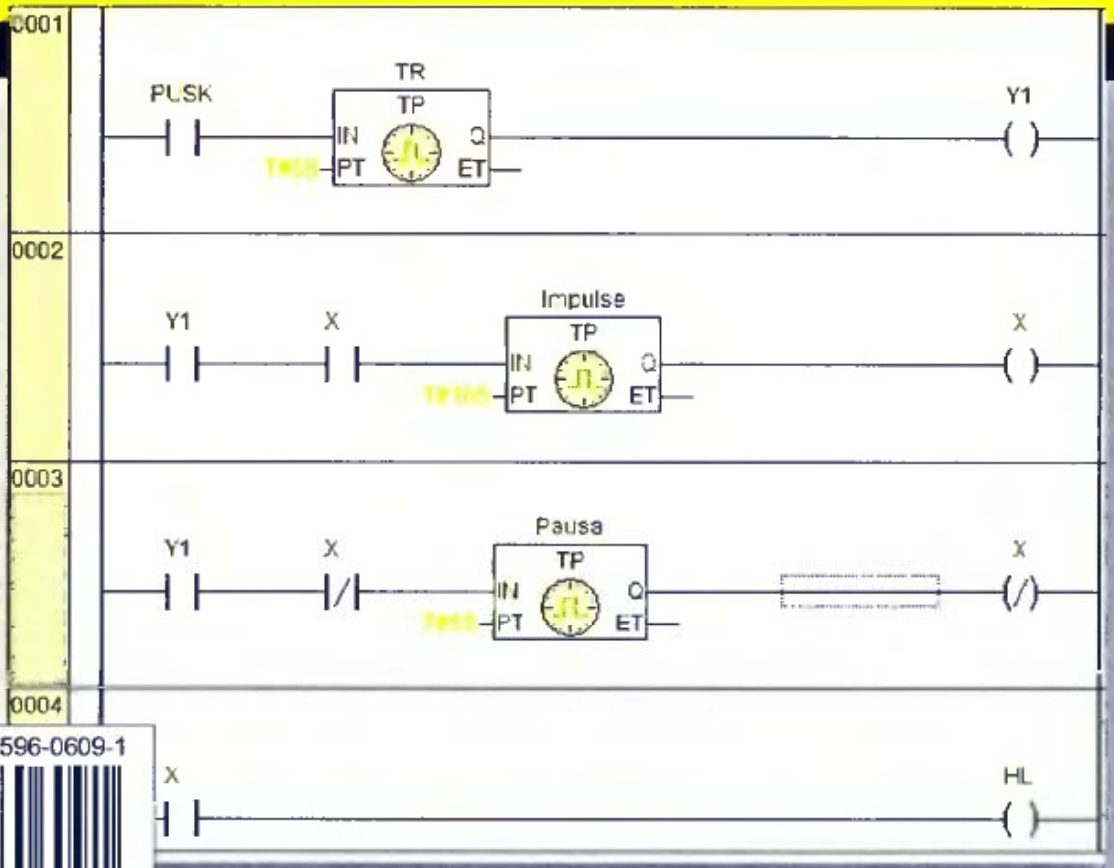
Программируемые Логические Контроллеры

И. Г. Минаев,
В. В. Самойленко

Практическое руководство для начинающего инженера

Изложены общие сведения по применению программируемых логических контроллеров (ПЛК) в системах управления технологическими процессами, описываемых с позиций событийно-управляемой логики. Все примеры рассмотрены в комплексе CoDeSys на языке LD.

Для инженеров, начинающих осваивать программируемые контроллеры, а также студентов вузов и аспирантов, изучающих современные методы автоматизации.

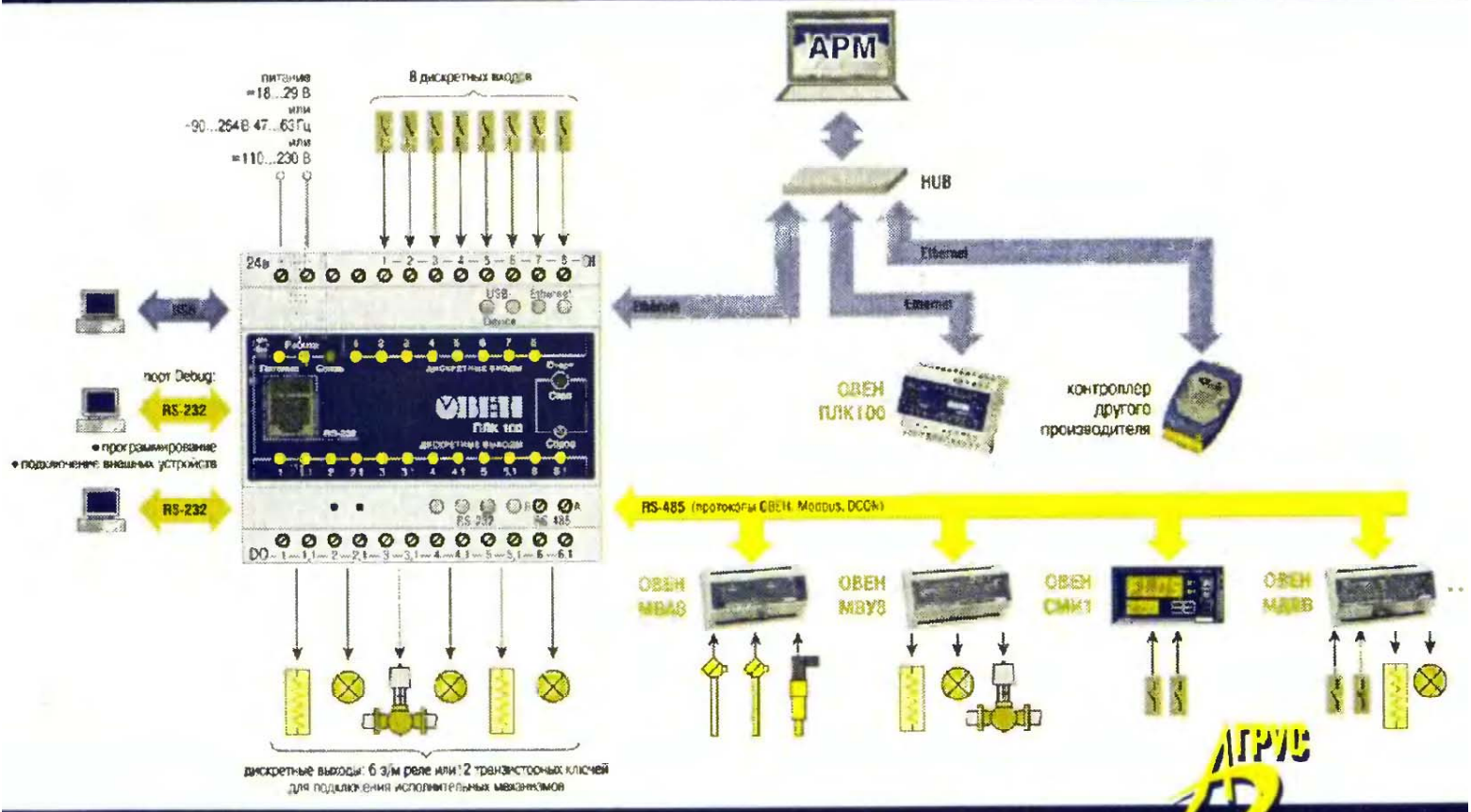


ISBN 978-5-9596-0609-1

И. Г. Минаев, В. В. Самойленко

Программируемые Логические Контроллеры

Практическое руководство для начинающего инженера



ДРУК

И. Г. Минаев, В. В. Самойленко

Программируемые Логические Контроллеры

Практическое руководство для начинающего инженера

Ставрополь
«АГРУС»
2009

УДК 681.5.
ББК 32.965
М57

Рецензент

доктор технических наук, профессор, заведующий кафедрой «Электрические машины и электропривод» КубГАУ
С. В. Оськин

Минаев, И. Г.

М57 Программируемые логические контроллеры : практическое руководство для начинающего инженера / И. Г. Минаев, В. В. Самойленко. – Ставрополь : АГРУС, 2009. – 100 с.

ISBN 978-5-9596-0609-1

Изложены общие сведения по применению программируемых логических контроллеров (ПЛК) в системах управления технологическими процессами, описываемых с позиций событийно-управляемой логики. Все примеры рассмотрены в комплексе CoDeSys на языке LD.

Для инженеров, начинающих осваивать программируемые контроллеры, а также студентов вузов и аспирантов, изучающих современные методы автоматизации.

УДК 681.5.
ББК 32.965

ISBN 978-5-9596-0609-1

ОГЛАВЛЕНИЕ

ВМЕСТО ВВЕДЕНИЯ, ИЛИ С ЧЕГО ВСЕ НАЧИНАЛОСЬ

А все начиналось с релейно-контактных систем логического управления (СЛУ) технологическими процессами и оборудованием как в производственной сфере, так и в быту.

Такие СЛУ, выполненные на релейно-контактных элементах или на потеснивших их микросхемах, имели фиксированную логику работы, и в случае необходимости изменения алгоритма управления требовалась существенная переделка всей монтажной схемы.

Бурное развитие электроники, особенно в сфере микропроцессорной техники, привело к созданию программируемых логических контроллеров (ПЛК), которые кардинально изменили сам подход к созданию конечных автоматов, вытеснив полностью контактные СЛУ. Это не означает, что электромагнитные элементы изжили себя. Всему своё место. И обычные реле широко применяются в тех же ПЛК в качестве выходных элементов, выполняя по существу функцию усилителя или устройства для гальванической развязки микропроцессорной части ПЛК от исполнительных механизмов.

Если промышленно развитые страны Старого и Нового Света давно и вовсю используют ПЛК и вышли на стабильный уровень их применения, то в России лишь в последние годы наблюдается резко возросший спрос на эту технику. В то же время почти полное отсутствие литературы по ПЛК создает значительные трудности в обучении и переподготовке киповского персонала.

В подобной ситуации в начале 70-х годов прошлого столетия оказались американские инженеры, обслуживающие релейные автоматы сборочных конвейеров и столкнувшиеся с нарастающим потоком микропроцессорной техники. На помощь им пришёл язык LD (Ladder Diagram), т. е. язык релейно-контактных схем, который не требовал каких-либо специальных знаний в области программирования, особенно с применением языков высокого уровня. Освоив же этот простой графический язык, уже легче оказалось перейти и к другим приемам программирования ПЛК.

Данная книга адресована не только инженерам, начинающим осваивать программируемые контроллеры, но и «новичкам», же-

лающим войти в мир ПЛК. Авторы отчетливо понимают при этом, что «нельзя объять необъятное» и, чтобы не отпугнуть читателя неоправданно большим объемом книги, излагают материал предельно лаконично. Возможно некоторые «новички» забыли или даже не знали теории релейно-контактных схем (РКС). Поэтому в первой главе мы позволим себе сделать такое напоминание, полагая, что читатель не увидит в этом намека на его некомпетентность.

Просто многие из них могут пропустить эту главу и начинать сразу со следующей. Но эта глава как бы ставит читателя в положение американского инженера прошлого века, пожелавшего освоить новейшую технику.

Поэтому мы также остановим свой выбор на языке LD, полагая при этом, что дальнейшее расширение арсенала методов проектирования СЛУ на базе ПЛК будет менее трудоемким. Как говорится, лиха беда начало.

По этим же соображениям из большого разнообразия известных комплексов программирования МЭК 61131-3 будем использовать только CoDeSys как наиболее популярный у пользователей ПЛК.

Безусловно, у читателя «под рукой» должен быть персональный компьютер, т. к. подавляющую часть примеров, приведенных в книге, можно выполнить, не имея даже самого ПЛК. Режим ***эмуляции*** позволяет производить отладку проекта без аппаратных средств.

1. ПРИМЕНЕНИЕ АЛГЕБРЫ БУЛЯ ДЛЯ ОПИСАНИЯ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ И СИСТЕМ

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

В комбинационных СЛУ (их же называют однотактными) выходные сигналы формируются только при определённых комбинациях входных логических сигналов, принимающих значения **0** или **1**. В последовательностных СЛУ (их же называют многотактными) выходные сигналы зависят не только от комбинации входных, но и последовательности их поступления во времени, что обеспечивается наличием элементов памяти.

Словесное изложение работы даже простых СЛУ выглядит громоздко и затрудняет проведение анализа. Математическим аппаратом для описания СЛУ служит двузначная (бинарная) алгебра Буля, 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 иллюстрируются ниже на примере.

Логическая функция выражает зависимость выходных переменных от входных и также принимает, в зависимости от значения последних и связывающих их логических действий, 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 для n переменных будет равно

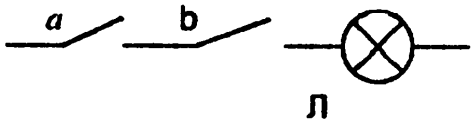
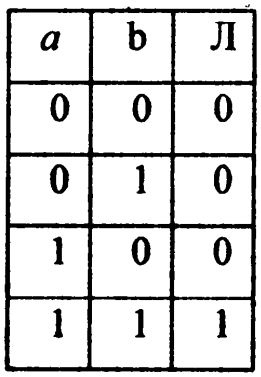
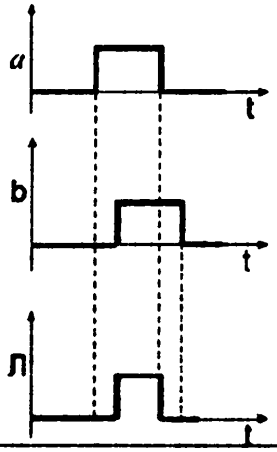
$$N = 2^n.$$

Все действия над переменными в бинарной алгебре выполняются с помощью следующих основных операций, которые наглядно иллюстрируются соответствующими РКС.

С целью упрощения последующих алгебраических действий примем следующие допущения: катушки электромагнитных реле, которые следовало бы по действующему стандарту обозначать *К1, К2* и т. д. и соответственно их контакты *К1.1, К1.2, К2.1, К2.2*, будем по мере необходимости именовать заглавными буквами *A, B, C* и т. д., а их замыкающие и размыкающие контакты прописными буквами *a, b, c* и $\bar{a}, \bar{b}, \bar{c}$ соответственно.

1.1. 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.

РКС	Таблица состояний	Временные диаграммы
		

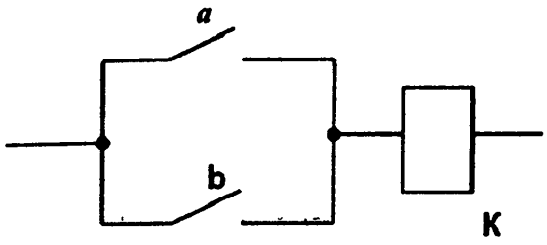
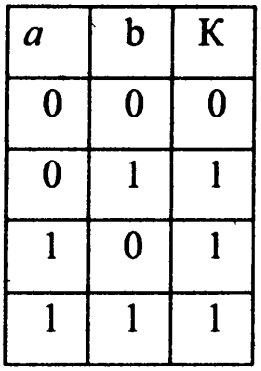
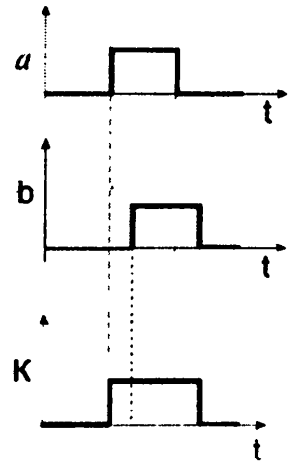
a	b	Л
0	0	0
0	1	0
1	0	0
1	1	1

Применяются и другие обозначения операции логического умножения:

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

1.2. Логическое сложение

Результат логического сложения (дизъюнкции, операции «***ИЛИ***») $K = a + b$, легко установить из анализа схемы с параллельным соединением контактов.

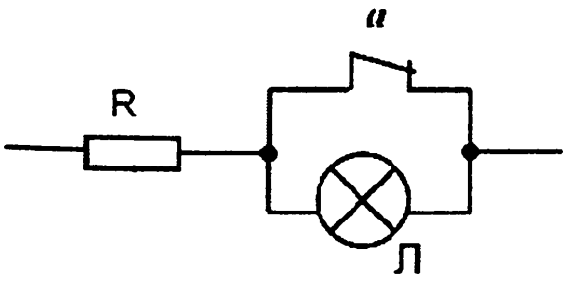
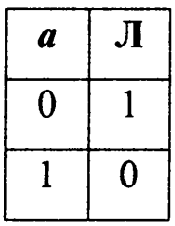
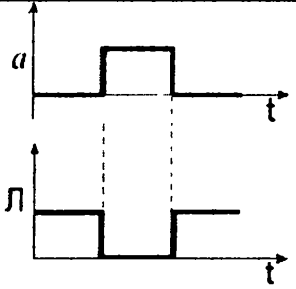
РКС	Таблица состояний	Временные диаграммы
		

a	b	K
0	0	0
0	1	1
1	0	1
1	1	1

Очевидно, что катушка реле K получит питание, если замкнуть контакты или a, или b, или оба вместе. Вместо знака «+» иногда употребляют «∨»: $a + b = a \vee b$.

1.3. Логическое отрицание

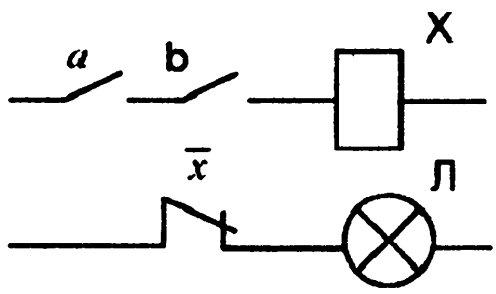
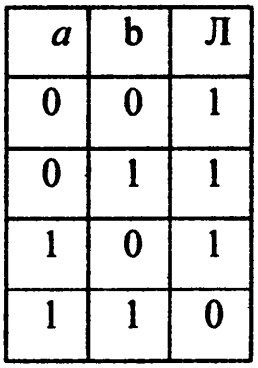
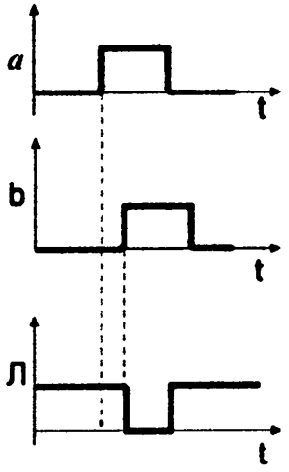
Логическое отрицание (инверсия, операция «***НЕ***») $Л = \bar{a}$, означающее, что значение логической функции $Л$ противоположно или неравносильно значению переменной a. В нашем примере лампа горит ($Л = 1$), если контакт a ***не*** замкнут ($a = 0$), и лампа гаснет ($Л = 0$), если контакт ***не*** разомкнут ($a = 1$).

РКС	Таблица состояний	Временные диаграммы
		

a	Л
0	1
1	0

1.4. Инверсия конъюнкции

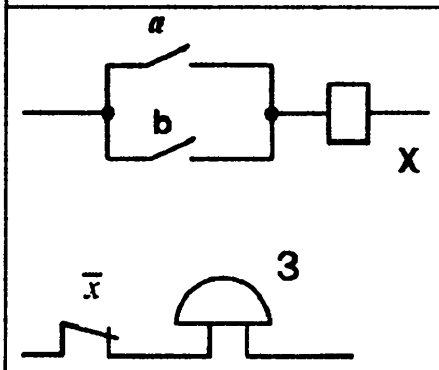
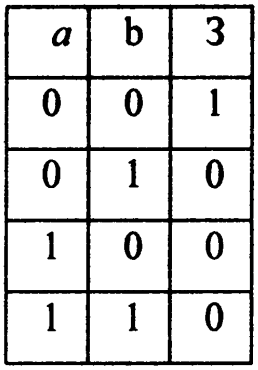
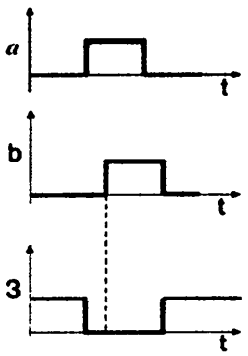
Функция «*И – НЕ*» $Л = \overline{a \cdot b}$.

РКС	Таблица состояний	Временные диаграммы
		

a	b	Л
0	0	1
0	1	1
1	0	1
1	1	0

1.5. Инверсия дизъюнкции

Функция «*ИЛИ – НЕ*» $З = \overline{a + b}$.

РКС	Таблица состояний	Временные диаграммы
		

a	b	З
0	0	1
0	1	0
1	0	0
1	1	0

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**.

1.6. Аксиомы и законы булевой алгебры

1.6.1. 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

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

Как и в обычной алгебре здесь действуют следующие 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. Действительно, умножим $(a + c)$ на $(b + c)$:

$$(a + c)(b + c) = ab + bc + ac + cc.$$

Так как $cc = c$, то

$$ab + bc + ac + c = ab + c(b + a + 1).$$

Так как $b + a + 1 = 1$, то

$$ab + c \cdot 1 = ab + c,$$

что и требовалось доказать.

Но есть и специфические аксиомы, законы и теоремы, которые легко доказать.

1.6.2. Аксиомы

$0 \cdot 0 = 0;\ 1 + 1 = 1;\ 0 + 0 = 0;\ 1 \cdot 1 = 1;\ 1 \cdot 0 = 0;\ 1 + 0 = 1;\ \bar{0} = 1;\ \bar{1} = 0.$

1.6.3. Законы булевой алгебры

Законы нулевого множества: $0\,a = 0;\ 0\,a\,b...k = 0;\ 0 + a = a.$

Законы универсального множества: $1\,a = a;\ 1 + a = 1;\ 1 + a + b + ... + k = 1.$

Законы повторения: $a\,a\,a....a = a;\ a + a + a + ... + a = a.$

Законы дополнительности: $a \cdot \bar{a} = 0;\ a + \bar{a} = 1.$

Законы инверсии: $\overline{a \cdot b} = \bar{a} + \bar{b};\ \overline{a + b} = \bar{a} \cdot \bar{b}.$

Законы поглощения: $a\,(a + b) = a;\ a + a\,b = a;\ a(\bar{a} + b) = a \cdot b;$

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

Закон двойного отрицания: $\bar{\bar{a}} = a;\ \bar{\bar{0}} = 0;\ \bar{\bar{1}} = 1.$

Законы склеивания: $a \cdot b + a \cdot \bar{b} = a;\ (a + \bar{b}) \cdot (a + b) = a.$

1.7. Применение законов и аксиом при анализе и синтезе СЛУ

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:

$$a \cdot \bar{a} = 0,\ a + \bar{a} = 1.$$

Действительно, последовательно соединенные замыкающий (a) и размыкающий ($\bar{a}$) контакты одного и того же реле (A) 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:

$$\overline{ab} = \bar{a} + \bar{b};\quad \overline{a + b} = \bar{a} \cdot \bar{b}.$$

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 1.1 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 1.1

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 $\overline{ab}$ и $\bar{a}+\bar{b}$ 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:

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

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

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

Как и в обычной алгебре здесь применяются различные приемы для доказательства равносильных логических функций. Например, докажем один из законов поглощения:

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

Учитывая, что $b + \bar{b} = 1$ и $a \cdot 1 = a$, «усложним» левую часть уравнения:

$$a(b + \bar{b}) + \bar{a}b = ab + a\bar{b} + \bar{a}b = \ldots$$

Далее, учитывая, что $ab + ab = ab$, ещё более усложним это выражение:

$$\ldots = ab + ab + a\bar{b} + \bar{a}b = \ldots$$

и как в обычной алгебре сделаем очевидные преобразования, вынеся общие сомножители в первом и третьем, а также во втором и четвертом слагаемых:

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

Так как $b + \bar{b} = 1$, $a + \bar{a} = 1$, это выражение принимает вид

$$... = a \cdot 1 + b \cdot 1 = a + b,$$

что и требовалось доказать.

1.8. Пример проектирования комбинационной СЛУ

Необходимо разработать СЛУ неким воображаемым технологическим процессом (рис. 1.1). В результате обследования самого процесса и беседы с технологами выявили требуемое количество и характеристики приемных и исполнительных элементов и сформулировали алгоритмы работы проектируемой СЛУ.

Итак, приемные элементы: кнопка пуска *SB*, реле уровня *SL* и реле температуры *SK* и исполнительные элементы: асинхронный двигатель *M* и маломощная сигнальная лампа низкого напряжения *HL*. Приемные элементы, вырабатывающие дискретные сигналы, подключены к катушкам электромагнитных реле *A*, *B* и *C*, установленных на входах СЛУ и имеющих достаточное количество замыкающих и размыкающих контактов для синтеза самого логического блока.

Поэтому для записи условий срабатывания исполнительных элементов будем для краткости использовать обозначения катушек реле, а не самих датчиков.

Условия срабатывания для *M*:

– *M* срабатывает, если срабатывают *A*, *B*, но не срабатывает *C*;
– *M* срабатывает, если срабатывают *A*, *C*, но не срабатывает *B*;
– *M* срабатывает, если срабатывают *A*, но не срабатывает *B* и *C*.

Условия срабатывания для *HL*:

1. *HL* срабатывает, если срабатывает *A*, но не срабатывают *B* и *C*.
2. *HL* срабатывает, если срабатывает *B*, но не срабатывают *A* и *C*.

Составляем основной элемент синтеза – таблицу состояний. В таблице 1.2 рассматриваем все возможные комбинации состояний приемных элементов. Так как приемных элементов три – то возможное число комбинаций 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.

В общем случае количество комбинаций N_{CP} срабатывания любого исполнительного элемента лежит в пределах

$$1 \le N_{CP} \le 2^n - 1,$$

т. к. если $N_{CP} = 0$, то исполнительный элемент никогда не включается. Если же $N_{CP} = 2^n$ – то никогда не отключится.

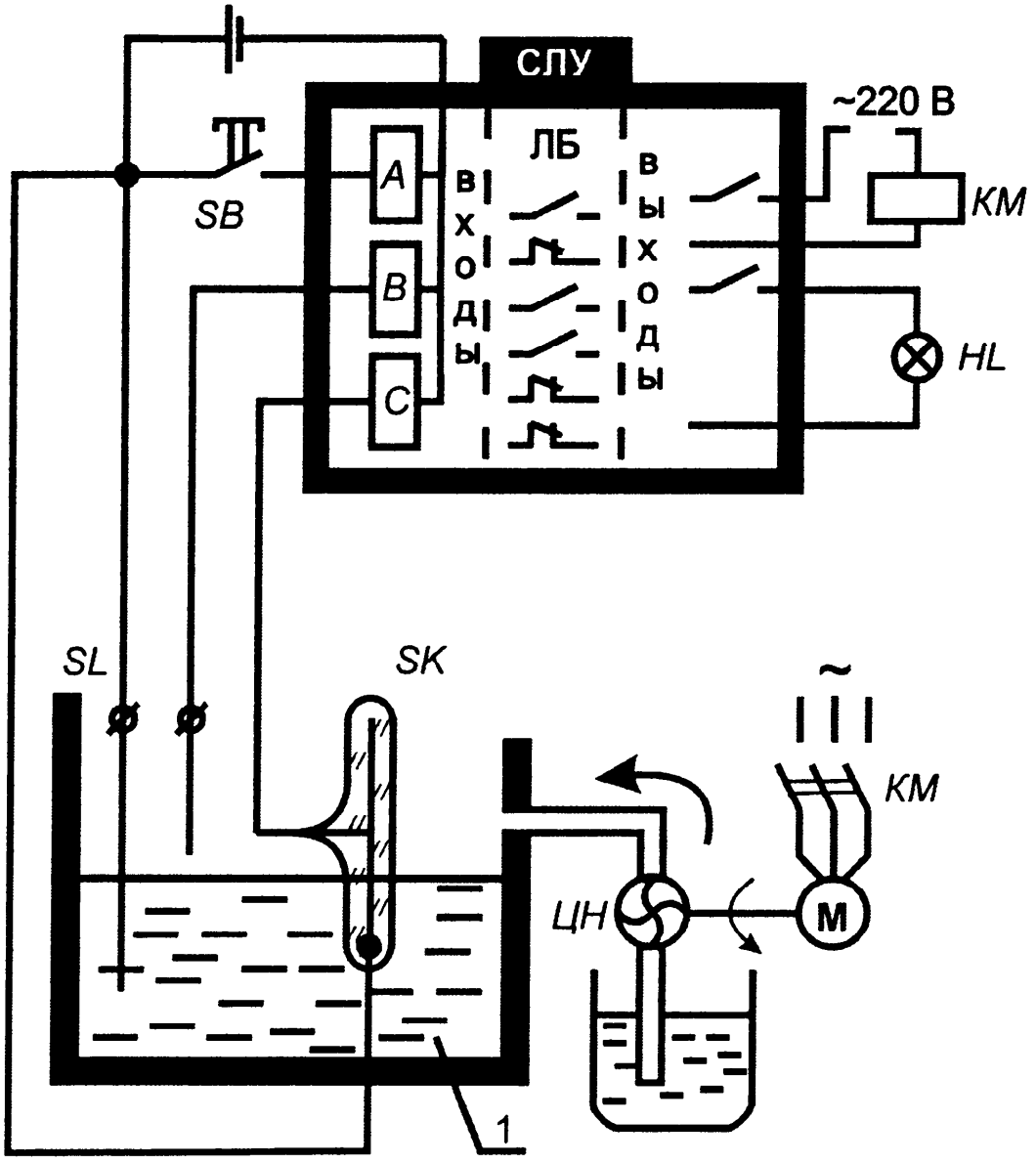


Рис. 1.1. Воображаемый технологический процесс:
1 – объект управления; *ЦН* – центробежный насос; *SK* – реле температуры (термоконтактор ртутный); *SL* – кондуктометрическое реле уровня; *A, B, C* – приемные элементы (катушки электромагнитных реле); исполнительные элементы: *M* – электродвигатель; *HL* – сигнальная лампа; *SB* – кнопка ручного пуска; *ЛБ* – логический блок, состоящий из контактов реле *A, B, C*; СЛУ – система логического управления; *KM* – магнитный пускатель

Таблица 1.2

Таблица состояний

	A	B	C	M	HL
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 *M*:

$$M_1 = a \cdot \bar{b} \cdot \bar{c}; \quad M_2 = a \cdot \bar{b} \cdot c; \quad M_3 = a \cdot b \cdot \bar{c}\,.$$

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

$$M = M_1 + M_2 + M_3 = a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c} = a \cdot \bar{b}\left(\underbrace{c + \bar{c}}_{1}\right) + a \cdot b \cdot \bar{c} =$$

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

Аналогично составляем логическую функцию для элемента *HL*:

$$\mathrm{HL}_1 = \bar{a} \cdot b \cdot \bar{c};$$

$$\mathrm{HL}_2 = a \cdot \bar{b} \cdot \bar{c};$$

$$\mathrm{HL} = \mathrm{HL}_1 + \mathrm{HL}_2 = \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} = \bar{c}(\bar{a} \cdot b + a \cdot \bar{b}).$$

Релейно-контактные варианты проектируемой СЛУ представлены на рисунке 1.2. Так как трехфазный асинхронный двигатель *M* нельзя включить в цепь оперативного тока, то в четвертую цепь пришлось установить катушку магнитного пускателя *КМ*. Очевидно, что логика работы СЛУ при этом не изменится, т. к. *КМ* = *M* (в булевом понимании).

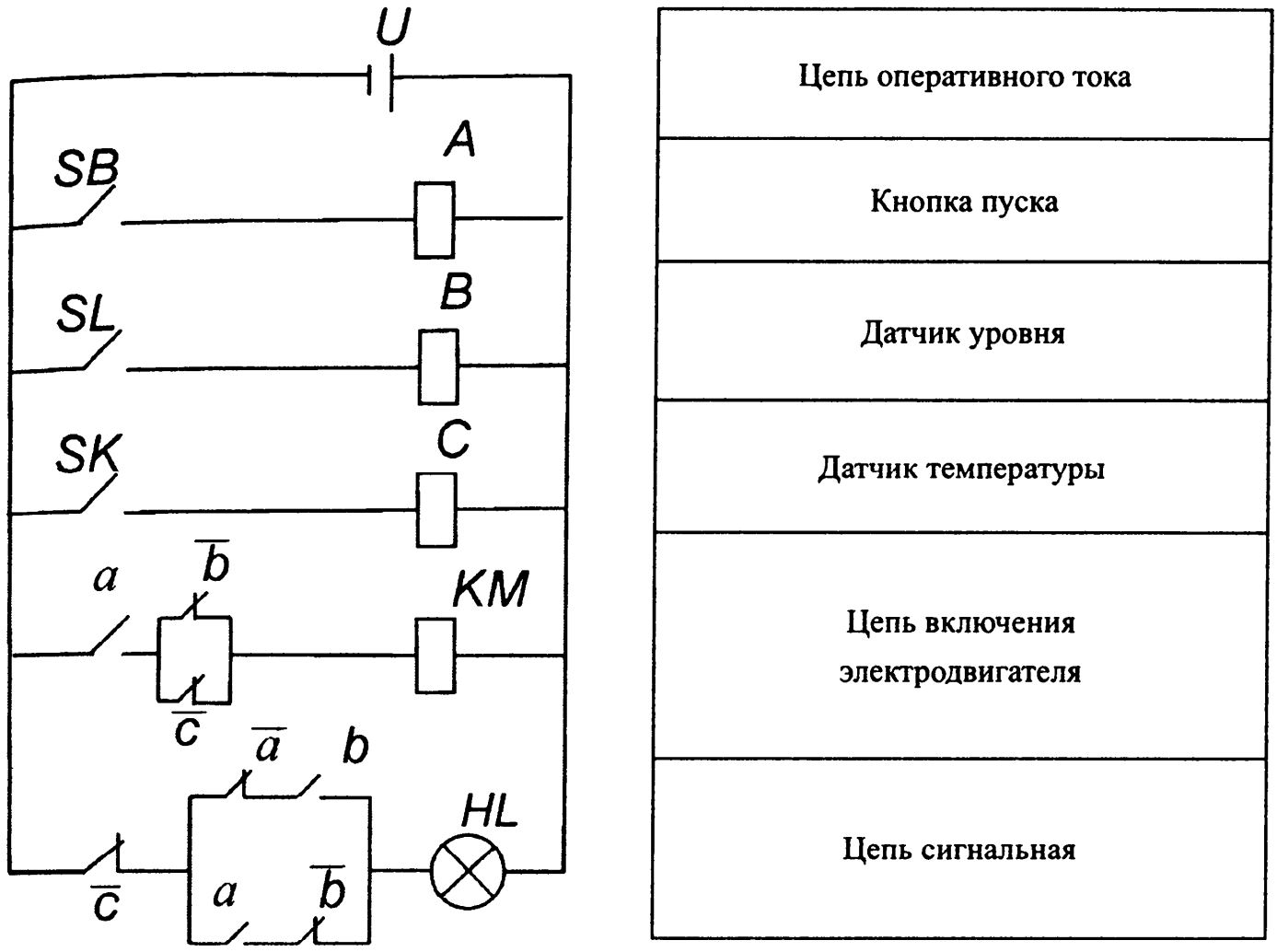


Рис. 1.2. Схема релейно-контактной комбинационной СЛУ

Логические функции для *M* и *HL* можно было получить исходя из условий несрабатывания.

Например, для *M* из первой строки таблицы следует:

$$M_1 = a + b + c.$$

Аналогично для других условий несрабатывания:

$$M_2 = a + b + \overline{c}; \quad M_3 = a + \overline{b} + c; \quad M_4 = a + \overline{b} + \overline{c}; \quad M_5 = \overline{a} + \overline{b} + \overline{c}.$$

Общее условие несрабатывания будет конъюнкцией частных условий:

$$M = (a + b + c) \cdot (a + b + \overline{c}) \cdot (a + \overline{b} + c) \cdot (a + \overline{b} + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c}).$$

По закону склеивания сгруппируем первый и второй, четвертый и пятый сомножители. Тогда

$$M = (a + b) \cdot (a + \overline{b} + c) \cdot (\overline{b} + \overline{c}).$$

Открываем скобки, и, выполнив рутинную работу по умножению, получаем

$$M = a(\overline{b} + \overline{c}).$$

Что и требовалось доказать.

