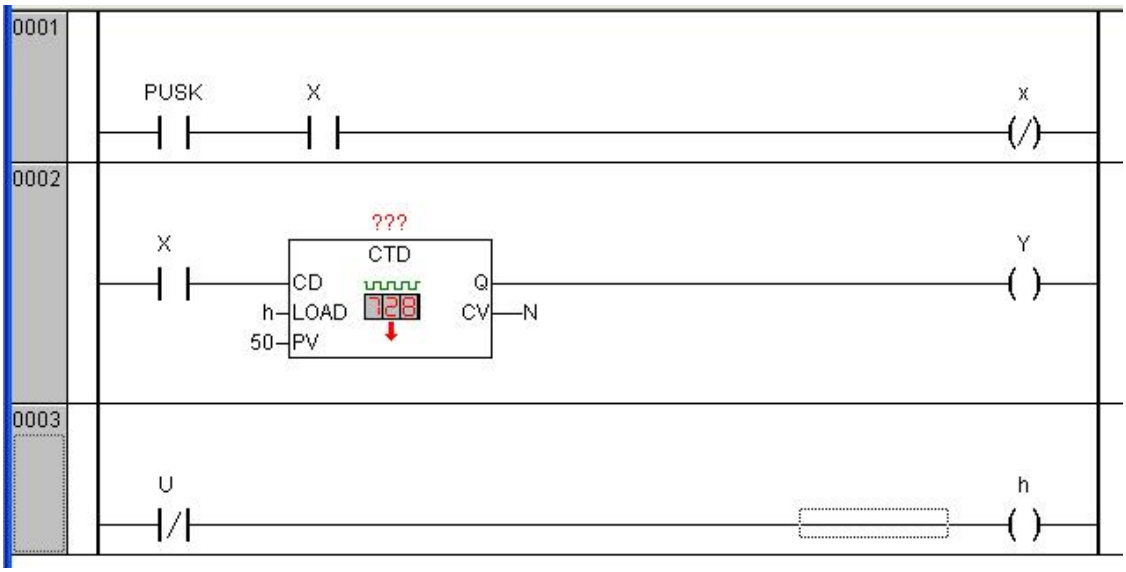


Рисунок 20 – Программа с CTD-счетчиком

Кнопка **PUSK** запускает генератор **X**. Счетчику присвоили имя **Q**, входу **LOAD - h**, входу **CV - N**.

По умолчанию принято, что реле, управляющие счетчиками **CTU** и **CTD** имеют на входах **RESET** и **LOAD** соответственно **замыкающий** контакт. В

программах (рисунках 18 и 20) это реле h. Но, учитывая отличие счетчика **CTD**, лучше в программе (рисунок 20) установить **размыкающий** контакт **U**. Тогда при включении схемы сразу сработает реле h и подаст сигнал TRUE на вход **LOAD**, что обеспечит загрузку выхода **CV** начальным значением **PV** (в данном примере 50). В этом положении имя переменной **h** на вход **LOAD** окрашено в синий цвет.

«Нажимаем» кнопку PUSK, начинает работать генератор X, на вход CD поступают импульсы. Но счетчик не активирован. На выходе Q имеем FALSE.

«Размыкаем» кнопку U. Реле h отключается, на вход LOAD приходит сигнал FALSE, активируется счетчик и начинается обратный отсчет на выходе CV. Как только CV = 0, счетчик остановится, на выходе Q сигнал становится TRUE, срабатывает реле Y.

Новый отсчет начнется после повторного замыкания и размыкания контакта U.

Для простых программ применение рассмотренных счетчиков практически равноценно. Дело вкуса проектировщика в выборе того или иного функционального блока FB. Кроме того, после срабатывания CTU-счетчика (Q = 1) счет импульсов по входу CU будет продолжаться, и это можно проконтролировать по выходу CV. В счетчике CTD после обнуления выхода CV все поступающие на вход CD сигналы будут потеряны.

