

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Курганский государственный университет

Кафедра автоматизации производственных процессов

ОСНОВЫ ПРОГРАММИРОВАНИЯ КОНТРОЛЛЕРА ОВЕН ПЛК100 НА ЯЗЫКЕ LD

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторной работы

по дисциплине «Технические средства автоматизации»

для студентов очной и заочной форм обучения специальности 220301

«Автоматизация технологических процессов и производств

(в машиностроении)», по дисциплине «Технические средства автоматизации и
управления» для студентов очной и заочной форм обучения направления

220400.62 «Управление в технических системах»,

по дисциплине «Средства автоматизации и управления»

для студентов очной и заочной форм обучения направления

220700.62 «Автоматизация технологических процессов и производств»

Курган 2012

Кафедра автоматизации производственных процессов

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

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

Утверждены на заседании кафедры «31» мая 2012 г.

Рекомендованы методическим советом университета «29» июня 2012 г.

ВВЕДЕНИЕ

Программируемые логические контроллеры (ПЛК), программирование которых осуществляется со встроенного или выносного пульта, встречаются сегодня достаточно редко. Как правило, это простые, специализированные ПЛК. Все программирование таких контроллеров сводится обычно к заданию набора констант.

Для программирования ПЛК универсального назначения применяют персональный компьютер. Процесс разработки и отладки программного обеспечения 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.

Однако открытость стандарта МЭК 61131 – с одной стороны, и сложность реализации высококлассных комплексов программирования – с другой, привели к появлению универсальных инструментов программирования ПЛК,

Одним из самых популярных в мире универсальных комплексов программирования является комплекс CoDeSys, который поддерживает стандарт МЭК и учитывает фирменные особенности более 150 моделей ПЛК.

Цель лабораторной работы – изучения основных приемов работы в программной среде CoDeSys и приобретение практических навыков программирования контроллера модели ПЛК100 фирмы ОВЕН на языке РКС (LD).

1 ОСНОВНЫЕ СВЕДЕНИЯ О ПРОГРАММНОМ КОМПЛЕКСЕ CoDeSys

1.1 Назначение и возможности CoDeSys

Комплекс CoDeSys (Controllers Development System) фирмы 3S (Smart Software Solutions) представляет проектировщику удобную среду для программирования контроллеров на языках МЭК [3 - 5]. 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 Компоненты организации программ (POU)

Компоненты создают под прикладное программное обеспечения ПЛК. Компоненты организации программ **POU (Program Organization Unit)** содержат функции, функциональные блоки и программы. Компонент выступает как «черный ящик», внутреннее устройство и содержание которого знать не нужно. В графическом изображении он представлен прямоугольником с входами (слева) и выходами (справа).

Выбор нужного **POU** производится в окне объявлений (рисунок 1) в строках **Программа**, **Функциональный блок** или **Функция**. Для LD будем использовать только **Программа**, т. к. нам потребуются только стандартные компоненты (контакты, катушки реле, FB).

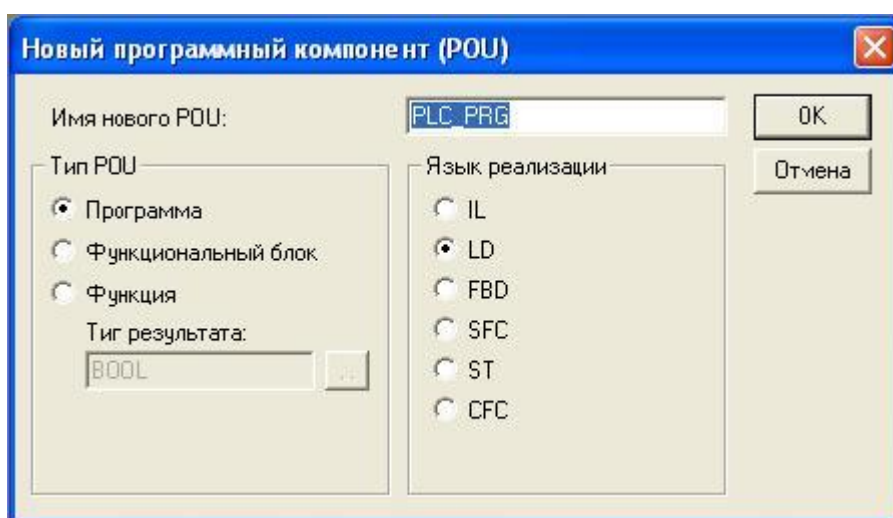


Рисунок 1 – Выбор языка программирования и задание имени программы

1.3 Запуск 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 **OK**. В появившемся окне **Новый программный компонент (POU)** (рисунок 1) выбираем тип **POU - Программа** и язык, на котором будет осуществляться написание программы - **LD**. Имя программы оставляем без изменения. Подтверждаем выбор нажатием на кнопку **OK**. После выполнения всех вышеописанных действий откроется главное окно (рисунок 2) среды CoDeSys.

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

- меню (рисунок 3);
- панель инструментов, которая содержит кнопки для быстрого вызова команд меню (рисунок 4);
- организатор объектов, имеющий вкладки «**POU**», «**Типы данных**», «**Визуализации**» и «**Ресурсы**»;

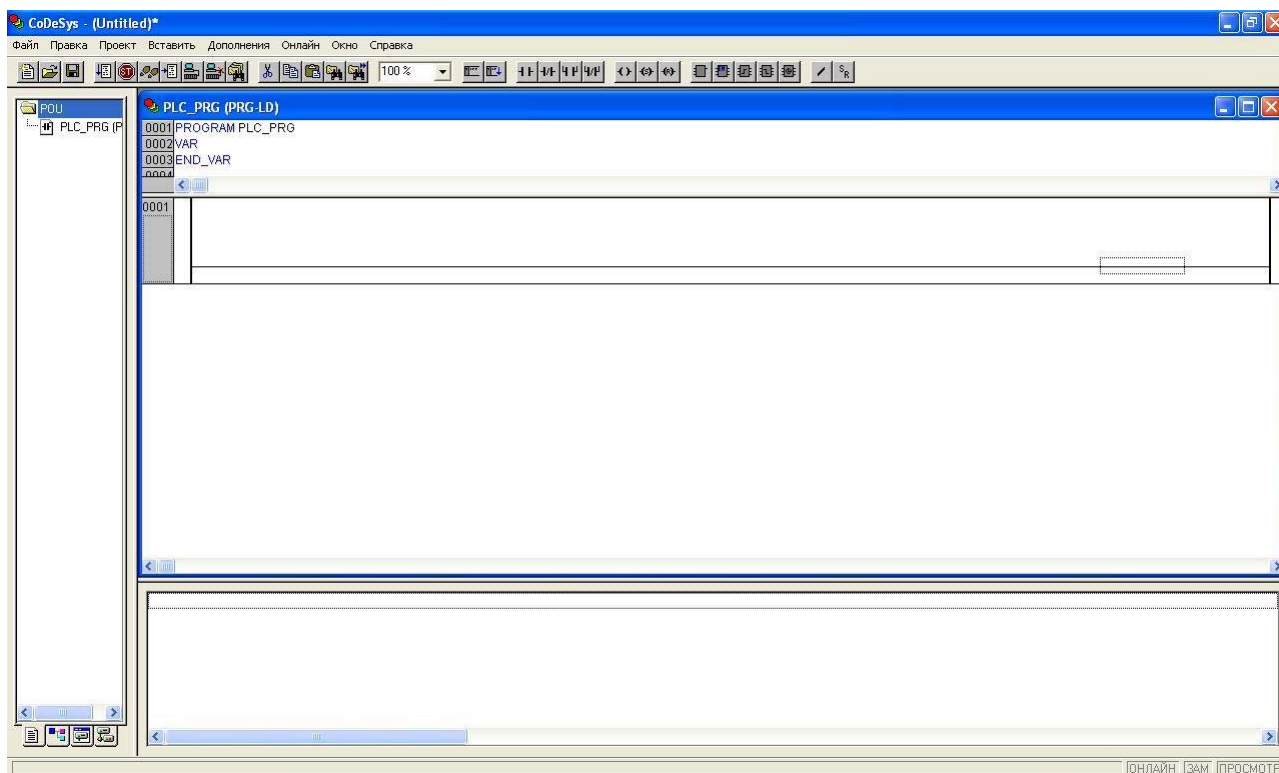


Рисунок 2 – Главное меню CoDeSys

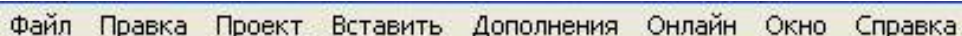


Рисунок 3 – Меню среды



Рисунок 4 – 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	Устанавливает/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 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.

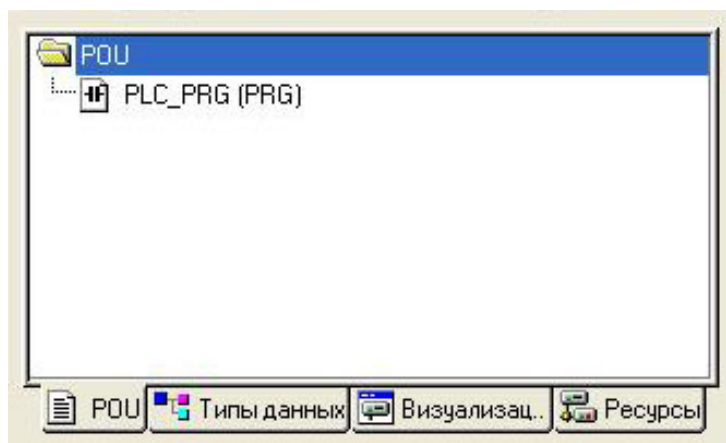


Рисунок 5 – Организатор объектов

1.4 Разделитель экрана

Разделитель 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 2) 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 6). Это меню, вызываемое 1ПКМ на рабочей области, содержит наиболее часто используемые команды.

Вырезать	Ctrl+X
Копировать	Ctrl+C
Вставить	Ctrl+V
Удалить	Del
Цепь (перед)	
Цепь (после)	Ctrl+T
Контакт	Ctrl+K
Инверсный контакт	Ctrl+G
Параллельный контакт	Ctrl+R
Параллельный контакт (инверсный)	Ctrl+D
Функциональный блок...	Ctrl+B
Детектор переднего фронта	
Детектор заднего фронта	
Таймер (TON)	
Обмотка	Ctrl+L
'Set' обмотка	Ctrl+I
'Reset' обмотка	
Элемент с EN	
Вставка в блоки	▶
Переход	
Возврат	
Комментарий	
Инверсия	Ctrl+N
Set/Reset	
Масштаб	Alt+Enter
Открыть экземпляр	

Рисунок 6 – 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 цепи.

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 2), 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 ЛКМ.

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 7.

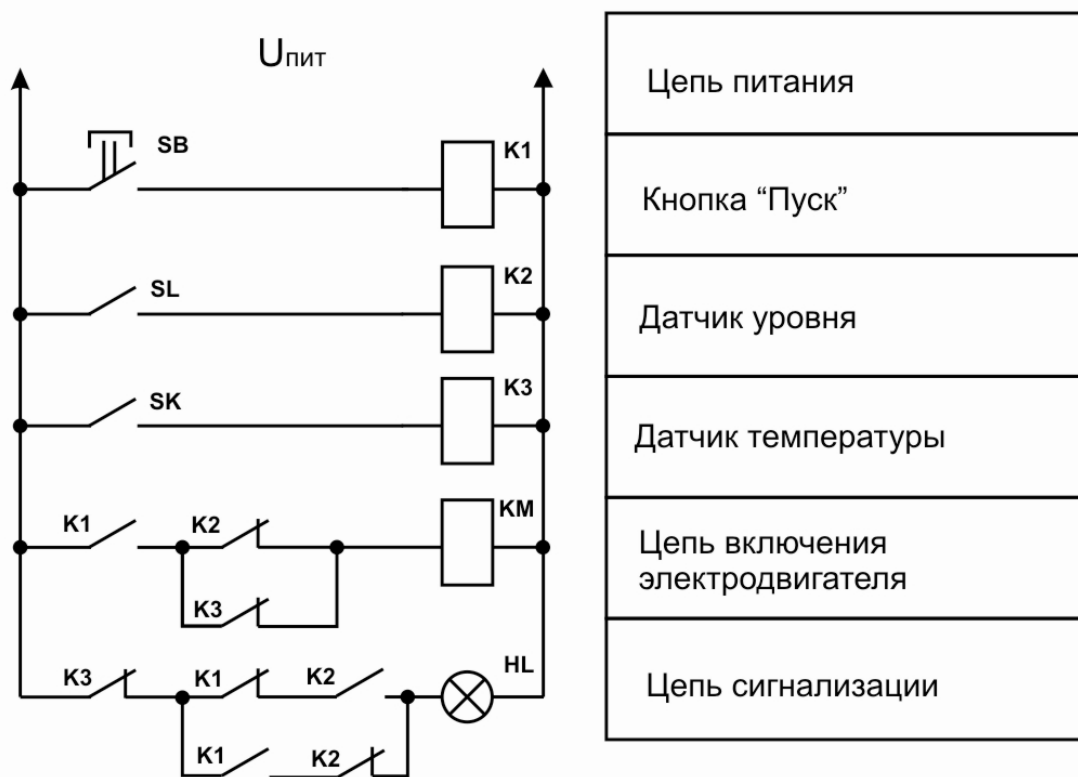


Рисунок 7 – Релейно-контактная схема системы логико-программного управления дискретным процессом

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 **K1**, **K2**, **K3**.

3 Два выходных устройства – обмотка магнитного пускателя **KM**, включающего электродвигатель, и сигнальная лампа **HL**.

2.3 Методика программирования на языке LD в среде CoDeSys

По аналогии с исходным вариантом (рисунок 7) в многоступенчатой LD - программе, **скорее всего**, также потребуется пять цепей. Слово «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 **K1**.

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 8). Эти ??? запрашивают **имя** или **идентификатор** внесенного в цепь компонента.

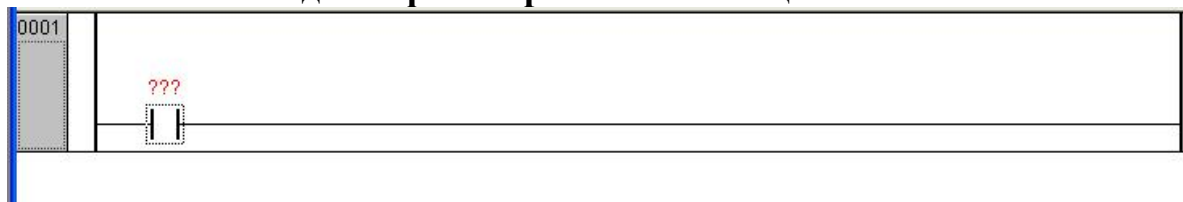


Рисунок 8 – Первая цепь с введенным контактом

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 9).

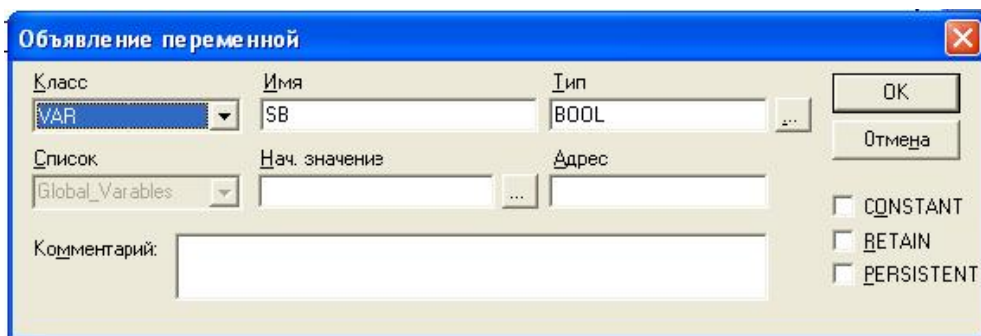


Рисунок 9 – Окно объявления переменной

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 10).

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 и пятой цепей) подключать к нему другие контакты.

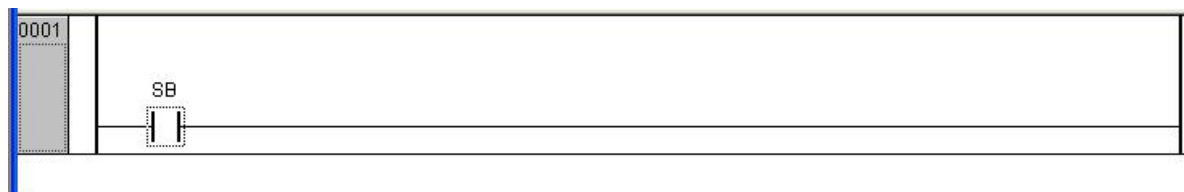



Рисунок 10 – Объявленный замыкающий контакт

Для завершения первой цепи необходимо ввести в нее катушку реле **K1**.

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 потребуются 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: **K1**. 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 11). Как видно на мониторе, зона цепи ограничена двумя горизонтальными линиями. При активации цепи обязательно 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.

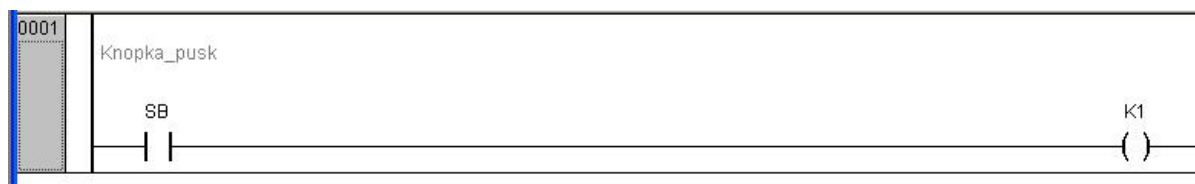


Рисунок 11 – Завершенная первая цепь

Для обозначения функционального назначения цепи в LD-диаграммах часто задают комментарии на любом языке, понятном, естественно, обслуживающему персоналу.

С этой целью наводим указатель на полосу, расположенную ниже полосы меток и щелкаем ЛКМ, появляется мигающий курсор, с клавиатуры задаем, например, «Кнопка_pusk» и нажимаем **Enter**.

Комментарии можно выполнить и на русском языке. С этой целью наводим курсор на поле цепи, щелкаем ПКМ. Открывается контекстное меню (рисунок 6). Щелкаем ЛКМ 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 сохраняться. 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 6) и кнопками  и . Воспользуемся третьим приемом как наиболее простым.

1 Наводим курсор на , щелкаем 1ЛКМ. Сразу появляется вторая цепь (рисунок 12).

2 По вышеописанной методике вносим в цепь замыкающий контакт датчика **SL** и катушку **K2**. Если есть необходимость, то можно вписать комментарий: например, «Datchik_urovnia».

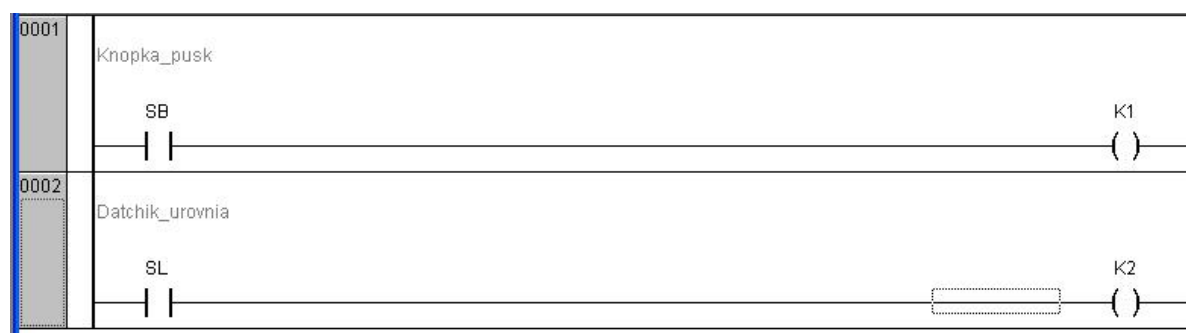






Рисунок 12 – Вторая цепь

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 **K3**.

Несколько сложнее будет программирование четвертой и пятой цепей.

По известной методике создаем четвертую цепь, последовательно внося в цепь замыкающий контакт **K1**, размыкающий **K2**. Теперь необходимо создать параллельную цепочку из размыкающего контакта **K2** и размыкающего **K3**. Для этого:

- Наводим курсор на контакт **K1**, щелкаем 1ЛКМ. Контакт активирован, о

чем свидетельствует пунктирный квадратик, охватывающий контакт.

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

Если активировать саму цепь, а не контакт **К2**, то замыкающий контакт **К3** охватил бы всю цепочку из контактов **К1** и **К2**, как показано на рисунке 13.

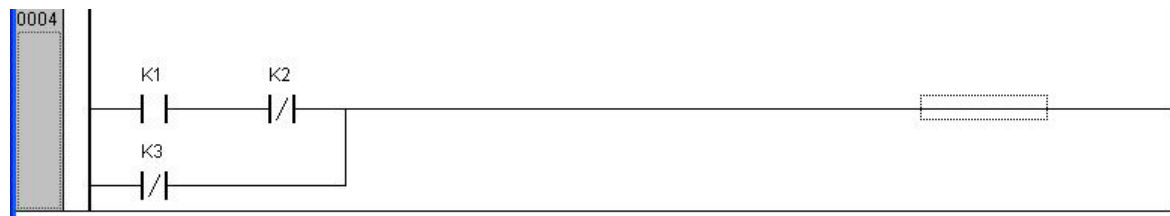


Рисунок 13 – Фрагмент программы после неудачного введения контакта **К3**

Вносим в цепь катушку **КМ**. Четвертая цепь завершена.

Программируем пятую цепь LD-программы.

По известной методике вносим в цепь размыкающие контакты **К3** и **К1**, замыкающий **К2** и обмотку (катушку) **HL**. Необходимости объявлять переменные **К1**, **К2** и **К3** нет, так как они уже внесены в список булевых переменных. Осталось создать параллельную цепочку из замыкающего контакта **К1** и размыкающего **К2**. Для этого:

- Активируем размыкающий контакт **К1**, нажимаем и не отпускаем «**Shift**», активируем замыкающий контакт **К2**, отпускаем «**Shift**», оба контакта оказались охваченными пунктирным прямоугольником.





- Наводим курсор на , щелкаем 1ЛКМ, и параллельно с этими контактами появится , которому присвоим имя **К1**. Фрагмент полученной схемы в пятой цепи показан на рисунке 14.



Рисунок 14 – Пятая цепь после введения замыкающего контакта **К1** и его активации

- Активируем новый контакт, наводим курсор на , щелкаем 1ЛКМ и последовательно с появляется . Присваиваем имя **К2** новому элементу.

Параллельная цепочка выполнена, и завершена вся многоступенчатая схема в виде LD-программы (рисунок 15).

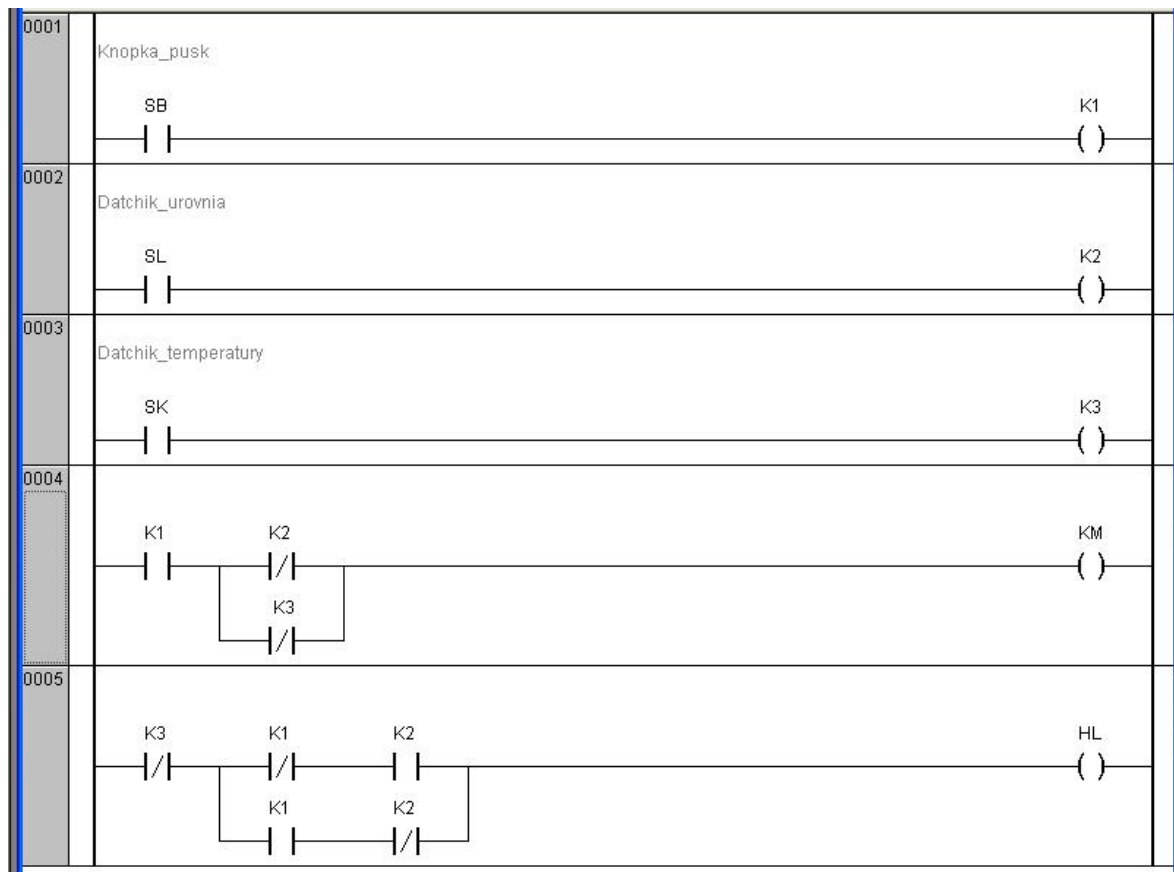



Рисунок 15 – Система логико-программного управления дискретным процессом в виде программы на языке LD

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 ПКМ и в открывшемся контекстном меню (рисунок 6) щелкнуть ЛКМ 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 ЛКМ 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, ФВ) или на саму цепь, щелкаем ПКМ и в открывшемся контекстном меню нажимаем ЛКМ 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 ЛКМ, 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 16).

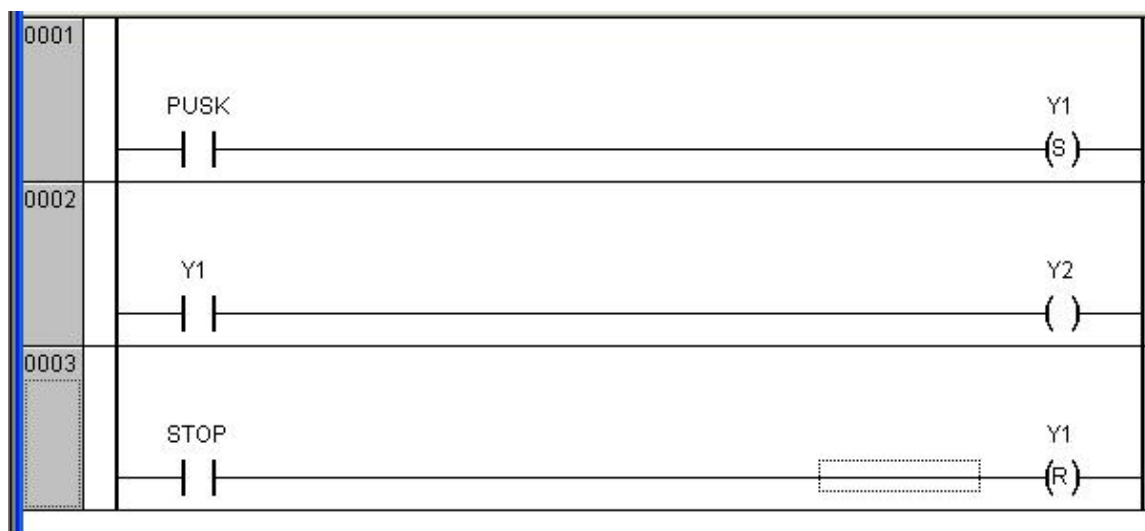


Рисунок 16 – LD-диаграмма фиксации катушки реле Y2 с помощью обмоток S и R

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 17).

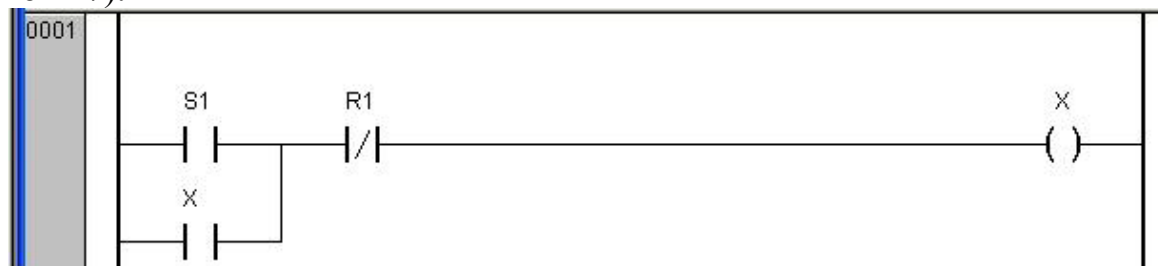


Рисунок 17 – Схема управления катушкой **X** самофиксацией состояния

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) широко используется на практике при организации логико-программного управления дискретными исполнительными устройствами.

2.6 Исследование **LD**-программы в режиме эмуляции

Итак, исходную **РКС** (рисунок 7) запрограммировали в **CoDeSys** на языке **LD** и представили её в виде многоступенчатой схемы (рисунок 15). 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 3), щелкаем **1ЛКМ**. Открывается меню (рисунок 18).

- 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.

Подключение	Alt+F8
Отключение	Ctrl+F8
Загрузка	
Старт	F5
Стоп	Shift+F8
Сброс	
Сброс (холодный)	
Сброс (заводской)	
Переключить точку останова	
Диалог точек останова	F9
Шаг поверху	F10
Шаг детальный	F8
Один цикл	Ctrl+F5
Записать значения	
Фиксировать значения	Ctrl+F7
Освободить фиксацию	F7
Диалог Запись/Фиксация	Shift+F7
Ctrl+Shift+F7	
Показать стек вызовов...	
Отображать поток выполнения	
✓ Режим эмуляции	
Параметры связи...	
Загрузка исходных текстов	
Создание загрузочного проекта	
Записать файл в ПЛК	
Читать файл из ПЛК	

Рисунок 18 – Меню Онлайн

- 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 **K1, K2, K3, KM** и **HL** 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 **K1** и вся первая цепь залятся синим цветом. Контакты **K1** в четвертой и пятой цепях

замыкаются и также окрашиваются в синий цвет. Размыкающие контакты **K3** и **K4** в этих цепях остаются замкнутыми, следовательно, срабатывают **KM** и **HL**.

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 **KM**. 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.

2.6 Загрузка программы в ПЛК

До сих пор рассматривались различные аспекты разработки LD-программ в предположении, что их исследование будет завершаться только в режиме эмуляции, т. е. без последующей привязки к конкретному ПЛК.

Восполним этот пробел на примере разработанной программы (рисунок 15).

Перед загрузкой программы в ПЛК необходимо выполнить конфигурацию входных и выходных переменных, т. е. ввести имена этих переменных определенным образом в конфигуратор ПЛК. Рассмотрим этот этап подробнее.

Для этого внизу окна организатора объектов (рисунок 5) необходимо нажать 1ЛКМ на вкладку **Ресурсы (Resources)** и из дерева ресурсов выбрать **Конфигурация ПЛК (PLC Configuration)**. Откроется окно (рисунок 19), в котором следует объявлять входные и выходные переменные.