Этот прием оправдывается в том случае, если условий несрабатывания значительно меньше, чем условий срабатывания. В нашем примере это условие не выполняется, что привело к повышению трудоемкости в получении результата.

1.9. Синтез последовательностной СЛУ на контактных элементах

Допустим, необходимо в рассмотренную СЛУ (рис. 1.1) по рекомендации технологов внести дополнительные функции, а именно: исполнительный элемент *HL* – сигнальная лампа должна включиться при выполнении ее условий срабатывания, но только после включения двигателя *M* через 20 с и перейти в мерцающий режим с длительностью в 2 с и с паузой в 1 с. При этом состояние *HL* не должно зависеть от изменения состояния *M*.

В этом случае в логической функции *HL* появляется третий аргумент – время *t*. Поэтому напрямую применить аппарат бинарной алгебры Буля нельзя. Надо искать другие приемы для описания таких систем.

Известны способы решения подобных задач с использованием теории автоматов Мили или автоматов Мура [1]. Но их рассмотрение не входит в нашу задачу, т. к. мы полагаем, что у читателя хватает опыта для анализа и синтеза простых СЛУ, опираясь на интуицию и здравый смысл.

Можно предложить такой вариант (рис. 1.3) СЛУ, отвечающей вышеописанным техническим условиям.

Очевидно, что факт (точнее, событие) срабатывания *M* необходимо зафиксировать в новом состоянии каким-то элементом памяти. В РКС это решается за счет применения промежуточного реле *D*, которое при срабатывании становится на самофиксацию за счет своего же контакта *d* и включает двигатель *M*, точнее катушку магнитного пускателя *КМ*.

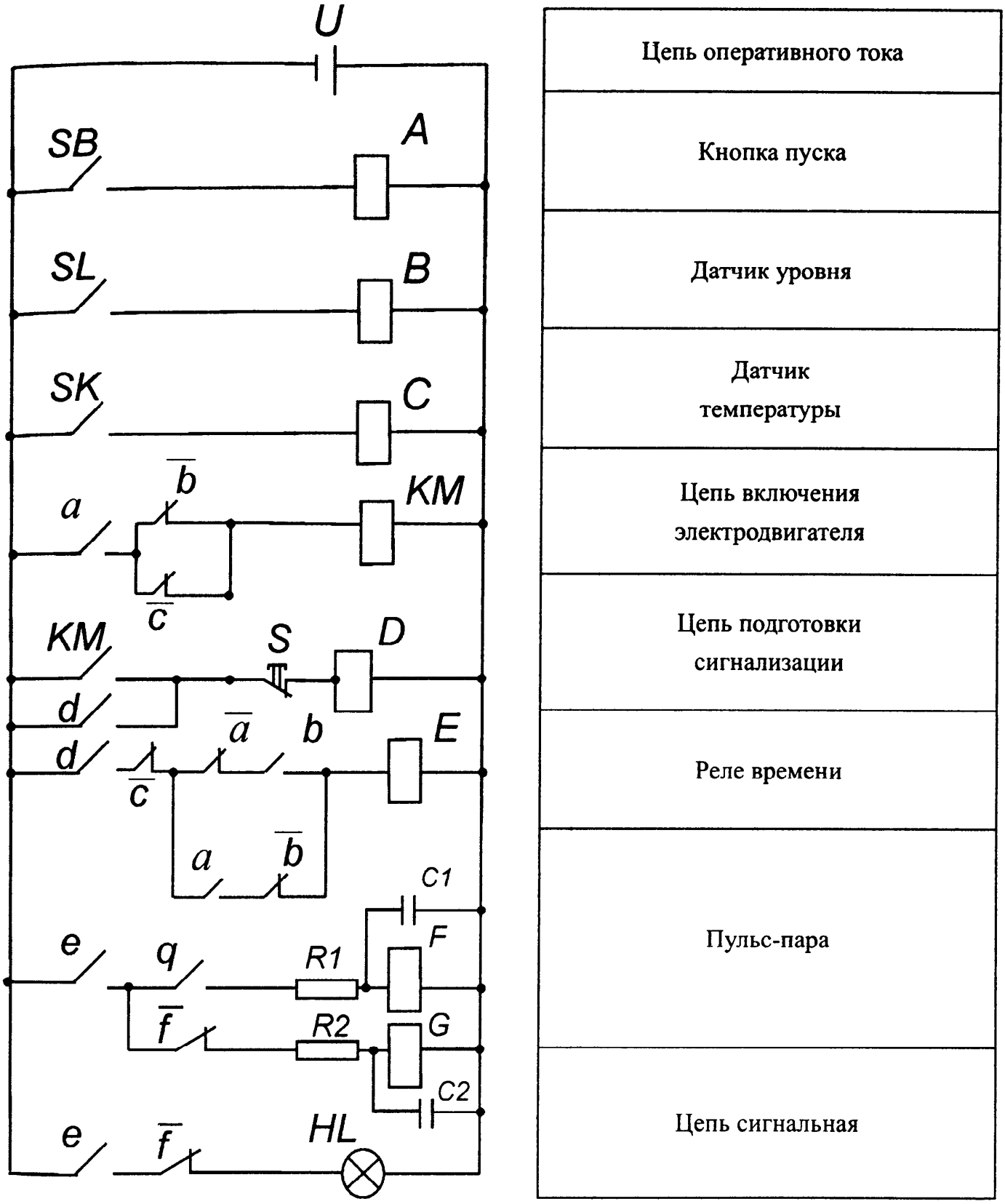


Рис. 1.3. Релейно-контактный вариант последовательностной СЛУ

Естественно, надо сразу же предусмотреть возможность оперативного вмешательства для снятия этой блокировки и отключения *M*. С этой целью потребуется кнопка *S*. Это же реле при выполнении условий срабатывания *HL* должно обеспечить включение промежуточного реле времени *E*, контакт которого замкнется лишь через 20 с за счет пневматического замедлите-

ля (такие реле еще выпускаются! Например, реле времени пневматическое РВП72).

Замыкающий контакт *e* этого реле включит лампу *HL* и так называемую пульс-пару (прошлый век!), выполненную на реле *F* и *G* и работающую в качестве генератора импульсов.

Длительность импульса (2 с) и паузы (1 с) подбирается с помощью R_1, R_2,C_1 и C_2. Очевидно, что напряжения срабатывания реле *F* и *G* должна быть в этом случае меньше напряжения питания, подведенного к шинам схемы.

Задача выполнена.

На этом краткий экскурс в середину прошлого века можно завершить. После знакомства с ПЛК еще вернемся к этому примеру.

2. КРАТКИЕ СВЕДЕНИЯ О ПЛК

Программируемые логические контроллеры представляет собой конечный (дискретный) автомат, имеющий конечное количество входов и выходов, подключенных посредством датчиков, ключей, исполнительных механизмов к объекту управления, и предназначенный для работы в режимах реального времени (рис. 2.1).

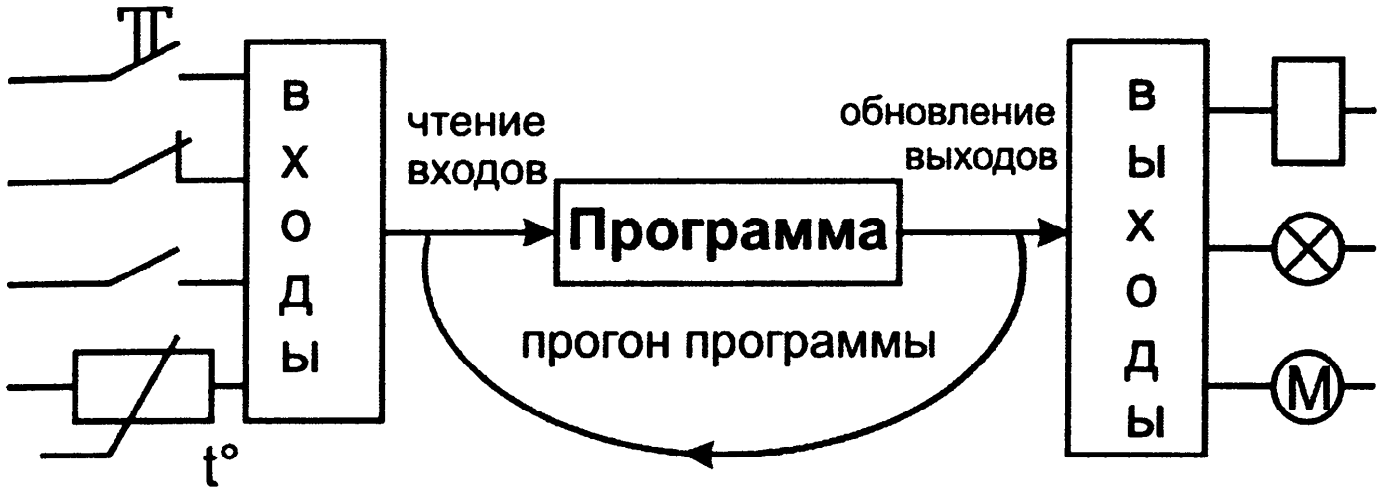


Рис. 2.1. Структура ПЛК

Конструктивно ПЛК выполняется как в моноблочном исполнении, так и в модульном. В первом варианте, так называемом мини-ПЛК, с небольшим количеством входов/выходов все собрано в одном корпусе. Модульные ПЛК имеют отдельные блоки (модули) ввода и вывода, которые могут находиться порой на значительном удалении от центральной части, т. е. установлены в непосредственной близости к управляемому объекту и «общаться» с микропроцессором по сети. В общем случае, все модули могут быть сосредоточены в одном месте на специальных стойках или каркасах.

Моноблочные ПЛК также имеют возможность подключаться к удаленным блокам ввода/вывода по цифровой сети. При этом сами клеммы входных и выходных плат ПЛК останутся свободными.

Например, самый простой моноблочный ПЛК 100 фирмы ОВЕН можно непосредственно соединить с входными цепями и исполнительными механизмами, а также по сети (рис. 2.2).

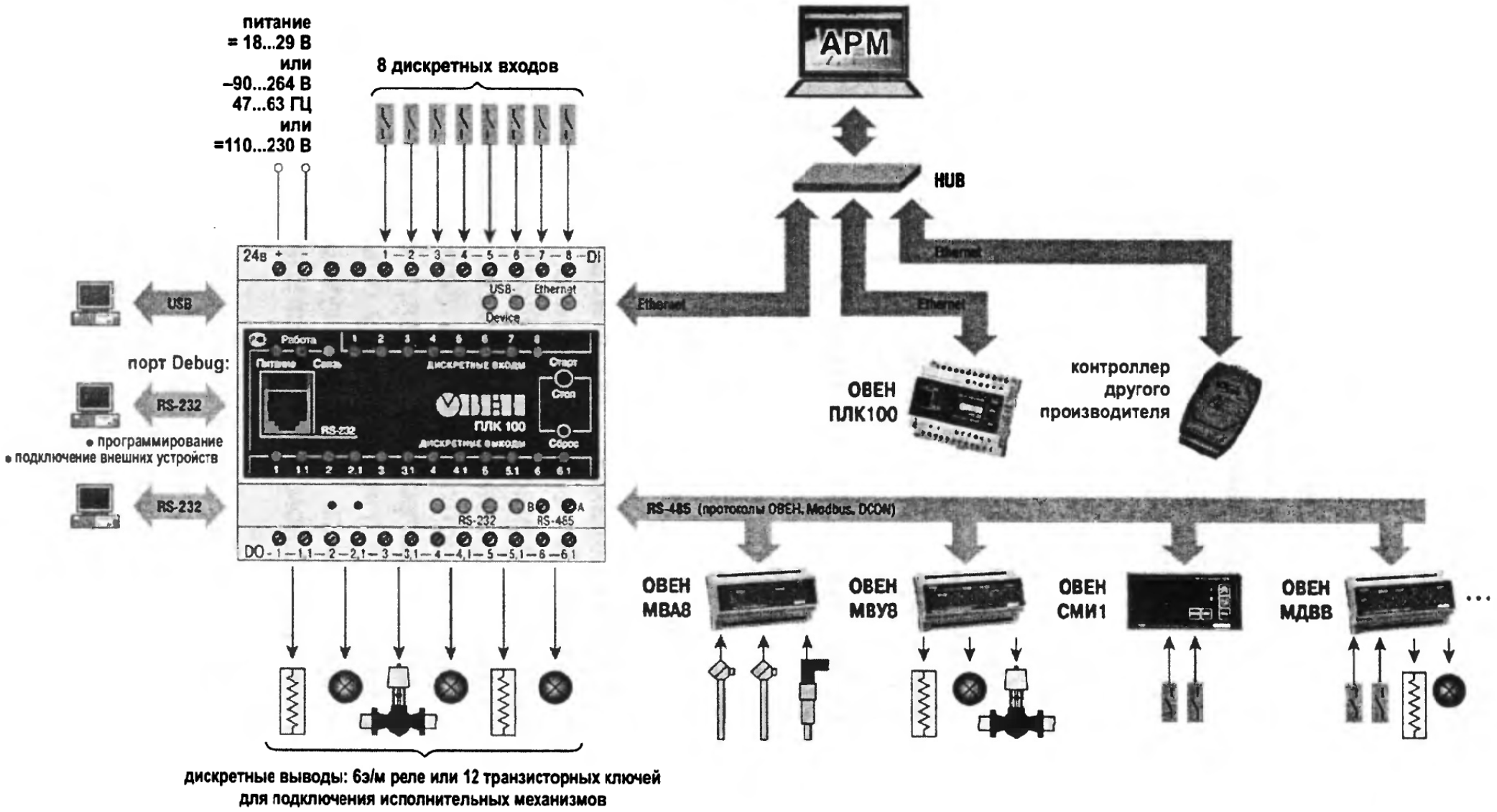


Рис. 2.2. Подключение ПЛК 100 ОВЕН: МВА 8 – модуль ввода аналоговый; МВУ 8 – модуль ввода управляющий; СМИ 1 – панель оператора; МДВВ – модуль дискретного ввода-вывода

В этой схеме необходимо выполнить настройку (конфигурирование) модуля МВА 8 на выдачу ***дискретного*** сигнала при достижении аналоговым сигналом от какого-то датчика порогового значения (уставки), т. к. этот ПЛК работает только с бинарными переменными.

ПЛК обычно применяются для управления ***последовательными*** процессами, используя дискретные входы и выходы для определения состояния объекта и выдачи управляющих воздействий.

Например, при управлении электрокалорифером оператор вручную нажимает кнопку «Пуск». Первым должен включиться двигатель вентилятора и через 5 с – нагревательные элементы. При нажатии на кнопку «Стоп» первыми отключаются нагревательные элементы и через 8 с – вентилятор. Можно предусмотреть включение и отключение с помощью контактов какого-то реле времени с суточной программой или использовать сигналы от датчика температуры воздуха в обогреваемом помещении. Скорее всего, следует предусмотреть какие-то средства защиты электрокалорифера (да и обслуживающего персонала). С этой целью потребуется еще ряд дискретных сигналов от датчиков, реагирующих на потерю фазы, перегрев или заклинивание двигателя, разрушение нагревательного элемента и т. д.

В каждом случае контроллеру необходимо опрашивать состояние входных элементов (кнопок, датчиков) и, соблюдая определенную временную последовательность, выдавать управляющие сигналы.

ПЛК также может управлять ***непрерывными*** процессами, т. е. получать и выдавать аналоговые сигналы.

Естественно, выходные аналоговые сигналы в ПЛК преобразуются с помощью АЦП в дискретную, т. е. в цифровую форму.

Учитывая широкое распространение в системах контроля металлических термосопротивлений и термопар, в некоторых ПЛК могут быть предусмотрены специализированные входы для подобных датчиков.

Большинство ПЛК работают циклически, т. е. производится сканирование или периодический опрос входных данных, выполнение пользовательской программы и обновление выходных данных.

Такая последовательность действий называется ***прогоном*** программы, а период цикла – временем прогона.

В начале прогона ПЛК «читает» состояние ***всех*** входов и фиксирует их значение в памяти. Изменение состояния каких-либо входных сигналов в течение прогона программы будет воспринято ПЛК только в следующем цикле.

Также ведут себя и выходные данные, которые обновляются ***одновременно*** в конце прогона программы.

Время цикла, т. е. время сканирования, влияет на скорость реакции ПЛК при изменении входного сигнала. В общем случае, входной сигнал должен иметь длительность не меньше, чем время прогона программы, составляющего обычно от единиц до десятков миллисекунд.

Если изменение состояния входного сигнала произошло как раз в начале прогона, то реакция ПЛК проявится по завершении цикла.

Если же этот сигнал немного «запоздал», то потребуется еще один прогон программы для ответных действий ПЛК.

Различают системы ***жёсткого*** и ***мягкого*** реального времени. В первом случае выдвигается технологическое требование в отношении временного порога, превышение которого может создать аварийную ситуацию. В системах мягкого реального времени последствия такой задержки в реакции ПЛК практически неощутимы.

Системное программное обеспечение ПЛК доступно только их изготовителям. Прикладное программное обеспечение возлагается на пользователя ПЛК. Код прикладной или пользовательской программы размещается в энергонезависимой памяти и может многократно изменяться по мере необходимости.

Сейчас редко встречаются ПЛК, программирование которых выполняется с встроенного или выносного пульта. Современные ПЛК универсального назначения, поддерживающие стандарт МЭК 61131-3, программируются на ПК с помощью специализированных комплексов, среди которых наиболее популярным является CoDeSys, разработанный фирмой 3S (Smart Software Solutions). Этот универсальный инструмент программирования будет рассмотрен в следующий главе.

Прикладная программа становится переносимой, т. е. ее можно использовать в любом ПЛК, отвечающем требованиям стандарта МЭК.

Простейшее и наиболее популярное применение ПЛК – это создание автономной системы управления каким-либо технологическим процессом. В случае проектирования автоматизированной системы управления производством (АСУП) или разветвленным технологическим процессом (АСУТП) на ПЛК возлагаются функции нижнего звена в обработке входных сигналов и выработке управляющих воздействий в пределах каждого локального участника.

Используя стандартные протоколы обмена данными, среди которых обычно отдают предпочтение технологии OPC (OLE for Process Control), можно войти в SCADA – систему.

Более подробную информацию о ПЛК можно получить в источниках 2, 3.

3. КРАТКИЕ СВЕДЕНИЯ О КОМПЛЕКСЕ CoDeSys

3.1. Выбор комплекса и языка

Ведущие изготовители ПЛК, опираясь на собственные фирменные наработки в плане инструментального программного обеспечения и поддержавшие стандарт МЭК 61131-3, используют с различными вариациями один или несколько комплексов. В каждом из них есть свои «плюсы» и «минусы». Поэтому предпочтение тому или иному инструменту программирования диктуется в основном предыдущим накопленным опытом.

Наибольшей популярностью все-таки пользуется комплекс CoDeSys, который насчитывает более 150 адаптаций, не противоречащих стандарту МЭК, но учитывающий фирменные особенности.

CoDeSys (**Co**ntrollers **De**velopment **Sys**tem) представляет проектировщику удобную среду для программирования контроллеров на языках МЭК. Используемые редакторы и отладочные средства базируются на широко известных принципах.

На этой стадии ознакомления с данной книгой рекомендуем читателю установить в своем ПК среду CoDeSys. Это не составит труда, поскольку инсталляция среды представляет собой обычную процедуру. Производители ПЛК обычно поставляют этот комплекс в качестве бесплатного приложения или размещают его на своих сайтах.

Авторы книги выбрали в качестве предмета исследований ПЛК 100 PL фирмы ОВЕН как самый простой моноблочный мини- (иногда говорят, микро-) контроллер. На официальном сайте ОВЕН www.owen.ru в свободном доступе имеется этот комплекс и его описание на русском языке. Стоит отметить, что CoDeSys поставляется изначально в англоязычной версии, что представляет собой определенную трудность в ее освоении. Однако при использовании актуальной версии CoDeSys V2.3.9. и выше никаких русификаторов не следует загружать, поскольку русский интерфейс уже входит в дистрибутив. Если нет возможности обновить программу до более свежей версии или выучить в полном объеме курс английского языка – не беда, мы

постарались изложить весь материал и представить его на обоих языках. Установив CoDeSys на свой ПК, можно приступать к освоению приемов программирования, используя пока режим эмуляции и не имея даже на какое-то время самого ПЛК.

CoDeSys позволяет использовать языки IL, ST, LD, SFC, FBD и CFC. Мы же, как договаривались выше, будем использовать только графический язык LD.

3.2. Редакторы

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 т. д.) и трассировку их соединений; нумерацию цепей; масштабирование изображения, что позволяет увидеть всю диаграмму или какую-то её часть и выделять цветом активные цепи.

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

3.3. Инсталляция среды CoDeSys

Итак, среда CoDeSys уже установлена. Следующим обязательным шагом должна стать инсталляция целевой (аппаратной) платформы **TSP** (Target Support Packages). В нее включено все необходимое для CoDeSys при создании кода, отладки и конфигурировании аппаратуры. Платформа определяет параметры генератора кода, распределение памяти, функциональность ПЛК, модули ввода-вывода. Кроме того, в **TSP** могут входить дополнительные библиотеки, драйверы связи и список команд ПЛК-Браузера.

Центральным местом в **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. 3.1) – 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 скачены 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.

Аналогичные действия следует выполнить, если читатель планирует свою работу с ПЛК других производителей, руководствуясь при этом их рекомендациями.

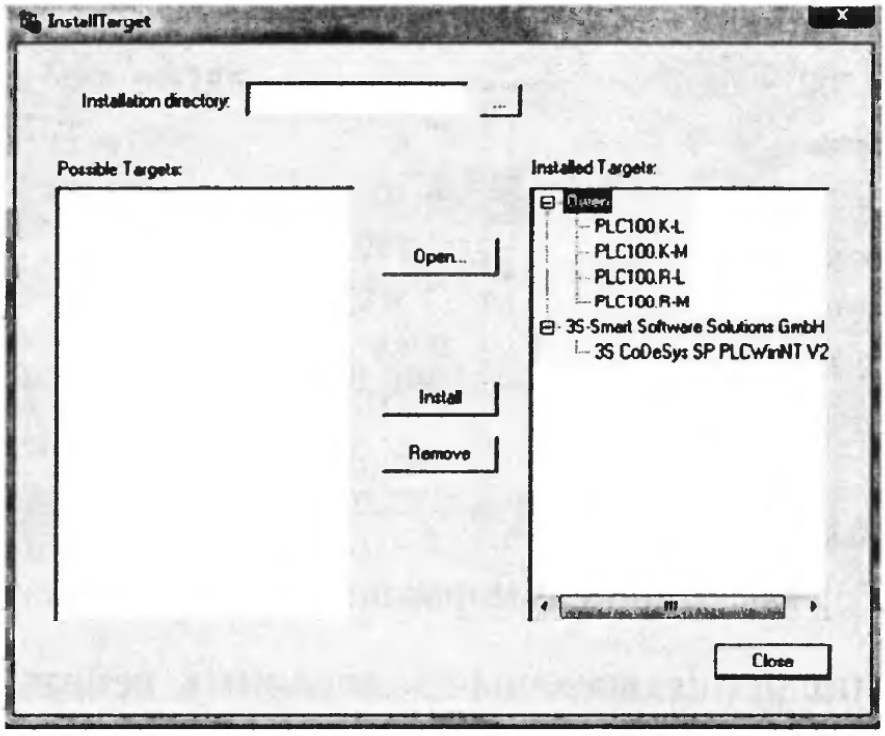


Рис. 3.1. Окно **«InstallTarget»** утилиты **InstallTarget**

3.4. 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 (рис. 3.2) в строках Program, Function Block или Function. Для LD будем использовать только Program, т. к. нам потребуются только стандартные компоненты (контакты, катушки реле, FB).

3.5. Запуск CoDeSys

Произведем первый запуск среды CoDeSys. В окне **Target Settings** напротив строки **Configuration** выбираем тип логического контроллера **PLC 100.R-L**, поскольку в нашем случае мы остановились именно на нем, и нажмем **OK**. В появившемся окне **New POU** (рис. 3.2) выбираем тип **POU** Program (Программа) и язык, на котором будет осуществляться написание программы – LD. Имя программы оставляем без изменения. Подтверждаем выбор нажатием на кнопку **OK**.

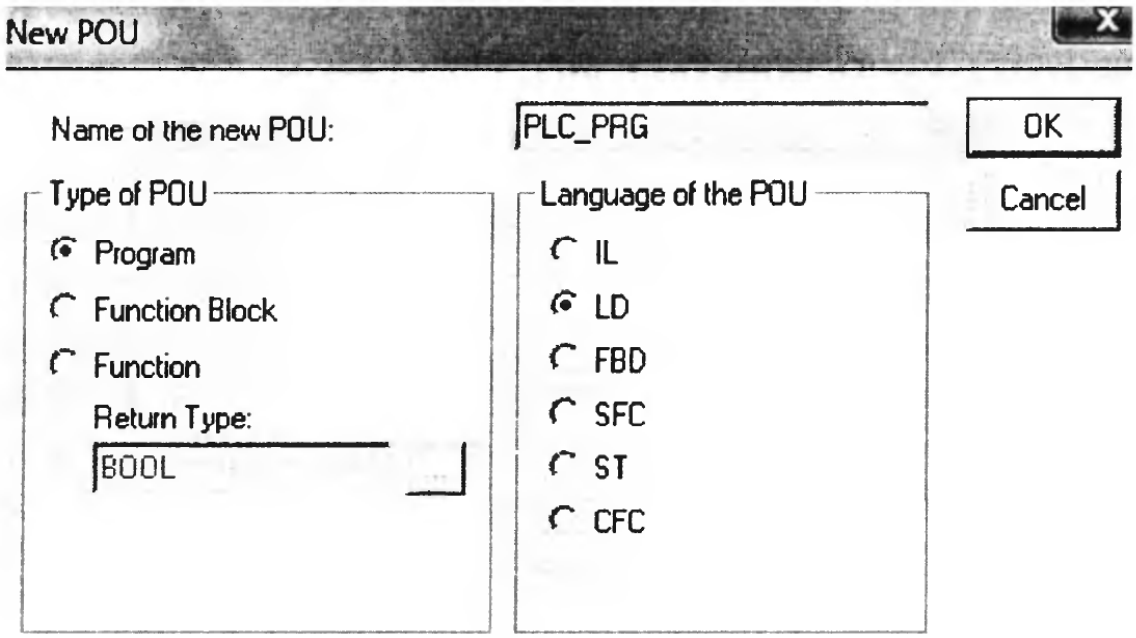


Рис. 3.2. Выбор языка программирования и задание имени программы

После выполнения всех вышеописанных действий откроется главное окно (рис. 3.3) среды CoDeSys.

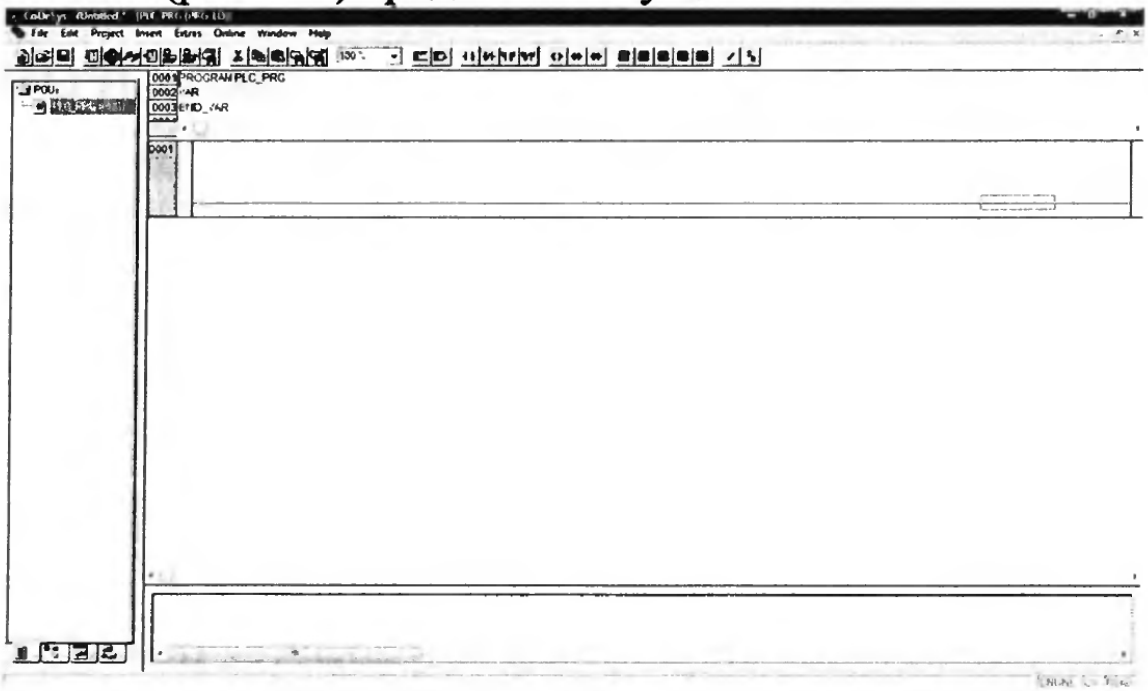


Рис. 3.3. Главное окно среды CoDeSys

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

- Меню (рис. 3.4).
- Панель инструментов. На ней находятся кнопки для быстрого вызова команд меню (рис. 3.5).
- 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.

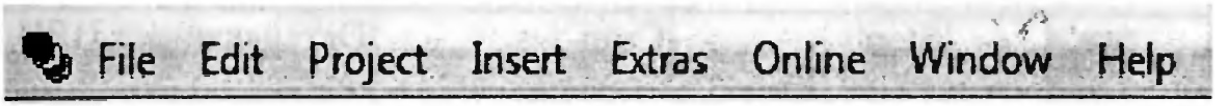


Рис. 3.4. Меню среды

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

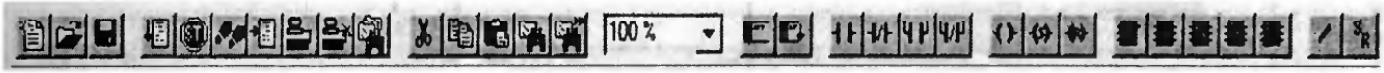


Рис. 3.5. Панель инструментов

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

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. Рассмотрим каждую из них по отдельности.

Кнопок много. Однако они очень важны, и их использование упрощает составление программы на языке LD. В других языках эта панель выглядит иначе.

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

Таблица 3.1

Описание графических изображений кнопок в панели инструментов

№ п/п	Графическое изображение кнопки	Англоязычная среда	Русскоязычная среда	Описание
1		New	Создать	Создает новый проект
2		Open	Открыть	Открывает проект
3		Save	Сохранить	Сохраняет содержимое измененного объекта
4		Run	Старт	Запускает ПЛК
5		Stop	Стоп	Останавливает ПЛК
6		Step over	Шаг по верху	Перешагивает через текущую инструкцию, даже если это вызов подпрограммы
7		Toogle break-point	Переключить точку останова	Устанавливает / убирает точку останова в текущей позиции
8		Login	Подключение	Устанавливает связь с контроллером и включает режим On-Line
9		Logout	Отключение	Отключает режим On-Line
10		Global search	Глобальный поиск	Ищет заданную строку по всему проекту
11		Cut	Вырезать	Перемещает выделенную область в буфер обмена
12		Copy	Копировать	Копирует выделенную область в буфер обмена
13		Paste	Вставить	Вставляет содержимое из буфера обмена в текущую позицию
14		Find…	Найти	Ищет заданную строку в текущем окне
15		Find next	Найти далее	Повторяет последний поиск

№ п/п	Графическое изображение кнопки	Англоязычная среда	Русскоязычная среда	Описание
16	100 %	–	Масштаб	Увеличение / уменьшение масштаба
17		Network (before)	Цепь (вперед)	Вставляет цепь перед текущей
18		Network (after)	Цепь (после)	Вставляет цепь после текущей
19		Contact	Контакт	Вставляет последовательный (замыкающий) контакт
20		Contact (negated)	Инверсный контакт	Вставляет инверсный (размыкающий) контакт
21		Parallel Contact	Параллельный контакт	Вставляет параллельный (замыкающий) контакт
22		Parallel Contact (negated)	Параллельный инверсный контакт	Вставляет инверсный параллельный (размыкающий) контакт
23		Coil	Обмотка	Вставляет обмотку (катушку) «Реле»
24		«Set» Coil	«Set» обмотка	Вставляет «Set» обмотку (катушку)
25		«Reset» coil	«Reset» обмотка	Вставляет «Reset» обмотку
26		Function block	Функциональный блок	Вставляет функциональный блок
27		Box with «EN»	Элемент с «EN»	Вставляет элемент со входом разрешения
28		Rising edge detection	Детектор переднего фронта	Вставляет детектор переднего фронта
29		Falling edge detection	Детектор заднего фронта	Вставляет детектор заднего фронта
30		Timer «TON»	Таймер «TON»	Вставляет таймер «TON»
31		Negate	Инверсия	Инвертирует выбранный выход или вход
32		«Set» / «Reset»	«Установка» / «сброс»	Преобразует выход в «Set» / «Reset» выход

Рис. 3.6. Организатор объектов

Организатор объектов (рис. 3.6) 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.

3.6. Разделитель экрана

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

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

Рабочая область (рис. 3.3) 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.

3.7. 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 сообщении 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.)

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

3.8. Статусная строка

Статусная строка (рис. 3.7) находится в нижней части главного окна CoDeSys и предоставляет информацию о проекте и командах меню.

Рис. 3.7. Статусная строка

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

Если 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.

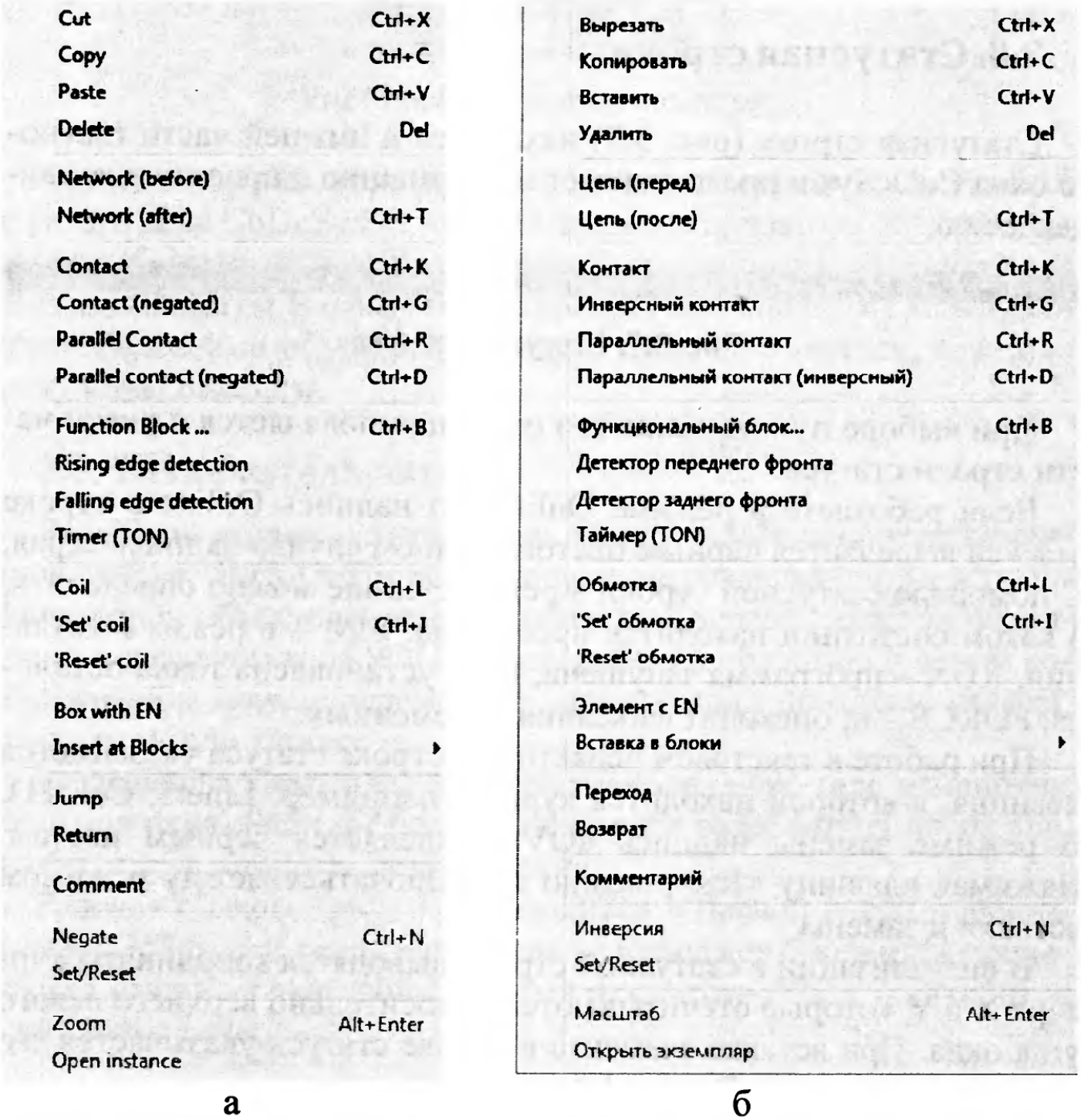
В визуализации в статусной строке выводятся координаты курсора X и Y, которые отсчитываются относительно верхнего левого угла окна. При вставке элемента в строке статуса указывается его название (например, Rectangle).