CTUD - инкрементный/декрементный счетчик по фронту сигнала на накопительном входе CU увеличивается на 1. По фронту же сигнала на вычитающем входе CD уменьшается на 1 (вплоть до 0).

Если на входе RESET = 1, то счетчик CV обнуляется. По значению входа LOAD = 1 счетчик загружается значением, равным PV. На выходе CV по аналогии с вышерассмотренными счетчиками можно контролировать изменение результата счета.

Выход QU = 1, если CV ≥ PV, иначе QU = 0.

Выход QD = 1, если CV =0, иначе QD = 0.

3 ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

В данной лабораторной работе необходимо выполнить исследование работы таймеров и счетчиков в составе LD программы на примере элементов прикладной программы управления роботизированным технологическим комплексом (РТК) токарной обработки деталей (рисунок 21).

В состав РТК входят:

1 Токарный станок с ЧПУ №1

2 Токарный станок с ЧПУ №2

3 Промышленный робот

4 Накопитель №1

5 Накопитель №2

Обрабатываемые детали размещаются в накопителях №1 (поз.4) и №2 (поз.5). Загрузочно-разгрузочные операции выполняются промышленным роботом (ПР) (поз. 2).

Основными элементами технологического цикла работы РТК являются:
1 Поворот ПР к в зону загрузки токарного станка №1
2 Поворот ПР к в зону загрузки токарного станка №2
3 Поворот ПР к накопителю №1
4 Поворот ПР к накопителю №2
5 Зажим / разжим схвата руки ПР
6 Выдвижение / задвижение руки ПР
7 Обработка деталей по управляющим программам на станках №1 и №2
8 Подача накопителем №1 очередной заготовки и др.

