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

Кафедра «Автоматизация производственных процессов»

РАЗРАБОТКА ПРОГРАММ УПРАВЛЕНИЯ ДЛЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

для студентов направлений
15.03.04 «Автоматизация технологических процессов и производств»
27.03.04 «Управление в технических системах»

Курган 2017

Кафедра автоматизации производственных процессов.

Дисциплины: «Технические средства автоматизации»,
«Технические средства автоматизации и управления».

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

Утверждены на заседании кафедры 29 сентября 2016 г.

Рекомендованы методическим советом университета 17 декабря 2015 г.

ВВЕДЕНИЕ

Одним из основных средств автоматизации в локальных систем управления технологическими объектами являются программируемые логические контроллеры (ПЛК). Процесс разработки и отладки прикладного программного обеспечения для современных ПЛК происходит при помощи специализированных комплексов программ, обеспечивающих комфортную среду для работы прикладного программиста [1, 2].

Традиционно все ведущие производители ПЛК имеют собственные фирменные наработки в области инструментального программного обеспечения.

Одним из самых популярных в мире универсальных комплексов программирования является комплекс CoDeSys (Controllers Development System) фирмы 3S (Smart Software Solutions), который поддерживает стандарт МЭК 61131 и учитывает фирменные особенности более 150 моделей ПЛК.

Цель контрольной работы – изучения основных приемов работы в программной среде CoDeSys и приобретение практических навыков программирования контроллеров на языке лестничных диаграмм (LD).

1 ОСНОВНЫЕ СВЕДЕНИЯ О ПРОГРАММНОМ КОМПЛЕКСЕ CoDeSys

1.1 Назначение и возможности CoDeSys

Комплекс CoDeSys представляет проектировщику удобную среду для программирования контроллеров на языках МЭК [2 - 4]. Используемые редакторы и отладочные средства базируются на широко известных принципах.

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.

1.2 Установка программы CoDeSys

Многие производители ПЛК поставляют комплекс CoDeSys в качестве бесплатного приложения и размещают его на своих сайтах. Установочные файлы можно бесплатно скачать, например, с официального сайта компании ОВЕН [5].

Если дистрибутив скачен, то необходимо запустить программу установки и просто соглашаться с тем, что она предлагает. При таком подходе все необходимое будет установлено на персональный компьютер в папки, заданные по умолчанию. Вместе с бесплатной системой программирования CoDeSys установятся дополнительные программные компоненты в формате демонстрационных версий. В контрольной работе они окажутся невостребованными.

В конце установки операционная система может выдать сообщение, приведенное на рисунке 1. Смысл данного сообщения состоит в том, что те самые демонстрационные версии требуют покупки лицензий для полнофункционального использования. Сама система программирования CoDeSys 2.3 в отличие от дополнительных компонентов устанавливается компьютер без ограничений абсолютно бесплатно. Именно она нужна для выполнения контрольной работы. Поэтому нажав на кнопку «ОК», ждем завершения установки программы.

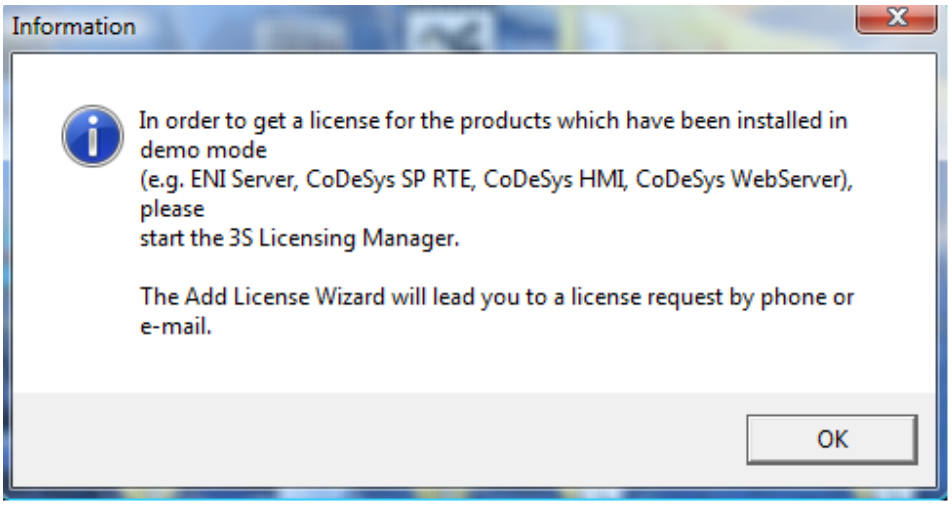


Рисунок 1 – Сообщение при установке CoDeSys

1.3 Запуск CoDeSys

После окончания установки на рабочем столе компьютера появится ярлык с тремя разноцветными шестиугольниками  для запуска. Можно воспользоваться им или выбрать в меню **Пуск** следующий путь **Все программы - 3S Software – CoDeSys 2.3**. Если ярлык на рабочем столе не появился, желательно содать его для удобства запуска системы программирования.

В меню **Файл** необходимо выбрать пункт **Создать**, либо немного ниже найти иконку  и воспользоваться ею. В появившемся окне **Настройки целевой** платформы ничего не меняем, нажимаем кнопку **ОК** (рисунок 2).

В следующем окне CoDeSys предлагает выбрать новый программный компонент и язык реализации.

Компоненты создают под прикладное программное обеспечения ПЛК. Компоненты организации программ **POU (Program Organization Unit)**

содержат функции, функциональные блоки и программы.

Рисунок 2 – Настройка целевой платформы

Выбор нужного **POU** производится в окне объявлений (рисунок 3) в строках **Программа**, **Функциональный блок** или **Функция**. Будем использовать только тип **Программа**, т. к. нам потребуются только стандартные компоненты.

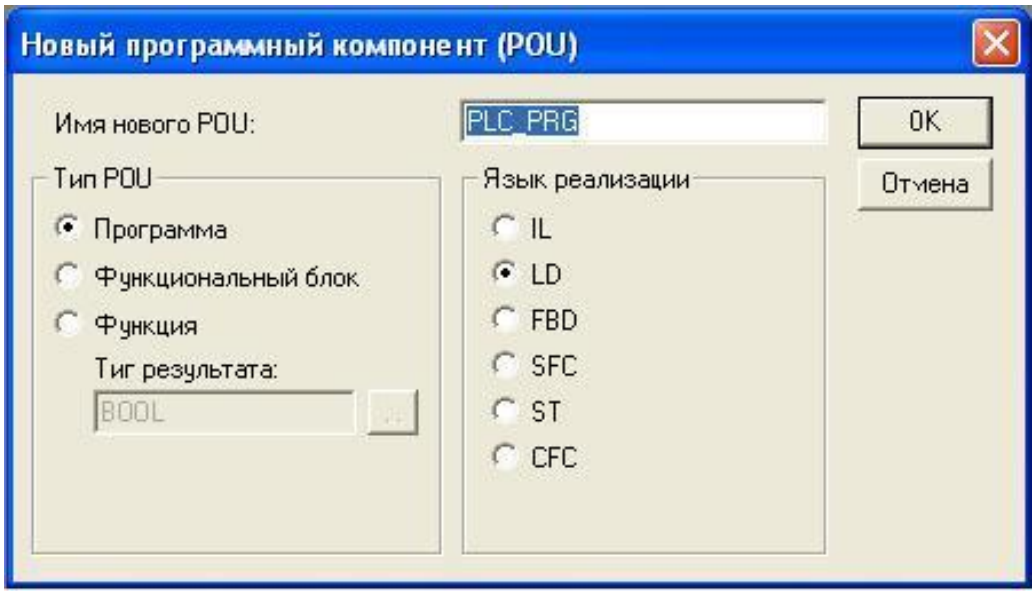


Рисунок 3 – Выбор языка программирования и задание имени программы

В появившемся окне **Новый программный компонент (POU)** выбираем тип **POU - Программа** и язык, на котором будет осуществляться написание программы - **LD**. Имя разрабатываемой программы можно оставить без изменения или изменить на другое. Подтверждаем выбор нажатием на кнопку **ОК**. После выполнения всех вышеописанных действий откроется главное окно (рисунок 4) среды CoDeSys.

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

- меню (рисунок 5);
- панель инструментов, которая содержит кнопки для быстрого вызова команд меню (рисунок 6);
- организатор объектов, имеющий вкладки «**POU**», «**Типы данных**», «**Визуализации**» и «**Ресурсы**»;

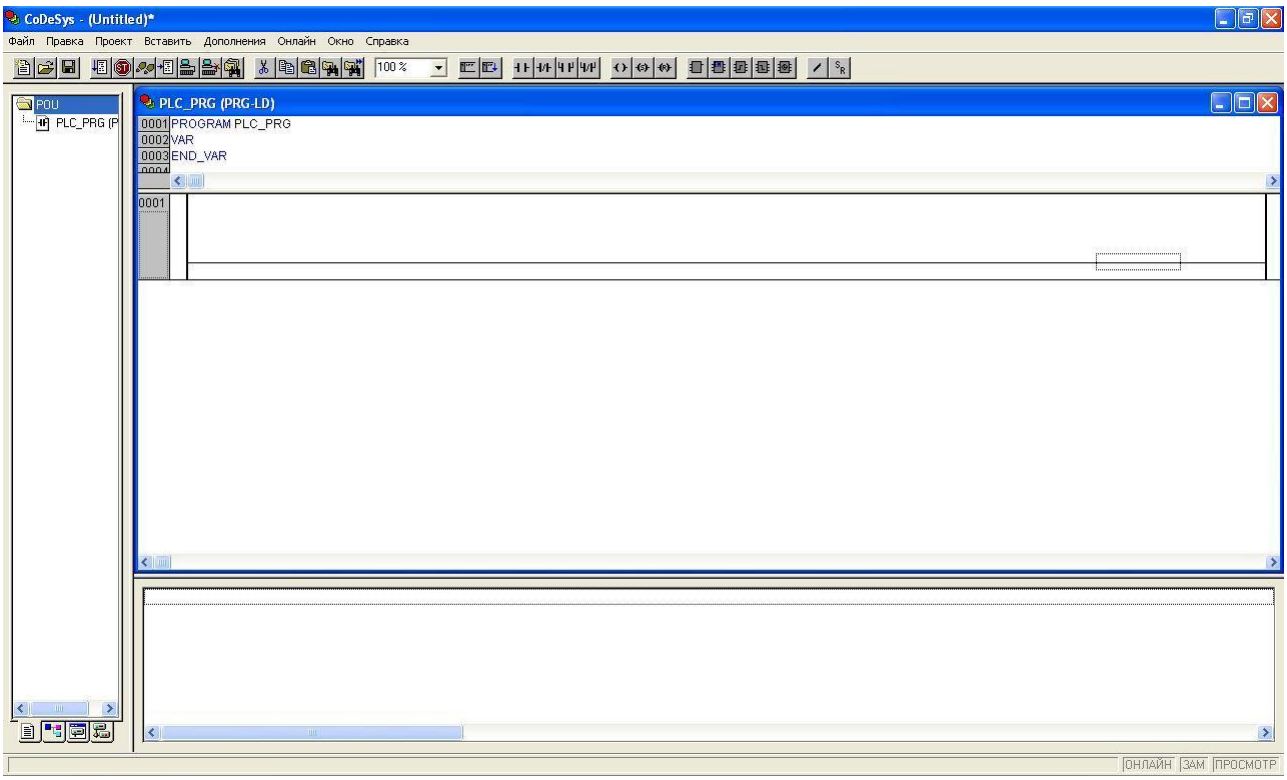


Рисунок 4 – Главное меню CoDeSys

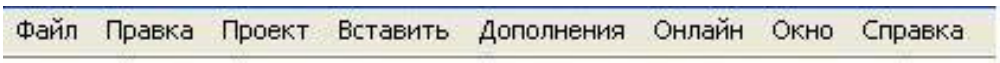


Рисунок 5 – Меню среды

Рисунок 6 – Панель инструментов

- 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.

Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню. Вызванная с помощью кнопки на панели инструментов команда автоматически выполняется в активном окне. Команда выполнится, как только нажатая на панели инструментов кнопка будет отпущена. Если вы поместите указатель мышки на кнопку панели инструментов, то через небольшой промежуток времени увидите название этой кнопки в подсказке. Кнопки на панели инструментов различны для разных редакторов CoDeSys. Получить информацию относительно назначения этих кнопок можно в описании редакторов. Кнопки на панели инструментов очень важны, и их использование

упрощает составление программы на языке LD. В других языках эта панель выглядит иначе. Назначение кнопок на панели инструментов приведены в таблице 1.

Таблица 1 – Кнопки на панели инструментов

Графическое изображение кнопки	Название	Назначение	Графическое изображение кнопки	Название	Назначение
	Создать	Создает новый проект		Цепь (вперед)	Вставляет цепь перед текущей
	Открыть	Открывает проект		Цепь (назад)	Вставляет цепь после текущей
	Сохранить	Сохраняет содержимое измененного объекта		Контакт	Вставляет последовательный замыкающий контакт
	Старт	Запускает ПЛК		Инверсный контакт	Вставляет последовательный инверсный (размыкающий) контакт
	Стоп	Останавливает ПЛК		Параллельный контакт	Вставляет параллельный замыкающий контакт
	Шаг по верху	Перешагивает через текущую инструкцию, даже если это вызов подпрограммы		Параллельный инверсный контакт	Вставляет параллельный инверсный (размыкающий) контакт
	Переключить точку останова	Уставливает/убирает точку останова в текущей позиции		Обмотка	Вставляет обмотку (катушку) «Реле»
	Подключение	Устанавливает связь с ПЛК и включает режим On-Line		«Set» обмотка	Вставляет Set обмотку (катушку)
	Отключение	Отключает режим On-Line		Reset обмотка	Вставляет Reset обмотку (катушку)
	Глобальный поиск	Ищет заданную строку по всему проекту		Функциональный блок	Вставляет функциональный блок
	Вырезать	Перемещает выделенную область в буфер обмена		Элемент с «EN»	Вставляет элемент со входом разрешения
	Копировать	Копирует выделенную область в буфер обмена		Детектор переднего фронта	Вставляет детектор переднего фронта
	Вставить	Вставляет содержимое буфера обмена в текущую позицию		Детектор заднего фронта	Вставляет детектор заднего фронта
	Найти	Ищет заданную строку в текущем окне		Таймер «TON»	Вставляет таймер «TON»
	Найти далее	Повторяет последний поиск		Инверсия	Инвертирует выбранный выход или вход
	Масштаб	Увеличение/уменьшение масштаба		Установка/сброс	Преобразует выход в Set/ Reset выход

При желании панель инструментов можно отключить (в меню - **«Проект»**, в выпадающем списке - **«Опции…»**, категория **«Рабочий стол»**, убрать галочку **«Панель инструментов»).**

Организатор объектов (рисунок 7) 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.

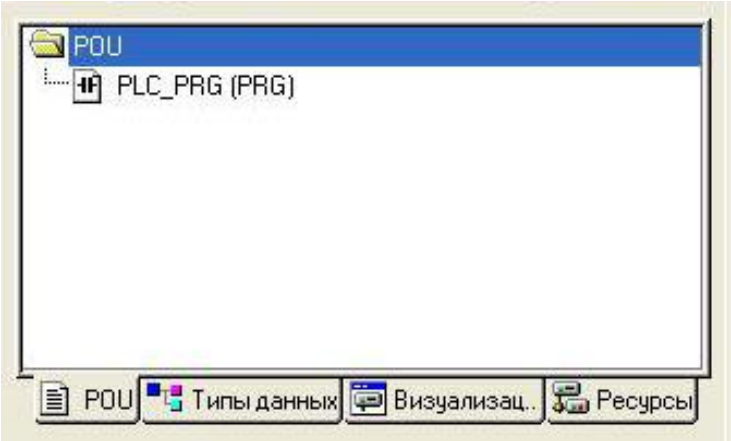


Рисунок 7 – Организатор объектов

1.4 Разделитель экрана

Разделить экрана - это граница между двумя непересекающимися окнами. В CoDeSys есть следующие разделители: между организатором объектов и рабочей областью, между разделом объявлений и разделом кода POU, между рабочей областью и окном сообщений. Вы можете перемещать разделители с помощью мышки, нажав ее левую кнопку.

Разделитель сохраняет свое положение даже при изменении размеров окна. Если вы больше не видите разделителя на экране, значит, стоит изменить размеры окна.

Рабочая область (рисунок 4) находится в правой части главного окна CoDeSys. Все редакторы, а также менеджер библиотек открываются именно в этой области. Имя открытого объекта находится в заголовке окна.

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 8). Это меню, вызываемое 1ПКМ на рабочей области, содержит наиболее часто используемые команды.

Рисунок 8 – Контекстное меню программы CoDeSys

2 РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ДЛЯ ПЛК НА ЯЗЫКЕ LD

2.1 Язык лестничных диаграмм LD

Язык лестничных диаграмм LD (Ladder Diagram) или релейно-контактных схем (РКС) - графический язык, реализующий структуры электрических цепей [1, 2]. В начале 70-х гг. XX века устройства управления, построенные на релейных элементах, начали постепенно вытесняться программируемыми контроллерами. Некоторое время те и другие работали одновременно и обслуживались одним и тем же персоналом.

Так появилась задача прозрачного переноса релейных электрических схем в ПЛК. Различные варианты программной реализации релейных схем создавались практически всеми ведущими производителями ПЛК. Благодаря простоте представления, язык LD обрел заслуженную популярность, что и стало основной причиной включения его в стандарт МЭК.

Графически LD-диаграмма (программа на языке РКС) представлена в виде двух вертикальных шин питания. Между ними расположены цепи, образованные соединением контактов. Нагрузкой каждой цепи обычно служит реле (обмотка). Каждое реле имеет контакты, которые можно использовать в других цепях. Количество контактов в цепи произвольно, реле одно. Если последовательно соединенные контакты замкнуты, ток идет по цепи и реле включается. При необходимости можно включить параллельно несколько реле, последовательное включение не допускается.

Важно понимать, что контакты, обмотки реле, цепи, шины питания и пр. в LD-диаграмме это не физические устройства, а элементы, которые реализованы **программно**.

В LD каждому контакту ставится в соответствие логическая переменная, определяющая его состояние. Если контакт замкнут, то переменная имеет значение ИСТИНА (логическая 1). Если разомкнут — ЛОЖЬ (логический 0). Имя переменной пишется над контактом и фактически служит его названием.

Последовательное соединение контактов или цепей равноценно логической операции И. Параллельное соединение образует монтажное ИЛИ.

Цепь может быть либо замкнутой (ON), либо разомкнутой (OFF). Это как раз и отражается на обмотке реле и соответственно на значении логической переменной обмотки (ИСТИНА/ЛОЖЬ или TRUE/FALSE).

Контакт может быть инверсным — нормально замкнутым. Такой контакт обозначается с помощью символа —┤/├— и замыкается, если значение переменной ЛОЖЬ. Инверсный контакт равнозначен логической операции НЕ.

Обмотки реле также могут быть инверсными, что обозначается символом **—(/)—**. Если обмотка инверсная, то в соответствующую логическую переменную копируется инверсное значение состояния цени.

Логически последовательное (И), параллельное (ИЛИ) соединение контактов и инверсия (НЕ) образуют базис алгебры Буля. В результате LD идеально подходит для программной реализации комбинационных логических

систем управления. Благодаря возможности включения в LD-программы функций и функциональных блоков, выполненных на других языках, сфера применения языка практически не ограничена.

2.2 Основные элементы программы

После открытия главного окна CoDeSys появляется на мониторе рабочая область (рисунок 4), в которой и будем изображать многоступенчатую схему (лестничную диаграмму).

Эта схема представляет собой набор горизонтальных цепей, напоминающих ступеньки лестницы, соединяющих вертикальные шины питания. Если необходимо увеличить размер рабочей области, то подводим курсор к кнопке □ в верхнем правом углу этой области и нажимаем 1ЛКМ.

Первая цепь появляется в рабочей области сразу (рисунок 2). Слева на сером фоне автоматически возникнет её номер: 0001. Наличие пунктирного прямоугольника в правой части цепи свидетельствует о том, что она активирована, т. е. готова принимать вносимые в неё компоненты: контакты, функциональные блоки FB, катушки реле.

Будем считать, что сама система логико-программного управления дискретным процессом в обычном, т. е. релейно-контактном исполнении, уже имеется, и наша задача перенести её в LD-диаграмму. В качестве такой системы управления возьмем схему, представленную на рисунке 9.

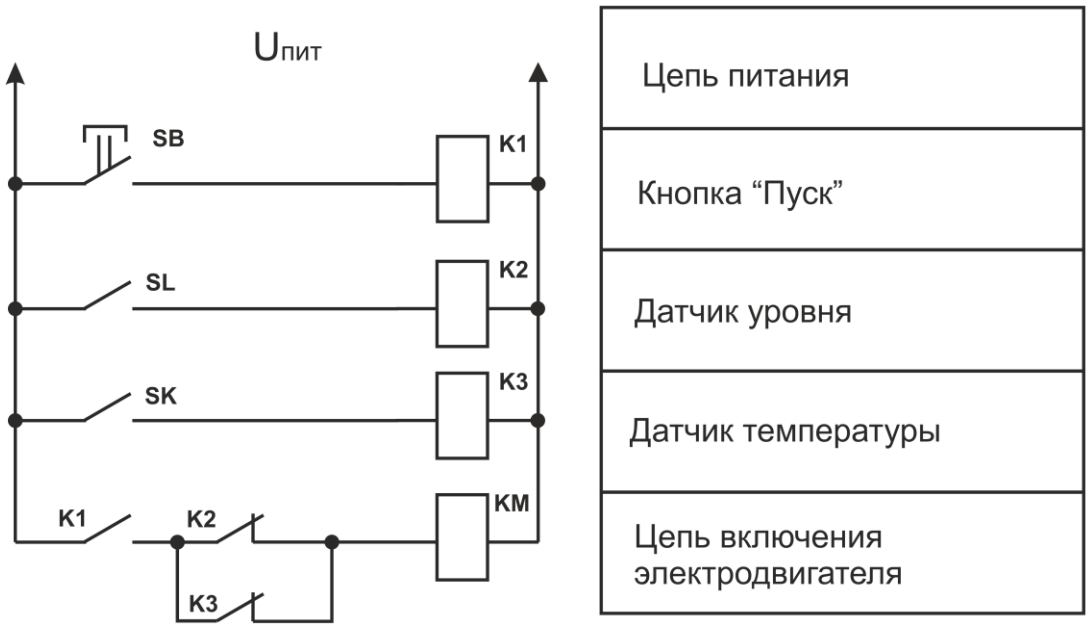


Рисунок 9 – Релейно-контактная схема системы логико-программного управления дискретным процессом

В состав схемы входят:

1 Три источника входных дискретных сигнала (кнопка управления «Пуск» **SB**, два электроконтактных дискретных (не аналоговых) датчика –

датчик температуры **SK** и уровня **SL**).