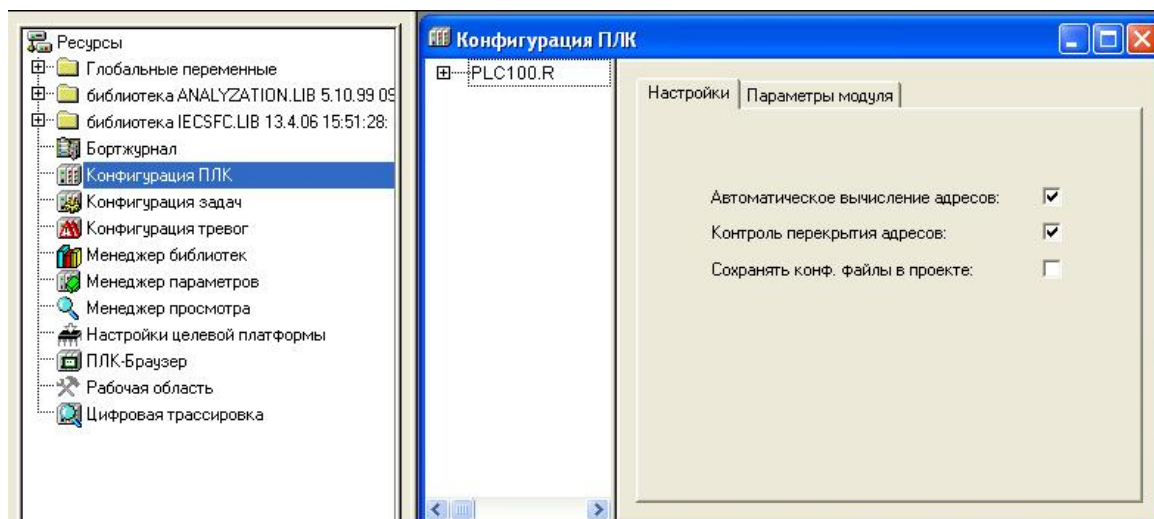


Рисунок 19 – Открытие окна «Конфигурация ПЛК»

Напомним, что в разработанной программе три входные переменные (**SB**, **SL**, **SK**) и две выходные (**KM** и **HL**). Начнем с входных элементов.

В левой части окна конфигуратора следует раскрыть вкладку PLC100.R, щелкнув 1ЛКМ по символу «+». Потом 2ЛКМ щелкнуть по **Discrete Input 8 bit**. Так как в контроллере ОВЕН ПЛК100 всего 8 дискретных входов, то в нашем распоряжении будет 8 строк: от AT%IB0.0.0 до AT%IB0.0.7. Мы вправе выбрать любые два входа из восьми возможных. Например, нажав на «+», раскрываем вкладку AT % IB0.0: BYTE. Далее щелкаем 1ЛКМ в самом начале строчки AT % IB0.0.0: BOOL и вводим имя переменной **SB**. Завершаем ввод клавишей Enter. Аналогичным образом, но только уже в строке AT % IB0.0.1: BOOL присвоим значение переменной **SL**, а строке AT % IB0.0.2: BOOL – переменной **SK**.

Для ввода выходных переменных необходимо нажать на «+» возле строки **Discrete Output-Relay**. В контроллере ОВЕН ПЛК100 шесть дискретных выходов. Поэтому откроются 6 строк: от AT%QX1.0 до AT%QX6.0. Выбираем любые две из шести. Например, щелкаем 1ЛКМ в начале надписи AT в строке AT%QX1.0: BOOL и вводим имя первой выходной переменной **KM**. Аналогичным образом следует ввести имя второй переменной вывода **HL** в строку AT %QX2.0: BOOL.

Итак, переменные введены, и полученный результат представлен в окне конфигуратора (рисунок 20).

Выполненные действия по конфигурации ПЛК предполагают, что при аппаратной реализации данной системы управления кнопка управления «Пуск» **SB**, дискретные датчики температуры **SK** и уровня **SL**, обмотка магнитного пускателя **KM** и сигнальная лампа **HL** будут подключены к соответствующим входам и выходам контроллера.

