
Иванов В.Н.



**ПРОГРАММИРОВАНИЕ
ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ**



(Учебное пособие)

**СОЛОН-Пресс
Москва
2021**

УДК 681.5
ББК 32.96
И20



Иванов В.Н.

Программирование логических контроллеров. Учебное пособие. — М. СОЛОН-Пресс, 2021. — 356 с.

ISBN 978-5-91359-404-4

В учебном пособии рассмотрено программирование пользующихся широкой известностью в нашей стране логических контроллеров OWEN, ONI и Siemens LOGO! Рассмотрена работа с программным обеспечением Multisim, Logo! Soft Comfort, ONI PLR Studio, Owen Logic, Codesys.

При изложении материала автор постарался сохранить баланс между необходимым теоретическим минимумом и практикой программирования логических контроллеров. В процессе проведения лабораторных работ студенты имеют возможность поработать с «живыми» образцами программируемых контроллеров, в качестве которых использовались ONI PLR-S-CPU-1206, Owen ПР200 и LOGO! шестой и восьмой серий.

Доступный стиль изложения делает возможным использовать учебное пособие, как в высших, так и средних профессиональных учебных заведениях. Некоторые материалы учебного пособия могут использоваться для занятий в инженерных классах средней школы.

Автор Иванов Виктор Никитович, преподаватель высшей категории, кандидат технических наук.

По вопросам приобретения обращаться:

ООО «СОЛОН-Пресс»

Тел: (495) 617-39-64, (495) 617-39-65

E-mail: kniga@solon-press.ru, www.solon-press.ru

ISBN 978-5-91359-404-4

© СОЛОН-Пресс, 2021

© Иванов В.Н., 2021

Оглавление

ВВЕДЕНИЕ	6
ГЛАВА 1. ЭЛЕМЕНТЫ ТЕОРИИ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ.....	8
1.1. Типовая схема автоматического регулирования	8
1.2. Типовые динамические звенья	10
1.3. Соединение звеньев в САУ	18
1.4. Регуляторы в автоматическом управлении.....	21
1.5. Моделирование регуляторов в SimInTech	26
Контрольные вопросы и задания	34
ГЛАВА 2. ЭЛЕМЕНТЫ ЦИФРОВОЙ ТЕХНИКИ	37
2.1. Основные понятия алгебры логики	37
2.2. Законы и правила алгебры логики	40
2.3. Проектирование логической схемы.....	42
2.4. Проектирование релейно-контактных схем.....	52
Контрольные вопросы и задания	57
ГЛАВА 3. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ.....	59
3.1. Модульное исполнение	59
3.2. Внутренняя структура контроллера.....	61
3.3. Входы и выходы контроллера	63
3.4. Промышленные шины.....	68
Контрольные вопросы и задания	71
ГЛАВА 4. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ SIEMENS LOGO!	72
4.1. Общая характеристика	72
4.2. Базовые модули LOGO!	74
4.3. Модули расширения	78
4.4. Подключение внешних цепей	80
4.5. Программное обеспечение LOGO! Soft Comfort.....	82
4.5.1. Общие сведения	82
4.5.2. Пользовательский интерфейс.....	83
4.5.3. Построение коммутационной программы	88
4.5.4. Таймеры	93
4.5.5. Счетчики	105
4.5.6. Аналоговые функции	109

4.5.7. Тексты сообщений.....	127
4.5.8. Реализация циклического подключения выходов.....	132
4.5.9. Примеры управляющих программ в LOGO! Soft Comfort.....	140
ГЛАВА 5. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ONI	148
5.1. Номенклатура контроллеров ONI	148
5.2. Базовые контроллеры.....	150
5.3. Микро ПЛК ONI PLR-M.....	153
5.4. Программируемые реле ONI PLR-S	154
5.5. Программное обеспечение ONI PLR Studio.....	158
5.5.1. Пользовательский интерфейс.....	158
5.5.2. Примеры управляющих программ в ONI PLR Studio.....	161
Контрольные вопросы и задания	175
ГЛАВА 6. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ОВЕН	177
6.1. Общая характеристика программируемых логических контроллеров ..	177
6.2. Общая характеристика логических реле (ПЛР) ОВЕН.....	181
6.3. Монтаж электрических цепей	184
6.4. Среда программирования OWEN LOGIC	187
6.4.1. Пользовательский интерфейс.....	187
6.4.2. Создание нового проекта	191
6.4.3. Размещение компонентов на рабочем поле и создание коммутационной программы.....	192
6.4.4. Работа с панелью симуляции.....	196
6.4.5. Библиотека компонентов.....	198
6.4.6. Примеры управляющих программ в OWEN Logic	199
Контрольные вопросы и задания	210
ГЛАВА 7. ПРОГРАММИРОВАНИЕ ОВЕН В CODESYS.....	212
7.1. Установка Codesys на компьютер	212
7.2. Пользовательский интерфейс.....	212
7.3. Константы и переменные.....	214
7.4. Библиотеки	217
7.5. Обзор языков программирования	218
7.5.1. Язык ST	218
7.5.2. Примеры управляющих программ на языке ST	231
7.5.3. Язык LD	243
7.5.4. Язык SFC.....	254
Контрольные вопросы и задания	270
ГЛАВА 8. ПРОЕКТИРОВАНИЕ СИСТЕМЫ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ.....	272
8.1. Планирование системы логического управления	272
8.2. Разработка логической схемы.....	273

8.3. Разработка управляющей программы.....	275
8.4. Разработка электрической схемы.....	278

ГЛАВА 9. ЛАБОРАТОРНЫЕ РАБОТЫ.....279

9.1. Лабораторные работы с контроллером Siemens LOGO!	280
9.1.1. Описание лабораторного стенда	280
9.1.2. Лабораторная работа №1 «Тестирование таймеров».....	284
9.1.3. Лабораторная работа №2 «Тестирование аналоговых функций»	290
9.1.4. Лабораторная работа №3 «Автоматическая система импульсного регулирования температуры воздуха в помещении».....	296
9.1.5. Лабораторная работа №4 «Автоматическая система регулирования температуры воздуха в помещении с помощью ПИ-регулятора»	299
9.1.6. Лабораторная работа №5 «Система автоматического регулирования скорости двигателя постоянного тока с помощью П-, И-, ПИ-регуляторов»	304
9.1.7. Лабораторная работа №6 «Подсветка взлетной полосы»	309
9.2. Лабораторные работы с контроллером ONI	321
9.2.1. Лабораторная работа №7 «Многоканальный пожарный извещатель».....	321
9.2.2. Лабораторная работа №8 «Сигнал SOS»	327
9.2.3. Лабораторная работа №9 «Управление двумя вентиляторами» ..	329
9.3. Лабораторные работы с контроллером ОВЕН.....	333
9.3.1. Лабораторная работа №10 «Светофор»	333
9.3.2. Лабораторная работа №11 «Элементы автоматики»	337
9.3.3. Лабораторная работа №12 «Охранная сигнализация с ИК датчиком движения»	343
9.3.4. Лабораторная работа №13 «Охранная сигнализация с инфракрасным и микроволновым датчиками движения»	347
9.3.5. Лабораторная работа №14 «Подключение к контроллеру силовой нагрузки»	351

ЛИТЕРАТУРА.....355

ВВЕДЕНИЕ

Первый программируемый логический контроллер (ПЛК) появился в 1968 году. Это было громоздкое и дорогое устройство. Но начиная с 1972 года, с появлением мощных и недорогих микроконтроллеров, рынок ПЛК начал стремительно развиваться. В России ПЛК начали активно применяться сравнительно недавно. Они пришли на смену релейно-контактным схемам управления, собранным на дискретных компонентах жесткой логики. Принципиальное отличие ПЛК от релейных схем заключается в том, что в ПЛК все алгоритмы управления реализованы программно. При этом надежность работы системы не зависит от ее сложности. Использование ПЛК позволяет заменить одним логическим контроллером любое количество отдельных элементов релейной автоматики, что увеличивает надежность системы, минимизирует затраты на ее тиражирование, ввод в эксплуатацию и обслуживание. С появлением ПЛК произошел революционный прорыв в сфере автоматизации. ПЛК используются в системах противоаварийной защиты, в станках с ЧПУ, в системах управления дорожным движением, в системах коммунального хозяйства, в медицинском оборудовании, в робототехнике, в системах связи, в системах автоматизации технологических процессов. Этот неполный список применения логических контроллеров в настоящее время продолжает только расти. До сих пор остается много отраслей промышленности, куда программируемые логические контроллеры начинают только проникать, например, ПЛК находят все более широкое применение в области создания интеллектуальных устройств.

Первые логические контроллеры имели свои системы команд, разработанные только для данного контроллера. Программа, написанная для одного контроллера, не могла использоваться для другого контроллера. В 1993 году был принят стандарт ИЕС (МЭК) 1131-3, (позже был переименован в ИЕС 61131-3) который систематизировал языки программирования логических контроллеров. Стандарт ИЕС 61131-3 установил пять языков программирования логических контроллеров:

- ST (Structured Text) – структурированный текст;
- SFC (Sequential Function Chart) – последовательные функциональные схемы;
- FBD (Function Block Diagram) – диаграммы функциональных блоков;
- LD (Ladder Diagram) – релейно-контактные схемы или релейные диаграммы;
- IL (Instruction List) – список инструкций.

Языки SFC, FBD, LD являются графическими языками, языки IL и ST являются текстовыми языками. Появление стандарта способствовало расширению областей применения контроллеров, а также обеспечило повторяемость и взаимозаменяемость программ, написанных для разных контроллеров.

В учебном пособии рассмотрено программирование широко известных, применяемых в нашей стране контроллеров ONI, OWEN и Siemens LOGO! Главы

1...3 посвящены общим принципам автоматического регулирования, общим принципам построения логических выражений и логических схем, являющихся прообразом коммутационной программы на языке FBD, и общим принципам построения логических контроллеров. В главах 4...7 изложено описание линеек программируемых контроллеров Siemens LOGO!, ONI, OWEN. Рассмотрено программирование логических контроллеров на языках LD, ST, FBD, CFC в системах программирования LOGO! Soft Comfort, ONI PLR Studio, OWEN Logic, CoDeSys. В главе 8 полученные знания используются для проектирования системы автоматического управления освещением и охранной сигнализацией коттеджа. В главе 9 студентам предоставляется возможность поработать с «живыми» контроллерами. Особое внимание в этой главе уделено вопросу создания электрических схем соединений и вопросу переноса вновь созданной программы из компьютера в контроллер.

Владение методикой программирования логических контроллеров является одним из необходимых навыков современного выпускника электротехнического учебного заведения. Специалисты, владеющие программированием контроллеров, кроме всего прочего, повышают свои шансы при трудоустройстве на интересную и высокооплачиваемую работу.

При написании учебного пособия было использовано программное обеспечение: ПО «Multisim», предоставленное представительством компании National Instruments, ПО «LOGO! Soft Comfort», предоставленное представительством компании Siemens, ПО «SimInTech», предоставленное компанией «ЗВ-сервис», ПО Owen Logic и Codesys, предоставленные компанией OWEN, ПО ONI PLR Studio компании ONI.

Автор благодарит компанию OWEN за предоставленный для учебных целей комплект оборудования, которое было использовано при подготовке лабораторных работ с программируемым реле OWEN.



ГЛАВА 1. ЭЛЕМЕНТЫ ТЕОРИИ АВТОМАТИЧЕСКОГО РЕГУЛИРОВАНИЯ

1.1. Типовая схема автоматического регулирования

Поскольку программируемые логические контроллеры широко используются в системах автоматического управления и регулирования, приведем некоторые сведения из теории этих систем и ознакомимся с некоторыми понятиями из этой области.

Функциональная схема системы автоматического регулирования (САР) представлена на рис. 1.1.

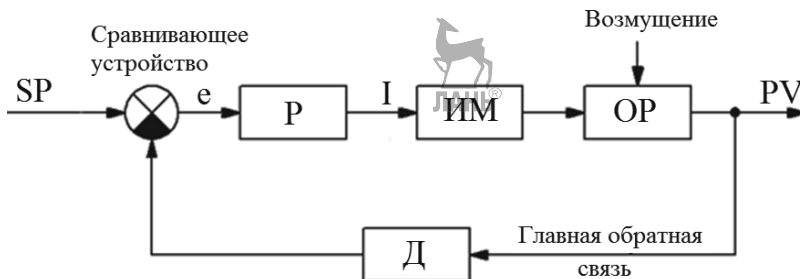


Рис. 1.1. Функциональная схема САР

ОР – объект регулирования (печь, электродвигатель, резервуар и др.) или какой-либо управляемый технологический процесс;

Р – регулятор;

ИМ – исполнительный механизм (шаговый двигатель, сельсины, электромагнит и др.);

Д – датчик (термопара, терморезистор, датчик уровня, скорости и др.);

SP (Set Process) – заданное значение управляемого параметра процесса;

PV (Process Value) – фактическое значение управляемого параметра процесса;

$e = SP - PV$ – ошибка регулирования;

I – регулирующая переменная.

В качестве примера рассмотрим схему регулирования температуры в помещении. **Объектом регулирования** является помещение, в котором поддерживается постоянная температура. Температура в помещении является **регулируемой переменной**. Требуемая температура в помещении является **заданным значением** регулируемой переменной и называется **уставкой**. При открытии окна температура в помещении может повышаться или понижаться, т.е. **фактическое значение** регулируемой переменной не равно заданному значению. Открытое окно назовем **переменной возмущения**. После открытия окна

температура в помещении должна быть скорректирована. Функцию управления температурой выполняет **регулятор**. Регулятор с помощью **датчика** считывает реальную температуру, сравнивает ее с заданным значением (уставкой), и выдает команду **исполнительному механизму** на компенсирование отклонения температуры от заданного значения. В качестве исполнительного механизма используется электронагреватель с реостатом. С помощью реостата изменяется ток через нагреватель, и, тем самым, изменяется количество тепла, отдаваемого электронагревателем. Ток через электронагреватель является **регулирующей переменной**. Задача регулятора – выработать такой управляющий сигнал, а в нашем случае так изменить ток через нагреватель, чтобы свести ошибку контура регулирования к нулю оптимальным образом. **Ошибкой контура** регулирования будет разность заданного и фактического значений регулируемой переменной. Иными словами, ошибка контура – это отклонение регулируемой переменной от заданного значения.

$$e = SP - PV \quad (1.1)$$

Пусть заданное значение температуры в помещении 20°C , а фактическая температура 18°C , тогда ошибка контура $e = 20 - 18 = 2^{\circ}\text{C}$. Ошибка контура имеет положительное значение. Это значит, что понижение температуры в помещении необходимо компенсировать повышением тока через нагреватель. Если фактическая температура в помещении 22°C , тогда ошибка контура $e = 20 - 22 = -2^{\circ}\text{C}$ имеет отрицательное значение. Это значит, что повышение температуры в помещении необходимо компенсировать снижением тока через нагреватель. Таким образом, между входом и выходом должна существовать обратная связь, и эта связь должна быть отрицательной: уменьшение регулируемой переменной на выходе должно компенсироваться увеличением регулирующей переменной на входе и наоборот. В этом и заключается смысл отрицательной обратной связи.

Итак, необходимо определить, по какому закону должен изменяться электрический ток через электронагреватель, чтобы обеспечить скорейший возврат фактической температуры в помещении к заданному значению. Простейший вариант – регулирование температуры путем включения и выключения электронагревателя. Это дискретная схема управления. График изменения температуры в помещении и тока через нагреватель в зависимости от времени показаны на рис. 1.2. При таком изменении регулирующей величины невозможно достичь точного значения уставки, т.е. заданной температуры в помещении, она будет колебаться в некотором диапазоне. Такая система управления используется, например, для управления температурой в холодильнике. Рассмотрим другой пример. Предположим, для управления скоростью автомобиля используется автопилот (у автомобиля он называется круиз-контроль), в который заложен дискретный способ управления скоростью. Тогда, чтобы держать скорость, скажем 100 км/ч , автомобиль разгоняется до скорости 105 км/ч , затем двигатель отключается, и скорость автомобиля падает до 95 км/ч , затем двигатель запускается, и автомобиль опять разгоняется до 105 км/ч ,

затем скорость падает до 95 км/ч и т.д. Понятно, что такая система управления является некомфортной и потенциально аварийной. В данном случае дискретная система управления неприменима.

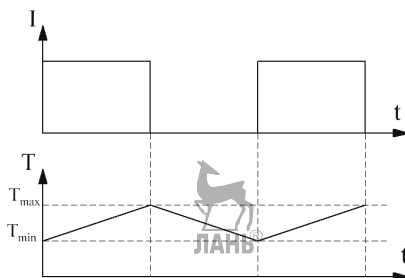


Рис. 1.2. Диаграмма дискретного управления

Для получения на выходе точного значения управляемого параметра применяют систему регулирования, в которой выходной сигнал регулятора изменяется по непрерывному закону. В цифровых системах регулирования регулятор представляет собой вычислительное устройство, которое вычисляет значение регулирующего параметра. В нашем примере регулятор определяет величину тока через электронагреватель в соответствии с заданным законом управления.

Различают три закона изменения регулирующего параметра, в соответствии с которыми различают три основных типа регуляторов:

- регулятор пропорционального действия (П-регулятор);
- регулятор интегрального действия (И-регулятор);
- регулятор дифференциального действия (Д-регулятор).

Как правило, реально используемый регулятор представляет собой комбинацию перечисленных типов регуляторов. В частности, ПИ-регулятор представляет собой комбинацию П-регулятора и И-регулятора. Самым эффективным является ПИД-регулятор, который представляет собой комбинацию П, И, Д-регуляторов.

Работу систем автоматики, в том числе регуляторов, можно смоделировать на компьютере. Для этого применяются различные программы, наиболее известные из которых: Vissim, Matlab Simulink, ПК МВТУ, SimInTech. В учебном пособии будет использоваться программа SimInTech (Simulation In Technic), описание которой приведено в [3]. Это программа российской разработки, является дальнейшим развитием программы ПК МВТУ.

1.2. Типовые динамические звенья

Перед тем, как изготавливать промышленную установку с автоматическим управлением, необходимо убедиться в ее работоспособности, а для этого необходимо проверить работоспособность установки на математической модели. Математическая модель системы управления состоит из уравнений, которые

описывают работу всех блоков системы. Эти уравнения могут быть как алгебраическими, так и дифференциальными.

Прежде, чем составлять систему уравнений разрабатывают функциональную и структурную схемы системы управления. Функциональная схема состоит из блоков, названия которых указывают на выполняемые функции, например, датчик, регулятор, исполнительный элемент и т.д. Структурная схема системы управления может быть получена из функциональной схемы, если известны передаточные функции и параметры всех элементов, входящих в ее состав. Структурная схема представляет собой графическую форму математической модели автоматической системы. Блоки структурной схемы называются звеньями, каждое из которых отображает алгоритм преобразования сигнала – математическую или логическую операцию. В зависимости от физических свойств различают статические и динамические звенья. У статического звена значение выходного сигнала не зависит от времени. Связь между входным и выходным сигналами статического звена обычно описывается алгебраической функцией. Для описания динамического звена, в отличие от статического, используются уравнения, в которых входные и выходные величины являются функциями времени. Динамические звенья, которые описываются обыкновенными дифференциальными уравнениями первого и второго порядка называются **типовыми динамическими звеньями**.

При решении дифференциальных уравнений используется преобразование Лапласа, и решение записывается в виде передаточных функций. **Передаточной функцией** называется отношение выходного сигнала к входному сигналу преобразованного по Лапласу дифференциального уравнения. Исходное дифференциальное уравнение называется **оригиналом**, а преобразованное и записанное в операторной форме – **изображением**. Преобразование Лапласа применимо только для линейных дифференциальных уравнений с постоянными коэффициентами. Обозначим входную величину системы управления, как $x(t)$, а выходную – $y(t)$. Суть преобразования Лапласа заключается в замене функций вещественных переменных $x(t)$, $y(t)$ на функции комплексных переменных $X(s)$, $Y(s)$, где s – оператор Лапласа (комплексное число $s = \pm m \pm in$). Входные и выходные параметры оригинала и изображения связаны между собой интегралом Лапласа

$$X(s) = \int_0^{\infty} x(t)e^{-st} dt$$

$$Y(s) = \int_0^{\infty} y(t)e^{-st} dt$$

$x(t)$, $y(t)$ – функции-оригиналы;

$X(s)$, $Y(s)$ – функции-изображения.

(в технической литературе часто можно встретить обозначение оператора Лапласа буквой p вместо s).

При нулевых начальных условиях переход от оригиналов к изображениям происходит по формулам, представленным в таблице 1.1.



Таблица 1.1.

Оригинал	Изображение
$1(t)$	$1/s$
$f(t)$	$F(s)$
$af(t)$	$aF(s)$
$af_1(t) + bf_2(t)$	$aF_1(s) + bF_2(s)$
$\frac{d}{dt}(f(t))$	$sF(s)$
$\frac{d^2}{dt^2}(f(t))$	$s^2F(s)$
$\int f(t)dt$	$\frac{1}{s}F(s)$
$f(t - \tau)$	$e^{-s\tau}F(s)$

Используя данные таблицы 1.1, можно легко перейти к операторной форме записи дифференциальных уравнений и получить аналитические зависимости для передаточных функций.

Пример 1.1. Пусть система описывается дифференциальным уравнением:

$$a_0 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_2 y = kx(t)$$

$x(t)$ – входная величина;

$y(t)$ – выходная величина.

Необходимо найти передаточную функцию $W(s)$ при следующих исходных данных

$$k = 1; a_0 = 1; a_1 = 3; a_2 = 2$$

Решение.

Применим к дифференциальному уравнению преобразование Лапласа.

Получим

$$(a_0 s^2 + a_1 s + a_2)Y(s) = kX(s)$$

Из этого уравнения, с учетом исходных данных, находим значение передаточной функции

$$W(s) = \frac{Y(s)}{X(s)} = \frac{k}{a_0 s^2 + a_1 s + a_2} = \frac{1}{s^2 + 3s + 2}$$

Как следует из рассмотренного примера, с помощью преобразования Лапласа осуществляется переход от дифференциального уравнения к алгебраическому уравнению, решать которое гораздо проще. И в этом заключается большое практическое значение преобразования Лапласа.

К простейшим динамическим звеньям относятся:

- пропорциональное (усилительное);
- инерционное (апериодическое 1-го порядка);
- интегрирующее;
- дифференцирующее;
- колебательное (апериодическое 2-го порядка);

– запаздывающее.

Пропорциональное звено. Уравнение взаимосвязи входного и выходного сигналов пропорционального звена является алгебраическим и имеет следующий вид

$$y(t) = kx(t)$$

k – коэффициент пропорциональности.

Передаточная функция пропорционального звена запишется в виде

$$W(s) = \frac{Y(s)}{X(s)} = k$$

Пропорциональное звено мгновенно (без инерции) реагирует на возмущающее воздействие. График изменения выходной величины от времени называется **кривой разгона**. Из рис. 1.3 видно, что кривая разгона (выходная величина) пропорционального звена имеет постоянное значение, равное коэффициенту усиления k .

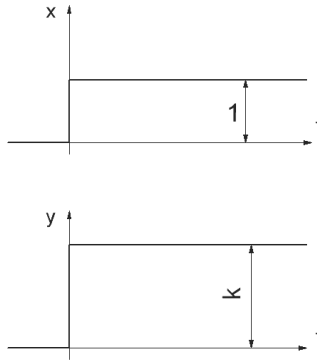


Рис. 1.3. Входной сигнал и кривая разгона пропорционального звена

Инерционное (апериодическое 1-го порядка) звено. Апериодическое звено описывается дифференциальным уравнением

$$T_0 \frac{dy(t)}{dt} + y(t) = kx(t)$$

T_0 – постоянная времени;

k – коэффициент передачи.

Перейдем в дифференциальном уравнении к операторной форме записи

$$T_0 s Y(s) + Y(s) = kX(s)$$

Перепишем уравнение в виде

$$Y(s)(T_0 s + 1) = kX(s)$$

отсюда получаем выражение для передаточной функции

$$W(s) = \frac{Y(s)}{X(s)} = \frac{k}{T_0 s + 1}$$

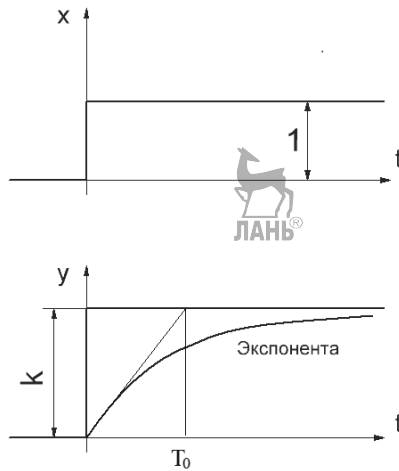


Рис. 1.4. Входной сигнал и кривая разгона инерционного звена

Кривая разгона инерционного звена показана на рис. 1.4. Она представляет собой экспоненту. Установившееся значение, к которому стремится экспонента, равно k . Касательная к экспоненте пересекает линию установившегося значения в точке $t = T_0$. Можно показать, что время переходного процесса (время регулирования) для этого графика приблизительно в три раза превышает постоянную времени T_0 . **Время регулирования** – это время, после которого разница между текущим и установившимся значениями не будет превышать некоторой, наперед заданной малой величины Δ . Как правило, Δ задается равной 5% от установившегося значения.

Интегрирующее звено. Дифференциальное уравнение интегрирующего звена имеет вид

$$T \frac{dy(t)}{dt} = x(t)$$

Операторная форма записи этого уравнения

$$TsY(s) = X(s)$$

Передаточная функция имеет вид

$$W(s) = \frac{Y(s)}{X(s)} = \frac{1}{Ts}$$

Кривая разгона интегрирующего звена приведена на рис. 1.5



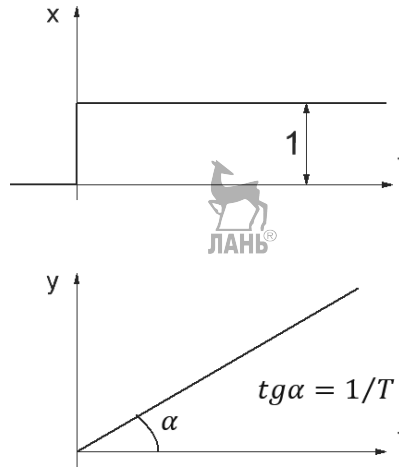


Рис. 1.5. Входной сигнал и кривая разгона интегрирующего звена

По кривой разгона можно определить коэффициент T в передаточной функции интегрирующего звена

$$T = 1/tg\alpha = arctg\alpha$$

В литературе передаточную функцию интегрирующего звена иногда записывают в виде

$$W(s) = \frac{k}{s}, \quad \text{где } k = \frac{1}{T}$$

Дифференцирующее звено. Дифференцирующее звено реагирует не на изменение входной величины, а на изменение ее производной, то есть на тенденцию ее развития. Поэтому говорят, что дифференцирующее звено обладает упреждающим, прогнозирующим действием. С помощью этого звена можно ускорить реакцию системы на возмущающее воздействие. Различают идеальное и реальное дифференцирующее звено. Рассмотрим сначала идеальное дифференцирующее звено, уравнение которого имеет вид

$$y(t) = k \frac{dx(t)}{dt}$$

Операторная форма записи этого дифференциального уравнения имеет вид

$$Y(s) = ksX(s)$$

Из этого уравнения находим передаточную функцию идеального дифференцирующего звена

$$W(s) = \frac{Y(s)}{X(s)} = ks$$

Идеальное дифференцирующее звено является технически нереализуемым устройством и представляет чисто теоретический интерес, поэтому рассмотрим реальное дифференцирующее звено. Оно описывается дифференциальным уравнением

$$T_0 \frac{dy(t)}{dt} + y(t) = k \frac{dx(t)}{dt}$$

Операторная форма записи этого уравнения

$$T_0 s Y(s) + Y(s) = k s X(s)$$

Отсюда можно получить аналитическое выражение для передаточной функции реального дифференцирующего звена, которое называется еще инерционным дифференцирующим звеном.

$$W(s) = \frac{Y(s)}{X(s)} = \frac{ks}{T_0 s + 1}$$

Фактически, это последовательное соединение идеального дифференцирующего звена и апериодического звена.

Кривая разгона реального дифференцирующего звена показана на рис. 1.6

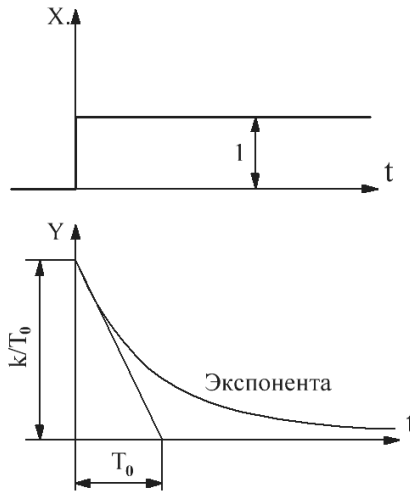


Рис. 1.6. Входной сигнал и кривая разгона реального дифференцирующего звена

Колебательное (апериодическое 2-го порядка) звено. Дифференциальное уравнение колебательного звена имеет вид

$$T_1 \frac{d^2 y(t)}{dt^2} + T_2 \frac{dy(t)}{dt} + y(t) = kx(t)$$

Операторная форма дифференциального уравнения запишется в виде

$$T_1 s^2 Y(s) + T_2 s Y(s) + Y(s) = kX(s)$$

Находим отсюда передаточную функцию

$$W(s) = \frac{k}{T_1 s^2 + T_2 s + 1}$$

Типовая кривая разгона колебательного звена показана на рис. 1.7

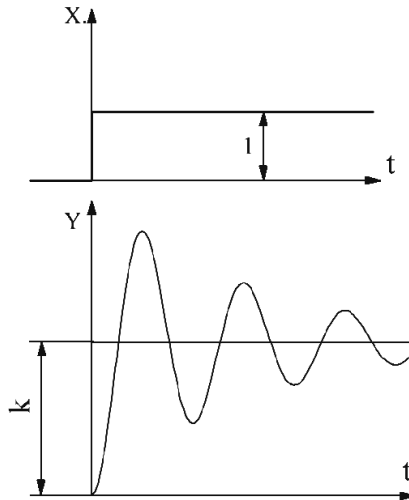


Рис. 1.7. Входной сигнал и кривая разгона колебательного звена

Запаздывающее звено. Представим себе трубу, через которую вентилятор прокачивает воздух (рис. 1.8). В начале трубы установлен нагреватель, а температура воздуха измеряется датчиком в точке А.