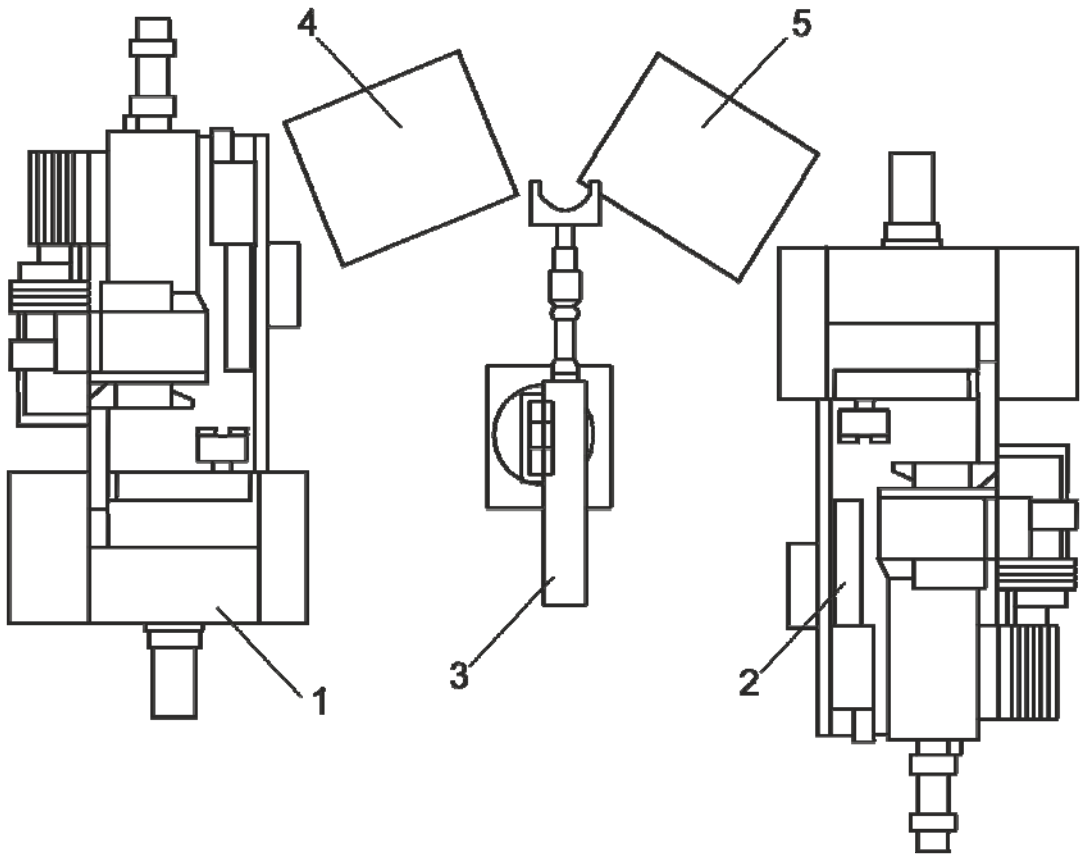


Рисунок 21 – Планировка РТК

Управление ПР и накопителями РТК выполняет программируемый контроллер ОВЕН ПЛК100-R.L.

Положение перемещающихся элементов технологического оборудования, управляемого контроллером, контролируется дискретными измерительными преобразователями перемещения. В качестве указанных датчиков в системе управления применяют контактные и бесконтактные путевые выключатели.

При выполнении лабораторной работы, студент должен из таблицы 3 выбрать исходные данные в соответствии с номером варианта, указанным преподавателем.

Таблица 3 – Исходные данные для лабораторной работы

№ варианта	№ временной диаграммы	T1, с	T2, с	T3, с	№ варианта	№ временной диаграммы	T1, с	T2, с	T3, с
1	1	10	10	15	7	1	14	14	13
2	2	12	10	8	8	2	19	10	15
3	1	14	14	11	9	1	17	9	9
4	2	19	17	13	10	2	14	16	12
5	1	10	16	12	11	1	19	11	15
6	2	20	10	20	12	2	17	16	20

Временные диаграммы №1 - №2, представленные на рисунках 22 – 23, являются фрагментами общей циклограммы работы РТК. На диаграммах показаны изменения входных сигналов с путевых выключателя SQ1 и BQ1, установленных на накопителях №1 и №2, и временные последовательности включения исполнительных устройств К1 – К6, подключенных к выходам ПЛК.