2 Три промежуточных реле **К1**, **К2**, **К3**.

3 Одно выходное устройство – обмотка магнитного пускателя **КМ**.

2.3 Методика программирования на языке LD в среде CoDeSys

По аналогии с исходным вариантом (рисунок 9) в многоступенчатой LD - программе, **скорее всего,** также потребуется четыре цепи. Словосочетание «скорее всего» использовано потому, что, несмотря на кажущуюся схожесть РКС и LD-диаграмм, есть принципиальное различие в последовательности срабатывания цепей.

В первую цепь необходимо внести контакт **SB** и катушку реле **К1**.

1 Наводим курсор на кнопку в панели инструментов с изображением замыкающего контакта  , нажимаем 1ЛКМ, и этот контакт появляется в цепи с тремя вопросительными знаками красного цвета (рисунок 10). Эти **???** запрашивают **имя** или **идентификатор** внесенного в цепь компонента.

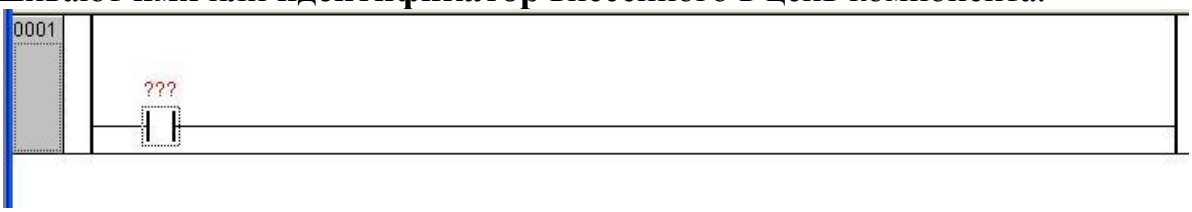


Рисунок 10 – Первая цепь с введенным контактом

2 Наводим курсор на **???**, щелкаем 1ЛКМ. Вопросы становятся белыми на фоне синего прямоугольника. С помощью клавиатуры на **английском** языке задаем имя. В нашем примере **«SB»**. Буквы **русского** алфавита использовать **нельзя!**

3 Нажимаем **«Enter»**. Открывается окно **Объявление переменной**, запрашивающее, к какому классу переменных будет отнесен наш элемент (рисунок 11).

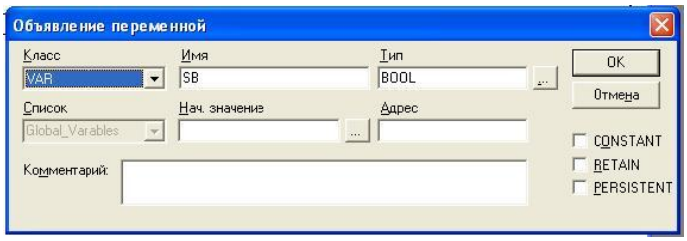


Рисунок 11 – Окно объявления переменной

Если проектируемая схема предназначена для учебных целей и будет работать только в режиме эмуляции, то можно сразу нажать 1ЛКМ на ОК, и имя появится над элементом (рисунок 12).

Нужно обратить внимание на пунктирный квадратик, охватывающий контакт SB, и на исчезновение пунктирного прямоугольника в конце самой

цепи. Это свидетельствует о том, что активирован сам контакт SB, т. е. можно (если была бы такая необходимость) последовательно и/или параллельно этому кон такту (как будет показано при создании четвертой цепи) подключать к нему другие контакты.

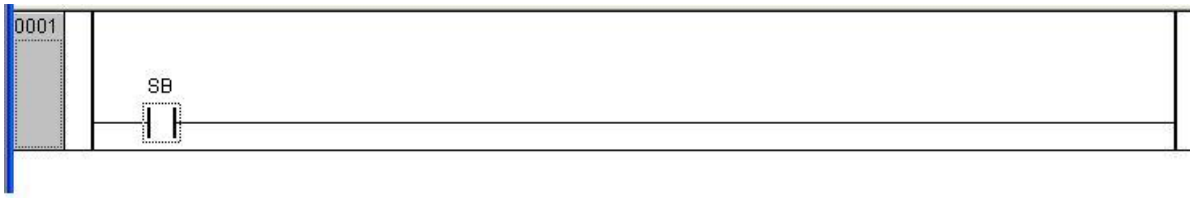


Рисунок 12 – Объявленный замыкающий контакт

Для завершения первой цепи необходимо ввести в нее катушку реле **К1**.

4 Наводим курсор на  , щелкаем 1ЛКМ и ... ничего не получилось.

Надо сначала активировать цепь. Для этого наводим курсор на линию цепи, щелкаем 1ЛКМ, т. е. активируем её (**появился пунктирный прямоугольник!**), переводим курсор на  , щелкаем 1ЛКМ и в цепи появляется катушка с тремя белыми ??? в синем прямоугольнике.

Катушки и, как будет показано ниже, функциональные блоки FB появляются всегда с такими ???. Если имя сразу не присвоить и перейти к другим действиям, то ??? станут красными, как и в случае включения контактов. Потом же потребуется лишняя операция при идентификации катушки: опять навести курсор на красные ???, щелкнуть 1ЛКМ, вопросы становятся белыми на синем фоне и т. д. Но это не принципиально.

5 Присваиваем катушке имя: **К1**. Нажимаем **Enter**, открывается окно **Объявление переменной,** нажимаем 1ЛКМ на ОК.

Можно сначала включить в цепь катушку, затем контакт. Можно включить эти элементы в любой последовательности и лишь потом присвоить им имена. Результат будет тем же.

Первая цепь завершена (рисунок 13). Как видно на мониторе, зона цепи ограничена двумя горизонтальными линиями. При активации цепи необязательно наводить курсор точно на цепь. Достаточно попасть в эту зону за исключением полосы, примыкающей к верхней линии и являющейся как бы продолжением строки, начинающейся с номера цепи. В нашем случае с 0001. Эта полоса может быть использована для задания на английском языке адресов меток переходов.

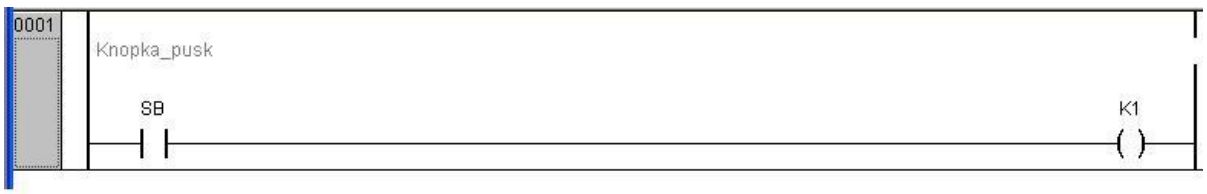


Рисунок 13 – Завершенная первая цепь

Для обозначения функционального назначения цепи в LD-диаграммах

часто задают комментарии на любом языке, понятном, естественно, обслуживающему персоналу.

С этой целью наводим указатель на полоску, расположенную ниже полоски меток и щелкаем 1ЛКМ, появляется мигающий курсор, с клавиатуры задаем, например, «Knopka_puska» и нажимаем **Enter**.

Комментарии можно выполнить и на русском языке. С этой целью наводим курсор на поле цепи, щелкаем 1ПКМ. Открывается контекстное меню (рисунок 8). Щелкаем 1ЛКМ на строке **Комментарий**. Слева над цепью появится слово **«Comment»,** в строке с которым можно написать на любом языке необходимые пояснения. Слово **«Comment»** потом можно удалить.

Перед тем как взяться за следующую цепь, обсудим ещё некоторые важные моменты.

Если реальное реле имеет ограниченное количество замыкающих, размыкающих и переключающих контактов, то в LD таких ограничений нет, и виртуальные контакты могут применяться в любой цепи в любом количестве.

Во всех цепях одной схемы имя логической переменной контактов одного и того же реле должно сохранятся. Имя может быть однобуквенным (**X, Y, Z** и т. д.), иметь цифровые индексы (**X1, X2** и т. д.), вписываемые без **пробела.** Если есть необходимость в таком пробеле, например, при написании двух или более слов, то в пробел ставится символ подчеркивания. Этот символ является значимым, т. е. имена **X_1**, **X1**, **_X1** и **_X_1** воспринимаются программой как самостоятельные. Но цифру на первое место ставить нельзя: **1X**, **2X** - неправильно!

Нельзя применять в качестве имен операторы других языков. Например, операторы языка инструкций **IL: LD, ST, S, R, AND, MUL, JMP** и др. С индексами, символами подчеркивания или другими буквами можно. Например **S1, RU, AND_, MULTI** и т. д. Даже не зная весь список операторов, легко установить ошибку. После нажатия на клавишу Enter в процессе присваивания имени переменной запрещенный идентификатор высветится синим цветом. Его необходимо удалить и вписать другое имя.

Регистр букв не влияет на работу ПЛК. Так, имена «**SET**» и «**Set**» воспринимаются одинаково.

В сложных схемах трудно запомнить назначение того или иного элемента при упрощенной (однобуквенной) системе идентификации. Поэтому имя переменной (т. е. идентификатор) можно записать в развернутом виде, не используя буквы русского языка.

Например, если есть трудности с английским языком, можно присвоить имена на русском языке **«Dvigatel», «pusk», «BLOKIROVKA»** и т. д.

Каждая цепь заканчивается обмоткой (катушкой) реле. Последовательно соединять катушки нельзя. Параллельно - можно.

Создаем вторую цепь.

Есть три способа: с помощью меню **«Вставка»** контекстного меню (рисунок 8) и кнопками  и  . Воспользуемся третьим приемом как наиболее простым.

1 Наводим курсор на , щелкаем 1ЛКМ. Сразу появляется вторая цепь (рисунок 14).

2 По вышеописанной методике вносим в цепь замыкающий контакт датчика **SL** и катушку **К2**. Если есть необходимость, то можно вписать комментарий: например, **«Datchik_urovnia»**.

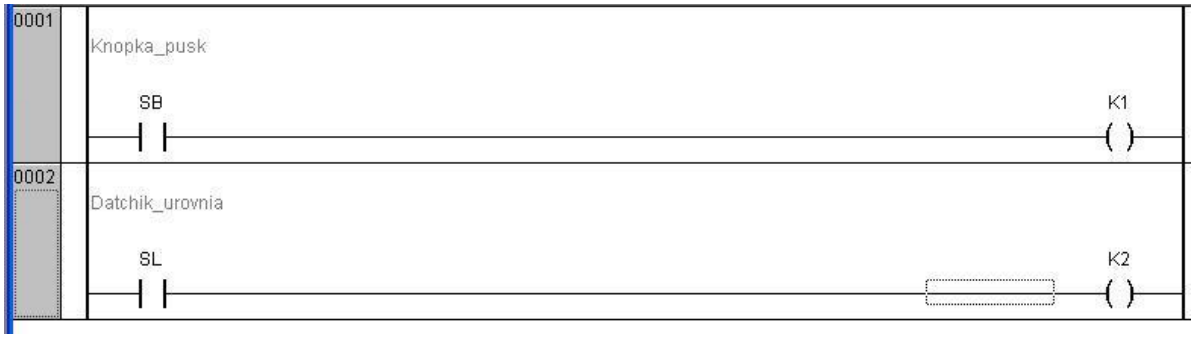


Рисунок 14 – Вторая цепь

Если по ошибке или преднамеренно нажали не , а , т. е. **Цепь (вперед),** то новая цепь станет впереди всех созданных и автоматически присвоит себе номер 0001, а последующим цепям нумерацию увеличит на единицу. Если же проектировщик решил вставить новую цепь между двумя уже созданными, то можно это выполнить разными приёмами:

- щелкнуть 1ЛКМ по цепи 0001, затем по , т. е. **Цепь (после).** Новая цепь станет после цепи 0001, получит номер 0002, а вторая ранее созданная цепь станет под номером 0003;

- щелкнуть 1ЛКМ по цепи 0002, затем по и новая цепь появится впереди цепи 0002, присвоит себе её номер, а ранее созданной цепи передаст номер 0003.

Создаем третью цепь, включая в её состав замыкающий контакт датчика температуры **SK** и обмотку **К3.**

Несколько сложнее будет программирование четвертой цепи.

По известной методике создаем четвертую цепь, последовательно внося в цепь замыкающий контакт **К1,** размыкающий **К2**. Теперь необходимо создать параллельную цепочку из размыкающего контакта **К3**. Для этого:

- наводим курсор на контакт **К1**, щелкаем 1ЛКМ. Контакт активирован, о чем свидетельствует пунктирный квадратик, охватывающий контакт;

- наводим курсор на , щелкаем 1ЛКМ и соединение выполнено. Присваиваем имя: **К3**.

Если актировать саму цепь, а не контакт **К2**, то замыкающий контакт **К3** охватил бы всю цепочку из контактов **К1** и **К2**, как показано на рисунке 15.

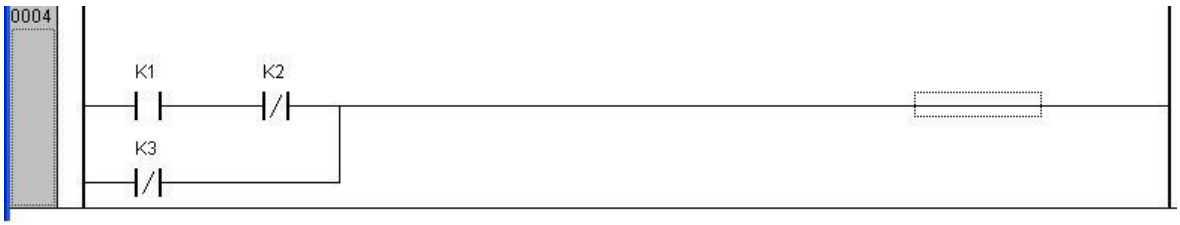


Рисунок 15 – Фрагмент программы после неудачного введения контакта К3

Вносим в цепь катушку **KM**. Четвертая цепь создана, и завершена вся многоступенчатая схема в виде LD-программы (рисунок 16).

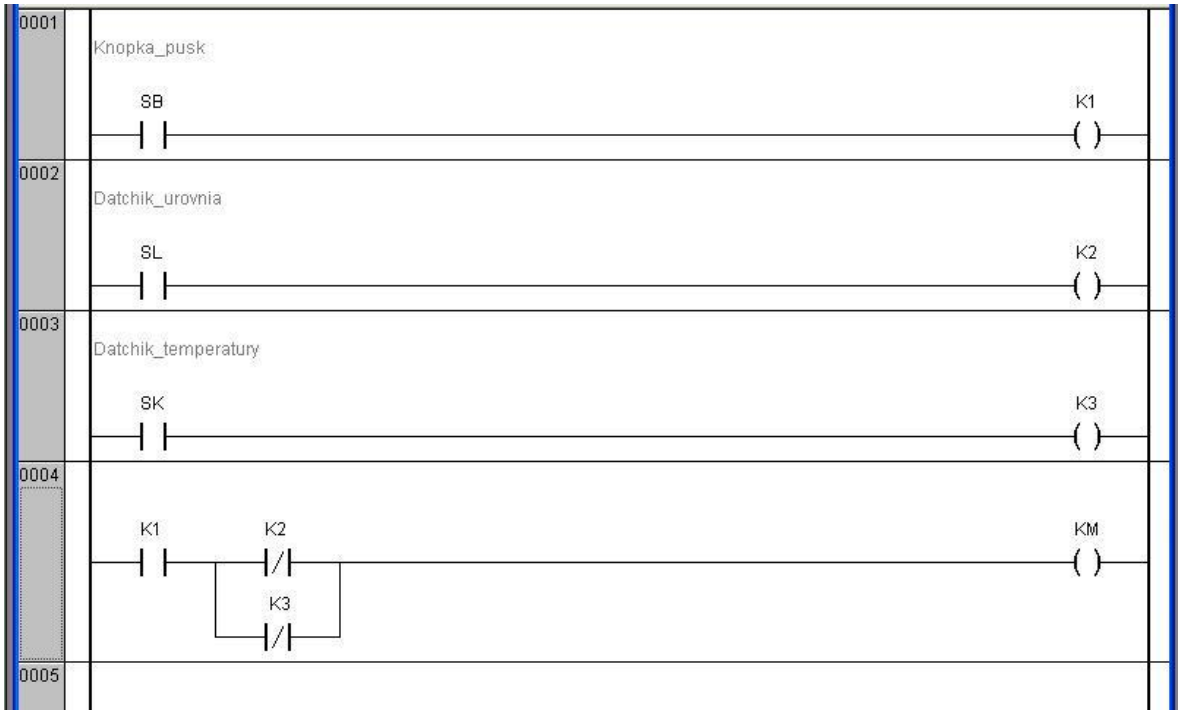


Рисунок 16 – Система логико-программного управления дискретным процессом в виде программы на языке LD

2.4 Дополнительные приемы при разработке LD-программ

Если по ошибке введен вместо замыкающего контакта размыкающий или наоборот, то можно навести на него курсор, щелкнуть 1ПКМ и в открывшемся контекстном меню (рисунок 8) щелкнуть 1ЛКМ на четвертой снизу строке **Инверсия** (**Negate**). Произойдет инверсия этого элемента. Имя, если оно уже было присвоено, остается прежним.

Эту операцию можно выполнить проще, если есть кнопка  . (В некоторых версиях CoDeSys она может отсутствовать). Тогда наводим курсор на подлежащий изменению контакт, щелкаем 1ЛКМ и нажимаем на указанную кнопку.

Если необходимо удалить какой-то компонент цепи или всю цепь со

всеми элементами, то наводим курсор на этот компонент (контакт, катушку, FB) или на саму цепь, щелкаем 1ПКМ и в открывшемся контекстном меню нажимаем 1ЛКМ на самой верхней строке **Вырезать** (**Cut**). Исчезнет элемент или вся цепь соответственно.

Если нужно изменить идентификатор какого-то компонента цепи, наводим курсор на это имя, щелкаем 1ЛКМ, нажатием на клавишу **Backspace** удаляем имя, задаем новое, нажимаем **Enter**.

Если возникла необходимость переместить какой-то компонент схемы по одной цепи или даже перекинуть его в другую цепь, то захватываем этот элемент курсором и при нажатой ЛКМ переставляем его в новое место. В момент перемещения на входе и выходе каждого элемента высвечиваются небольшие прямоугольники. Как только курсор подойдет к одному из них, прямоугольник становится зеленым. Отпускаем ЛКМ и перемещаемый элемент займет новое место.

Если по мере увеличения количества элементов в цепи им уже не хватает места в пределах границ рабочей области, то пользуясь горизонтальной «прокруткой» смещаем цепь влево. Если по мере добавления цепей им уже негде размещаться, то с помощью вертикальной «прокрутки» поднимаем всю многоступенчатую схему или уменьшаем масштаб.

2.5 Задание в программе выходных дискретных сигналов

Помимо «обычных» реле **—( )—** можно применять «инверсный» аналог, обозначаемый в программе **—(/)—**.

Это реле может иметь сколько угодно замыкающих и размыкающих контактов, но логика их действия противоположна поведению контактов обычного реле: при отсутствии тока в **—(/)—** замыкающий контакт —┤ ├— замкнут, размыкающий контакт —┤/├— разомкнут. При подаче питания в катушку **—(/)—** состояние его контактов меняется на противоположное.

В наборе программных компонентов имеются также специальные обмотки **SET** и **RESET**, обозначенные в линейке кнопок как **—(S)—** и **—(R)—**. С их помощью можно фиксировать условия управления исполнительным механизмом.

Если обмотка **S** «сработает», т. е. примет значение **ИСТИНА** (**TRUE**), то изменить это состояние на противоположное, т. е. **ЛОЖЬ** (**FALSE**), можно лишь с помощью обмотки **R** (рисунок 17).

Эта схема работает как классический **RS**-триггер: при кратковременном нажатии кнопки **PUSK** срабатывает катушка **S**, которой присвоим имя **Y1**, и своим контактом **Y1** включает нагрузку – катушку реле **Y2**. Выключить реле **Y2** можно только нажатием кнопки **STOP.** Одновременное нажатие на **PUSK** и **STOP,** как и в классическом RS-триггере недопустимо.

Следует заметить, что катушкам R и S присвоено одно и то же имя! В нашем примере **Y1**.

Эту же задачу самофиксации можно выполнить и на обычном реле (рисунок 18).

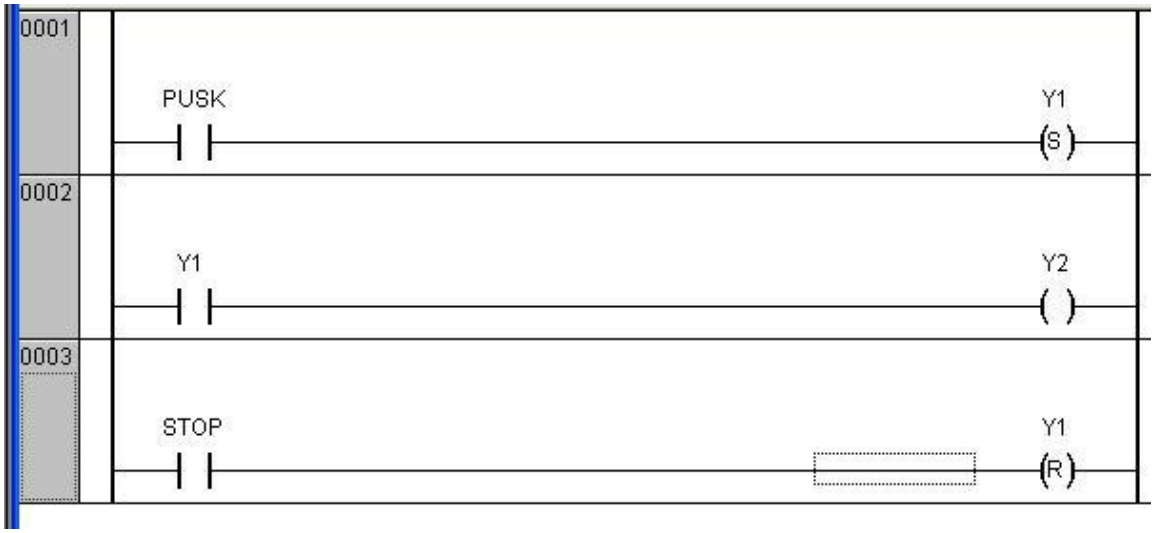


Рисунок 17 – LD-диаграмма фиксации катушки реле Y2 с помощью обмоток S и R

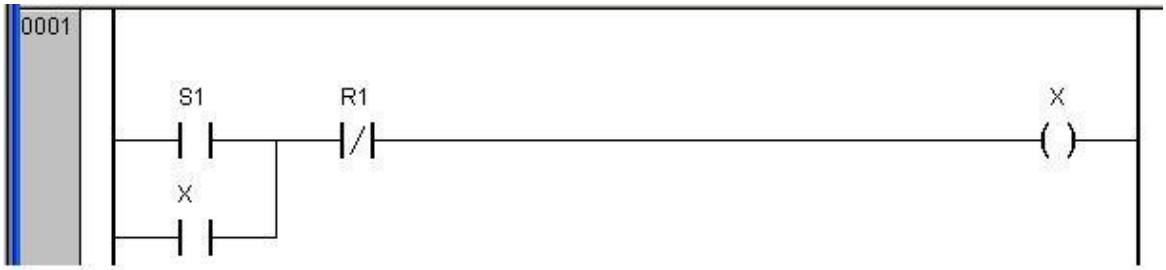


Рисунок 18 – Схема управления катушкой X самофиксацией состояния

При кратковременном нажатии на кнопку S1 происходит срабатывание реле **X**, которое своим контактом **X** фиксирует это состояние. Отключение реле **X** возможно только нажатием на кнопку **R1**.

Указанный прием самофиксации (самопитания или «самоподхват») широко используется на практике при организации логико-программного управления дискретными исполнительными устройствами.

2.6 Исследование LD-программы в режиме эмуляции

Итак, исходную РКС (рисунок 9) запрограммировали в CoDeSys на языке LD и представили её в виде многоступенчатой схемы (рисунок 16). Теперь необходимо проверить выполнение запланированных условий срабатывания и отсутствие ложных включений исполнительных элементов.

Все это можно выполнить в режиме эмуляции, не используя реальные аппаратные средства и сам ПЛК. Для этого:

- наводим курсор на **Онлайн** (**Online**) (рисунок 5), щелкаем 1ЛКМ. Открывается меню (рисунок 19);

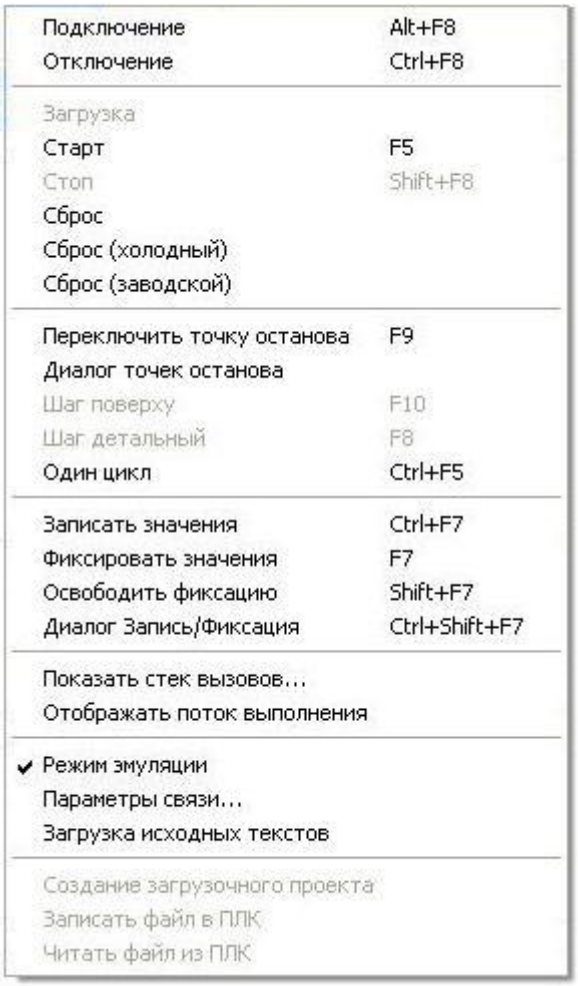


Рисунок 19 – Меню Онлайн

- смещаем курсор к строке **Режим эмуляции** (**Simulation Mode**), щелкаем 1ЛКМ. Перед этой строкой появится «галочка» √, что свидетельствует о готовности программы к «диалогу». И в дальнейшем при работе с этой программой даже в случае внесения в неё изменений, дополнений повторять эту операцию не надо;

- снова открываем это меню. Щелкаем 1ЛКМ по строке **Подключение** (**Login**). Левая вертикальная «шина питания», размыкающие контакты реле, участки цепей от этой шины до какого-то «непроводящего» в этом положении элемента цепи (например, замыкающего контакта) окрашиваются в синий цвет.

Возможна ситуация, когда после щелчка 1ЛКМ по **Подключение** будет выдано извещение о допущенной ошибке в программе. Возможно, забыли присвоить имя какому-то компоненту или снять ??? на входе того или иного функционального блока FB. В любом случае, не внеся требуемые коррективы, двигаться дальше нельзя. Будем считать, что в нашей схеме нет ошибок или мы их уже устранили;

- опять открываем меню **Онлайн** и щелкаем 1ЛКМ по строке **Старт** (**Run**). Программа готова к приему входных сигналов. Можно приступить к её

исследованию;

- начнем с комбинации входных сигналов, равных логическому нулю **SB** = 0, **SL** = 0, **SK** = 0, т. е. **SB, SL** и **SK** не сработали. Открываем меню **Онлайн** и щелкаем 1ЛКМ по строке **Записать значения** (**Write Values**). Убеждаемся, что катушки **К1**, **К2**, **К3**, **КМ** не сработали, о чем свидетельствует неизменившийся их вид;
- переходим ко второй комбинации. Должен замкнуться контакт кнопки управления **SB** (**SB** = 1). Наводим курсор на контакт **SB**, щелкаем 2ЛКМ. В полости этого контакта появится синий квадратик. Открываем меню **Онлайн** и щелкаем 1ЛКМ по строке **Записать значения**. Контакт **SB**, катушка **К1** и вся первая цепь зальются синим цветом. Контакт **К1** в четвертой цепи замыкается и также окрашивается в синий цвет. Размыкающие контакты **К2** и **К3** в этой цепи остаются замкнутыми, следовательно, срабатывает **КМ**.

Аналогично необходимо исследовать работу программы при всех других комбинациях входных сигналов с целью проверки правильности реализации алгоритма управления и выявления возможности ложных срабатываний исполнительных элементов.

Например, необходимо «разомкнуть» контакт **SB** и «замкнуть» **SK**. Для этого наводим курсор на **SB**, щелкаем 1ЛКМ. В полости контакта появится зеленый квадратик и останется часть синей заливки. Наводим курсор на **SK**, щелкаем 1ЛКМ. Контакт заливается синим цветом. Открываем меню **Онлайн,** щелкаем 1ЛКМ по строке **Записать значения**. Убеждаемся, что выключилась катушка **КМ**. И так проходим последовательно по всем строкам таблицы состояний входных и выходных устройств.

В случае исследования такой простой системы, как в нашем примере, можно сразу проверить только условия срабатывания исполнительных элементов, создавая «замыкание» и «размыкание» соответствующих входных элементов **SB**, **SL** и **SK** по вышеописанной методике.

Если, приступая к исследованию реакции LD-программы на следующую комбинацию состояний входных сигналов, необходимо вернуть схему в исходное состояние (**SB** = 0, **SL** = 0 и **SK** = 0), то для этого достаточно щелкнуть 1ЛКМ по строке **Отключение** (**Logout**), т. е. отключить режим **Онлайн.** Затем по освоенной уже методике запустить режим эмуляции, создать требуемую комбинацию состояний входных элементов и, нажав 1ЛКМ по строке **Записать значения**, зафиксировать поведение исполнительных элементов.

При большом количестве цепей многоступенчатой схемы нет возможности видеть ее на мониторе целиком даже при самом малом масштабе. Поэтому перед нажатием 1ЛКМ на строку **Записать значения** можно, пользуясь прокрутками, переместить интересующий нас участок схемы в поле зрения и лишь потом запустить ее.

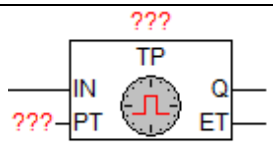
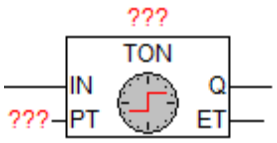
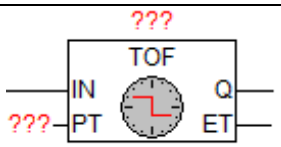
3 ПРОГРАММИРОВАНИЕ ТАЙМЕРОВ

3.1 Основные типы таймеров в CoDeSys

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

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.

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

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

Работу этих таймеров поясняют временные диаграммы входных и выходных сигналов (рисунок 20).

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 20 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 20 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 20 в).

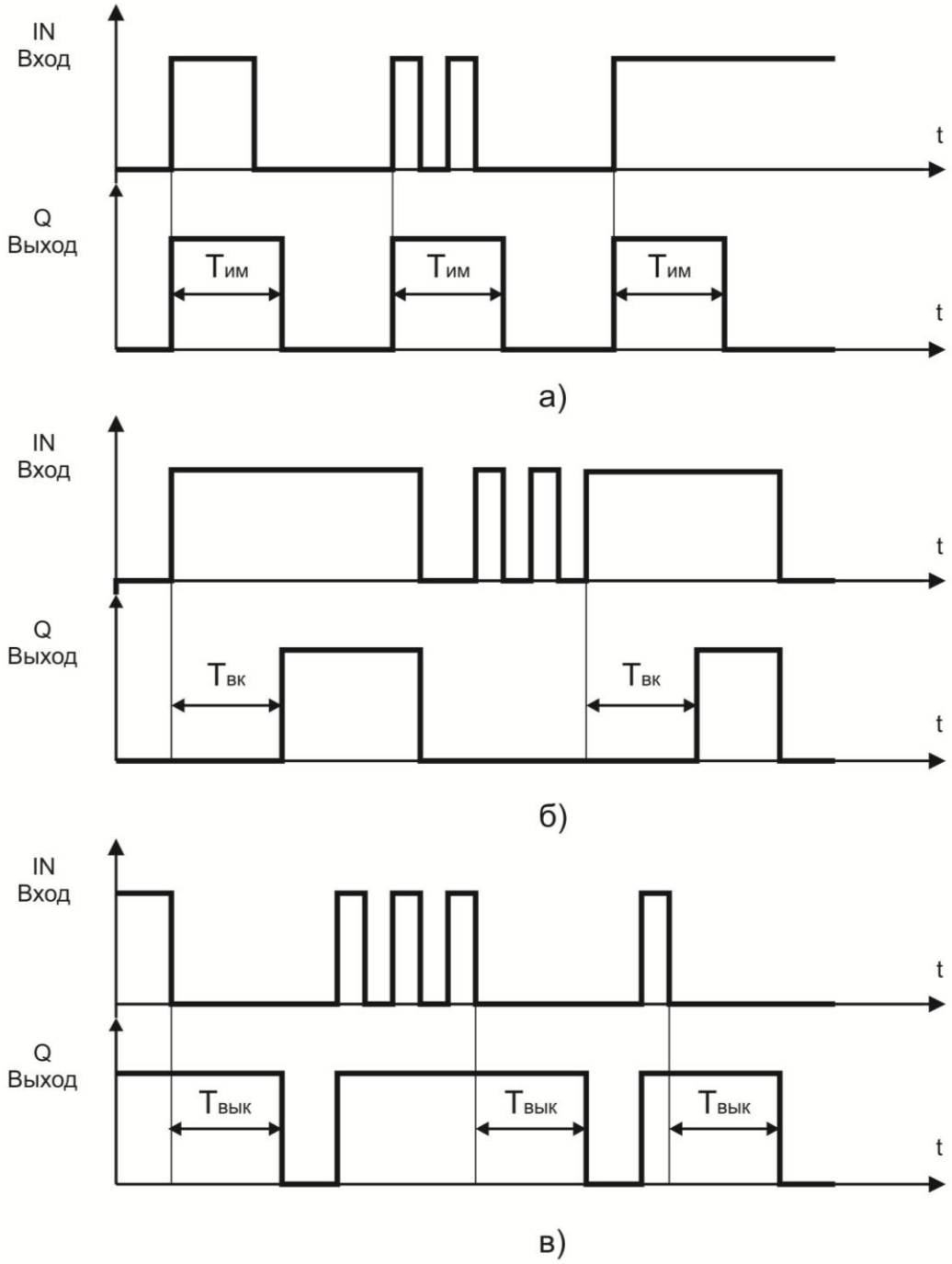


Рисунок 20 - Временные диаграммы таймеров

3.2 Пример LD программы с таймером

На рисунке 21 показаны временные диаграммы изменения значения двух входных сигналов SQ1, SQ2 и двух выходных сигналов КМ1, КМ2.

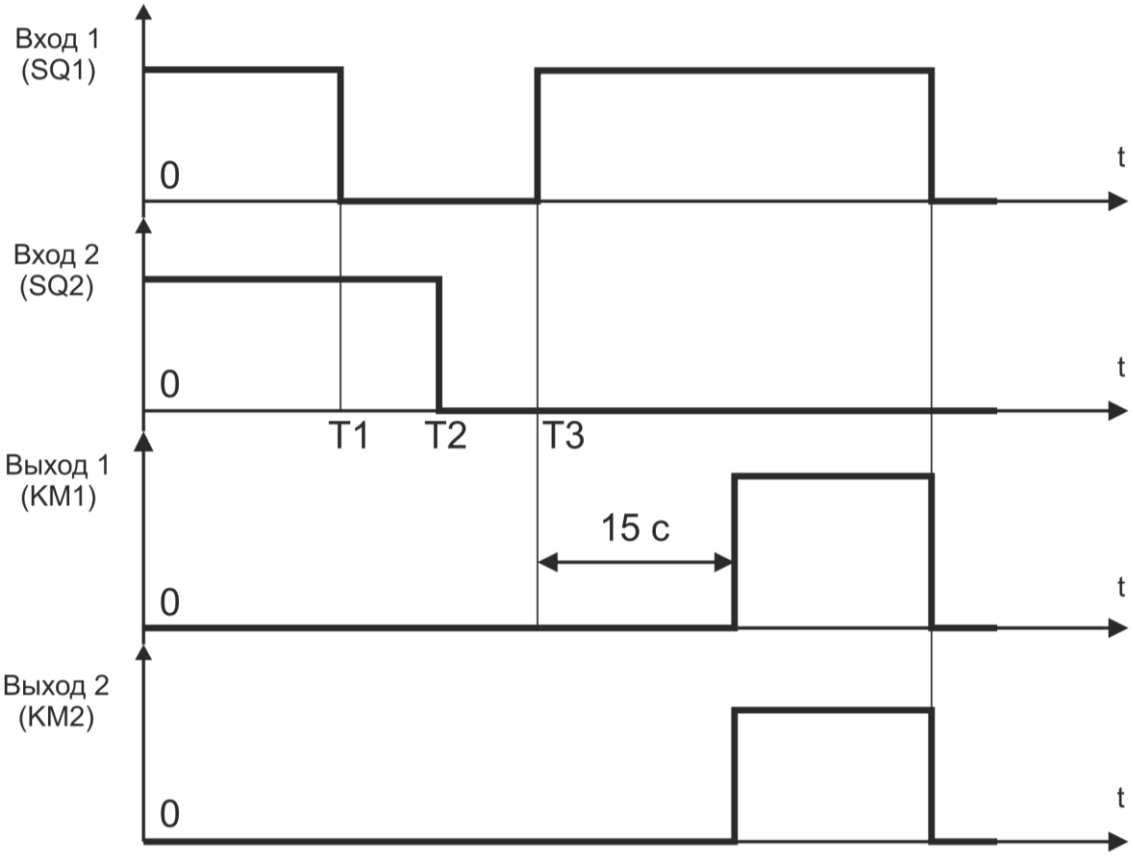


Рисунок 21 – Временные диаграммы

Оба выходных сигнала КМ1, КМ2 равны логическому 0 в следующие интервалы времени: от 0 до Т1, в течение которого входные сигналы SQ1 и SQ2 имеют высокий уровень (логическая 1), от Т1 до Т2, в течение которого сигнал SQ1 имеет низкий уровень (логический 0), а сигнал SQ2 – высокий уровень (логическая 1), от Т2 до Т3 – оба входных сигнала имеют низкий уровень. В момент времени Т3 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.

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 (рисунок 7) 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 22) выбираем 1ЛКМ строку **Добавить библиотеку**.

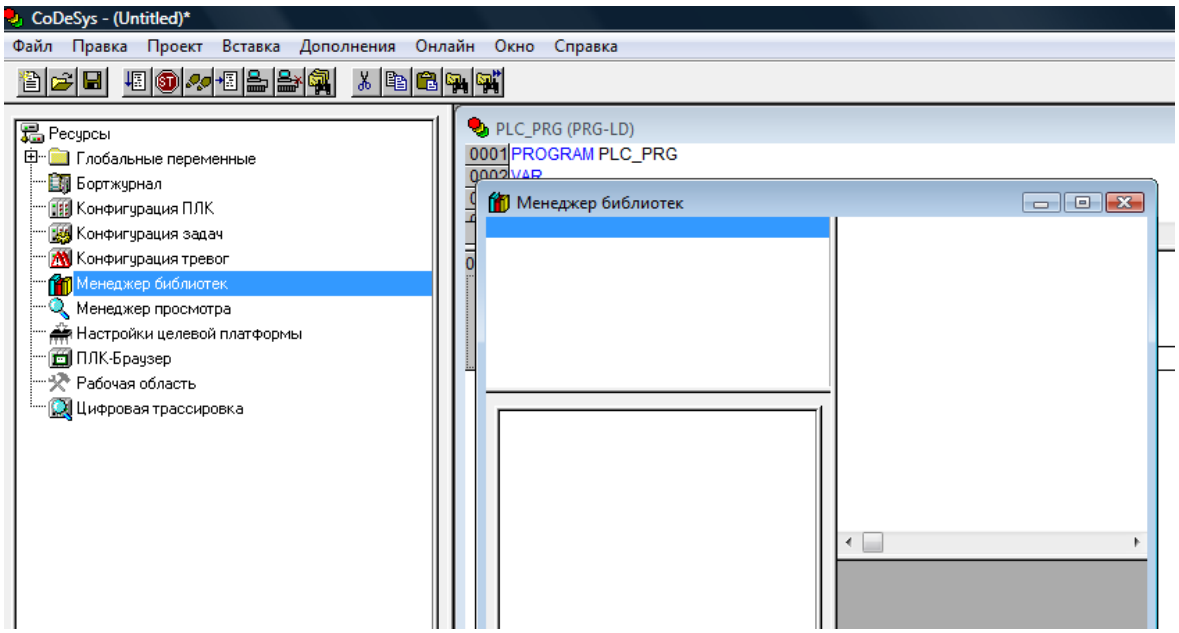


Рисунок 22 – Добавление библиотеки в проект

В окне **Открыть** выбираем и открываем файл **Standard.lib** (рисунок 23). Библиотека стандартных функциональных блоков загружена в проект. С помощью вкладки **POU** организатора объектов (рисунок 7) можно вернуться в проектируемую программу.

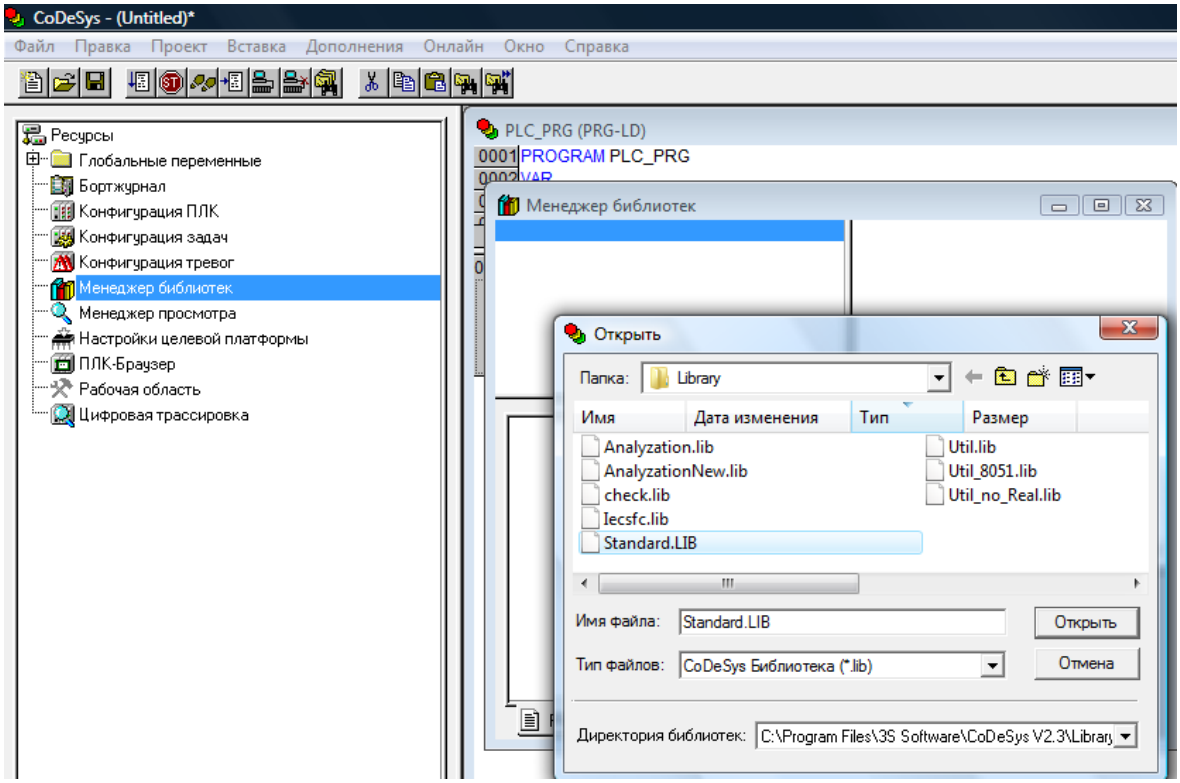


Рисунок 23 – Выбор библиотеки

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 24.

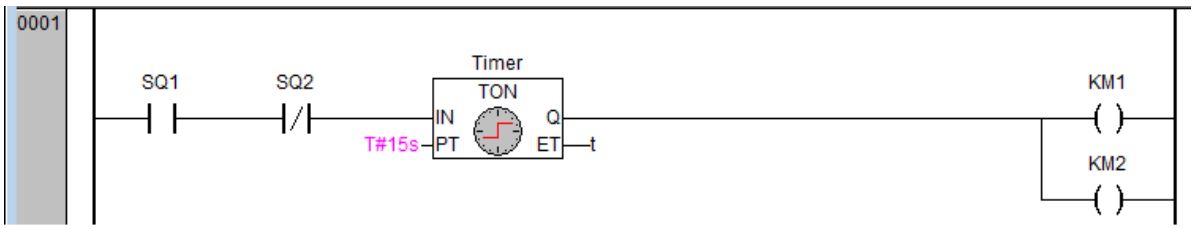


Рисунок 24 – Программа с TON-таймером

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 25). Щелкаем 1ЛКМ по 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 20 б) и исходной временной диаграммой (рисунок 21) сбрасывает значение выхода таймера Q и выключает выходы KM1 и KM2.

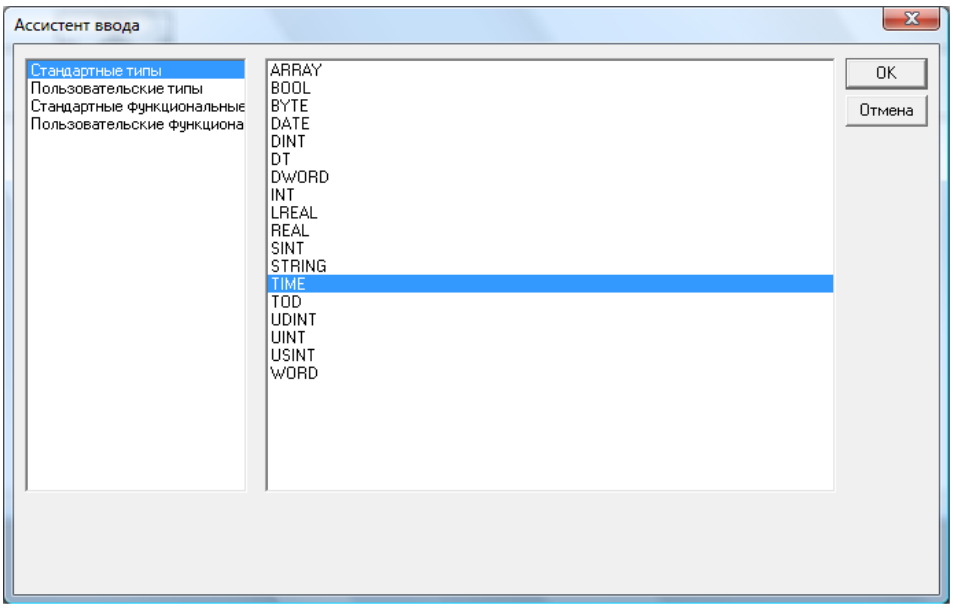


Рисунок 25 – Окно **Input assistant**

4 ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

4.1 Выбор задания

При выполнении контрольной работы, студент должен из таблицы 1 выбрать исходные данные в соответствии с номером варианта, указанным преподавателем.

Таблица 2 – Исходные данные для контрольной работы

№ варианта	№ временной диаграммы	T1, с	T2, с	№ варианта	№ временной диаграммы	T1, с	T2, с
1	1	10	10	**14**	2	14	14
2	2	12	12	**15**	3	19	11
3	3	14	14	**16**	4	17	9
4	4	19	19	**17**	5	14	14
5	5	10	16	**18**	6	19	11
6	6	20	10	**19**	1	17	9
7	1	18	21	**20**	2	14	14
8	2	22	12	**21**	3	19	11
9	3	15	14	**22**	4	17	9
10	4	25	19	**23**	5	14	14
11	5	8	16	**24**	6	19	11
12	6	13	10	**25**	1	17	9
13	1	21	21	**26**	2	14	14

Продолжение таблицы 2

27	3	15	12	**44**	2	19	11
28	4	10	18	**45**	3	15	19
29	5	12	22	**46**	4	16	16
30	6	14	15	**47**	5	17	10
31	1	19	25	**48**	6	18	21
32	2	16	8	**49**	1	19	12
33	3	10	13	**50**	2	20	14
34	4	21	21	**51**	3	15	19
35	5	12	15	**52**	4	11	21
36	6	14	18	**53**	5	22	20
37	1	19	22	**54**	6	18	17
38	2	16	15	**55**	1	10	21
39	3	10	9	**56**	2	15	15
40	4	21	14	**57**	3	22	18
41	5	12	11	**58**	4	14	22
42	6	10	9	**59**	5	8	15
43	1	12	14	**60**	6	9	21

Временные диаграммы №1 - №6 представлены на рисунках 26 – 31.

Для временной диаграммы №4 значение времени T3=T2, T4=T1 (рисунок 29).