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).

3.9. Контекстное меню

Альтернативой использования главного меню для вызова команд может стать контекстное меню. Это меню, вызываемое 1ПКМ на рабочей области, содержит наиболее часто используемые команды. На рисунке 3.8 представлены меню из версий CoDeSys на английском и русском языках соответственно.



а б

Рис. 3.8. Контекстное меню программы CoDeSys:
а – контекстное меню на английском языке;
б – контекстное меню на русском языке

Для проведения ознакомительных исследований компонентов LD-диаграмм и самих многоступенчатых схем пока достаточно этих сведений, т. к. все наши действия будут проведены без участия самого ПЛК, т. е. в режиме эмуляции.

В главе 5 продолжим описание CoDeSys в плане переноса программы с ПК в память ПЛК и последующих операций.

4. ПРОЕКТИРОВАНИЕ СЛУ НА ЯЗЫКЕ LD

4.1. Контакты, катушки

После открытия главного окна CoDeSys появляется на мониторе рабочая область, в которой и будем «рисовать» многоступенчатую схему (рис. 3.3).

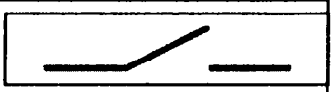
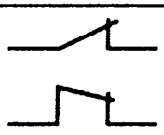
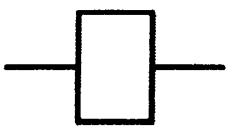
Эта схема представляет собой набор ***горизонтальных*** цепей, напоминающих ступеньки лестницы, соединяющих ***вертикальные*** шины питания. Если необходимо увеличить размер рабочей области, то подводим курсор к кнопке □ в верхнем правом углу этой области и нажимаем 1ЛКМ.

Первая цепь появляется в рабочей области сразу (рис. 3.3). 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, FB, катушки.

Будем считать, что сама СЛУ в обычном, т. е. релейно-контактном исполнении уже имеется, и наша задача перенести её в LD-диаграмму. В качестве такой СЛУ возьмем схему, представленную на рис. 1.2. Так как графическая символика в LD создавалась под американские стандарты, придется согласиться с отступлением от требований ЕСКД в изображении элементов схемы и воспользоваться таблицей 4.1.

Таблица 4.1

Изображение релейных элементов в LD и ЕСКД

Название компонента	Изображение	
	LD	ЕСКД
Замыкающий контакт		
Размыкающий контакт		
Катушка реле		

4.2. Построение СЛУ в LD

4.2.1. По аналогии с исходным вариантом (рис. 1.2) в многоступенчатой схеме, ***скорее всего,*** также потребуется пять цепей. Мы говорим «скорее всего», т. к. несмотря на кажущуюся схожесть РКС- и LD-диаграмм, есть принципиальное различие в последовательности срабатывания цепей. Но этот вопрос пока оставим без комментариев и вернемся к нему в п. 4.8.2.

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

– 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 (рис. 4.1).

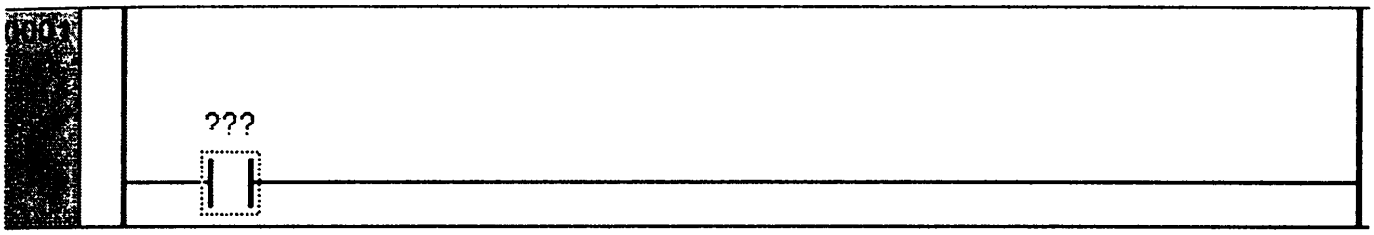


Рис. 4.1. Первая цепь с введённым контактом

(Далее для краткости подобные операции будем записывать так:

– наводим курсор на К19, нажимаем 1ЛКМ...., где К19 – кнопка в панели инструментов с порядковым номером 19 по табл. 3.1).

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

– Наводим курсор на ???, щелкаем 1ЛКМ. Вопросы становятся белыми на фоне синего прямоугольника. С помощью клавиатуры на ***английском*** языке «вбиваем» имя. В нашем примере «**SB**».

Буквы ***русского*** языка использовать ***нельзя***!

– Нажимаем <Enter>.

Открывается окно **Declare Veriable** (Объявление переменной), запрашивающее к какому классу переменных будет отнесен наш элемент (рис. 4.2).

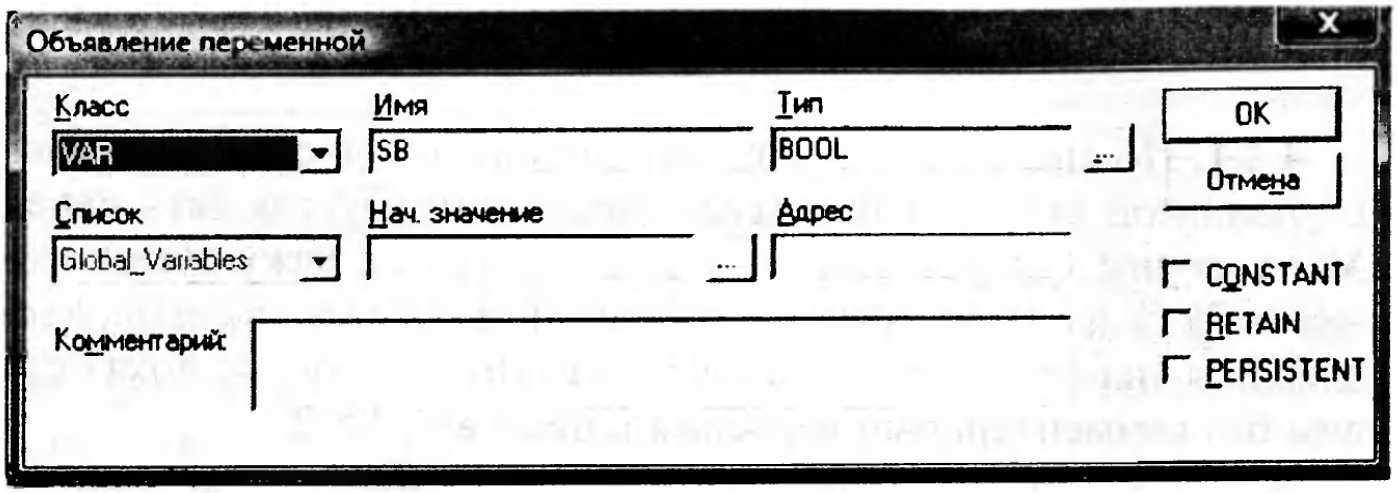


Рис. 4.2. Окно объявления переменной

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. 4.3).

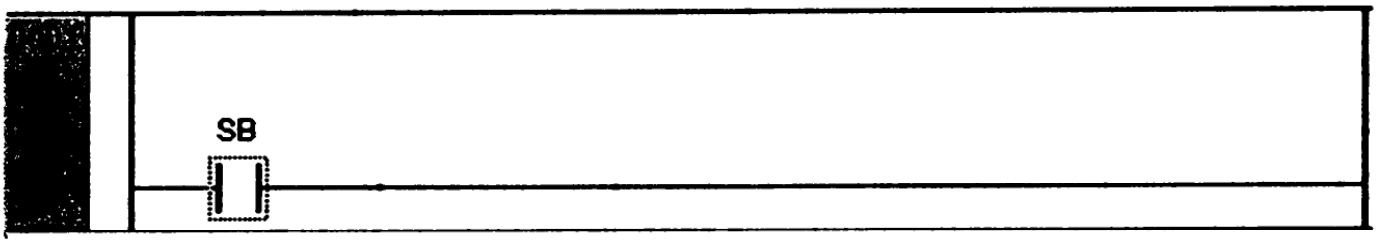


Рис. 4.3. Объявленный замыкающий контакт

Обратите 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 контакту (как будет показано при создании четвертой и пятой цепей) подключать к нему другие контакты.

Нам же осталось для завершения первой цепи ввести в нее катушку реле **A**.

– Наводим курсор на К23, 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 К23, щелкаем 1ЛКМ и в цепи появляется катушка с тремя белыми ??? в синем прямоугольнике.

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

– Присваиваем катушке имя: A. Нажимаем <Enter>, открывается окно **Declare Veriable**, нажимаем 1ЛКМ на ОК.

Можно было сначала включить в цепь катушку, затем контакт.

Можно было включить эти элементы в любой последовательности и лишь потом присвоить им имена. Результат будет тем же.

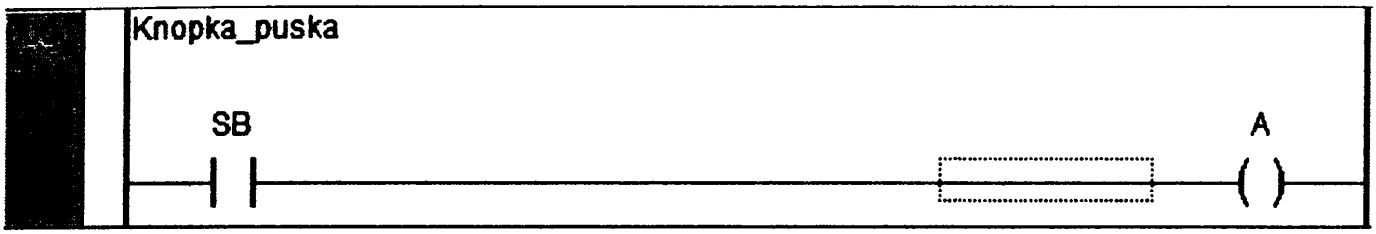


Рис. 4.4. Завершенная первая цепь

Первая цепь завершена (рис. 4.4). Как видно на мониторе, зона цепи ограничена двумя горизонтальными линиями. При активации цепи не 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 написания на ***английском*** языке функционального назначения данной цепи.

Если в этом есть необходимость. А такая необходимость возникает в сложных LD-диаграммах, состоящих из десятков или сотен цепей. В РКС эти сведения обычно приводят в виде таблицы слева от схемы, как показано на рисунке 1.1, на любом языке, понятным, естественно, обслуживающему персоналу.

С этой целью наводим указатель в эту полоску, щелкаем 1ЛКМ, появляется мигающий курсор, клавиатурой вбиваем, например, «Knopka_ puska» (если у читателя есть проблемы с английским языком), нажимаем <Enter>. И всё.

4.2.2. Комментарии можно выполнить и на русском языке. С этой целью наводим курсор на поле цепи, щелкаем 1ПКМ. Открывается контекстное меню (рис. 3.8). Щелкаем 1ЛКМ на строке Comment (Комментарий). Слева над цепью появится слово «Comment», в строке с которым можно написать на любом языке необходимые пояснения. Слово «Comment» потом можно удалить.

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

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

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

Все перечисленные требования к написанию имен действуют и при документировании функциональных характеристик цепей; «Knopka_pusk» или «Datchik_urovnia».

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

Нельзя применять в качестве имен операторы других языков. Например, операторы **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 русские имена «**Dvigatel**», «**pusk**», «**BLOKIROVKA**» и т. д.

Каждая цепь ***заканчивается*** «катушкой» реле. ***Последовательно*** соединять катушки нельзя. ***Параллельно*** – можно. Хотя в этом особой необходимости нет.

Некоторые фирмы, выпускающие ПЛК, допускают завершение цепи не катушкой, а FB. Например, счетчиком, виртуальные контакты которого, имеющие тот же идентификатор, что и FB, могут использоваться в цепях LD.

Создаем вторую цепь. Есть три способа: с помощью меню <Insert>, контекстного меню (рис. 3.9) и кнопками К17 и К18. Воспользуемся третьим приемом, как наиболее простым. (Другие способы читатель потом легко освоит самостоятельно).

- Наводим курсор на К18, щелкаем 1ЛКМ. Сразу появляется вторая цепь (рис. 4.5).
- По вышеописанной методике вносим в цепь **SL** и катушку **B**. Если есть необходимость (а для такой простой СЛУ она вряд ли есть), то можно в верхней полосе зоны второй цепи вписать: «Datchik_urovnia», или окончательно освоив английский язык, «level_sensor».

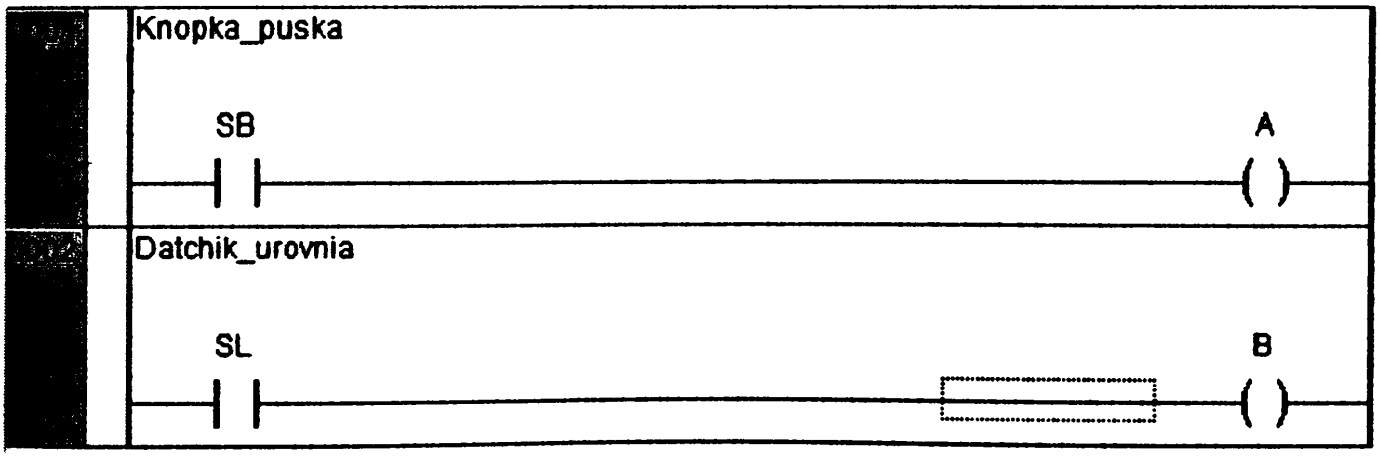


Рис. 4.5. Диаграмма с новой цепью

Также создается третья цепь. Если по ошибке или преднамеренно нажали не К18, а К17, т. е. Network (before) или 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 К18, т. е. по Network (after) или 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 К17 и новая цепь появится впереди цепи 0002, присвоит себе её номер, а ранее созданной цепи передаст номер 0003.

Несколько сложнее будет программирование четвертой и пятой цепей.

Начнем с четвертой цепи. По освоенной методике создаем саму цепь, вносим контакт реле **A**. Присваиваем ему имя «**A**» или «**a**». Для большего сходства с исходной РКС впишем «**a**». Нажимаем <Enter>. Однако окно объявления переменной (рис. 4.2) не открылось. В этом нет необходимости, т. к. имя реле **A** уже внесено в «список» булевых переменных. И сколько бы раз этот идентификатор «**A**» или «**a**» не появлялся в схеме, окно открываться не будет. (Нам меньше хлопот!).

Далее с помощью К20 вносим размыкающий контакт, присваиваем ему имя **b** по вышеописанной методике.

Теперь необходимо параллельно **b** ввести размыкающий контакт ***c***. Для этого:

– Наводим курсор на контакт **b**, щелкаем 1ЛКМ. Контакт активирован, о чем свидетельствует пунктирный квадрат, охватывающий контакт.
– Наводим курсор на К22, щелкаем 1ЛКМ и соединение выполнено. Присваиваем имя: ***c***.

(Если активировать саму цепь, а не контакт **b**, то замыкающий контакт **c** охватил бы всю цепочку из контактов **a** и **b**, как показано на рис. 4.6).

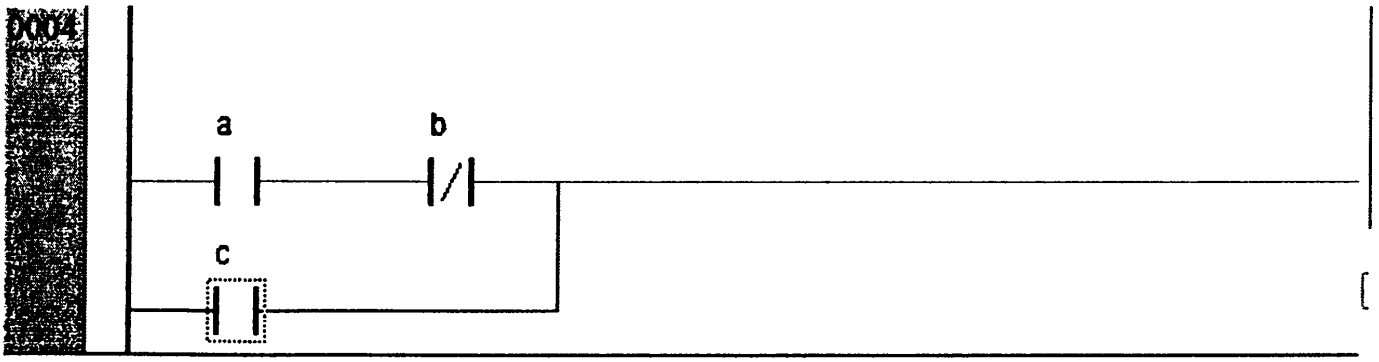


Рис. 4.6. Фрагмент схемы после неудачного введения контакта **c**

Вносим в цепь катушку реле КМ. Цепь завершена.

Программируем последнюю цепь.

По известной методике последовательно вносим в цепь размыкающие контакты **c**, **a**, замыкающий **b** и катушку **HL**. Осталось лишь создать параллельную цепочку из замыкающего контакта **a** и размыкающего **b**. Для этого:

– Активируем размыкающий контакт **a**, нажимаем и не отпускаем «Shift», активируем замыкающий контакт **b**, отпускаем «Shift».

Оба контакта оказались охваченными пунктирным прямоугольником.

– Наводим курсор на К21, щелкаем 1ЛКМ, и параллельно с этими контактами появится —| |— , которому присвоим имя **a**. Фрагмент полученной схемы в пятой цепи показан на рисунке. 4.7.

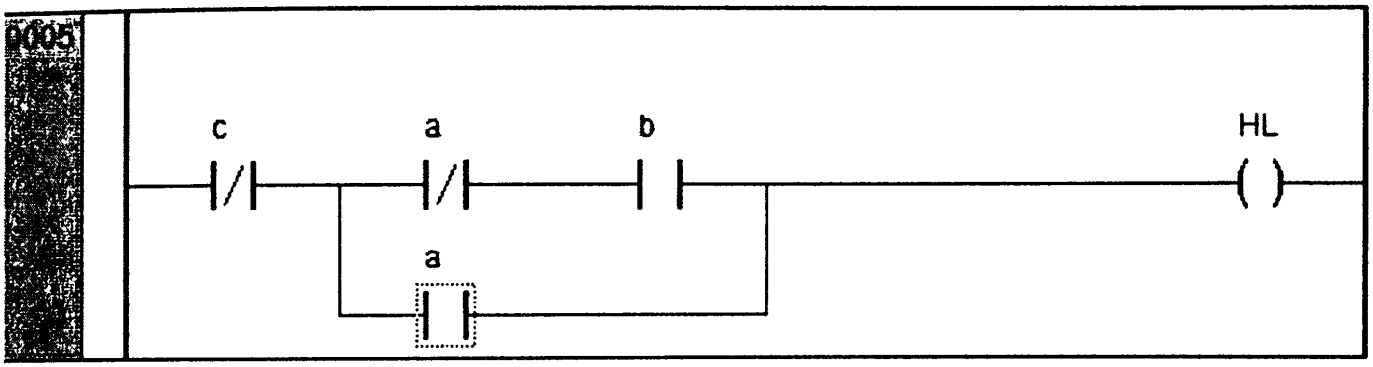


Рис. 4.7. Пятая цепь после введения контакта и его активации

– Активируем новый контакт **a**, наводим курсор на К20, щелкаем 1ЛКМ и последовательно с **a** появляется —|/|—. Присваиваем имя **b** новому элементу.

Параллельная цепочка выполнена, и завершена вся многоступенчатая схема в LD (рис. 4.8).

4.3. Подключение ПЛК

Монтажная схема (подключение приемных и исполнительных элементов непосредственно к ПЛК 100-24 PL ОВЕН) показана на рисунке 4.9. По принятой в ОВЕНе маркировке изделий «24» означает наличие внешнего маломощного источника питания на 24В (например, ОВЕН БП15), а буква Р свидетельствует, что все дискретные выходы обеспечиваются замыкающими контактами электромагнитных реле, рассчитанными на нагрузку 8А при ~220В.

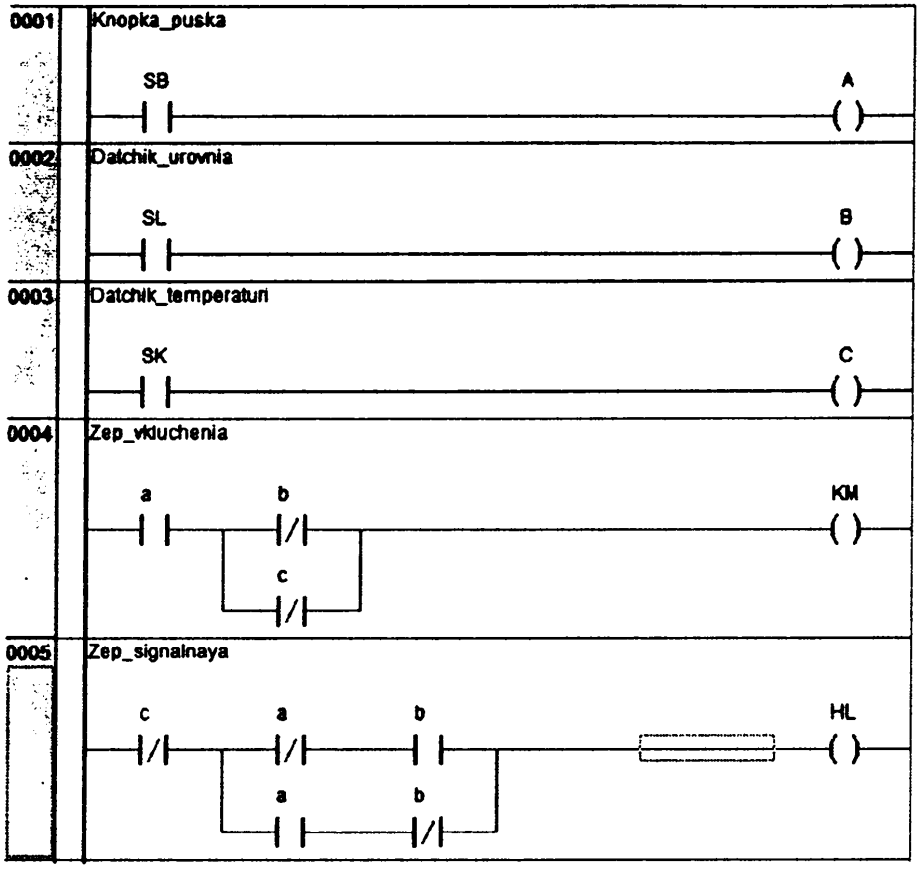


Рис. 4.8. СЛУ на языке LD

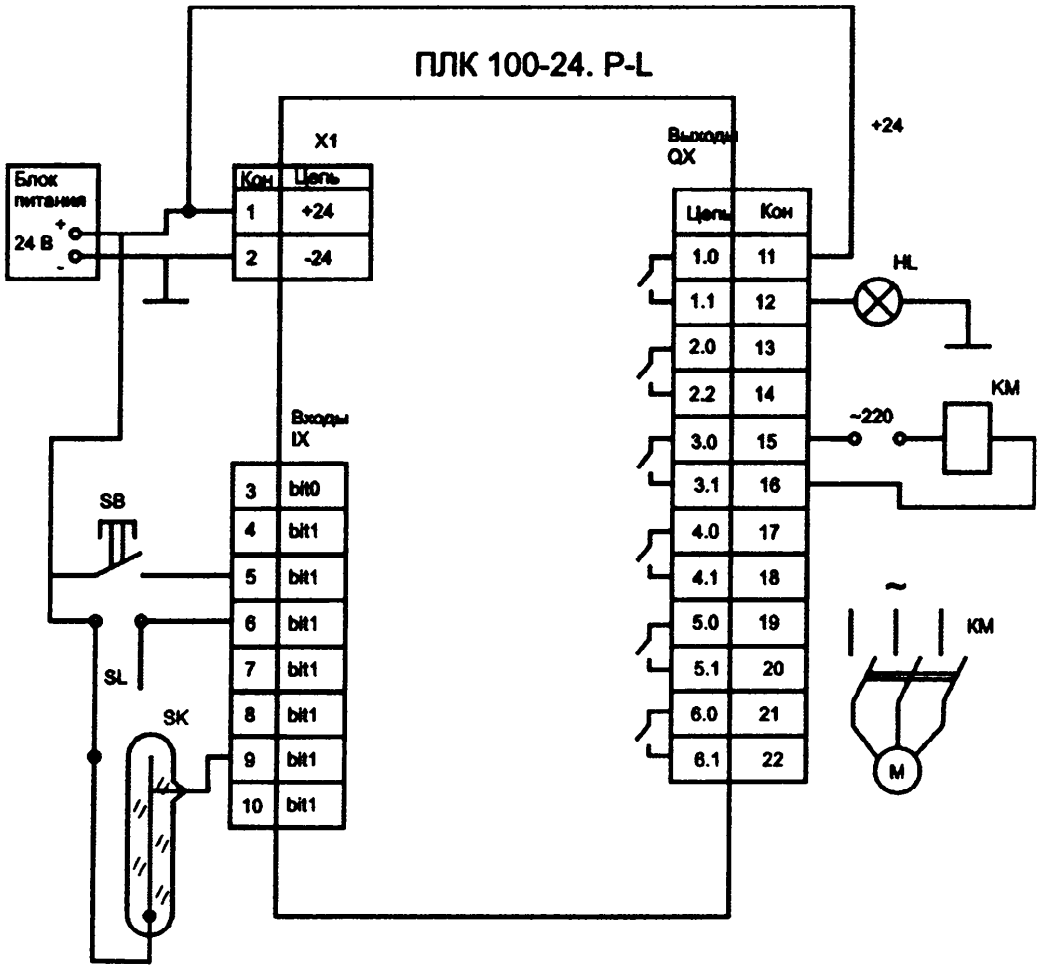


Рис. 4.9. Схема подключения цепей к ПЛК 100-24. P-L

Вопросы, касающиеся конфигурирования ПЛК, т. е. привязки имен входных и выходных элементов к конкретным входам/выходам ПЛК, будут рассмотрены в главе 5. Допустим, что эта процедура выполнена и выбраны монтажные клеммы, как показано на рисунке 4.9. Остальные цепи входов/выходов остаются свободными или могут быть задействованы в другой программе.

Для надежной работы кондуктометрического реле уровня следовало бы поставить блок согласования датчика (например, ОВЕН БКК 1) с входом ПЛК. Но не будем усложнять схему, т. к., возможно, с учетом конкретных технологических требований и условий эксплуатации разработанной системы потребуется ещё установка барьеров искрозащиты (например, ОВЕН ИСКРА), устройства защитного отключения трехфазного электродвигателя (например, ОВЕН УЗОТЭ-2У) и др. Но эти вопросы выходят за рамки нашей книги.

4.4. Дополнительные приемы при построении LD-диаграмм

4.4.1. Если по ошибке введен вместо замыкающего контакта размыкающий или наоборот, то можно навести на него курсор, щелкнуть 1ПКМ и в открывшемся контекстном меню (рис. 3.9) щелкнуть 1ЛКМ на четвертой снизу строке Negate (инверсия). Произойдет инверсия этого элемента. Имя, если оно уже было присвоено, остается прежним.

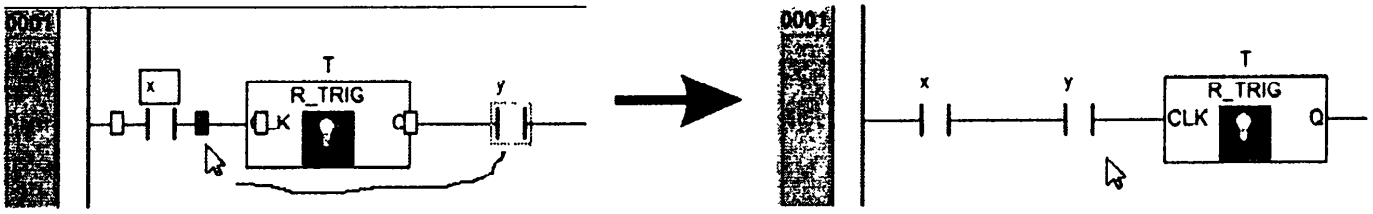
Эту операцию можно выполнить проще, если есть кнопка К31. (В некоторых версиях CoDeSys она может отсутствовать). Тогда наводим курсор на подлежащий изменению контакт, щелкаем 1ЛКМ и нажимаем на К31.

4.4.2. Если необходимо удалить какой-то компонент цепи или всю цепь со всеми элементами, то наводим курсор на этот компонент (контакт, катушку, FB) или на саму цепь, щелкаем 1ПКМ и в открывшемся контекстном меню нажимаем 1ЛКМ на самой верхней строке Cut (вырезать). Исчезнет элемент или вся цепь соответственно.

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

4.4.4. 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 (рис. 4.10).



а б

Рис. 4.10. Вид цепи при перетаскивании элемента:
а – до переноса; б – после переноса

Как только курсор подойдет к одному из них, прямоугольник становится зеленым. Отпускаем ЛКМ и перемещаемый элемент займет новое место.

4.4.5. 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.

4.5. 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 поведению 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. 4.11а).

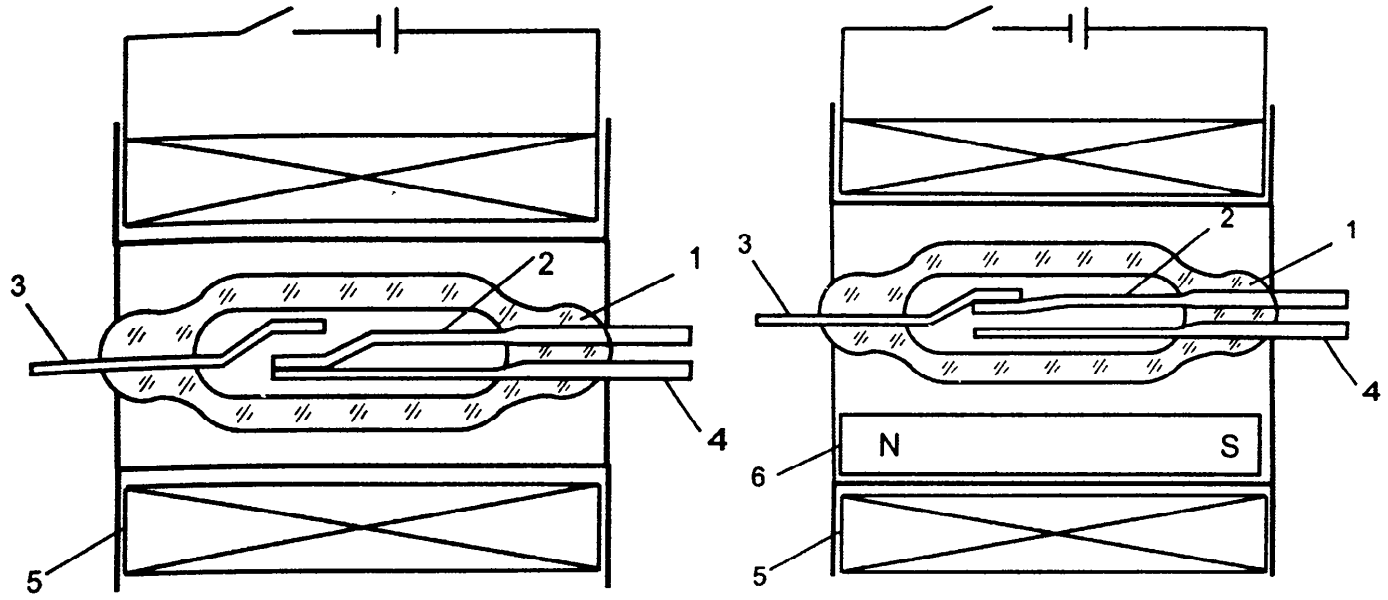


Рис. 4.11. Устройство герконового реле:
а – обычное исполнение; б – поляризованное реле

В нормальном состоянии подвижный контакт *2* замкнут с неподвижным контактом *4*. 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. 4.11б), то при отсутствии тока в катушке *5* подвижный контакт *2* замкнет цепь с контактом *3* и разорвет цепь с контактом *4*.

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. 4.12).

В наборе программных компонентов имеются также специальные обмотки **SET** и **RESET**, обозначенные в линейке кнопок как —(**S**)— и —(**R**)— соответственно (т. е. кнопки К24 и К25 по табл. 3.1). 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. 4.13).

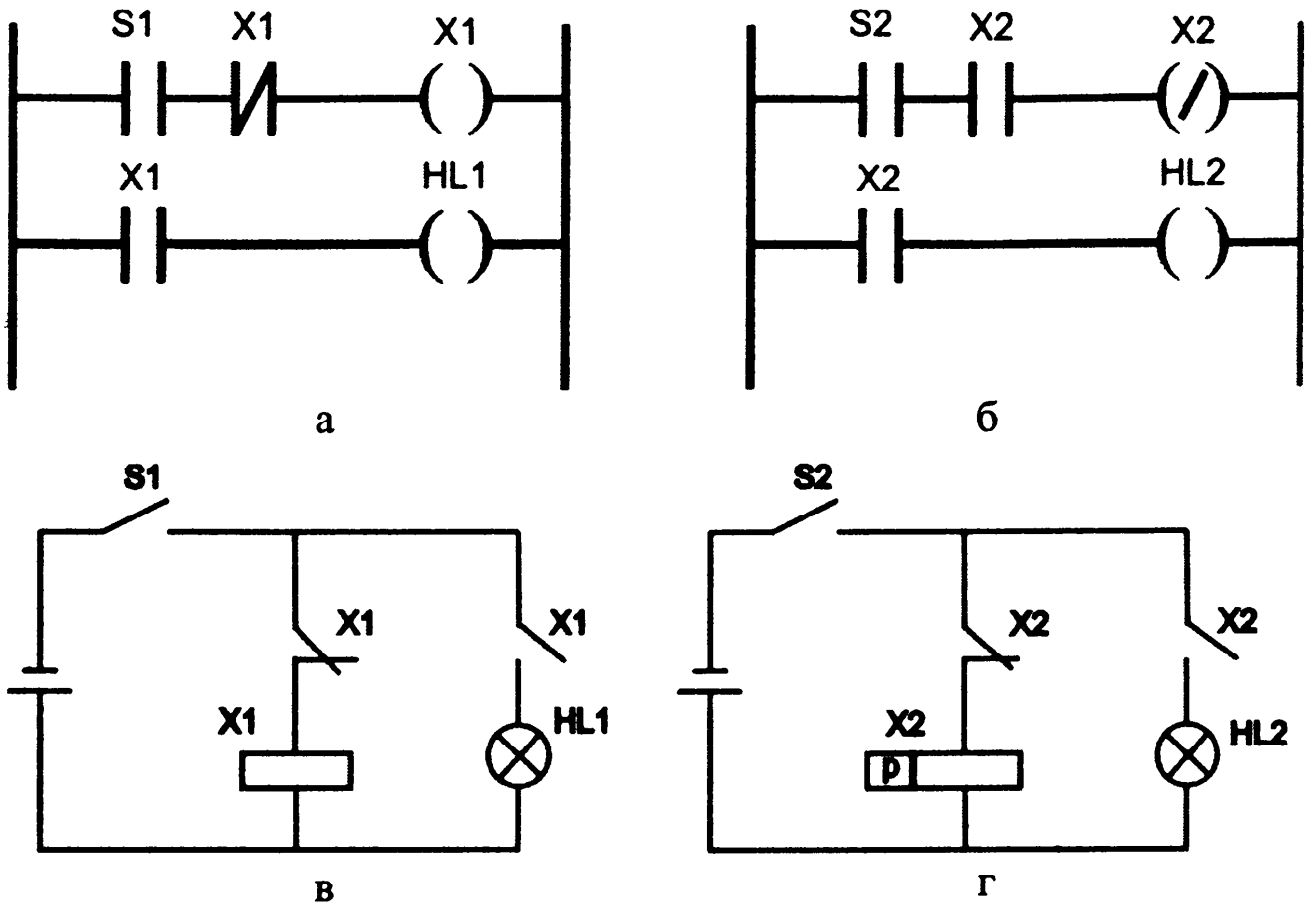


Рис. 4.12. Простейшие генераторы импульсов (а и б) и их релейные аналоги (в и г): S1 и S2 – кнопки пуска; HL1 и HL2 – приемники импульсов

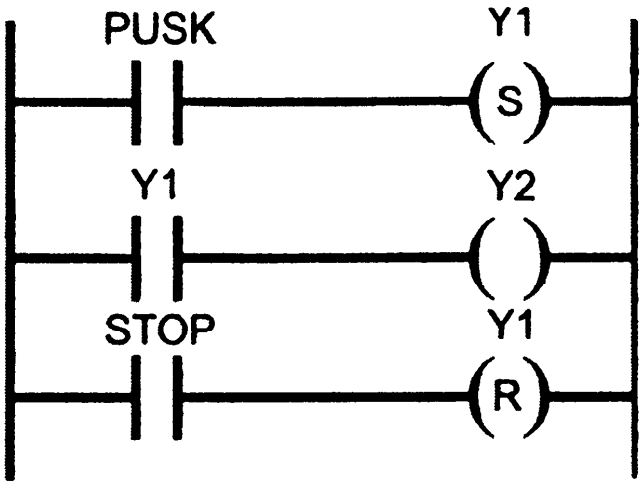


Рис. 4.13. Схема фиксации включения катушки реле Y2 с помощью обмоток S и R

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. 4.14).

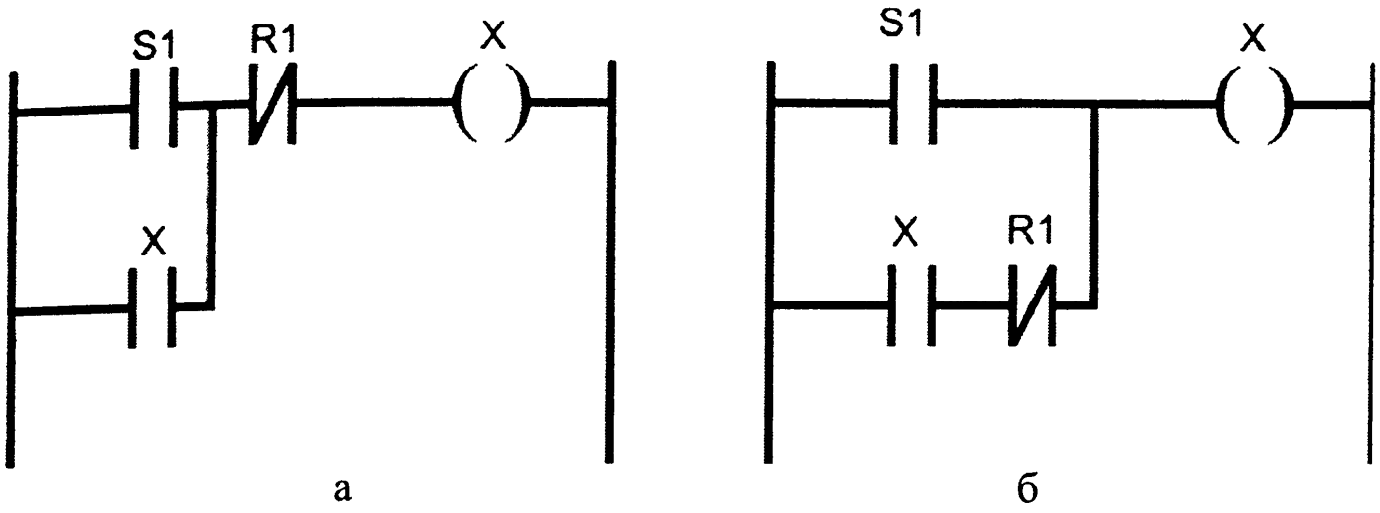


Рис. 4.14. Схемы управления катушкой **X** с самофиксацией состояния

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 (рис. 4.14а) катушка **X** ***отключится*** (если она была включена) или останется не включенной. В схеме (рис. 4.14б) катушка **X** ***включится*** (если она была отключена) или останется включенной.

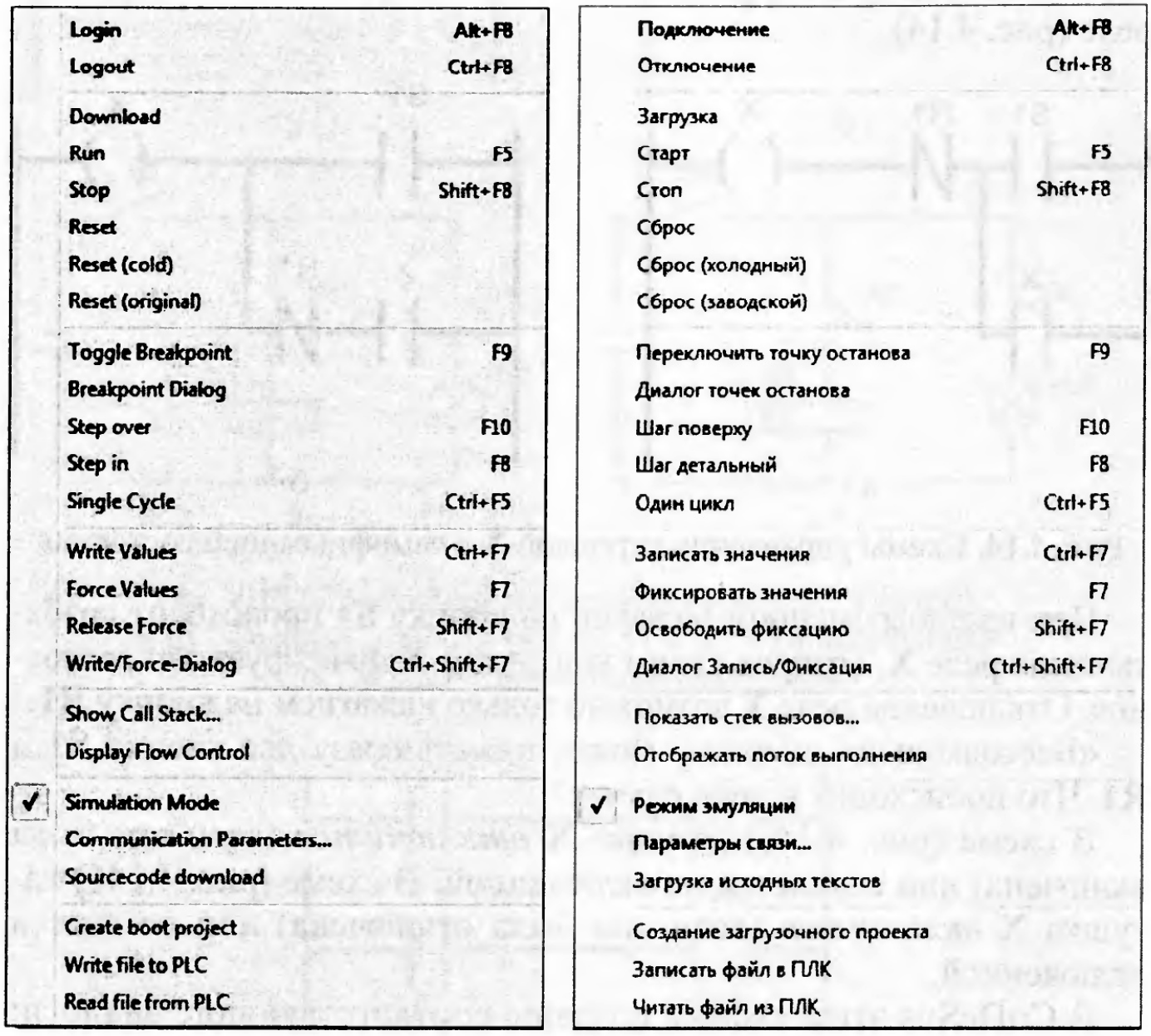
В CoDeSys этим схемам созданы соответствующие аналоги: **RS**-триггер с доминантой выключения – для схемы (рис. 4.14а) и **SR**-триггер с доминантой включения – для схемы (рис. 4.14б), которые будут рассмотрены ниже.

4.6. Исследование СЛУ в режиме эмуляции

Итак, исходную РКС по рисунку 1.2 запрограммировали в CoDeSys на языке LD и представили её в виде многоступенчатой схемы (рис. 4.8). 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 Online (рис. 3.4), щелкаем 1ЛКМ. Открывается меню (рис. 4.15).



а б

Рис. 4.15. Меню Online:
а – в оригинале; б – русскоязычная версия

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

– Снова открываем это меню. Щелкаем 1ЛКМ по строке **Login** (подключение). 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 **Login** откроется со «звоном» окно (рис. 4.16), которое извещает о допущенной ошибке в программе. Возможно, забыли присвоить имя какому-то компоненту или снять ??? на входе того или иного FB (что будет рассмотрено ниже). В любом случае, не внеся требуемые коррективы, двигаться дальше нельзя.

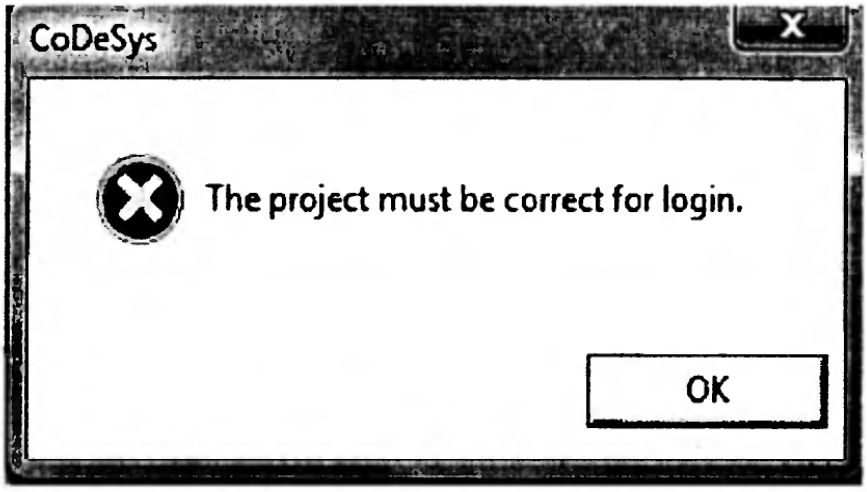


Рис. 4.16. Сигнальное окно

Будем считать, что в нашей схеме нет ошибок (или мы их уже устранили).

– Опять открываем меню **Online** и щелкаем 1ЛКМ по строке **Run** (Старт).

В этом случае катушки, оказавшиеся в проводящей цепи, сработают (станут синими), и возможно некоторые цепи «заработают». Например, «оживут» генераторы по рисунку 4.12, если бы кнопки S1 и S2 были бы изначально представлены размыкающими контактами. В схеме по рисунку 4.8 таких цепей нет.

Схема готова к приему входных сигналов. Можно приступить к её исследованию. В нашей схеме три приемных элемента **SB**, **SL** и **SK** и восемь комбинаций их состояний (табл. 1.2).

– Начнем с первой строки. **A** = 0; **B** = 0, **C** = 0, т. е. **SB**, **SL** и **SK** не сработали. Открываем меню **Online** и щелкаем 1ЛКМ по строке **Write Values** (записать значения). Убеждаемся, что **M** и **HL** не сработали, о чем свидетельствует не изменившийся их вид.

– Переходим ко второй строке таблицы. Должен сработать датчик **SK**. Наводим курсор на контакт **SK**, щелкаем 2ЛКМ. В полости этого контакта появится синий квадратик. Открываем меню **Online** и щелкаем 1ЛКМ по **Write Values**. Контакт **SK** и катушка **C** и вся третья цепь зальются синим цветом. Как и следовало ожидать, **M** и **HL** не сработали.

Далее можно поступать по-разному. Если в таблице 16, 32 или того более различных комбинаций, и чтобы не пропустить ни одной из них для выявления возможности ложных срабатываний исполнительных элементов, следует двигаться строго по таблице состояний.

Для этого необходимо «разомкнуть» контакт **SK** и «замкнуть» **SL**, т. е. создать комбинацию приемных элементов по третьей строке таблицы 1.2. Для этого наводим курсор на **SK**, щелкаем 1ЛКМ. В полости контакта появится зеленый квадратик и останется часть синей заливки. Наводим курсор на **SL**, щелкаем 1ЛКМ. Контакт заливается синим цветом. Открываем меню **Online**, щелкаем 1ЛКМ по **Write Values**. Убеждаемся, что не сработал **M**, и включилась катушка **HL**. И так проходим последовательно по всем строкам таблицы состояний.

В случае исследования такой простой СЛУ (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.

Можно также, приступая к исследованию реакции СЛУ на следующую комбинацию состояний приемных элементов, вернуть схему в исходное состояние (**A** = 0, **B** = 0 и **C** = 0).

Для этого достаточно щелкнуть 1ЛКМ по К9(**Logout**), т. е. отключить режим **Online**. Затем, по освоенной уже методике, запустить режим эмуляции, создать требуемую комбинацию состояний приемных элементов и, нажав 1ЛКМ по **Write Values**, 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 **Write Values** можно, пользуясь прокрутками, переместить интересующий нас участок схемы в поле зрения и лишь потом запустить ее.

4.7. Триггеры

4.7.1. R_TRIG и **F_TRIG** или детекторы импульсов отличаются лишь реакцией на изменение входного сигнала. Первый из них формирует на выходе **Q** одиночный импульс по ***переднему*** фронту сигнала, поступающего на его вход **CLK**, а другой – по ***заднему*** фронту или спаду (рис. 4.17).

Самый простой способ переноса в цепь этих FB – щелкнуть 1ЛКМ по К28 или К29. Появившиеся ??? над этими блоками запрашивают присвоение идентификатора (п. 4.2).

Поближе с этим FB познакомимся после освоения работы с таймерами.

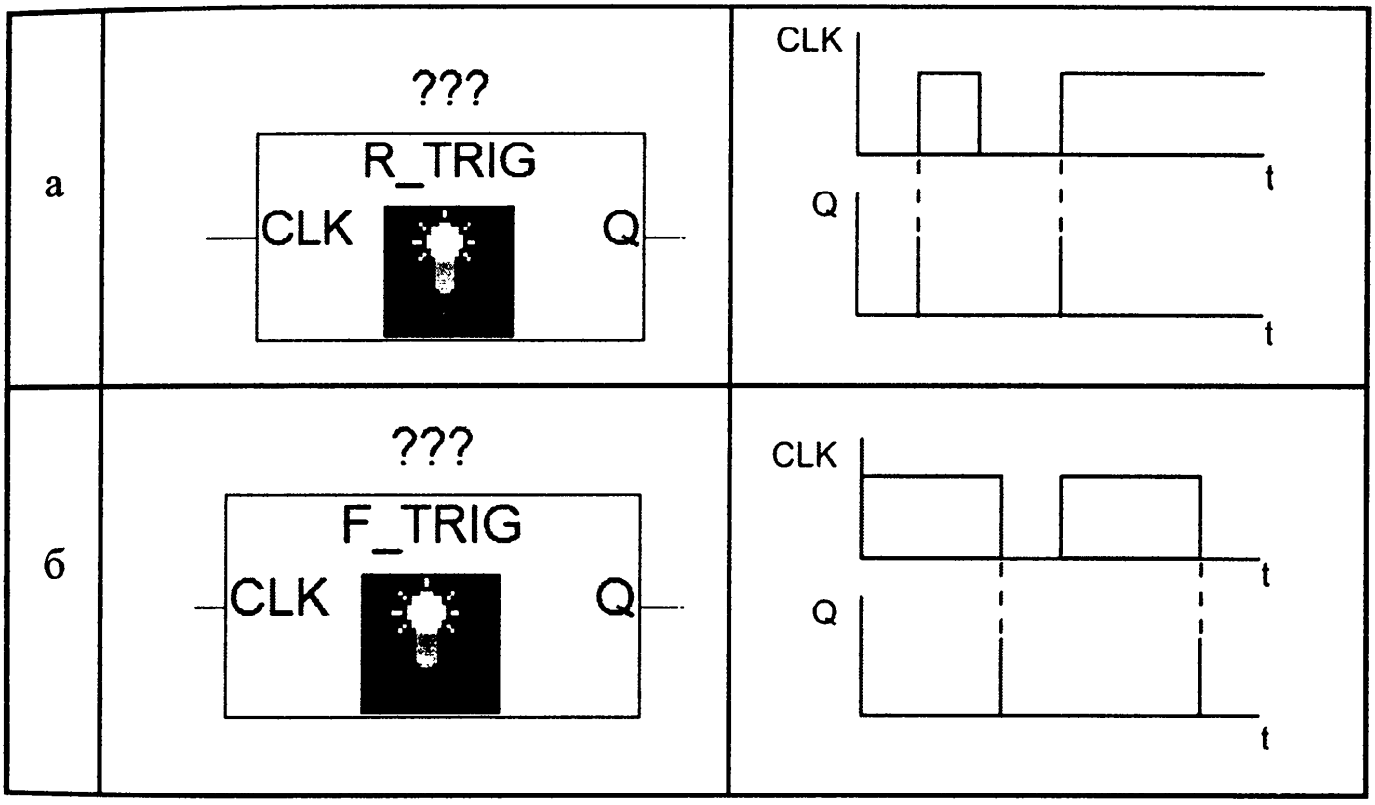


Рис. 4.17. Изображение детекторов импульсов
и их временные диаграммы:
а – для **R_TRIG**; б – **F_TRIG**

4.7.2. RS- и **SR**-триггеры, как уже отмечалось в п. 4.5, поразному реагируют на одновременное поступление сигналов на оба входа: **SET** и **RESET**. Напоминаем, что для классического RS-триггера, применяемого в электронике, такое состояние входов считается запрещенным, т. к. приводит к неоднозначности выходного сигнала. Временные диаграммы этих FB изображены на рисунке 4.18.

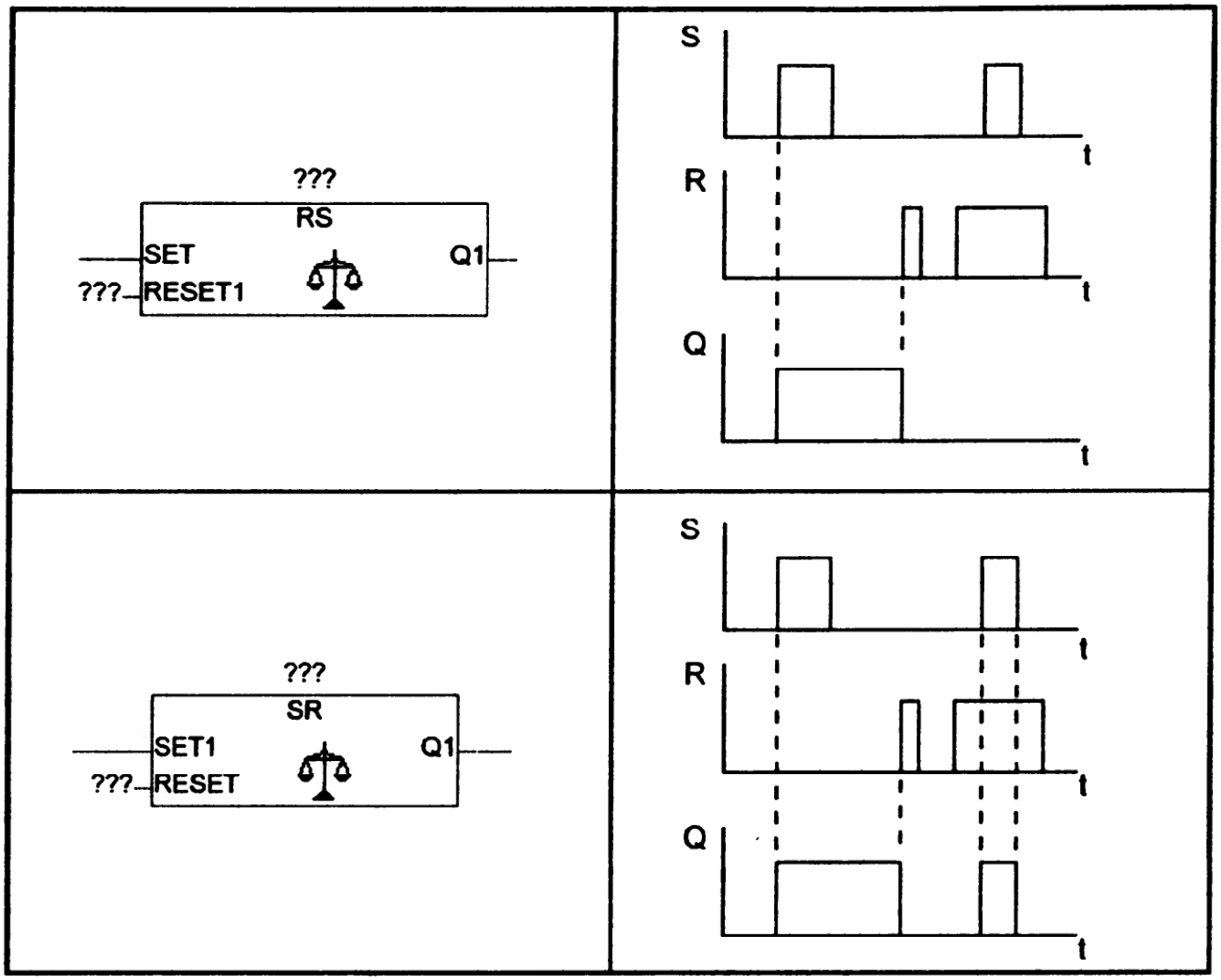


Рис. 4.18. Изображение триггеров и их временные диаграммы

Для включения в цепь LD этих FB необходимо щелкнуть 1ЛКМ по К26. Откроется окно **Input Assistant** (рис. 4.19).

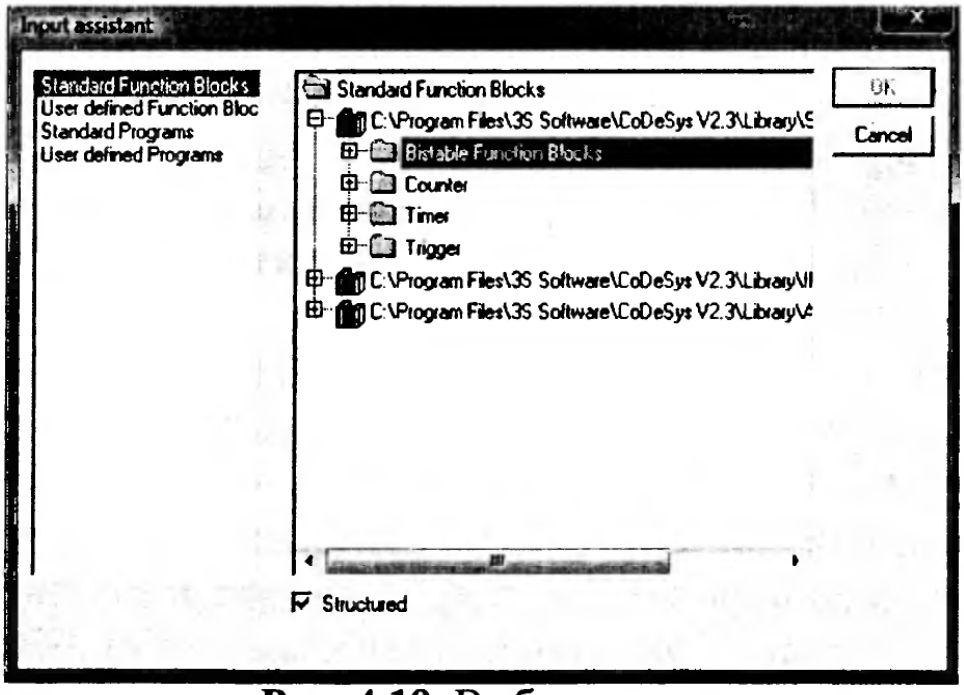


Рис. 4.19. Выбор папки

Подводим курсор к кнопке **Bistable Function Blocks**, щелкаем 1ЛКМ. Открывается папка с FB (рис. 4.20).

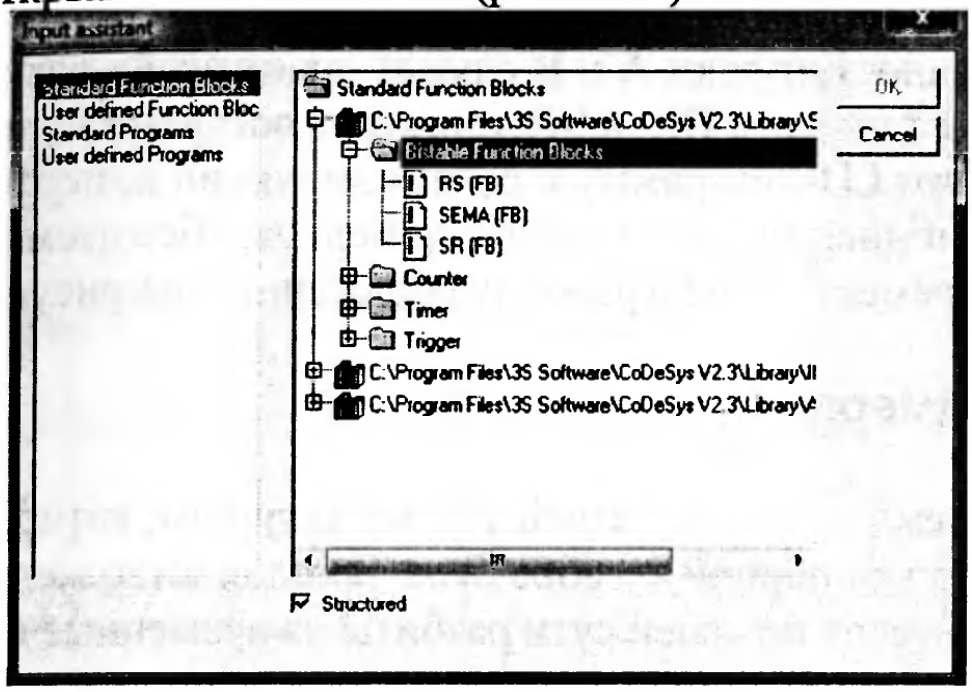


Рис. 4.20. Выбор триггера

Выбираем нужный триггер, например, **RS**. Наводим курсор на строку с его названием, щелкаем 1ЛКМ. Переводим курсор на ОК, щелкаем 1ЛКМ. **RS**-триггер появляется в цепи. Помимо ??? над триггером, запрашивающих как обычно имя этого FB, возникли ??? на входе **RESET 1**.

Эти ??? по освоенной уже методике следует заменить именем того реле, которое будет производить сброс, т. е. своим замыкающим контактом подавать логическую 1 на вход **RESET 1**.

Сам этот контакт в LD-диаграмме не изображается.

Рекомендуем собрать схему с этими триггерами (рис. 4.21).

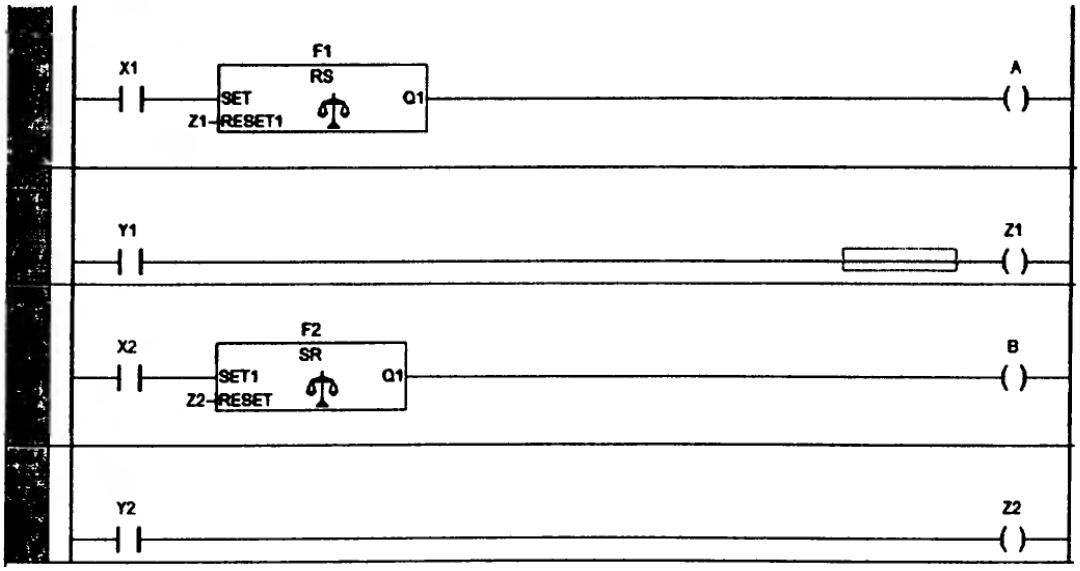


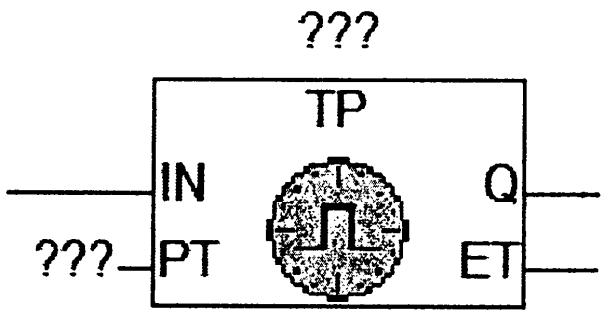
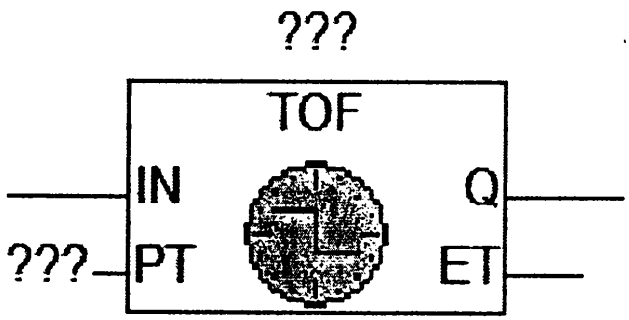
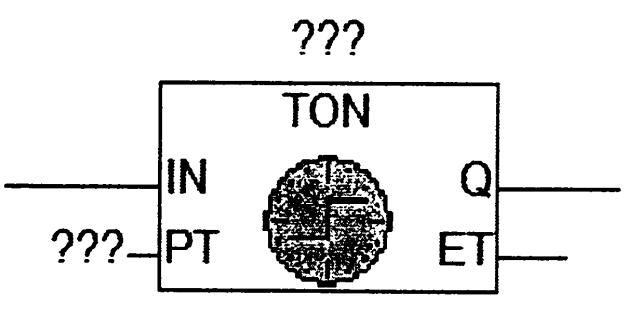
Рис. 4.21. Многоступенчатая LD-диаграмма для исследования **RS**- и **SR**-триггеров

В этой схеме **X1** и **X2** – кнопки запуска триггеров; **Y1** и **Y2** – кнопки сброса, управляющие катушками **Z1** и **Z2**, замыкающие контакты которых (на схеме их нет!) обслуживают входы **RESET 1** и **RESET** соответственно. Катушки **A** и **B** служат «нагрузкой» для этих триггеров. **F1** и **F2** – имена **RS-** и **SR**-триггеров соответственно.

Переводим LD-диаграмму в режим эмуляции и, перебирая возможные комбинации для входных элементов, убеждаемся в правомочности временных диаграмм, изображенных на рисунке 4.18.

4.8. Таймеры

4.8.1. Время – это тот самый третий аргумент, которому не нашлось места в бинарной алгебре Буля. Большинство же технологических процессов по своей сути разбиты на временные интервалы, чередующиеся ***события***, что и легло в основу самого определения событийно управляемой логики. Для формирования этих временных интервалов и фиксации событий в CoDeSys применяются в основном три типа таймеров: **TP**, **TOF** и **TON**.

	**TP**-таймер или генератор **одиночного импульса** с заданной по входу **PT** длительностью
	**TOF**-таймер с задержкой **выключения**
	**TON**-таймер с задержкой **включения**

У этих таймеров есть вход **IN** для логических сигналов, вход **PT** для установки требуемых параметров, логический выход **Q** и выход **ET** (в WORDe).

Работу этих таймеров поясняют временные диаграммы. Покажем лишь временные диаграммы входных и выходных сигналов (рис. 4.22).

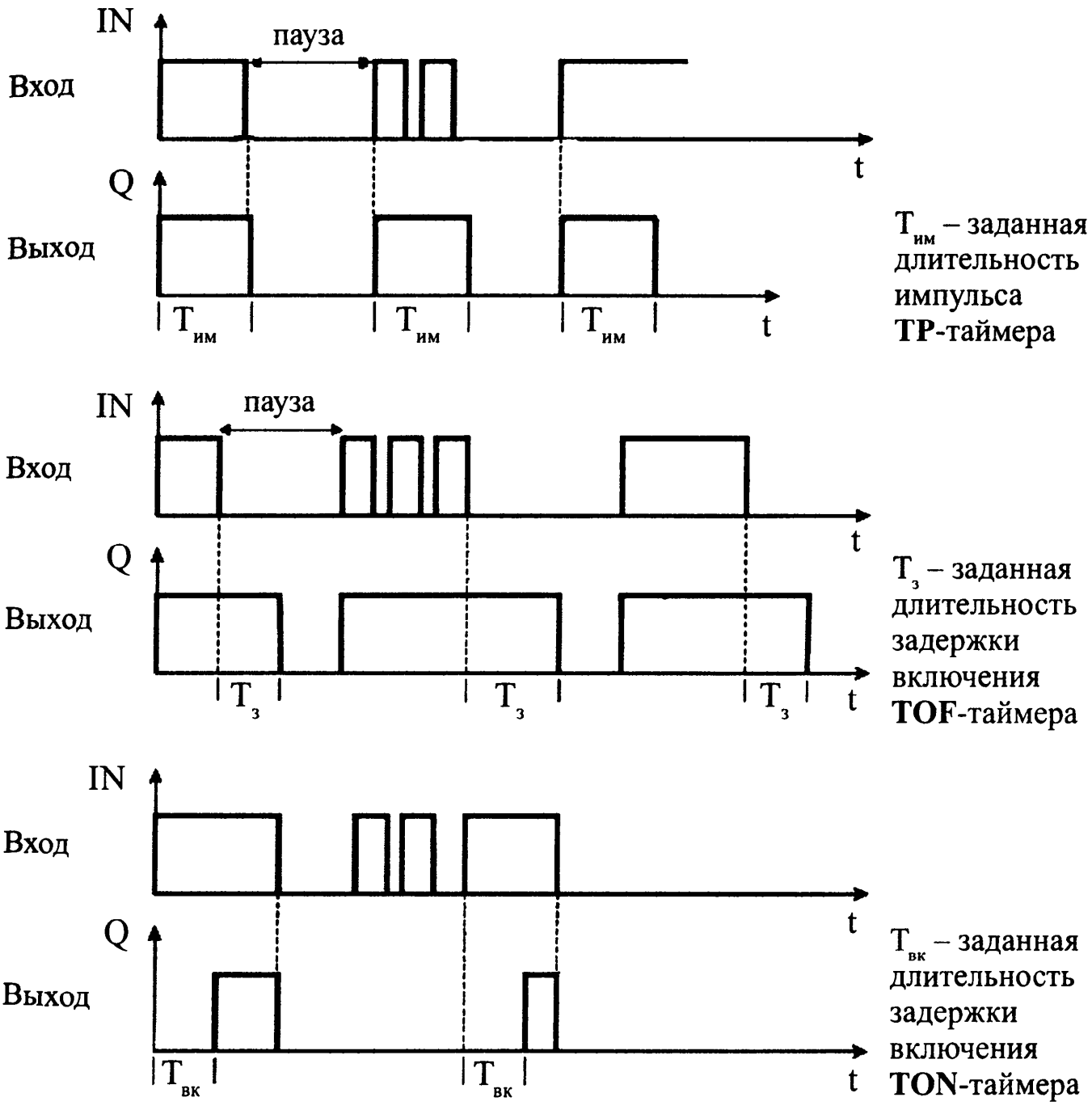


Рис. 4.22. Временные диаграммы таймеров

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

TOF-таймер также срабатывает по фронту входа **IN**. Выход **Q** сбрасывается после спада входного сигнала с задержкой времени $T_з$, установленной по входу **PT**. ***Пауза*** между входными сигналами должна быть не меньше времени задержки.

TON-таймер срабатывает по переднему фронту входа IN, но сигнал на выходе **Q** появится с задержкой $T_{вк}$, установленной по входу **PT**.

Таймер не реагирует на импульсы продолжительностью менее значения $T_{вк}$.

Для включения в цепь этих таймеров можно щелкнуть 1ЛКМ по К26. Откроется окно (рис. 4.19). Подводим курсор к кнопке Timer, щелкаем 1ЛКМ. Открывается папка с перечислением FB. Подводим курсор к строке с нужным таймером; щелкаем 1ЛКМ. Затем направляем курсор на ОК и нажимаем 1ЛКМ. Таймер появился в цепи. **TON**-таймер можно сразу включить в цепь, щелкнув 1ЛКМ по К30.

При включении любого таймера в цепь многоступенчатой схемы программа запрашивает кроме имени этого FB (вопросительные знаки над таймером) временную уставку по входу **PT** (вопросительные знаки у входа **PT**).

Щелкнув 1ЛКМ по верхним ???, присваиваем имя. Например **N1**. Щелкнув по ??? у входа **PT**, нажав и не отпуская клавишу **Shift**, нажать T, затем #. Отпустить **Shift**, набрать требуемое значение задержки (например, 15), единицу времени (например, S – т. е. секунды) и **Enter**. Фрагмент схемы будет выглядеть, как показано на рис. 4.23.

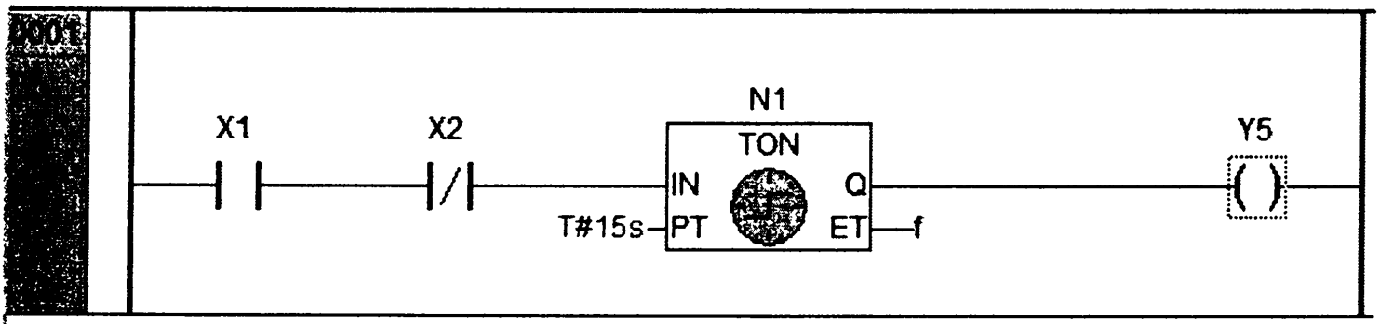


Рис. 4.23. Фрагмент схемы с **TON**-таймером после снятия ???

Временные уставки задают в миллисекундах (mS), секундах (S), минутах (m) или часах (h).

На выходе **ET** можно получать информацию о текущем значении уставки **PT** с момента срабатывания таймера. С этой целью

при написании программы наводим курсор на выход **ET**, щелкаем 1ЛКМ. Появляется мерцающий курсор. Вбиваем идентификатор этого выхода, например, **f** (рис. 4.23). Нажимаем **Enter**. Открывается окно объявления переменной (рис. 4.2). Подводим курсор к кнопке (в конце строки с типом переменной BOOL), щелкаем 1ЛКМ. Открывается окно **Input assistant** (рис. 4.24). Щелкаем 1ЛКМ по TIME, потом OK.

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

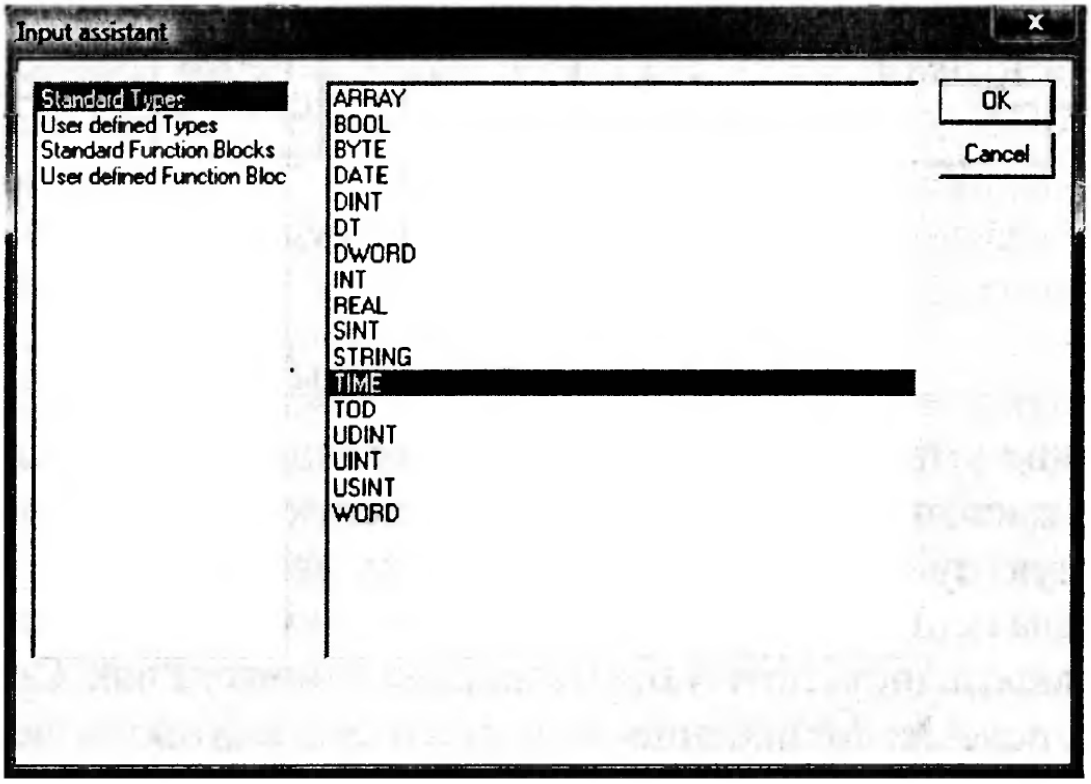


Рис. 4.24. Окно **Input assistant**

Наличие таймера не вызывает задержки в прогоне программы.

4.8.2. Сделаем небольшое, но весьма важное отступление от хода нашего изложения. Тем более, что полученные сведения уже позволяют экспериментально проверить достоверность нижеследующего положения.

Как отмечалось в п. 4.1, есть кажущаяся схожесть РКС- и LD-диаграмм. Однако есть принципиальное отличие: в РКС параллельные цепи получают питание одновременно, а в LD «чтение», т. е. сканирование многоступенчатой схемы выполняется последовательно слева направо и сверху вниз. Поэтому, если какое-то реле в цепи изменит свое состояние, то цепи расположенные ***ниже*** получат новое значение переменной ***сразу***, а цепи, находящиеся ***выше***, – только в следующем цикле прогона программы.

Соберем схемы по рисунку 4.25.

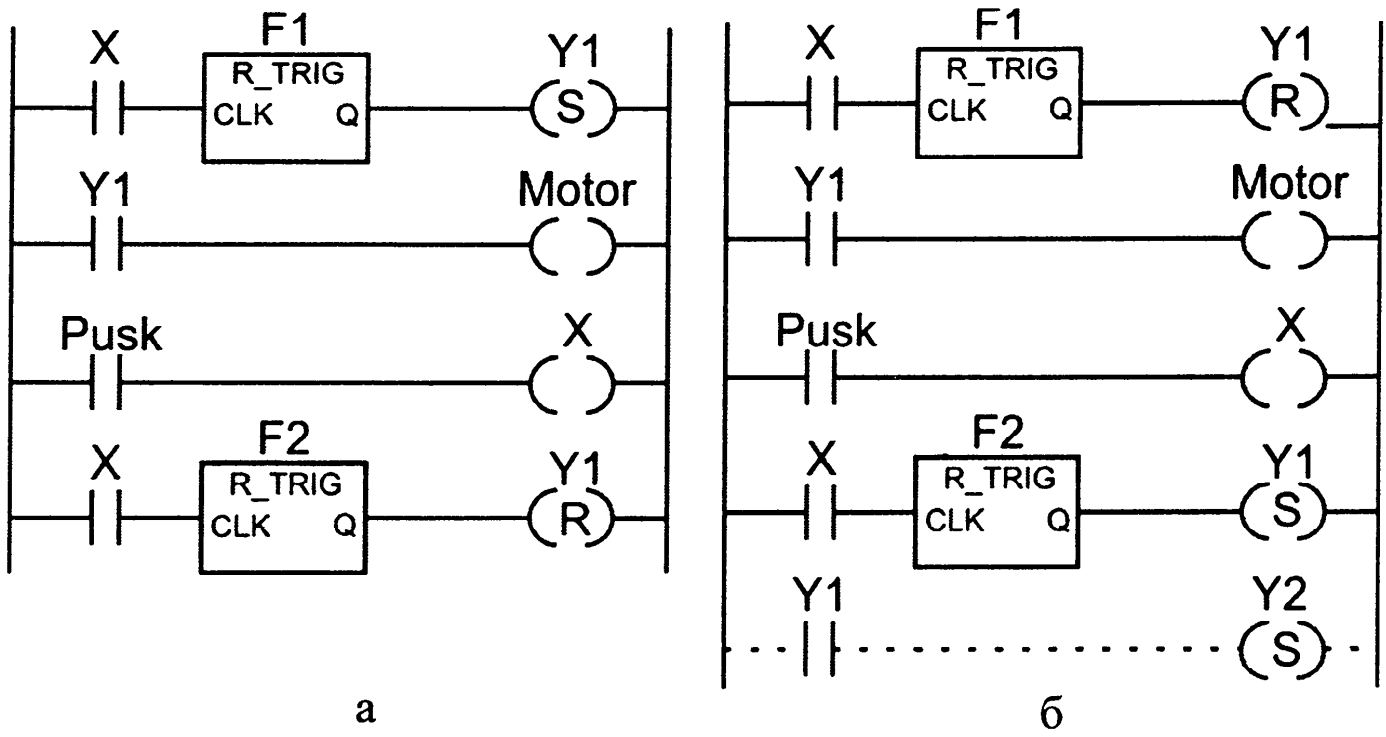


Рис. 4.25. Опытные схемы

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

Сначала испытаем в режиме эмуляции схему по 4.25а, соблюдая последовательность по п. 4.6, «Нажимаем» кнопку **Pusk**. Сработала катушка реле **X**. Мгновенно замкнулся его контакт в четвертой цепи. Через детектор переднего фронта **F2** пришел очень короткий импульс на **R**-катушку реле **Y1**. В следующем цикле замкнулся контакт **X** в первой цепи и через **F1** вызвал срабатывание катушки **S** реле **Y1**. Реле сработало и своим контактом во второй цепи включило катушку воображаемой нагрузки **Motor**.

Повторяем опыт с другой схемой (без пятой цепи). После команды **Pusk** сработало только реле **X**!

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

*В том, что срабатывание все-таки имело место легко убедиться, добавив пятую цепь с еще одной катушкой **S**'. (Но с другим, разумеется, именем. Например, **Y2**). Вторую катушку **R** этого реле не используем. Просто для возврата схемы в исходное состояние можно воспользоваться кнопкой К9 (табл. 3.1).*

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

Замыкающий контакт **Y1** во второй цепи останется разомкнутым, и катушка реле **Motor**, имитирующего исполнительный элемент какой-то СЛУ, не сработает.

Схема 4.25б позволяет поставить еще один интересный опыт. Для этого снова запрограммируем LD-диаграмму, заменив **R_TRIG F1** и **F2** на **TP**-таймеры. Сначала временные уставки по входам **PT** этих FB сделать равными. Например, по $T_1 = T_2$=50 мс.

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

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

Далее начинаем уменьшать на 5–10 мс уставку по входу **PT** таймера в первой цепи (или увеличивать с таким же интервалом уставку второго **TP**-таймера). После каждой коррекции повторяем **Pusk**. Опыт продолжаем до тех пор, пока не сработают все цепи. Что же произошло в этом случае?

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

После команды **Pusk** срабатывает катушка реле X (третья цепь) и своим контактом X в четвертой цепи запускает **TP**-таймер, который в течение t_2 мс будет держать «под напряжением» катушку **S** реле **Y1**.

Цикл сканирования на этом заканчивается, и начинается новый прогон программы. Срабатывает контакт **X** в первой цепи, запускается **TP**-таймер, который в течение T_2 мс будет держать «под напряжением» катушку сброса **R** реле **Y1**.

Если период T_2 еще не закончился, то наступает пауза запрещенного состояния для **Y1**: на его катушки **S** и **R** поступают сигналы. Но если эта пауза не превышает продолжительность цикла, то заметить на мониторе некорректность предложенной программы нельзя. (Мы не ставим задачу получить красивую программу). Зато этот пример позволяет оценить продолжительность прогона программы, т. к. минимальная разность $T_2 - T_1$, при которой прои-

зошло срабатывание катушки **Motor**, будет равна удвоенному интервалу циклу сканирования (но только не самого ПЛК, а Windows эмулятора, привязанного к тактам системного таймера, используемого в работе с ПК).

4.8.3. Продолжим исследование таймеров.

Соберем схему генератора с регулируемой длительностью импульса и паузы, как показано на рисунке 4.26 [3].

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

Первая цепь предназначена для пуска и остановки генератора.

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

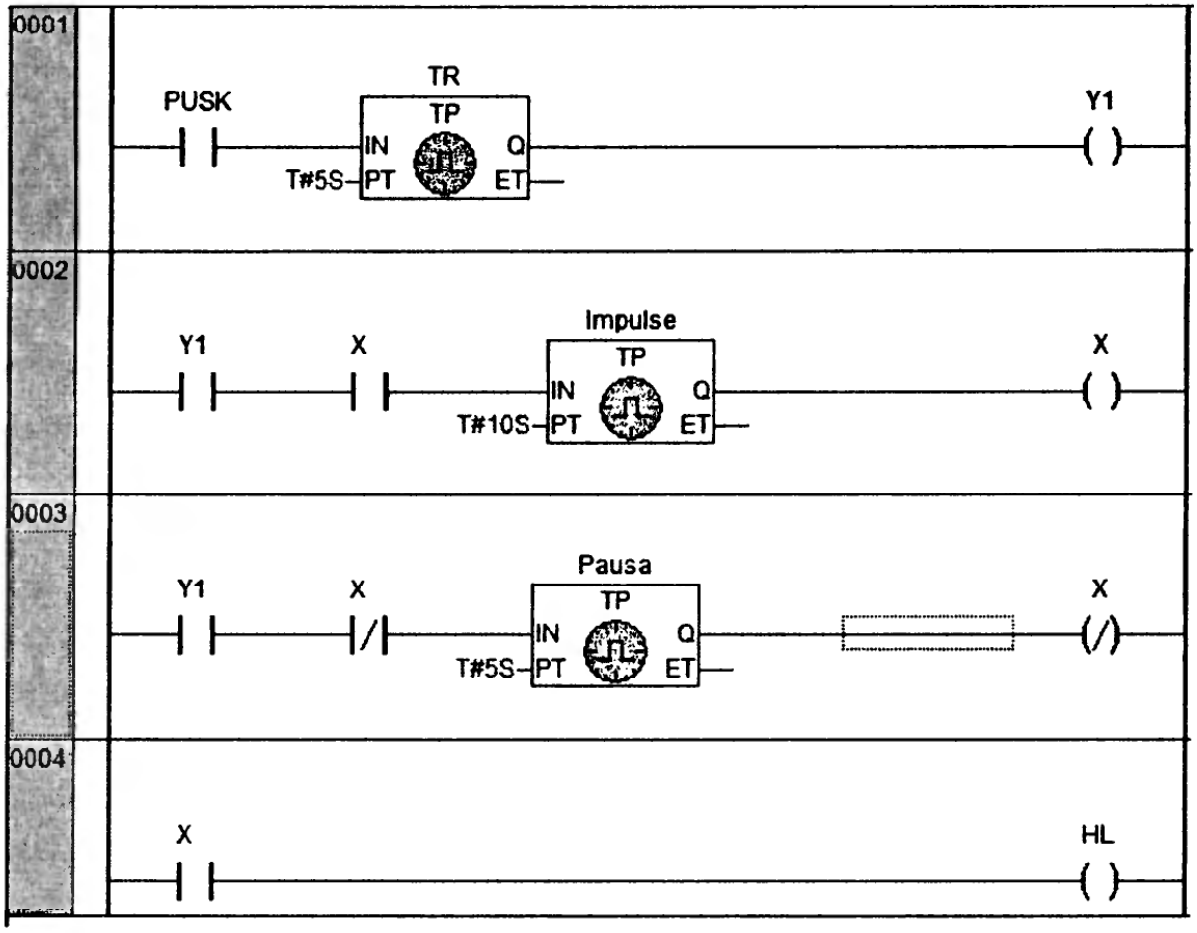


Рис. 4.26. Генератор импульсов

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

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

Теперь можно собрать схему для исследования таймеров (рис. 4.27).

Первая цепь служит для запуска уже известного нам генератора, занимающего 2-ю и 3-ю 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**. Временные параметры генератора (**T1** – длительность импульса, **T2** – паузы) и исследуемых таймеров (**T3**, **T4**, **T5**) нужно будет менять.

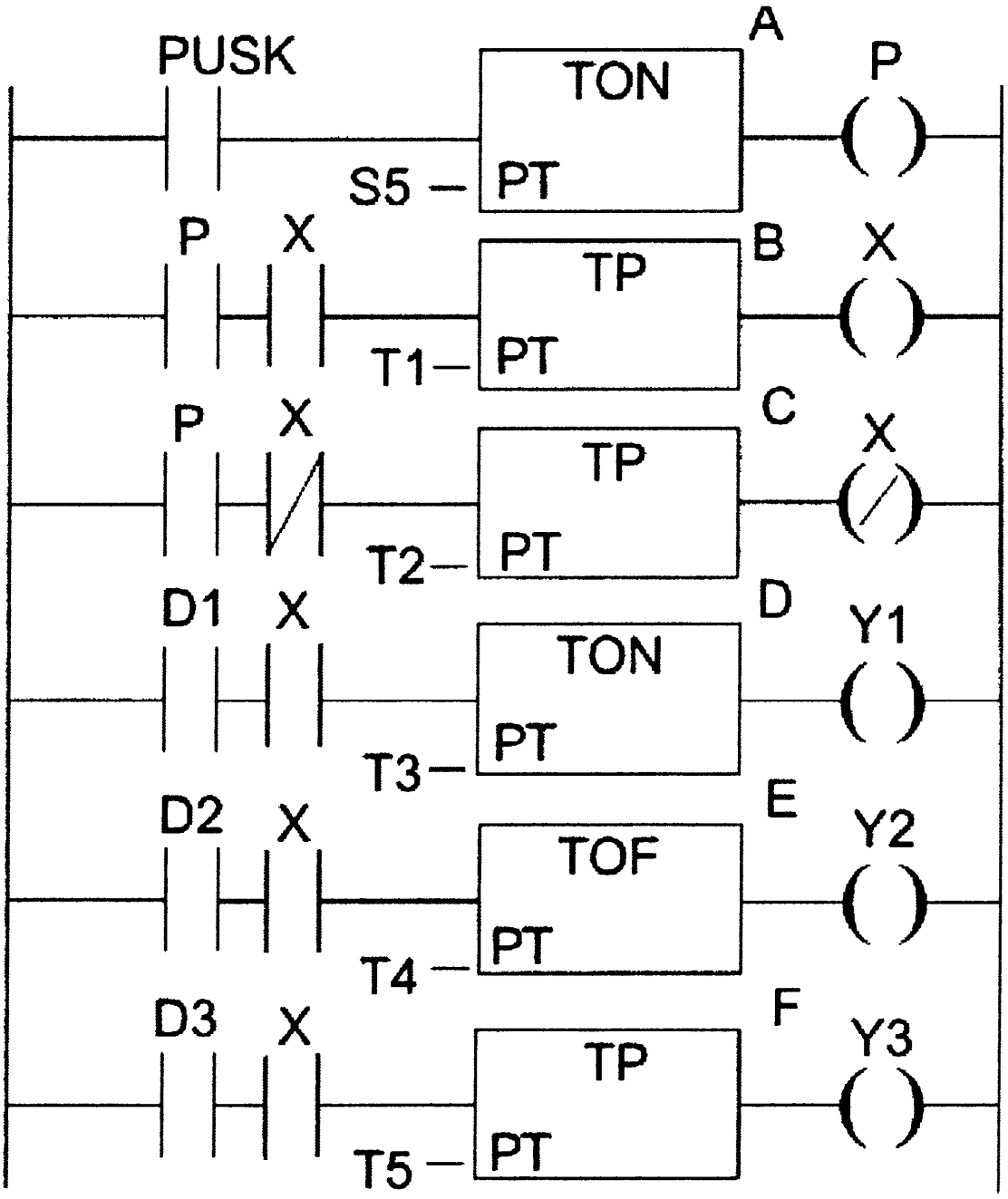


Рис. 4.27. Схема исследования таймеров

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

Например, установим **T1** = 5 с, **T2** = 3 с, **T3** = 8 с. Включаем **PUSK** и кнопкой **D1** таймер **TON**. Таймер не сработает, т. к. **T1** < **T3** (рис. 4.22).

Устанавливаем **T1** = 8 с, **T2** = 3 с и **T3** = 8 с. Таймер также не сработает, т. к. **T1** = **T3**. Новая установка: **T1** = 10 с; **T2** = 3 с; **T3** = 8 с. Таймер **TON** сработает, т. к. **T1**>**T3**, и **Y1** начнет «мигать» синим цветом, что можно еще раз показать на временной диаграмме (рис. 4.28).

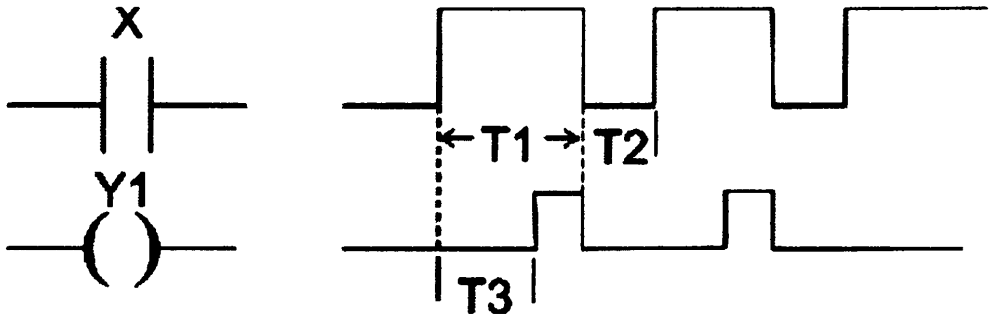


Рис. 4.28. Временные диаграммы работы **TON**-таймера D

Подобные опыты поставить для **TOF** и **TP** и убедиться в справедливости выводов в п. 4.8.1 или рисунке 4.22.

4.8.4 Схема с применением всех таймеров и детекторов импульсов.

Завершая этот раздел, рекомендуем собрать схему (рис. 4.29) 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 рисунке 4.29 пришлось сделать «перенос» цепи).

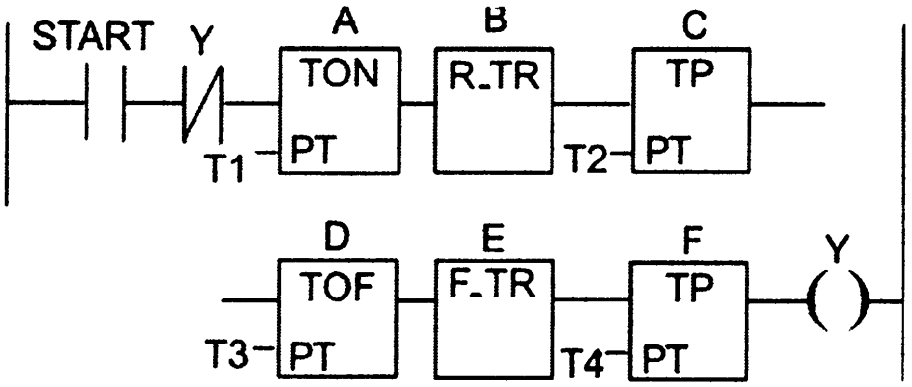


Рис. 4.29. Учебная схема

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 со-

отношения значений **T2** и **T3** на поведение системы в целом, воспользовавшись рисунком 4.30.

Напоминаем, что наличие сигнала на выходе какого-либо элемента подтверждается синим цветом канала, соединяющего этот выход с входом следующего блока. Пользуясь горизонтальной прокруткой, можно перемещать в поле зрения оператора интересующий его участок цепи.

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

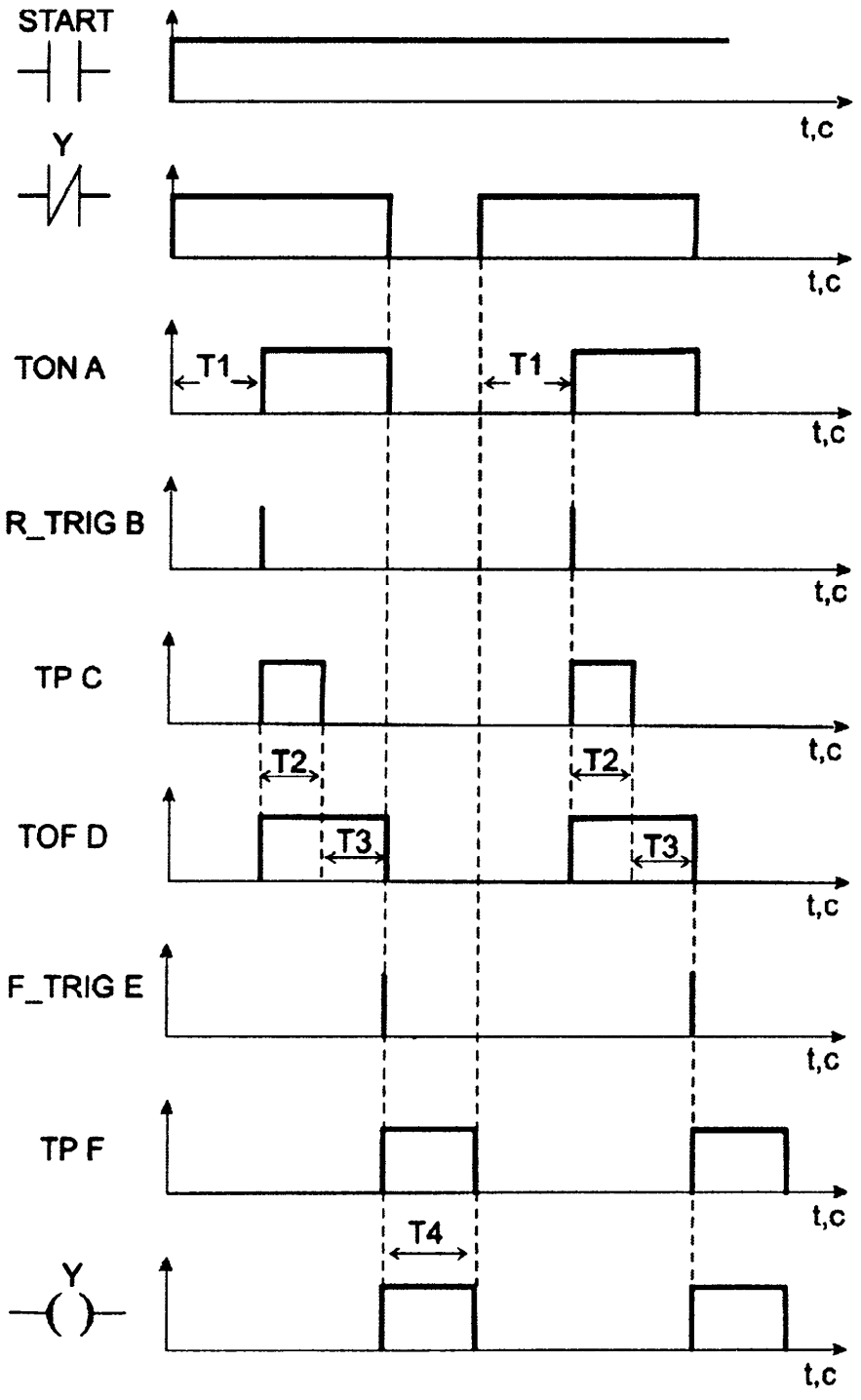


Рис. 4.30. Временные диаграммы состояний элементов учебной схемы

4.9. Счетчики

Можно привести немало технологических процессов, для управления которыми необходимо вести подсчет чего-либо. Например, бутылок с минеральной водой, перемещаемых конвейером для последующей упаковки в ящик. Возможно, при этом надо с помощью какого-то датчика выявлять в потоке некондиционные бутылки, отправлять их в сторону и тоже вести их учет. Для решения подобных задач в ПЛК применяются различные виды счетчиков.

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

4.9.1. Соберем простую схему с **CTU**-счетчиком (рис. 4.31).

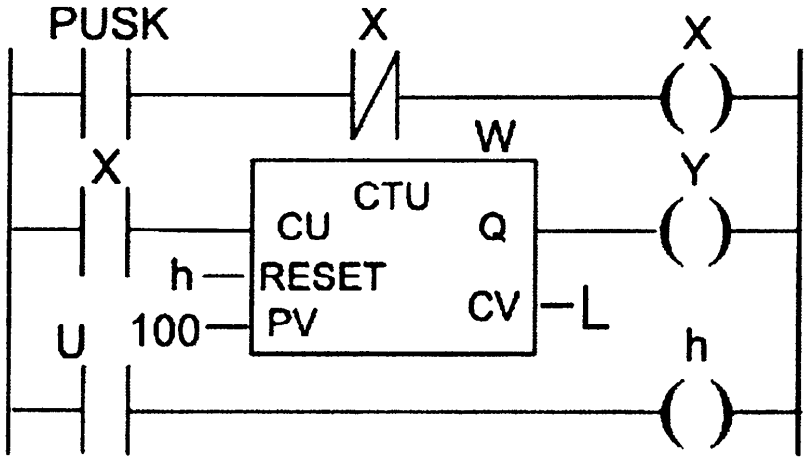


Рис. 4.31. Схема с **CTU**-счетчиком

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

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

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

С выхода **CV** можно в WORDe снимать значение накопленных сигналов. Для этого по полной аналогии с приемами активации выхода **ET** у таймеров (п. 4.8.1.) активируем выход **CV**, присваиваем имя и т. д. Но в окне **Input assistant** (рис. 4.24) выбираем переменную WORD.

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

В схеме (рис. 4.31) счетчику присвоено имя **W**, входу **RESET** – имя реле **h**, выходу **CV** – **L** 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 **X** (рис. 4.12а).

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

Кстати, эта схема (рис. 4.31) также позволяет оценить время прогона программы. Достаточно секундомером замерить время от момента пуска генератора до момента срабатывания реле **Y**. Период импульсов генератора равен удвоенной длительности рабочего цикла. Количество же циклов известно и равно уставке **PV**. Вместо секундомера можно воспользоваться любым таймером с активированным выходом **ET** и уставкой **PT**, заведомо превышающей предполагаемое время накопления импульсов, количество которых задано по входу **PV**. Для этого следует создать дополнительную четвертую цепь (рис. 4.32) и в момент срабатывания реле **Y** снять показания на выходе **ET**. (Катушка не потребовалась).

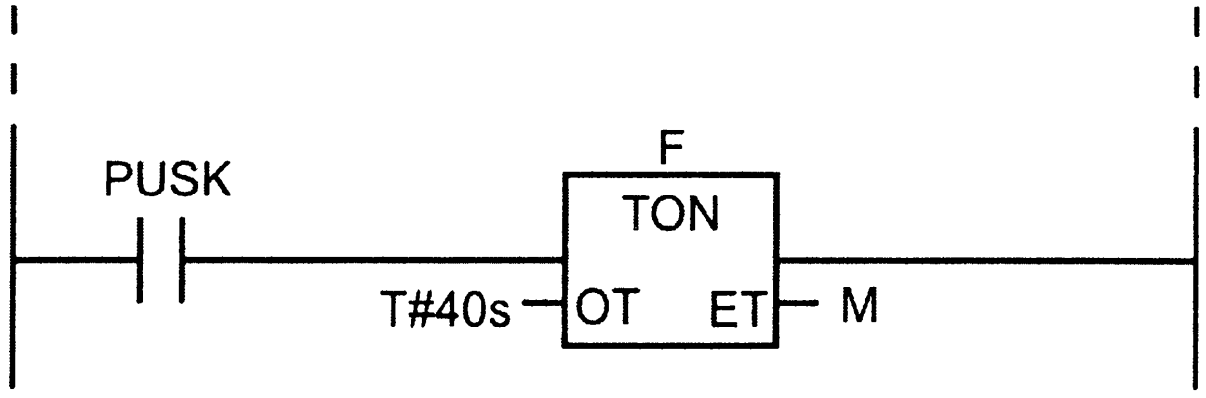


Рис. 4.32 Дополнительная цепь к схеме по рисунку 4.31

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

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

Можно собрать схему с **CTD**-счетчиком (рис. 4.33).

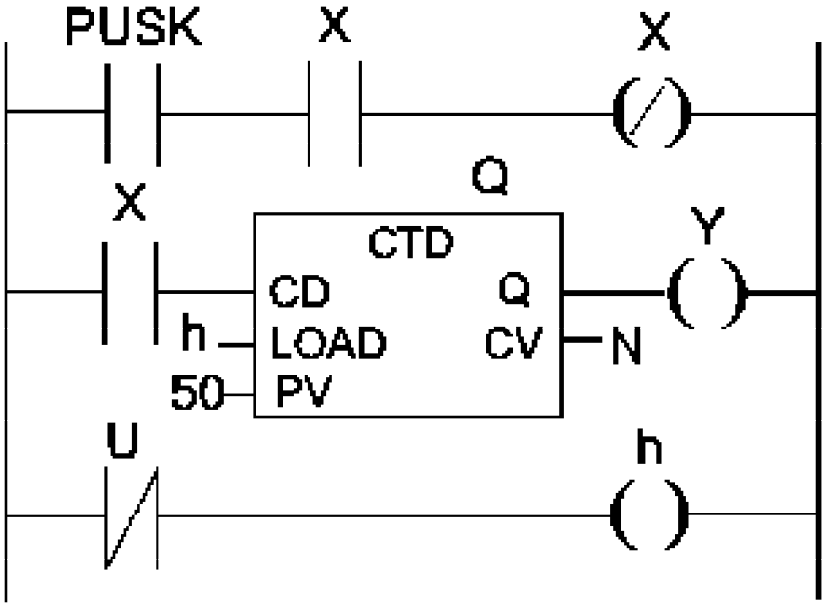


Рис. 4.33. Схема с **CTD**-счетчиком

Кнопка **PUSK** запускает генератор **X** (для разнообразия собран по схеме, отображенной на рисунке 4.12б). Счетчику присвоили имя **Q**, входу **LOAD – h**, входу **CV-N**.

По умолчанию принято, что реле, управляющие счетчиками **CTU** и **CTD** имеют на входах **RESET** и **LOAD** соответственно **замыкающий** контакт. В схемах (рис. 4.30 и 4.33) это реле **h**. Но, учитывая отличие счетчика **CTD**, лучше в схеме (рис. 4.33) установить **размыкающую** кнопку **U**. Тогда при включении схемы сразу сработает реле **h** и подаст сигнал TRUE на вход **LOAD**, что обеспечит загрузку выхода **CV** начальным значением **PV** (в 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**.

Для простых логических систем применение рассмотренных счетчиков практически равноценно. Дело вкуса проектировщика в

выборе того или иного FB. Кроме того, после срабатывания **CTU**-счетчика (**Q** = 1) счет импульсов по входу **CU** будет продолжаться, и это можно проконтролировать по выходу **CV**. В счетчике **CTD** после обнуления выхода **CV** все поступающие на вход **CD** сигналы будут потеряны.

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

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

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

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

Рекомендуем собрать более сложную схему для ознакомления со счетчиком **CTUD** (рис. 4.34).

Схема по рисунку 4.34 практически не нуждается в комментариях, т. к. они сделаны на русском языке в поле каждой цепи. (п. 4.2.2).

Тем не менее уточним некоторые положения.

Первые две цепи служат для формирования импульсов как для суммирующего входа **CU** с помощью кнопки **PLUS** и контакта **X1**, так и на вычитающем входе **CD** с помощью кнопки **MINUS** и контакта **X2**, «присутствие» которого в схеме обозначено лишь его именем.

Некоторое усложнение этих цепей будет оправдано упрощением процедуры генерирования одиночных импульсов для входов **CU** и/или **CD** (разумеется, при испытаниях в режиме эмуляции). Действительно, катушки реле **X1** и **X2** будут кратковременно срабатывать как при «замыкании» кнопок **PLUS** или **MINUS** (за счет **R_TRIG F1** или **F3**), так и при их размыкании (благодаря **F_TRIG F2** или **F4**).

Идентификация бинарного выхода **QD** выполняется по той же методике, что и присвоение имен входам **CD**, **RESET** и **LOAD**.

Значение уставки **PV** намеренно взято небольшим, чтобы не утруждать исследователя многократными операциями с кнопками **PLUS** и **MINUS**.

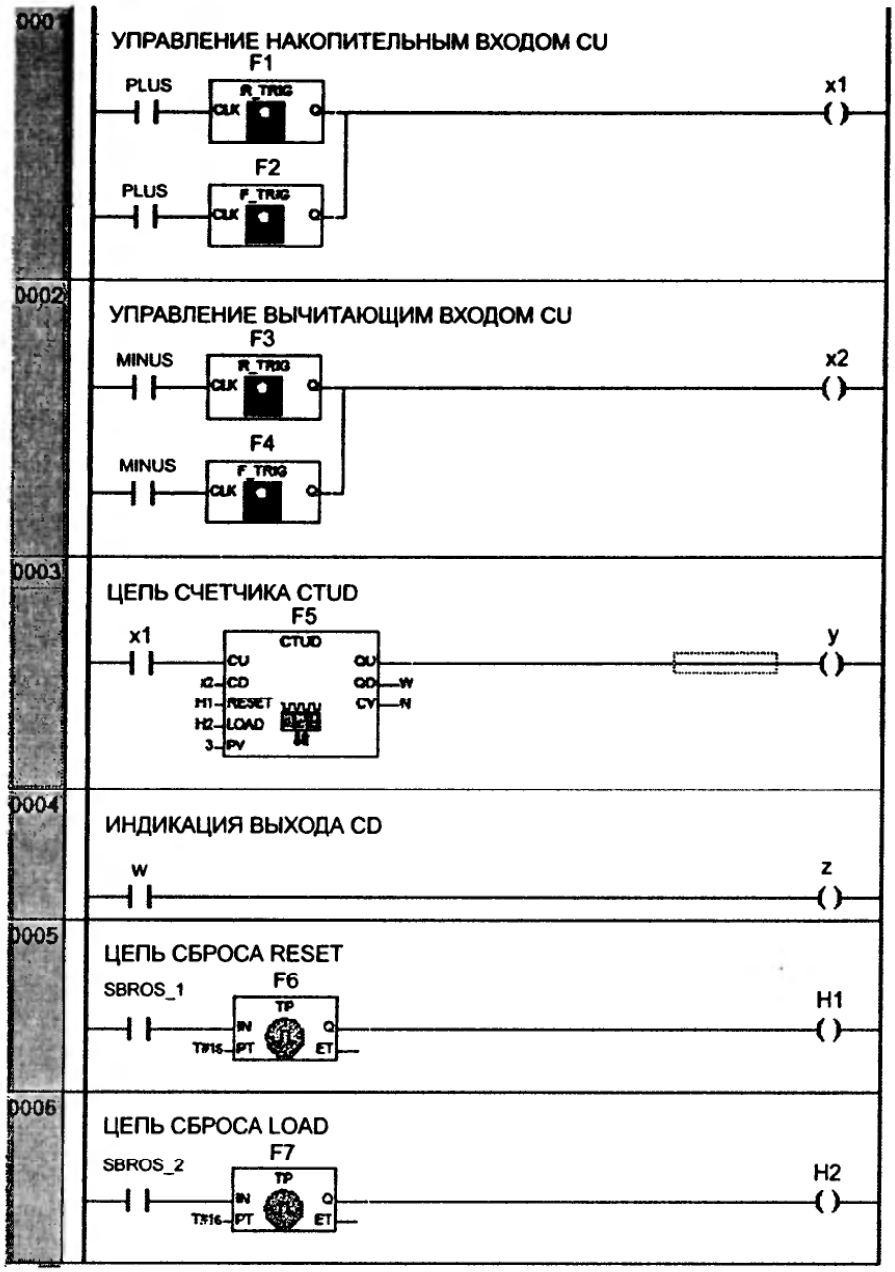


Рис. 4.34. Схема для испытания **CTUD**-счетчика

Эта схема позволяет сделать выводы:

- Выход **CV** загружается значением **PV** лишь после нажати на кнопку **SBROS_2**.
- Выход QU примет значение TRUE при **CV** = **PV**, и значе ние **CV** будет нарастать при дальнейшем поступлении им пульсов на вход **CU**. (В ПЛК других производителей бывак **CTUD**-счетчики, прекращающие в этом случае дальнейше накопление сигналов).
- Выход **CV** обнуляется в любой стадии счета после нажати кнопки **SBROS_1**.
- Если выход **CV** доведен до 0, то дальнейшее поступлени импульсов на входе **CD** не подсчитывается.
- При одновременном поступлении импульсов на входы **CU** **CD** происходит увеличение **CV**, т. е. сигнал на вычитающе

входе теряется. Поэтому если для **CTU**- и **CTD**-счетчиков длительность подсчитываемых импульсов не критична, то для **CTUD**-счетчика время действия сигналов на входах **CU** и **CD** не должно превышать продолжительность цикла сканирования, т. к. может сложиться ситуация, рассмотренная в п. 4.8.2.

Полученные навыки позволяют теперь реализовать на языке LD ранее спроектированную СЛУ (рис. 1.3) и представленную в виде многоступенчатой схемы на рисунке 4.35.

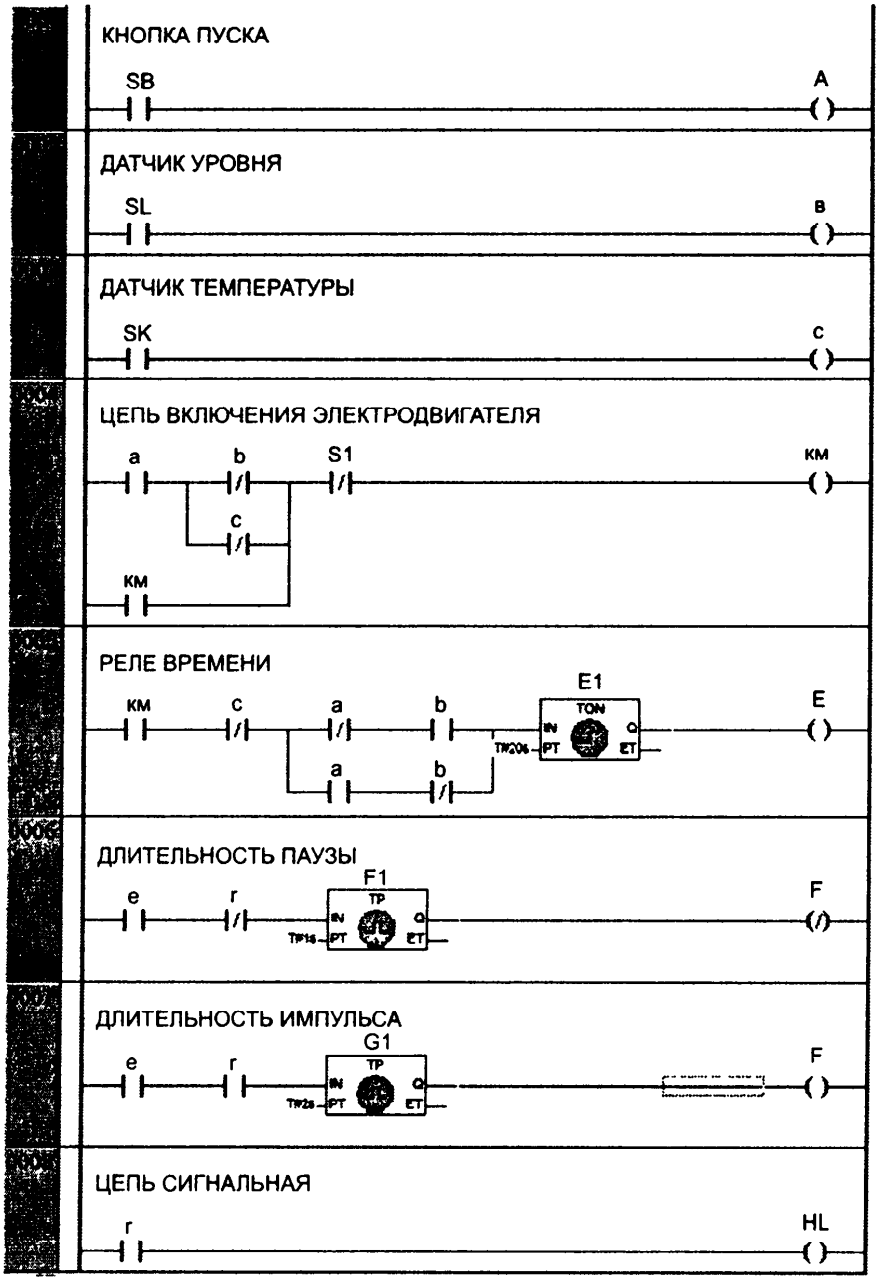


Рис. 4.35. СЛУ на LD

Промежуточное реле **D** по рисунку 1.3 не потребовалась, т. к. реле **KM** в LD может иметь любое количество виртуальных контактов, необходимых для получения требуемой логики работы, и в то же время обеспечить реальный выход ПЛК для подключения катушки магнитного пускателя.

5. ПРИМЕР ПРОЕКТИРОВАНИЯ СЛУ

Ознакомившись с основными компонентами схем и приемами программирования на языке LD, можно приступить к проектированию простых систем, описываемых с позиций комбинаторной и/или событийно-управляемой логики.

Напоминаем, что в системах комбинаторной логики состояние выходных элементов, т. е. исполнительных механизмов определяется только комбинацией состояний входных или приемных элементов (реле, кнопок, контактных датчиков и т. д.). В системах событийно-управляемой логики состояние выходных элементов зависит не только от комбинаций состояний входных элементов, но и последовательности их изменения во времени.

Существуют различные методы проектирования конечных автоматов. Мы же воспользуемся более простым методом временных диаграмм, т. е. когда словесное описание проектируемой системы можно представить в виде графиков изменения входных и выходных сигналов, используя при этом эвристический подход для достижения поставленной цели.

5.1. Постановка задачи

Например, необходимо создать систему управления электроприводами горизонтального **A** и наклонного **B** транспортеров для уборки навоза в животноводческом помещении (рис. 5.1).

Сначала необходимо составить словесное описание проектируемой системы. На этой стадии следует выявить количество и технические характеристики входных элементов и исполнительных механизмов, опираясь на известные решения подобных задач и личный опыт, пожелания технологов и обслуживающего персонала, требования безопасности.

Допустим, в результате этих действий выявили, что необходимо иметь на пульте управления два приемных элемента: кнопки «Пуск» и «Стоп», работающие с самовозвратом, т. е. без фиксации включенного состояния и для привода транспортеров два исполнительных механизма: **M1** и **M2**. Проектируемую СЛУ можно отнести к системам мягкого реального времени. В качестве контроллера имеем возможность использовать ОВЕН ПЛК 100PL.

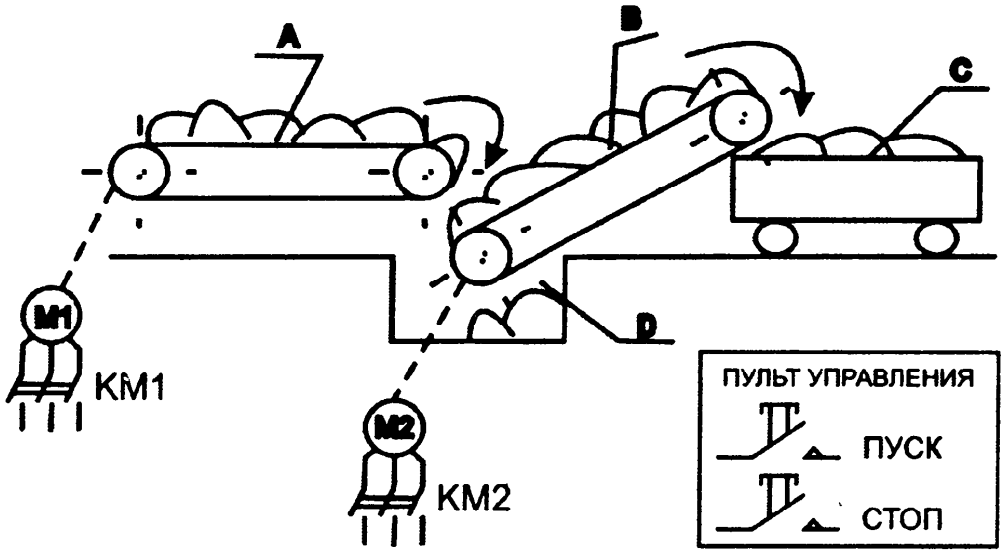


Рис. 5.1. Двухтранспортерная линия:
A – горизонтальный транспортер; **B** – наклонный транспортер;
C – тракторная тележка; **D** – навозная яма;
КМ1 и **КМ2** – контакты магнитных пускателей

При пуске линии первым должен включиться **М2** (чтобы не было завала в навозной яме!) и через 10 с (по совету технологов) двигатель **М1**.

При нормальном режиме работы наклонный транспортер должен работать 10 мин, но первым за 20 с до остановки **М2** должен выключиться **М1**, чтобы за это время разгрузить навозную яму и облегчить последующий запуск линии.

В случаи малого количества навоза оператор может досрочно, нажав кнопку «Стоп», остановить процесс. При этом первым должен остановиться **М1** и через 20 с – **М2**. При необходимости (аварийная ситуация!) сразу остановить **М1** и **М2**, кратковременно нажав обе кнопки «Пуск» и «Стоп» одновременно. Повторное (возможно ошибочное) нажатие этих кнопок не должно приводить к включению транспортеров. Повторный запуск транспортеров допускается не менее, чем через 10 мин с целью охлаждения двигателей **М1** и **М2** или для выяснения причин аварийной остановки и их устранения.

Эта простая программа действий при дальнейшем анализе системы может быть расширена. Например, потребуется установить датчики тепловой защиты двигателей (еще два входных элемента!), аварийное отключение при отказе одного из двигателей (особенно плохо – если **М2**!) и т. д.

И перечень таких ситуаций можно продолжать. Поэтому ограничимся вышеуказанной схемой работы, представив ее в виде временной диаграммы на рисунке 5.2.

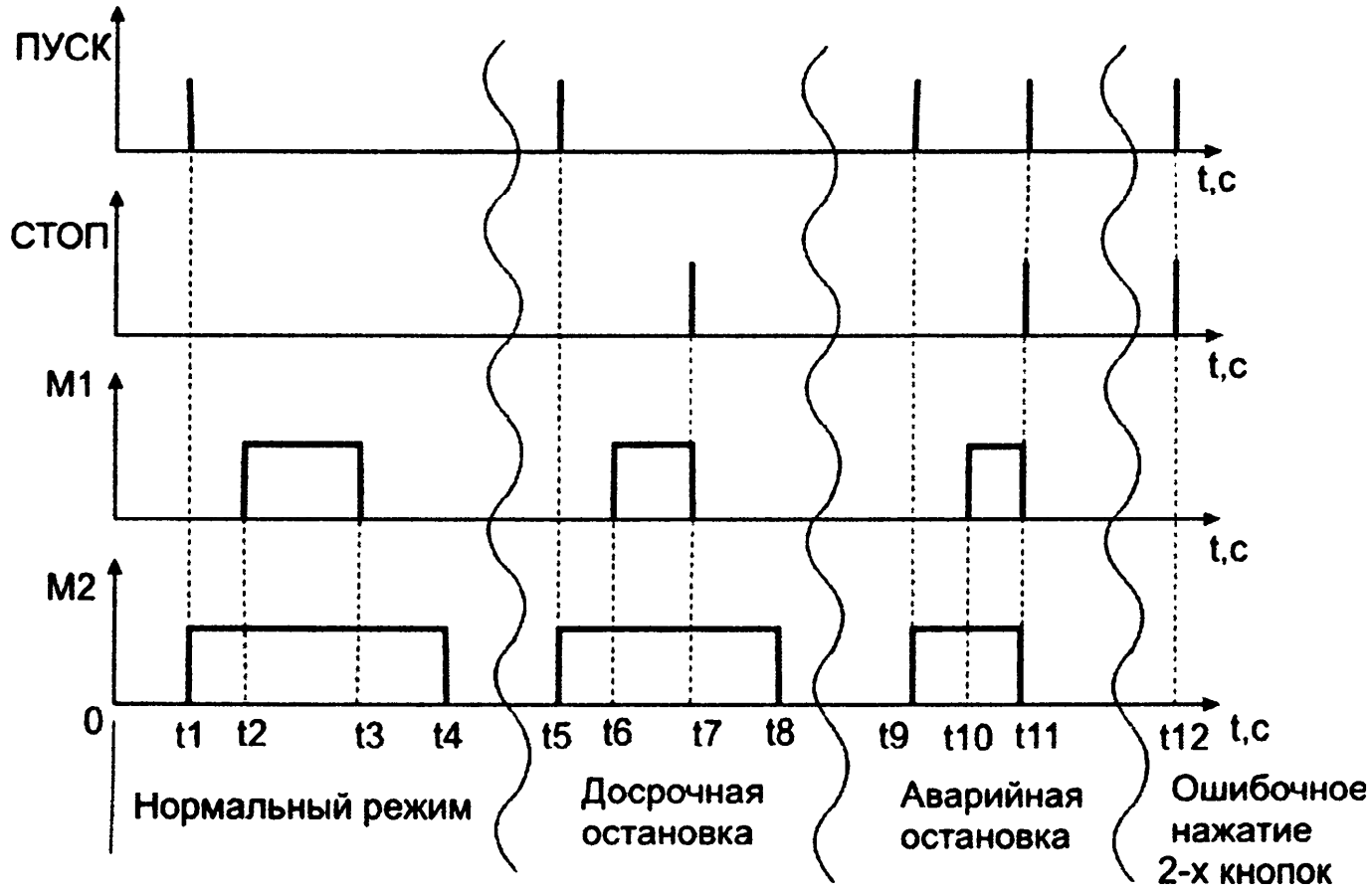


Рис. 5.2. Временная диаграмма состояний входных и выходных элементов проектируемой системы

Очевидно: $t2 - t1 = t6 - t5 = t10 - t9 = 10$ с;

$t4 - t3 = t8 - t7 = 20$ с;

$t4 - t1 = 600$ с;

$t3 - t2 = 600 - 10 - 20 = 570$ с.

Опишем режимы работы по этой диаграмме и попытаемся одновременно проектировать схему на LD.

Результат во многом зависит от опыта проектировщика, его предпочтений при выборе того или иного способа получения требуемых решений.

Например, фиксацию кратковременного нажатия кнопки «Пуск» можно выполнить, по меньшей мере тремя методами (рис. 4.13, 4.14, 4.18).

Естественно, имена компонентов будем писать на английском языке (или с применением английского шрифта).

Итак, начинаем.

5.2. Нормальный режим работы

В момент **t1** оператор нажал кнопку **PUSK**, и через **TP**-таймер **F1** кратковременно сработало виртуальное реле **К1** (рис. 5.3). Кстати, этот таймер потом при подключении кнопки к реальному ПЛК защитит его от так называемого «дребезга контактов», имеющего место в механических контактных элементах. Во время этого дребезга возникает хаотичная группа импульсов, что в ряде случаев может привести к ложным срабатываниям ПЛК.

TP-таймер же реагирует на первый импульс, игнорируя последующие, если те укладываются во временной интервал уставки по его входу **PT** (рис. 4.22). Обычно достаточно принять уставку порядка десятков миллисекунд, т. к. по техническим условиям на контактные изделия продолжительность дребезга не должна превышать 1мс. Но значение **PT**, в общем случае, не должно превышать время прогона программы (п. 4.8.2).

Первая цепь имитирует работу кнопки с самовозвратом, т. е. при нажатии на кнопку **PUSK** кратковременно (на период действия импульса с **TP**-таймера **F1**) сработают контакты этого реле. Теперь необходимо зафиксировать это кратковременное срабатывание. С этой целью во второй цепи поставим **RS**-триггер **F2**. При замыкании контакта **К1** на входе **SET** этого **FB** на его выходе **Q1** появится сигнал, запускающий **TP**-таймер **F3** с уставкой **PT** на 10 мин или 600 с. Можно установить по желанию любой вариант **PT**. Это обеспечить включение в тот же момент **t1** двигателя наклонного транспортера.

В третьей цепи виртуальный контакт реле **М2** через **TON**-таймер **F4** с задержкой на 10 с запустит **TP**-таймер **F5** с уставкой **PT** на 9,5 мин. Эта цепь обеспечит в момент **t2** запуск **М1** и его отключение в момент **t3**, т. е. за 20 с до остановки **М2**. Начальный фрагмент проектируемой системы изображен на рисунке 5.3.

Пока ??? на входе **RESERT1** в **RS**-триггере оставим без ответа. Формально нормальный режим работы транспортеров в интервале 0...**t4** вроде бы выполнен.

Надо теперь предусмотреть возможность досрочной остановки процесса уборки навоза.

С этой целью в четвертой цепи установим кнопку **STOP** и по аналогии с вышеописанной схемой **TP**-таймер **F6**. В момент **t7** при нажатии на кнопку **STOP** кратковременно срабатывает реле **К2**, контакт которого в следующей цепи вызывает срабатывание реле **Y1**.

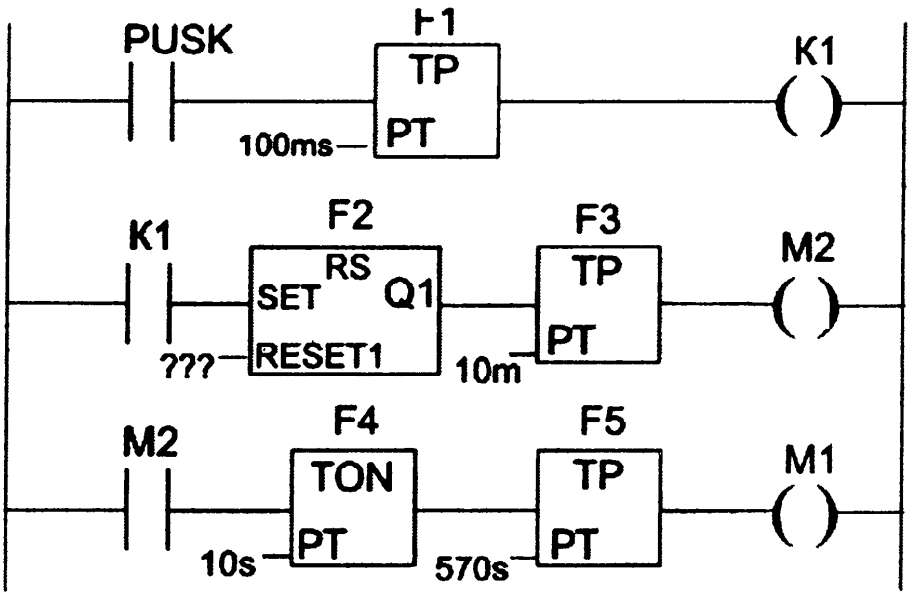


Рис. 5.3. Начальный фрагмент проектируемой СЛУ

Это реле становится на самоблокировку и в следующем цикле прогона программы, т. е. с позиции системы мягкого реального времени практически в тот же момент **t7** должно остановить **M1**. С этой целью в третью цепь поставим размыкающий контакт **Y1**. Кроме того, надо через 20 с отключить **M2.** Потребуется новая шестая цепь (рис. 5.4), в которой **Y1** через **TON**-таймер **F7** запускает виртуальное реле **Y2**, которое своим размыкающим контактом во второй цепи выполнит эту процедуру и в момент **t8**, т. е. через 20 с после остановки **M1** отключится **M2**. Естественно, когда-то придется снять самофиксацию с реле **Y1**. Для этого по известной схеме (рис. 4.14а) поставим размыкающий контакт в цепь с катушкой **Y1**. Но имя этого элемента оставим пока под ???.

Вроде бы выполнили и второй режим досрочной остановки.

«Вроде бы», т. к. еще не решены вопросы с перезапуском **RS**-триггера и аварийной остановкой **M1** и **M2**.

Можно предложить такой вариант ответа на эти вопросы.

Создадим еще одну седьмую цепь.

Контактами **K1** и **K2**, которые срабатывают в момент **t11** при нажатии на кнопки **PUSK** и **STOP** через **TP**-таймер **F8** запустим реле **h**, а имя этого реле поставим на входе **RESET1** **RS**-триггера **F2**.

Кроме того, размыкающий контакт этого реле в тот же момент **t11**, а точнее в следующем цикле прогона программы снимет блокировку с реле **Y1** и обесточит двигатели **M1** и **M2**. Кстати, вы-

полнить точно одномоментное нажатие и включение механических кнопок **PUSK** и **STOP** под силу не каждому оператору. Здесь нам помогут таймеры **F1** и **F6**, которые не только защищают ПЛК от дребезга контактов, но и устраняют последствия возможной асинхронности в действиях оператора при нажатии на кнопки. Уставка **TP**-таймера **F8** должна быть не менее уставки запущенного ранее **TP**-таймера **F3**. За это время снова запустить технологический процесс нельзя. (Надо установить и устранить причину аварийной остановки!)

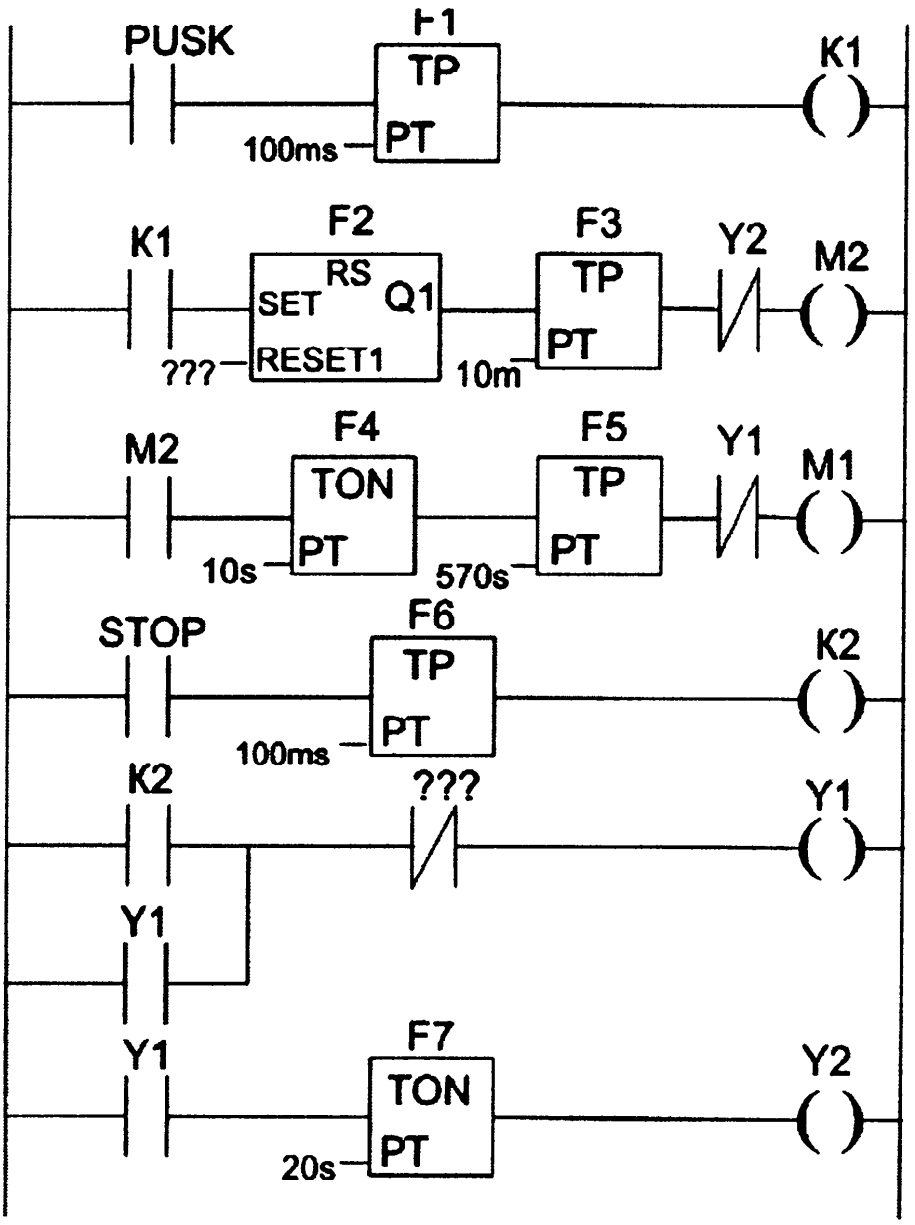


Рис. 5.4. Развитие начального фрагмента

Получили вроде бы окончательный вариант проектируемой СЛУ (рис. 5.5).

Если запустить эту программу в режим эмуляции, то при первом запуске все сработает, как и планировалось.

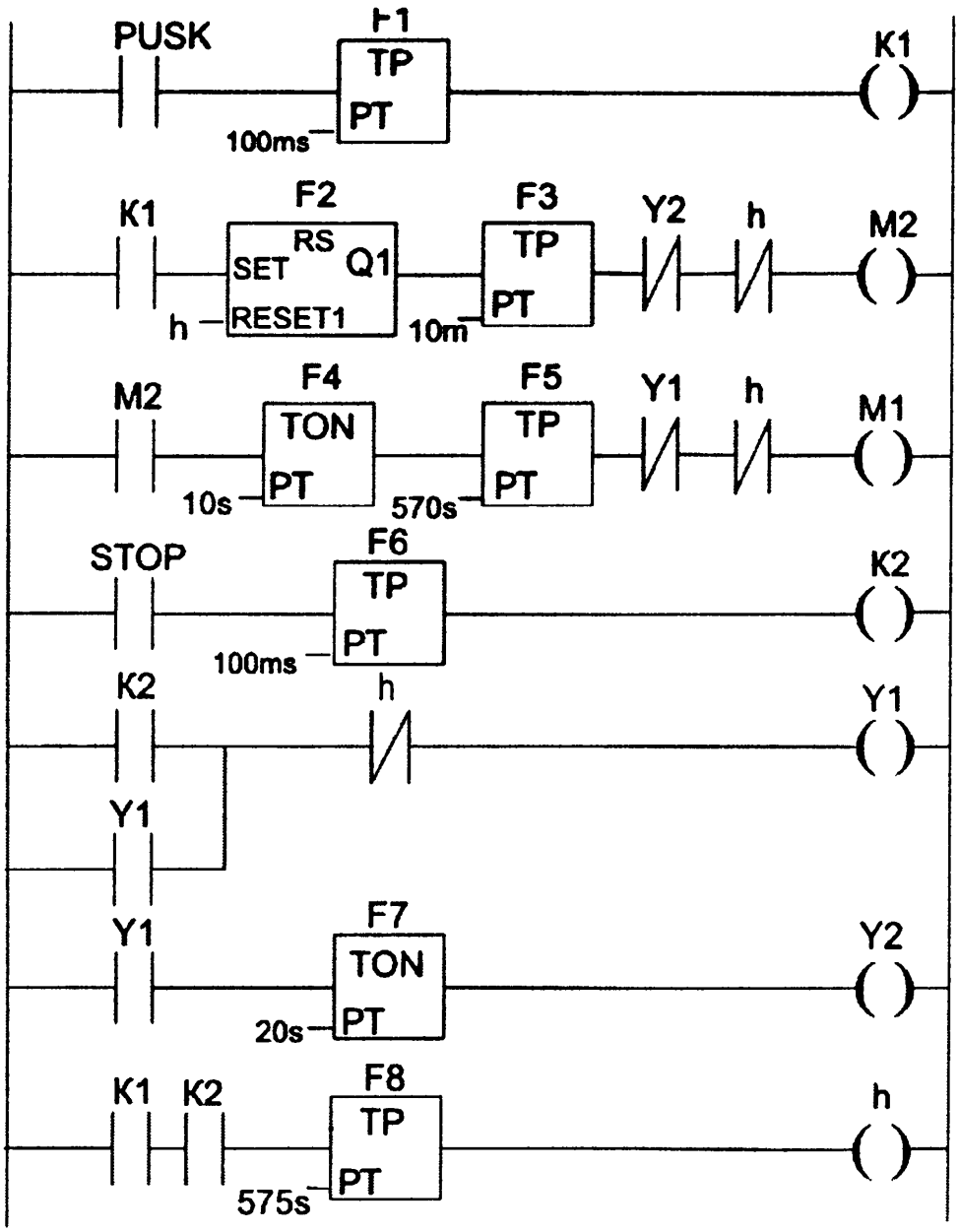


Рис. 5.5. Почти окончательный вариант СЛУ

Но попытка повторного запуска в первом и втором режимах (рис. 5.2) не дает желаемого результата. В чем дело? Ответ простой. Выход **Q1 RS**-триггера после поступления сигнала по **SET** входу остается в состояние TRUE, пока не получит сигнал сброса на вход **RESET1**. А этот сигнал поступит только в режиме аварийной остановки, т. е. когда будут нажаты кнопки **PUSK** и **STOP.** Следовательно, надо добиться, чтобы при каждом нажатии на кнопку **PUSK** производилась кратковременная подача сигнала на вход **RESET1**.

С этой целью создадим еще одну (восьмую) цепь, в которой контакт **К1**, срабатывающий при нажатии кнопки **PUSK**, запустит **TP**-таймер **F9** с выдержкой времени значительно меньше, чем у таймера **F1** (рис. 5.6). Сработает реле **К3** и своим замыкающим

контактом в следующем цикле приведет к кратковременному срабатыванию катушки реле **h**, установленной в седьмой цепи. Это реле уже в *следующем* цикле прогона программы подаст своим замыкающим контактом (на схеме его нет!) кратковременный сигнал TRUE по входу **RESET1** RS-триггера, подготовив тем самым его к приему сигнала по входу **SET**. Поэтому уставка **PT** у **F1** должна быть, по крайней мере, больше аналогичной уставки для **F9** на двойную продолжительность цикла сканирования. Иначе RS-триггер не сработает (рис. 4.18).

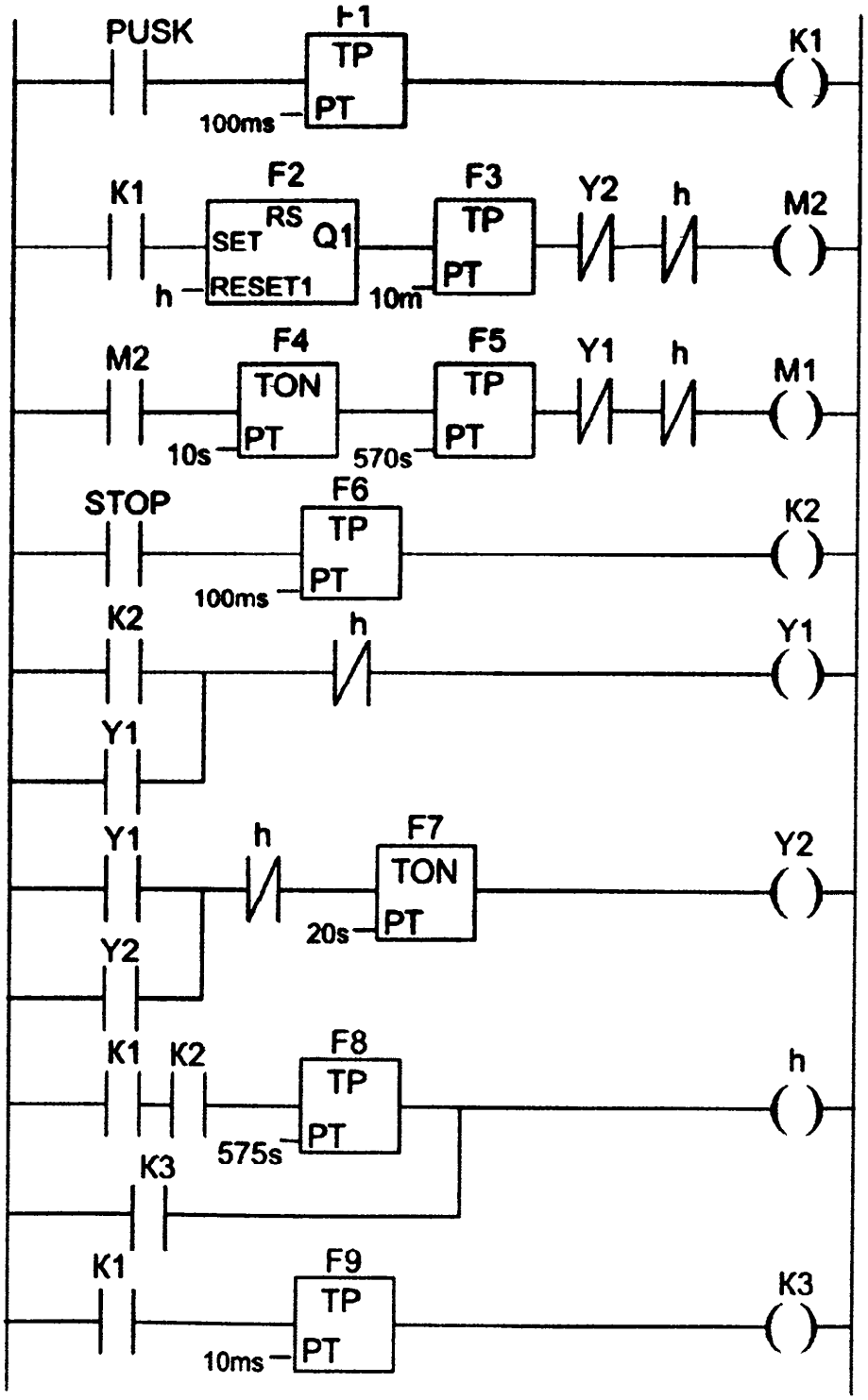


Рис. 5.6. Окончательный вариант проектируемой СЛУ

Можно сократить эту разницу в уставках **PT** для **F1** и **F9** на один период прогона программы, «подняв» восьмую цепь хотя бы на «ступеньку» выше седьмой цепи, т. е. поменять их местами. Но так как проектируемая СЛУ будет работать в режиме мягкого реального времени, этого делать не обязательно.

5.3. Вопросы безопасности

С целью безопасной работы технологической линии желательно кнопку **STOP** выбирать с размыкающим контактом, т. к. в случае обрыва линии связи этой кнопки с ПЛК появится сигнал, требующий вмешательства технологов. С позиций безопасного обслуживания подобного технологического процесса можно утверждать, что лучше не запустить, чем вовремя не остановить!

Тогда четвертую цепь в схеме по рисунку 5.6 можно заменить, как показано на рисунке 5.7. Потребуется еще один блок **F_TRIG F10**.

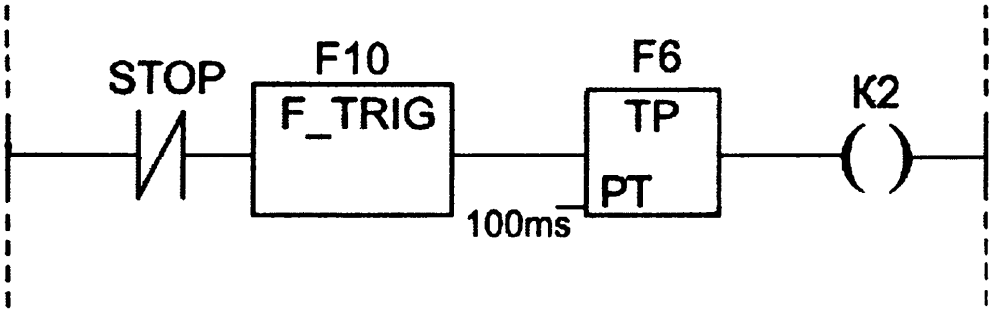


Рис. 5.7. Фрагмент схемы с размыкающей кнопкой STOP

5.4. Перенос программы в ПЛК

До сих пор мы строили различные многоступенчатые схемы в предположении, что их исследование будет завершаться только в режиме эмуляции, т. е. без последующей привязки к конкретному ПЛК. Даже простенькую СЛУ по рисунку 4.8 «подключили» к ПЛК по схеме, представленной на рисунке 4.9, не объясняя предшествующих действий.

Восполним этот пробел на примере программирования СЛУ по рисунку 5.6. Будем считать, что проект этой СЛУ составлен пока только на бумаге, и нам предстоит перенести его на язык LD в CoDeSys сначала в ПК, а затем отладить на самом ПЛК. Для этого необходимо выполнить целый ряд еще незнакомых действий.

5.4.1. Итак, перед нами рабочая область с еще пустыми цепями. Перед их заполнением следует выполнить конфигурацию входных и выходных переменных, т. е. ввести имена этих переменных определенным образом в конфигуратор ПЛК. Рассмотрим этот этап подробней.

Для этого под окном организатор объектов (рис. 3.6) необходимо нажать 1ЛКМ на вкладку Resources (ресурсы) и из дерева ресурсов выбрать PLC Configuration (конфигурация ПЛК). Откроется окно (рис. 5.8), в котором и следует объявлять входные и выходные переменные.

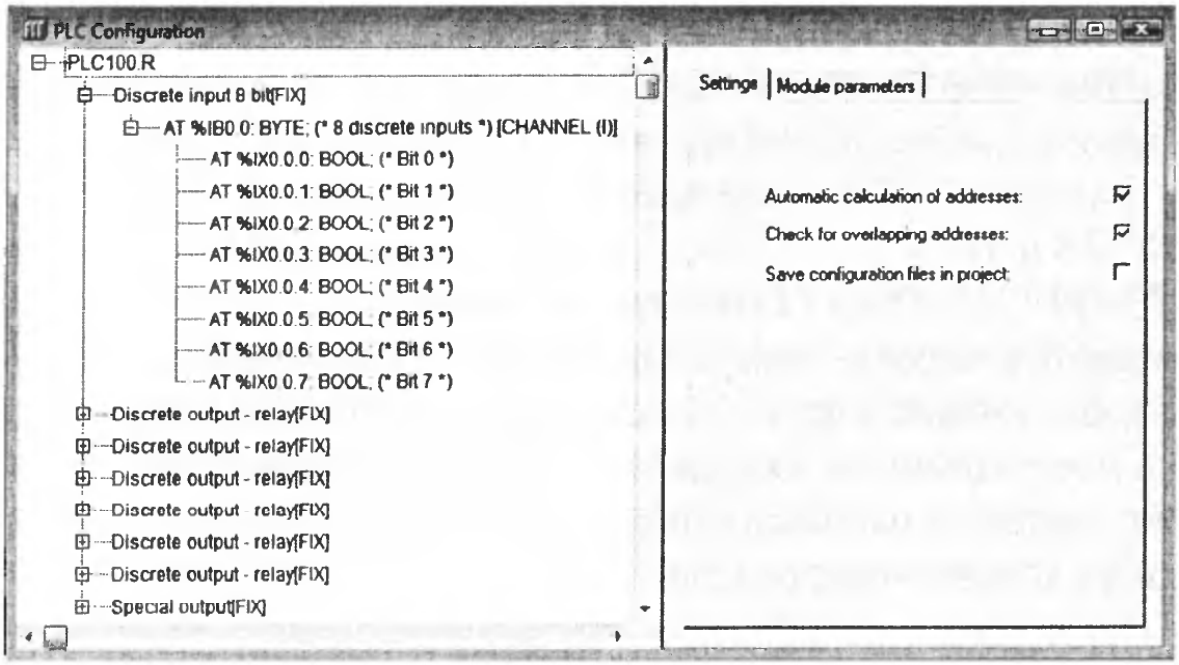


Рис. 5.8. Окно конфигуратора ПЛК

Напомним, что в проектируемой СЛУ две переменные ввода (**PUSK** и **STOP**) и две вывода (**M1** и **M2**). 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 ПЛК-100 всего 8 дискретных входов, то в нашем распоряжении будет и 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 2ЛКМ по надписи AT строчки AT % IB0.0.0: BOOL и вводим имя переменной **PUSK.** Завершаем ввод клавишей Enter. Аналогичным образом, но только уже в строку AT % IB0.0.1: BOOL присвоим значение переменной **STOP**.

Для ввода выходных переменных необходимо нажать на «+» возле строки **Discrete Output-Relay**. В нашем ПЛК всего 6 дискретных выходов. Поэтому откроются 6 строк: от AT%QX1.0 до AT%QX6.0. Выбираем любые две из шести. Например, щелкаем 2ЛКМ по надписи AT в строке AT%QX1.0: BOOL и вводим имя первой выходной переменной **M1**. Аналогичным образом следует ввести имя второй переменной вывода **M2** в строку AT %QX2.0: BOOL.

Выполненные действия по конфигурации ПЛК предполагают, что при аппаратной реализации данной СЛУ кнопки **Pusk** и **Stop**, а также катушки магнитных пускателей **КМ1** и **КМ2** будут подключены к соответствующим входам и выходам ПЛК.

*Если бы требовалось выполнить конфигурацию ПЛК для СЛУ по рисунку 4.9, то входные переменные **SB**, **SL** и **SK** следовало «привязать» к строчкам в окне конфигуратора (рис. 5.9) AT%IB0.0.2, AT%IB0.0.3 и AT%IB0.0.6. Выходные переменные **HL** и **KM** к строчкам AT%QX1.0, AT%QX3.0 соответственно.*

Остальные строки следует оставить без изменения, они могут быть использованы в дальнейшей модернизации СЛУ, путем включения в нее различных входных и выходных устройств.

Итак, переменные введены, и полученный результат представлен в окне конфигуратора (рис. 5.9).

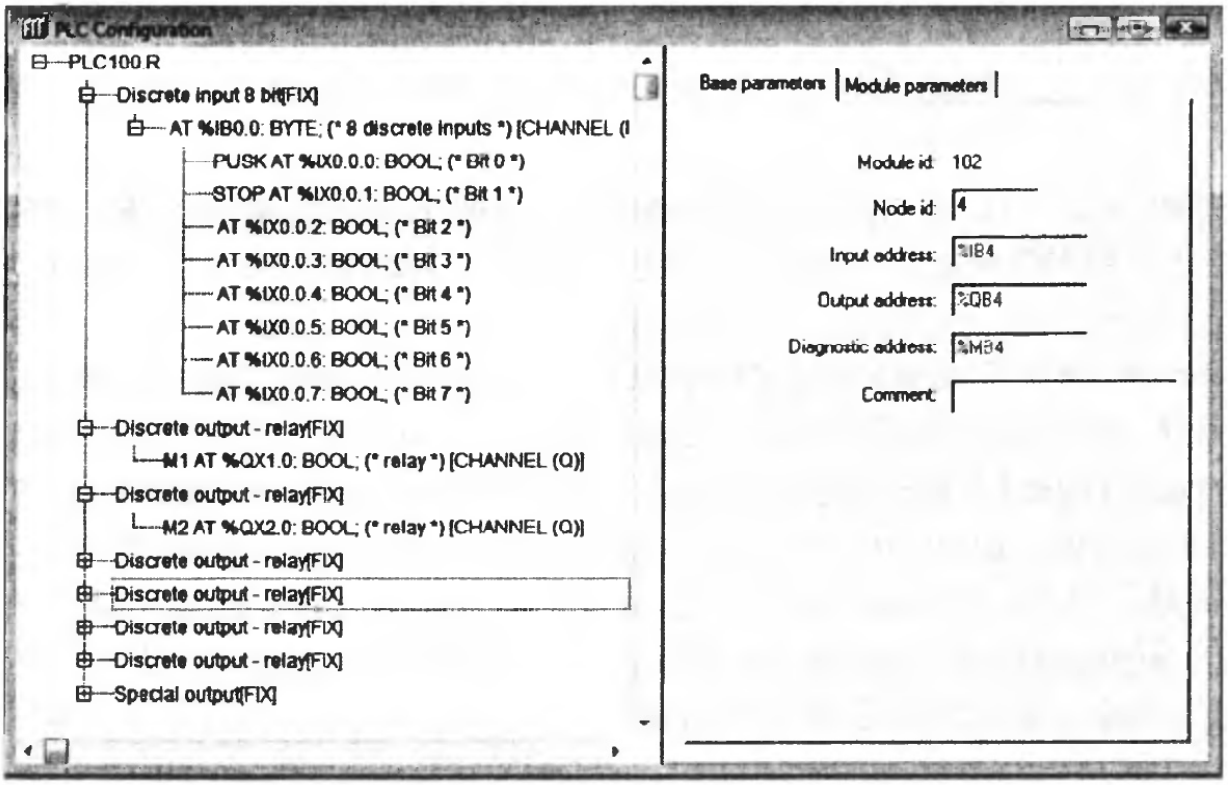


Рис. 5.9. Окно конфигуратора ПЛК

5.4.2. Следующим этапом является внесение остальных элементов СЛУ в программу. Этот процесс уже хорошо известен читателю по предыдущим главам, поэтому пропустим его. Однако, необходимо отметить, что при инициализации контактов **PUSK** и **STOP**, катушек **M1** и **M2** не будет всплывать окно **Declare Variable** (рис. 4.2), поскольку программа уже «запомнила» имена переменных в конфигураторе.

5.4.3. Программа составлена. Но до записи и запуска ее в ПЛК требуется еще выполнить несколько операций.

Для начала следует подключить контроллер к компьютеру. Для этого ПЛК имеет следующие коммуникационные интерфейсы: **Ethernet 10/100 Mbps**, **RS232** и **RS485**. Мы советуем для связи контролера с ПК выбрать первый вариант со скоростью соединения 100 Mbps, поскольку интерфейс **Ethernet** универсален, прост в настройке и совместим с современным оборудованием, содержащим данный порт.

В соответствии с паспортом и руководством по эксплуатации ОВЕН ПЛК100 подключать интерфейс **Ethernet** следует 8-ми жильным кабелем «витая пара» категории 5. На кабель необходимо установить оконечные разъемы без экрана. При подключении к концентратору используется обычный (прямой) кабель. А при коммутации к сетевой плате или к иному оборудованию используется кабель **Up-Link** (кабель с перекрестным монтажом первой и второй пар).

Подключим контролер к компьютеру. Один конец кабеля соединяем с портом **RJ-45** контроллера, другой с аналогичным, расположенным на материнской или сетевой карте компьютера.

Для настройки в окне CoDeSys на вкладке **Online** выбираем **Communication parameters**. Откроется окно (рис. 5.10), в котором производим следующие действия:

- Нажимаем кнопку **New** и указываем тип соединения **TCP/IP (Level 2)**.
- Нажимаем 2ЛКМ на **«localhost»** в строке **Address** и вводим **IP-адрес** нашего контроллера 10.0.6.10, который устанавливается изготовителем. Завершаем ввод клавишей Enter.

Настройку компьютера выполняем следующим образом: в параметрах сетевого окружения в разделе протокола **TCP/IPv4** вводится **IP-адрес** компьютера в подсети 10.0.6, отличный от IP-адреса контроллера (10.0.6.10). Например: 10.0.6.1. Маска подсети 255.255.0.0.

Все эти действия удобно выполнять с помощью видеоинструкций, которые прилагаются на установочном диске ПЛК-100.

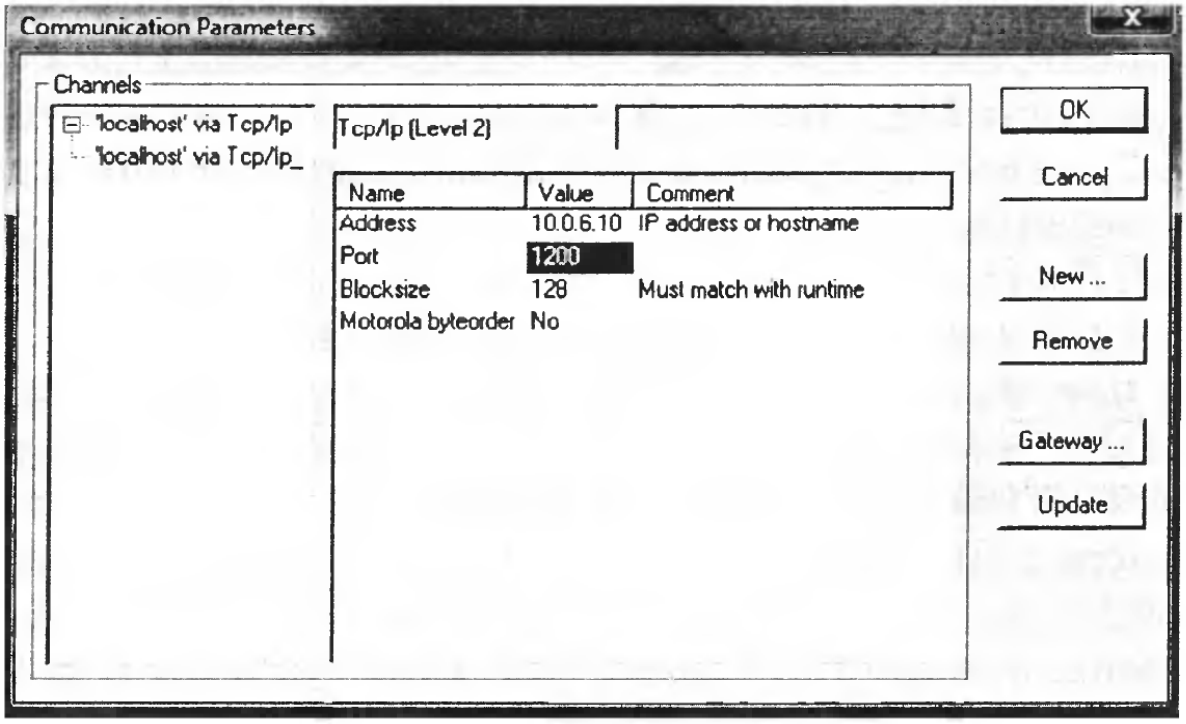


Рис. 5.10. Окно коммуникационных параметров

5.5. Визуализация

Вряд ли такой простой технологический процесс нуждается в визуализации, да и в самом пульте оператора. Поэтому, создавая графическую оболочку программы визуализации, мы преследовали чисто обучающие цели, чтобы ввести читателя в эту проблему и на простом примере показать пути ее решения.

Визуализация предназначена для графической и символьной интерпретации хода технологического процесса, вывода необходимой информации на экран дисплея компьютера, построения графиков, гистограмм. При этом обычно возникает необходимость отразить движение изделий или потоков продукта, заполнение резервуаров, перемещение рабочих органов и т.п.

Движение в LD можно отразить различными приемами. Например, изменением цвета. Двигатель остановлен – его изображение залито серым цветом. Двигатель включен – красным цветом. Однако такая статическая картина притупляет внимание оператора. Поэтому лучше применять мигающую анимацию. Например, включение того же двигателя отображается на дисплее пульсацией его окраски с заданной длительностью чередования фаз серого и красного цвета. Или движение продукта по трубопроводу, ленте транспортера можно изобразить мигающими стрелками.

На других языках программирования в CoDeSys можно с использованием средств анимации изобразить движение как непрерывный процесс. Например, заполнение резервуара жидкостью, перемещение «стрелки» виртуального прибора и др.

Весьма удобным для постижения азов конструирования графических объектов управления является Редактор визуализации в среде CoDeSys.

Программный продукт предоставляет набор готовых графических элементов, которые могут быть связаны соответствующим образом с переменными проекта.

Для того чтобы создать визуализацию в среде CoDeSys, выбираем вкладку **Visualization** в Организаторе объектов (рис. 3.6). Нажимаем 1ПКМ в окне организаторов объектов и выбираем **Add object.** В открывшемся окне вводим наименование визуализации (*обязательно на английском языке*), например **transporter**, нажимаем ОК. Откроется окно, в котором и будет создаваться графический объект управления.

Обратим внимание на Панель инструментов. В ней появились новые компоненты, представленные в таблице 5.1 и используемые при создании визуализации.

Желательно в процессе создания визуализации ориентироваться на схему работы транспортера, описанную выше (рис. 5.1), чтобы графический объект получился более реалистичным и понятным.

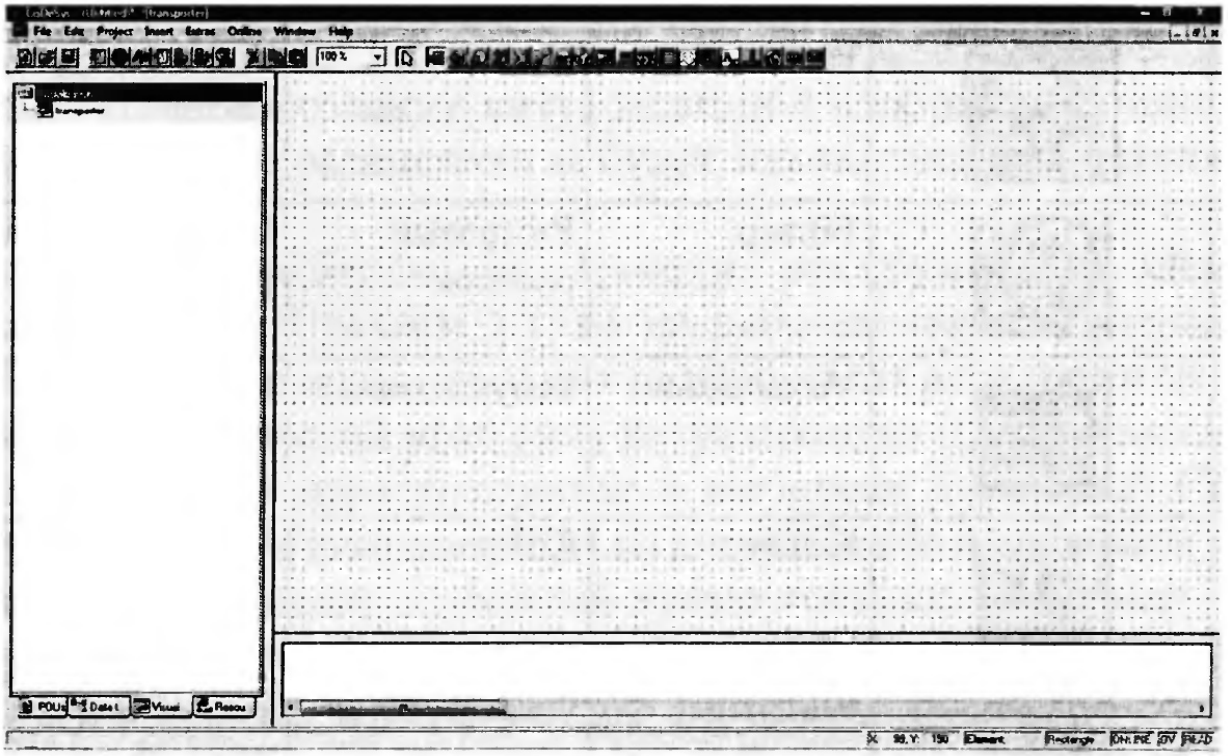
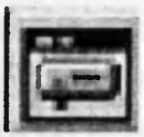
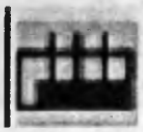


Рис. 5.11. Окно визуализации

Таблица 5.1

Описание графических изображений кнопок в панели инструментов Редактора визуализации в CoDeSys

№ п/п	Графическое изображение кнопки	Англоязычная среда	Русскоязычная среда	Описание
1		Rectangle	Прямоугольник	Вставить прямоугольник
2		Rounded Rectangle	Закругленный прямоугольник	Вставить прямоугольник с закругленными углами
3		Ellipse	Эллипс	Вставить эллипс
4		Polygon	Многоугольник	Вставить многоугольник
5		Polyline	Ломаная линия	Вставить ломаную линию
6		Curve	Кривая	Вставить кривую Безье
7		Pie	Сектор	Вставить круговой сектор
8		Bitmap	Растровый рисунок	Вставить растровый рисунок
9		Visualization	Визуализация	Вставить существующую визуализацию
10		Button	Кнопка	Вставить кнопку
11		WMF file	Файл WMF	Вставить метафайл Windows

12		ActiveX-Element	Элемент управления ActiveX	Вставить элемент управления ActiveX
13		Table	Таблица	Вставить таблицу для элементов массива
14		Meter	Стрелочный прибор	Вставить стрелочный индикатор
15		Bar Display	Столбчатый указатель	Вставить столбчатый указатель
16		Histogram	Гистограмма	Вставить гистограмму
17		Alarm table	Таблица тревог	Вставить таблицу тревог
18		Trend	Тренд	Вставить самописец для графического представления тенденции изменения значения переменной

5.5.1. Начнем создание визуализации с изображения площадки, на которой впоследствии и будут находиться все остальные механизмы.

Для этого выберем из Панели инструментов элемент **«Rectangle»**  . Щелкаем 1ЛКМ на рабочем поле и, удерживая ЛКМ, растягиваем прямоугольник до требуемых размеров и смещаем его в левый нижний угол окна визуализации. Нажимаем 2ЛКМ на получившемся прямоугольнике и выбираем категорию **Color**. В области **Color** нажимаем 1ЛКМ на кнопке Inside и выбираем цвет, которым и будет залит созданный прямоугольник. Например – коричневый. Щелкаем 1ЛКМ на кнопке **Frame**, присваиваем тот же цвет (коричневый), которым будет окрашены линии объекта. По аналогии создаем второй прямоугольник меньших размеров, изображающий дно ямы, и поместим его рядом с первым. Также по-

строим третий прямоугольник, являющийся платформой для тележки. Площадка готова (рис. 5.12).

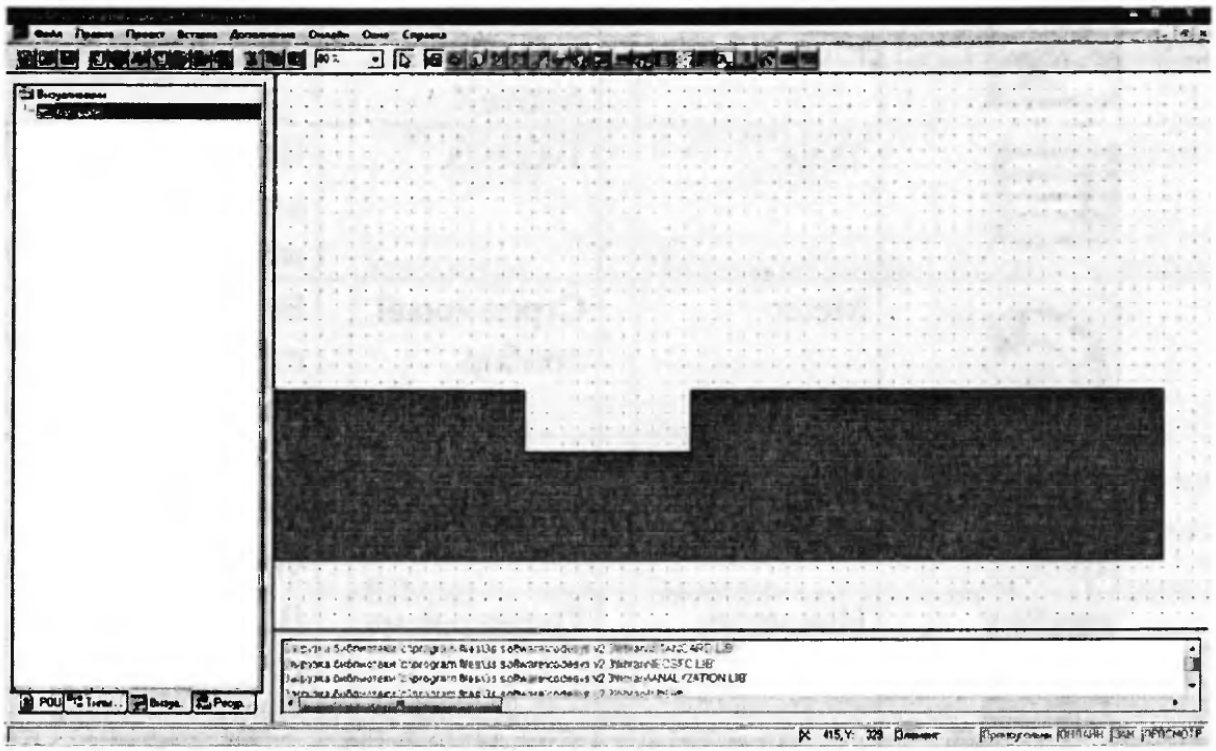


Рис. 5.12. Визуализация площадки линии по уборке навоза

Для визуализации состояния транспортеров выберем самый простой вариант индикации – изменение цвета.

Создадим графическую интерпретацию транспортера **A** (рис. 5.1).

Выбираем элемент из меню Визуализации **«Ellipse»** и нарисуем две окружности с диаметром около 1 сантиметра. Для этого щелкнем 1ЛКМ на рабочем поле и, удерживая ЛКМ, растянем появившуюся окружность до требуемого размера. Щелкаем 2ЛКМ мышью на окружности. Появится диалоговое окно для настройки элемента визуализации. Выбираем категорию **Variables** и в поле **Change color** вводим имя переменной **.M1**. Так как транспортер **A** приводится в движение двигателем **M1,** то это имя сохраняется и в программе визуализации с добавлением точки. ***Точка перед наименованием переменной обязательна!!!*** Эта переменная будет управлять цветом нарисованной окружности.

Теперь выберем цвета в покое и в тревожном состоянии. Для этого выбираем категорию **Color**. В области **Color** нажмите кнопку **Inside** и в появившемся окне нажимаем на любой нейтральный цвет, например, серый. Нажимаем на кнопку **Inside** в области **Alarm Color** и выбираем зеленый цвет. Полученная окружность

будет серой, когда значение переменной ложно, и зеленой, когда переменная истинна.

Вставляем прямоугольник в визуализацию, для этого выбираем в меню элемент **«Rectangle»** . Повторим те же действия по вводу переменной. Цвета выберем те же: серый и зеленый. Первый транспортер закончен.

Транспортер **B** (рис. 5.1) расположен под углом, поэтому процесс создания его рисунка будет немного сложней. Выбираем в меню многоугольник **«Polygon»** , создаем наклоненный прямоугольник. Техника работы с этим инструментом следующая: наводим курсор мыши на место, с которого следует начать создание объекта. Щелкаем 1ЛКМ. Теперь отводим курсор на место следующей точки, нажимаем 1ЛКМ и так, проходя все точки, завершаем построение фигуры щелчком 2ЛКМ на последней точке.

На концах прямоугольника создаем окружности, тем самым получаем наклонный транспортер, вводим в соответствие с вышеописанной методикой переменную **.M2**. Не забываем указать цвета (серый и зеленый). Оба транспортера готовы к работе (рис. 5.13).

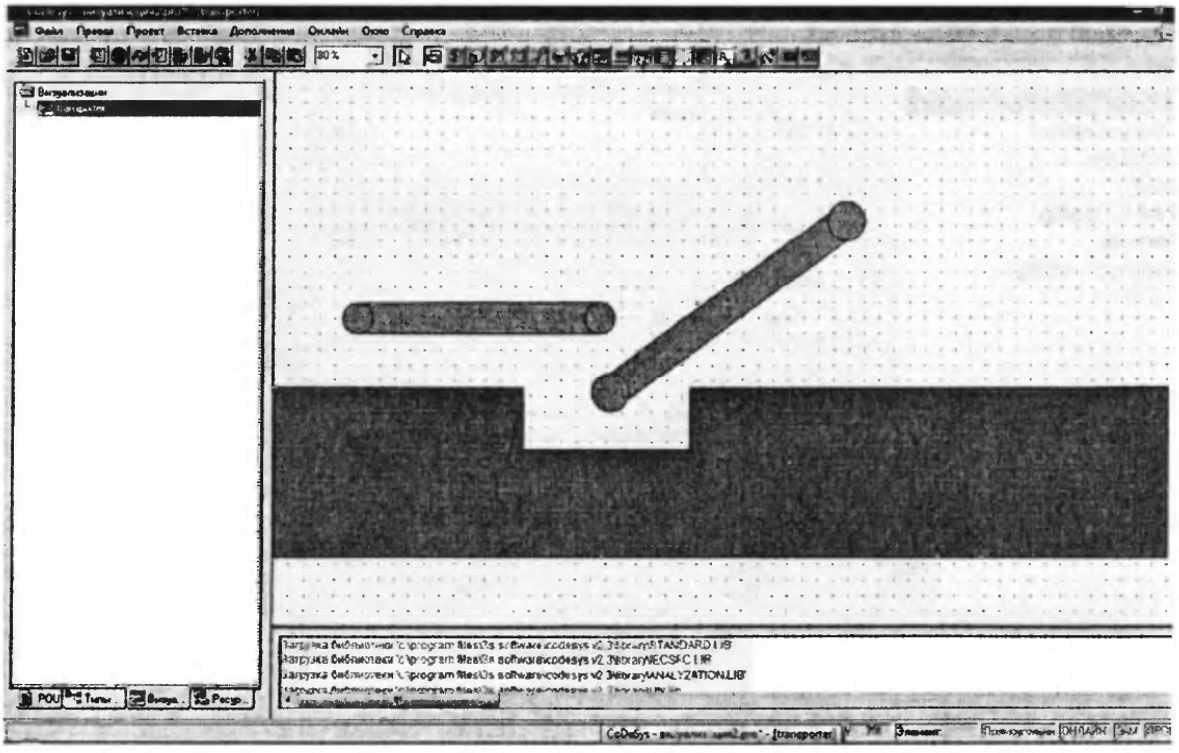


Рис. 5.13. Визуализация транспортеров

Чтобы не утруждать себя созданием рисунков электродвигателей, можно импортировать растровый рисунок. Для этого

выбираем элемент «**Bitmap**»  , выделяем область добавления рисунка до требуемых размеров нажатой ЛКМ. После завершения операции раскрывается диалог открытия файла. В папках компьютера находим заранее выбранный файл с рисунком и завершаем добавление кнопкой Открыть. После чего он появляется в выделенной области. Его размеры можно корректировать, а изображение переносить в окне визуализации, удерживая его ЛКМ.

Добавить рисунки можно трех форматов: *.bmp, *.jpg, *.tif. Нельзя забывать, что вставленные графические файлы должны находиться в папке с файлом программы, иначе они исчезнут. Если же нет готового изображения двигателя, то читатель может повысить свой уровень художественной живописи, нарисовав его с помощью имеющихся элементов в меню Визуализации.

Теперь добавим блок управления. Для этого вставляем кнопку  из меню визуализации. Рисуем кнопку нужных размеров, нажимаем на ней 2ЛКМ. В поле **Content** категории **Text** вводим имя кнопки **pusk** (рис. 5.14).

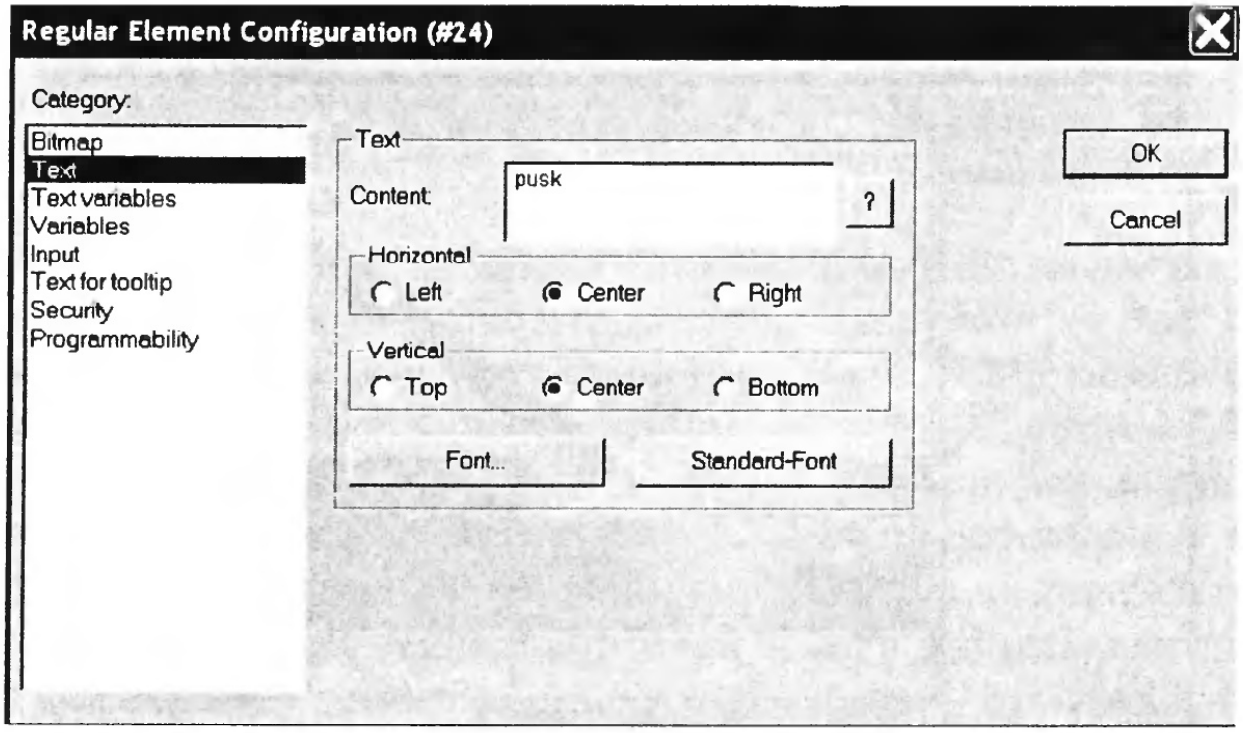


Рис. 5.14 Окно инициализации кнопки

Для того чтобы переменная **pusk** переключалась при щелчке мышкой на этом элементе, в поле **Tap variable** (рис. 5.15) категории **Input** вводим переменную **.pusk** (*не забываем точку*). Созданный переключатель будет включать транспортеры.

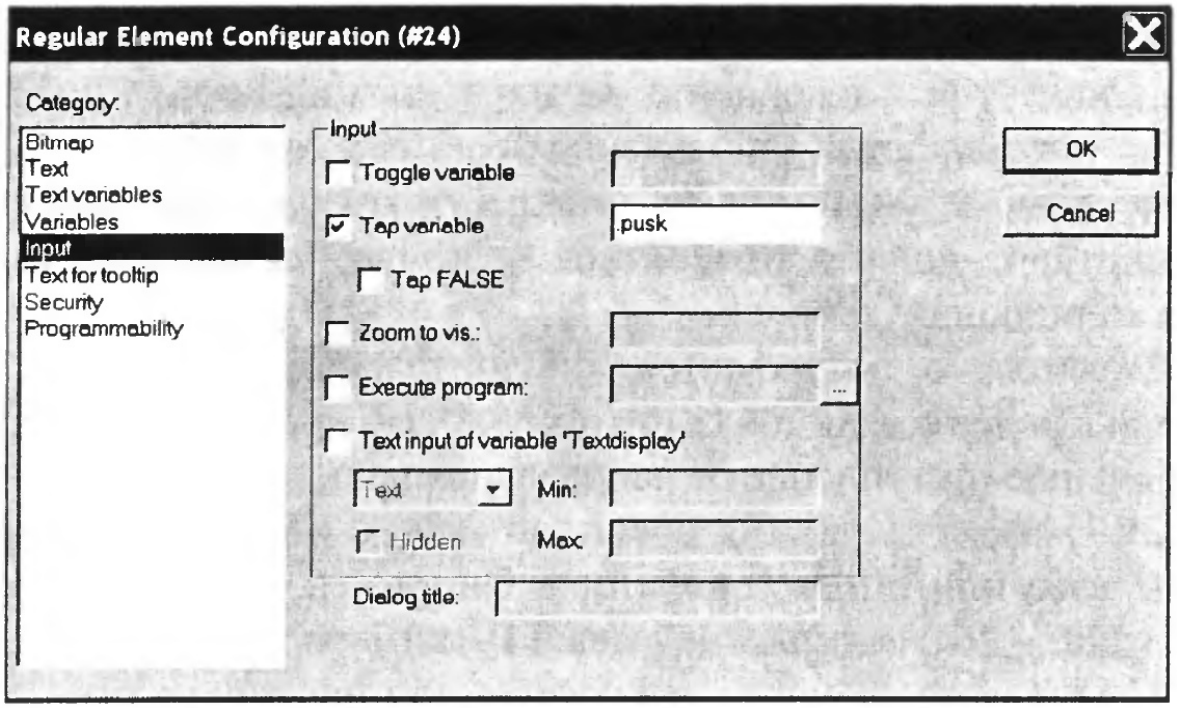


Рис. 5.15. Окно инициализации кнопки

Те же действия выполним для управления кнопкой **STOP**.

Теперь добавим последние штрихи в визуализацию. Изобразим тележку с помощью уже известных элементов **«Rectangle»**, с помощью которого мы изобразим корпус тележки и **«Ellipse»**, который поможет нам нарисовать колеса.

С помощью элемента **Curve** добавим некоторые элементы.

В итоге, визуализация будет иметь вид, показанный на рисунке 5.16.

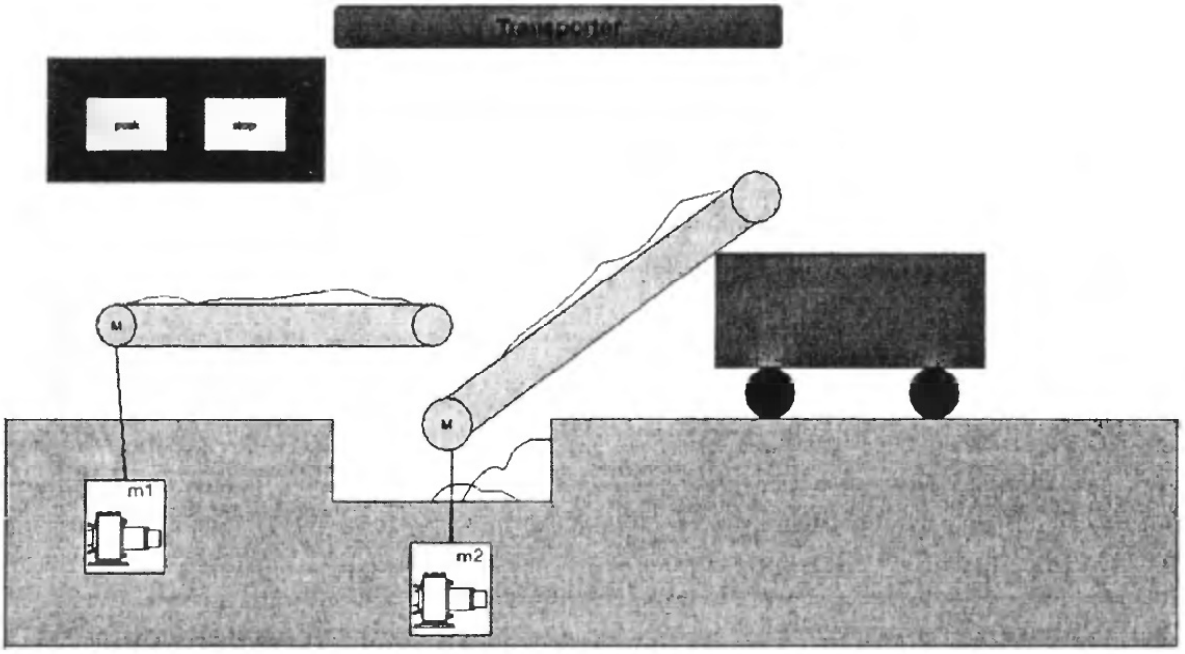


Рис. 5.16. Визуализация транспортера

5.5.2. Однако статическая картина на дисплее, как отмечалось выше, притупляет бдительность оператора. Поэтому лучше ввести

элементы с тревожной индикацией движения транспортеров за счет применения эффекта мерцания их окраски и/или дополнительных стрелок, указывающих направления перемещения навоза.

Допустим, с учетом пожелания оператора, при включении транспортеров должен изменяться не только их цвет, но и появляться мерцающие стрелки.

Следовательно, потребуются дополнительные компоненты и новые цепи в ранее созданной СЛУ (рис. 5.6). Но предварительно сделаем небольшое отступление от направления нашего повествования.

5.5.3. Эффект мерцания очевидно можно обеспечить с применением того или иного генератора. Например, по уже известной схеме (рис. 4.26), содержащей два **TP**-таймера и занимающей две цепи в LD. Так необходимо два комплекта мерцающих стрелок (по одному для каждого транспортера), то дополнительных цепей потребуется четыре (рис. 5.17).

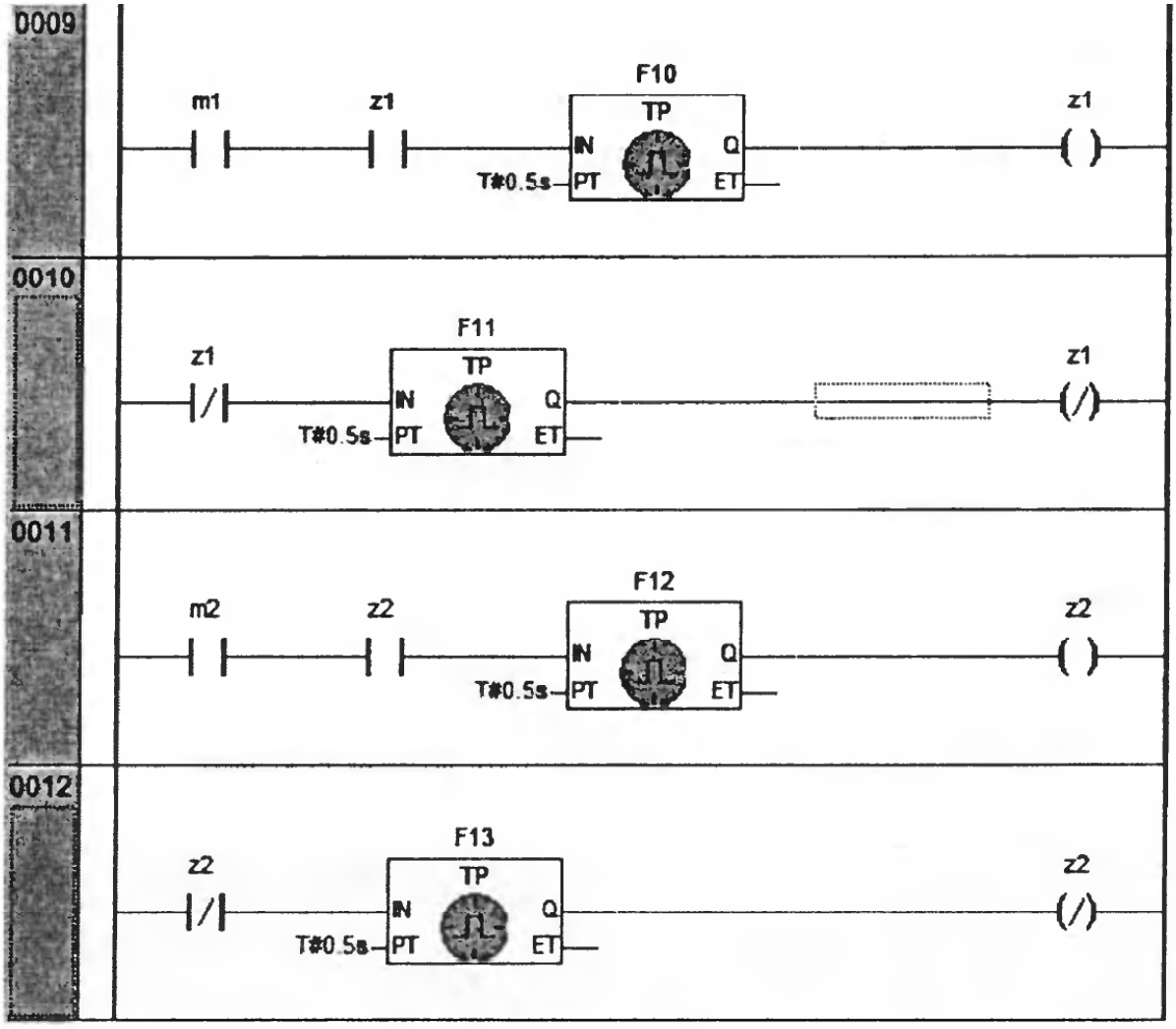


Рис. 5.17. Возможный вариант развития многоступенчатой схемы для получения мерцания стрелок с именами z1 и z2

Однако есть более компактный вариант решения этой задачи за счет применения нового FB – **BLINK**.

Для этого придется добавить библиотеку функциональных блоков. В уже знакомом окне организаторов объектов (рис. 3.6) выберем вкладку **Resources** и в ней найдем **Library Manager**, щелкнем 2ЛКМ на нем. Открывается окно библиотек, в котором находим окно выбора «библиотек» (рис. 5.18).

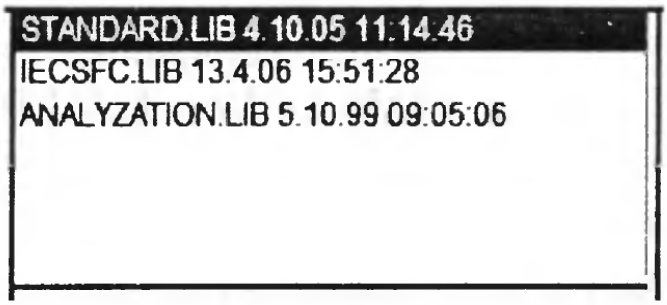


Рис. 5.18. Окно выбора библиотек

В области этого окна нажимаем 1ПКМ, выбираем **Additional Library** (рис. 5.19).

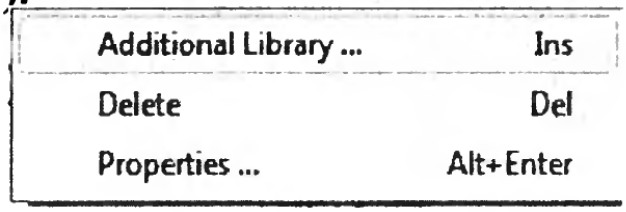


Рис. 5.19. Контекстное меню выбора библиотек

В открывшемся окне имеющихся библиотек находим Util.lib. Завершаем операцию нажатием кнопки **Открыть** (рис. 5.20).

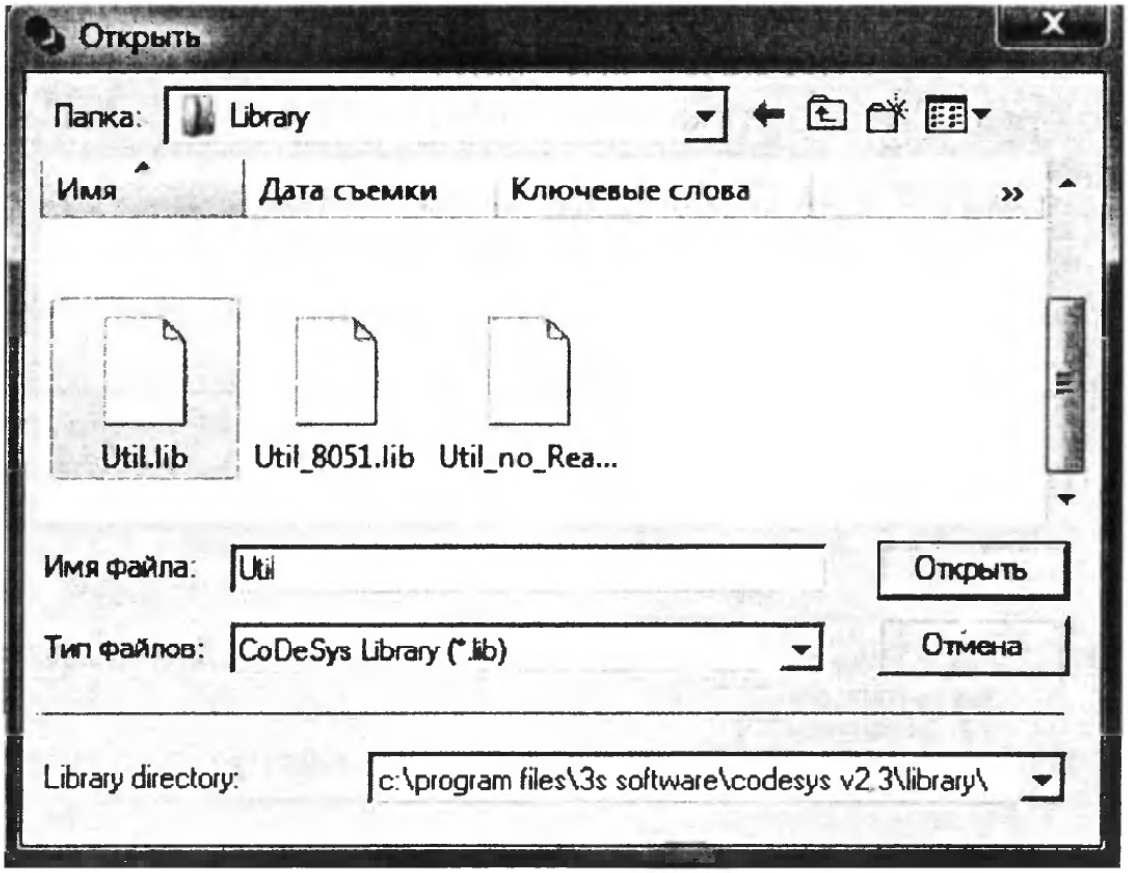


Рис. 5.20. Выбор новой библиотеки

Познакомимся с новым компонентом **BLINK** (рис. 5.21), который представляет собой генератор прямоугольных сигналов. Для того чтобы включить его в цепь, пользуемся уже привычной кнопкой добавления функциональных блоков К26. В открывшемся окне выбираем новую библиотеку, раздел **Signals – BLINK(FB)**. Генератор начинает работать при условии, что на его вход **ENABLE** поступит сигнал TRUE, после чего он начинает выдавать сигналы с выхода **OUT**. Параметры **TIMELOW** и **TIMEHIGH** типа **TIME**. Первый отвечает за время паузы, второй за продолжительность импульса. Уставка значений по этим входам выполняется по аналогии с параметром **PT** для таймеров (п. 4.8).

Применение этого FB делает схему более изящной (рис. 5.22).

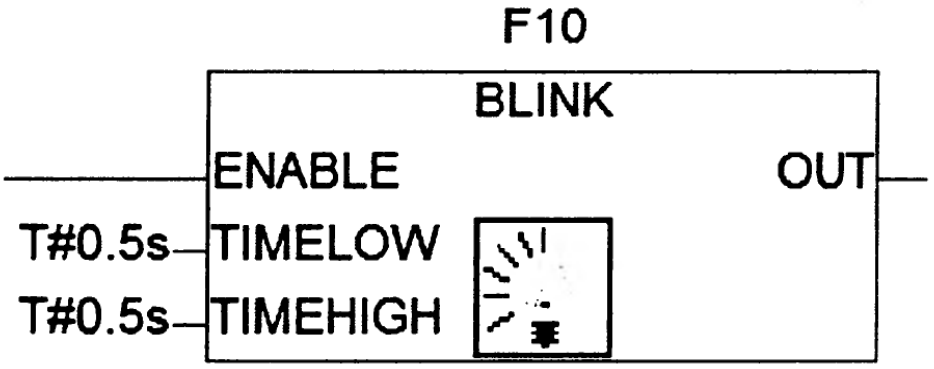


Рис. 5.21. Генератор прямоугольных импульсов BLINK

Внесем изменения в нашу схему, для этого вернемся в программу (окно организаторов проектов – **POU – PLC_PRG(PRG)**) и добавим две цепи (рис. 5.22).

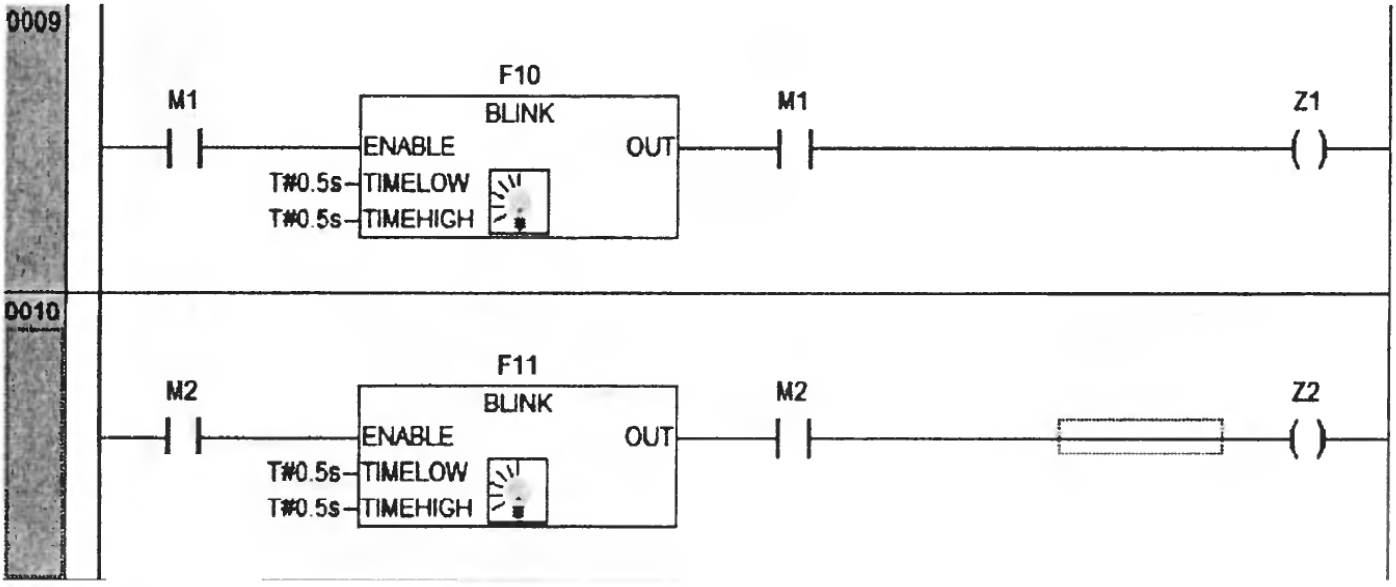


Рис. 5.22. Заключительные цепи программы

При переходе входа **ENABLE** в состояние **FALSE** значение выхода **OUT** определяется его состоянием в момент отключения входа. Так, если на выходе была 1, она и сохраняется. Поэтому следует поставить замыкающие контакты **M1** и **M2** до и после **F10** и **F11** соответственно.

Можно эти контакты оставить только после этих **FB**. Тогда **F10** и **F11** будут постоянно генерировать импульсы. Но катушки **Z1** и **Z2**, как и планировалось, станут «мерцать» только при включении **M1** и/или **M2.**

Прокомментируем созданные цепи.

Как уже отмечалось выше, переменная **M1** отвечает за работу транспортера **A**. В цепи 0009 добавляем переменную **Z1**, которая будет отвечать за создание пульсаций стрелок на транспортере **A** при срабатывании **M1.** Импульсы вырабатываются генератором **F10**. **Z1** является локальной переменной **VAR** для программы **PLC_PRG**, объявление которой является уже процессом привычным для читателя.

В цепь 0010 ввели локальную переменную **Z2**, отвечающую за пульсацию стрелок на транспортере **B**, переменной которого является **M2.** Импульсы вырабатываются генератором сигналов **F11.**

Теперь вернемся в окно, где создавали визуализацию (Организатор объектов – Visualizations – 2ЛКМ на **transporter**).

С помощью элемента **«Polygon»** над транспортерами создаем стрелки, которые будут сигнализировать о работе транспортеров. Выбираем категорию **Color**. В области **Color** необходимо поставить отметку над **NO frame color**. Нажимаем кнопку **Inside** в области **Alarm Color** и выбираем, например красный цвет. Зададим этим стрелкам значения переменных. Для стрелок, относящихся к транспортеру **A**, введем переменную **.PLC_PRG.Z1** *(не забываем точку).* Приставка «**.PLC_PRG**» означает, что переменная **Z1** является локальной для программы PLC_PRG.

Для стрелок второго транспортера соответственно задаем переменную **.PLC_PRG.Z2**.

В итоге получаем конечную визуализацию нашей СЛУ (рис. 5.23).

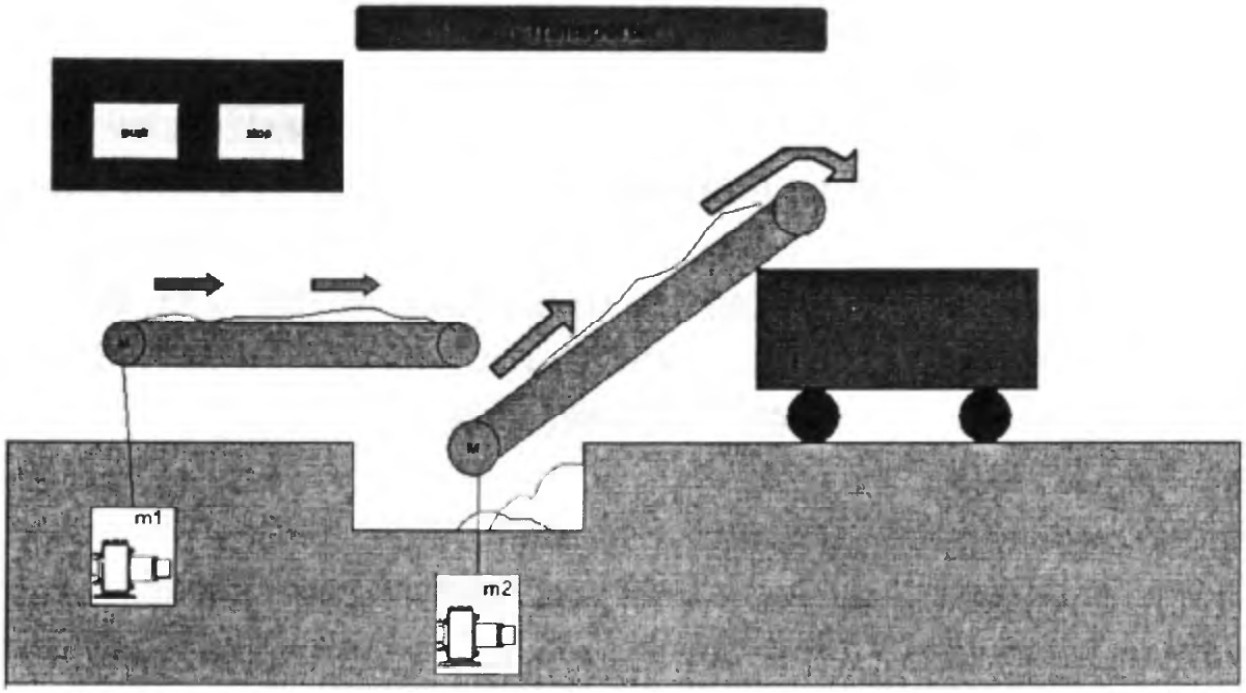


Рис. 5.23. Визуализация СЛУ

Более подробно читатель может познакомиться с процессом разработки визуализации и различными приемами ее создания в Дополнении к руководству пользователя по программированию ПЛК в CoDeSys 2.3 «ВИЗУАЛИЗАЦИЯ В CoDeSys», которое прилагается на диске вместе с программной средой или доступно на сайте www.owen.ru.

ЗАКЛЮЧЕНИЕ

Если читатель активно поработал с этой книгой, т. е. последовательно шаг за шагом решал предложенные задачи по исследованию основных компонентов многоступенчатых схем, освоил приемы их построения и проверки работоспособности в режиме эмуляции, переноса программы в ПЛК и получения визуализации технологического процесса на дисплее ПК, то можно рассчитывать на дальнейшее успешное его продвижение в этой области.

Естественно, авторы показали даже не дорогу, а тропиночку в мир ПЛК, надеясь, что увлеченный читатель на этом не остановится и пойдет дальше по пути освоения других языков и приемов программирования, позволяющих с применением новых FB, операторов, программ и функций решать более сложные задачи, в том числе с участием аналоговых сигналов.

СОКРАЩЕНИЯ И ТЕРМИНЫ

1ЛКМ или 2ЛКМ – один или два щелчка левой кнопкой манипулятора «мышь»;

1ПКМ – один щелчок правой кнопкой манипулятора «мышь»;

АЦП – аналого-цифровой преобразователь;

ИМ – исполнительный механизм;

ЛКМ – левая кнопка манипулятора «мышь»;

МЭК – международная электротехническая комиссия;

ПКМ – правая кнопка манипулятора «мышь»;

РКС – релейно-контактная схема;

СЛУ – система логического управления;

ЦАП – цифро-аналоговый преобразователь.

CoDeSys – Controllers Development System – современный инструмент для программирования контроллеров;

ET – вход таймеров для отсчета прошедшего времени (TIME).

FALSE – ложь (логический 0);

FB – функциональный блок;

LD – Ladder Diagram – релейные диаграммы;

PT – вход таймеров для установки времени (TIME);

RESET – сброс;

SCADA – (Supervisory Control And Data System) – система сбора данных и оперативного диспетчерского управления;

SET – установка;

STD – счетчик декрементный;

STU – счетчик инкрементный;

STUD – счетчик инкрементный/декрементный;

TON – таймер с задержкой включения;

TOF – таймер с задержкой отключения;

TP – таймер, формирующий одиночный импульс заданной по входу PT длительностью;

TRUE – истина (логическая 1).

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Карпов, Ю. Г. Теория автоматов : учебник для вузов. – СПб. : Питер, 2002. – 206 с.
2. Парр, Э. Программируемые контроллеры : руководство для инженера. – М. : Бином ; Лаборатория знаний, 2007 . – 516 с.
3. Петров, И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. – М. : СОЛОН-Пресс, 2004 . – 246 с.
4. www.owen.ru.
5. www.iec.ch.
6. www.3s-software.ru.

Учебное пособие

Минаев Игорь Георгиевич
Самойленко Владимир Валерьевич

Программируемые Логические Контроллеры

Практическое руководство для начинающего инженера

Главный редактор *И. А. Погорелова*
Заведующий издательским отделом *А. В. Андреев*
Редактор *А. Г. Сонникова*
Техническое редактирование и компьютерная верстка *С. А. Мельник*

Подписано в печать 04.06.2009. Формат 60x84 $^{1}/_{16}$. Усл. печ. л. 5,8.
Гарнитура «Таймс». Бумага офсетная. Печать офсетная.
Тираж 500. Заказ № 305.

Налоговая льгота – Общероссийский классификатор продукции ОК 005-93-953000

Издательство Ставропольского государственного аграрного университета
«АГРУС», 355017, г. Ставрополь, пер. Зоотехнический, 12.
Тел./факс (8652) 35-06-94.
E-mail: agrus2007@mail.ru; http://agrus.stgau.ru.

Отпечатано в типографии издательско-полиграфического комплекса СтГАУ «АГРУС»,
г. Ставрополь, ул. Мира, 302.