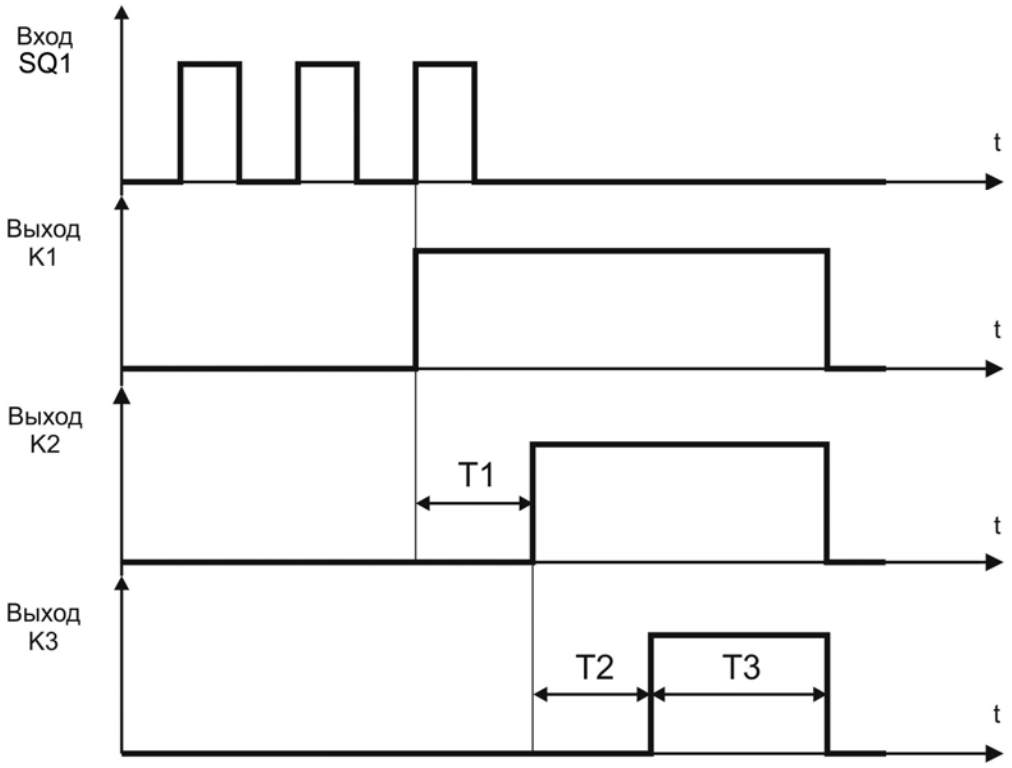


Рисунок 22 – Временная диаграмма №1

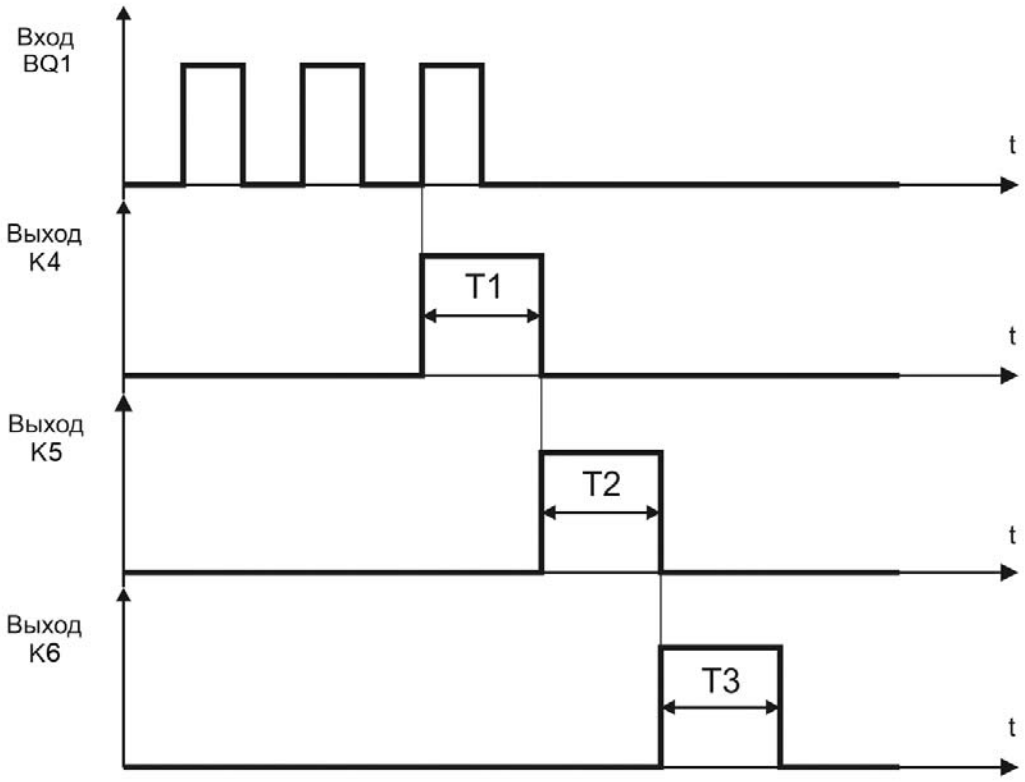


Рисунок 23 – Временная диаграмма №2

4 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторную работу необходимо выполнять в следующей последовательности:

1 Ознакомиться с характеристиками контроллера ПЛК100 и изучить основные сведения составе и возможностях программного комплекса CoDeSys.

2 Запустить на компьютере программный комплекс CoDeSys.

3 Пользуясь данными методическими указаниями, изучить методику программирования таймеров и счетчиков, отработать в среде CoDeSys все примеры, приведенные в разделе 2.