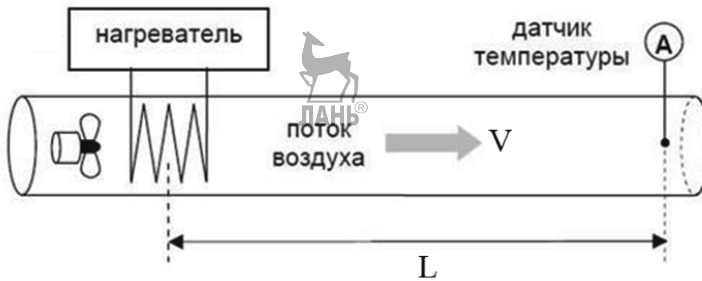


Рис. 1.8. Динамическое звено запаздывания

Изменение температуры воздуха датчик обнаружит не сразу, а через время $\tau = L/V$, где L – длина трубы, V – скорость потока воздуха. В этом случае говорят, что в системе есть транспортное **запаздывание** на величину τ .

Запаздывание в системе просто сдвигает сигнал вправо на временной оси, не меняя его формы. Математически это можно записать следующим образом

$$y(t + \tau) = x(t)$$

Если переписать это уравнение в виде

$$y(t) = x(t - \tau)$$

то операторная форма записи уравнения примет вид

$$Y(s) = e^{-s\tau} X(s)$$

Отсюда находим передаточную функцию

$$W(s) = \frac{Y(s)}{X(s)} = e^{-s\tau}$$

Кривая разгона запаздывающего звена показана на рис. 1.9

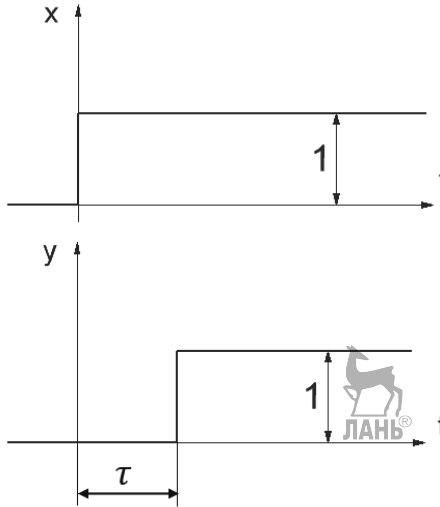


Рис. 1.9. Входной сигнал и кривая разгона запаздывающего звена

1.3. Соединение звеньев в САУ

Реальная система автоматического управления может состоять из нескольких типовых звеньев, сложным образом соединенных друг с другом. Однако любую сложную систему можно разбить на отдельные блоки, включающие три типа соединений: последовательное, параллельное, и соединение с обратной связью.

Последовательное соединение звеньев. Последовательное соединение звеньев показано на рис. 1.10.

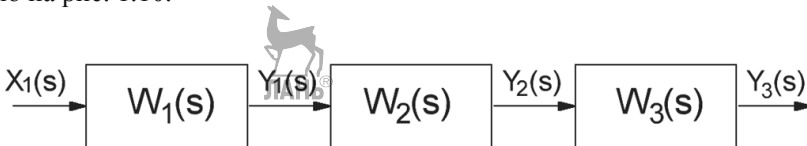


Рис. 1.10. Последовательное соединение звеньев

Передаточная функция всей системы, по определению, запишется следующим образом

$$W(s) = \frac{Y_3(s)}{X_1(s)} \tag{1.2}$$

Для каждого звена можно записать свои передаточные функции

$$W_1(s) = \frac{Y_1(s)}{X_1(s)} \text{ или } Y_1(s) = X_1(s)W_1(s) \quad (1.3)$$

$$W_2(s) = \frac{Y_2(s)}{Y_1(s)} \text{ или } Y_2(s) = Y_1(s)W_2(s) \quad (1.4)$$

$$W_3(s) = \frac{Y_3(s)}{Y_2(s)} \text{ или } Y_3(s) = Y_2(s)W_3(s) \quad (1.5)$$

Подставляя последовательно (1.5), (1.4), (1.3) в (1.2) получим

$$W(s) = W_1(s) \cdot W_2(s) \cdot W_3(s) \quad (1.6)$$

Таким образом, передаточная функция последовательно соединенных звеньев равна произведению передаточных функций отдельных звеньев.

Приведем без вывода формулы для соединений двух других типов звеньев.

Пример 1.2. Найти передаточную функцию двух последовательно соединенных инерционных звеньев:

$$W_1(s) = \frac{k_1}{T_1s + 1}; \quad W_2(s) = \frac{k_2}{T_2s + 1}$$

Решение. Передаточная функция двух последовательно соединенных звеньев равна произведению передаточных функций этих звеньев

$$\begin{aligned} W(s) = W_1(s) \cdot W_2(s) &= \frac{k_1}{T_1s + 1} \cdot \frac{k_2}{T_2s + 1} = \frac{k_1 k_2}{(T_1s + 1) \cdot (T_2s + 1)} = \\ &= \frac{k_1 k_2}{T_1 T_2 s^2 + (T_1 + T_2)s + 1} \end{aligned}$$

В результате получили типовое колебательное звено.

Параллельное соединение звеньев показано на рис. 1.11

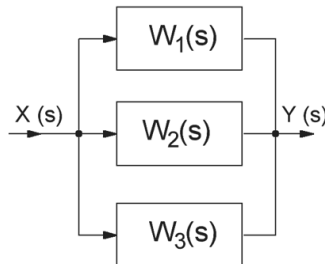


Рис. 1.11. Параллельное соединение звеньев

Передаточная функция такого соединения будет равна сумме передаточных функций отдельных звеньев

$$W(s) = W_1(s) + W_2(s) + W_3(s) \quad (1.7)$$

Соединение звеньев с обратной связью, показано на рис. 1.12.

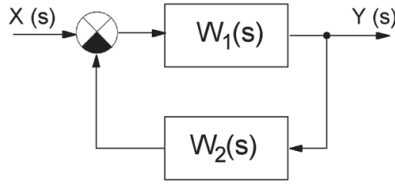


Рис. 1.12. Соединение звеньев с обратной связью

Передаточная функция такого соединения имеет вид

$$W(s) = \frac{W_1(s)}{1 \pm W_1(s)W_2(s)} \quad (1.8)$$

Знак (+) соответствует отрицательной обратной связи, знак (-) соответствует положительной обратной связи.

Если второе звено в цепи обратной связи отсутствует, то выражение (1.8) запишется в виде

$$W(s) = \frac{W_1(s)}{1 \pm W_1(s)}$$

Пример 1.3. Задана структурная схема САР (рис. 1.13), передаточные функции имеют вид

$$W_1(s) = k_1; \quad W_2(s) = \frac{k_2}{s}; \quad W_3(s) = \frac{k_3}{T_3 s + 1}; \quad W_4(s) = k_4$$

Найти эквивалентную передаточную функцию этой схемы.

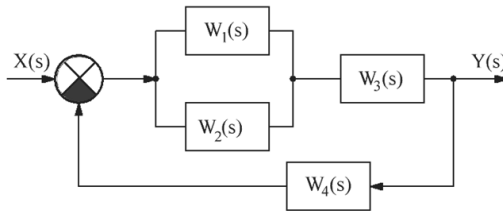


Рис. 1.13. Структурная схема САР

Решение. Сначала определим эквивалентную передаточную функцию в общем виде

$$W(s) = \frac{(W_1(s) + W_2(s)) \cdot W_3(s)}{1 + (W_1(s) + W_2(s)) \cdot W_3(s) \cdot W_4(s)}$$

Теперь подставим сюда значения передаточных функций

$$W(s) = \left(k_1 + \frac{k_2}{s}\right) \cdot \left(\frac{k_3}{T_3s + 1}\right) / \left(1 + \left(k_1 + \frac{k_2}{s}\right) \cdot \left(\frac{k_3}{T_3s + 1}\right) \cdot k_4\right)$$

После преобразований получим

$$W(s) = \frac{k_1k_3s + k_2k_3}{T_3s^2 + (k_1k_3k_4 + 1)s + k_2k_3k_4}$$

1.4. Регуляторы в автоматическом управлении

П-регулятор. Регулятор пропорционального действия изменяет регулирующую переменную пропорционально ошибке контура.

$$I(t) = k_P \cdot e(t) \tag{1.9}$$

$I(t)$ – выходной сигнал регулятора;

k_P – коэффициент пропорциональности (усиление П-регулятора);

$e(t)$ – ошибка регулирования;

t – время.

На рис. 1.14 показано скачкообразное изменение регулируемого параметра (например, температуры) и скачкообразное изменение регулирующего параметра П-регулятора (например, электрического тока в нагревателе).

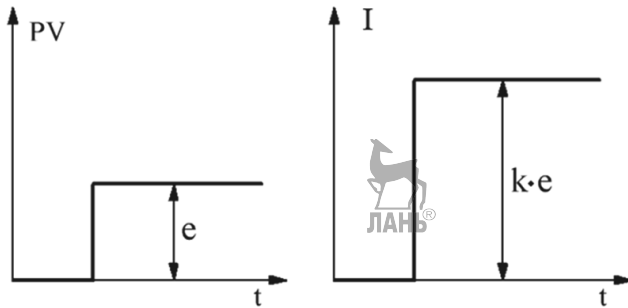


Рис. 1.14. Управляющий сигнал П-регулятора

Из рис.1.14 следует, что при возникновении в контуре управления ошибки «е», вырабатывается регулирующий сигнал, превосходящий эту ошибку в k раз, т.е. при скачкообразном изменении температуры так же скачкообразно изменяется значение тока, текущего через электронагреватель.

На рис. 1.15 показан пример механического пропорционального регулятора. Через входную трубу в бак поступает вода. Уровень воды отслеживается

поплавок. Если расход воды через выходную трубу большой, то поплавок опускается, задвижка поднимается, и приток воды в бак увеличивается. Если расход через выходную трубу уменьшается, то поплавок поднимается, а задвижка опускается и перекрывает трубу. Регулирующей величиной является положение задвижки. Фактическим значением параметра является уровень воды в баке. Коэффициент пропорциональности (усиление регулятора) зависит от положения опоры. Смещение опоры по горизонтальной оси будет приводить к изменению коэффициента пропорциональности в формуле (1.9), который равен $k_p = L_2/L_1$

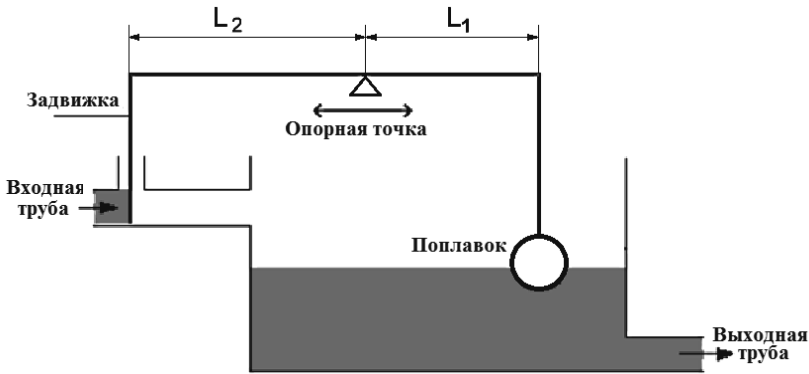


Рис. 1.15. Пример пропорционального регулятора

Преимуществом пропорционального регулятора являются быстрота реакции и простая конструкция. Однако он имеет существенный недостаток, который состоит в том, что управляемая величина стабилизируется на значении, которое отличается от заданного. Будет иметь место статическая ошибка, т.е. фактическое значение регулируемой величины всегда будет отличаться от заданного значения. Чем больше коэффициент усиления, тем меньше статическая ошибка. Но рост этого коэффициента может привести к автоколебаниям в системе и, в конечном итоге, к потере устойчивости.

И-регулятор. Регулятор интегрального действия вырабатывает регулирующий сигнал, пропорциональный ошибке контура «е» и времени «t». И-регулятор реагирует на ошибку контура регулирования с задержкой. Для того, чтобы рассчитать регулирующий сигнал в заданный момент времени, необходимо вычислить значение интеграла за предыдущие моменты времени (т.е. учесть ошибки контура в предыдущие моменты времени)

$$I(t) = k_i \cdot \int_0^t e(\tau) d\tau \quad (1.10)$$

k_i – коэффициент усиления И-регулятора;

τ – переменная интегрирования (принимает значения от 0 до t).

Если на вход регулятора интегрального действия поступает постоянный сигнал (ступенька), то изменение выходного значения происходит непрерывно до тех пор, пока отклонение системы не будет компенсировано. На рис. 1.16 показано непрерывное нарастание регулируемого сигнала И-регулятора.

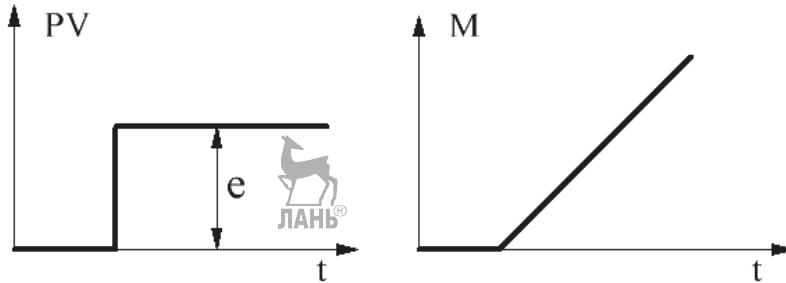


Рис. 1.16. Управляющий сигнал И-регулятора

И-регулятор не только способствует сокращению времени регулирования, Другое его важное достоинство заключается в том, что И-регулятор убирает статическую ошибку, присущую П-регулятору. Однако в чистом виде И-регулятор применяется редко, так как имеет склонность к нестабильности и слишком медленно реагирует на быстрые изменения.

Д-регулятор. Дифференциальный регулятор учитывает скорость изменения ошибки от времени, т.е. отслеживает интенсивность нарастания или убывания ошибки регулирования.

$$I(t) = k_d \frac{de(t)}{dt} \quad (1.11)$$

k_d – коэффициент усиления дифференциального регулятора.

Поскольку производная от константы равна нулю, этот регулятор не работает там, где ошибка носит постоянный характер и, наоборот, действует тем эффективнее, чем сильнее происходит изменение ошибки регулирования. Изменение ошибки сильнее всего происходит в начальный момент времени. Вследствие этого Д-регулятор начинает действовать сразу и с максимальной эффективностью, когда отклонение регулируемой величины еще не достигло каких-либо существенных значений. Это является преимуществом Д-регулятора, и его применяют тогда, когда необходимо повысить быстродействие автоматической системы.

ПИ-регулятор представляет собой комбинацию П и И-регуляторов. Он вырабатывает сигнал, представляющий собой комбинацию скачкообразного изменения в начальный момент времени и плавного нарастания в последующие моменты времени, как показано на рис. 1.17. Пи-регулятор широко применяется в технике.

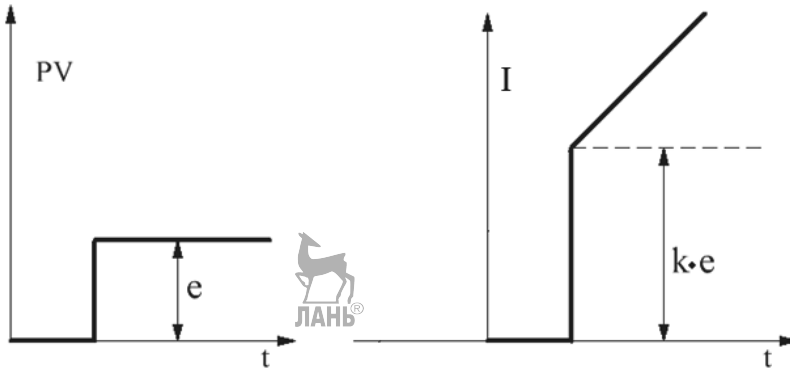


Рис. 1.17. Управляющий сигнал ПИ-регулятора

$$I(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau \quad (1.12)$$

В качестве примера опять рассмотрим ситуацию с автомобилем. Предположим, автомобиль отклонился от заданной траектории движения влево. Чтобы вернуть автомобиль на траекторию движения водитель поворачивает руль вправо. Причем поворачивает его тем больше, чем дальше автомобиль отклонился от заданной траектории. Это пропорциональное управление. Но автомобиль не может мгновенно изменить траекторию. Движение автомобиля влево хотя и замедляется, но еще продолжается. И чтобы не выехать на встречную полосу, водитель еще больше выкручивает руль вправо. Тем самым водитель интуитивно учитывает не только отклонение от траектории в текущий момент времени, но и накопленное отклонение от траектории, которое уже достаточно велико и одного пропорционального управления мало. Вот эта добавка к пропорциональному управлению, учитывающая историю движения, учитывающая большее отклонение автомобиля, и является интегральной составляющей управления.

ПИД-регулятор является самым эффективным типом регулятора и представляет собой комбинацию П, И, Д-регуляторов.

$$I(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \quad (1.13)$$

Выражение (1.13) – это так называемая параллельная форма записи ПИД-регулятора. Существует также стандартная форма записи ПИД-регулятора, которая имеет вид

$$I(t) = k_p \left(e(t) + \frac{k_i}{k_p} \int_0^t e(\tau) d\tau + \frac{k_d}{k_p} \frac{de(t)}{dt} \right) \quad (1.14)$$

Обозначим

$$\frac{k_i}{k_p} = \frac{1}{T_i}; \quad \frac{k_d}{k_p} = T_d \quad (1.15)$$

Тогда (1.14) переписывается в виде

$$I(t) = k_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \quad (1.16)$$

где коэффициенты T_i и T_d имеют следующий физический смысл:

T_i – время интегрирования;

T_d – время дифференцирования.

Передаточная функция ПИД-регулятора, записанного в форме (1.13) имеет вид

$$W(s) = k_p + \frac{k_i}{s} + k_d \cdot s \quad (1.17)$$

s – оператор Лапласа.

Передаточная функция ПИД-регулятора, записанного в форме (1.16) имеет вид

$$W(s) = k_p \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right) \quad (1.18)$$

Коэффициенты ПИД-регулятора необходимо подбирать таким образом, чтобы в процессе регулирования фактическое значение регулируемой переменной как можно быстрее и точнее достигало заданного значения. Возможные значения входного и выходного сигналов поясняются рисунком 1.18.

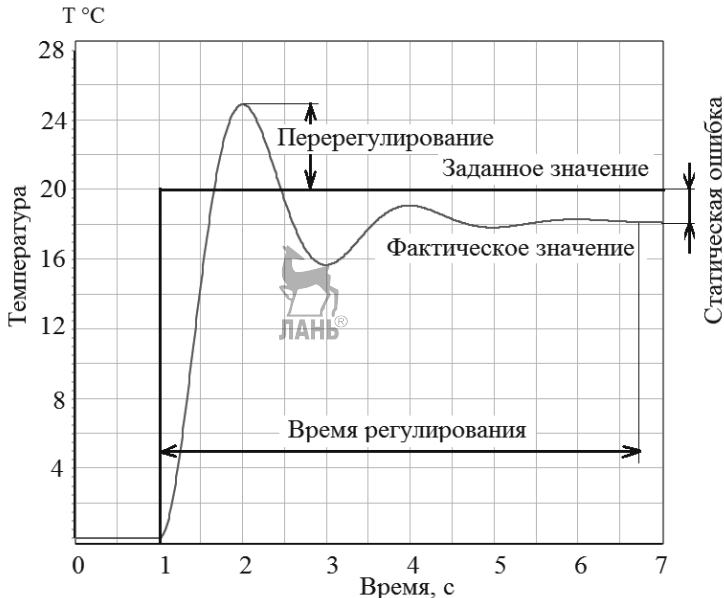


Рис. 1.18. Сигналы на входе и выходе САР

Разработаны различные методы настройки ПИД-регуляторов: теоретические и практические. При практической настройке ПИД-регулятора для колебательного звена сначала нужно установить значения k_i и k_d равными нулю и увеличивать пропорциональный коэффициент k_p до тех пор, пока система не достигнет колебательного поведения и не появится перерегулирование. Далее надо так отрегулировать k_i , чтобы убрать статическую ошибку и, наконец, так отрегулировать k_d , чтобы получить быстрый отклик и минимальное перерегулирование.



1.5. Моделирование регуляторов в SimInTech

Программу SimInTech можно скачать с сайта www.simintech.ru. Установка программы проходит штатно и не вызывает трудностей. Полное руководство по работе с программой изложено в справочной системе программы, а также в [3]. В данном подразделе рассмотрен конкретный пример использования программы для моделирования регуляторов.

После установки и запуска программы появляется Главное окно, показанное на рис. 1.19.

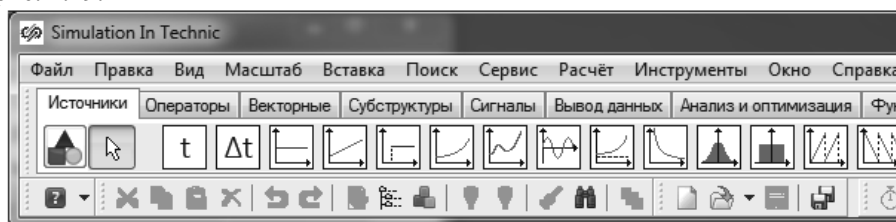


Рис. 1.19. Главное окно программы SimInTech

Далее открываем Схемное окно. Для этого щелкаем левой кнопкой мыши по опциям *Файл* > *Новый проект* > *Схема модели общего вида*. Располагаем схемное окно под главным окном, как показано на рис. 1.20. Позиции 1...4 на этом рисунке относятся к главному окну и позиции 5...9 к схемному окну:



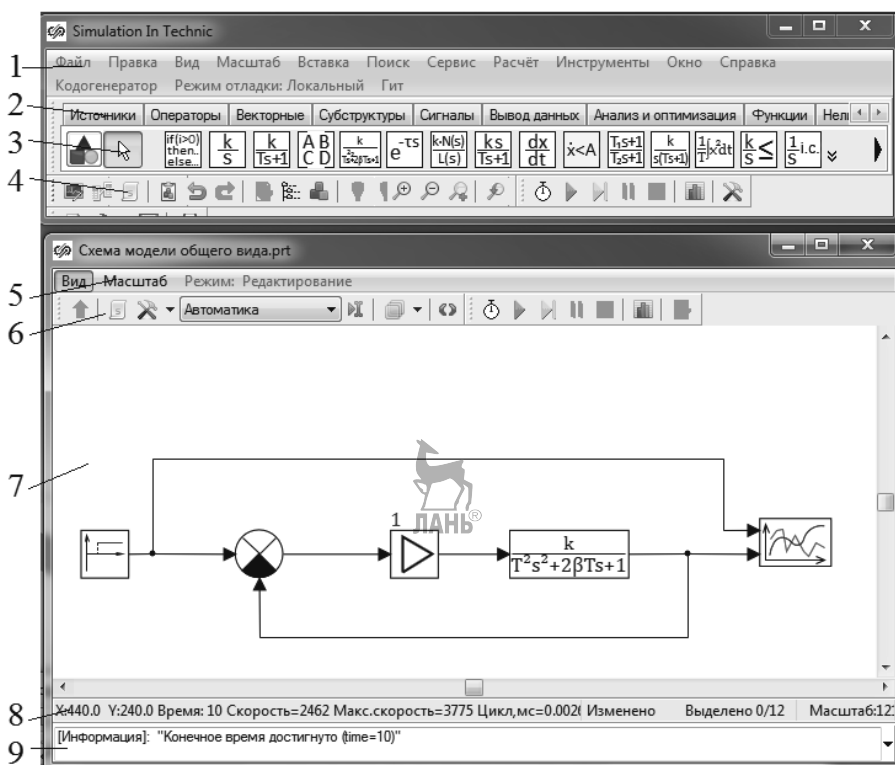


Рис. 1.20. Главное и схемное окна программы SimInTech

1. Меню главного окна.
2. Заголовки палитры блоков.
3. Палитры блоков. При выборе заголовка открывается соответствующая этому заголовку палитра блоков.
4. Панель инструментов.
5. Меню схемного окна.
6. Панель инструментов схемного окна.
7. Рабочее поле, в котором собирается структурная схема.
8. Панель информации. В частности, здесь можно задать видимость сетки и привязку к узлам сетки.
9. Панель сообщений. Здесь выводятся сообщения об ошибках, и информация об окончании работы программы.

Структурная схема программы для анализа эффективности работы регуляторов представлена на рис. 1.21.

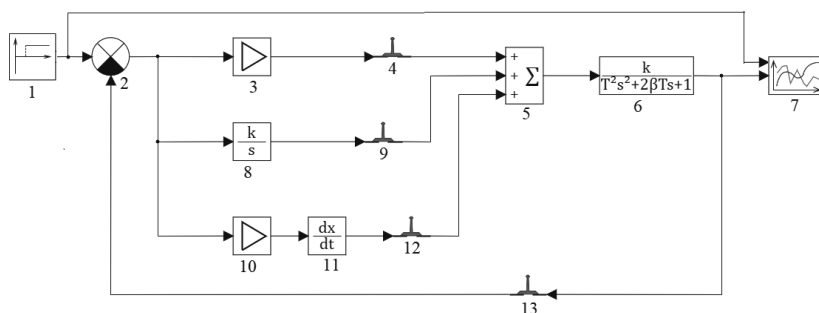
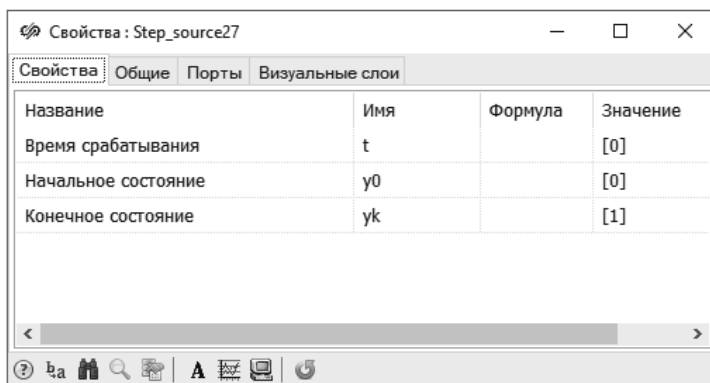


Рис. 1.21. Структурная схема

Для составления структурной схемы сначала надо разместить на рабочем поле все блоки схемы. Перечислим названия блоков и определим их параметры.

Блок 1 – *Ступенька* (ступенчатое воздействие) задает входной сигнал и находится в группе блоков *Источники*. Щелкаем по этому блоку левой кнопкой мыши, затем щелкаем внутри рабочего поля. На рабочем поле появится этот блок. Дважды щелкаем по блоку, появляется окно свойств блока, показанное на рис. 1.22.

Рис. 1.22. Окно свойств блока *Ступенька*

Во вкладке *Свойства* этого окна в столбце *Значение* устанавливаем *Начальное состояние* – 0, *Время срабатывания* – 0, *Конечное состояние* – 1. Время срабатывания и конечное состояние могут быть отличны от нуля. Под временем срабатывания понимается момент начала действия ступеньки на систему. Параметры во вкладках *Общие*, *Порты* и *Визуальные слои* оставляем по умолчанию.

Блок 2 – *Сравнивающее устройство* находится в группе блоков *Операторы*. Вытаскиваем блок на рабочее поле. Свойства этого блока не трогаем.

Блоки 3, 10 – *Усилитель* находится в группе блоков *Операторы*. В окне свойств блока можно задать коэффициент усиления. Блок имеет два порта: input и output. По умолчанию input – вход, output – выход. Но в окне свойств блока можно

переназначить input – выход, output – вход. Это, например, можно сделать, когда блок применяется в линии обратной связи. Однако лучше не менять назначение входов, а отразить блок зеркально на 180 градусов. Опцию зеркального отражения можно найти в контекстном меню, которое появляется после щелчка по блоку правой кнопкой мыши. Чтобы отразить блок зеркально, надо щелкнуть по опции *Зеркальное отражение* и провести мышкой около блока вертикальную линию.

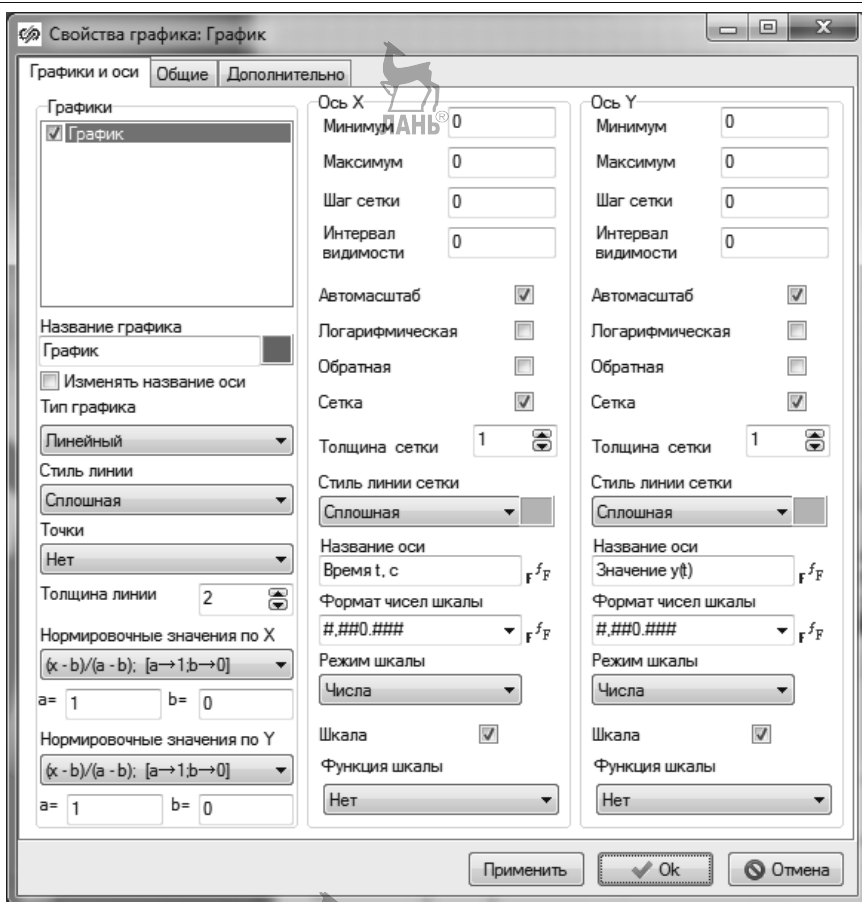
Блоки 4, 9, 12, 13 – *Ключ ручной управляемый* находится в группе блоков *Ключи*. В окне свойств блока галочка, поставленная в квадратике, соответствует замкнутому состоянию ключа, и отсутствие галочки соответствует разомкнутому состоянию ключа. Состояние ключа надо устанавливать именно в окне свойств блока. При щелчке левой кнопкой мыши по ключу его состояние может меняться только в процессе моделирования.

Блок 5 – *Сумматор* находится в группе блоков *Операторы*. По умолчанию сумматор имеет два входа. Чтобы изменить количество входов, надо в окне свойств блока в столбце *Значение* задать, например, [1,1,1]. Тогда будет три входа со значением «1» на каждом входе. Если задать [2,2,2,2], то будет четыре входа со значением «2» на каждом входе.

Блок 6 – *Колебательное звено* находится в группе блоков *Динамические*. При двойном щелчке по блоку левой кнопкой мыши появляется окно свойств блока, в котором нужно задать значения коэффициентов k , T , b . Начальные условия задаем нулевыми.

Блок 7 – *Временной график* находится в группе блоков *Вывод данных*. При двойном щелчке по блоку появляется область графика. Если внутри этой области щелкнуть правой кнопкой мыши и затем левой кнопкой мыши по строке *Свойства*, то появится окно *Свойства графика*, показанное на рис. 1.23.



Рис. 1.23. Окно блока *Графики*

Это окно предоставляет широкие возможности для редактирования графиков. Например, сняв галочку около строки *Автомасштаб* по осям X и Y, можно вручную задать нижнее и верхнее значение графика, шаг сетки и т.д.

Блок 8 – *Интегратор* находится в группе блоков *Динамические*. В окне свойства блока можно задать коэффициент усиления интегратора k и начальные условия.

Блок 11 – *Производная* находится в группе блоков *Динамические*.

После того, как все блоки размещены в рабочем поле, их надо соединить между собой линиями связи. Для этого щелкаем левой кнопкой мыши по выходному порту блока, тянем линию к входному порту следующего блока и щелкаем левой кнопкой мыши в области входного порта. Чтобы переместить блок внутри рабочего поля, надо навести на него курсор, зажать левую кнопку мыши и подвинуть блок. Чтобы удалить линию связи, надо щелкнуть по ней левой кнопкой мыши и нажать клавишу Delete.

Чтобы привязаться к линии связи (поставить на ней точку), надо зажать клавишу Alt и щелкнуть левой кнопкой мыши по линии связи. Появится точка и новая линия связи, которую можно протянуть к нужному блоку.

Чтобы задать параметры расчета: минимальный шаг интегрирования, максимальный шаг интегрирования, конечное время расчета, абсолютную и относительную ошибку и др., надо щелкнуть правой кнопкой мыши в области рабочего поля, появится контекстное меню (рис. 1.24), в котором надо щелкнуть по строке *Параметры расчета*. Появится окно выбора параметров проекта (рис. 1.25), в котором во вкладке *Параметры расчета* в столбце *Значение* нужно проставить нужные данные. Слева от строки *Параметры расчета* на рис. 1.24 имеется пиктограмма. Аналогичная пиктограмма имеется на панели инструментов схемного окна. Поэтому, чтобы вызвать окно параметров проекта (рис. 1.25) можно щелкнуть по этой пиктограмме. Это второй способ вызова окна параметров проекта.

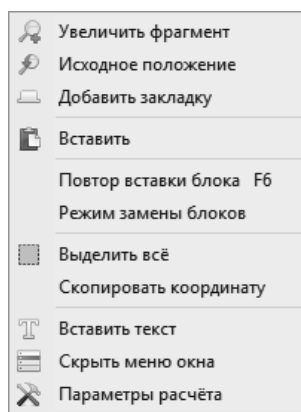


Рис. 1.24. Окно контекстного меню

Время интегрирования (конечное время расчета) ориентировочно задают на порядок больше, чем наибольшая постоянная времени исходной системы автоматического регулирования. В процессе моделирования время интегрирования уточняют. Оно должно быть не меньше времени регулирования.

Шаг интегрирования задают двумя значениями: максимальным и минимальным. При этом максимальное значение шага интегрирования принимают в 5–10 раз меньше наименьшей постоянной времени исходной САР. Минимальное значение шага интегрирования принимают в 10–100 раз меньше максимального значения шага интегрирования. Если в процессе моделирования не обеспечивается заданная точность интегрирования, то минимальный шаг интегрирования уменьшают до значений, при которых будет достигнута заданная точность.

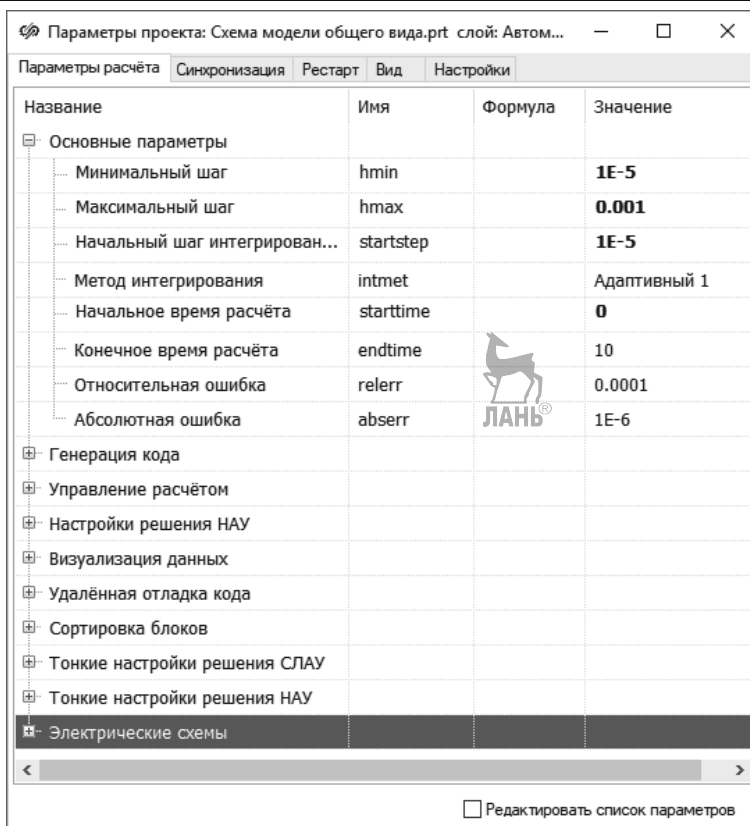


Рис. 1.25. Окно выбора параметров расчёта

Точность интегрирования определяется двумя величинами: относительной и абсолютной ошибками интегрирования. Их значения задают, исходя из условий сходимости численного решения задачи десятичным числом: например, $1e - 4 = 0,0001$ (0,01 %). Не следует задавать относительную ошибку больше 0,01 или меньше $1e - 10$. В большинстве случаев значения относительной и абсолютной ошибок можно принимать равными по умолчанию $1e - 4$ и $1e - 6$ соответственно.

Шаг вывода результатов ориентировочно принимают равным или больше максимального значения шага интегрирования. В процессе моделирования его можно изменять и уточнять, исходя из требований к качеству изображения графика переходного процесса.



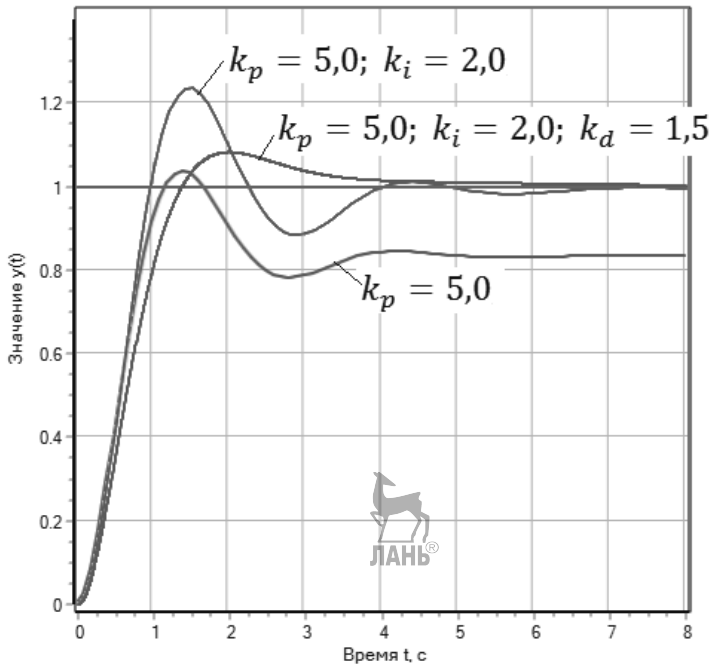


Рис. 1.26. Результаты моделирования регуляторов

Результаты моделирования регуляторов представлены на рис. 1.26. В качестве объекта управления выбрано колебательное звено с параметрами: $k=1$; $T=1$; $b=1$. Путем замыкания и размыкания ключей 9; 12 моделировались различные типы регуляторов: П; ПИ; ПИД-регуляторы. Значения коэффициентов регуляторов показаны на графике. Из графика следует, что П-регулятор дает статическую ошибку, ПИ-регулятор статическую ошибку убирает, но увеличиваются колебательные характеристики выходного сигнала, в том числе, увеличивается перерегулирование выходного сигнала. Наилучшие результаты показывает ПИД-регулятор, у которого нет статической ошибки и который дает наименьшее перерегулирование.

Проведем модификацию структурной схемы (рис. 1.21). Дифференциальная ветвь ПИД-регулятора является идеальным звеном. Поставим в дифференциальную ветвь вместо идеального звена реальное инерционно-дифференцирующее звено, как показано на рис. 1.27. Результаты моделирования схемы (рис. 1.27) показаны на рис. 1.28. Значения коэффициентов ПИД-регулятора указаны на графике.

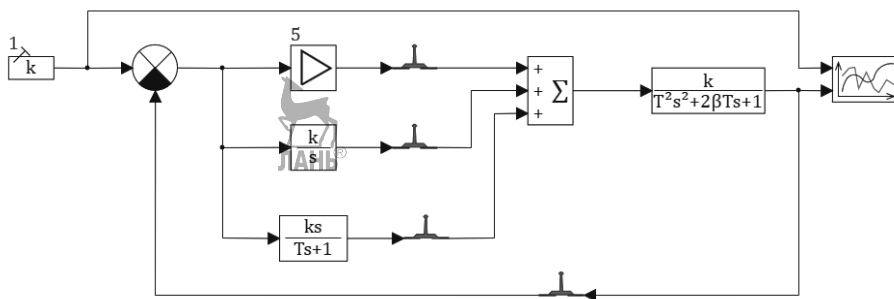


Рис. 1.27. Схема с модифицированным ПИД-регулятором

Из графика (рис. 1.28) следует, что эта схема дает практически идеальный результат: перерегулирования нет, статической ошибки нет, время выхода на установившийся режим (попадание в трубку допуски) составляет около 1 с.

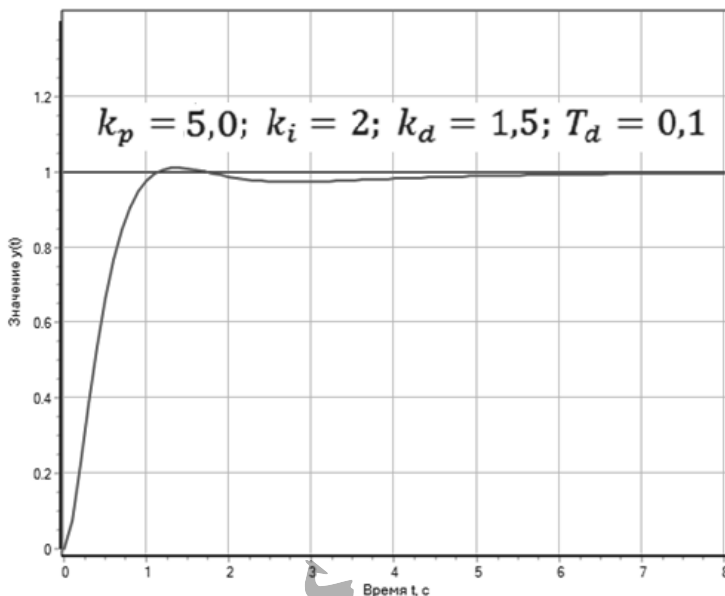


Рис. 1.28. Результаты моделирования

Контрольные вопросы и задания

1. Чем отличаются функциональная и структурная схемы системы автоматического управления?
2. Начертить типовую функциональную схему системы автоматического регулирования.

3. Перечислить типовые динамические звенья и записать аналитические выражения для передаточных функций этих звеньев.
4. Какие преимущества дает использование преобразования Лапласа при решении дифференциальных уравнений?
5. Приведите примеры дискретных систем управления в бытовой технике.
6. Какой цикл управления, замкнутый или разомкнутый, имеет стиральная машина, выполняющая заданную программу стирки?
7. В чем заключается смысл отрицательной обратной связи?
8. Получить передаточную функцию уравнения

$$a_1 \frac{dy(t)}{dt} + a_2 y = kx(t)$$

9. На объект регулирования поступают два входных сигнала $x_1(t)$; $x_2(t)$, как показано на рис. 1.29.

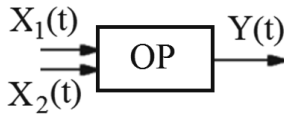


Рис. 1.29. Объект регулирования

Для уравнения

$$a_0 \frac{d^2 y(t)}{dt^2} + a_1 \frac{dy(t)}{dt} + a_2 y = k_1 x_1(t) + k_2 x_2(t)$$

получить передаточные функции по сигналу $x_1(t)$ и сигналу $x_2(t)$

$$W_1(s) = \frac{Y(s)}{X_1(s)}; \quad W_2(s) = \frac{Y(s)}{X_2(s)}$$

10. Записать эквивалентную передаточную функцию для структурной схемы на рис. 1.30

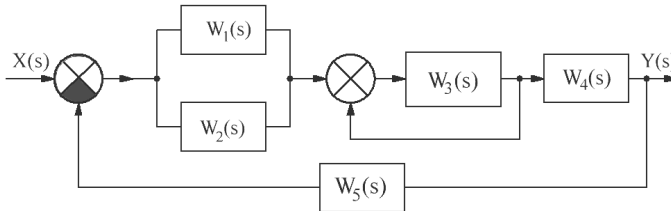


Рис. 1.30. Структурная схема

11. Повторить в программе SimInTech схему на рис. 1.21 и получить графики, аналогичные рис. 1.26.
12. Добавить на схему рис. 1.21 инерционное звено после ПИД-регулятора и подобрать коэффициенты инерционного звена такими, чтобы перерегулирование было меньше 5%, и время регулирования меньше 3 с. Коэффициенты ПИД регулятора задать следующими: $k_p = 5,0$; $k_i = 2,0$; $k_d = 1,5$.

13. Найти с помощью моделирования в программе SimInTech значение коэффициента k , при котором САР имеет минимальное перерегулирование и минимальное время регулирования (при $k_1 = 2$; $T_1 = 0,5$ с)

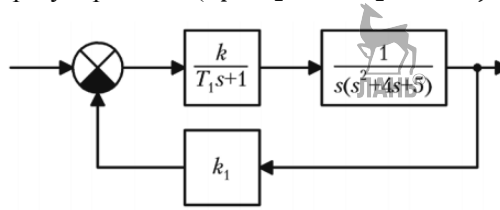


Рис. 1.31. Структурная схема

14. Найти с помощью моделирования в программе SimInTech значение коэффициента k , при котором САР имеет минимальное перерегулирование и минимальное время регулирования (при $k_2 = 3$; $T_1 = 0,5$ с; $T_2 = 1$ с; $T_3 = 0,25$ с)

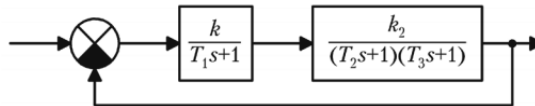


Рис. 1.32. Структурная схема

15. Найти с помощью моделирования в программе SimInTech значение коэффициента k , при котором САР имеет минимальное перерегулирование и минимальное время регулирования (при $k_2 = 1,5$; $k_3 = 1$; $k_4 = 1$; $T_1 = 0,1$ с; $T_2 = 1$ с.)

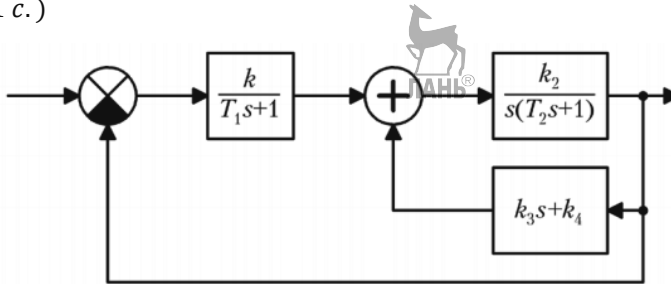


Рис. 1.33. Структурная схема

ГЛАВА 2. ЭЛЕМЕНТЫ ЦИФРОВОЙ ТЕХНИКИ

2.1. Основные понятия алгебры логики

Изучение основ цифровой техники начнем с определения понятий. **Логические переменные** – это исходные данные, которые могут принимать только два значения: «0» и «1». Логические переменные обычно выражают наличие или отсутствие какого-либо факта. Значение логической переменной в технике часто представляют уровнями электрического напряжения: высоким Н (High) и низким L (Low). Н – уровень соответствует логической единице, обычно близок к напряжению питания. Как правило, +5 В, L – уровень соответствует логическому нулю, близок к нулевому напряжению. В случае, когда более высокий уровень сигнала принимают за «1», а более низкий – за «0», говорят о позитивной или положительной логике. Если же принято противоположное соответствие, то имеет место отрицательная (негативная) логика. Далее будем пользоваться положительной логикой.

Совокупность логических операций, выполняемых над логическими переменными, образует **логическую функцию**. Логическая функция, как и логические переменные, принимает только два значения, «0» или «1». Логическую функцию чаще всего записывают либо в аналитической форме в виде **уравнения функции**, либо в форме **таблицы истинности**. Таблица истинности – это совокупность значений функции, соответствующих всем возможным комбинациям значений логических переменных. Число строк (k) в таблице истинности (если речь идет о бинарном коде) равно числу возможных наборов двоичных значений, и может быть вычислено по формуле $k = 2^n$, где n – число переменных. Например, число строк в таблице истинности (а значит и число значений функции) для двух переменных (A, B) равно $2^2 = 4$, для трех переменных (A, B, C) равно $2^3 = 8$ и т.д.

Инструментом связи между логическими переменными и логическими функциями служит булева алгебра, названная по имени английского математика Дж. Буля. В булевой алгебре к **базовым логическим функциям** относятся функции, которые образуют полный набор, с помощью которого можно записать любую другую логическую функцию, и можно реализовать логическое устройство любой сложности. Такой набор базовых функций называется **функционально полным логическим базисом**. Таких базисов четыре:

- И, НЕ (2 элемента)
- ИЛИ, НЕ (2 элемента)
- И-НЕ (1 элемент)
- ИЛИ-НЕ (1 элемент).



Базовым логическим функциям НЕ, ИЛИ, И соответствуют логические операции отрицания, сложения, умножения.

В технике базовые логические функции реализуются в виде устройств, которые называются **логическими элементами**. Иногда их называют вентилями. Логические элементы выпускаются в виде микросхем. Одна микросхема может содержать несколько одинаковых логических элементов. Из логических элементов составляют **логические схемы**, иногда очень сложные, которые позволяют выполнять арифметические операции и хранить информацию. На базе логических элементов И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ созданы цифровые устройства, используемые в том числе и в вычислительной технике: триггеры, счетчики, регистры, сумматоры, шифраторы, дешифраторы. Логические контроллеры также представляют собой класс технических устройств, позволяющих выполнять логические операции, состоящие как из базовых логических функций, так и специальных функций, таких, как таймеры, счетчики и др.

Техническое устройство, в котором реализована одна логическая схема, относится к устройствам жесткой логики, предназначенным для выполнения конкретной и единственной последовательности операций. Программируемый логический контроллер является устройством гибкой логики, его можно запрограммировать на выполнение различных логических операций.

Алгоритм построения логических схем следующий:

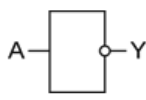

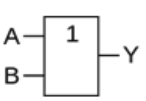

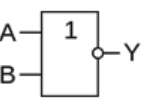

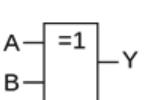

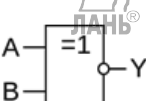

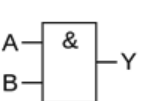

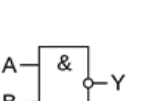

1. Определить число логических переменных.
2. Определить количество базовых логических операций и их порядок.
3. Графически изобразить для каждой логической операции соответствующий ей логический элемент.
4. Соединить логические элементы в порядке выполнения логических операций.

При графическом изображении логических схем используются условно-графические обозначения (УГО) базовых логических функций, представленные в таблице 2.1. Показаны обозначения в соответствии со стандартами ГОСТ и ANSI. Стандарт ГОСТ применяется в России. Похожие обозначения используются в международном электротехническом стандарте – МЭК (IEC – International Electrotechnical Commission). Стандарт ANSI – национальный американский стандарт, разработан Американским национальным институтом стандартизации (American National Standards Institute).

Для логических операций «И», «ИЛИ», «НЕ» применяются следующие знаки

И	ИЛИ	НЕ
•	+	черта сверху
∧	∨	¬

Таблица 2.1

Логические функции	ГОСТ	ANSI	Таблица истинности		
			A	B	Y
НЕ – отрицание (инвертор)		 INV	A		$Y = \bar{A}$
			0		1
			1		0
ИЛИ – дизъюнкция (логическое сложение)		 OR	A	B	$Y = A \vee B$
			0	0	0
			0	1	1
			1	0	1
			1	1	1
ИЛИ-НЕ – инверсия дизъюнкции (стрелка Пирса)		 NOR	A	B	$Y = A \downarrow B$
			0	0	1
			0	1	0
			1	0	0
			1	1	0
Исключающее ИЛИ		 XOR	A	B	$Y = A \oplus B$
			0	0	0
			0	1	1
			1	0	1
			1	1	0
Исключающее ИЛИ-НЕ		 XNOR	A	B	$Y = A \leftrightarrow B$
			0	0	1
			0	1	0
			1	0	0
			1	1	1
И – конъюнкция (логическое умножение).		 AND	A	B	$Y = A \wedge B$
			0	0	0
			0	1	0
			1	0	0
			1	1	1
И-НЕ – инверсия конъюнкции (штрих Шеффера)		 NAND	A	B	$Y = A B$
			0	0	1
			0	1	1
			1	0	1
			1	1	0

Пример 2.1. Составить логическую схему по логическому выражению

$$Y = (A \cdot B) + \overline{(A + B)} \cdot \bar{B}$$

Имеем:

число логических переменных – две: A, B;

число логических операций – шесть: «И»; «ИЛИ»; «НЕ»; «НЕ»; «И»; «ИЛИ».

Таким образом, в логической схеме должно быть шесть логических элементов, равное числу логических операций.

Схему строим слева направо в соответствии с порядком логических операций, как показано на рис. 2.1.

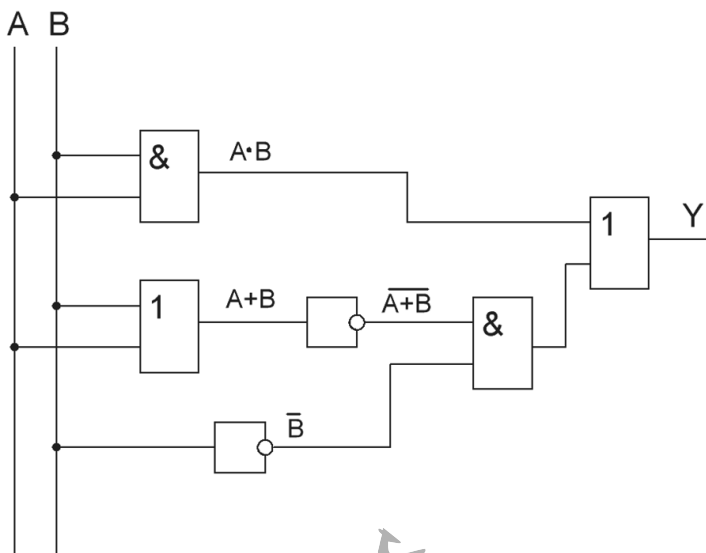


Рис. 2.1. Логическая схема

Полученная логическая схема, по сути, является программой, по которой будет работать логический контроллер.

В данном примере логическая функция содержит шесть логических операций. Встречаются логические функции, которые содержат десятки и сотни логических операций, и практическая реализация таких функций становится затруднительной. Однако существуют законы и правила алгебры логики, которые позволяют упрощать сложные логические выражения. Применяя эти законы и правила, можно привести логическое выражение к более простому виду, что потребует и более простых технических средств для его практической реализации.

2.2. Законы и правила алгебры логики

Законы и правила логики управляют связыванием переменных (A, B, C и т.д.) и констант (0, 1).

Основные законы. Основные законы управляют связыванием констант.

Связка И	Связка ИЛИ	Связка НЕ
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\bar{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

Законы поглощения. Законы поглощения описывают, когда переменная сама себя воспроизводит или аннулирует.

$A \cdot 0 = 0$	$A + 0 = A$	$A \cdot (A + B) = A$
$A \cdot 1 = A$	$A + 1 = 1$	$A + (A \cdot B) = A$
$A \cdot A = A$	$A + A = A$	$A + \bar{A} \cdot B = A + B$
$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$	

Переместительный закон. Переместительный закон гласит, что изменение последовательности переменных не влияет на результат.

$$A \cdot B = B \cdot A$$

$$A + B = B + A$$

Сочетательный (ассоциативный) закон. Сочетательный закон гласит, что изменение сочетаний операторов не влияет на результат.

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

$$A + B + C = (A + B) + C = A + (B + C)$$

Распределительный (дистрибутивный) закон. Распределительный закон гласит, что выражения в скобках перемножаются при соблюдении последовательности операторов.

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$(A + B) \cdot (A + C) = A + B \cdot C$$

Второе равенство требует пояснений:

$$(A + B) \cdot (A + C) = A \cdot A + B \cdot A + A \cdot C + B \cdot C = A + A \cdot B + A \cdot C + B \cdot C$$

$$= A \cdot (1 + (B + C)) + B \cdot C = A + B \cdot C$$

так как $(1 + (B + C)) = 1$ при любом сочетании значений 0 и 1 для B и C.

Законы Де Моргана

$$\overline{A \cdot B \cdot C} = \bar{A} + \bar{B} + \bar{C}$$

$$\overline{A + B + C} = \bar{A} \cdot \bar{B} \cdot \bar{C}$$

Законы Де Моргана гласят, что отрицание конъюнкции (дизъюнкции) переменных равно дизъюнкции (конъюнкции) отрицаний этих переменных.

Двойное отрицание

$$\overline{\bar{A}} = A$$

Правило связи

$$A + B \cdot C = A + (B \cdot C)$$

Правило связи гласит, что связка «И» должна выполняться перед связкой «ИЛИ».

Рассмотрим примеры упрощающих преобразований логических выражений.

Пример 2.2.

$$(A \cdot B) + (A \cdot \bar{B}) = A \cdot B + A \cdot \bar{B} = A \cdot (B + \bar{B}) = A \cdot 1 = A$$

При преобразовании использовались: распределительный закон, сочетательный закон, закон поглощения.

Пример 2.3.

$$\begin{aligned} (A + B) \cdot (A + \bar{B}) &= (A + B) \cdot A + (A + B) \cdot \bar{B} = A \cdot A + B \cdot A + A \cdot \bar{B} + B \cdot \bar{B} \\ &= A + B \cdot A + A \cdot \bar{B} + 0 = A + A \cdot B + A \cdot \bar{B} = A \cdot (1 + B + \bar{B}) = A \end{aligned}$$

При преобразовании использовались: распределительный закон, распределительный закон, закон поглощения, переместительный закон, сочетательный закон, закон поглощения.

2.3. Проектирование логической схемы

При работе с логическими устройствами различают схемный анализ и схемный синтез. **Схемный анализ** заключается в определении уравнения логической функции по заданной логической схеме. **Схемный синтез** заключается в проектировании логической схемы по имеющемуся уравнению логической функции. Поскольку книга посвящена программированию контроллеров, то нас, в первую очередь, будет интересовать схемный синтез, поскольку именно он позволяет разработать программу, управляющую работой логического контроллера. Схемный синтез или создание логической схемы состоит из трех этапов. На первом этапе, исходя из описания решаемой задачи, составляется таблица истинности. На втором этапе по таблице истинности составляется уравнение логической функции. Для вывода уравнения логической функции используются или **нормальная форма «ИЛИ»**, или **нормальная форма «И»**, или **карты Карно**. На третьем этапе по логическому выражению составляется логическая схема, которая является прообразом программы для контроллера.

Пример 2.4. Рассмотрим пример на составление логической схемы. Пусть имеется четыре потребителя электрической энергии: А, В, С, D. Потребитель А потребляет 5 кВт электроэнергии, потребитель В – 15 кВт, потребитель С – 15 кВт, потребитель D – 20 кВт. Общее потребление электроэнергии всеми потребителями не должно превышать 25 кВт. В противном случае будет превышена допустимая нагрузка на сеть и будет выдан сигнал тревоги.

Составим таблицу истинности для данной задачи (табл. 2.2).

Таблица 2.2

D	C	B	A	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Y – значение логической функции, которая принимает значение «0», если потребление электроэнергии меньше или равно 25 кВт, и принимает значение «1», если потребление электроэнергии превышает 25 кВт. Для составления логической схемы будем использовать нормальную форму «ИЛИ». Она заключается в следующем. Для строк, у которых значение функции Y равно «1», составим частичные функции путем связки «И». Переменные с состоянием «0» отображаются при этом с отрицанием:

$$\begin{array}{llll} \bar{D} \cdot \bar{C} \cdot B \cdot \bar{A} (7) & \bar{D} \cdot C \cdot B \cdot A (8) & D \cdot \bar{C} \cdot B \cdot \bar{A} (11) & D \cdot \bar{C} \cdot B \cdot A (12) \\ D \cdot C \cdot \bar{B} \cdot \bar{A} (13) & D \cdot C \cdot \bar{B} \cdot A (14) & D \cdot C \cdot B \cdot \bar{A} (15) & D \cdot C \cdot B \cdot A (16) \end{array}$$

В скобках указаны номера строк, для которых записаны частичные функции.

Вся функция Y выражается теперь через связку «ИЛИ» для всех частичных функций

$$Y = \bar{D} \cdot \bar{C} \cdot B \cdot \bar{A} + \bar{D} \cdot C \cdot B \cdot A + D \cdot \bar{C} \cdot B \cdot \bar{A} + D \cdot \bar{C} \cdot B \cdot A + \\ + D \cdot C \cdot \bar{B} \cdot \bar{A} + D \cdot C \cdot \bar{B} \cdot A + D \cdot C \cdot B \cdot \bar{A} + D \cdot C \cdot B \cdot A$$

Применив законы алгебры логики, полученное уравнение можно упростить до вида

$$Y = B \cdot C + B \cdot D + C \cdot D$$

Логическая схема для этого уравнения показана на рис. 2.2.

Теперь построим логическую схему, используя нормальную форму «И». Для этого в таблице истинности 2.2 выделим строки, для которых Y=0, как показано в табл. 2.3.

Таблица 2.3

D	C	B	A	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Для выделенных строк составим частичные функции путем связки «ИЛИ». Затем запишем полную функцию из частичных функций путем связки «И». Переменные с состоянием «1» отображаются при этом с отрицанием:

$$Y = (D + C + B + A) \cdot (D + C + B + \bar{A}) \cdot (D + C + \bar{B} + A) \cdot (D + C + \bar{B} + \bar{A}) \cdot (D + \bar{C} + B + A) \cdot (D + \bar{C} + B + \bar{A}) \cdot (\bar{D} + C + B + A) \cdot (\bar{D} + C + B + \bar{A})$$

После упрощения получим

$$Y = B \cdot C + B \cdot D + C \cdot D \quad (2.1)$$

Таким образом, для нормальной формы «И» получили формулу, аналогичную формуле для нормальной формы «ИЛИ», что и следовало ожидать.

На практике для составления уравнения целесообразно выбирать такую нормальную форму, которая имеет меньше строк. Например, если строк со значением «1» меньше, чем со значением «0», то для составления уравнения надо выбирать нормальную форму «ИЛИ». Тогда уравнение функции становится короче и с ним легче работать.

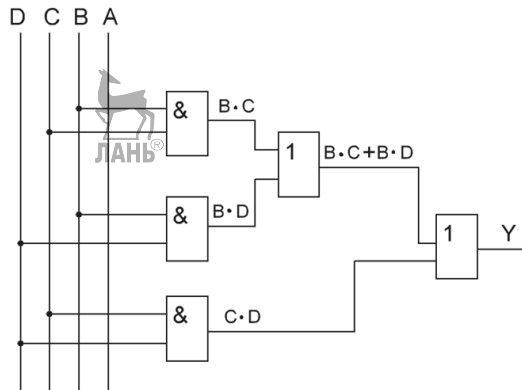


Рис. 2.2. Логическая схема

Упрощение полученного уравнения функции может оказаться нетривиальной задачей. Но этот этап работы можно существенно упростить, если использовать программу компьютерного моделирования Multisim. Описание работы с программой Multisim приведено в [1].

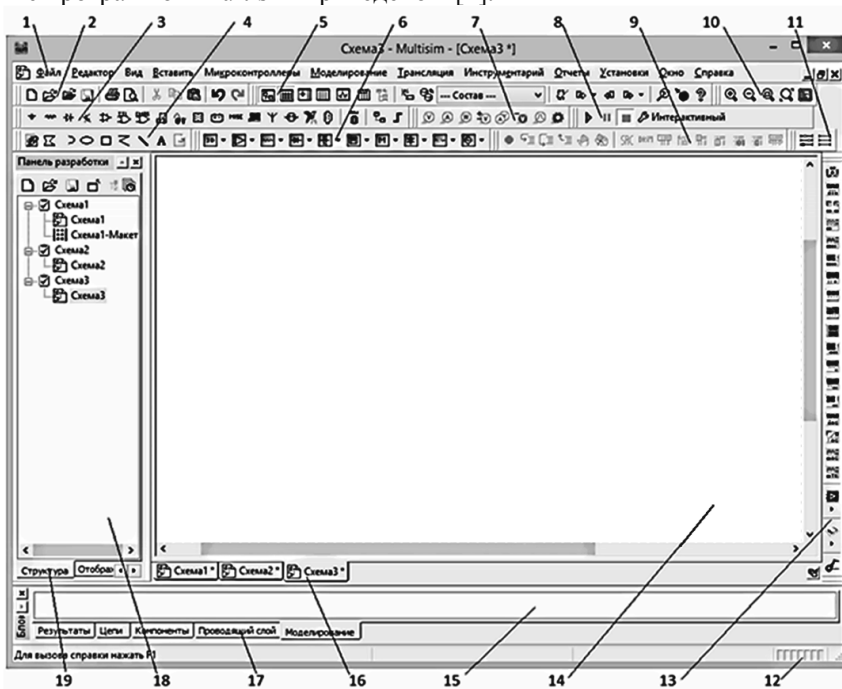


Рис. 2.3. Интерфейсное окно программы Multisim



Интерфейсное окно программы Multisim 14 представлено на рис. 2.3, где цифрами обозначено:

1. Главное меню.
2. Панель *Стандартная*.
3. Панель *Реальные компоненты*.
4. Панель *Рисование*.
5. Панель *Главная*.
6. Панель *Виртуальные компоненты*.
7. Панель *Установить пробник*.
8. Панель *Моделирование*.
9. *MCU* (панель микроконтроллеров).
10. Панель *Вид*.
11. Панель *Многозвенные схемы*.
12. Строка состояния.
13. Приборы.
14. Рабочее поле.
15. Блок информации.
16. Закладки рабочих полей.
17. Закладки блока информации.
18. Панель разработки.
19. Закладки панели разработки.

На панели *Приборы* (поз.13) имеется пиктограмма вызова прибора *Логический преобразователь*. Эту пиктограмму можно найти по подсказке, которая появляется при наведении курсора мыши на пиктограммы.

Логический преобразователь (Logic Converter) – это виртуальный прибор, не имеющий реальных аналогов. Он предназначен для работы с логическими схемами и логическими выражениями. Условное изображение логического преобразователя показано в верхней части рисунка 2.4 и его лицевая панель в нижней части рисунка 2.4. Лицевая панель прибора появляется после щелчка мыши по условному изображению. Прибор имеет 8 входов (8 входных переменных) А, В, С, D, E, F, G, H и один выход Out (крайний правый контакт).

С помощью логического преобразователя можно осуществлять следующие операции:

- получение таблицы истинности для логической схемы (клавиша 1);
- преобразование таблицы истинности в логическое выражение (клавиша 2);
- минимизация логических выражений (клавиша 3);
- обратное преобразование логического выражения в таблицу истинности (клавиша 4);
- синтез логических схем по логическому выражению в базисе И, ИЛИ, НЕ (клавиша 5);
- синтез логических схем по логическому выражению в базисе И-НЕ (клавиша 6).



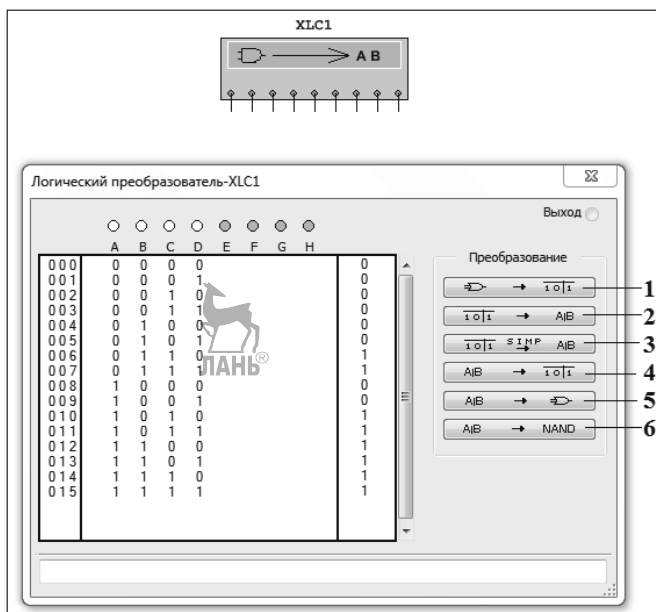


Рис. 2.4. Условное изображение логического преобразователя (сверху) и лицевая панель логического преобразователя (снизу)

Щелчком левой кнопкой мыши по буквам А, В, С, D и в средней части лицевой панели автоматически появится стандартная таблица истинности для 4-х переменных. Теперь необходимо заполнить правый столбец, который соответствует значениям Y логической функции. Значения для Y возьмем из таблицы 2.2. Первоначально, вместо значений «0» и «1» в правом столбце расположены вопросительные знаки. Чтобы задать соответствующее значение функции, нужно несколько раз щелкнуть левой кнопкой мыши по вопросительному знаку, пока не появится нужное значение.

Когда вся правая часть таблицы заполнена, щелчком по клавише 2, и в нижней строке лицевой панели появится логическое уравнение для функции Y, как показано на рис. 2.5.



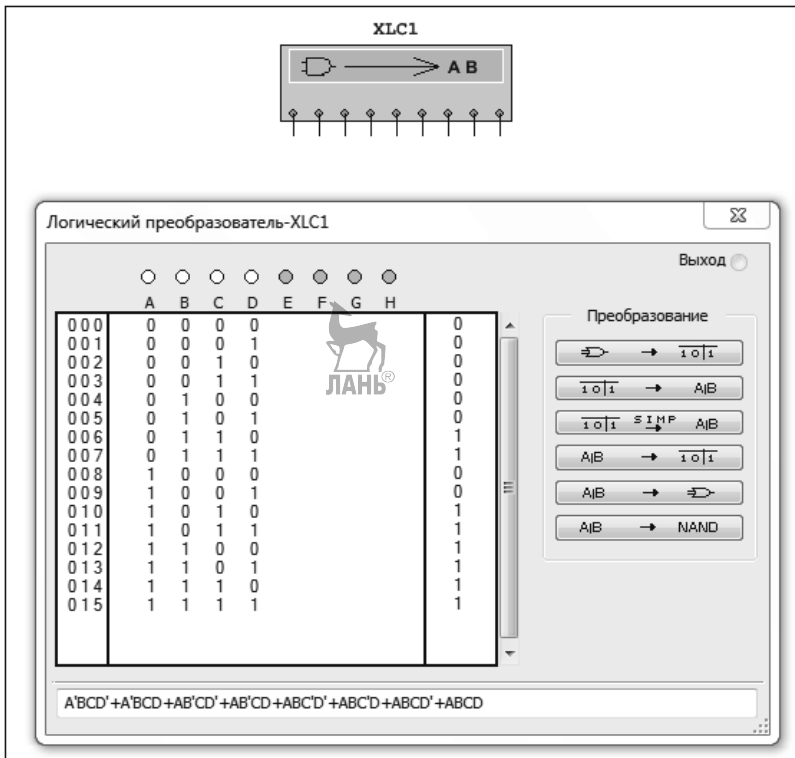


Рис. 2.5. Лицевая панель логического преобразователя с логическим выражением

Чтобы упростить это уравнение, щелкнем по клавише 3. Появится упрощенное выражение, показанное на рис. 2.6.

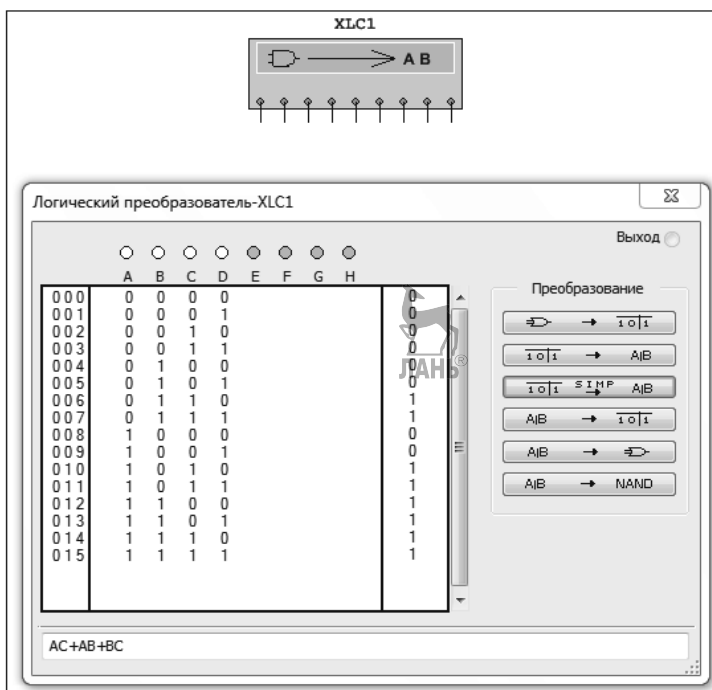


Рис. 2.6. Лицевая панель логического преобразователя с упрощенным логическим выражением

Итак, получили упрощенное уравнение для функции Y в виде

$$Y = A \cdot C + A \cdot B + B \cdot C \quad (2.2)$$

При внешнем различии (2.1) и (2.2) – это одна и та же формула. Их внешнее различие происходит из-за того, что в таблице истинности 2.2 и в таблице истинности на рис. 2.4 принят различный порядок расположения переменных A , B , C , D . Если в таблице истинности 2.2 применить порядок расположения переменных аналогично Multisim, а именно: D заменить на A , C на B , B на C , A на D , то форма записи (2.1) преобразуется к форме записи (2.2). В дальнейшем, чтобы не возникало разночтений, при составлении таблицы истинности будем придерживаться порядка расположения переменных, который принят в программе Multisim.

Пример 2.5. Бортовой компьютер контролирует работу 3-х систем автомобиля. При неисправности одной из систем выдается короткий гудок. При неисправности любых двух и более систем выдается длинный гудок.

Составить логическую схему.

Решение. Составим таблицу истинности (табл. 2.4) по описанию задачи. Трех системам автомобиля соответствуют логические переменные A , B , C . Пусть

короткому гудку соответствует логическая функция $Y1$ и длинному гудку – логическая функция $Y2$.

Таблица 2.4.

A	B	C	Y1	Y2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Чтобы запрограммировать контроллер, работающий по описанному алгоритму, надо:

- воспользоваться нормальной формой «ИЛИ» (так как строк со значением «1» меньше) и составить по таблице истинности логическое уравнение;
- упростить полученное уравнение;
- по уравнению составить логическую схему, которая послужит прообразом программы для контроллера.

Поступим проще и получим логические выражения с помощью программы Multisim. Для этого откроем программу Multisim, запустим логический преобразователь, щелкнем левой кнопкой мыши по буквам A, B, C. Программа автоматически заполнит столбцы под этими буквами. Далее, в соответствии с таблицей истинности 2.4, надо заполнить крайний правый столбец сначала для функции $Y1$ (рис. 2.7), а затем для функции $Y2$ (рис. 2.8). Получим логические выражения:

- для короткого гудка

$$Y1 = \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C}$$

- для длинного гудка

$$Y2 = A \cdot C + A \cdot B + B \cdot C$$



По полученным логическим выражениям строим логическую схему.

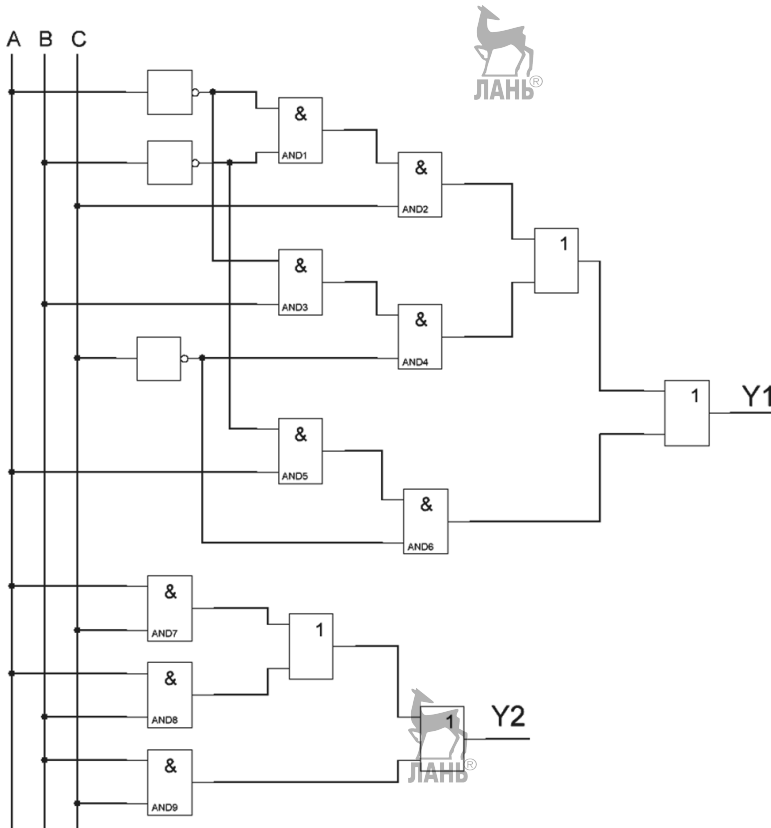


Рис. 2.9. Логическая схема

Отметим, что логическую схему можно построить с помощью программы Multisim, нажав справа на лицевой панели 5-ю сверху клавишу.

2.4. Проектирование релейно-контактных схем

Сведения по построению логических схем, изложенные в предыдущем подразделе, будут использоваться в дальнейшем для изучения языка программирования логических контроллеров – FBD (язык функциональных блоков). В этом подразделе изложим сведения по проектированию релейно-контактных схем, которые будут использоваться в дальнейшем для изучения языка программирования – LD (язык лестничных диаграмм). Для описания релейно-контактных схем будем придерживаться терминологии и условно-графических обозначений, принятых в программе Multisim. Это позволит избежать путаницы в обозначениях при моделировании разрабатываемых схем в

Multisim. Релейно-контактная схема, показанная на рис. 2.12 содержит следующие элементы:

- боковые звенья L1, L2, L3 и т.д. (Ladder Diagrams → Ladder Rungs → L1 (левое) / L2 (правое));
- ключи S1, S2 (Basic → Switch → SPST);
- катушки реле M1; M2 (Ladder Diagrams → Ladder Relay Coils → Relay Coil);
- нормально разомкнутые контакты X1...X4 (Ladder Diagrams → Ladder Contacts → Relay Contact NO);
- лампы X7; X8 (Indicators → Lamp).

В скобках справа от элементов указан раздел программы Multisim, в котором можно найти перечисленные элементы. Раздел указывается в выпадающем окошке **Выбор компонента**, которое появляется после выбора строки **Компонент** во вкладке меню **Вставить**. Двухбуквенное обозначение внутри символа катушки реле означает сокращение CR – Coil Relay.

Сверху над контактом на рис. 2.10 указывается обозначение контакта, а под контактом указывается ссылка на катушку, которая управляет данным контактом. Чтобы задать ссылку на катушку, надо дважды щелкнуть по контакту и в выпадающем окошке во вкладке **Параметры** указать ссылку.

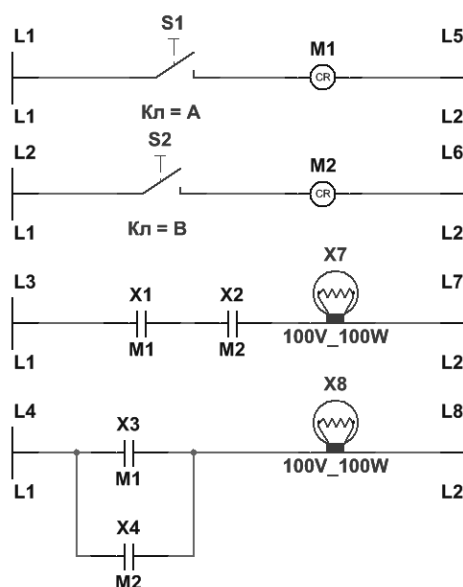
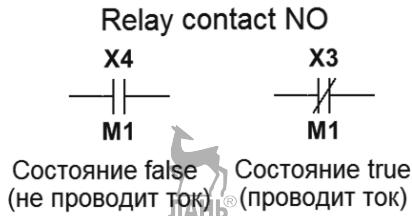


Рис. 2.10. Релейно-контактная схема

Когда в катушке протекает ток, контакты, содержащие ссылку на данную катушку, изменяют свое состояние. Например, нормально разомкнутые контакты замыкаются.

Контакты, в свою очередь, бывают нормально разомкнутые (NO – Normal Open) и нормально замкнутые (NC – Normal Close), как показано на рис. 2.11.



Чтобы лампочка X7 на рис. 2.10 включилась, необходимо, чтобы оба контакта X1, X2 замкнулись. Это соответствует функции «И», когда обе переменные должны быть равны «1». Чтобы включилась лампочка X8, должен замкнуться один из контактов X3 или X4. Это соответствует функции «ИЛИ», когда значение «1» должна иметь одна из переменных. Ключи S1, S2 подключают/отключают катушки реле.

Пример 2.6. Имеются две лампочки, которые управляются тремя катушками реле. Будем считать катушки логическими переменными, которые принимают значения «0» и «1». Значение «0» соответствует разомкнутому состоянию катушки и «1» – замкнутому. Поскольку катушки обозначаются в Multisim буквой M, то назовем переменным обозначения M1, M2, M3. Пусть лампочка EL1 включается, когда замыкаются катушки или M1 и M2, или M2 и M3. Лампочка EL2 включается, когда замыкаются все три катушки. Лампочке EL1 поставим в соответствие логическую функцию Y1, а лампочке EL2 – функцию Y2. Составим таблицу истинности (табл. 2.5).

Таблица 2.5

M1	M2	M3	Y1	Y2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	1

Применим нормальную форму «ИЛИ», чтобы написать уравнения функций.

$$Y1 = \overline{M1} \cdot M2 \cdot M3 + M1 \cdot M2 \cdot \overline{M3} \quad (2.3)$$

$$Y2 = M1 \cdot M2 \cdot M3 \quad (2.4)$$

Функцию (2.3) можно переписать в виде

$$Y1 = M2 \cdot (\overline{M1} \cdot M3 + M1 \cdot \overline{M3}) \quad (2.5)$$

Функция (2.4) состоит из операций «И», функция (2.5) состоит из операций «И», «ИЛИ».

Составим релейно-контактную схему (рис. 2.12). Переменным без черты сверху соответствуют NO-контакты и переменным с чертой сверху соответствуют NC-контакты.

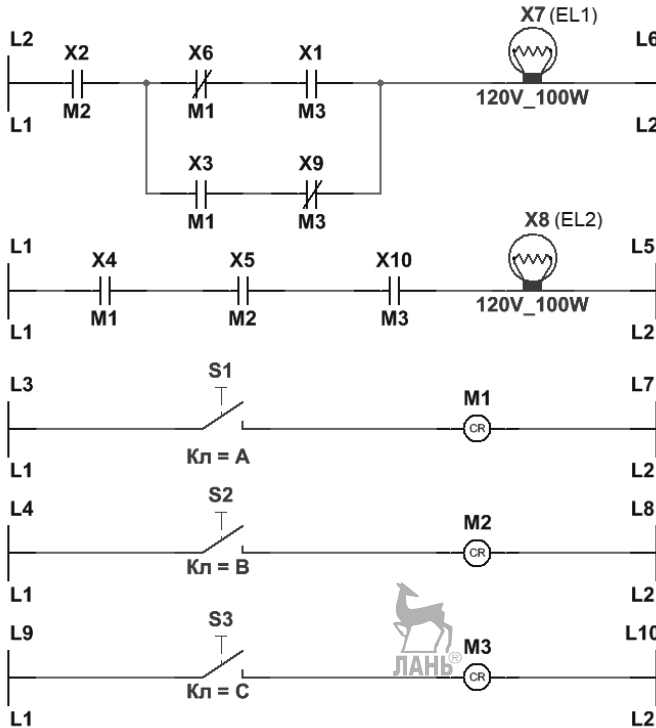


Рис. 2.12. Релейно-контактная схема

Лампочки X7, X8 будут зажигаться при замыкании ключей S1, S2, S3 в соответствии с заданным алгоритмом включения.

Пример 2.7. Имеется электродвигатель, вал которого может вращаться вправо или влево, в зависимости от того, какой замкнут ключ. При одновременном замыкании ключей питание блокируется, чтобы не повредить двигатель. При этом загорается

тревожная лампа. Чтобы визуализировать в схеме вращение двигателя вправо или влево установим на выходы соответствующих цепей сигнальные лампы. Составим таблицу истинности. Назначим входным переменным обозначения M1, M2. Лампочкам EL1 и EL2 поставим в соответствие логические функции Y1 и Y2 (вправо, влево), а тревожной лампочке EL3 – функцию Y3 (табл.2.6).

Таблица 2.6

M1	M2	Y1	Y2	Y3
0	0	0	0	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Применим нормальную форму «ИЛИ», чтобы написать логические уравнения

$$Y1 = \overline{M1} \cdot M2; Y2 = M1 \cdot \overline{M2}; Y3 = M1 \cdot M2 \quad (2.6)$$

Составим релейно-контактную схему (рис. 2.13)

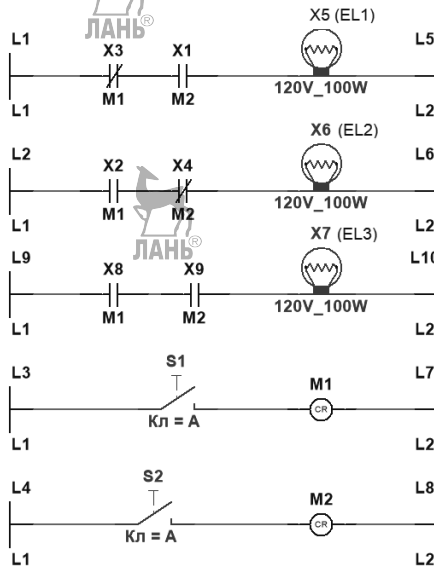


Рис. 2.13. Релейно-контактная схема

При составлении релейно-контактных схем необходимо иметь в виду:

- обозначения под контактами должны соответствовать номерам управляющих катушек;
- все обозначения на схеме должны набираться в английской раскладке;
- номера контактов не влияют на работоспособность схемы.

Изображенные на рис. 2.10, 2.12, 2.13 схемы можно запустить на моделирование в Multisim и проверить заложенную в них логику работы. Запуск на моделирование

осуществляется щелчком по зеленому треугольнику в интерфейсном окне программы. При запуске моделирования на звенья автоматически подается напряжение.

Отметим, что релейно-контактные схемы можно представить в виде логических схем и, наоборот, логические схемы можно представить в виде релейно-контактных схем. Построим логическую схему по уравнениям (2.6). Логические элементы в программе Multisim можно найти по адресу: База данных: *Основная*; Раздел: *Misc Digital*; Семейство: *TTL*. Логическая схема показана на рис. 2.14.

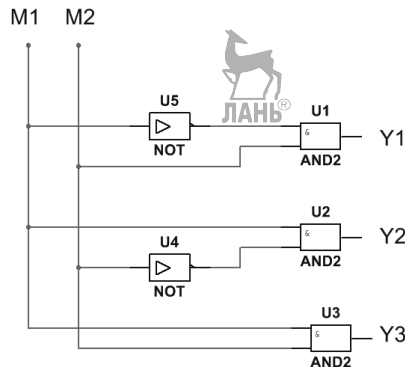


Рис. 2.14. Логическая схема

Чтобы смоделировать работу этой схемы в Multisim, надо добавить источник питания, индикаторы и ключи, как показано на рис. 2.15.

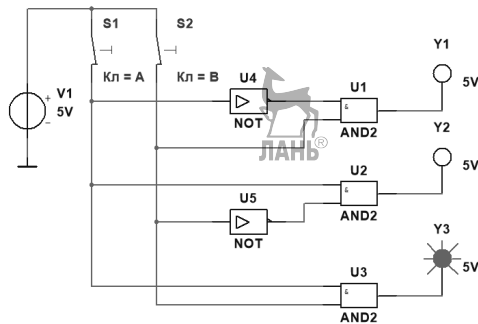


Рис. 2.15. Моделирование логической схемы в Multisim

Замыкая и размыкая ключи получаем значения переменных Y на выходе.

Контрольные вопросы и задания

1. Что называется логической функцией, и какие значения она может принимать?
2. Перечислить названия основных логических функций и привести их условно-графические обозначения по ГОСТ.

3. Какие знаки используются при аналитической записи логических функций «И», «ИЛИ», «НЕ»?
4. Привести формулу для вычисления количества строк таблицы истинности при заданном количестве логических переменных?
5. Из каких этапов состоит схемный синтез (процесс создания логической схемы)?
6. Как применить нормальную форму «ИЛИ» в таблице истинности при выводе уравнения логической функции?
7. Как применить нормальную форму «И» в таблице истинности при выводе уравнения логической функции?
8. Какой виртуальный прибор программы Multisim может использоваться для упрощения логической функции?
9. Чем отличаются контакты NO от контактов NC?
10. Каким образом на релейно-контактной схеме в программе Multisim указывается связь катушки реле и контакта?
11. Холодильный склад оборудован тремя датчиками температуры. Если один из датчиков указывает на перегрев, то это предполагается случайным срабатыванием и холодильный агрегат не включается. Если срабатывают два и более датчика, то включается холодильный агрегат.
Составить логическую схему.
12. Бортовой компьютер контролирует работу 4-х систем автомобиля. При неисправности одной из систем выдается короткий гудок. При неисправности любых двух и более систем выдается длинный гудок.
Составить логическую схему.
13. Четыре переключателя А, В, С, D управляют работой трех насосов. Первый насос включается, если сигнал на его включение поступает с одного любого переключателя. Второй насос включается, если на него поступают сигналы с двух любых переключателей. Третий насос включается, если на него поступают сигналы с трех любых переключателей.
Составить логическую схему.



ГЛАВА 3. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ



3.1. Модульное исполнение

Программируемые логические контроллеры (ПЛК, англ. PLC – Programmable Logic Controller) имеют ряд особенностей, отличающих их от других электронных приборов, применяемых в промышленности:

- в отличие от микроконтроллера (однокристального компьютера), предназначенного для управления электронными устройствами, логический контроллер предназначен для управления технологическими процессами в промышленном производстве;
- в отличие от компьютера, логический контроллер охватывает более широкий круг выполняемых технологических операций, включающий регистрацию поступающих от датчиков сигналов, обработку их по соответствующим алгоритмам, и выдачу команд на исполнительные механизмы;
- в отличие от встраиваемых систем, ПЛК изготавливаются как самостоятельные изделия, отдельные от управляемого ими оборудования.

Программируемый контроллер функционирует в циклическом режиме. Цикл состоит из нескольких этапов:

- считывание входных сигналов, в том числе от клавиатуры;
- обработка поступившей информации;
- выдача управляющих сигналов на выходы.

В силу специфики работы контроллера программирование ПЛК имеет отличия от традиционного программирования. При программировании ПЛК используются флаги - булевы переменные, которые отслеживают прохождение контроллером тех или иных процедур. Например, флаги могут отслеживать процедуру начальной инициализации системы после сброса или включения питания. Эту процедуру необходимо исполнять однократно, поэтому в программу вводят булеву переменную (флаг) завершения процедуры инициализации. Программа анализирует этот флаг, и в последующем обходит процедуру инициализации.

Конструктивно ПЛК, как правило, состоит из нескольких модулей (рис. 3.1). Каждый модуль предназначен для выполнения определенных действий и имеет средства коммуникации со смежным оборудованием. Модульный принцип позволяет получить оптимальный ПЛК для решения конкретной задачи и минимизировать затраты на оборудование. При этом сохраняется возможность нарастить систему для решения более сложных задач автоматизации.

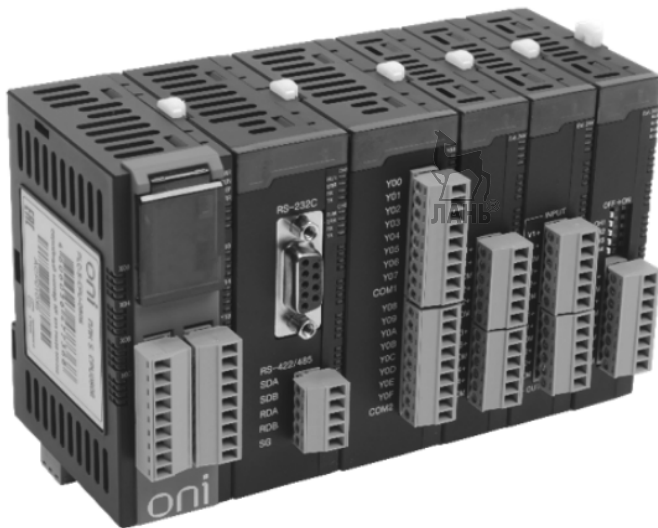


Рис. 3.1. Модульное исполнение контроллера

У некоторых производителей в линейке программируемых логических контроллеров встречаются как высокопроизводительные модели, предназначенные для управления сложными технологическими процессами, так и более скромные по своим возможностям контроллеры для решения задач малой автоматизации. Они получили название программируемых логических реле (ПЛР). Обычно они изготавливаются в модульном корпусе с креплением на DIN-рейку, имеют ограниченный набор входов/выходов и оснащаются кнопками управления на передней панели. Некоторые модели ПЛР имеют небольшой дисплей, что позволяет программировать контроллер непосредственно с лицевой панели. Часто возможности ПЛР можно значительно расширить, подключив к ним несколько модулей расширения. Поэтому граница между ПЛК и ПЛР довольно условна.

Ниже перечислены некоторые **области применения** программируемых логических контроллеров и программируемых логических реле:

- автоматизация производственных процессов изготовления и сборки;
- автоматизация тепличных комплексов;
- управление насосными станциями;
- управление системами обогрева и отопления гражданских и промышленных зданий;
- управление приточно-вытяжной вентиляцией и центральными кондиционерами, климат-контроль;
- управление системами водоснабжения;
- управление системами освещения;
- управление лифтами и подъемниками;

- управление системами открывания и закрывания ворот и дверей;
- управление системами «Умный дом»,
- управление системами складского хозяйства;
- управление сушильными и холодильными камерами;
- управление системами распределения электрических нагрузок.

Разработкой и производством программируемых логических контроллеров и программируемых логических реле занимается большое количество зарубежных и отечественных компаний, среди которых можно отметить Schneider Electric, OMRON, SIEMENS, Mitsubishi Electric, ABB, Moeller, OВЕН (OWEN), ОНИ (ONI) и многие другие.

3.2. Внутренняя структура контроллера

Типовая внутренняя структура ПЛК изображена на рис. 3.2. ПЛК включает в себя микропроцессор (МП), память (ПЗУ и ОЗУ), устройства ввода вывода, аналого-цифровой преобразователь (АЦП), цифро-аналоговый преобразователь (ЦАП), коммуникационные интерфейсы.

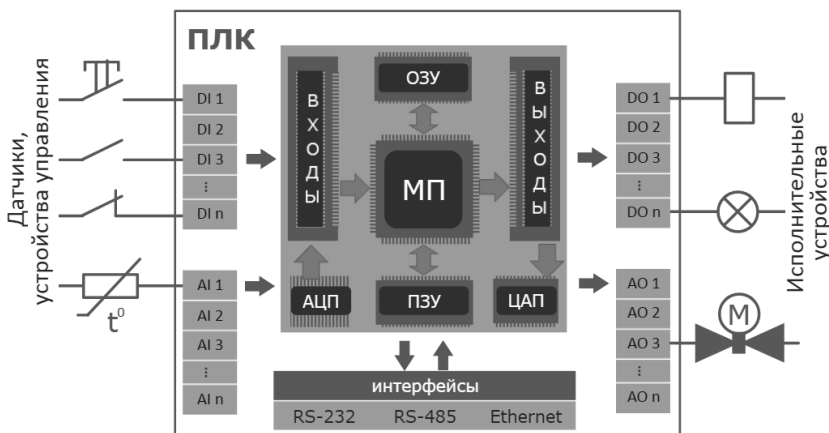


Рис. 3.2. Типовая внутренняя структура ПЛК

Микропроцессор – это программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и реализованное в одной или нескольких больших интегральных схемах (БИС). В состав микропроцессора входят арифметико-логическое устройство, управляющее устройство и блок внутренних регистров. Рабочая частота процессора задается **генератором тактовых импульсов**. Генератор тактовых импульсов вырабатывает периодические импульсы, синхронизирующие работу всех узлов компьютера. Тактовая частота генератора измеряется количеством тактов в секунду. Такт - это промежуток времени между началом подачи текущего импульса и началом подачи

следующего. На выполнение процессором каждой операции отводится определенное количество тактов. Тактовая частота измеряется в мегагерцах (МГц). Частота работы современных микропроцессоров составляет десятки и сотни мегагерц. Поэтому обычно длительность такта измеряется несколькими миллисекундами (мс).

Другая важная характеристика микропроцессора, определяющая его быстродействие – разрядность. **Разрядностью** называют максимальное количество разрядов двоичного кода, которые могут обрабатываться микропроцессором одновременно. Разрядность микропроцессора определяется разрядностью регистров, в которые помещаются обрабатываемые данные. Например, если регистр имеет размер 1 байт, то разрядность процессора равна 8, если 2 байта, то разрядность процессора равна 16, если 4 байта, то разрядность 32, если 8 байт, то разрядность 64.

Память разделяется на две составные части: **постоянное запоминающее устройство (ПЗУ)** и **оперативное запоминающее устройство (ОЗУ)**. Постоянное запоминающее устройство (в зарубежной терминологии ROM – Read Only Memory) - это устройство памяти, в котором хранится программа, обеспечивающая работоспособность микропроцессорной системы. Программа разрабатывается заранее изготовителем системы и размещается в ПЗУ. В таких случаях говорят, что программа жестко «зашита» в запоминающем устройстве. Если раньше запрограммировать ПЗУ можно было только один раз, и после введения программы содержимое памяти уже нельзя было изменить, то сейчас для ПЗУ используются микросхемы памяти, которые допускают многократное перепрограммирование. Такая память называется EEPROM (Electrically Erasable Programmable ROM). Информация, записанная в ПЗУ, сохраняется при отключении питания.

ОЗУ – оперативное запоминающее устройство (в зарубежной терминологии RAM – Random Access Memory) используется для текущего хранения данных и результатов вычислений, а в некоторых микропроцессорных системах и для хранения программ. ОЗУ допускает как запись, так и считывание информации. Информация, содержащаяся в ОЗУ, стирается при отключении питания.

Емкость памяти измеряется в килобайтах, мегабайтах, гигабайтах:

- 1 Кбайт = 2^{10} байт = 1024 байт;
- 1 Мбайт = 2^{10} Кбайт = 2^{20} байт;
- 1 Гбайт = 2^{10} Мбайт = 2^{20} Кбайт = 2^{30} байт.

Память можно представить, как длинную страницу, состоящую из отдельных строк. Каждая такая строка называется ячейкой памяти и, в свою очередь, разделяется на двоичные разряды. Содержимым любого разряда может быть либо 0, либо 1. Один разряд называется битом. Все ячейки памяти пронумерованы. Номер ячейки называют её адресом. Наличие у каждой ячейки адреса позволяет отличать ячейки друг от друга, обращаться к любой ячейке, чтобы записать в неё новую информацию или извлечь информацию, которая в ней хранится.

Устройства ввода-вывода являются устройствами сопряжения, обеспечивающими связь микропроцессорной системы с периферийными

устройствами. Устройства сопряжения называют интерфейсными устройствами или просто интерфейсом системы. Интерфейсы обеспечивают необходимое согласование (сопряжение) устройств по форме представления сигналов (аналоговое, цифровое), по величине сигналов и по последовательности прохождения информации.

Взаимодействие всех составных частей ПЛК между собой осуществляется благодаря внутренней магистрали (**шине**), по которой передаются команды, данные и адреса памяти в виде двоичных чисел.

Работа ПЛК происходит циклически или по кругу, т.е. контроллер выполняет некоторую последовательность действий постоянно и многократно в течение всего времени работы. Этот процесс заканчивается только при выключении ПЛК. После включения питания ПЛК выполняет ряд операций по подготовке к работе. К ним относятся самодиагностика, очистка памяти, загрузка пользовательской программы и другие. После этого ПЛК переходит к циклическому выполнению действий. Первое действие – это опрос и чтение информации на входах ПЛК. Второе действие – выполнение пользовательской программы. Третье действие – передача результатов выполнения программы на выходы ПЛК. После этого идет этап внутренних дополнительных действий. Он включает в себя обслуживание ресурсов, настройку таймеров, АЦП и других встроенных в ПЛК устройств. После этого процесс повторяется заново.

Поскольку микропроцессор является цифровым вычислительным устройством, сигналы, поступающие на аналоговые входы, должны быть переведены в цифровую форму. Эту функцию выполняет аналого-цифровой преобразователь. При выводе сигналов из микропроцессора на аналоговые выходы, их нужно преобразовать из цифровой формы в аналоговую. Эту функцию выполняет цифро-аналоговый преобразователь.

3.3. Входы и выходы контроллера

Дискретные входы (DI – digital input). Входы этого типа предназначены для получения дискретных (двоичных) сигналов. К дискретным входам ПЛК допускается подключать только такие устройства, которые формируют дискретные сигналы, например, ключи «включено-выключено». Источники сигналов обычно используют внешний источник питания. Если дискретный вход рассчитан на напряжение 24 В, то напряжение внешнего источника питания не должно превышать 24 В.

Дискретные выходы (DO – digital output) позволяют управлять исполнительными устройствами, которые должны подключаться к источнику питания, например, лампы, реле, контакторы, автоматические выключатели и др. Выходы ПЛК могут быть построены на основе электромагнитных реле, имеющих гальваническую развязку с цепями управления ПЛК, либо на основе полупроводниковых ключей. Применение полупроводниковых ключей позволяет обеспечить высокое быстродействие и практически неограниченный рабочий ресурс выхода. Наиболее часто применяют схему с открытым коллектором и

гальванической развязкой. Для этого используют транзисторные оптопары. Но такие схемы должны содержать ограничители напряжения, чтобы избежать повреждений управляющих цепей ПЛК.

Аналоговые входы (AI – analog input). Основным элементом аналогового входа является аналого-цифровой преобразователь. В качестве входного сигнала могут выступать напряжение или ток. Соответственно, различают АЦП напряжения или тока. К аналоговым входам обычно подключают всевозможные датчики: температуры, давления, перемещения и т.д.

Аналого-цифровой преобразователь (АЦП) — это электронное устройство, преобразующее входной аналоговый сигнал в двоичный цифровой код. **Разрядность АЦП** — это число n , равное количеству разрядов выходного двоичного числа. Весь допустимый диапазон входного аналогового сигнала разбивается на $2^n - 1$ интервалов или шагов квантования (рис. 3.3). Точность аппроксимации аналогового сигнала определяется шагом квантования. Величина шага квантования определяется выражением:

$$\Delta = \frac{U_{max} - U_{min}}{2^n - 1}$$

U_{max} , U_{min} — максимальное и минимальное допустимые значения аналоговой величины на входе АЦП.



Рис. 3.3. Аппроксимация аналогового сигнала дискретными значениями

Из рис. 3.3. следует, что чем меньше шаг квантования, тем выше точность аппроксимации аналоговой величины цифровым кодом.

Аналоговые выходы (АО – analog output). На выходах этого типа формируется аналоговый сигнал. Для этого используется цифро-аналоговый преобразователь (ЦАП), который выполняет функцию, обратную АЦП. В качестве выходного сигнала выступает напряжение или ток. Важно при использовании ЦАП обращать внимание на нагрузочную способность

аналоговых выходов (АО) и не перегружать их, иначе выход будет повреждён.

Аналоговые величины, которые измеряет датчик, могут иметь различную физическую природу. Это могут быть температура, давление, угловая скорость, перемещение и т.д. Но поскольку контроллер – электрический прибор, то физическая величина должна быть преобразована в электрическую величину. Если это преобразование выполняет датчик, то на выходе датчика должны быть напряжение или ток. Поскольку аналоговые входы ПЛК (с питанием 12 или 24 В постоянного тока) могут обрабатывать напряжения от 0 до 10 вольт или токи 0...20 мА/4...20 мА, то электрические сигналы на выходе датчика должны лежать именно в этом диапазоне. Преобразователи сигналов могут либо непосредственно встраиваться в датчики, либо устанавливаться отдельно, как показано на рис. 3.4.

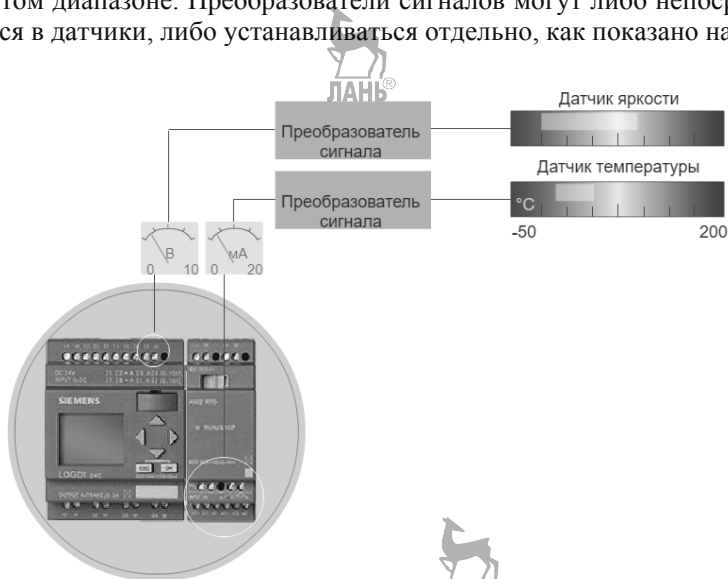


Рис. 3.4. Согласование сигналов датчиков с входами контроллера

Электрические сигналы с выхода датчика преобразуются контроллером во внутренние нормализованные числа от 0 до 1000. Например, входное напряжение, лежащее в диапазоне от 0 до 10 В, преобразуется контроллером в нормализованное число, лежащее в диапазоне от 0 до 1000. Чтобы установить связь между физическими значениями параметров, измеряемых датчиком, и нормализованными значениями величин, с которыми работает контроллер, вводится линейная зависимость

$$A = k \cdot N + b \quad (3.1)$$

где k – усиление;

b – смещение;

A – аналоговая величина, измеренная датчиком (физическое значение измеряемой величины);

N – нормализованное, преобразованное контроллером, значение аналоговой величины, лежащее в диапазоне 0...1000.

Подставляя в формулу (3.1) максимальное и минимальное показания датчика и максимальное и минимальное нормализованные значения, получим систему уравнений

$$A_{max} = k \cdot N_{max} + b \quad (3.2)$$

$$A_{min} = k \cdot N_{min} + b$$

где A_{max} , A_{min} – максимальное и минимальное значения физической величины, измеряемой датчиком;

N_{max} , N_{min} – максимальное и минимальное значения нормализованной величины в контроллере.

Из системы уравнений (3.2) находим

$$k = \frac{A_{max} - A_{min}}{N_{max} - N_{min}} \quad (3.3)$$

$$b = \frac{A_{min} \cdot N_{max} - A_{max} \cdot N_{min}}{N_{max} - N_{min}} \quad (3.4)$$

Если одна из величин k или b найдена, то для нахождения другой величины можно использовать формулы

$$k = \frac{A_{min} - b}{N_{min}} \quad (3.5)$$

$$b = A_{min} - k \cdot N_{min} \quad (3.6)$$

Формулы (3.5), (3.6) получаются из второго уравнения системы (3.2).

Пример 3.1. Найти *Усиление* и *Смещение* для термопары, измеряющей температуру от -30 до $+70^\circ\text{C}$ и выдающей на выходе напряжение от 0 до 10 В (т.е. от 0 до 1000 в контроллере).

Решение. Находим *Усиление* по формуле (3.3)

$$k = \frac{70 - (-30)}{1000 - 0} = 0,1$$

Находим *Смещение* по формуле (3.6)

$$b = -30 - 0,1 \cdot 0 = -30$$

Пример 3.2. Температурный датчик измеряет температуру от -50 до $+100^\circ\text{C}$ и преобразует их в диапазон напряжений от 0 до 10 В. Какое нормализованное значение контроллера соответствует температуре $+40^\circ\text{C}$?

Решение. Находим *Усиление* и *Смещение* по формулам (3.3), (3.6)

$$k = \frac{100 - (-50)}{1000 - 0} = 0,15$$

$$b = -50 - 0,15 \cdot 0 = -50$$

Из (3.1) находим требуемое значение N

$$N = \frac{A - b}{k} = \frac{40 - (-50)}{0,15} = 600$$

Пример 3.3. Датчик угловой скорости вращения измеряет скорость от 500 до 3000 об/мин и выдает на выходе значения от 4 до 20 мА, которым соответствуют нормализованные значения от 0 до 1000 в контроллере.

Какое значение тока, и какое значение угловой скорости соответствуют нормализованному числу 700?

Решение. Найдем вначале показания датчика (значение тока на выходе датчика).

Рассчитаем усиление и смещение по формулам (3.3), (3.6)

$$k = \frac{20 - 4}{1000 - 0} = 0,016$$

$$b = 4 - 0,016 \cdot 0 = 4$$

По формуле (3.1) найдем показания датчика

$$A = 0,016 \cdot 700 + 4 = 15,2 \text{ мА}$$

Теперь найдем значение угловой скорости. Для этого можно воспользоваться теми же формулами, а можно определить значение скорости на основании пропорции, обозначив в ней неизвестное число оборотов буквой x

$$\frac{x - 500}{3000 - 500} = \frac{15,2 - 4}{20 - 4}$$

Отсюда

$$x = 2500 \frac{11,2}{16} + 500 = 2250 \text{ об/мин}$$

Подведем краткие итоги:

1) Контроллер может считывать на аналоговом входе электрическое напряжение от 0 до 10 В или ток от 0 до 20 мА или от 4 до 20 мА. Поэтому физические величины (например, температура, давление, частота вращения и т.п.) должны быть преобразованы в электрические величины в этом диапазоне. Это преобразование выполняется либо непосредственно датчиком, либо отдельным устройством, как показано на рис. 3.4. (Примечание. Имеются аналоговые входы, разработанные специально для температурных датчиков РТ100 или РТ1000 для измерения температур от -50 до +200°C, для которых преобразователи не требуются).

2) Контроллер преобразует напряжение или ток в нормализованное число в диапазоне от 0 до 1000, например, напряжение 5В в число 500. Потом это число используется в коммутационной программе на входе аналоговой функции.

3) Для того, чтобы согласовать нормализованные значения с конкретной областью изменения физической величины, в программном обеспечении контроллера используются специальные аналоговые функции, учитывающие усиление и смещение, например, аналоговый усилитель.

4) На выходе контроллера нормализованные значения можно преобразовывать обратно в электрические величины, например, напряжение. В этом случае напряжение может принимать значения в пределах от 0 до 10 В. С помощью этого напряжения контроллер может управлять внешним исполнительным механизмом.

3.4. Промышленные шины

Для связи ПЛК с другими цифровыми устройствами используются промышленные шины – унифицированные технические, программные и конструктивные средства, позволяющие устройствам взаимодействовать друг с другом. С помощью подобных шин контроллер может быть соединен с датчиками, исполнительными устройствами, другими ПЛК и т.д. Наиболее популярными являются промышленные шины с открытой архитектурой. Протоколы таких шин стандартизованы и доступны для общего применения. Благодаря этому разработчики могут применять устройства различных компаний в одном проекте.

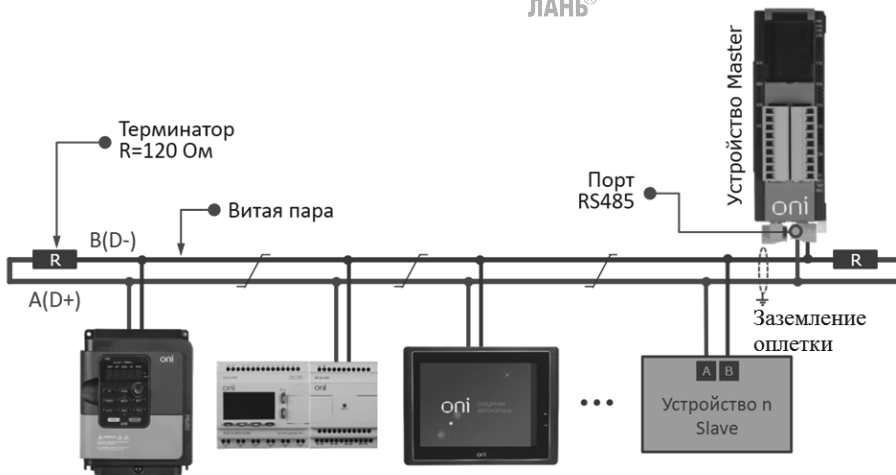


Рис. 3.5. Промышленная шина Modbus

В настоящее время используется более десяти видов различных промышленных шин. Одной из самых распространенных является промышленная шина Modbus (рис. 3.5). Протокол этой шины Modbus RTU получил широкое распространение благодаря своей простоте и универсальности. Он обеспечивает передачу данных в двоичном коде. В сети Modbus используется соединение устройств по топологии «общая шина», при этом реализуется модель взаимодействия master-slave (ведущий-ведомый). Модель подразумевает наличие в сети главного устройства – ведущего, и нескольких подчиненных устройств – ведомых. Обмен может быть инициирован только ведущим устройством. Протокол позволяет ведущему устройству обратиться к любому ведомому устройству или ко всем ведомым устройствам одновременно. Запрос содержит команду, а при необходимости и данные, которые необходимо обработать. После получения запроса ведомое устройство выполняет предписанные действия и формирует ответ. Для подключения к сети используется преимущественно интерфейсы RS-232, RS-485, RS-422.

Интерфейс RS-232 принципиально не позволяет создавать сети, так как может использоваться для соединения только двух устройств (так называемое

соединение “точка - точка”), поэтому одним из наиболее распространенных интерфейсов связи в Modbus является интерфейс RS-485. Сеть, построенная на интерфейсе RS-485, представляет собой приемопередатчики, соединенные при помощи витой пары с волновым сопротивлением 120 Ом. Для защиты от помех экран (оплетка) витой пары заземляется в любой точке, но только один раз. Выбор точки, в которой следует заземлять кабель, не регламентируется стандартом, но, как правило, экран линии связи заземляют на одном из ее концов (рис. 3.6).

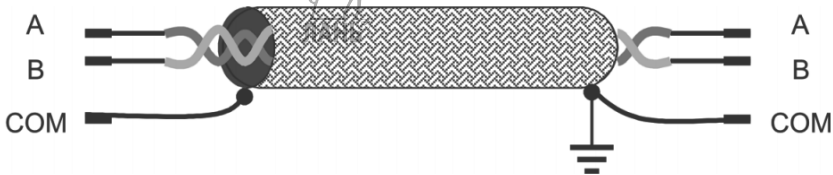


Рис. 3.6. Витая пара интерфейса RS-485

На расстояниях до 600 м допускается использовать витую пару с медной жилой сечением 0,35 мм (например, кабель КММ 2x0,35). Для расстояний больше 600 метров сечение кабеля необходимо пропорционально увеличить.

В основе интерфейса RS-485 лежит принцип дифференциальной передачи данных. Суть его заключается в передаче одного сигнала по двум проводам. Причем по одному проводу (условно А) идет оригинальный сигнал, а по-другому (условно В) – его инверсная копия. Другими словами, если на одном проводе "1", то на другом "0" и наоборот. Таким образом, между двумя проводами витой пары всегда есть разность потенциалов: при "1" она положительна, при "0" – отрицательна (рис. 3.7).

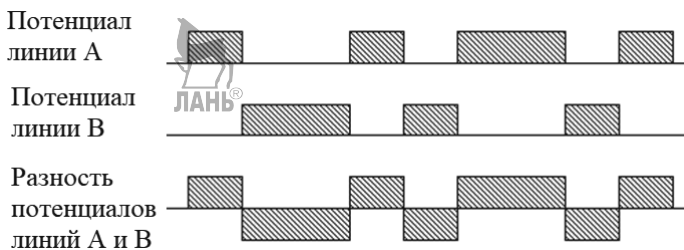


Рис. 3.7. Схема передачи сигнала по интерфейсу RS-485

Именно этой разностью потенциалов и передается сигнал. Такой способ передачи обеспечивает высокую устойчивость к синфазной помехе. Синфазной называют помеху, действующую на оба провода линии одинаково. К примеру, электромагнитная волна, проходя через участок линии связи, наводит в обоих проводах потенциал. Но если два провода пролегают близко друг к другу, и вдобавок перевиты, то наводка на оба провода одинакова, поэтому разность потенциалов помехи в двух проводах равна нулю и, тем самым, обеспечивается высокая помехозащищенность линии связи.

Линия связи RS-485 состоит из одного кабеля витой пары. К этому кабелю присоединяются все периферийные устройства (приемники и передатчики). Длинные ответвления (шлейфы) от магистрали до периферийных устройств не допускаются. Устройства подключаются к сети RS-485 последовательно, с соблюдением полярности контактов А и В, как показано на рис. 3.8.

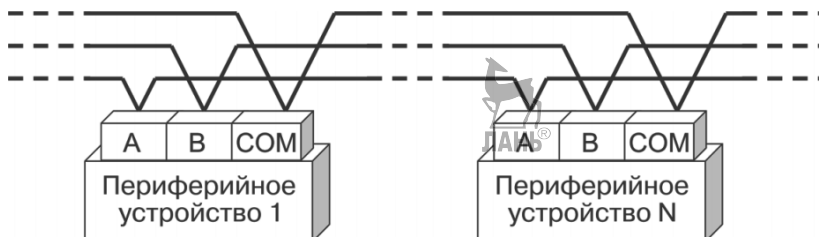


Рис. 3.8. Подключение периферийных устройств к сети RS-485

Для исключения отражений в оба наиболее удаленных конца линии связи включают согласующие резисторы с сопротивлением $R = 120 \text{ Ом}$ (0.25 Вт). Если в системе только один передатчик, и он находится в конце линии, то достаточно одного согласующего резистора на противоположном конце линии.

В обычном PC-совместимом персональном компьютере (не промышленного исполнения) интерфейс RS-485 отсутствует, поэтому необходим специальный адаптер - преобразователь интерфейса RS-485/232 или USB/RS-485. Подключение линии связи RS-485 к компьютеру с помощью преобразователя интерфейса показано на рис. 3.9.

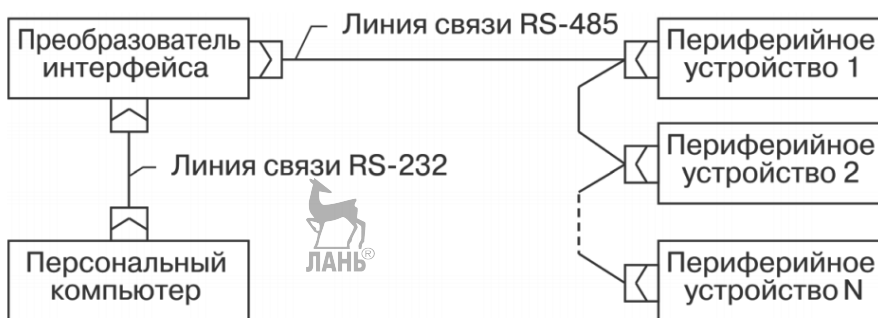


Рис. 3.9. Подключение линии связи RS-485 к компьютеру

Версии протокола Modbus существуют не только для последовательных линий связи. Протокол Modbus в сочетании с Ethernet TCP/IP послужил основой для создания Modbus TCP - открытого протокола Ethernet. Подключение по протоколу Modbus TCP может использоваться с любыми устройствами, имеющими программно-аппаратные средства поддержки протокола TCP/IP.

Контрольные вопросы и задания

1. Чем отличаются программируемые логические контроллеры от программируемых логических реле?
2. Из каких блоков состоит ПЛК?
3. Объясните необходимость использования блоков АЦП и ЦАП в логических контроллерах.
4. Какие два типа памяти используются в ПЛК и чем они отличаются?
5. В каких единицах измеряется память?
6. Сколько байт содержит 1 Мбайт?
7. Почему использование перепрограммируемых ПЗУ является более предпочтительным?
8. Какими буквами обозначаются цифровые и аналоговые входы и выходы ПЛК?
9. Какие устройства подключаются к цифровым входам?
10. Какие устройства подключаются к аналоговым входам?
11. Опишите последовательность работы ПЛК.
12. В какой момент ПЛК заканчивает работу?
13. Чем объясняется повышенная помехозащищенность интерфейса RS-485?
14. Температурный датчик измеряет температуру от (-50°C) до $(+100^{\circ}\text{C})$. Найти нормализованное значение величины, соответствующей температуре $+60^{\circ}\text{C}$.
15. Датчик измеряет угловую скорость от 0 до 3000 об/мин. Найти угловую скорость, соответствующую нормализованному значению 650.
16. Аналоговая величина изменяется в диапазоне от 21 до 47 вольт. Разрядность АЦП, преобразующего аналоговый сигнал в цифровой код, равна 8 ($n=8$). Определить шаг квантования аналогового сигнала.

ГЛАВА 4. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ SIEMENS LOGO!

4.1. Общая характеристика

В модельный ряд контроллеров Siemens входят программируемые логические контроллеры (ПЛК) серии Simatic S7 и логические модули LOGO, которые относятся к классу программируемых логических реле (ПЛР).

Simatic – сокращение, построенное из слов «Siemens» и «Automatic». У истоков этой серии стояла модель S7-200, выпуск которой начался в 1992 году. Текущие модели серии – SIMATIC S7-1200 и SIMATIC S7-1500.



Рис. 4.1. ПЛК Simatic S7-1200

Контроллер SIMATIC S7-1200 (рис. 4.1) предназначен для построения систем низкой и средней степени сложности. Усовершенствованный контроллер SIMATIC S7-1500 (рис. 4.2) предназначен для решения задач среднего уровня сложности и комплексных задач автоматизации. Периферийный контроллер SIMATIC ET 200 предназначен для использования в системах распределенного ввода-вывода. Программный контроллер SIMATIC S7-1500S предназначен для построения компьютерных систем управления.



Рис. 4.2. ПЛК Simatic S7-1500

Контроллеры SIMATIC S7-1200 и SIMATIC S7-1500S программируются с помощью интегрированной среды TIA Portal (Totally Integrated Automation Portal). В TIA Portal интегрированы следующие программные пакеты:

- Simatic Step 7 для программирования контроллеров S7-1200, S7-1500, S7-300, S7-400 и WinAC;
- Simatic WinCC для разработки человеко-машинного интерфейса (от простейших кнопочных панелей до сложных конфигураций уровня SCADA);

- Sinamics StartDrive для программирования и диагностики приводов Sinamics;
- Simatic PLCSIM - симулятор ПЛК;
- Simatic Step 7 Safety;
- Simatic Visualization Architect;
- Simatic Energy Suite.

Регулярно появляются обновления программы или выпускаются новые версии. На данный момент актуальной является версия STEP 7 Basic (Professional) V15 (или TIA Portal 15) Service Pack 1.

Логические модули LOGO! (рис. 4.3) – это универсальные программируемые модули, предназначенные для построения простейших устройств автоматического управления. Они могут использоваться автономно или дополняться необходимым набором модулей расширения. Модульная архитектура поддерживается, начиная с версии – 0BA3. Текущая версия логического модуля – 0BA8. Компактные размеры, относительно низкая стоимость, простота программирования, монтажа и эксплуатации позволяют получать на основе модулей LOGO! множество решений для различных областей промышленного производства и автоматизации зданий. Они могут применяться для решения следующих задач:

- для управления электрическим освещением, дверями, воротами, рольставнями;
- для управления системами кондиционирования и отопления;
- для управления технологическим оборудованием (насосами, компрессорами, прессами);
- для управления автоматическим включением резерва на насосных станциях и в распределительных устройствах;
- для управления поливом в оранжереях и теплицах;
- для управления коммутационной аппаратурой (АВР, АПВ и т.д.);
- для управления конвейерными системами;
- в системах управления дорожным движением.

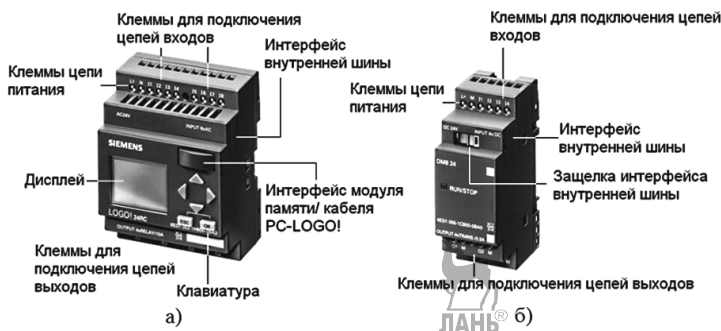


Рис. 4.3. Общий вид модулей Siemens LOGO! а) модуль **LOGO! Basic**, б) модуль расширения

Все модули LOGO! монтируются на 35 мм DIN-рейку или на плоскую поверхность. Объединение логических модулей и модулей расширения в единое устройство осуществляется через внутреннюю шину. Каждый блок может быть дополнен аналоговыми или коммуникационными модулями, подключаемыми к нему с правой стороны. Подключение модуля расширения (рис. 4.4) к внутренней шине логического модуля можно выполнить лишь в том случае, если его кодировочные штифты попадают в кодировочные пазы предшествующего модуля. Для объединения внутренней шины обоих модулей достаточно перевести защелку на передней панели модуля расширения в рабочее положение.

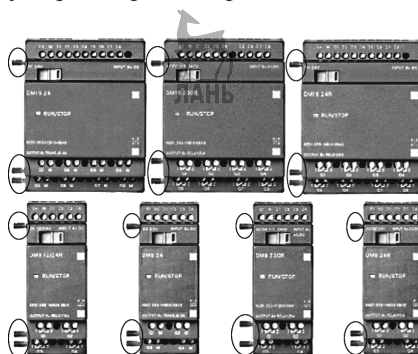


Рис. 4.4. Модули расширения

4.2. Базовые модули LOGO!

Семейство LOGO! включает в себя следующие компоненты:

- Базовые логические модули:
 - LOGO! Basic с встроенной клавиатурой и дисплеем;
 - LOGO! Pure без клавиатуры и дисплея.
- Модули расширения:
 - 8- и 16-канальные модули ввода-вывода дискретных сигналов DM8 и DM16;
 - 2-канальные модули ввода аналоговых сигналов AM2, AM2 RTD (AM2 RTD предназначен для подключения температурных датчиков PT100);
 - 2-канальный модуль вывода аналоговых сигналов AM2 AQ.
- 4-канальные неуправляемые коммутаторы Industrial Ethernet LOGO! CSM.
- Модуль LOGO! CMR 2020 для поддержки GSM/GPS соединений.
- Текстовый дисплей LOGO! TDE.
- Модули коммутации 3-фазных цепей переменного тока LOGO! Contact.
- Модули блоков питания LOGO! Power.
- Дополнительные принадлежности.
- Программное обеспечение LOGO! Soft Comfort.

В настоящее время Siemens выпускает параллельно логические модули трех поколений: LOGO! 0BA6, LOGO! 0BA7 и LOGO! 0BA8. Наиболее

перспективными являются логические модули LOGO! 0BA8. Логические модули LOGO! 0BA6 и 0BA7 в новых проектах использовать не рекомендуется. Все без исключения логические модули могут использоваться в качестве готовых функционально законченных блоков управления. При необходимости они могут дополняться модулями расширения. Модули LOGO! 0BA6 и 0BA7 позволяют использовать общий набор модулей расширения. Модули LOGO! 0BA8 имеют собственный набор модулей расширения и не могут работать с модулями расширения предшествующих версий. Сравнительные характеристики базовых модулей приведены в таблице 4.1.

Таблица 4.1

Параметры	LOGO! 0BA6	LOGO! 0BA7	LOGO! 0BA8
Ширина корпуса	72 мм	108 мм	72 мм
Порт программирования	Специальный, RS232	Ethernet 10/100 Мбит	Ethernet 10/100 Мбит
Внешняя память	LOGO! Memory Card	Стандартная SD карта	Стандартная Micro SD карта
Часы реального времени	Есть	Есть	Есть
Кол-во функциональных блоков	200	400	400
Работа в сети Ethernet	Нет	Есть	Есть
Определяемые пользователем функции (макросы)	Нет	Есть	Есть
Регистрация данных	Нет	Есть	Есть
Поддержка новых функциональных блоков	Нет	Есть	Есть
Кол-во дискретных входов на систему	24	24	24
Кол-во сетевых входов через Ethernet		64	64
Кол-во дискретных выходов на систему	16	16	20
Кол-во сетевых выходов	–	64	64

через Ethernet			
Кол-во аналоговых входов на систему	8	8	8
Кол-во аналоговых входов через Ethernet	–	32	32
Кол-во аналоговых выходов на систему	2	2	8
Кол-во аналоговых выходов через Ethernet	–	16	16
Встроенный Web сервер	–	–	Есть

Примечание:

- 1) Электрические и временные параметры входных и выходных сигналов одинаковы для всех модулей соответствующих модификаций.
- 2) В выделенных строках показаны опции, которые отсутствуют в логических модулях версии 0BA6.

В зависимости от модификации, логические модули рассчитаны на постоянное напряжение питания 12/ 24 В, постоянное и переменное напряжение 24 В, переменное напряжение 115/ 230 В. Напряжение питания модуля определяет и напряжение питания его входных цепей (исключая аналоговые входы). Все логические модули оснащены 8 входными и 4 выходными дискретными каналами. В моделях с питанием постоянным напряжением 12/24 В или 24 В часть входных каналов имеет универсальное назначение, что позволяет использовать:

- все входы для ввода дискретных сигналов постоянного тока;
- входы I1, I2, I7 и I8 для ввода аналоговых сигналов 0...10 В с включением в работу двух (I7 и I8) или четырех входов;
- входы I1, I2, I3 и I4 для подсчета импульсов, следующих с частотой до 5 кГц.

На рис. 4.5 показана допустимая коммутация входных цепей базового модуля, дополненного аналоговым модулем расширения AM2. Базовый модуль предназначен для питания постоянным напряжением 24 В, имеет транзисторные выходы с допустимым максимальным током 0,3 А. К входам I7, I8 базового модуля могут подключаться, как дискретные, так и аналоговые датчики с выходным напряжением 0...10 В. Входы аналогового модуля расширения AM2 расположены в нижней части прибора, к ним могут подключаться аналоговые датчики как с выходным напряжением 0...10 В, так и с токовым выходом 0-20 мА/4-20 мА.

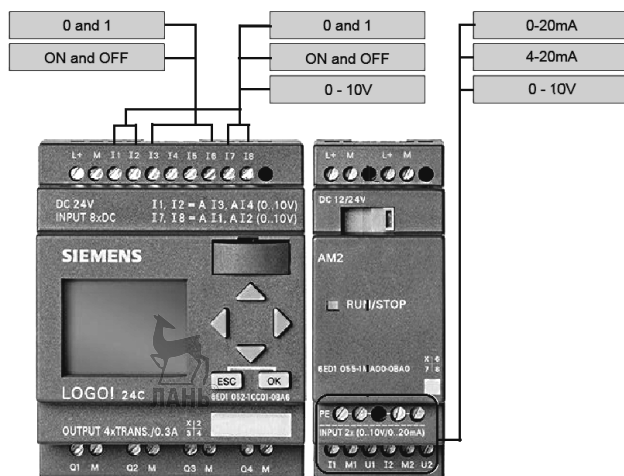


Рис. 4.5. Допустимая коммутация входных цепей

Выходные каскады модулей LOGO! выполняются на основе герконовых реле или транзисторных ключей. В моделях с транзисторными выходами два выхода могут использоваться в импульсном режиме. Например, для формирования сигналов широтно-импульсной модуляции.

Расшифровка обозначений модулей LOGO! и модулей расширения различных версий:

- 12/24: модуль предназначен для постоянного напряжения 12/24 В;
- 24: модуль предназначен либо для постоянного, либо для постоянного и переменного напряжения 24 В;
- 230: версия для постоянного и переменного напряжения 115 - 240 В;
- R: релейные выходы (без символа «R» –модуль имеет дискретные выходы с транзисторными ключами);
- C: имеется встроенный таймер;
- o: версия LOGO! Pure без дисплея (без этого символа – версия LOGO!Basic с дисплеем и клавиатурой) ;
- E: наличие встроенного интерфейса Ethernet;
- DM: модуль ввода-вывода дискретных сигналов (DM8 – 4 входа и 4 выхода; DM16 – 8 входов и 8 выходов);
- AM: аналоговый модуль расширения;
- CM: коммуникационный модуль расширения (например, модуль EIB/KNX);
- TD: текстовый дисплей.

Модификации базовых модулей приведены на рис. 4.6.



4.3. Модули расширения

Наиболее простые устройства управления могут быть построены на основе одного базового модуля. Для построения более сложных устройств базовый модуль дополняется необходимым набором модулей расширения. Обмен данными с большинством модулей расширения выполняется через внутреннюю шину логического модуля. Один базовый модуль LOGO! 0BA8 с необходимым набором модулей расширения способен обслуживать до 24 дискретных и до 8 аналоговых входов, а также до 20 дискретных и до 8 аналоговых выходов. Для повышения быстродействия непосредственно за базовым модулем рекомендуется устанавливать сначала дискретные, потом аналоговые модули расширения. Существуют определенные ограничения на порядок размещения и стыковки модулей расширения. В основном эти ограничения связаны с различными уровнями напряжения питания модулей. Допустимые варианты подключения модулей LOGO! друг к другу и к базовому модулю приведены в таблице 4.2.

Таблица 4.2

Установленный модуль	Подключаемый модуль расширения				
		DM8 12/24R	DM8 24 DM16 24	DM8 24R DM16 24R	DM8 230R DM16 230R

LOGO!12/24 RCE (RCEo)	+	+	+	–	+
LOGO!24 CE (CEo)	+	+	+	–	+
LOGO!24 RCE (RCEo)	+	+	+	–	+
LOGO!230 RCE (RCEo)	–	–	–	+	+
LOGO!DM8 12/24 R	+	+	+	–	+
LOGO!DM8 24	+	+	+	–	+
LOGO!DM8 24R	+	+	+	–	+
LOGO!DM8 230R	–	–	–	+	+
LOGO!DM16 24	+	+	+	–	+
LOGO!DM16 24R	+	+	+	–	+
LOGO!DM16 230R	–	–	–	+	+
LOGO!AM2	+	+	+	–	+
LOGO!AM2 RTD	+	+	+	–	+
LOGO!AM2 AQ	+	+	+	–	+

Примечание:

1. Знаком «+» отмечены допустимые варианты подключений.
2. Выделенным строкам соответствуют базовые модели.

Кроме модулей расширения к каждому логическому модулю LOGO! может быть дополнительно подключен внешний текстовый дисплей LOGO! TDE. Логические модули LOGO! 0BA7 и LOGO! 0BA8 оснащены встроенным интерфейсом Ethernet и позволяют производить дальнейшее расширение своей системы ввода-вывода за счет сетевого обмена данными с другими модулями LOGO! В такой системе один логический модуль выполняет функции ведущего, остальные модули – функции ведомых устройств. К одному ведущему модулю может быть подключено до 8 ведомых модулей. Каждый логический модуль может иметь собственный набор модулей расширения. Программу выполняет только ведущий логический модуль. Ведомые модули собственной программы не имеют и выполняют функции блоков расширения ведущего логического модуля. За счет этого ведущий логический модуль способен обслуживать:

- до 64 сетевых дискретных входов (NI1 ... NI64),
- до 64 сетевых дискретных выходов (NQ1 ... NQ64),
- до 32 сетевых аналоговых входов (NAI1 ... NAI32)
- до 16 сетевых аналоговых выходов (NAQ1 ... NAQ16).

Для исключения ошибок при заказе модулей LOGO! рекомендуется использовать конфигуратор TIA Selection Tool, учитывающий все ограничения, накладываемые на конфигурацию базового модуля с модулями расширения. Этот конфигуратор доступен в Интернете по адресу: www.siemens.com/tia-selection-tool.

Модули расширения дискретных сигналов имеют два исполнения:

- LOGO! DM8 с 4 дискретными входами и 4 дискретными выходами.
- LOGO! DM16 с 8 дискретными входами и 8 дискретными выходами.

Внутренняя шина модулей LOGO! DM не имеет устройств гальванического разделения цепей. Поэтому напряжение питания и тип тока модуля расширения должны совпадать с аналогичными параметрами модуля, к которому он подключается. Для исключения ошибок при монтаже все модули LOGO! DM оснащены кодировочными пазами и штифтами. Выполнить подключение к внутренней шине можно лишь в том случае, если кодировочные штифты модуля расширения вошли в кодировочные пазы предшествующего модуля.

При использовании модулей расширения следует иметь в виду:

- Модули DM8/DM16 могут подключаться только к модулям с таким же уровнем напряжения питания и типом тока.
- Аналоговые и коммуникационные модули могут подключаться к модулям любого типа.

4.4. Подключение внешних цепей

Для подключения внешних цепей логических модулей и модулей расширения LOGO! рекомендуется использовать провода и кабели с медными жилами сечением от 1.5 до 2.5 мм². К одному контакту модуля может подключаться один проводник сечением 2.5 мм² или два проводника сечением 1.5 мм².

Подключение питания. Цепь питания постоянного тока рекомендуется защищать предохранителем. Для этой цели рекомендуется использовать:

- предохранитель 0.8 А в цепи питания модуля LOGO! 12/24;
- предохранитель 2.0 А в цепи питания модуля LOGO! 24.

Цепь питания переменного тока рекомендуется защищать металлооксидным варистором, рассчитанным на 120%-ое номинальное напряжение питания. Например, для этой цели можно использовать варистор S10K275 (рис. 4.7).



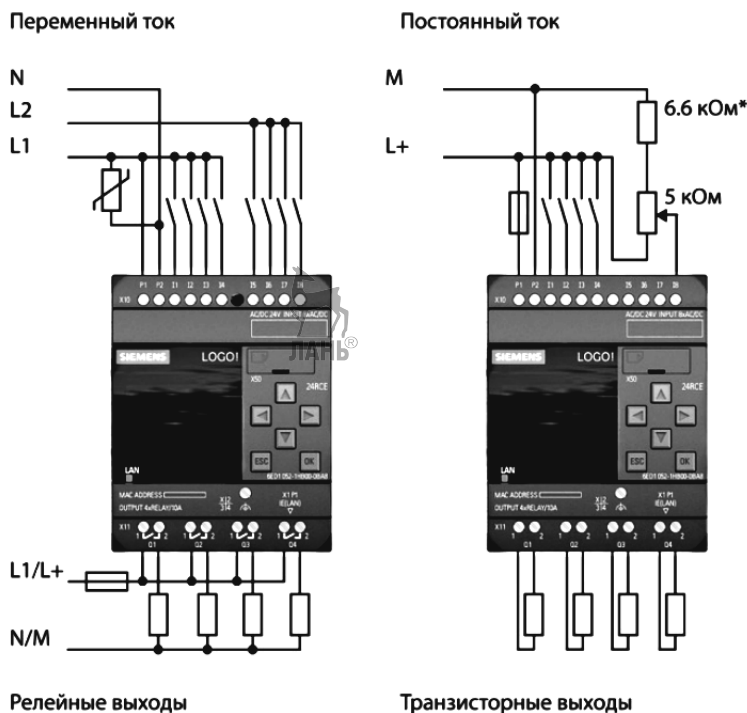


Рис. 4.7. Подключение внешних цепей

Подключение датчиков. В модулях LOGO! 12/24 и LOGO! 24 все дискретные входы объединены в одну потенциальную группу и должны получать питание от общего блока питания. В модулях LOGO! 230 дискретные входы разделены на две потенциальные группы: I1 ... I4 и I5 ... I8. При необходимости каждая группа входов может получать питание от своей фазы. Подключение входов одной группы к разным фазам недопустимо.

Аналоговые входы модулей LOGO! 12/24 и LOGO! 24 рассчитаны на входное напряжение 0...10 В. При использовании напряжения питания 24 В последовательно с датчиком включается резистор сопротивлением 6.6 кОм. Если напряжение питания равно 12 В, то этот резистор не нужен. **Подключение нагрузки.** Транзисторные выходы логических модулей LOGO! оснащены защитой от коротких замыканий в цепи нагрузки и получают питание от внутренней электроники модуля. Дополнительного блока питания нагрузки не требуется. Номинальный ток выхода равен 0.3 А при напряжении 24 В.

В модулях с релейными выходами все выходы выполнены в виде изолированных друг от друга “сухих” контактов. Для питания нагрузки необходим внешний источник питания. Цепь питания нагрузки рекомендуется защищать автоматическим выключателем на 16 А.

Подключение внешних цепей модулей DM8 и DM16. Рекомендации по подключению внешних цепей базовых модулей LOGO! справедливы и для модулей расширения DM8 и DM16 соответствующих модификаций.

4.5. Программное обеспечение LOGO! Soft Comfort



4.5.1. Общие сведения

Для программирования логических модулей LOGO! используется набор функций, встроенных в их операционную систему. Все функции сгруппированы в две библиотеки. Библиотека GF (**General Functions**) содержит набор базовых функций, позволяющий использовать в программе модуля все основные логические операции. Библиотека SF (**Special Functions**) содержит набор специальных функций, к которым относятся триггеры, таймеры, счетчики, компараторы, часы и календари, элементы задержки включения и отключения, генераторы, функции работы с аналоговыми величинами и др. Общий объем программы ограничен 200 функциями для модулей LOGO! 0BA6 и 400 функциями для модулей LOGO! 0BA7 и 0BA8. Это значит, что один модуль LOGO! способен заменить схему, включающую в свой состав несколько сотен электронных и электромеханических компонентов. Программирование может выполняться тремя способами:

- с клавиатуры модуля LOGO! Basic;
- установкой запрограммированного модуля или карты памяти;
- с компьютера, оснащенного пакетом программ LOGO! Soft Comfort.

Программирование с клавиатуры логического модуля.

Программирование модулей LOGO! с клавиатуры логического модуля выполняется на языке FBD. Этот вариант программирования возможен только для модулей LOGO! Basic с дисплеем. Процесс программирования сводится к извлечению из библиотеки требуемых функций, определению соединений входов и выходов данной функции с входами и выходами логического модуля или других функций, а также установке параметров настройки данной функции, например, времени задержки включения или отключения. Во время программирования на экране дисплея отображается только одна из всех используемых в программе функций. Готовая программа может быть переписана в модуль или карту памяти, вставленную в модуль LOGO! Все операции программирования поддерживаются встроенной системой меню модуля.

Программирование с клавиатуры – это трудоемкий и длительный процесс, при котором легко допустить ошибку, поэтому рекомендуется использовать этот тип программирования только для простых коммуникационных программ.

Программирование с помощью модуля/карты памяти.

Программирование логических модулей LOGO! может выполняться установкой заранее запрограммированного модуля или карты памяти. После установки модуля/ карты памяти и включения питания в LOGO! Pure программа

автоматически копируется из модуля памяти в память логического модуля, после чего выполняется автоматический запуск программы. В LOGO! Basic после установки модуля памяти и включения питания на экран дисплея выводится меню, из которого можно произвести перезапись программы из модуля/карты памяти в память логического модуля и осуществить запуск выполнения программы.

Программирование с помощью LOGO! Soft Comfort.

Программное обеспечение LOGO! Soft Comfort предоставляет наиболее широкие возможности по разработке и отладке программ. Разработка программы может выполняться на языках LAD (Ladder Diagram) или FBD (Function Block Diagram). Допускается использование символьных имен для переменных и функций, а также необходимых комментариев. В отличие от программирования с клавиатуры обеспечивается наглядное представление всей программы, поддерживается множество сервисных функций, повышающих удобство разработки и редактирования программы. Разработка, отладка и полное тестирование работы программы может осуществляться в автономном режиме без наличия реального модуля LOGO! Готовая программа может загружаться в логический модуль или записываться в модуль/карту памяти, а также сохраняться на жестком диске компьютера. Текущая версия LOGO! Soft Comfort V8.0 позволяет программировать логические модули LOGO! всех поколений.

Далее будет рассматриваться только программирование с помощью ПО LOGO!Soft Comfort, как наиболее информативный и совершенный способ составления и редактирования программ.

4.5.2. Пользовательский интерфейс

В процессе ознакомления с программным обеспечением (ПО) LOGO!Soft Comfort будут рассматриваться примеры составления программ. Эти примеры не будут содержать специфических требований к версии программного обеспечения. Поэтому для большинства примеров будет использоваться седьмая версия ПО (V7.0), а для каких-то примеров будет использоваться восьмая версия (V8.0). В данном случае это не принципиально, так как примеры не требуют тех новых возможностей, которые предоставляет восьмая версия.



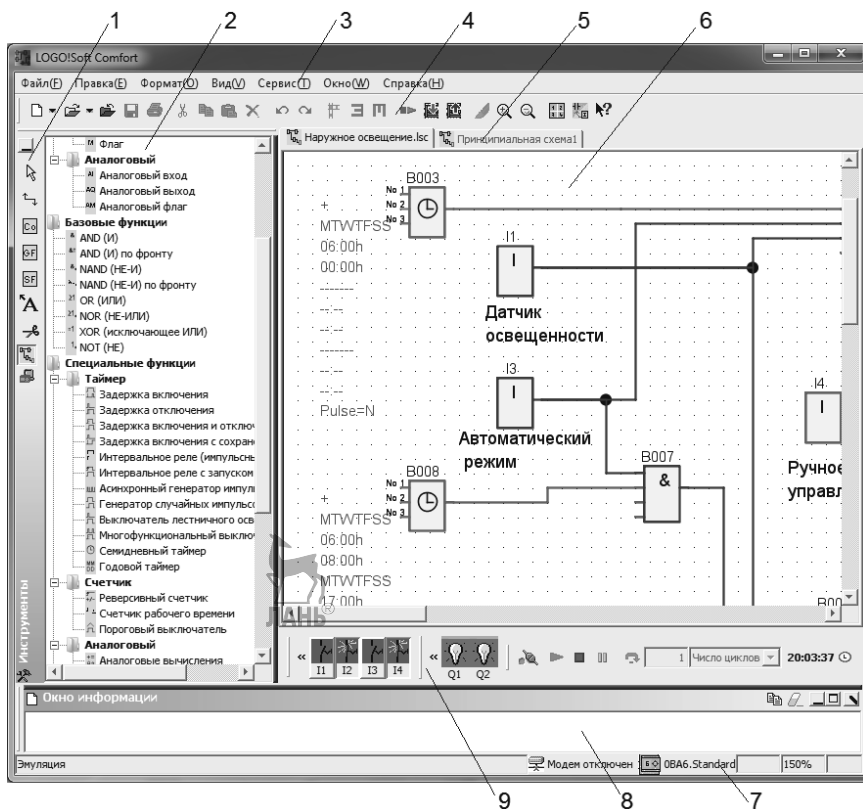


Рис. 4.8. Пользовательский интерфейс

Пользовательский интерфейс программы версии V7.0 показан на рис. 4.8, где цифрами обозначено:

- 1 – панель программирования;
- 2 – панель инструкций. Инструкции представляют собой совокупность базовых функций, специальных функций и постоянных;
- 3 – панель меню;
- 4 – стандартная панель инструментов;
- 5 – вкладки рабочих окон;
- 6 – рабочее окно, в котором создается программа;
- 7 – строка состояния;
- 8 – окно информации;
- 9 – панель инструментов моделирования. (Появляется при режиме «Эмуляция»).

Панель инструментов программирования (поз.1) представлена на рис. 4.9 и содержит следующие инструменты:



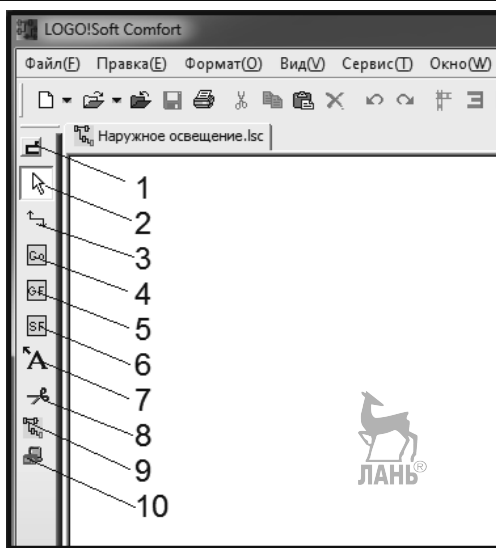


Рис. 4.9. Панель инструментов программирования

- 1 – скрыть/показать каталог инструкций;
- 2 – инструмент выбора;
- 3 – инструмент создания соединений;
- 4 – постоянные и клеммы;
- 5 – базовые функции;
- 6 – специальные функции;
- 7 – вставить комментарий;
- 8 – разрезать/связать соединение;
- 9 – эмуляция;
- 10 – оперативное тестирование.

Панель инструментов моделирования (поз. 9) представлена на рис. 4.10.

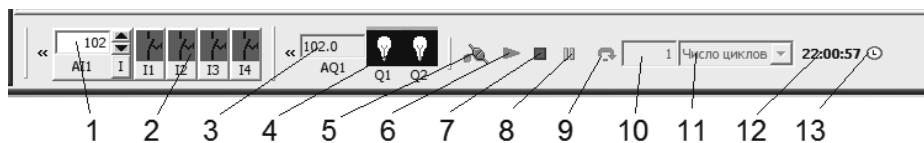


Рис. 4.10. Панель инструментов моделирования

В режиме «Эмуляция» (нажата кнопка 9 панели программирования) появляется панель инструментов моделирования, которая содержит следующие опции и инструменты:

- 1 – внутреннее значение входной аналоговой величины;
- 2 – индикаторы (переключатели) цифровых входов;
- 3 – внутреннее значение выходной аналоговой величины;

- 4 – индикаторы цифровых выходов;
- 5 – индикатор питания;
- 6 – запуск эмуляции;
- 7 – останов эмуляции;
- 8 – пауза;
- 9 – выполнить эмуляцию в пошаговом режиме;
- 10 – число циклов в одном шаге;
- 11 – единица пошагового режима;
- 12 – текущее время;
- 13 – установить текущую дату и время для эмуляции.



Панель инструкций (поз. 2) для восьмой версии программного обеспечения LOGO!Soft Comfort представлена на рис. 4.11. Программа LOGO! Soft Comfort V8 имеет следующие отличительные особенности по сравнению с предыдущими версиями:

- Улучшенный графический интерфейс пользователя (GUI).
- Графические ссылки на функции.
- Графическое отображение сети.
- Автоматическая конфигурация Ethernet интерфейса.
- Автоматическое определение доступных узлов сети.
- Статусная таблица с возможностью сохранения в ПК в формате CSV.
- До 400 функциональных блоков на программу для всех восьми базовых модулей.
- 64 аналоговых флага.
- 64 дискретных флага.
- 4 сдвигающих регистра по 8 бит каждый.
- Наименование порта, пароль и передача параметров.
- Импорт/экспорт наименований портов.
- Замена функциональных блоков.
- Астрономические часы с настраиваемой задержкой включения/выключения.
- Офлайн сетевой симулятор.
- Обмен данными между модулями LOGO! осуществляется с помощью Drag & Drop.

Программы, созданные с помощью более ранних версий ПО LOGO! Soft Comfort могут использоваться без каких-либо ограничений;

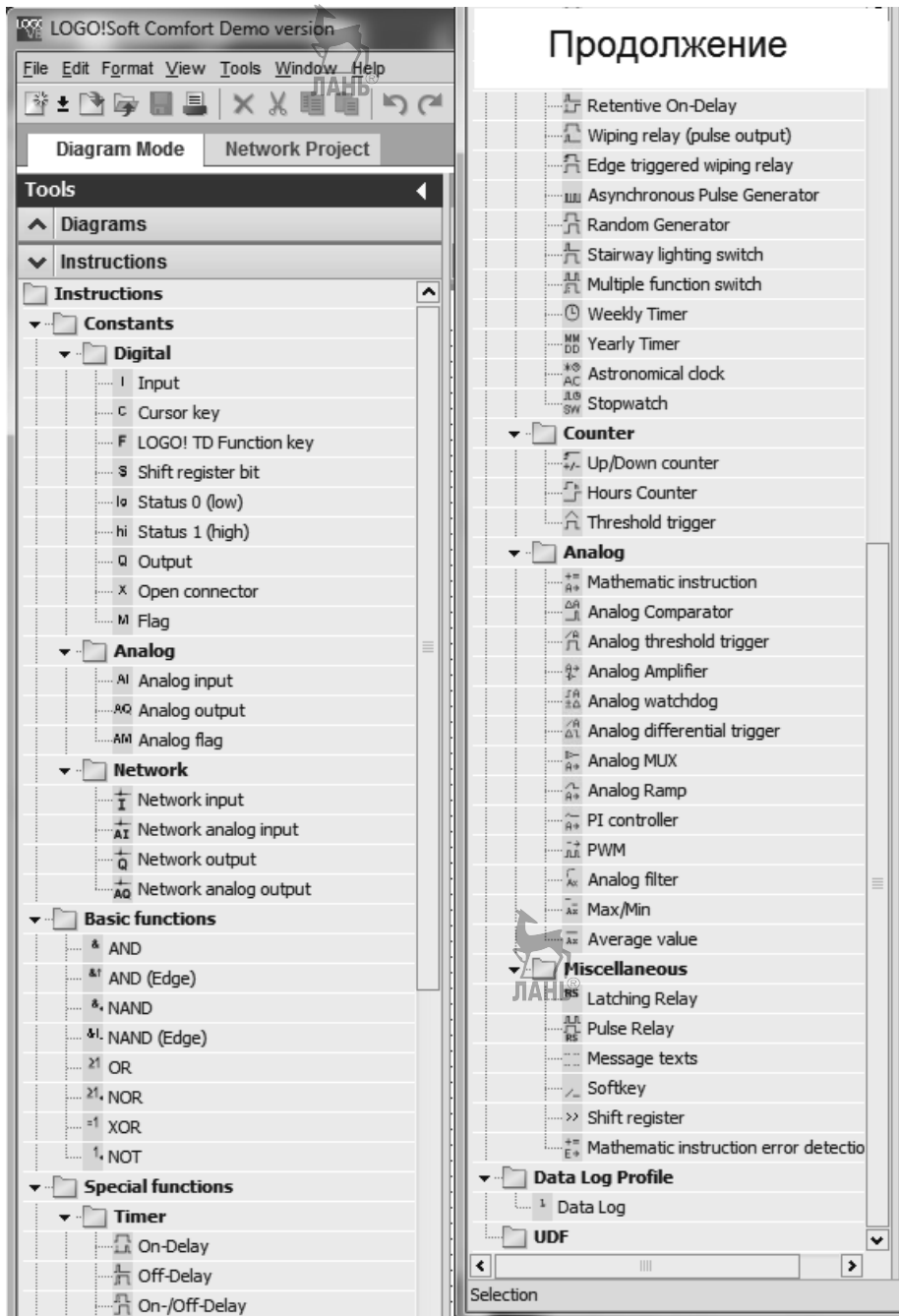


Рис. 4.11. Панель инструкций LOGO!Soft Comfort версии V8.0

4.5.3. Построение коммутационной программы

Программное обеспечение LOGO!Soft Comfort предоставляет пользователю два варианта построения коммутационных программ:

- в виде релейно-контактных схем (LAD);
- в виде функциональных блок-схем (FBD).

Пользователям, которые привыкли работать с принципиальными электрическими схемами, удобнее использовать при построении релейно-контактные схемы. Пользователям, которые привыкли работать с логическими блоками, удобнее использовать функциональные блок-схемы. Чтобы выбрать вариант построения коммутационной программы, нужно щелкнуть по вкладке меню *Сервис* и далее по строке *Параметры*. После этого мы попадаем в окно, представленное на рис. 4.12. Щелкнув по треугольной стрелочке в правой части окна, можно выбрать либо *Редактор функциональных блок-схем*, либо *Редактор релейно-контактных схем*.

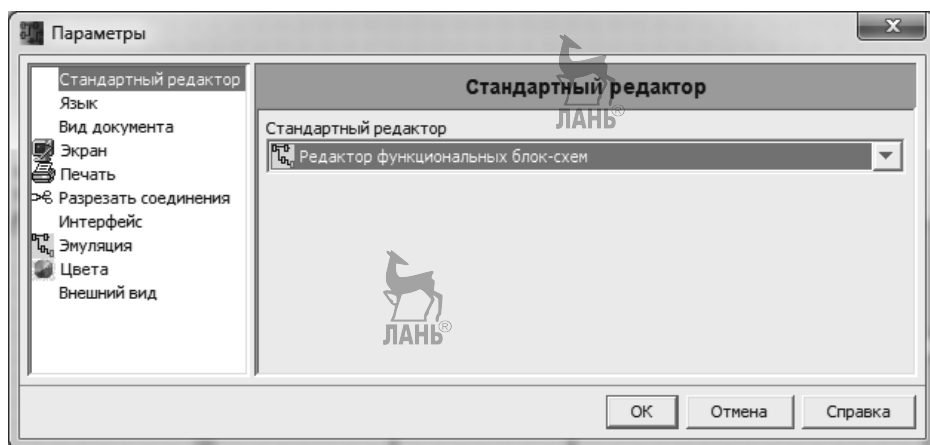


Рис. 4.12. Окно для задания типа коммутационной программы

Для того, чтобы преобразовать уже созданную схему из одного вида в другой можно воспользоваться соответствующей опцией из вкладки меню *Файл*.

Чтобы начать создание коммутационной программы, надо щелкнуть по кнопке *Создать* стандартной панели инструментов. Справа от этой кнопки имеется треугольная стрелочка, с помощью которой можно выбрать тип оформления создаваемой схемы. После щелчка по кнопке *Создать* появляется окошко, в котором надо указать свойства вновь создаваемого проекта. Можно это окно не заполнять, оставить все по умолчанию, и сразу щелкнуть по кнопке *ОК*. Теперь необходимо выбрать функциональные блоки для новой схемы и определить последовательность, в какой они будут выводиться на рабочее поле. Это особенно важно, если потом разработанная программа будет вручную набираться с помощью встроенного меню модуля LOGO!, так как последовательность вывода блоков на рабочее поле определяет

последовательность присваиваемых им номеров. Если же разработанная на компьютере программа будет перезаписываться в память модуля с помощью переходного кабеля между компьютером и модулем, то последовательность нумерации блоков особого значения не имеет.

Блоки располагаются в *Панели инструкций* интерфейсного окна. В разделе *Постоянные (Constants)* находятся входы, выходы, постоянные и др. элементы программирования. В разделе *Базовые функции (GF)* находятся основные логические функции. В разделе *Специальные функции (SF)* находятся специальные функции и т.д. Чтобы вытащить блок (функцию) на рабочее поле, надо щелкнуть по нему левой кнопкой мыши и затем щелкнуть в области рабочего окна. Блок отобразится в рабочем окне. Или можно просто перетащить блок на рабочее поле, зажав левую кнопку мыши. Если теперь дважды щелкнуть по значку блока, то появится выпадающее окно, в котором можно задать свойства блока. Например, для блока *Задержка включения* можно задать время задержки включения в секундах, минутах или часах. При этом заданные параметры отображаются слева от значка блока. На рис. 4.13 для блока *Задержка включения* установлено время задержки – 5 сек.

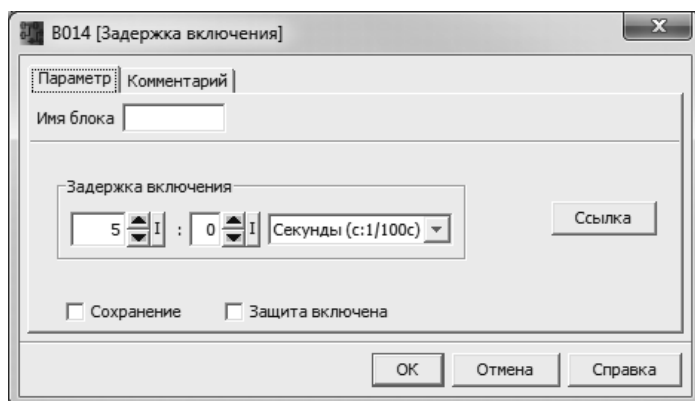


Рис. 4.13. Окно свойств блока *Задержка включения*

После того, как все блоки вытащены на рабочее поле, можно приступить к установлению связей между блоками. Для установления связей используется кнопка *Подключить* (поз. 3 на рис. 4.9) на *Панели программирования*. Чтобы соединить, например, выход одного блока с входом другого, надо подвести курсор к выходу блока, при этом на выходе должен появиться маленький квадратик, нажать левую кнопку мыши и, не отпуская ее, потянуть линию к входу другого блока, пока на этом входе также не появится маленький квадратик. После этого можно отпустить кнопку мыши. Если немного задержать указатель мыши в конечном положении, то справа от указателя отобразится подсказка с названием подсоединяемого вывода. При этом обратите внимание, аналоговые выходы блоков отображаются более жирной линией, нежели цифровые выходы.

Аналоговые выводы можно соединять только с аналоговыми выводами, а цифровые выводы только с цифровыми.

Сформулируем правила, которых надо придерживаться при соединении блоков:

- ✓ Можно подсоединить один выход к нескольким входам.
- ✓ Нельзя подсоединять несколько выходов к одному входу.
- ✓ Нельзя соединить между собой входы и выходы в одном тракте коммутационной программы. Рекурсия не допускается. Возможно соединение с помощью флага, если это необходимо.
- ✓ Специальные функции имеют дополнительные входы, которые не используются для соединений, а служат для задания параметров функции.
- ✓ Не допускается подключение аналоговых входов и выходов к цифровым входам и выходам.
- ✓ Блоки можно перемещать по рабочему полю. Для этого их надо предварительно выделить, щелкнув по ним левой кнопкой мыши. Группы блоков выбираются «захватом» их указателем мыши. Для «захвата» блоков удерживайте левую кнопку мыши нажатой, нарисуйте рамку вокруг блоков, затем отпустите кнопку мыши. «Захваченные» блоки выделяются небольшими красными квадратами в углах блоков. Для произвольного выбора нескольких блоков нужно выделить их один за другим, удерживая нажатой клавишу [Ctrl]. Можно удалять блоки клавишей [Del]. Можно также вырезать, копировать, вставлять блоки при помощи соответствующих значков на панели инструментов.

Соединительные линии можно редактировать. Для этого их надо выбрать щелчком левой кнопки мыши. Выбранные соединительные линии отмечаются круглыми синими маркерами и квадратными синими и красными маркерами. Круглые маркеры могут использоваться для перемещения линий под прямым углом. Квадратные синие маркеры могут использоваться для переназначения начала или конца линии. Можно оптимизировать расположение соединительных линий при помощи инструмента *Разрезать/Связать* (ножницы) на *Панели программирования*. При щелчке ножницами по соединительной линии происходит графическое разделение выбранной соединительной линии, но соединение между блоками продолжает оставаться активным. Открытые концы разрезанного соединения отображаются значками в виде стрелок, указывающими на направление протекания сигнала. Над стрелками отображаются перекрестные ссылки, включая номер противоположного блока и номер вывода противоположного блока. Разрезанное соединение можно закрыть, щелкнув правой кнопкой мыши по стрелке на открытом конце. При этом появляется выпадающее окно (рис. 4.14), в котором надо выбрать команду *Соединить*. Кнопка *Разрезать/связать* является очень эффективным средством для упрощения компоновки больших и запутанных коммутационных схем с большим количеством пересекающихся соединений.

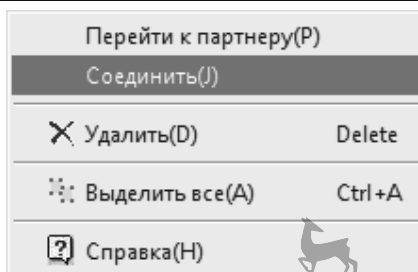


Рис. 4.14. Окно для редактирования соединительной линии

Чтобы быстро перейти к ответной части стрелки, надо в выпадающем окне выбрать команду *Перейти к партнеру*. Это позволит быстро перейти к партнерской стороне разрезанного соединения. Еще одним преимуществом инструмента *Разрезать/связать* является возможность его применения для схем, расположенных более чем на одной распечатываемой странице.

На подготовленной коммутационной программе можно разместить комментарий. Для этого надо воспользоваться кнопкой *Вставить комментарий* на *Панели программирования* (поз. 7 на рис. 4.9). Если при создании комментария щелкнуть по свободной области рабочего поля, то мы получим текст, который можно свободно перемещать по рабочему полю. Если же при создании комментария щелкнуть по блоку, то мы получим связанный с блоком текст, который перемещается по рабочему полю вместе с блоком.

После того, как программа создана, ее можно запустить на исполнение с помощью кнопки *Эмуляция* (поз. 9 на рис. 4.9). При запуске эмуляции программа LOGO!Soft Comfort сначала проверяет коммутационную программу и отображает все имеющиеся в ней ошибки в *Окне информации*, расположенном в нижней части интерфейсного окна. Если окно информации отсутствует, то его можно вызвать при помощи команды меню *Вид > Окно информации* или нажатием функциональной клавиши [F4]. Запустить программу на симуляцию можно также с помощью функциональной клавиши [F2]. При нажатии этой клавиши программа LOGO!Soft Comfort выводит результат в строке состояния.

В режиме эмуляции в нижней части интерфейсного окна появляется *Панель инструментов моделирования*, которая позволяет наблюдать за поведением коммутационной программы. Цифровые входы отображаются в виде бегущих человечков и отмечаются латинской буквой I. Цифровые выходы обозначаются в виде лампочек и отмечаются буквой Q. Кроме цифровых входов в программе используются аналоговые входы, которые обозначаются буквами AI, и аналоговые выходы, которые обозначаются буквами AQ. Если в программе щелкнуть по аналоговому входу, то появляется горизонтальный ползунковый регулятор, с помощью которого можно установить значение входной аналоговой величины. Текущие значения входных цифровых и аналоговых величин можно установить с помощью команды *Сервис > Параметры эмуляции*. При этом появляется таблица, в которой указаны значения входных цифровых и аналоговых величин.

В процессе моделирования цифровые и аналоговые величины могут менять свои значения. При изменении статуса входных и выходных **цифровых** величин, бегущие человечки и лампочки могут получать свечение или гаснуть. За изменением **аналоговых** входных и выходных величин можно наблюдать в горизонтальных однострочных окнах, расположенных в *Панели моделирования*. Кроме функции наблюдения *Панель моделирования* может выполнять и функцию управления, т.е. с помощью этой панели можно включать и выключать цифровые входы и задавать значения входных аналоговых величин.

С помощью *Панели моделирования* можно эмулировать аварию питания нажатием на значок *Питание* и отключением питания по всем входам. Эта функция может использоваться для проверки реакции схемы на аварию питания и перезапуск и способность схемы к сохранению.

В коммутационной программе могут использоваться *Флаги* (маркерные блоки), которые выдают на своем выходе сигнал, приложенный на их входе в предыдущем цикле. Флаг M8 в пусковом периоде всегда имеет состояние «1», и после первого цикла происходит его сброс. После этого он может использоваться, как обычный маркер.

Пример 4.1. В главе 2 в примере 2.1 получена логическая схема, показанная на рис. 4.15.

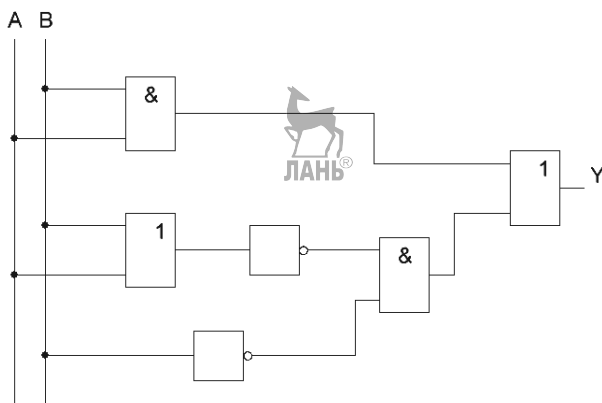


Рис. 4.15. Логическая схема

Этой схеме соответствует таблица истинности (табл. 4.3)

Таблица 4.3

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

В соответствии с этой таблицей значение Y равно «1», если обе переменных A и B одновременно равны «0» либо «1».

Создадим в LOGO!Soft Comfort коммутационную программу (рис. 4.16), соответствующую логической схеме (рис. 4.15).

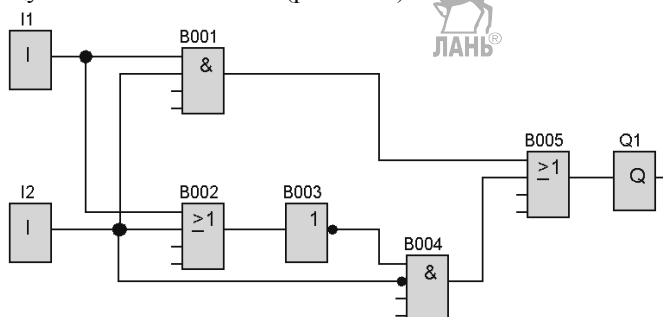


Рис. 4.16. Коммутационная программа

При создании коммутационной программы были использованы блоки:

I1, I2 – цифровые входы;

B001, B004 – базовые функции AND (И);

B002, B005 – базовые функции OR (ИЛИ);

B003 – базовая функция NOT (НЕ);

Q1 – цифровой выход.

Отрицание в программе LOGO!Soft Comfort можно задавать двумя способами: можно использовать блок NOT (блок B003), а можно поставить точку на входе последующего блока (точка на входе блока B004). Чтобы поставить точку, надо дважды щелкнуть левой кнопкой мыши по соответствующему контакту на входе блока.

По умолчанию на рабочем поле присутствует сетка. Чтобы убрать сетку, надо щелкнуть левой кнопкой мыши по вкладке *Формат*, выбрать в выпадающем окне опцию *Формат сетки* и далее в окне *Сетка* снять галочку около опции *Видимость*.

Если запустить коммутационную программу на рис. 4.16 на симуляцию, то можно убедиться, что программа работает в соответствии с таблицей истинности.

4.5.4. Таймеры

Задержка включения.

Перейдем к рассмотрению специальных функций. Специальные функции начинаются с функции *Задержка включения* и объединены общим заголовком *Таймеры*. Все эти функции, так или иначе, связаны с временем включения и отключения выходного сигнала.

Будем включать специальные функции в состав простейших блок-схем, с помощью которых легче понять алгоритм работы рассматриваемых функций. Начнем с функции *Задержка включения*. Блок-схема с функцией *Задержка*

включения (блок В001) представлена на рис. 4.17. Функция связана с помощью соединительных линий с блоком «I» (цифровой вход) и блоком «Q» (цифровой выход). Цветовая индикация позволяет определить состояние соединительной линии. По умолчанию, цвет соединительной линии с сигналом «0» – синий, с сигналом «1» – красный.

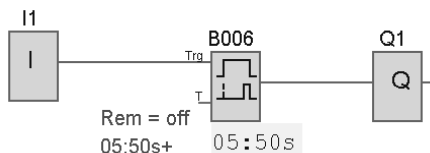


Рис. 4.17. Блок-схема с функцией *Задержка включения*

После подачи сигнала «1» на цифровой вход Trg, запускается отсчет времени, который можно наблюдать под блоком В001. По истечении 5,5 сек сигнал «1» появляется на выходе. Соединительная линия между блоком В001 и выходом Q1 меняет цвет с синего на красный. Время задержки (в данном случае 5,5 с) устанавливается в выпадающем окне (рис. 4.18), которое появляется после двойного щелчка по блоку.

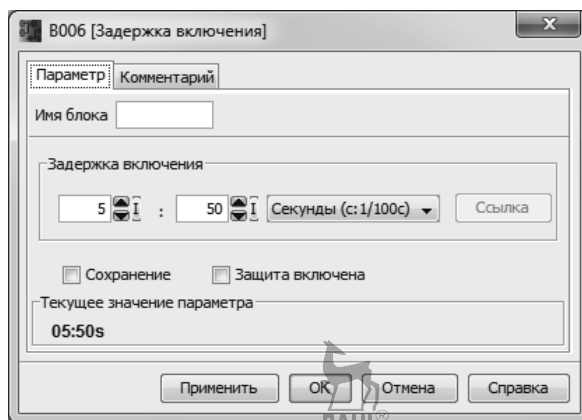
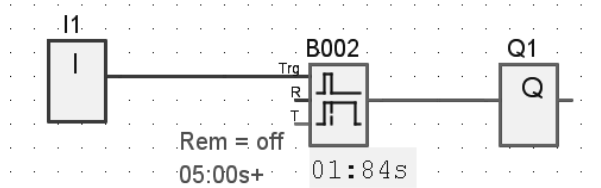


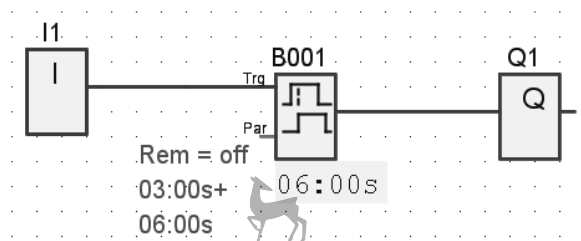
Рис. 4.18. Окно задания параметров блока *Задержка включения*

Если щелкнуть по треугольной стрелочке в центре окна (справа от слова секунды), то можно выбрать для измерения времени часы, минуты или секунды. В прямоугольных рамках слева от слова секунды задаются количество целых и сотых значений времени, например, 5 и 50, как показано на рис. 4.18. Таким образом, заданное время равно $5+50/100=5,5$ с. При подаче сигнала «0» на вход Trg, выход сбрасывается, т.е. на выходе появляется «0».

Задержка отключения.Рис. 4.19. Блок-схема с функцией *Задержка отключения*

Блок – схема с функцией *Задержка отключения* (блок B002) представлена на рис. 4.19. Функция связана с помощью соединительных линий с блоком «I1» (цифровой вход) и блоком «Q» (цифровой выход).

В начальный момент времени при подаче «1» на вход, сигнал «1» сразу проходит на выход, обе соединительные линии имеют красный цвет. При подаче «0» на вход, выход Q1 отключается не сразу. Начинается непрерывный отсчет времени, который отображается под блоком B002. По истечении 5 сек выход отключается, соединительная линия из красной становится синей. Время задержки, как и в предыдущем случае, устанавливается в выпадающем окне, которое появляется после двойного щелчка по блоку B002. В данном случае время задержки установлено 5 сек.

Задержка включения и отключения.Рис. 4.20. Блок-схема с функцией *Задержка включения и отключения*

Блок – схема с функцией *Задержка включения и отключения* представлена на рис. 4.20. При двойном щелчке по блоку появляется дополнительное окно (рис. 4.21), в котором устанавливается время включения (в данном случае 3 сек.) и время отключения (6 сек.).

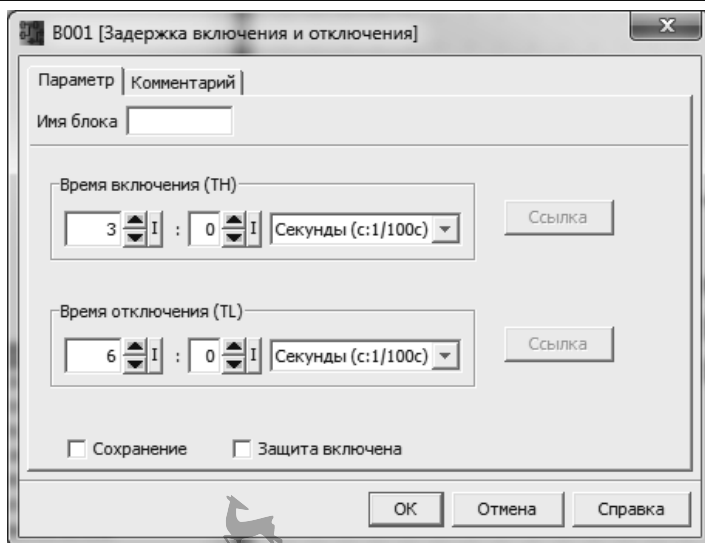


Рис. 4.21. Окно выбора параметров

После первичной подачи сигнала (подача сигнала «1») на цифровой вход Trg, начинается отсчет времени, и через 3 сек сигнал проходит на выход Q1. Выход Q1 включается. После вторичной подачи сигнала на вход Trg (подача сигнала «0») начинается новый отсчет времени от 0 до 6 сек., и через 6 сек. выход Q1 отключается.

Временная диаграмма работы блока представлена на рис. 4.22.

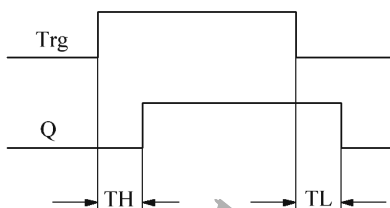
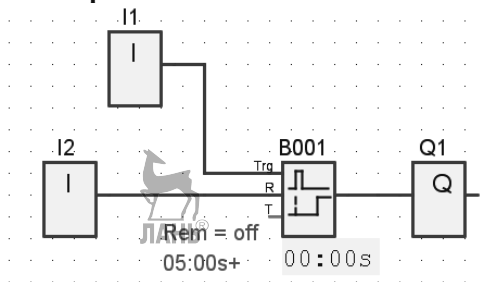
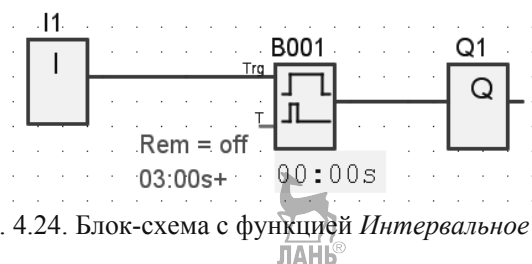


Рис. 4.22. Временная диаграмма

Положительный фронт сигнала на входе Trg запускает отсчет времени задержки включения (ТН) выхода Q. Отрицательный фронт сигнала на входе Trg запускает отсчет времени отключения (ТЛ) выхода Q.

Задержка включения с сохранением.Рис. 4.23. Блок-схема с функцией *Задержка включения с сохранением*

Блок – схема с функцией *Задержка включения с сохранением* представлена на рис. 4.23. Функция имеет два входа: Trg (Trigger) – запуск и R (Reset) – сброс. Функция обеспечивает задержку включения выхода Q1 на заданное время. На рис. 4.23 это время равно 5 сек. Данная функция отличается от функции *Задержка включения* тем, что при вторичной подаче сигнала (подаче сигнала «0») на вход Trg, сброса выхода Q1 в состояние «0» не происходит, выход остается в состоянии «1». Чтобы сбросить выход в состояние «0» необходимо подать сигнал «1» на вход R.

Интервальное реле (импульсный выход).Рис. 4.24. Блок-схема с функцией *Интервальное реле*

Блок – схема с функцией *Интервальное реле (импульсный выход)* представлена на рис. 4.24. Входной сигнал генерирует выходной сигнал заданной длительности, т.е. «1» на входе сразу дает единицу на выходе, которая сохраняется в течение установленного времени, затем выход сбрасывается на «0». Длительность выходного сигнала устанавливается в выпадающем окне, которое появляется после двойного щелчка по блоку. На рис. 4.24 длительность выходного сигнала задана равной 3 сек. Отсчет текущего времени после подачи запускающего импульса отображается под блоком B001. Как только это время достигнет трех секунд, выход отключается.

Интервальное реле с запуском по фронту.

Блок – схема с функцией *Интервальное реле с запуском по фронту* представлена на рис. 4.25. Если предыдущая функция *Интервальное реле* генерирует на выходе один импульс, то данная функция *Интервальное реле с запуском по фронту* генерирует на выходе несколько (серию) импульсов.