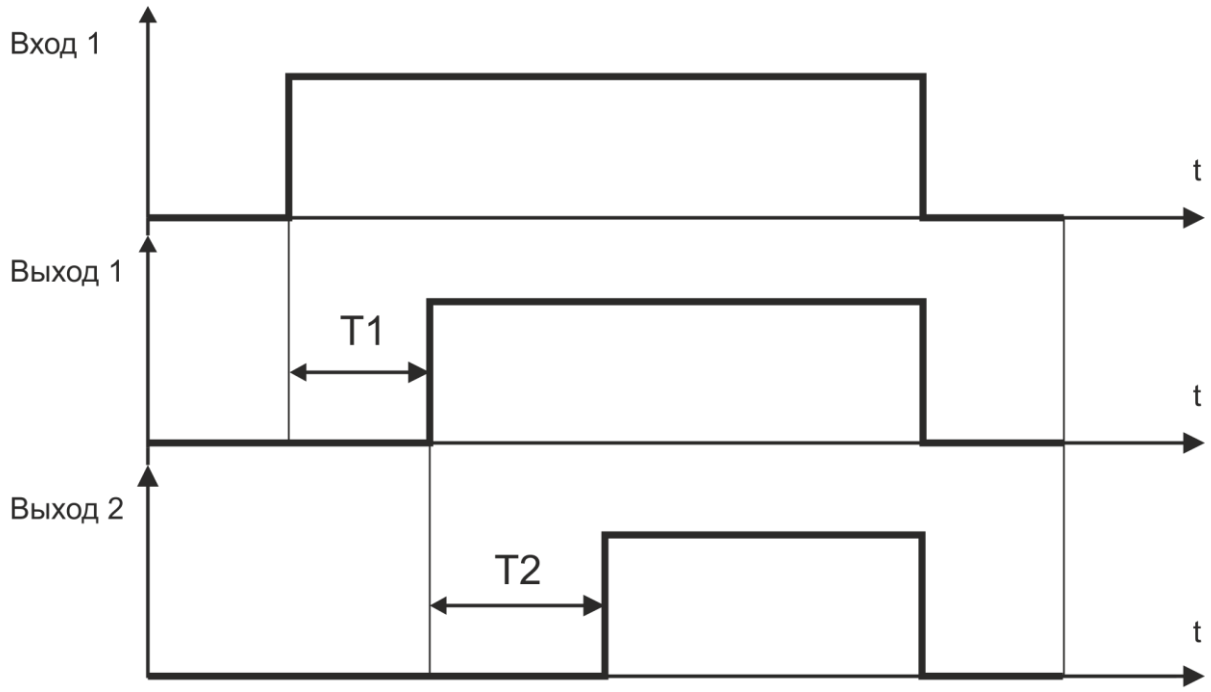


Рисунок 26 – Временная диаграмма №1

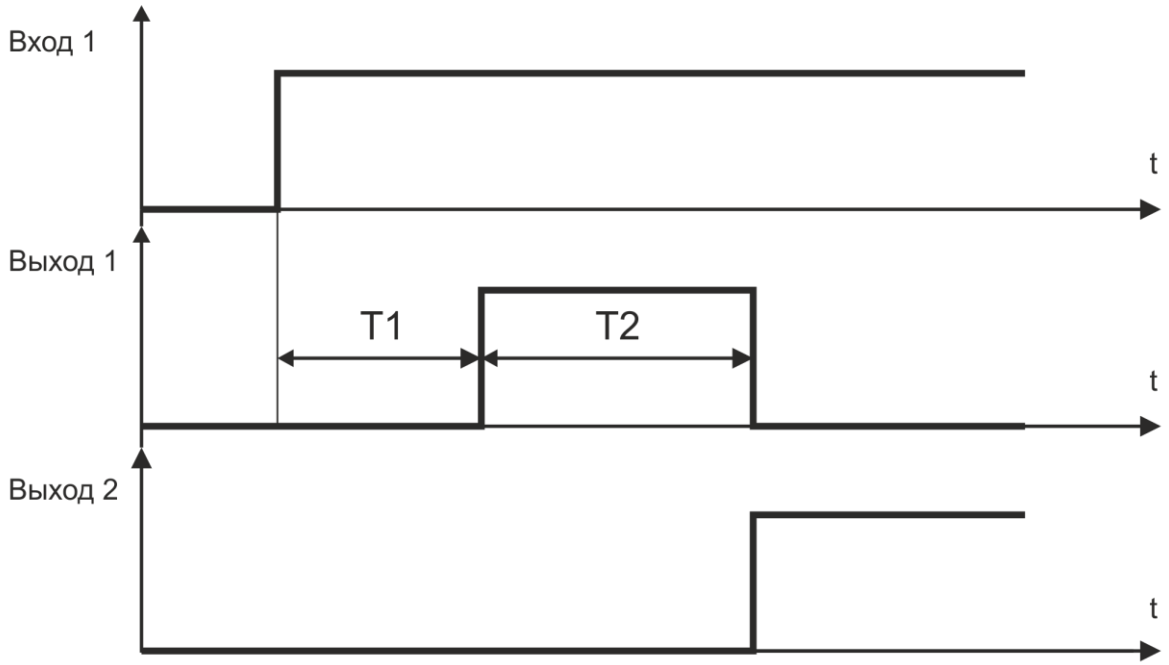


Рисунок 27 – Временная диаграмма №2

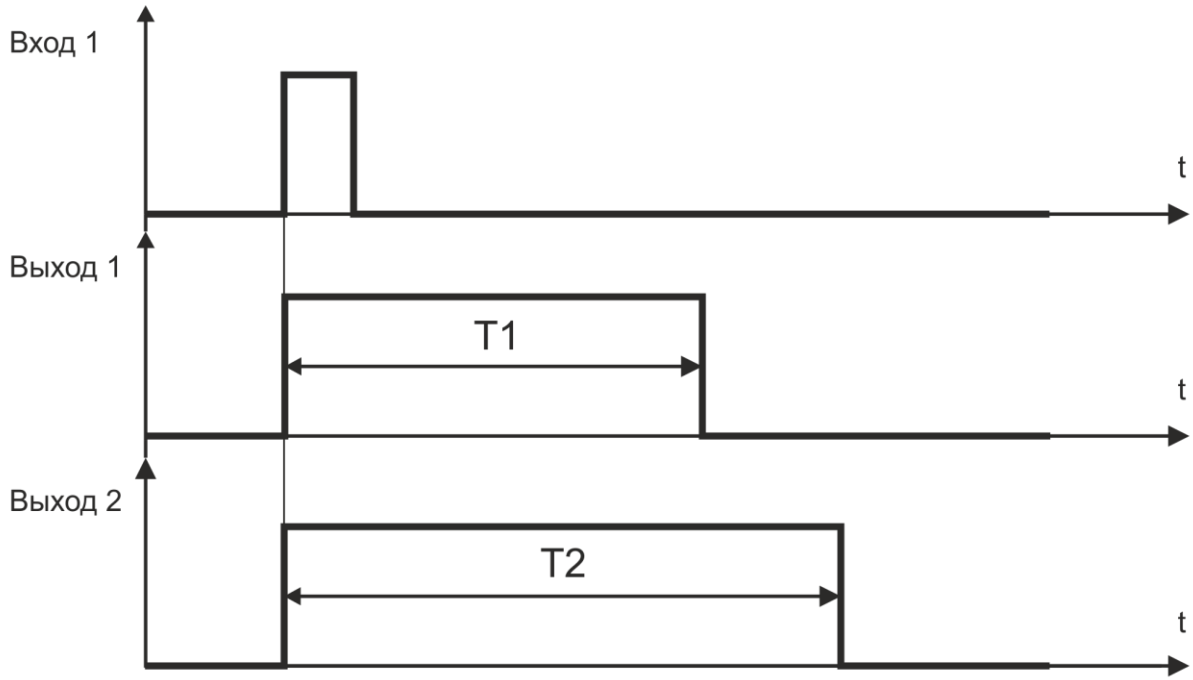
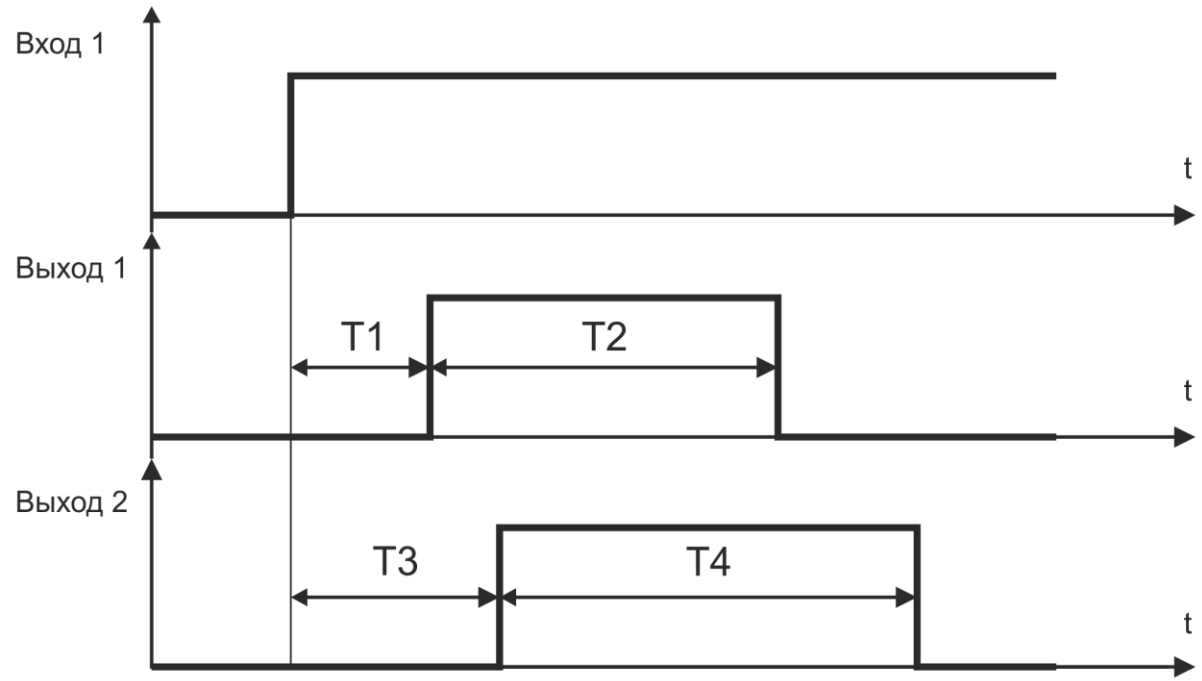


Рисунок 28 – Временная диаграмма №3



T3=T2, T4=T1

Рисунок 29 – Временная диаграмма №4

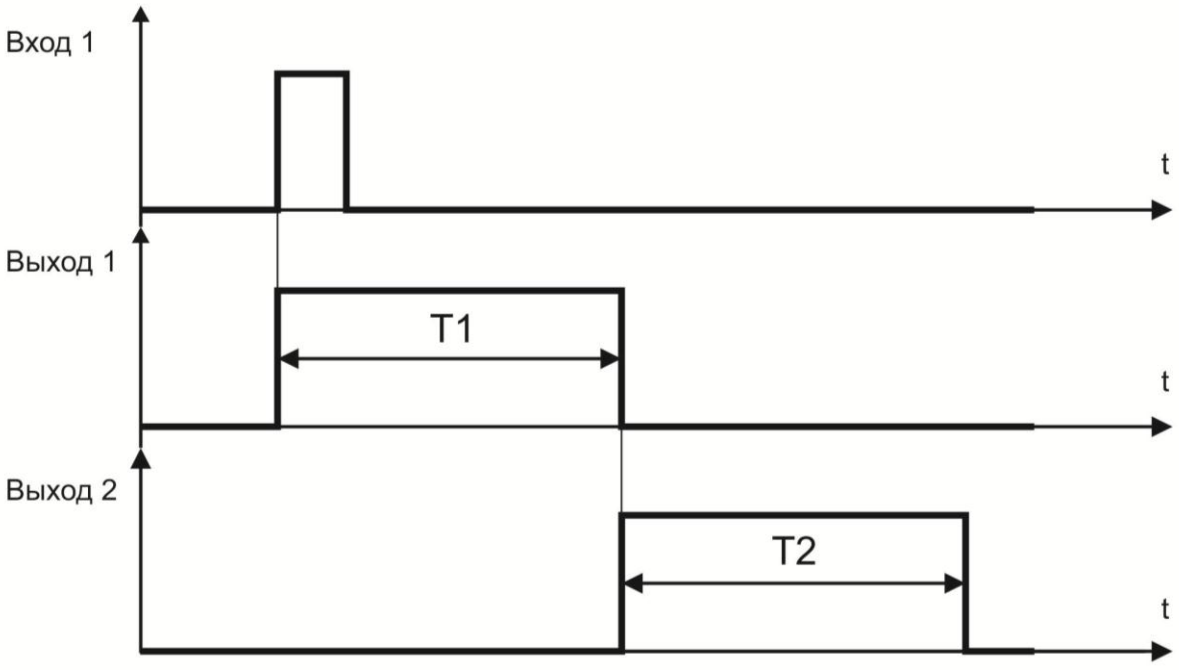


Рисунок 30 – Временная диаграмма №5

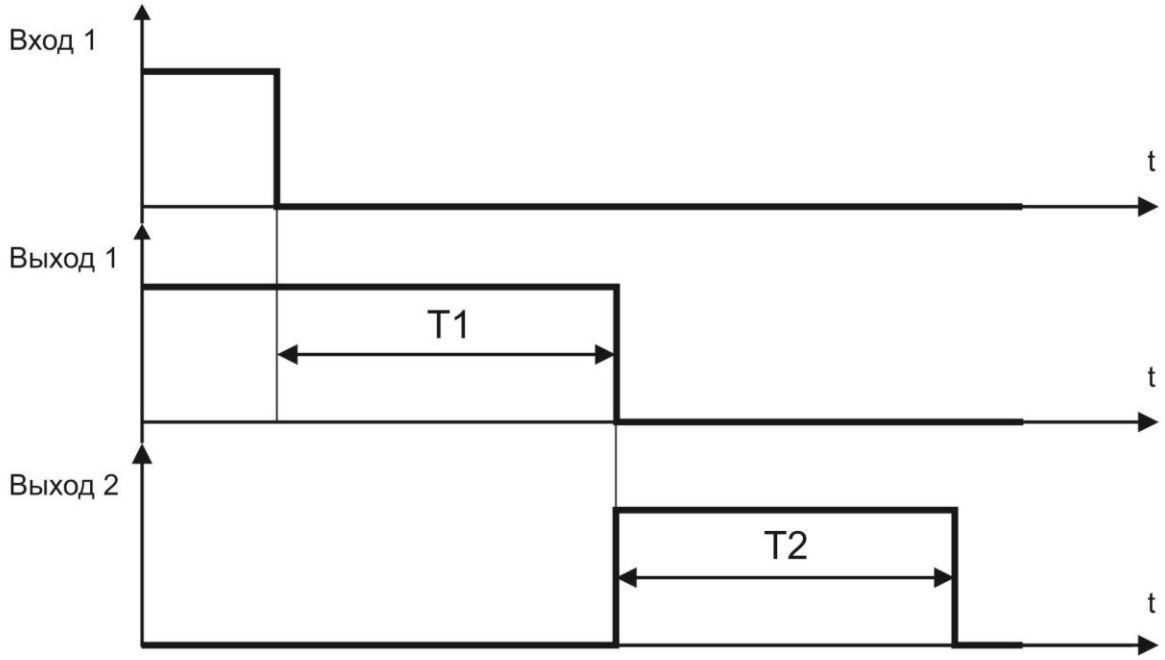


Рисунок 31 – Временная диаграмма №6

4.2 Рекомендации к выполнению контрольной работы

1 Выполнить установку на персональный компьютер программы CoDeSys.

2 Перед началом выполнения индивидуального задания, пользуясь настоящими методическим указаниями, внимательно изучить методику разработки прикладных программ ПЛК на языке LD в инструментальной системе CoDeSys.

3 **Обязательно отработать все примеры, приведенные в методических указаниях.** Особое внимание уделить исследованию создаваемых программ средствами встроенного эмулятора. Сопоставить реакцию эмулятора на заданную комбинацию входных сигналов с логикой программы управления.

4 Только после выполнения п.1 – 3 приступить к выполнения заданного варианта контрольной работы.

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

6 Пользуясь встроенным в CoDeSys эмулятором, проверить правильность разработанной программы.

7 Сделать выводы о проделанной работе.

4.3 Оформление контрольной работы

Контрольная работа должна содержать следующие разделы:

1 Титульный лист

2 Содержание

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

4 Снимок экрана (скриншот) с рабочей областью CoDeSys, в которой создана лестничная диаграмма (LD-программа), реализующая заданный алгоритм (см. рисунок 24).

5 Снимок экрана (скриншот) с рабочей областью CoDeSys в режиме эмуляции, после отработки таймерами заданных временных параметров (уставок).

6 Выводы о проделанной работе.

7 Список использованных источников

5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

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

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

4 http://www.кодесис.рф.

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

Сбродов Николай Борисович

РАЗРАБОТКА ПРОГРАММ УПРАВЛЕНИЯ ДЛЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ К КОНТРОЛЬНОЙ РАБОТЕ

для студентов направлений
15.03.04 «Автоматизация технологических процессов и производств»,
27.03.04 «Управление в технических системах»

Авторская редакция

Подписано в печать 06.04.17	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ. л. 2,0	Уч. изд. л. 2,0
Заказ №59	Тираж 25	Не для продажи

БИЦ Курганского государственного университета.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.