4 Получить у преподавателя вариант задания на разработку программы управления.

5 На основе анализа алгоритма управления определить состав входных и выходных дискретных сигналов.

6 По заданной временной диаграмме разработать в системе CoDeSys соответствующую LD-программу.

7 Исследовать разработанную программу в режиме эмуляции.

8 Выполнить конфигурацию входных и выходных переменных

9 Компилировать полученную LD-программу.

10 Выполнить подключение контроллера к компьютеру и подать напряжение питания на ПЛК.

11 Загрузить код разработанной программы в ПЛК.

12 Промоделировать на лабораторном стенде работу системы логико-программного управления. Для этого запустить программу в ПЛК. Имитируя с помощью переключателей и кнопок управления поступление в контроллер входных дискретных сигналов, проверить правильность реализации заданного алгоритма, наблюдая за состоянием индикаторных ламп, имитирующих дискретные исполнительные устройства.

13 Остановить выполнение программы в ПЛК и отключить питание.

14 Оформить отчет по лабораторной работе.

5 СОДЕРЖАНИЕ ОТЧЕТА

В отчете указывается цель лабораторной работы и приводятся следующие результаты:

1 Номер варианта и задание (исходные данные в виде временной диаграммы).

2 Таблицы входных и выходных сигналов с указанием наименования сигналов, их условных обозначений и источников (датчиков) и приемников (исполнительных устройств) сигналов.

3 Копия экрана PrtScr (Print Screen) с рабочей областью CoDeSys, в которой создана лестничная диаграмма (LD-программа), реализующая заданный алгоритм.

4 Копия экрана PrtScr (Print Screen) с рабочей областью CoDeSys в режиме эмуляции после отработки таймерами и счетчиками заданных уставок.

5 Выводы по результатам исследования программы в режиме эмуляции и моделирования работы системы управления на лабораторном стенде.

6 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Назовите основные технические характеристики контроллера ОВЕН ПЛК100.

2 Назовите основные элементы интерфейса программного комплекса CoDeSys.

3 Каковы основные типы таймеров используются в CoDeSys?

4 Поясните временные диаграммы таймеров TP, TON, TOF.

5 Каково назначение входов и выходов таймеров TP, TON, TOF?

6 Какая библиотека функциональных блоков должна быть подключена в проекте при создании таймеров и счетчиков?

7 Каковы основные типы счетчиков используются в CoDeSys?

8 Каково назначение входов и выходов счетчиков CTU, CTD и CTUD?

9 Поясните алгоритм работы счетчиков CTU, CTD и CTUD.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 http://www.owen.ru/catalog/73637995

2 http://www.3s-software.ru

3 http://www.owen.ru/catalog/48953113

4 Минаев И.Г., Самойленко В.В. Программируемые логические контроллеры: практическое руководство для начинающего инженера.– Ставрополь: АРГУС, 2009. – 100 с.

5 Петров И.В. Программируемые контроллеры. Стандартные языки и инструменты. – М.: СОЛОН-Пресс, 2003. – 256 с.

6 Сбродов Н.Б. Основы программирования контроллера ОВЕН ПЛК100 на языке LD: Методические указания к выполнению лабораторной работы для студентов специальности (направлений) 220301, 220400.62, 220700.62.– Курган: КГУ, 2012. – 25 с.

Сбродов Николай Борисович

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ В ИНСТРУМЕНТАЛЬНОЙ СИСТЕМЕ CoDeSys

Методические указания
к выполнению лабораторной работы
по дисциплине «Технические средства автоматизации»
для студентов очной и заочной форм обучения направления
220700.62 «Автоматизация технологических процессов и производств»

Авторская редакция

Подписано к печати 03.02.14	Формат 60x84 1/16	Бумага тип. №1
Печать цифровая	Усл. печ. л. 1,75	Уч.-изд. л. 1,75
Заказ 39	Тираж 45	Не для продажи

РИЦ Курганского государственного университета.
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет.