МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
СТАВРОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ
КАФЕДРА АВТОМАТИКИ, ЭЛЕКТРОНИКИ И МЕТРОЛОГИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

К КУРСОВОЙ РАБОТЕ ПО ДИСЦИПЛИНЕ

“АВТОМАТИКА”

Для студентов очной и заочной форм обучения, по специальностям
110302.65-“Электрификация и автоматизация сельского хозяйства”
110300.02. – «Агроинженерия»
140211.65 –«Электроснабжение»
210300.62 –«Радиотехника»

Ставрополь, 2011

Методические указания и задания разработаны: кандидатом технических наук профессором И.Г.Минаевым, генеральным директором ООО НПО «Электроимпульс» В.В. Самойленко.

Методические указания предназначены для студентов очной и заочной форм обучения, по специальностям:

110302.65 - “Электрификация и автоматизация сельского хозяйства”.
110300.02. – «Агроинженерия»
140211.65 –«Электроснабжение»
210300.62 –«Радиотехника»

Одобрены и рекомендованы к изданию заседанием кафедры АЭиМ (протокол № 22 от 02.03. 2011 года).

“ Утверждаю”
Заведующий кафедрой Автоматики
электроники и метрологии
профессор Минаев И. Г.

“____”_____ 200 _г.

Задание

на курсовую работу по дисциплине “Автоматика” выдано студенту _____ курса _____ группы дневной (заочной) формы обучения электроэнергетического факультета

Ставропольского Государственного Аграрного Университета.

_____

(фамилия, имя, отчество и № зачетной книжки)

1. Для системы автоматического управления технологическим процессом, имеющей в своем составе; четыре приемных элемента: A, B, C, D; два исполнительных элемента: X, Y; и работающей по следующему алгоритму:

а) элемент X срабатывает, если срабатывают элементы: (Приложение 1) ,

_____ , _____ , _____ ;

б) элемент Y срабатывает, если срабатывают элементы: (Приложение 1) ,

_____ , _____ , _____ ;

выполнить следующее:

1.1 .Спроектировать на контактных элементах систему логического управления технологическим процессом.

1.2 .Проверить выборочно выполнение нескольких частных условий срабатывания и несрабатывания исполнительных элементов спроектированной системы.

1.3 .Спроектировать бесконтактный вариант той же самой системы логического управления в базисе логических элементов “ и-не ”, (“ или- не ”). (Приложение 2)

1.4 Составить программу работу системы по п.1.1 для программируемого логического контроллера (типа ПЛК 100 RL ОВЕН) в среде CoDeSys на языке LD в соответствии с временными диаграммами (Приложение 3).

Задание получил _____

Задание выдал _____

ПРЕДИСЛОВИЕ

Цель курсовой работы - получение студентами теоретических сведений и практических навыков проектирования систем логического управления (СЛУ).

В результате выполнения курсовой работы студент овладевает методикой составления логических функций, реализации минимизированных функций в релейно-контактном и бесконтактном варианте, выбора аппаратных средств реализации логических устройств и осваивает приемы программирования ПЛК.

Курсовая работа состоит из двух частей. В первой части необходимо спроектировать систему логического управления технологическим процессом.

Во второй части работы необходимо спроектированную контактную СЛУ запрограммировать на языке LD в среде CoDeSys для программируемого логического контроллера ПЛК100 RL ОВЕН, проверить ее работоспособность в режиме эмуляции на ПК, и распечатать эту программу на принтере.

Исходные данные указаны в Приложениях 1-3 и выбираются студентом по номеру его зачетной книжки.

Пояснительная записка оформляется в соответствии с требованиями ГОСТов ЕСКД на листах бумаги формата А4 (197х210). Все расчеты в пояснительной записке должны сопровождаться комментариями и ссылками на литературу и ранее полученные выражения.

Пояснительная записка к курсовой работе должна содержать следующие разделы: содержание, введение, расчетная часть, заключение, список использованных литературных и INTERNET источников.

Защита курсовой работы производится до сдачи экзамена по дисциплине. По результатам защиты выставляется оценка.

С целью уменьшения объема курсовой работы приняты следующие упрощения:

- Приемные элементы – катушки электромагнитных реле A, B, C, и D и их замыкающие и размыкающие контакты a и $\bar{a}$, $в$ и $\bar{в}$, c и $\bar{c}$, d и $\bar{d}$ соответственно следовало бы по действующему стандарту обозначить К1, К2 и т.д., а их контакты К1.1, К1.2, К2.1, К2.2, К2.3 и т.д., что значительно бы осложнило запись логических функций.
- Электрические параметры исполнительных элементов X и Y не указаны, что также освобождает проектировщика от необходимости преобразования и усиления выходного сигнала СЛУ (по напряжению, мощности) и выбора усилителей.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Применение алгебры логики для описания логических элементов и систем

Системы логического уравнения (СЛУ) бывают комбинационными и последовательностными. Последние называются также событийно – управляемыми автоматами.

В комбинационных СЛУ выходные сигналы формируются только при определённых комбинациях входных логических сигналов, принимающих значения 0 или 1. В последовательностных СЛУ выходные сигналы зависят не только от комбинации входных, но и последовательности их поступления во времени, что обеспечивается наличием элементов памяти. В настоящее время последовательстные СЛУ в зависимости от сложности решаемых задач выполняются на базе программируемых логических контроллеров – ПЛК.

В курсовой работе студентам предстоит выполнить анализ и синтез комбинационной СЛУ на контактных и бесконтактных элементах (для четырёх входных и двух выходных сигналов.) При проектировании этой же СЛУ в среде CoDeSys вводим дополнительные условия (Приложение 3), которые приводят её в класс событийно-управляемой логики. Эту задачу выполнить на контактных элементах было бы значительно труднее.

Словесное изложение работы даже простых СЛУ выглядит громоздко и затрудняет проведение анализа. Математическим аппаратом для описания СЛУ служит двузначная (бинарная) алгебра логики, все переменные в которой могут принимать только два значения (**0** или **1**).

Основным понятием, используемым при анализе и синтезе систем логического управления, является логическая функция.

Принципы построения логической функции по алгоритму работы системы автоматики иллюстрируется ниже на примере.

Логическая функция выражает зависимость выходных переменных от входных и также принимает, в зависимости от значения последних и связывающих их логические действия, два состояния: **0** или **1**. Можно встретить и такую запись этих состояний : Ложь или Истина; FALSE или TRUE).

Так как каждая переменная может иметь только два значения, то возможное количество различных комбинаций (наборов) $N_{комб}$ для *n* переменных будет равно:

$$N_{комб} = 2^n$$

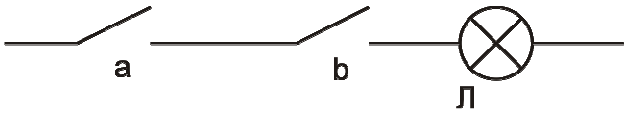
Все действия над переменными в бинарной алгебре выполняются с помощью следующих основных операций, которые наглядно иллюстрируются соответствующими релейно-контактными схемами.

В настоящее время «релейная автоматика» звучит весьма архаично (вроде как ламповый компьютер). Но, зная навыки построения контактных СЛУ и используя современный язык LD в среде CoDeSys, можно легко эти схемы перевести в программу для ПЛК. Это язык и был в свое время разработан для инженеров, знающих релейные автоматы, но не владеющими навыками программирования на языках высокого уровня.

Логическое умножение

Логическое умножение (конъюнкция, функция "**И**") $Л = a \cdot b$**;** равнозначно последовательному соединению контактов. Все возможные комбинации входных сигналов и соответствующие им значения функции сведены в таблицу состояний. Очевидно, что только в одном случае результатом логического умножения станет единица, т.е. лампа **Л** "сработает", если замкнуть контакты и *a*, и *b*. Из этой словесной формулы союз "**И**" перешел в обозначение функции (как синоним логическому умножению) и в название бесконтактного элемента.

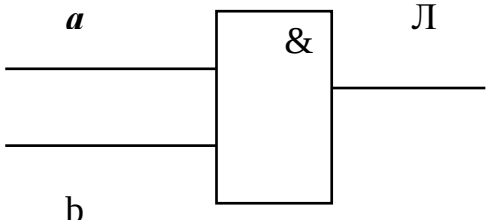
а) Релейно-контактный вариант реализации:



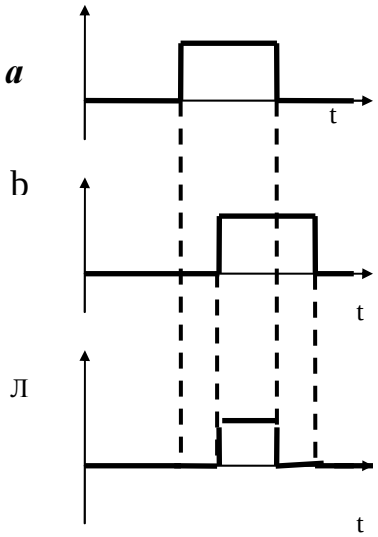
б). Таблица состояний:

a	b	Л
0	0	0
0	1	0
1	0	0
1	1	1

в). Обозначение бесконтактного элемента:



г). Временные диаграммы:



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

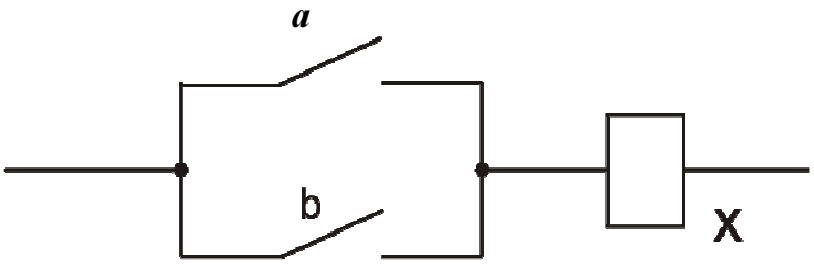
Применяются и другие обозначения операции логического умножения:

$$a \cdot b = a \wedge b = a \& b$$

Логическое сложение

Результат логического сложения (дизъюнкции, операции "**ИЛИ**") ***X = a + b;*** легко установить из анализа схемы с параллельным соединением контактов.

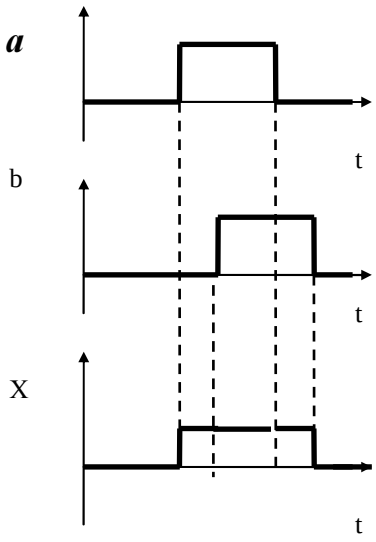
а) Релейно-контактный вариант реализации:



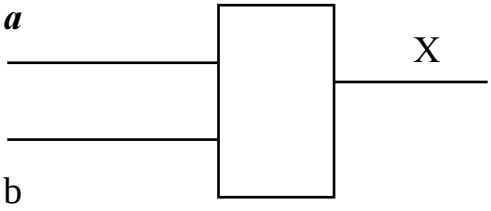
б) Таблица состояний:

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

г). Временные диаграммы:



в) Обозначение бесконтактного элемента:



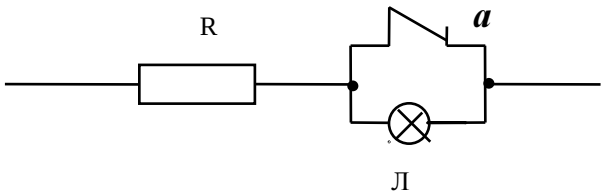
Очевидно, что катушка реле **X** получит питание, если замкнуть контакты или *a*, или *b*, или оба вместе.

Вместо знака "+" иногда употребляют «∨»: a+b=a∨b

Логическое отрицание.

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

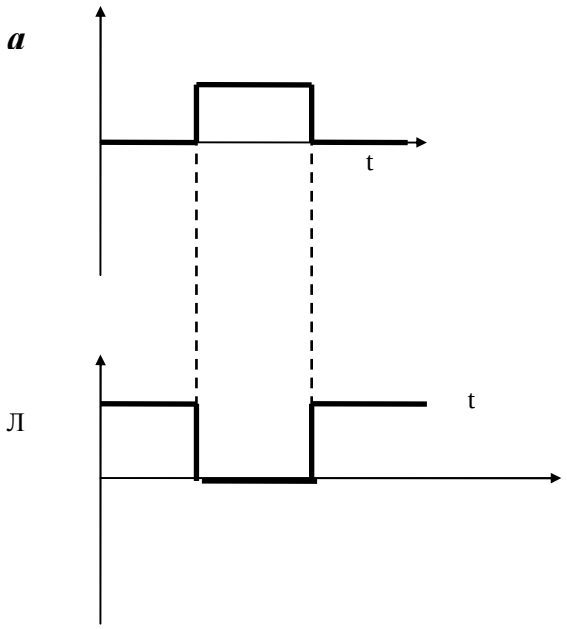
а) Релейно-контактный вариант реализации:



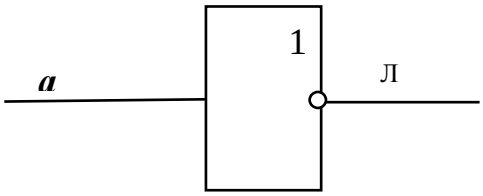
б) Таблица состояний:

a	Л
0	1
1	0

г) Временные диаграммы:



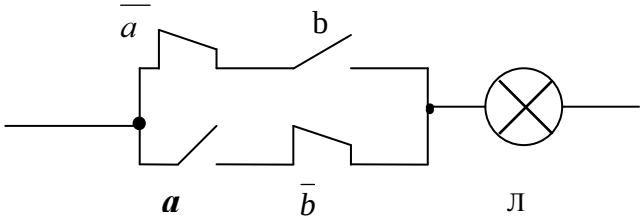
в) Обозначение бесконтактного элемента:



<u>Логическое сложение по модулю 2,</u> функция **”ИСКЛЮЧАЮЩЕЕ ИЛИ”**

$$Л = a \cdot \overline{b} + \overline{a} \cdot b$$

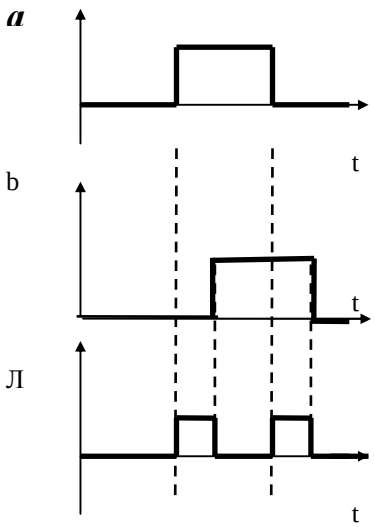
а) Релейно-контактный вариант реализации:



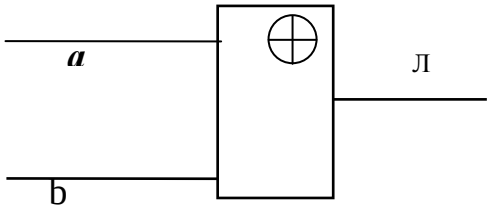
б) Таблица состояний:

a	b	Л
0	0	0
0	1	1
1	0	1
1	1	0

г) Временные диаграммы:



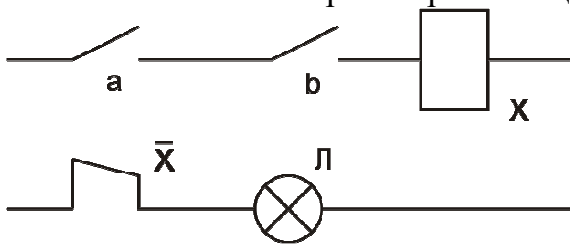
в) Обозначение бесконтактного элемента:



<u>Инверсия конъюнкции,</u> функция **“И - НЕ”:**

$$Л = \overline{a \cdot b}$$

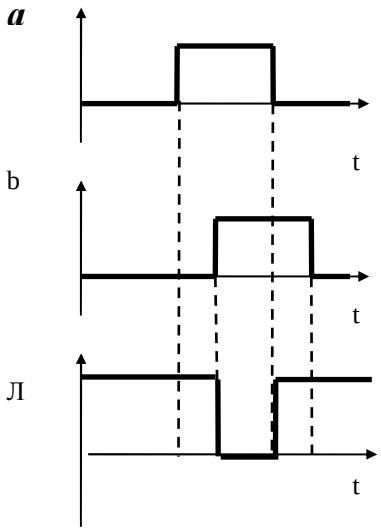
а) Релейно-контактный вариант реализации:



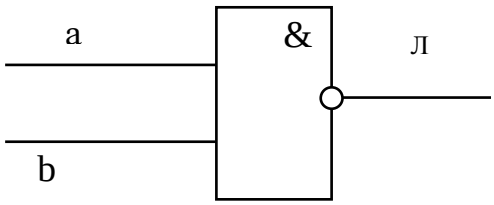
б) Таблица состояний:

a	b	Л
0	0	1
0	1	1
1	0	1
1	1	0

г) Временные диаграммы:



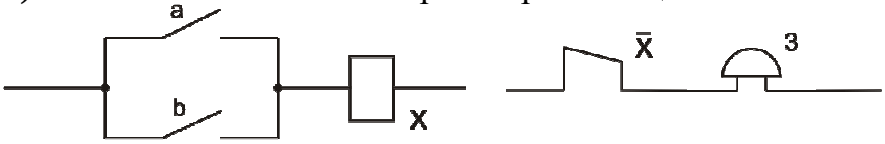
в) Обозначение бесконтактного элемента:



<u>Инверсия дизъюнкции,</u> функция **“ИЛИ -НЕ”:**

$$З = \overline{a + b}$$

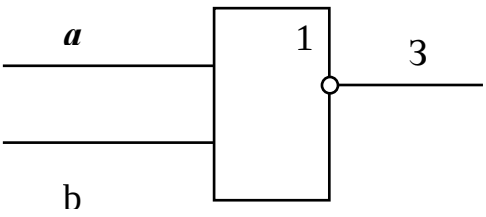
а) Релейно-контактный вариант реализации:



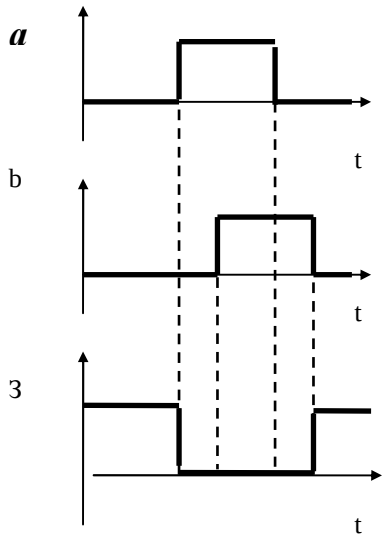
б) Таблица состояний:

a	b	З
0	0	1
0	1	0
1	0	0
1	1	0

в) Обозначение бесконтактного элемента:



г) Временные диаграммы:



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

Знак "=", который в обычной алгебре является знаком равенства, в данном применении выражает равносильность связываемых логических операций, так как сами функции лишены количественной меры и могут принимать лишь два качественных состояния: **0** или **1**.

На схемах, во избежание ошибок, входы бесконтактных логических элементов рисуют слева или сверху, а выходы - справа или снизу большей грани прямоугольника , условно изображающего элемент .

Простейший элемент "**НЕ**" (инвертор), является четырехполюсником. Более сложные элементы, имеющие несколько входов и выходов, - многополюсники. Но для упрощения схем общие (заземленные) клеммы входов и выходов, как и вспомогательные цепи (питания, смещения) не показывают, оставляя только используемые потенциальные вводы и выводы.

Все основные логические операции могут быть представлены через основные (элементарные) действия: **И, ИЛИ, НЕ** (рис.1).

При записи и чтении сложных логических функций предполагается, что знак инверсии связывает сильнее, чем другие знаки, а знак умножения связывает сильнее знака логического сложения. Этот принцип позволяет сокращать количество различных скобок. Например, логическую функцию:

$$X = \overline{((\ (a \cdot b) + \overline{(c + d)} \ ) \cdot c + a \ )} \cdot b$$

следует записать в более простой форме:

$$X = \overline{\left(a \cdot b + \overline{c + d}\right) \cdot c + a} \cdot b$$

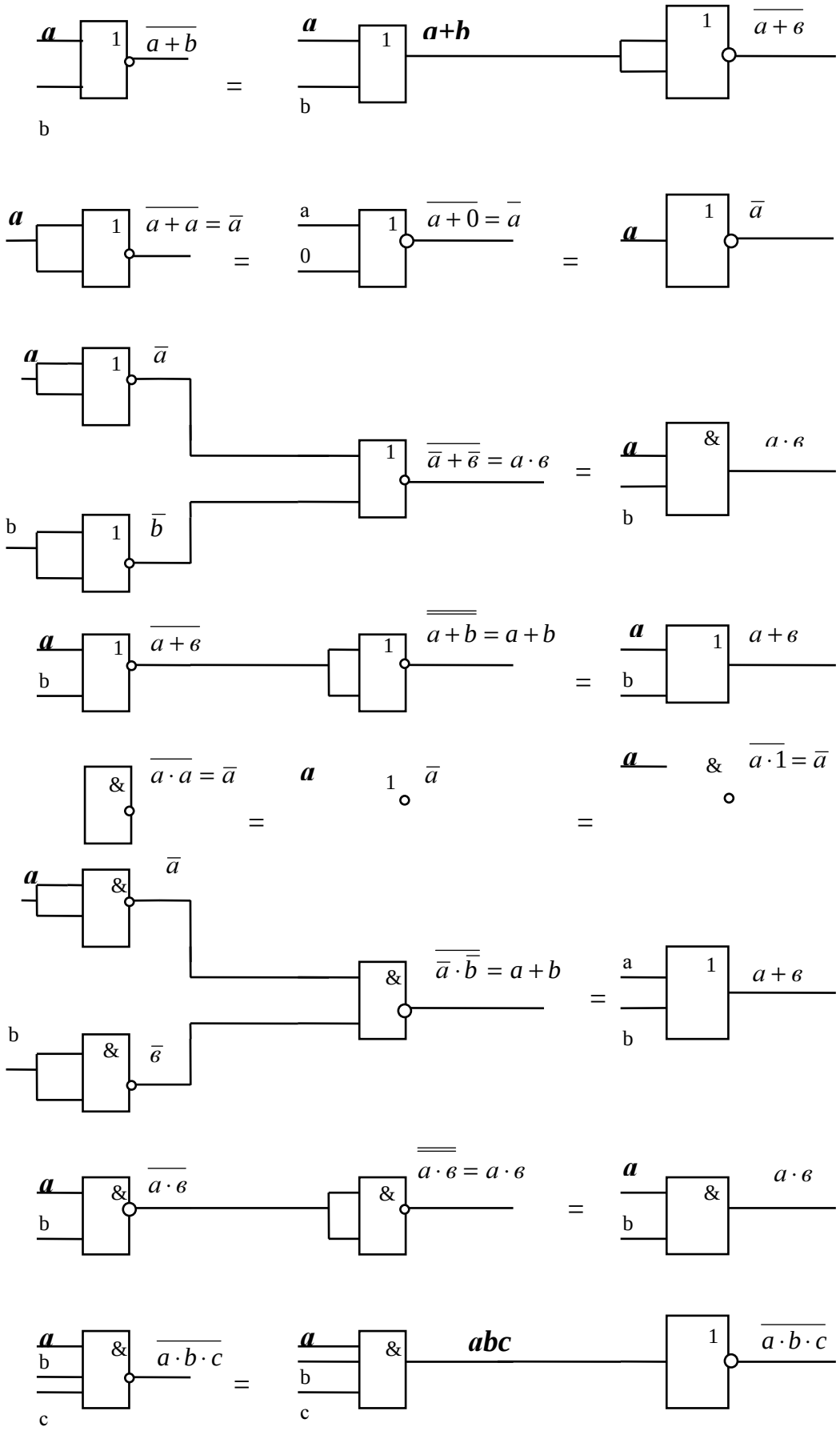


Рис. 1. Примеры элементарных логических операций

Как и в обычной алгебре здесь действуют законы:

Переместительный (коммутативный)

а) относительно логического умножения: $a\ b=b\ a;$

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

Сочетательный (ассоциативный)

а) относительно логического умножения: $(a\ b)c=(a\ c)b;$

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

Распределительный (дистрибутивный)

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

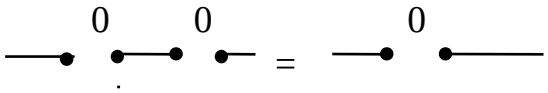
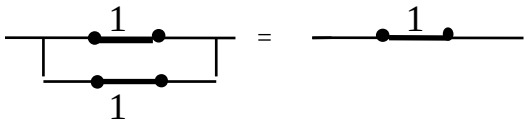
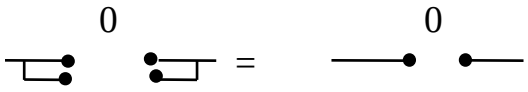
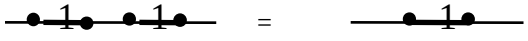
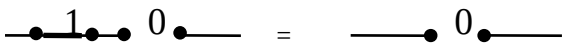
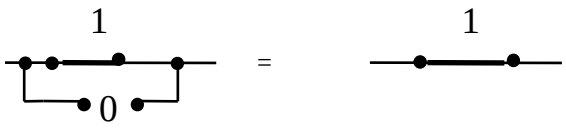
б) относительно логического сложения: $a\ b+c=(a+c)(b+c).$

Следует обратить внимание на отсутствие формальной аналогии между распределительным законом относительно логического сложения для бинарной алгебры и таким же законом для обычной алгебры.

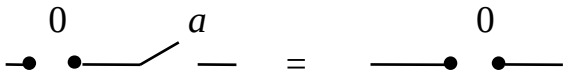
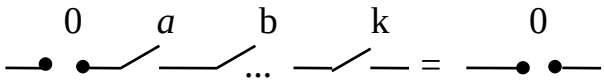
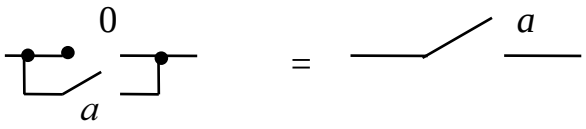
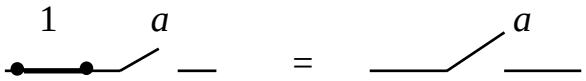
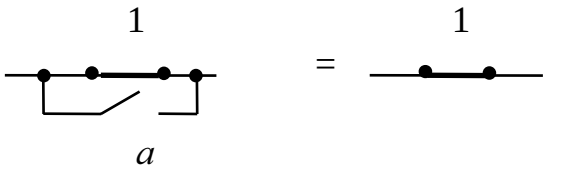
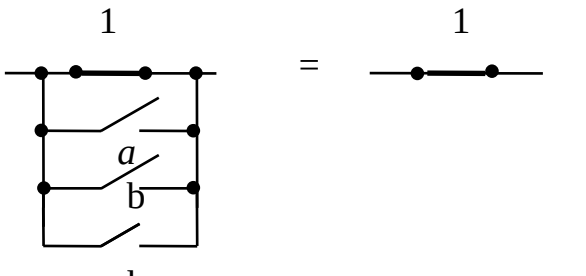
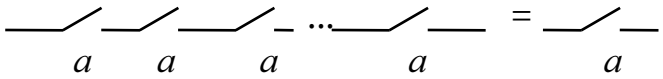
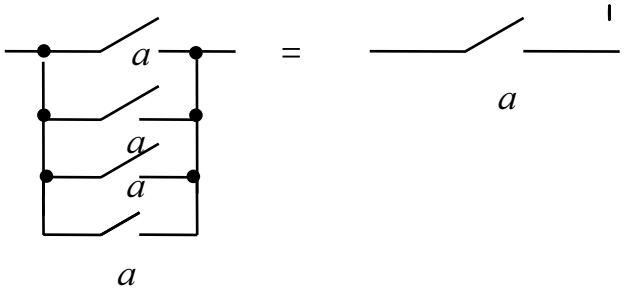
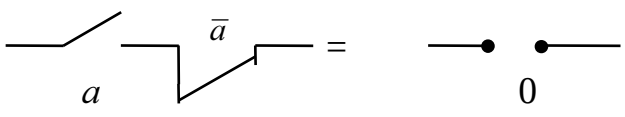
Но есть и специфические аксиомы, законы и теоремы, которые легко воспринимаются при рассмотрении соответствующих релейно-контактных схем (табл. 1).

Таблица 1.

Основные аксиомы и законы алгебры логики

№ п/п	Наименование	Формулы	Схемы
1	2	3	4
1:	Аксиомы	$0 \cdot 0 = 0$	
		$1+1= 1$	
		$0+0=0$	
		$1 \cdot 1 = 1$	
		$1 \cdot 0 = 0$	
		$1+0=1$	
		$\overline{0} = 1$	

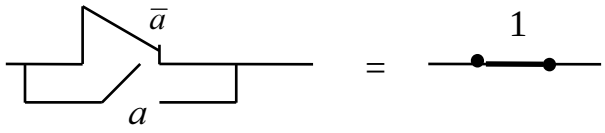
Продолжение таблицы 1.

1	2	3	4
		$\overline{1} = 0$	
2. Законы нулевого множества		$0\ a = 0$	
		$0\ a\ b...k = 0$	
		$0 + a = a$	
3.Законы универсального множества		$1\ a = a$	
		$1 + a = 1$	
		$1+a+b+...+k=1$	
4. Законы повторения		$a\ a\ a....a = a$	
		$a+a+a+...+a=a$	
5.Законы дополнительности		$a \cdot \overline{a} = 0$	

Продолжение таблицы 1.

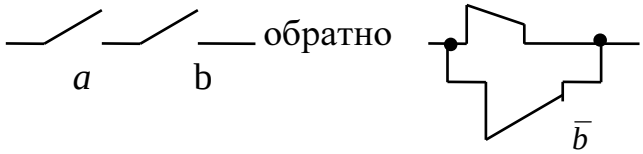
1	2	3	4

$a + \bar{a} = 1$

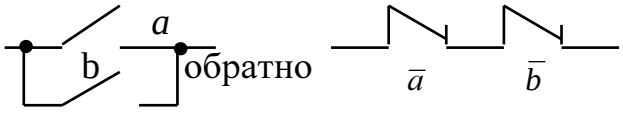


6.Законы инверсии

$\overline{a \cdot b} = \bar{a} + \bar{b}$

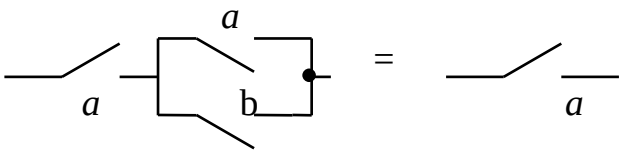


$\overline{a + b} = \bar{a} \cdot \bar{b}$

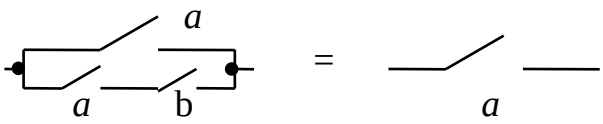


7.Законы поглощения

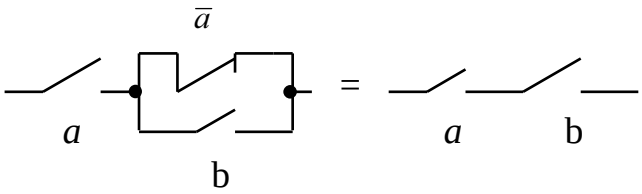
a (a+b) = a



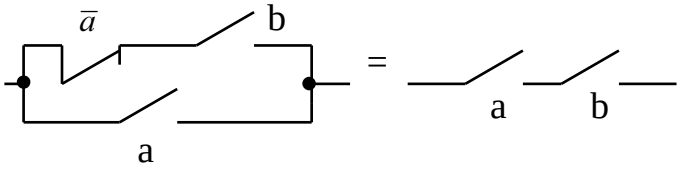
a + ab = a



$a(\bar{a} + b) = a \cdot b$



$a + \bar{a} \cdot b = a + b$

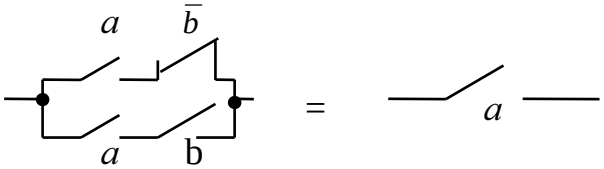


8.Закон двойного отрицания

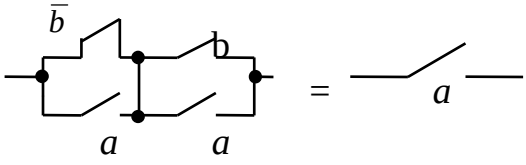
$\bar{\bar{a}} = a; \bar{\bar{0}} = 0; \bar{\bar{1}} = 1.$

9.Законы склеивания

$a \cdot b + a \cdot \bar{b} = a$



$(a + \bar{b}) \cdot (a + b) = a$



Следует напомнить, что контакты, обозначенные одинаковыми буквами, принадлежат одному реле, то есть они в идеализированном виде срабатывают одновременно. Поэтому не вызывает сомнения запись, например, закона дополнительности:

$$a \cdot \bar{a} = 0 \qquad a + \bar{a} = 1$$

Действительно, последовательно соединенные замыкающий (a) и размыкающий ($\bar{a}$) контакты одного и того же реле (**А**) всегда будут создавать разрыв цепи (**0**).

Параллельная цепь этих же контактов равносильна постоянной перемычке (шунту) с проводимостью **1**.

Очень полезными для анализа и синтеза СЛУ являются законы инверсии:

$$\overline{a\,b} = \bar{a} + \bar{b}; \quad \overline{a + b} = \bar{a}\,\bar{b}$$

Эти законы легко доказать методом перебора всех возможных комбинаций переменных a и b . Если окажется, что для каждой комбинации переменных логические функции совпадут, то они равносильны. Например, рассмотрим первый закон инверсии, для чего составим таблицу состояний, в которой число различных комбинаций входных сигналов равно четырем (первый и второй столбцы), а в остальных столбцах таблицы приведены результаты элементарных логических операций. Таблица состояний представлена в таблице 2.

Таблица 2.

Таблица состояний

1	2	3	4	5	6	7
a	b	ab	$\overline{a \cdot b}$	$\bar{a}$	$\bar{b}$	$\bar{a} + \bar{b}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Столбцы третий, пятый и шестой - вспомогательные, содержащие результаты промежуточных вычислений. Как видно из таблицы значения ab и $a + b$ полностью совпадали для каждой комбинации переменных a и b.

Законы инверсии справедливы для любого числа переменных, причем представленных как в нормальной, так и инверсной форме :

$$\overline{\bar{a} \cdot \bar{b} \cdot \bar{c}} = \bar{\bar{a}} + \bar{\bar{b}} + \bar{\bar{c}} = a + b + c \;; \qquad \overline{a + 0} = \bar{a} \cdot \bar{0} = \bar{a} \cdot 1 = \bar{a}$$

$$\overline{\bar{a} + \bar{b} + \bar{c}} = \bar{\bar{a}} \cdot \bar{\bar{b}} \cdot \bar{\bar{c}} = a \cdot b \cdot c; \qquad \overline{a \cdot 1} = \bar{a} + \bar{1} = \bar{a} + 0 = \bar{a}$$

$$\overline{a + b \cdot c + \bar{d}} = \bar{a} \cdot \overline{b \cdot c} \cdot \bar{\bar{d}} = \bar{a}\left(\bar{b} + \bar{c}\right)d; \quad \overline{\overline{a \cdot b} \cdot c} = \bar{\bar{a}}\bar{\bar{b}} + \bar{c} = ab + \bar{c}\,.$$

2. ПРИМЕР ПРОЕКТИРОВАНИЯ СЛУ.

Система логического управления содержит три приёмных элемента A,B,C и два исполнительных элемента X,Y.

Алгоритм работы системы следующий:

Для элемента X (в нашем примере маломощный электродвигатель постоянного тока):

1. Элемент X срабатывает, если срабатывают A,B, но не срабатывает C.
2. Элемент X срабатывает, если срабатывают A,C, но не срабатывает B.
3. Элемент X срабатывает, если срабатывает A, но не срабатывают B и C.

Для элемента Y (в нашем примере сигнальная лампа):

1.Y срабатывает, если срабатывает A, но не срабатывают B и C;
2.Y срабатывает, если срабатывает B, но не срабатывают A и C;

2.1 РЕШЕНИЕ. Составляем основной элемент синтеза - таблицу состояния.

В таблице состояния рассматриваем все возможные комбинации состояний приемных элементов. Так как приемных элементов три - то возможное число комбинаций состояния равно восьми, т.е. имеем восемь строк в таблице состояний. Состояние исполнительных элементов записываем в соответствии с алгоритмом: если элемент срабатывает - ставится 1, в противном случае - 0.

Таблица 1.

	A	B	C	X	Y
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	1
4	0	1	1	0	0
5	1	0	0	1	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	0	0

Перейдем к составлению логической функции. Для этого составим частные условия срабатывания для элемента X:

$$X_1 = a \cdot \overline{b} \cdot \overline{c};$$

$$X_2 = a \cdot \overline{b} \cdot c;$$

$$X_3 = a \cdot b \cdot \overline{c}.$$

Общие условия срабатывания запишем как дизъюнкцию частных условий срабатывания. Это означает, что элемент X сработает, если будет выполнено или одно частное условие срабатывания, или все частные условия, или их комбинация.

$$X = X_1 + X_2 + X_3 = a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c} = a \cdot \bar{b}\left(\underbrace{c + \bar{c}}_{1}\right) + a \cdot b \cdot \bar{c} = a \cdot \bar{b} + a \cdot b \cdot \bar{c} = a\left(\bar{b} + b \cdot \bar{c}\right) = a\left(\bar{b} + \bar{c}\right).$$

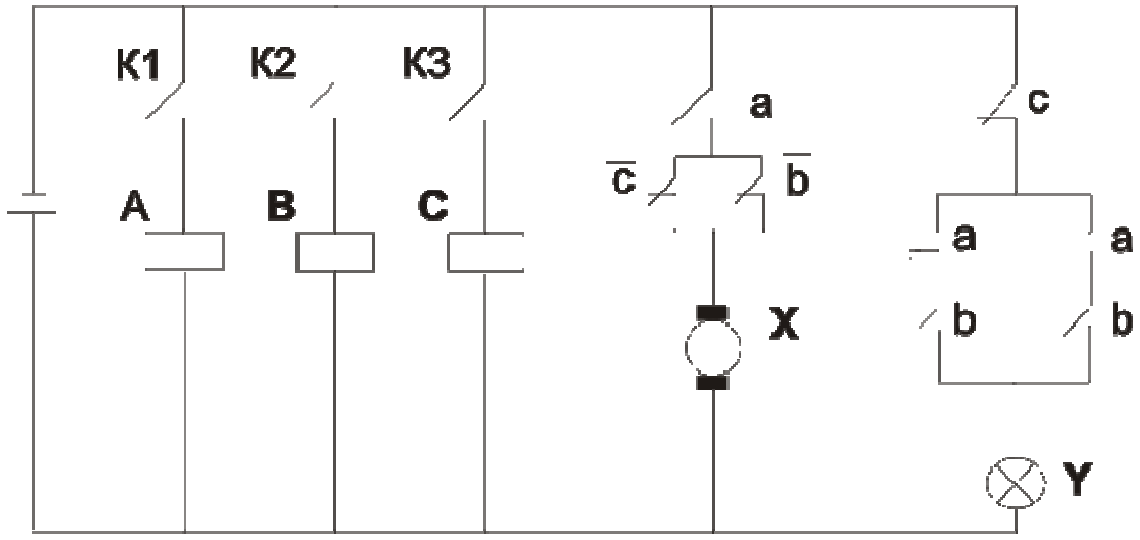
Аналогично составляем логическую функцию для элемента Y.

$$Y_1 = \bar{a} \cdot b \cdot \bar{c};$$

$$Y_2 = a \cdot \bar{b} \cdot \bar{c};$$

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

Релейно-контактные варианты проектируемой системы логического управления имеют следующий вид:



Напоминаем, что если приемный элемент, например А, не сработал, то состояние его замыкающих контактов будет равносильно 0, а размыкающих – 1, т.е. а=0, $\bar{a}$=1.

При срабатывания А произойдет инверсия, т.е. а=1 и $\bar{a}$=0.

2.2. Проверим условия срабатывания и несрабатывания X.

Для первой строки таблицы состояний (A=0, B=0, C=0):

$$X = a\left(\bar{b} + \bar{c}\right) = 0(1+1) = 0$$

Для второй строки (A=0, B=0, C=1):

$$X = a\left(\bar{b} + \bar{c}\right) = 0(1+0) = 0$$

Для третьей строки (A=0, B=1, C=0):

$$X = a\left(\bar{b} + \bar{c}\right) = 0(0+1) = 0$$

Для четвертой строки (A=0, B=1, C=1):

$$X = a\left(\bar{b} + \bar{c}\right) = 0(0+0) = 0$$

Для пятой строки (A=1, B=0, C=0):

$$X = a\left(\bar{b} + \bar{c}\right) = 1(1+1) = 1$$

Для шестой строки (A=1, B=0, C=1):

$$X = a\left(\bar{b} + \bar{c}\right) = 1(1+0) = 1$$

Для седьмой строки (A=1, B=1, C=0):

$$X = a\left(\bar{b} + \bar{c}\right) = 1(0+1) = 1$$

Для восьмой строки (A=1, B=1, C=1):

$$X = a\left(\bar{b} + \bar{c}\right) = 1(0+0) = 0$$

Аналогично можно проверить эти условия для Y.

Бесконтактный вариант СЛУ выполним в базисе элементов И-НЕ. Для этого следует логические функции для X и Y записать в более удобном для выбранного базиса с использованием очевидных преобразований:

$$X = a\left(\bar{b} + \bar{c}\right) = a \cdot \overline{b \cdot c}$$

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

Наметим шины a,b и c , на которые будут поступать потенциалы от приемных элементов.

Легко представить бесконтактный вариант проектируемой СЛУ (рис.2).

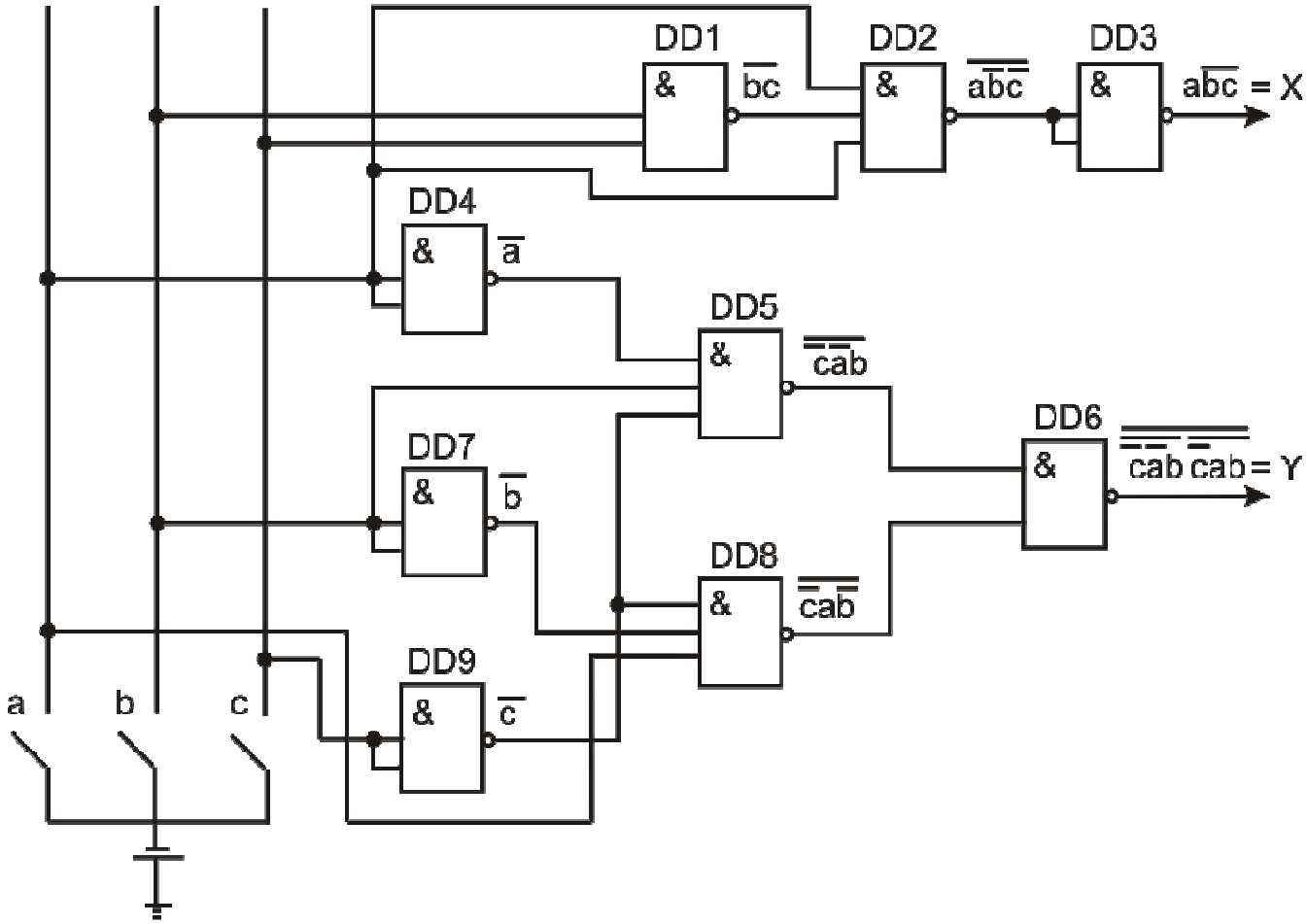


Рис. 2. Бесконтактная СЛУ в базисе И-НЕ.

Следовательно, для реализации этой схемы потребуется 9 элементов И-НЕ.

Если бы потребовалось проектируемую СЛУ выполнить в базисе элементов ИЛИ-НЕ, то следовало бы выполнить следующие очевидные преобразования:

$$X = a\left(\bar{b} + \bar{c}\right) = a \cdot \bar{b} + a \cdot \bar{c} = \overline{\bar{a} + b} + \overline{\bar{a} + c} = \overline{\overline{\bar{a} + b} \; \overline{\bar{a} + c}}.$$

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

Для реализации функции X потребовалось бы 5 элементов, а для функции Y – 6 элементов ИЛИ-НЕ, имеющих по три входа.

3. ОСНОВНЫЕ СВЕДЕНИЯ ПО CODESYS

CoDeSys - это современный инструмент для программирования контроллеров (**CoDeSys** образуется от слов **Co**ntrollers **De**velopment **Sys**tem).

CoDeSys предоставляет программисту удобную среду для программирования контроллеров на языках стандарта МЭК 61131-3. Используемые редакторы и отладочные средства базируются на широко известных и хорошо себя зарекомендовавших принципах, знакомых по другим популярным средам профессионального программирования (такие, как Visual C++).

Прежде всего, перед началом работы надо убедиться в том, что среда **CoDeSys** установлена на компьютере. В случае ее отсутствия необходимо произвести установку (среда **CoDeSys** находится в свободном распространении на официальном сайте ОВЕН www.owen.ru).

После этого произведем инсталляцию TSP (Target Support Packages). В TSP включены все файлы, необходимые CoDeSys для создания кода, отладки и конфигурирования аппаратуры. Платформа определяет параметры генератора кода, распределение памяти, функциональность ПЛК, модули ввода-вывода. Кроме того, в TSP могут входить дополнительные библиотеки, драйверы связи, ini-файлы сообщений об ошибках и список команд ПЛК-Браузера. Центральным компонентом TSP является один или несколько целевых файлов (Target files). В нем присутствуют данные о всех дополнительных файлах, необходимых для конфигурирования данной платформы. Для инсталляции TSP запускаем утилиту **InsallTarget,** которая обычно входит в стандартный пакет среды **CoDeSys**.

В открывшемся при запуске утилиты **InstallTarget** окне (Рис. 3) – нажать кнопку **Open** и указать путь доступа к инсталлируемому Target-файлу (имеющему расширение ***.tnf**). Target-файлы контроллеров ОВЕН ПЛК100 находятся на компакт-диске, поставляемом с контроллером, в папке «Target» или могут быть скачаны с сайта **www.owen.ru**.

После открытия требуемого файла в области **«Possible Targets»** окна отобразится папка **«Owen»**.

Открыв папку **«Owen»** и выделив находящуюся там строку, нажать кнопку **Install**. В области **«Installed Targets»** окна отобразится список инсталлированных Target-файлов. На этом установку можно считать законченной.

В лабораторных стендах все эти процедуры были выполнены, установлена среда CoDeSys и проинсталлированы используемые TSP.

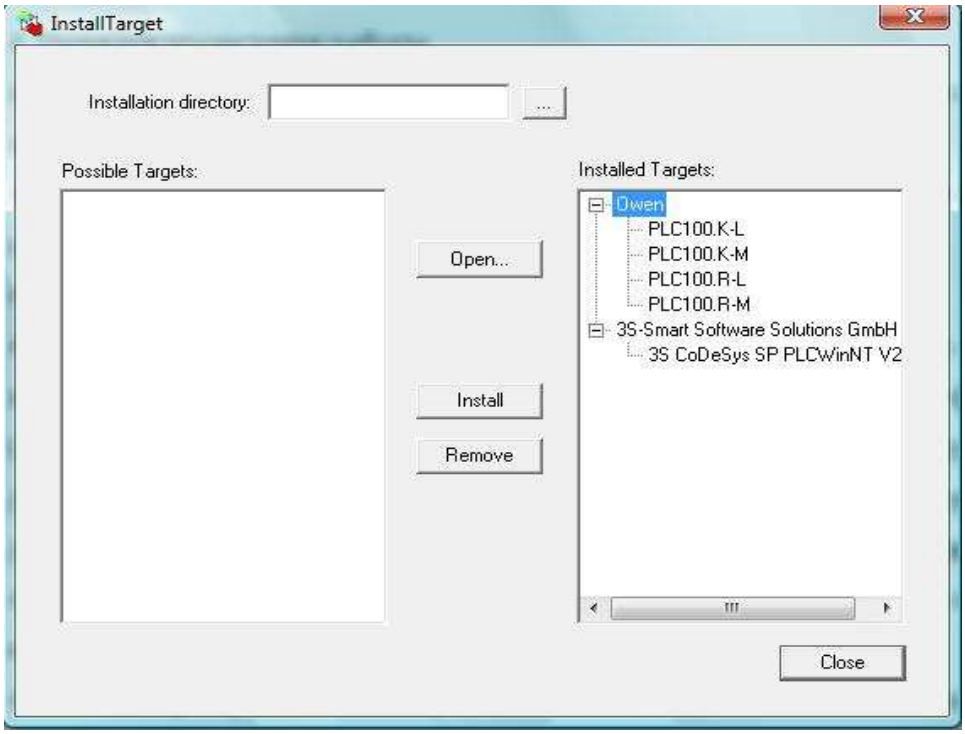


Рис. 3 Окно **«InstallTarget»** утилиты **InstallTarget**

Запустить программную среду **CoDeSys.** В окне **Target Settings** напротив строки **Configuration** выбрать тип логического контроллера **PLC 100.R-L**, т.к. именно он используется в лабораторных работах, и нажать **OK**. В появившемся окне **New POU** (Рис. 4) выбрать тип POU Программа (Program) и язык, на котором будет осуществляться написание программы **LD** (релейные диаграммы). Имя программы оставить без изменения. Подтвердить выбор нажатием на кнопку **OK**.

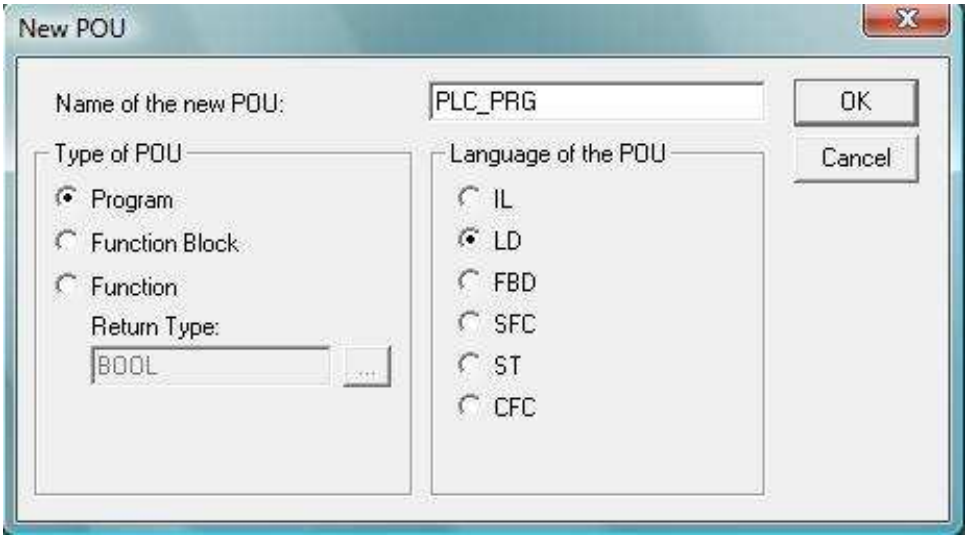


Рис. 4 Выбор языка программирования и задание имени программы

После выполнения всех вышеописанных действий откроется главное окно (Рис. 5) среды **CoDeSys**.

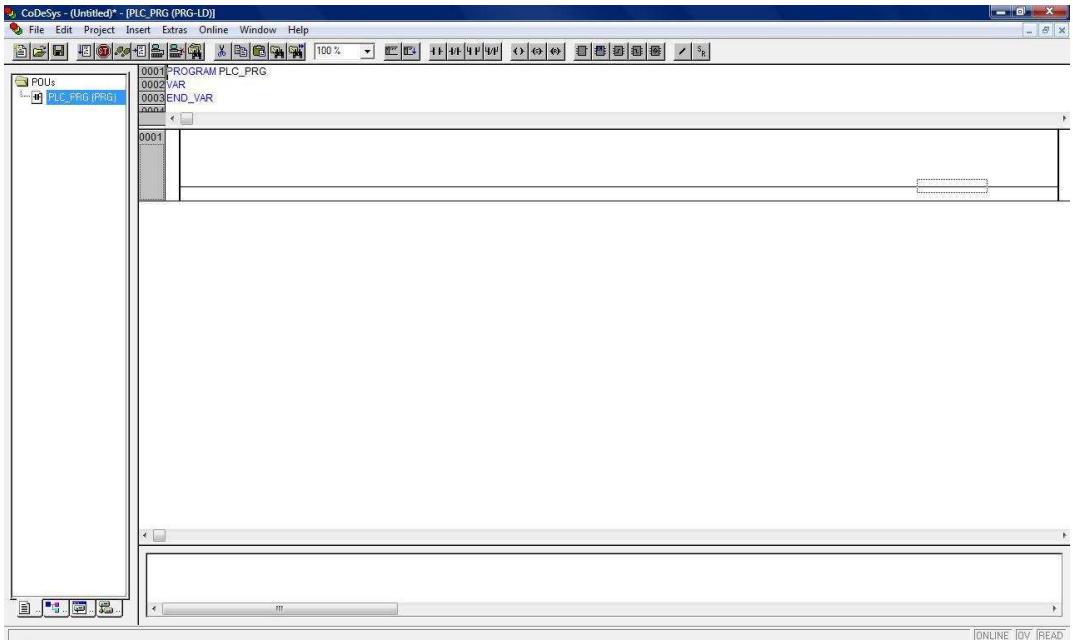


Рис. 5 Главное окно среды **CoDeSys**

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

- Меню (Рис.6).
- Панель инструментов. На ней находятся кнопки для быстрого вызова команд меню (Рис.7).
- Организатор объектов, имеющий вкладки POU, Data types, Visualizations и Resources.
- Разделитель Организатора объектов и рабочей области CoDeSys.
- Рабочая область, в которой находится редактор.
- Окно сообщений.
- Строка статуса, содержащая информацию о текущем состоянии проекта.

Рис. 6 Меню среды

Меню находится в верхней части главного окна. Оно содержит все команды **CoDeSys**.

Рис. 7 Панель инструментов

Кнопки на панели инструментов обеспечивают более быстрый доступ к командам меню. Вызванная с помощью кнопки на панели инструментов команда автоматически выполняется в активном окне.

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.

При желании панель инструментов можно отключить (см. 'Project' 'Options' категория Desktop убрать галочку Tool bar).

Рис. 8 Организатор объектов

Организатор объектов всегда находится в левой части главного окна CoDeSys. В нижней части организатора объектов находятся вкладки POUs, Data types (типы данных), Visualizations (визуализации) и Resources (ресурсы). Переключаться между соответствующими объектами можно с помощью мышки, нажимая на нужную вкладку.

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.

Рис. 9 Рабочая область

Рабочая область 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.

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 мышкой или при нажатии клавиши <Enter > на сообщении будет открыт объект, к которому относиться 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 и 2ПКМ, если один или два щелчка правой кнопкой.)

С помощью команд “Edit” ”Next error” и ”Edit ” ” Previous error ” можно быстро перемещаться между сообщениями об ошибках.

Окно сообщений можно убрать либо включить с помощью команды “Window” “ Message”.

Статусная строка

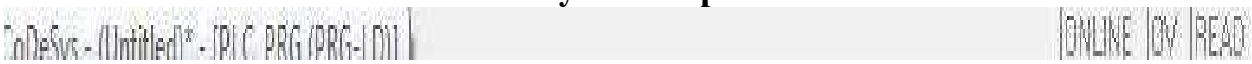


Рис. 10 Статусная строка

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

При выборе пункта меню его описание появляется в левой части строки статуса.

Если вы работаете в режиме online, то надпись Online в строке статуса выделяется черным цветом. В ином случае надпись серая. С помощью статусной строки в режиме online можно определить, в каком состоянии находится программа: SIM – в режиме эмуляции, RUN – программа запущена, BP- установлена точка останова, FORCE – происходит фиксация переменных.

При работе в текстовом редакторе в строке статуса указывается позиция, в которой находится курсор (например, Line:5, Col.:11). В режиме замены надпись “OV” выделяется черным цветом. Нажимая клавишу <Ins> можно переключаться между режимом вставки и замены.

В визуализации в статусной строке выводятся координаты курсора X и Y, которые отсчитываются относительно верхнего левого угла окна. Если указатель мыши находится на элементе или над элементом производятся какие-либо действия, то указывается номер этого элемента. При вставке элемента в строке статуса указывается его название (например, Rectangle).

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

Статусную строку можно убрать либо включить (см. 'Project' 'Options' категория Desktop).

Контекстное меню

Быстрый вызов: <Shift>+<F10>

Вместо того, чтобы использовать главное меню для вызова команд, можно воспользоваться контекстным меню. Это меню, вызываемое 1ПКМ на рабочей области, содержит наиболее часто используемые команды.

Приступим к разработке программы. В случае если необходимо будет реальное подключение компьютера к ПЛК, то необходимо будет ввести переменные в **Конфигуратор ПЛК**. Если же предполагается работа в режиме ***эмуляции***, то этот шаг можно пропустить.

Для внесения переменных в **Конфигуратор ПЛК** (Рисунок 11) под окном ***Организатор объектов*** необходимо нажать 1ЛКМ на вкладку **Resources** и из дерева ресурсов выбрать **PLC Configuration**. В конфигураторе в левом окне раскрыть вкладку **PLC100.R**, щелкнув 1ЛКМ по символу «+». Потом 2ЛКМ щелкнуть по **Discrete Input 8 bit**. Нажав на «+», раскрыть вкладку **AT %IB0.0: BYTE.** Далее щелкнуть 2ЛКМ по надписи **AT** строчки **AT %IB0.0.0: BOOL** и ввести имя переменной. Аналогичным образом, но только уже в строки **AT %IB0.0.1: BOOL**, **AT %IB0.0.2: BOOL**, **AT %IB0.0.3: BOOL**, **AT %IB0.0.4: BOOL** ввести имена остальных входных переменных.

Для ввода выходных переменных необходимо нажать на «+» возле строки **Discrete Output-Relay**. Щелкнуть 2ЛКМ по надписи **AT** в строке **AT %QX1.0: BOOL** и ввести имя первой выходной переменной. Аналогичным образом ввести остальные имена переменных в строки **AT %QX2.0: BOOL**, **AT %QX3.0: BOOL**, **AT %QX4.0: BOOL**, **AT %QX5.0: BOOL**, **AT %QX6.0: BOOL**.

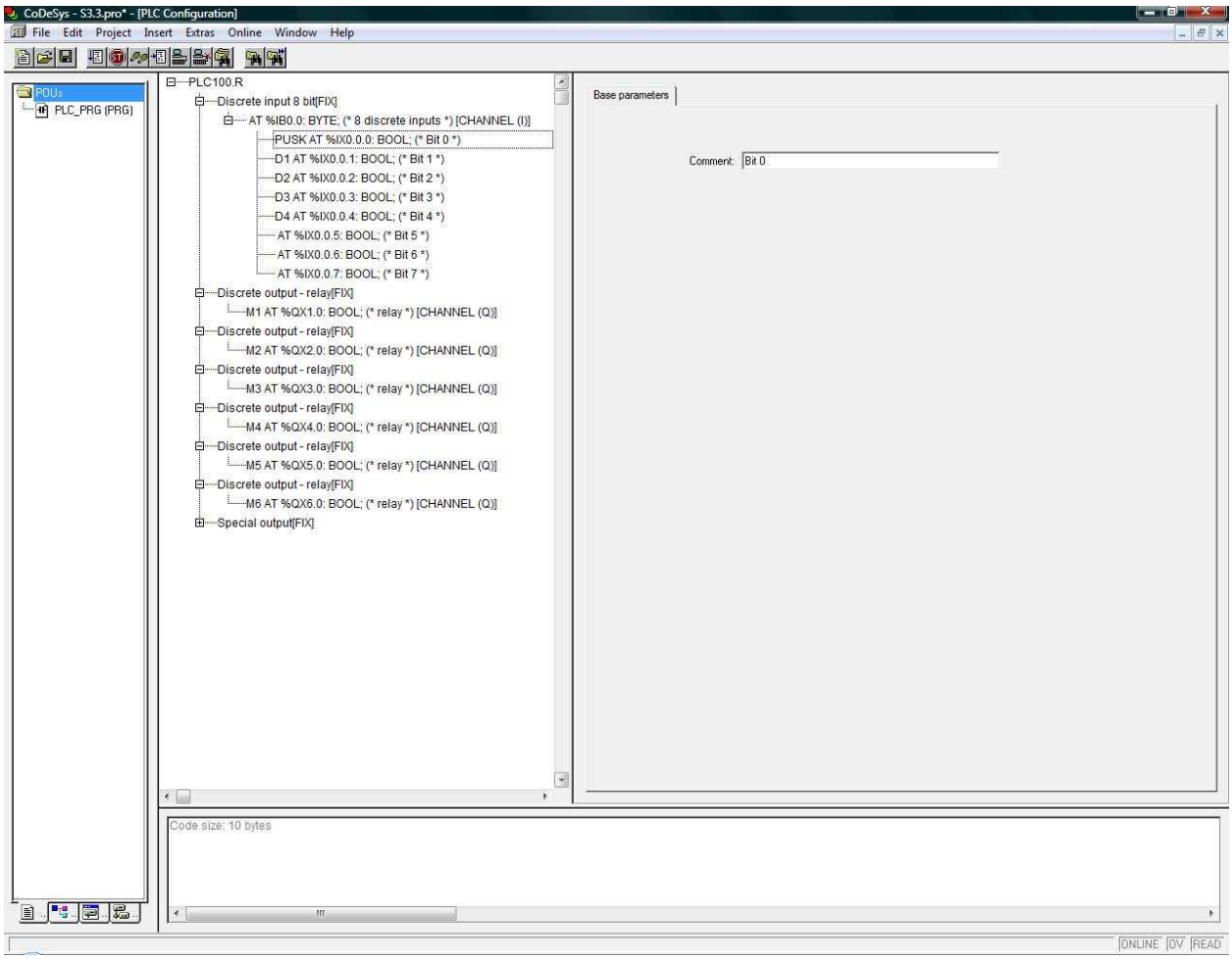


Рис. 11 Конфигуратор PLC 100 R-L

Теперь приступаем к проектированию самой программы.

Указания по методике построения схем LD.

После открытия главного окна среды CoDeSys на экране монитора появится первая цепь проектируемой системы. Это горизонтальная линия между двумя вертикальными шинами, которую надо заполнить необходимыми элементами: контактами, функциональными блоками FB, катушками реле.

Наводим курсор на линию цепи, щелкаем 1ЛКМ, переводим курсор на необходимую кнопку в правой части панели инструментов, на замыкающий контакт  , нажиманием 1 ЛКМ и этот контакт появится в цепи с «???» красного цвета (рис. 12)

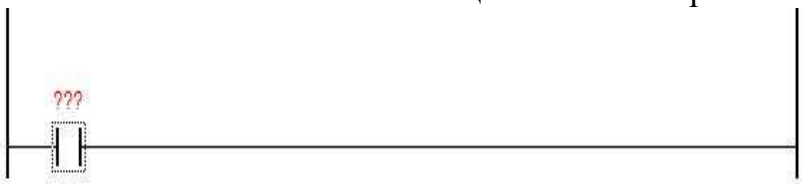


Рис.12 Замыкающий контакт

Наводим курсор на «???», щелкаем 1ЛКМ и клавиатурой набираем имя, например PUSK, нажимаем <Enter>.

Открывается окно Declare Veriable (Объявление переменной), запрашивающая к какому классу переменных будет отнесен наш элемент (рис. 13)

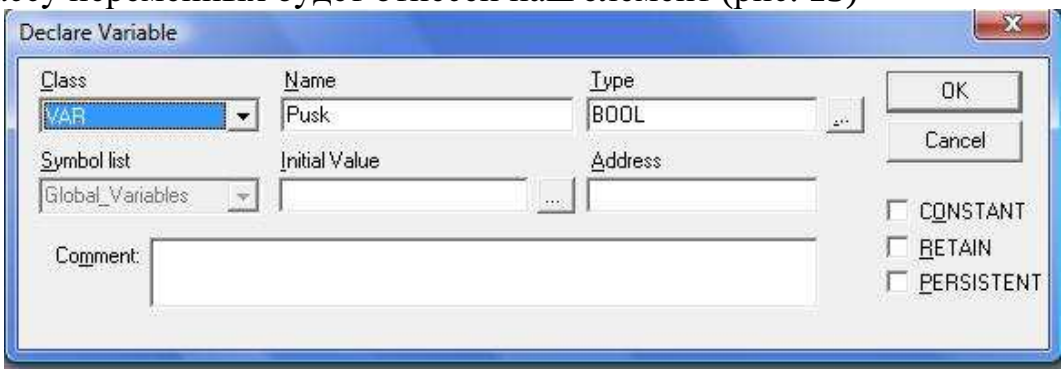


Рис.13 Окно объявления переменной

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

Рис.14 Объявленный замыкающий контакт

Если же проектируемую схему необходимо потом реализовывать на реальном ПЛК, то следует навести курсор на синюю строку в разделе ***Class***, нажав ЛКМ и сместившись вниз, открываем строки этого раздела. Останавливаемся на строке VAR_INPUT, если объявленный элемент будет подключен к входу ПЛК или на строке VAR_OUTPUT, если к входу. Нажимаем 1 ЛКМ на ОК.

Программа в среде CoDeSys выглядит следующим образом (рис. 15):

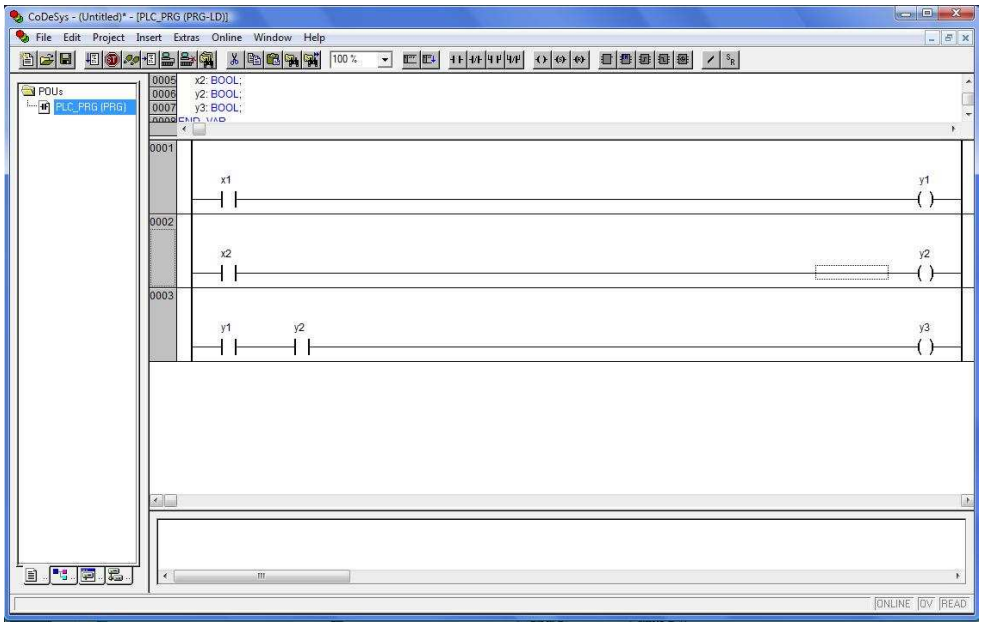


Рис. 15 Пример схемы в LD

а) б)

Рис. 16 – Контекстное меню программы **CoDeSys**

а) контекстное меню на английском языке;

б) контекстное меню на русском языке.

Редактор LD – это графический редактор. Наиболее важные команды расположены в контекстном меню (Рис. 16), которое вызывается 1ПКМ или сочетанием клавиш <Ctrl> + <F10>. На рисунке представлены меню из версий CoDeSys на английском и русском языках соответственно.

Элементы схемы можно в процессе программирования перемещать по цепи или менять их наименования с помощью перетаскивания его мышью (drag & drop).

Для переноса элемента, «захватив» элемент курсором и при нажатой ЛКМ, перемещают его в новое место.

В этот момент на входе и выходе каждого элемента высвечиваются небольшие прямоугольники (Рис. 17 – а).

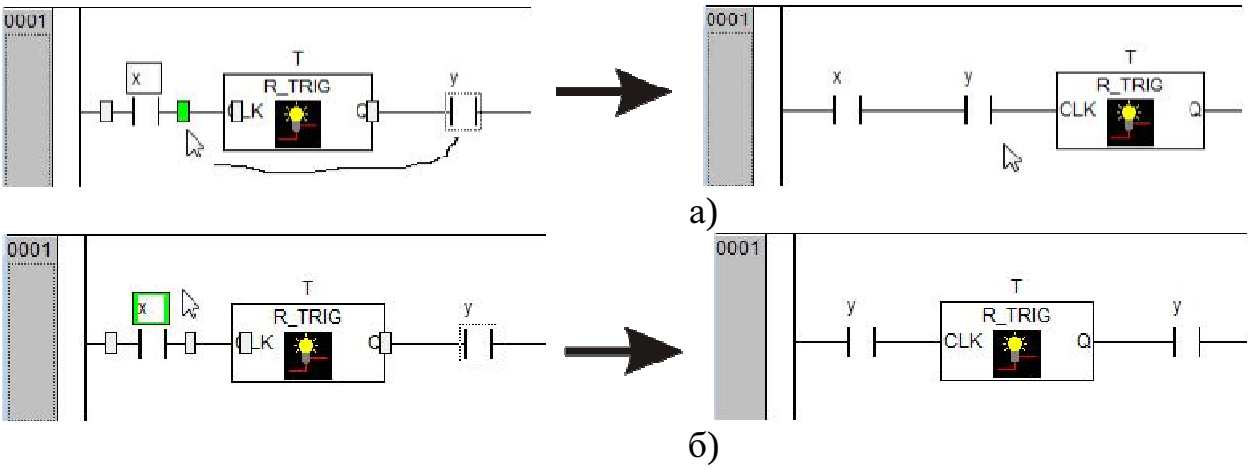


Рис. 17 – Вид цепи при перетаскивании элемента

Как только курсор подойдет к одному из них, прямоугольник становится зеленым. Отпускаем ЛКМ и перемещаемый элемент займет новое место. Если же нужно заменить какой-либо объявленный элемент схемы другим (рис.17 – б) нужно переместить элемент, как в вышеописываемом случае, но уже не в «прямоугольник» а в название изменяемого элемента.

Создание программы в CoDeSys.

Для создания программы следует открыть среду CoDeSys и провести все необходимые действия, которые детально описаны выше.

К примеру, необходимо составить программу, имеющую следующий вид (Рис. 18).

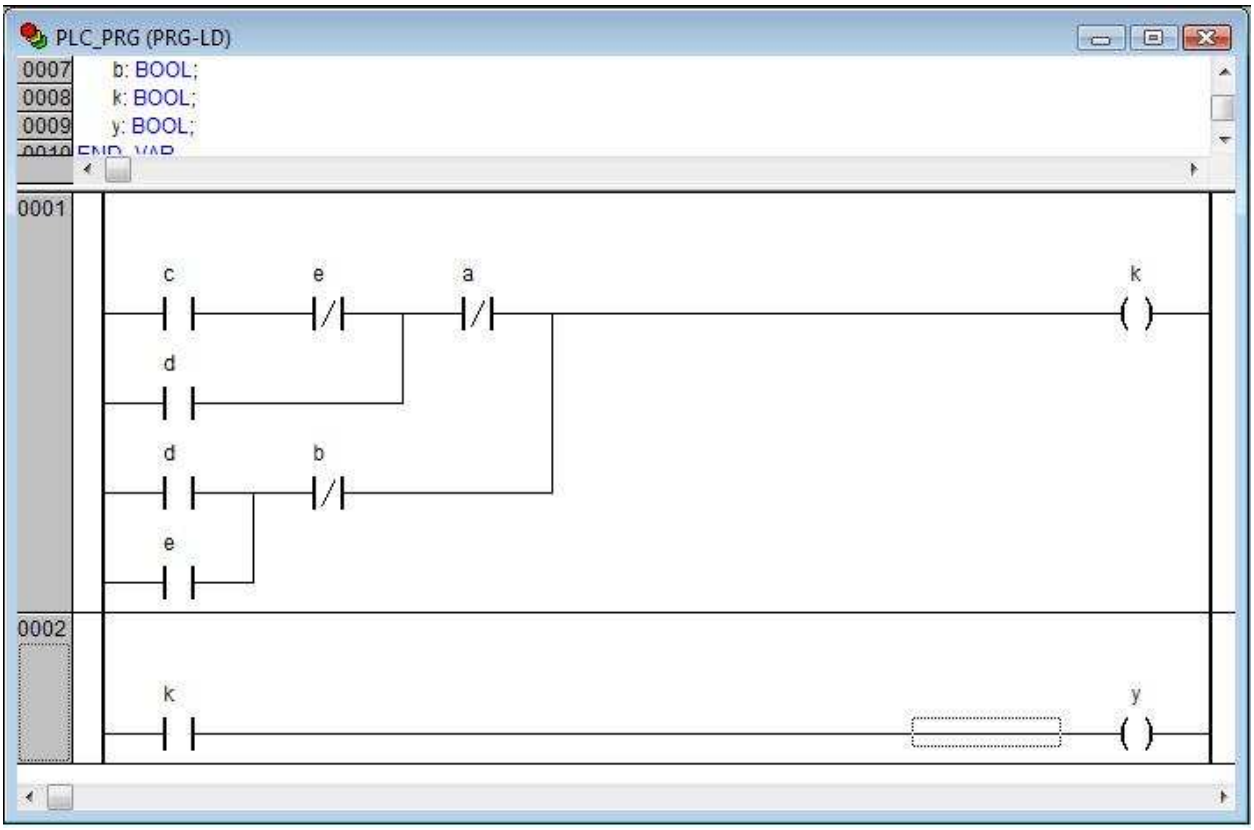


Рис.18 Программа

Для начала следует нажать 1ЛКМ на появившейся цепи программы (тем самым сделать ее активной), далее эту процедуру будем называть ***Активизацией цепи***.

Для нее явно потребуются 2 цепи. Для их добавления существует три способа, рассмотрим каждый:

Номер	**Панель**	**Добавление цепи выше**	**Добавление цепи ниже**
1.	**Меню (INSERT)**	**Network (before)**	**Network (after)**
2.	**Панель инструментов**		
3.	**Контекстное меню**	**Network (before)**	**Network (after)**

Обе цепи появились, сделаем первую активной, нажав 1ЛКМ на ней. Для вставки элемента перевести курсор мыши на панель инструментов и нажать 1ЛКМ на замыкающем контакте  , объявим его именем «**c**», пользуясь вышеописанной методикой. Далее следует перевести курсор для вставки следующего элемента, для этого нажмем на цепи 1ЛКМ, появится характерный прямоугольник из точек в конце цепи (Рис.19).

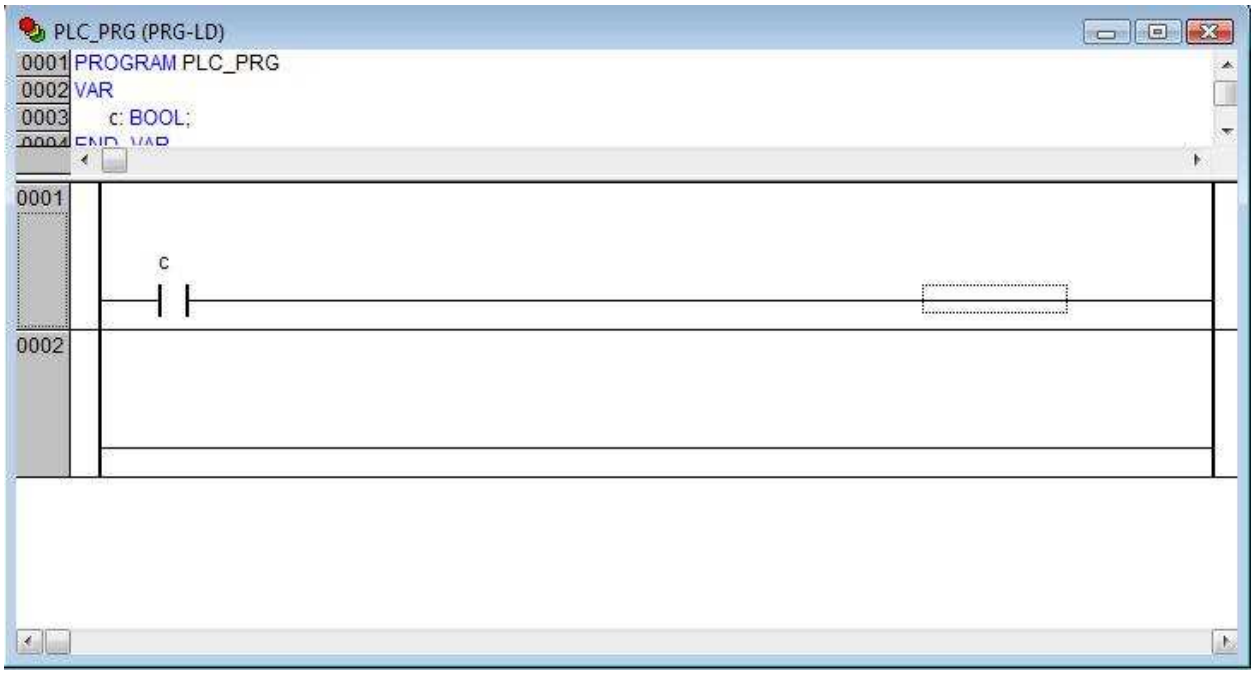


Рис. 19 Замыкающий контакт «**c**» и
характерный прямоугольник» в конце цепи

Добавим следующий элемент – размыкающий контакт. Для этого курсором мыши переведем в **Панель инструментов** и выберем кнопку  и объявим появившийся элемент именем «**e**». Далее сделаем активной ветвь. На схеме уже 2 элемента, теперь нужно параллельно им поместить замыкающий параллельный контакт, для этого выделим эти два контакта: нажмем 1ЛКМ на одном из них и, удерживая кнопку <**Shift**>, нажмем на второй контакт 1ЛКМ, они оба будут выделены «характерным прямоугольником». Далее следует щелкнуть 1ЛКМ на кнопке  в **Панели инструментов**, объявляем появившийся элемент именем «**d**». Активизируем цепь. Далее следует размыкающий контакт, который следует добавить аналогичным способом и объявить его именем «**a**». После выполнения этого действия, схема будет выглядеть следующим образом (Рис.20)

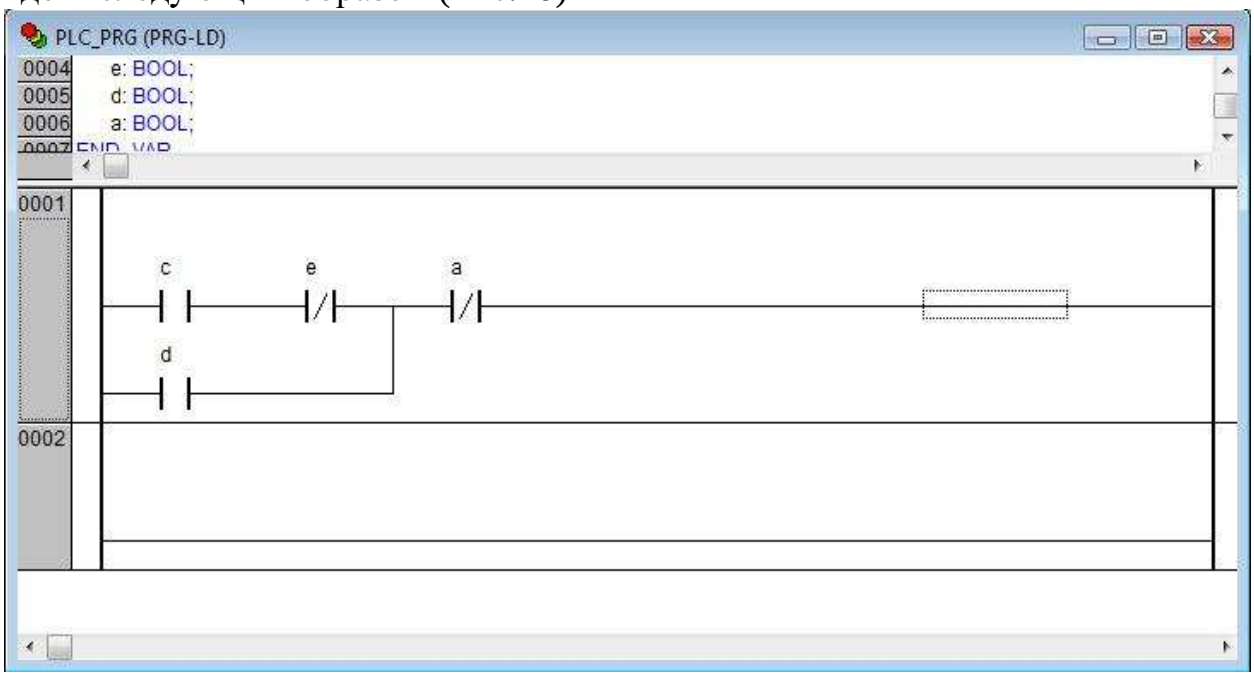


Рис.20 - Промежуточное состояние программы

Из рисунка видно, что мы имеем схему из 4 контактов, параллельно им нужно вставить размыкающий параллельный контакт, методика следующая: нажимаем 1ЛКМ на элементе «**a**», удерживая кнопку «**Shift**» нажмем на любой из оставшихся контактов, после чего вся схема выделится «характерным прямоугольником». Далее нажимаем на значок размыкающего контакта в **Панели инструментов**, появившемуся контакту присваиваем имя «**d**».

Следующим элемент – размыкающий контакт, поместить в схеме его можно следующим образом: щелкнуть 1ЛКМ на контакте «**d**»(нижний) и перевести курсор манипулятора в Панель инструментов и нажать на значок  размыкающего параллельного контакта, присвоить ему имя «**b**». Далее нужно перенести его дальше контакта «**d**», эту процедуру можно сделать, используя методику описанную выше. Программа на данном этапе будет выглядеть следующим образом (рис. 21):

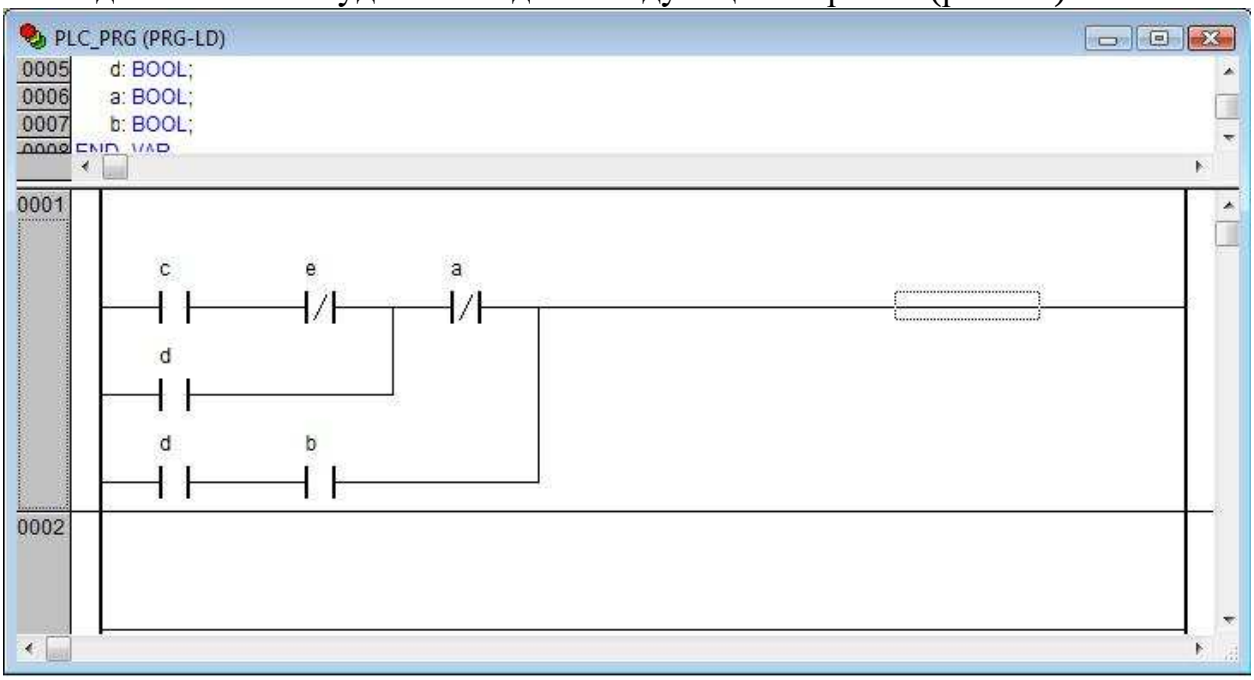


Рис.21 Промежуточное состояние программы

Далее параллельно контакту «**d**» объявим замыкающий параллельный контакт «**e**», используя уже известный значок  из **Панели инструментов** (обязательно выделить при этом только один контакт «**d**», чтобы добавляющийся элемент был параллельно только ему). Сделать ***активной*** цепь!

Заключая первую цепь, необходимо установить катушку реле, для этого курсором на Панели инструментов найти значок  и нажать 1ЛКМ. Далее следует присвоить новому элементу имя «**к**».

Для перехода на новую цепь следует нажать 1ЛКМ на этой цепи и на ней появится «характерный прямоугольник». После всех проведенных процедур программа будет иметь следующий вид (рис.22):

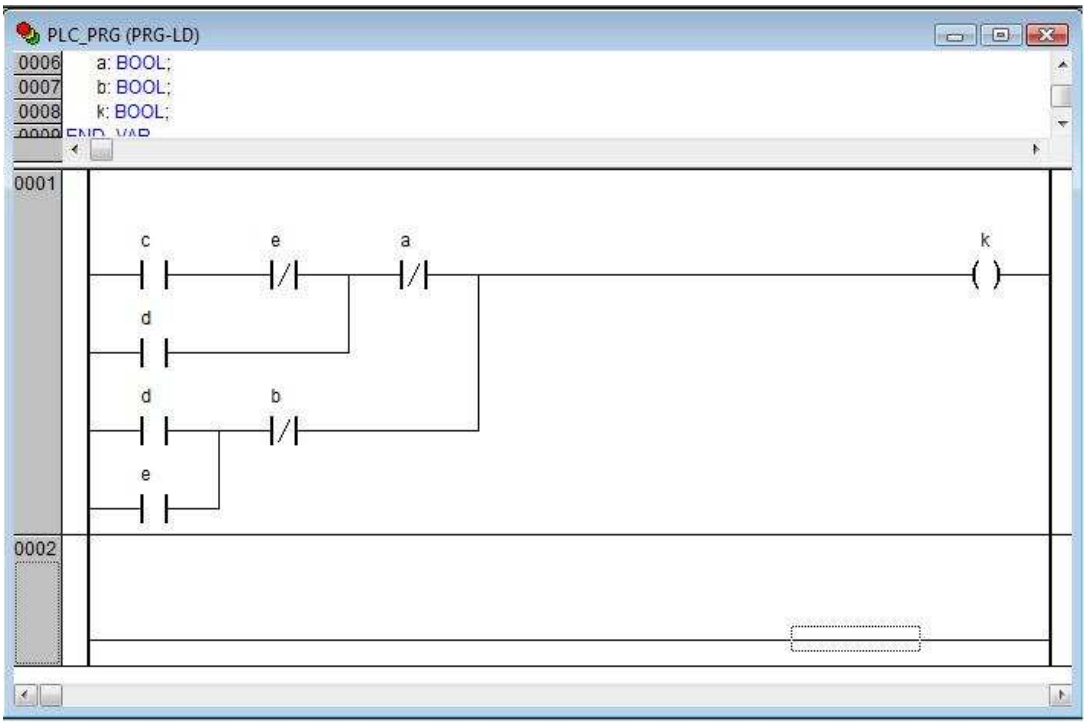


Рис.22 Переход ко второй цепи

Во второй цепи необходимо вставить замыкающий контакт «**к**» и катушку реле «**у**». После чего программа будет иметь окончательный вид (рис.23).

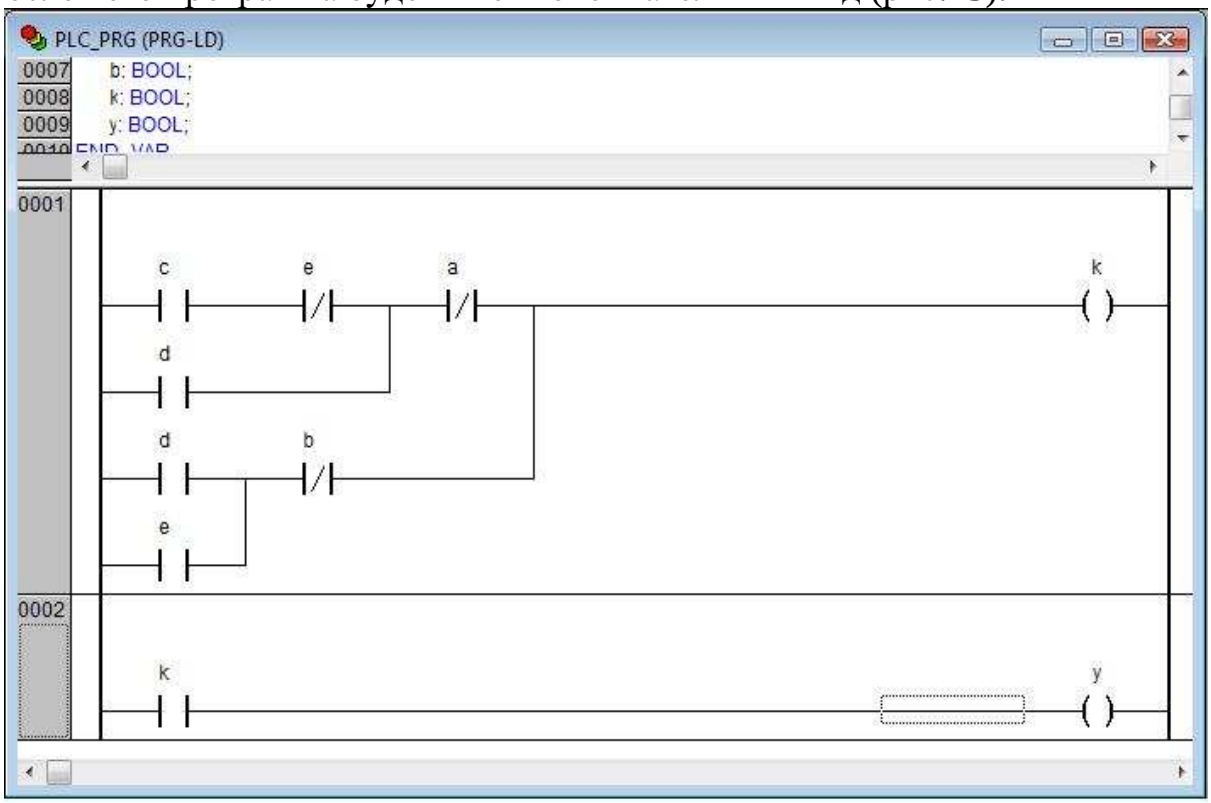


Рис. 23 Окончательный вид программы

Язык LD (Ladder Diagram), т.е. язык релейных диаграмм является достаточно популярным в силу своей наглядности и позволяет решать широкий круг задач комбинационной и событийно-управляемой логики.

Схемы на LD называют также многоступенчатыми, т.к. их цепи напоминают ступеньки вертикальной лестницы.

При написании программы на листе бумаге или сразу на мониторе ПК используются две вертикальные линии, изображающие шины питания, между которыми размещаются горизонтальные цепи, состоящие из воображаемых контактов и катушек реле, функциональных блоков FB. По меньшей мере хотя бы один контакт и одна катушка реле служат реальными входом и соответственно выходом ПЛК.

Так, например, реальная релейная схема управления электрическим двигателем (рис.1), содержащая кнопки пуска SB1 и остановки SB2, катушку КМ магнитного пускателя, имеющего блок-контакт КМ1 для самофиксации включенного состояния и контакты КМ2 для подачи питания на двигатель М, на языке LD в комплексе CoDeSys будет иметь вид, как показано на рис.24.

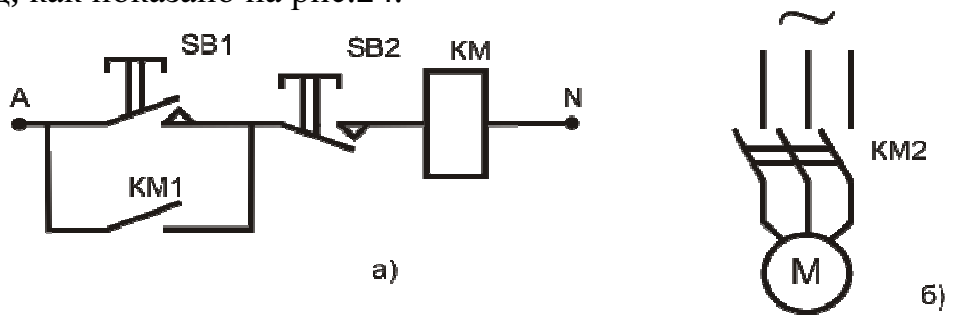


Рис.24. Контактная схема управления (а) с электрическим двигателем (б)

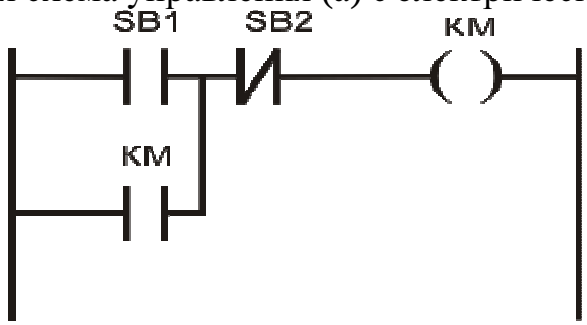


Рис.25. Многоступенчатая схема в LD управления электрическим двигателем

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

Каждая цепь заканчивается «катушкой» реле. **Последовательно** соединять катушки нельзя. **Параллельно** – можно.

Каждому элементу цепи (контакту, катушке, функциональному блоку) присваивается ***имя,*** которое на ***английском*** языке пишется ***над*** элементом.

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***.

Но цифру на первое место ставить нельзя: 1X, 2X – неправильно!

Нельзя применять в качестве имя буквы «S» и «R». С индексами, например R1, S2 или в сочетании с другими буквами (stop, reset, SK, RU и т.д.) – можно.

Регистр букв на 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. ее идентификатор) можно записать в развернутом виде, не используя буквы русского языка.

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

Начинающим даже простую схему лучше изобразить на бумаге, а затем переносить в LD.

Контакты и катушки реле

Помимо «обычных» реле —( )— можно применять аналог поляризованного реле, обозначаемого на схеме —(/)—.

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

Воображаемый аналог такого реле можно изготовить на базе переключающего геркона (рис.26)

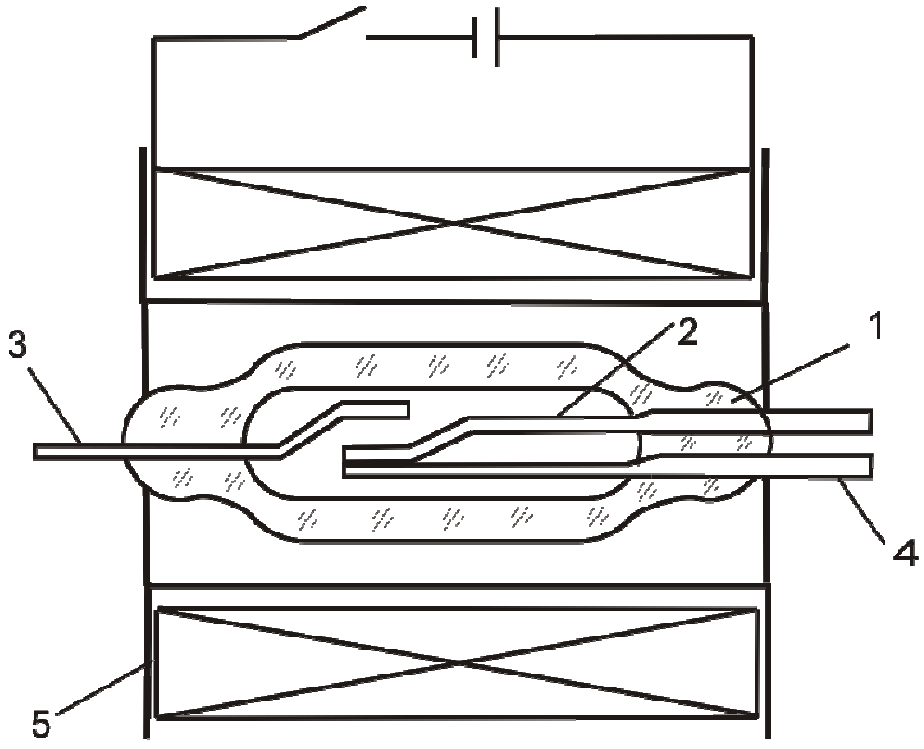


Рис.26. Устройство герконового реле

В нормальном состоянии подвижный контакт 2 замкнут с неподвижным контактом 4, выполненным из немагнитного металла. При пропускании тока через обмотку катушки 5 замыкаются контакты 2 и 3 и размыкаются контакты 2 и 4, т.е. реле работает в обычном режиме.

Если в катушку поместить постоянный магнит (рис.27), то при отсутствии тока в катушке 5 подвижный контакт 2 замкнет цепь с контактом 3 и разорвет цепь с контактом 4.

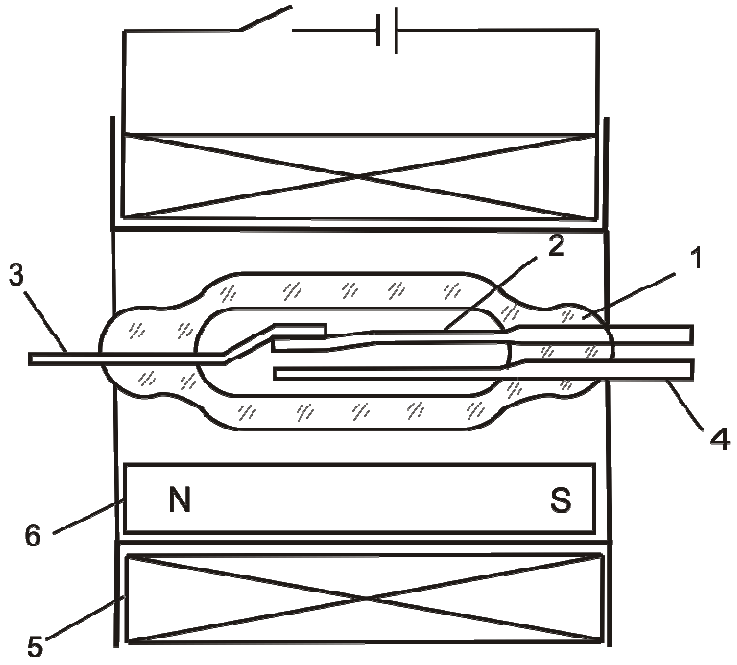


Рис.27 Устройство поляризованного герконового реле

В таком состоянии реле может находиться сколь угодно. При подаче тока ***размагничивания*** в катушку 5 магнитное поле постоянного тока (при правильном выборе полярности) скомпенсирует магнитное поле постоянного магнита 6 и контакт 2 вернется в исходное положение.

С использованием нормального реле —( )— и инверсивного реле —(/)— можно собрать генераторы (рис.28).

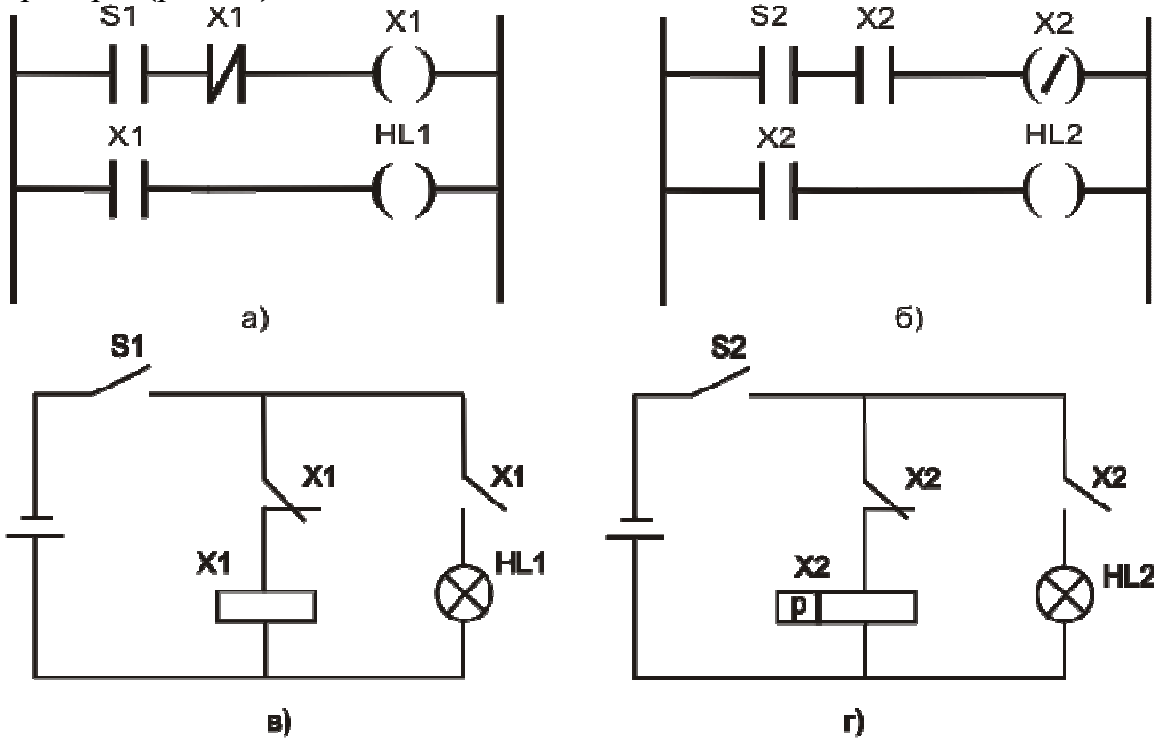


Рис.28. Простейшие генераторы импульсов (а и б) и их релейные аналоги (в и г): S1 и S2 – кнопки пуска, HL1 и HL2 – приемники импульсов

В наборе программных компонентов имеются также специальные обмотки SET и RESET, обозначенные в линейке кнопок как —(S)— и —(R)— соответственно. 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 (или FALSE) можно лишь с помощью обмотки R (рис.29).

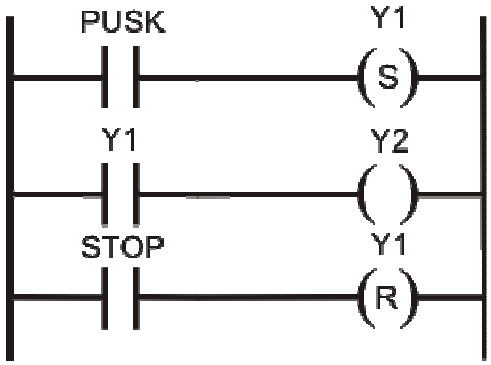


Рис.29. Схема фиксации включения катушки реле Y2 с помощью обмоток S и R

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 (рис.30).

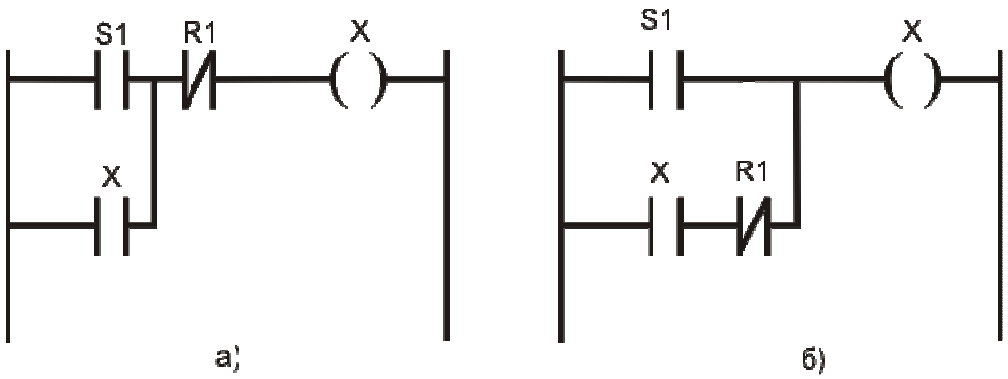


Рис.30. Схемы управления катушкой X с самофиксацией состояния

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.

«Бестолковый» оператор может нажать сразу две кнопки S1 и R1. Что происходит в этом случае?

В схеме а) катушка X **отключится** (если она была включена) или останется не включенной. В схеме б) катушка X **включится** (если она была отключена) или останется включенной.

В CoDeSys этим схемам созданы соответствующие аналоги: RS - триггер с доминантой выключения – для схемы рис.30-а и SR – триггер с доминантой включения – для схемы рис.30-б.

Триггеры

RS – и SR – триггеры отличаются лишь реакцией сигналов на оба входа: SET и RESET. Напоминаем, что для классического RS – триггера такое состояние входов является запрещенным, т.к. приводит к неоднозначности выходного сигнала.

Для исследования этих триггеров достаточно набрать лишь два фрагмента каких-то сложных многоступенчатых схем, в которых участвуют указанные функциональные блоки. (Сами схемы нас не интересуют!). На рис.31-а и 31-б изображены фрагменты схем с RS – и SR – триггерами.

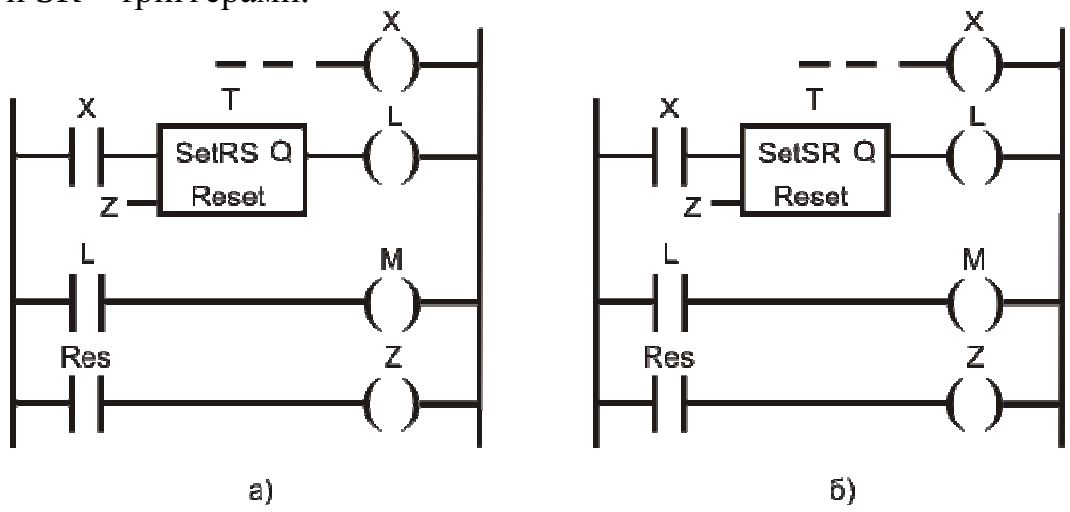


Рис.31. Фрагменты схем с RS – и SR – триггерами

При замыкании контакта X подается сигнал на вход SET каждого из триггеров (рис.31), что вызывает срабатывание реле L и через замыкающий контакт L этого реле приходит сигнал на обмотку реле M, включающего, например, электрическую машину. Последующие размыкания или замыкания не меняет состояние выходов этих триггеров, и катушки L и M остается включенными. Для отключения реле M необходимо кратковременно нажать кнопку Res.

Если одновременно замкнутся контакты X и Res, то в RS – триггере преобладающим будет сигнал на отключение L и соответственно M, а в SR – триггере – на включение этих реле.

Необходимо напомнить, что входу RESET этих триггеров необходимо присвоить идентификатор (имя) того реле, которое будет обеспечивать сброс. В нашем примере в третьей цепи каждого фрагмента (рис.31) стоит реле с именем «Z». Поэтому на входах RESET поставлен тот же символ – «Z».

Функциональные блоки R_TRIG и F_TRIG или детекторы импульсов. Первый из них генерирует одиночный импульс по переднему фронту, а другой – по заднему спаду входного сигнала. Исследование этих триггеров проведем после ознакомления с таймерами.

Таймеры

Три типа таймеров находят широкое применение:

TP таймер или генератор одиночного импульса заданной длительности;

TOF таймер с задержкой выключения;
TON таймер с задержкой включения.

У этих таймеров есть вход IN для логических сигналов, вход PT для установки требуемых временных параметров и логический выход Q.

Работу этих таймеров поясняют временные диаграммы (рис.32). Покажем лишь временные диаграммы входных и выходных сигналов (рис.32 а и б).