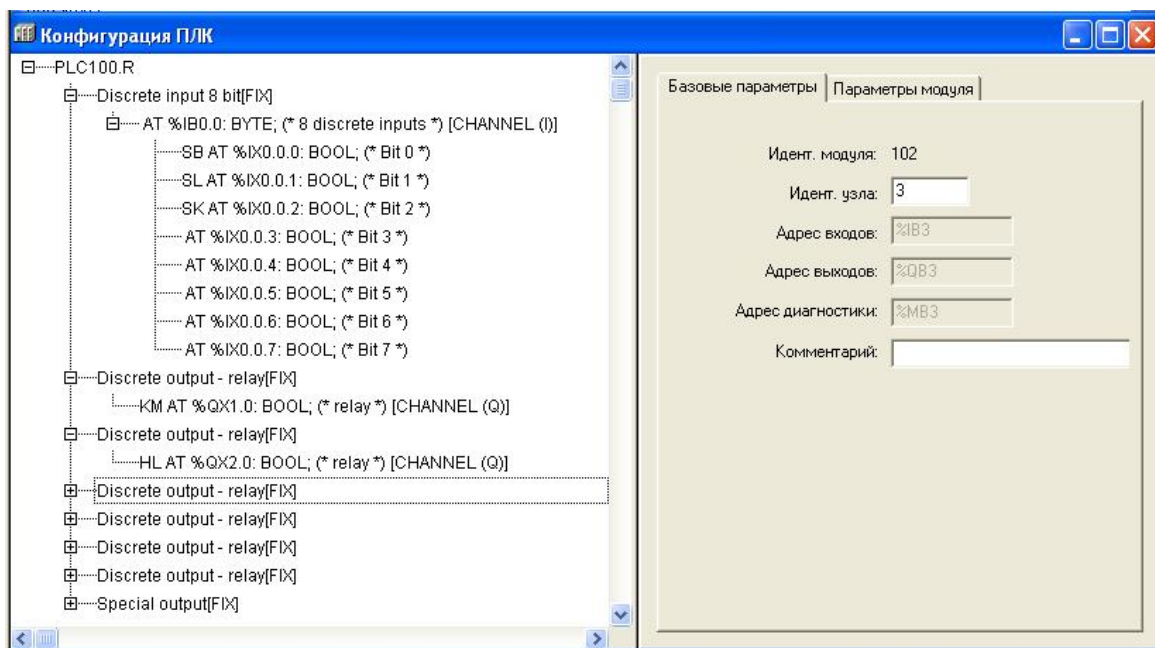


Рисунок 20 – Окно конфигуратора ПЛК

Теперь созданный проект можно откомпилировать командой **Компилировать** в меню **Проект** или клавишей **F11**. Если все сделано верно, то в нижней части окна должно появиться сообщение **0 ошибок** или **Проект актуален**. В противном случае необходимо исправить допущенные ошибки. В этом помогут развернутые сообщения об ошибках.

Следующим этапом является подключение контроллера к компьютеру. Контроллер ОВЕН ПЛК100 имеет следующие коммуникационные интерфейсы: Ethernet 10/100 Mbps, RS232 (два канала), RS485 и USB 2.0-Device. Связь с контроллером может быть организована различным образом: по интерфейсам Ethernet, Debug RS-232, USB-Device.

Один из вариантов организации связи ПЛК с компьютером обеспечивается через порт Debug RS-232. Данный порт расположен на лицевой панели контроллера ОВЕН ПЛК100 и предназначен для связи ПЛК со средой программирования, загрузки пользовательской программы и её отладки. Подключение к этому порту осуществляется кабелем, входящим в комплект поставки ОВЕН ПЛК100. Используемый протокол – GateWay (протокол системы CoDeSys).

После организации связи выполняется настройка канала соединения с контроллером в окне **Параметры связи (Communication parameters)**, вызываемом командой меню **Online**.

Далее в меню **Онлайн** щелкаем 1ЛКМ по строке **Подключение**. При этом **Режим эмуляции** должен быть предварительно отключен. Устанавливается связь между ПЛК и компьютером (системой CoDeSys). Если компиляция проекта командой **Компилировать** еще не выполнялась, то она будет выполнена автоматически при соединении контроллера с компьютером по команде **Подключение**. При этом выводятся обнаруженные ошибки и предупреждения.

Команда **Загрузка** в меню **Онлайн** загружает код проекта (программу) в контроллер.

Запускает программу в ПЛК команда **Старт** из меню **Онлайн**, а останавливает – команда **Стоп**.

3 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Лабораторную работу необходимо выполнять в следующей последовательности:

1 Изучить основные сведения о составе и возможностях программного комплекса CoDeSys (раздел 1).

2 Запустить на компьютере программу CoDeSys.

3 Пользуясь данными методическими указаниями, изучить методику программирования ПЛК на языке LD и отработать в среде CoDeSys все примеры, приведенные в разделе 2.

4 Получить у преподавателя задание на разработку программы управления дискретным технологическим объектом.

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

6 Написать LD-программу управления.

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.

4 СОДЕРЖАНИЕ ОТЧЕТА

В отчете указывается цель лабораторной работы и приводятся следующие результаты:

1 Задание на разработку программы управления дискретным технологическим объектом.

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

3 Блок-схема алгоритма управления.

4 Текст программы на языке LD.

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

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Каково назначение базовых команд языка?

2.Какова структура LD-программы?

3 Как в LD-программе задают проверку состояния входных дискретных сигналов?

4 Какими командами в LD-программе формируют выходные дискретные сигналы?

5 Каким образом можно фиксировать выходные дискретные сигналы?

6 Как выполнить конфигурацию входных и выходных переменных?

7 Как проверить правильность LD-программы?

8 Как загрузить код программы в ПЛК?

9 Как запустить программу в контроллере?

6 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.owen.ru/catalog/73637995>

5 <http://www.owen.ru/catalog/51162335>

Сбродов Николай Борисович

**ОСНОВЫ ПРОГРАММИРОВАНИЯ КОНТРОЛЛЕРА
ОВЕН ПЛК100 НА ЯЗЫКЕ LD**

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторной работы
по дисциплине «Технические средства автоматизации»
для студентов очной и заочной форм обучения специальности 220301
«Автоматизация технологических процессов и производств
(в машиностроении)», по дисциплине «Технические средства автоматизации и
управления» для студентов очной и заочной форм обучения направления
220400.62 «Управление в технических системах»,
по дисциплине «Средства автоматизации и управления»
для студентов очной и заочной форм обучения направления
220700.62 «Автоматизация технологических процессов и производств»

Редактор О.Г. Арефьева

Подписано к печати	Формат 60x84 1/16	Бумага тип. №1
Печать трафаретная	Усл. печ. л. 1,75	Уч.-изд. л. 1,75
Заказ	Тираж 50	Цена свободная

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