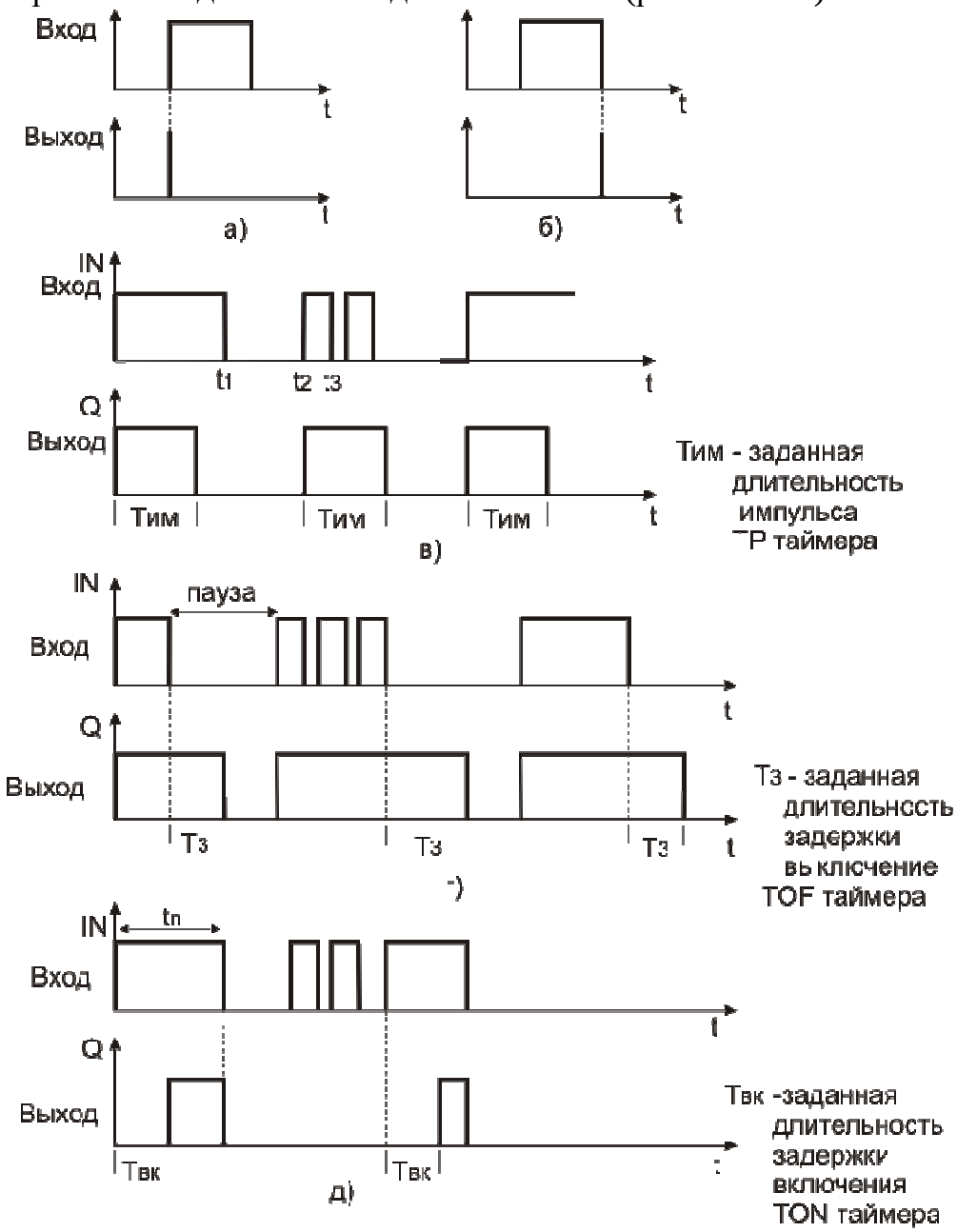


Рис.32. Временные диаграммы детекторов импульсов и таймеров:
а) R_TRIG; б) F_TRIG; в) TP; г) TOF; д) TON

Из этих диаграмм следует, что TP таймер запускается мгновенно передним фронтов входного сигнала и в течении времени $T_{и.м.}$ действия выходного сигнала не реагирует на новые импульсы, поступающие на вход IN (рис.32-а).

TOF таймер также срабатывает по фронту входа IN. Выход Q сбрасывается после спада входного сигнала с задержкой времени $T_{от}$, установленной по входу PT. **Пауза** между входными сигналами должна быть не меньше времени задержки (рис.32-б).

TON таймер срабатывает по переднему фронту входа IN, но сигнал на выходе Q появиться с задержкой T_B, установленной по входу PT.

Таймер не реагирует на импульсы продолжительностью менее значения T_B (рис. 32-в).

При включении любого таймера в цепь многоступенчатой схемы (рис.33) программа запрашивает имя этого функционального блока (вопросительные знаки над таймером) и временную уставку по входу PT (вопросительные знаки у входа PT).

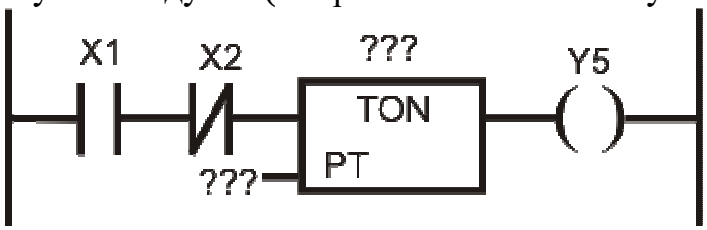


Рис.33. Фрагмент схемы с вставленным таймером

Щелкнув 1ЛКМ по верхним ???, присваиваем имя. Например N1. Щелкнув по ??? у входа PT, нажать, и не отпуская Shift нажать T, затем #, отпустить Shift, нажать требуемое значение задержки (например 15), единицу времени (например S – т.е. секунды) и Enter. Этот фрагмент будет выглядеть, как показано на рис.34.

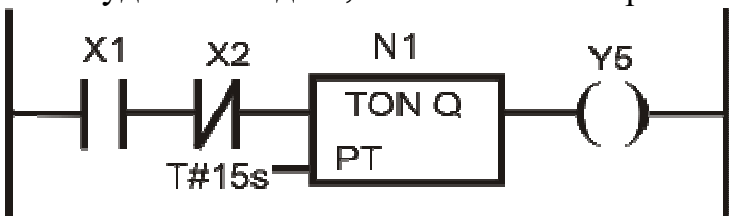


Рис.34. Фрагмент схемы по рис.33 после снятия ???

Временные уставки задают в миллисекундах (mS), секундах (S), минутах (m) или часах (h). Уставка дробной не может быть. Нельзя задать T#1,5m. Следует установить T#90s.

Наличие таймера не вызывает задержки в прогоне программы.

Примеры схем с таймерами и триггерами.

Собрать схему генератора с регулируемой длительностью импульса и паузы, как показано на рис.35.

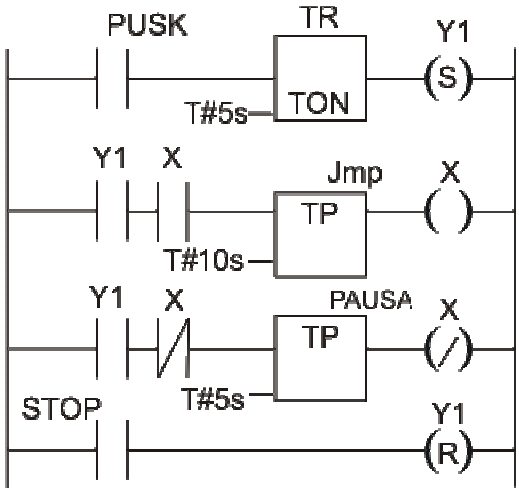


Рис.35. Генератор импульсов

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

Первая и четвертая цепи предназначены для пуска и остановки генератора.

Реле X является отдаленным аналогом поляризованного двухобмоточного электромагнитного реле. Поэтому эти «обмотки» в LD. Расположены в разных цепях, но имеют одинаковый идентификатор (в нашем примере - X).

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

Подобные схемы пуска рекомендуем применять и при испытании других схем.

Таймер с именем «Jmp» определяет длительность импульса, а таймер «Pausa» - длительность паузы. Естественно, студент по желанию может присвоить и другие имена и уставки по времени.

Собрать схему для исследования таймеров (рис.36).

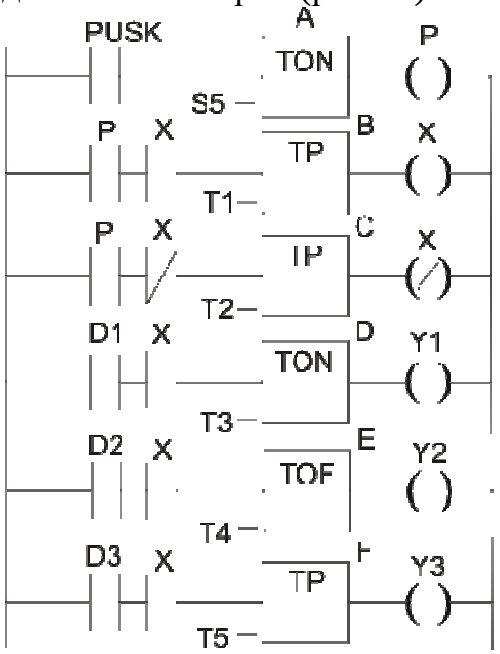


Рис.36. Схема исследования таймеров

Первая цепь служит для запуска уже известного нам генератора, занимающего 2ю и 3ю цепи.

Кнопки D1,D2 и D3 для поочередного подключения исследуемых таймеров TON, TOF и TP. Факт срабатывания таймеров можно наблюдать в режиме эмуляции за изменением состояний катушек реле Y1, Y2 и Y3. Синий цвет катушки указывает на включенное состояния. Временные параметры генератора (T1 - длительность импульса, T2 - паузы) и исследуемых таймеров (T3, T4, T5) нужно будет менять.

Естественно, временные параметры следует задавать в секундах, чтобы заметить реакцию катушек Y1, Y2 и Y3.

Например, установить T1=5с, T2=3с, T3=8с. Включить PUSK и кнопкой D1 таймер TON. Таймер не сработает, т.к. T1<T3 (см. рис. 32-в).

Потом установить T1=8с, T2=3с и T3=8с. Таймер также не сработает, т.к. T1=T3. Новая установка: T1=10с; T2=3с; T3=8с. Таймер TON сработает, т.к. T1>T3, и Y1 начнет «мигать» синим цветом, что можно еще раз показать на временной диаграмме (рис.37).

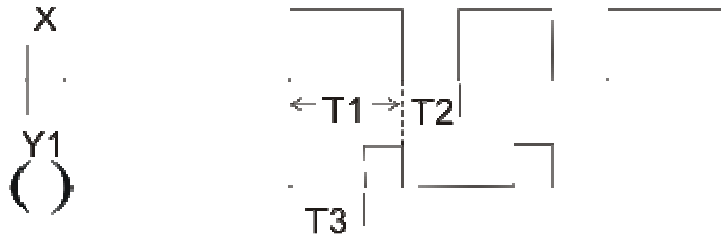


Рис.37. Временные диаграммы работы TON таймера D

Подобные опыты поставить для TOF и TP и убедиться в справедливости выводов по рис.32.

Учебная схема с применением всех таймеров.

Чисто в учебных целях рекомендуется собрать схему (рис.38) с применением всех таймеров и детекторов импульсов.

Чтобы на экране монитора в цепи поместились все элементы схемы необходимо по мере ее заполнения делать «прокрутку» влево. На рис.38 пришлось сделать «перенос» цепи.

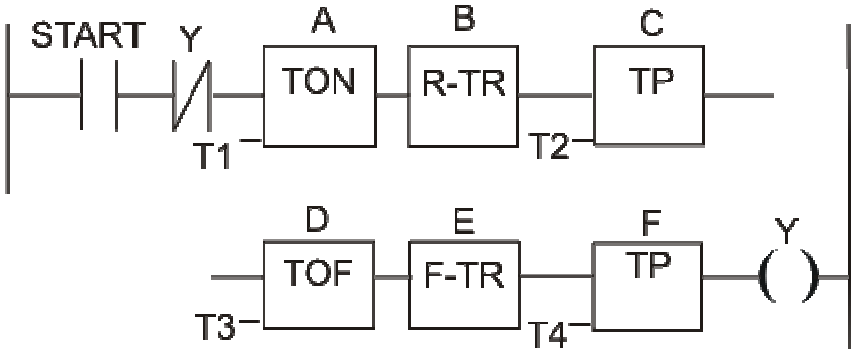


Рис.38. Учебная схема

Запустить схему в режиме эмуляции, построить временную диаграмму изменения состояний ее элементов, учитывая, что выходной сигнал одного FB служит входным для следующего за ним. Объяснить реакцию каждого элемента, а также влияние соотношения значений T2 и T3 на поведение системы в целом.

Напоминаем, что наличие сигнала на выходе какого-либо элемента подтверждается синим цветом канала, соединяющего этот выход с входом следующего блока.

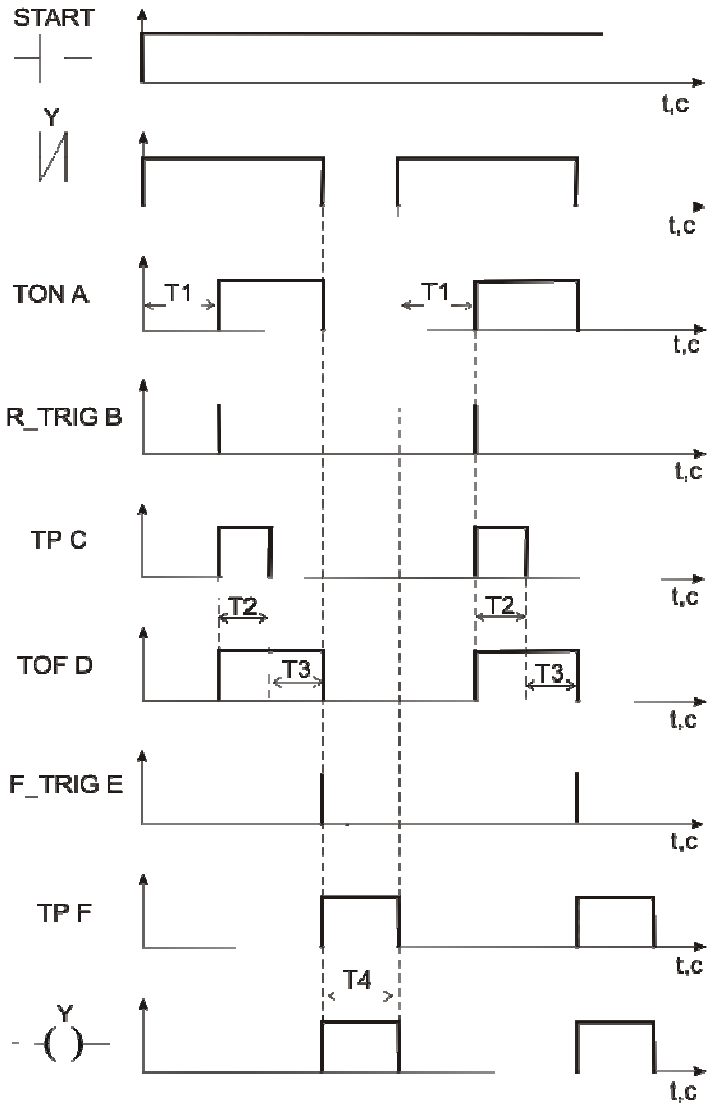


Рис.39. Временные диаграммы состояний элементов учебной схемы.

На выходе FB B и E ввиду малой длительности импульса заметить кратковременное изменение цвета затруднительно, особенно при масштабе, менее 100%.

Счетчики

CTU инкрементый, CTD декрементный и CTUD инкрементный/декреметный счетчики.

Собрать простую схему с CTU-счетчиком (рис.40).

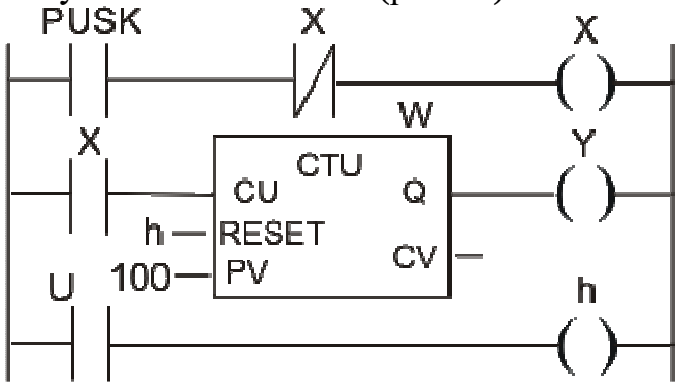


Рис.40. Схема с СТИ счетчиком

После переноса счетчика в цепь многоступенчатой схемы появятся ??? над блоком, на входе RESET и PV.

Знаки ??? над блоком, заменяем именем. Знаки ??? перед PV запрашивают значение уставки, т.е. требуемое количество импульсов, вызывающее срабатывание счетчика, при котором выход Q перейдет в TRUE (при условии, что на входе RESET был сигнал FALSE).

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

В схеме (рис.40) для счетчика присвоено имя W, для входа RESET имя реле h и принята уставка PV=100.

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

Первая цепь содержит кнопку PUSK и генератор импульсов на реле (см. рис.28-а).

По каждому фронту сигнала, поступающему на вход CU значение выхода CV ***возрастает*** на 1 и как только их сумма достигнет значения PV, счет останавливается. (На других языках программирования ПЛК с выхода CV можно снимать информацию о количестве поступивших импульсов для последующей обработки с помощью операторов и функций. Здесь этот вопрос не рассматривается).

Кстати, эта схема (рис.38) позволяет оценить время прогона программы. Достаточно секундомером замерить время от момента пуска генератора до момента остановки CTU (реле станет синим). Период импульсов генератора равен удвоенной длительности рабочего цикла ПЛК. Количество же циклов известно и равно уставке PV.

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

Важный момент! Счетчик CV загружается значением уставки, равным PV, только когда на входе LOAD есть сигнал TRUE.

Можно собрать схему с CTD счетчиком (рис.41).

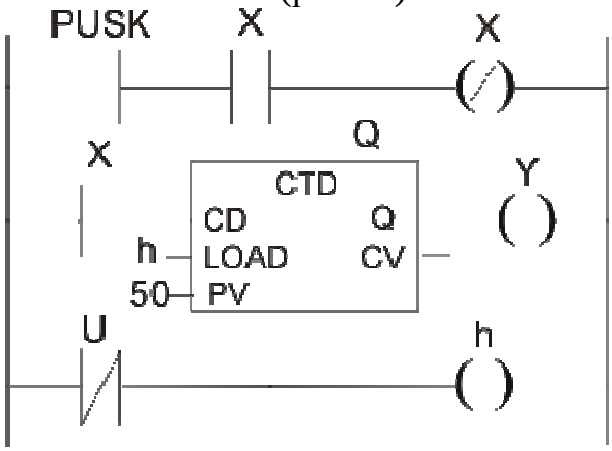


Рис.41. Схема с СТД счетчиком

Кнопка PUSK запускает генератор X (для разнообразия собран по схеме рис.28-б). Счетчику присвоили имя Q, уставка PV=50, имя реле, управляющего входом LOAD, h. По умолчанию принято, что реле, управляющие счетчиками CTU и CTD имеют на входах RESET и LOAD соответственно ***замыкающий*** контакт. В схемах (рис.40 и 41) это реле h. Но учитывая отличие счетчика CTD лучше в схеме (рис.41) установить ***размыкающую*** кнопку U. Тогда при включении схемы сразу сработает реле h и подает сигнал TRUE на вход LOAD, что обеспечит загрузку выхода CV начальным значением PV (в нашем примере 50). В этом положении имя переменной h на вход LOAD окрашено в синий цвет.

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

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

Новый отсчет начнется после повторного замыкания и размыкания кнопки U, например по схеме рис.42.

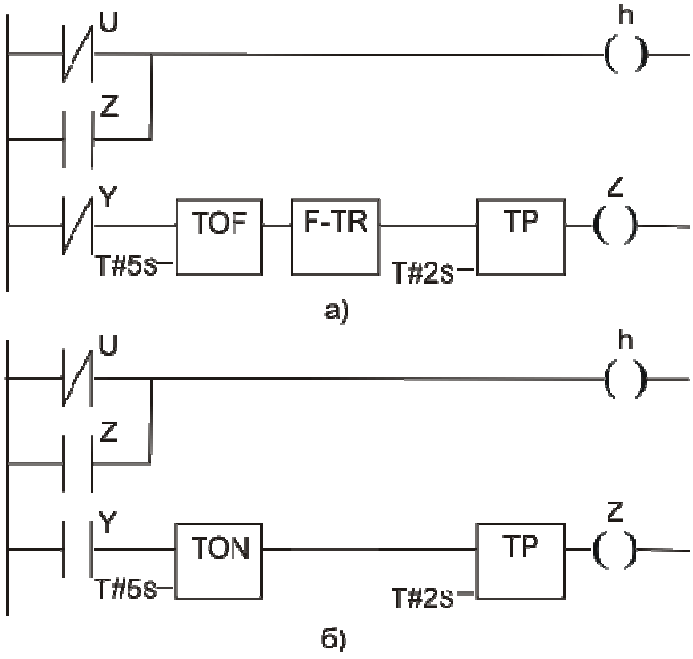


Рис.42. Варианты автоматического перезапуска счетчика CTD

Первые две цепи вариантов а) и б) на рис.42 не показаны, т.к. они полностью повторяют эти же цепи в схеме рис.41.

По таким же схемам можно осуществить перезапуск таймера CTU (рис.40).

У читателя уже достаточно знаний, чтобы проанализировать работу этих схем (рис.40, 41 и 42).

Для простых логических систем применение рассмотренных таймеров практически равноценно. Дело вкуса проектировщика в выборе того или иного счетчика.

CTUD-инкрементный/декрементный счетчик в данной работе не рассматривается. Отметим лишь, что у него есть накопительный вход CU (как в CTU), так и вычитающий CD (как в CTD).

4. ПРОЕКТИРОВАНИЕ СОБЫТИЙНО-УПРАВЛЯЕМОЙ СЛУ НА ЯЗЫКЕ LD В СРЕДЕ CoDeSys для ПЛК 100 RL ОВЕН.

Итак, спроектированная в п.2 комбинационная СЛУ на контактных элементах, содержащая три приемных элемента A, B и C и два исполнительных элемента X и Y описывается логическими функциями:

$$X = a\left(\bar{b} + \bar{c}\right),$$

$$Y = \bar{c}\left(a \cdot \bar{b} + \bar{a} \cdot b\right).$$

Теперь в учебных целях представим эту систему как последовательностную, т.е. описываемую с позиций событийно – управляемой логики. В первом событии должен сработать X, а затем по одному из вариантов приложения 3 произойдет срабатывание Y.

Допустим, выбран вариант 0. По условию этого варианта Y сработает через T секунд после срабатывания X.

Можно предложить такой вариант этой СЛУ в LD (рис.43).

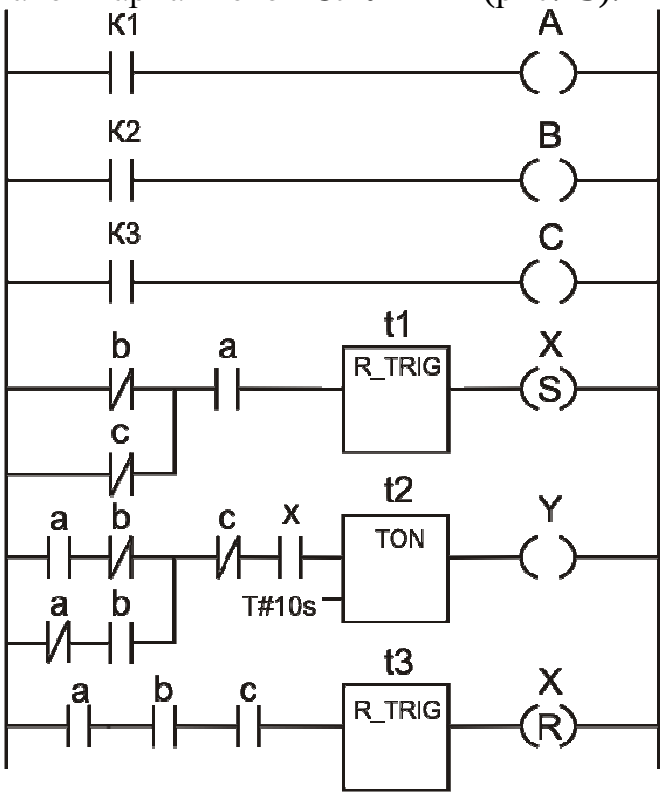


Рис.43. Возможный вариант спроектированной СЛУ.

Первые три цепи введены для удобства, чтобы в режиме эмуляции «нажимая» кнопки К1, К2 или К3, создавать желаемые комбинации состояний приёмных элементов A, B и C.

При срабатывании, например, только A и несрабатывании B и C, поступит сигнал на катушку -(S)- с именем исполнительного элемента X, и в нашем примере через 10с сработает Y.

Перебираем другие комбинации выходных сигналов и проверяем условия срабатывания и несрабатывания исполнительных элементов.

Седьмая цепь служит для снятия блокировки X. С этой целью на входе FB t3 можно поставить дополнительную кнопку сброса. Но можно взять какую-нибудь комбинацию контактов a,b и c, не приводящую к срабатыванию X и Y. В нашем примере взяли комбинацию восьмой строки из таблицы состояний.

Приложение 1

Две последние цифры номера зачетной книжки	Условия срабатывания исполнительных элементов	
	X	Y
00; 35; 70.	A, B, и C; A, C и D; B и C.	A и C; A и D; A и B; C; A,B и C.
01; 36; 71.	A, C и D; B, C и D; C и D.	A и D; A, B и C; A, C и D.
02; 37; 72.	B, C и D; A и D; A и C.	A,B и C; A и C; B,C и D.
03; 38; 73.	B и C; A, C и D; B, C и D.	B,C и D; A и D; C и D.
04; 39; 74.	A, B и C; A, C и D; A, B и D	B и D; A, B и D; B, C и D.
05; 40; 75.	C и D; B, C и D; A, C и D.	A и C; C и B,Д; B,C и D.
06; 41; 76.	A, C и D; A, B и D; B; C;	A и D; A и C; C, B и D; A,C и D
07; 42; 77.	A,B и C; B, C и D; C иДД;	D; A, B, C и D; A; A,B и C.
08; 43; 78.	A, B и C; B,C и D, A.	A; B; B и D; C и D; A,B и C
09; 44; 79.	A и B; A и C; B и C; C иD.	C и D; D; A и B; A и D.
10; 45; 80.	C и D; B,C и D.	A и C; A и D; B и C; B,C и D.

11; 46; 81.	C и D; B, C и D; A, B и D; A	A и B; C и D; A; B; C.
12; 47; 82.	C и D; B, C и D; B и C; A.	A и C; C и B; A.
13; 48; 83.	A, C и D; A, B и D; B; C.	B, C и D; A и D; C и D; A,C и D.
14; 49; 84.	A, B и C; A и C; B, C и D.	C и D; B, C и D; A, C и D.
15; 50; 85.	A, C иD; B, C и D; C и D; D.	A и D; A и C; A, B и C; A, C и D.
16; 51; 86.	A, C и D; A и C; B; D.	A и C; A и D; B и C.
17; 52; 87.	C и D; B, C и D; C и D; B.	B, C и D; A и D; A и C.
18; 53; 88.	A, B и D; A, B и C; C и D.	A и C; C и D; A; B.
19; 54; 89.	A, B и C; C и D; B и D.	A и C; B и D; A,B,C и D.
20; 55; 90.	A, B и C; A и C; C и D; A.	A и D; B, C, и D; A, B иD; A, B и C.
21; 56; 91.	A и C; B и D; C и D; D.	A, B и D; C иD; B и C.
22; 57; 92.	A, B и D; A; B; C; D.	C и D; B и C; A,B и D.

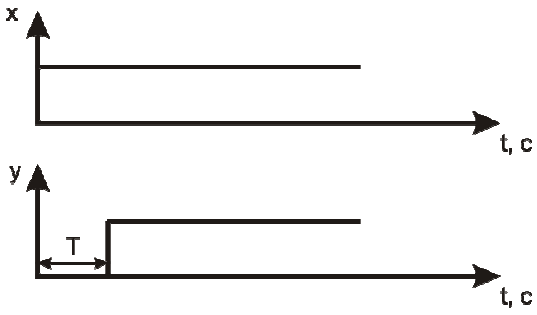
23; 58; 93.	A, B и C; B, C и D; C и D; C	A, B и C; C; B; B и D.
24; 59; 94.	A, B и D; A, C и D; D; B.	A и D; B и C; A, B и D; A.
25; 60; 95.	B, C и D; B и D; A и D; A.	A, C и D; A и B; B,C и D; B.
26; 61; 96.	A, B и C; A, C и D; B, C и D; C и D.	A и B; A и C; A и B; A и D, C и D.
27; 62; 97.	A, B и D; B, C и D; A и D; B и D.	A, B, C и D; B,C и D; C и D; D.
28; 63; 98.	A, B и C; B,C и D; C и D; C.	A и B; B и C; C и D; A; B.
29; 64; 99.	A, B и D; B,C и D; C и D; A.	A; A и B; B,C и D.
30; 65.	A,C и D; A и D; C и D; C.	A и B; A и C; A, B и D; A и D.
31; 66.	A, B, C и D; A и C; C и D; B	B и C; C и D; A; C.
32; 67.	A, C и D; A и B; B и D; A.	A и B; B и C; C и D; C; D.
33; 68.	A, B, C и D; A и B; B и C; C.	A, B и C; B, C и D; C и D; D; B и C
34; 69.	A, B и D; B, C и D; C и D; A и B.	A и B; B и C; A и D; A, C и D.

ПРИЛОЖЕНИЕ 2
К пункту 1.3. задания

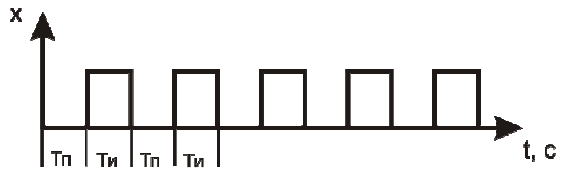
Если последняя цифра зачетной книжки чётная-то систему выполнить на блоках И-НЕ, если нечётная, то на блоках ИЛИ-НЕ.

ПРИЛОЖЕНИЕ 3

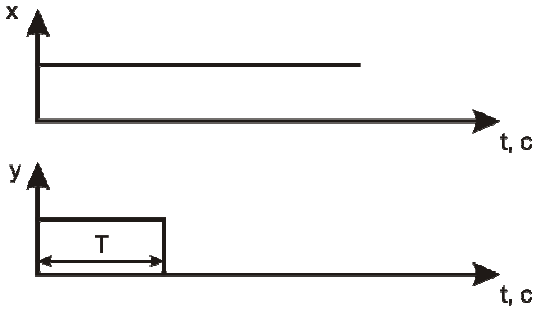
ВРЕМЕННЫЕ ДИАГРАММЫ РАБОТЫ ИСПОЛНИТЕЛЬНЫХ МЕХАНИЗМОВ X И Y



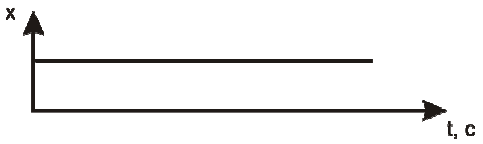
ВАРИАНТ 0 и 3

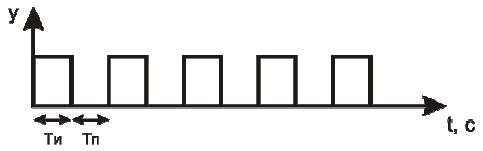


ВАРИАНТ 5

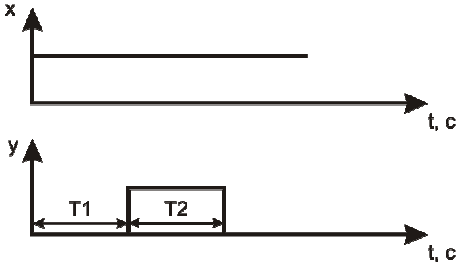


ВАРИАНТ 1

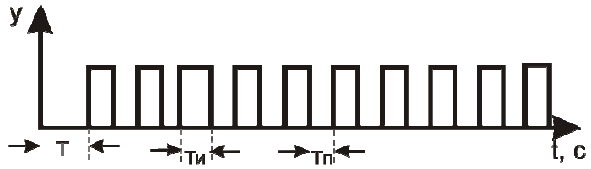




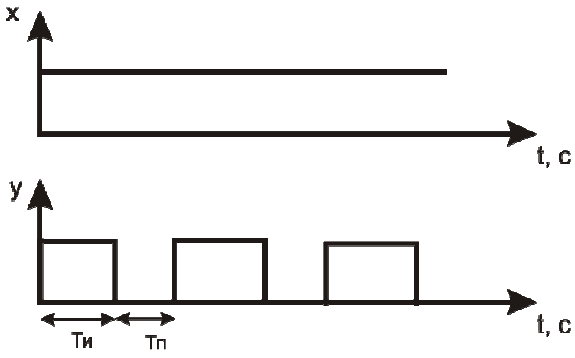
ВАРИАНТ 6



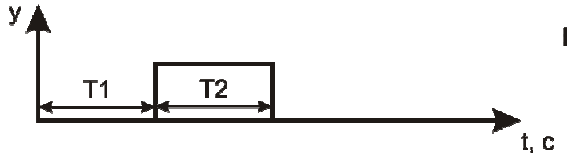
ВАРИАНТ 2и 8



ВАРИАНТ 7



ВАРИАНТ 4



ВАРИАНТ 9

Пояснения к Приложению 3

А) Номер варианта определяется последней цифрой в номере зачетной книжки.

Б) Временные параметры (длительность импульса, паузы, задержки включения – отключения взять произвольно. *Рекомендуем значения от 2х до 10 секунд, что облегчит наблюдение работы системы в режиме эмуляции.*

В) Временные диаграммы читаются так:

ВАРИАНТ 0 И 3

Y срабатывает после срабатывания X через T секунд.

ВАРИАНТ 1

Y сработает после сработает после срабатывания X и через T секунд отключится.

ВАРИАНТ 2 И 8

После срабатывания X через T_1 секунд сработает Y, который отключится через T_2 секунд.

ВАРИАНТ 4

После срабатывания X сразу сработает Y и переходит в пульсирующий режим с длительностью импульса $T_И$ секунд и паузы $T_П$ секунд.

ВАРИАНТ 5

После срабатывания X Y переходит через $T_П$ секунд в пульсирующий режим с длительностью импульса $T_И$ секунд.

ВАРИАНТ 6

После срабатывания X исполнительный механизм Y сразу переходит в режим генерации импульсов и после 10 импульсов остановится.

ВАРИАНТ 7

После срабатывания X исполнительный механизм Y через T секунд переходит в режим генерации с длительностью импульса $T_И$ и паузы $T_П$. При этом $T>T_И$ и после 20 импульсов остановится.

ВАРИАНТ 9

После срабатывания X отключится через T_1 секунд, после чего включится на T_2 секунд Y.

Г) Для комбинационных СЛУ X и Y между собой во времени не связаны, и каждый исполнительный элемент срабатывает независимо от другого в зависимости от состояния приемных элементов A,B и C. При какой-то комбинации сработает только X, при другой – только Y, а к каких-то случаях X и Y одновременно.

Поэтому при переходе к событийно-управляемым СЛУ чисто в учебных целях будем предполагать, что в проектируемую систему ввели дополнительное условие: сначала сработал X при заданной в приложении 1 комбинации состояний A,B, C и D(это первое состояние), затем сработал Y при (возможно) другой комбинации этих же приемных элементов. Поэтому будем рассматривать X как дополнительный входной элемент для Y.

С этой целью событие срабатывания X необходимо зафиксировать каким-то элементом памяти. Например, RS- или SR триггерами, S и R катушками или другими средствами. После этого состояние X не будет зависеть от изменения A, B, C и D пока не снята фиксация.

Если для X и Y есть одинаковые комбинации A,B,C и D , приводящие к их срабатыванию, то после срабатывания X и фиксации его состояния должно произойти срабатывание Y по заданной в приложении 3 временной диаграмме.

Если таких комбинаций нет, то после фиксации срабатывания X следует поочередно в режиме эмуляции перебрать все комбинации состояний приемных элементов и убедиться в правильности функционирования многоступенчатой схемы в LD.

Литература:

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного программирования. М.: Солон-Пресс. – 2004. – 253с.
2. Парр Э.Программируемые контроллеры. М.: Бином. – 2007. – 516с.
3. Карпов Ю.Г. Теория автоматов. Учебник для вузов. М.: ПИТЕР. – 2002. – 206с.
4. Калабеков В.А. Цифровые устройства и микропроцессорные системы./ В.А.Калабеков - М.: Горячая линия – Телеком, 2000.-С. 336с.
5. Минаев И.Г., Самойленко В.В. Программируемые логические контроллеры. Практическое руководство для начинающего инженера. Ставрополь, - «АГРУС», 2009.-100с.
6. Минаев И.Г., Шарапов В.М., Самойленко В.В., Ушкур Д.Г.. Программируемые логические контроллеры в автоматизированных системах управления. Ставрополь, - «АГРУС»,2010 – 128с.