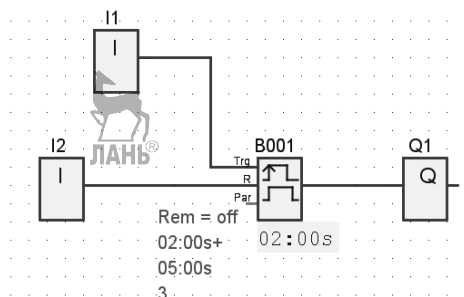


Рис. 4.25. Блок-схема с функцией *Интервальное реле с запуском по фронту*

Длительность импульса (ТН), длительность паузы (ТЛ) и число циклов импульсов устанавливается в выпадающем окне, показанном на рис. 4.26. Длительность импульса задана равной 2 сек, длительность паузы – 5 сек, число циклов импульсов – 3. Алгоритм работы функции следующий: при подаче «1» на вход Trg через 5 сек появляется сигнал на выходе, который сохраняется в течение 2 сек. По истечении 2 сек сигнал сбрасывается. Так повторяется 3 раза. Сигнал на входе R сбрасывает функцию в «0» не дожидаясь окончания циклов. Количество циклов удобно отслеживать по загоранию лампочки в *Панели моделирования* в нижней части интерфейсного окна.

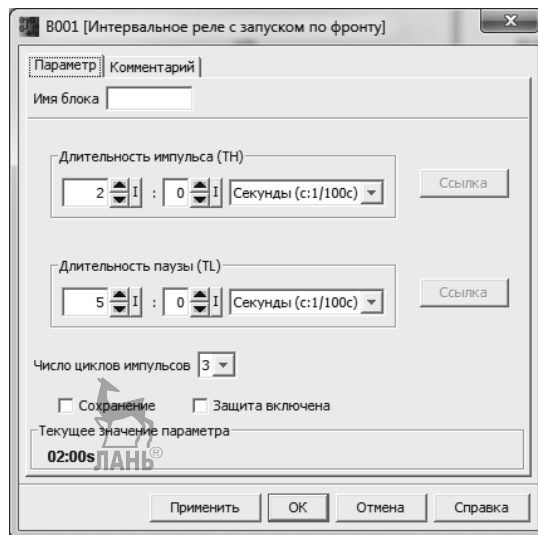
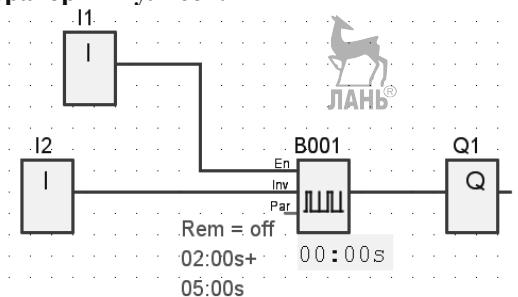


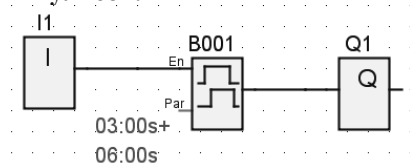
Рис. 4.26. Окно выбора параметров функции *Интервальное реле с запуском по фронту*

Асинхронный генератор импульсов.

Рис. 4.27. Блок-схема с функцией *Асинхронный генератор импульсов*

Блок – схема с функцией *Асинхронный генератор импульсов* представлена на рис. 4.27. Эта функция, как и предыдущая *Интервальное реле с запуском по фронту*, генерирует на выходе серию импульсов. Но в отличие от предыдущей функции, импульсы следуют друг за другом постоянно, их количество не ограничено каким-то заданным числом. На рис. 4.27 длительность импульса – 2 сек, длительность паузы – 5 сек. Алгоритм работы функции следующий: генератор запускается подачей «1» на вход En. На выходе сразу появляется «1». Длительность единичного импульса 2 сек. Далее следует пауза 5 сек и опять цикл повторяется. Если на вход Inv подать «1», то длительность и пауза меняются местами. Пауза становится равной 2 сек, а длительность импульса – 5 сек. При подаче «1» на вход En сначала выдерживается пауза 2 сек, а затем на выходе появляется сигнал длительностью 5 сек. Процесс останавливается при подаче «0» на вход En. За длительностью импульсов и пауз удобно следить по загоранию лампочки в *Панели моделирования* в нижней части интерфейсного окна.

Генератор случайных импульсов.

Рис. 4.28. Блок-схема с функцией *Генератор случайных импульсов*

Блок – схема с функцией *Генератор случайных импульсов* представлена на рис. 4.28. Функция генерирует импульсы случайной длительности со случайным временем задержки между импульсами. На рис. 4.28 длительность задержки включения может изменяться от 0 до 3 сек. Верхняя граница задержки включения обозначается буквами TH (в данном случае TH=3 сек). Длительность задержки отключения может изменяться от 0 до 6 сек. Верхняя граница задержки отключения обозначается буквами TL (в данном случае TL=6 сек). Время TH и TL устанавливается в выпадающем окне, которое появляется после двойного щелчка по блоку B001. Алгоритм работы функции следующий: При переходе из

«0» в «1» на входе En запускается случайная задержка включения выхода Q1. После включения выхода функция ждет следующей команды. При переходе из «1» в «0» на входе En запускается случайная задержка отключения выхода Q1. После отключения выхода функция ждет следующей команды. В дальнейшем циклы повторяются. В каждом цикле задержка включения и задержка отключения имеют свою случайную длительность. Алгоритм работы функции напоминает алгоритм работы функции *Задержка включения/отключения*, только в данном случае время задержки является случайной величиной.

Выключатель лестничного освещения.

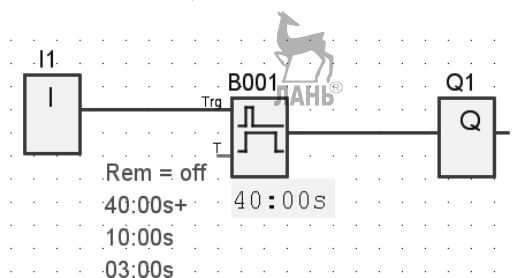


Рис. 4.29. Блок-схема с функцией *Выключатель лестничного освещения*

Блок – схема с функцией *Выключатель лестничного освещения* представлена на рис. 4.29. Предположим, вы щелкнули выключателем, который отключает на лестничной клетке освещение. Но вам еще нужно некоторое время, чтобы дойти до выходной двери и выйти на улицу. Таким образом, свет на лестничной клетке должен погаснуть не сразу, а через некоторое время. Функцию задержки отключения может выполнить *Выключатель лестничного освещения*. Но ранее мы уже рассмотрели аналогичную функцию *Задержка отключения*. Чем же отличается данная функция *Выключатель лестничного освещения* от функции *Задержка отключения*. В функции *Выключатель лестничного освещения* добавляется новое свойство: за некоторый промежуток времени до полного отключения света, он может на короткое время погаснуть (моргнуть), как бы предупреждая о скором полном отключении освещения. Время предупреждения (T!) и продолжительность предупреждения (T!) задаются в выпадающем окне, показанном на рис. 4.30. В данном случае время задержки отключения составляет 40 сек, время предупреждения – 10 сек, продолжительность предупреждения – 3 сек. Алгоритм работы функции следующий: при переходе сигнала на входе Trg из «1» в «0» (подача сигнала «0» на вход Trg, на котором до этого была установлена «1») включается задержка отключения (в нашем случае 40 сек). За 10 сек до полного отключения освещения, т.е. через 30 сек после того, как мы щелкнули выключателем, свет на 3 сек погаснет, затем включится и будет продолжать гореть еще 7 сек, пока полное время задержки не составит 40 сек.

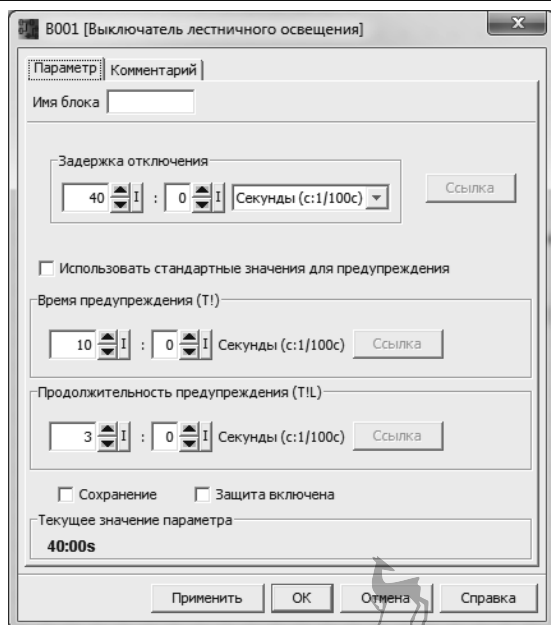


Рис. 4.30. Окно для задания параметров функции *Выключатель лестничного освещения*

Многофункциональный выключатель.

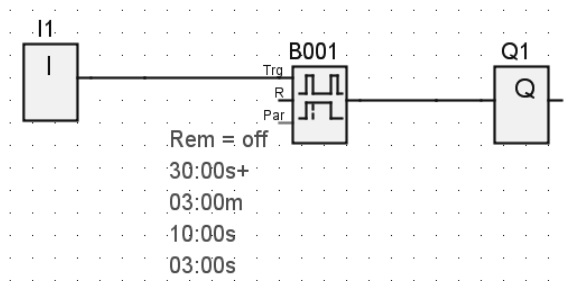


Рис. 4.31. Блок-схема с функцией *Многофункциональный выключатель*

Блок – схема с функцией *Многофункциональный выключатель* представлена на рис. 4.31. Функция работает как обычный выключатель при включении и как выключатель с замедлением при выключении. Временные интервалы, указанные около блока (функции), означают следующее:

- 30 сек – время задержки отключения (T);
- 3 мин – постоянное освещение (TL);
- 10 сек – время предупреждения (T!);
- 3 сек – продолжительность предупреждения (T!L).

Временные интервалы устанавливаются в выпадающем окне после двойного щелчка по блоку.

Алгоритм работы функции следующий. При переходе сигнала на входе Trg из «0» в «1» включается освещение. Если в течение 3 мин освещение не выключить, то оно будет гореть постоянно. Но если до истечения 3 мин. освещение выключить (подать «0» на вход Trg), то свет погаснет с задержкой 30 сек. При этом за 10 сек до полного отключения будет выдан предупредительный сигнал длительностью 3 сек.

Семидневный таймер.

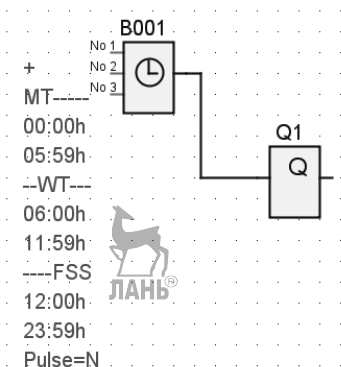
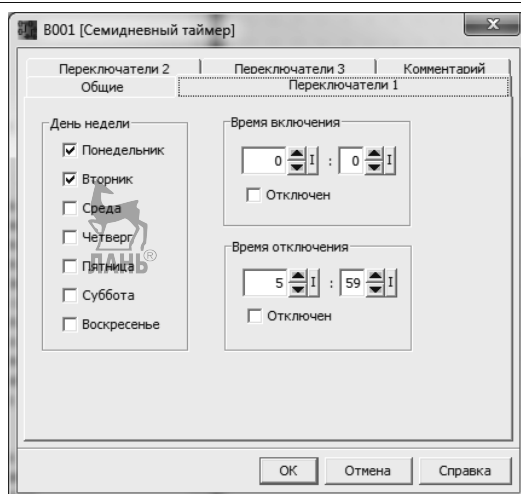
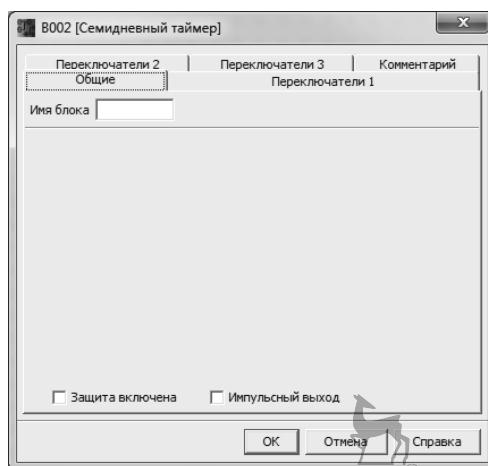


Рис. 4.32. Блок-схема с функцией Семидневный таймер

Блок – схема с функцией Семидневный таймер представлена на рис. 4.32. Функцию можно использовать при условии, что контроллер LOGO! оборудован часами реального времени. Напомним, что такие контроллеры имеют в своем обозначении букву С, например, LOGO! 12/24RC. С помощью этой функции можно задать время включения и отключения выхода Q1 для каждого дня недели. Для задания времени могут использоваться три переключателя. На рис. 4.32 первый переключатель установлен на понедельник, вторник (Monday, Tuesday), время включения 00:00 часов, время выключения 05:59 часов. Второй выключатель установлен на среду, четверг (Wednesday, Thursday), время включения 06:00 часов, время отключения: 11:59 часов. Третий выключатель установлен на пятницу, субботу, воскресенье (Friday, Saturday, Sunday), время включения 12:00 часов, время отключения: 23:59 часов. На рис. 4.33 в качестве примера показана установка времени включения и отключения на переключателе_1 для дней недели: понедельник, вторник. Если время включения и отключения переключателей перекрывается, то переключатель_3 имеет приоритет над переключателем_2, а переключатель_2 имеет приоритет над переключателем_1.

Рис. 4.33. Окно для задания параметров функции *Семидневный таймер*

В нижней части вкладки *Общие* на рис. 4.34 есть опция *Импульсный выход*. Если в квадратике около этой опции галочка не проставлена, то опция не активна и около слова Pulse на рис. 4.32 стоит буква N (NO). Если около опции *Импульсный выход* поставить галочку, то около слова Pulse появляется буква Y (Yes), а опция *Время отключения* во вкладках *Переключатели* становится не активной. Это означает что в этом случае в момент достижения времени включения, заданного переключателем, на выход Q1 подается «1», после чего выход сразу сбрасывается в «0», т.е. данная опция задействована только в течение одного цикла. Опция *Время отключения* в данном случае не работает. Установленный параметр импульса распространяется на все три переключателя.

Рис. 4.34. Вкладка *Общие* функции *Семидневный таймер*

Проконтролировать работу функции *Семидневный таймер* можно с помощью инструмента «Установить текущую дату и время для эмуляции (часы)» в *Панели моделирования* (поз 13).

Годовой таймер.

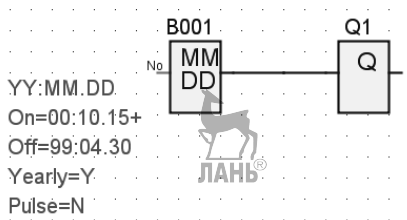


Рис. 4.35. Блок-схема с функцией *Годовой таймер*

Блок – схема с функцией *Годовой таймер* представлена на рис. 4.35. Выходной сигнал управляется настраиваемой датой включения и отключения. Можно настроить включение таймера в ежегодном, ежемесячном или пользовательском режиме. В любом режиме можно также настроить подачу импульсов на выход таймера в течение определенного периода времени. Период времени может быть настроен в диапазоне дат от 1 января 2000 года до 31 декабря 2099 года, как показано на рис. 4.36.

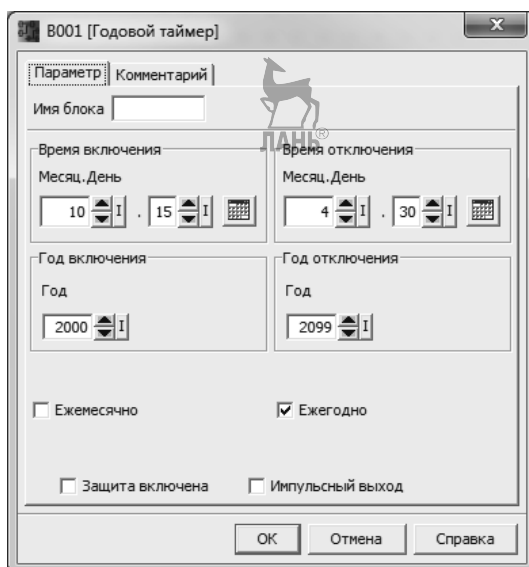


Рис. 4.36. Окно для настройки параметров функции *Годовой таймер*

Время включения служит для задания месяца и дня установки таймера. Время выключения служит для задания месяца и дня, когда выход таймера сбрасывается. Обратите внимание порядок следования полей времени включения

и выключения: первое поле определяет год, второе месяц, а третье день. Если отметить опцию *Ежемесячно*, то выход таймера будет включаться каждый месяц в заданный день и заданное время, и будет оставаться включенным до наступления времени выключения в заданный день. Год включения служит для задания исходного года, в который таймер будет задействоваться. Год выключения служит для задания последнего года, в который таймер будет выключаться. Максимальное значение года равно 2099. В случае, если вы отметите блок выбора *Ежегодно*, выход таймера будет включаться каждый год в заданное время включения в указанные месяц и день, и он будет оставаться включенным до наступления заданного времени выключения в указанные месяц года и день.

Если установлен флажок *Импульсный выход*, то выход таймера включается при наступлении времени включения на один цикл, после чего выход таймера сбрасывается. Можно использовать импульсы таймера в ежемесячном или ежегодном режиме, а также однократно в определенное время. Если ни один из флажков (ежемесячный, ежегодный или импульсный) не установлен, можно указать специальный период времени при помощи времени включения и времени отключения. Этот период может охватывать любое требуемое время.

Для организации процесса с многочисленными включениями и выключениями на различные интервалы времени в течение года можно задать множество годовых таймеров, выходы которых должны быть объединены функциональным блоком OR.

4.5.5. Счетчики

Реверсивный счетчик.

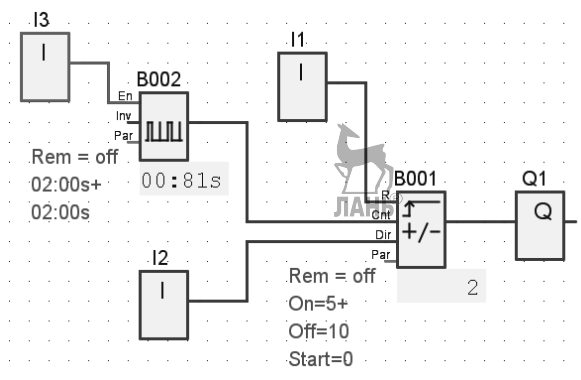


Рис. 4.37. Блок-схема с функцией *Реверсивный счетчик*

Группа следующих специальных функций: *Реверсивный счетчик*, *Счетчик рабочего времени*, *Пороговый выключатель*, объединяется общим названием *Счетчики*. Эта группа функций позволяет управлять выходом, опираясь на количество и длительность выходных импульсов.

Блок – схема с функцией *Реверсивный счетчик* представлена на рис. 4.37, где блок V001 – реверсивный счетчик, блок V002 – асинхронный генератор. Функция *Реверсивный счетчик* подсчитывает число импульсов на входе Cnt, т.е. число переходов из «0» в «1» (число переходов из «1» в «0» не подсчитывается). Вход R – сброс счетчика на «0». Вход Dir (Direction – направление) определяет направление счета: Dir=0: вверх, Dir=1: вниз. Как только число импульсов станет равным порогу включения, на выход подается «1». При этом реверсивный счетчик продолжает считать импульсы и как только их число станет равным порогу отключения, выход отключается. Если дважды щелкнуть по блоку V001, то появится окно для задания параметров реверсивного счетчика, показанное на рис. 4.38.

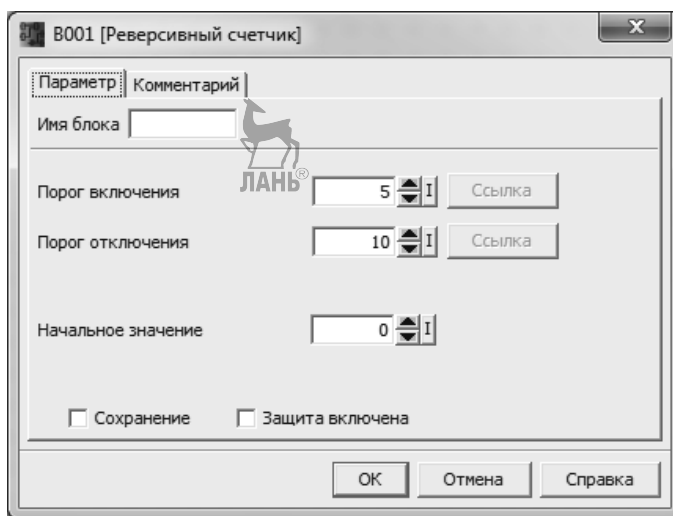


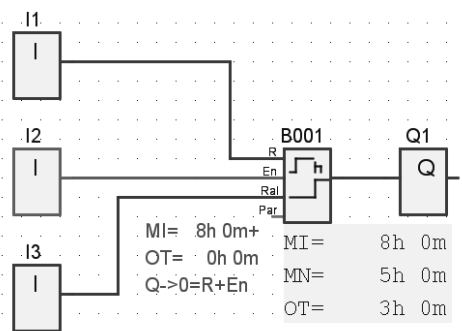
Рис. 4.38. Окно для задания параметров функции *Реверсивный счетчик*

Блок-схема, представленная на рис. 4.37, работает следующим образом. Асинхронный генератор V002 генерирует импульсы длительностью 2 с и паузой между импульсами также 2 с. Отсчет импульсов реверсивным счетчиком начинается с нуля. Когда количество импульсов достигнет 5, включается выход Q1. Когда число импульсов достигнет 10, выход отключается. Подача «1» на вход R обнуляет реверсивный счетчик, и процесс можно повторить сначала.

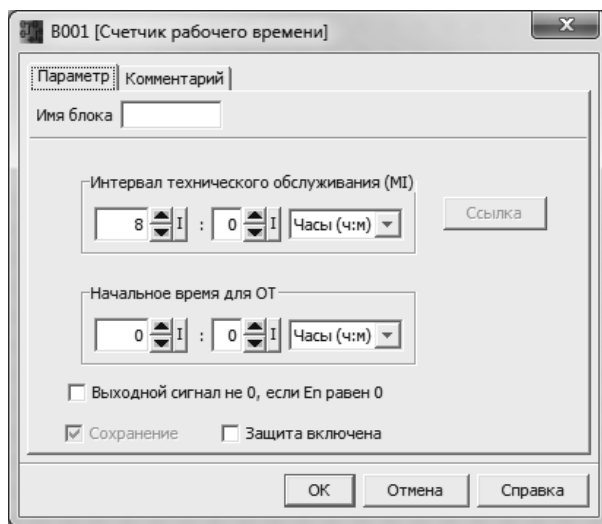
Пока $R = 1$, на выходе Q1 устанавливается 0, и подсчет импульсов на входе Cnt не выполняется.

Счетчик рабочего времени.

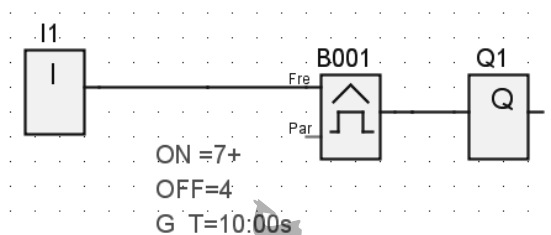
Блок – схема с функцией *Счетчик рабочего времени* представлена на рис. 4.39. Под рабочим временем в данном случае понимается длительность рабочего цикла какого-либо оборудования.

Рис. 4.39. Блок-схема с функцией *Счетчик рабочего времени*

Счетчик рабочего времени (блок B001) контролирует вход En. В момент подачи «1» на вход En начинается отсчет времени. Когда отсчитываемое время достигнет заданной величины, обозначаемой MI, «1» проходит на выход Q1. На рис. 4.39 время MI составляет 8 час. Буквами OT обозначается время, которое прошло с момента подачи «1» на вход En. В процессе работы функции также определяется остаточное время MN = MI – OT. Все перечисленные временные промежутки можно наблюдать под блоком B001. Временной интервал MI и начальное время OT задаются в дополнительном окне, показанном на рис. 4.40.

Рис. 4.40. Окно для задания параметров функции *Счетчик рабочего времени*

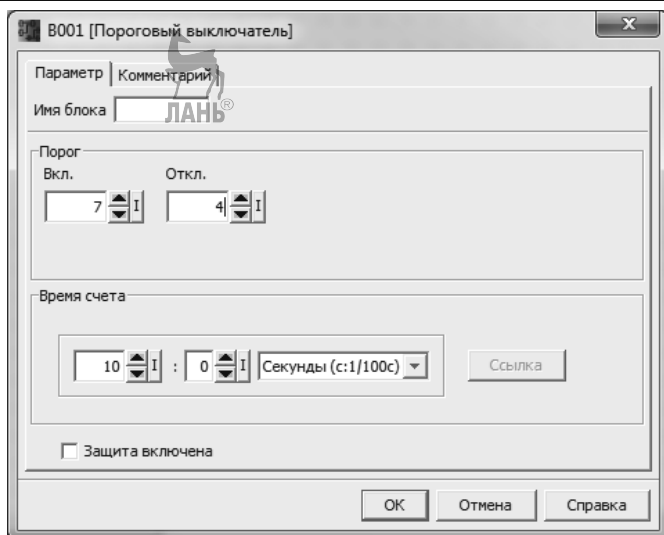
Входом R можно сбросить выход Q на «0» и перевести счетчик остаточного времени на заданное значение MI. При этом внутренний счетчик OT продолжает считать дальше. Входом Ral также можно сбросить выход Q на «0» и перевести счетчик остаточного времени на заданное значение MI. При этом счетчик OT также сбрасывается на «0».

Пороговый выключатель.Рис. 4.41. Блок-схема с функцией *Пороговый выключатель*

Блок – схема с функцией *Пороговый выключатель* представлена на рис. 4.41. Для этой функции задаются три параметра: порог включения (ON), порог выключения (OFF), время счета (G_T). Функция работает следующим образом. В процессе работы блок-схемы постоянно отсчитываются промежутки времени, длительностью 10 сек. Если на входе Fre число переходов из «0» в «1» за этот промежуток времени будет > 7 , то включается выход Q1 (выход переходит в состояние «1»). Далее выход будет оставаться в состоянии «1» до тех пор, пока число переходов на входе Fre не станет равным 4 или меньше. И так, значение ON включает выход, значение OFF отключает выход. После того как выход включился, он будет оставаться включенным до тех пор, пока на входе не будет достигнуто число OFF. Например, для схемы на рис. 4.41 выход будет оставаться включенным, если число переходов из «0» в «1» на входе будет равно 5 и больше. Описанный алгоритм работы продолжает действовать и в том случае, когда значение ON меньше или равно OFF. При необходимости, в соответствии с известными формулами, от промежутков времени можно перейти к частотам.

Выпадающее окно, в котором устанавливаются значения параметров ON, OFF, G_T показано на рис. 4.42.



Рис. 4.42. Окно для задания параметров функции *Пороговый выключатель*

4.5.6. Аналоговые функции

Аналоговые вычисления.

Следующие несколько специальных функций объединяются общим заголовком **«Аналоговые»**. Это означает, что аналоговый выход этих функций может принимать любое числовое значение, а не только «0» и «1».

Функция *Аналоговые вычисления* рассчитывает значение выхода AQ по формуле, образованной операндами и операторами. Операндами в данной функции называются числа V1, V2, V3, V4, которые могут принимать значения от -32768 до 32767. Операторами называются четыре арифметических действия: +, -, *, /. Для каждого оператора (арифметического действия) должен быть установлен неповторяющийся уровень приоритета: высокий («H» - High), средний («M» - Middle), низкий («L» - Low). Сначала будет выполнена операция с высоким приоритетом, затем – со средним, затем – с низким. Задавать можно только одну операцию каждого приоритета. В качестве значений операндов могут использоваться другие уже запрограммированные функции.

При записи формул необходимо использовать все четыре операнда и три оператора. Если требуется использовать меньшее число операндов, то надо используйте конструкции вида « + 0 » или « * 1 », чтобы заполнить вакантные места.

Блок-схема с функцией *Аналоговые вычисления* показана на рис. 4.43.

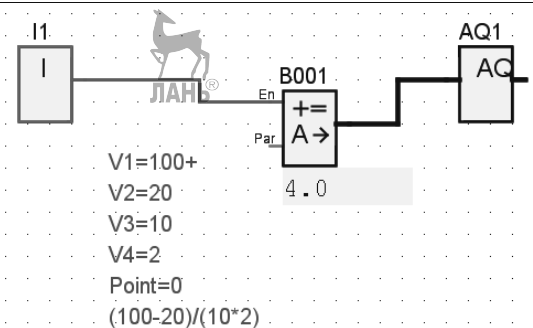


Рис. 4.43. Блок-схема с функцией *Аналоговые вычисления*

Пример 4.2. Предположим, надо вычислить значение выражения $(100 - 20)/(10 * 2) = ?$

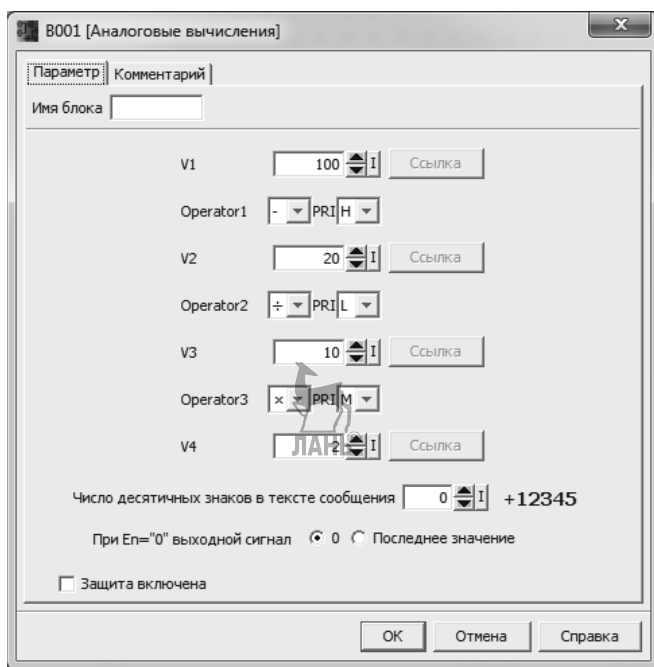


Рис. 4.44. Окно для задания параметров вычислений

В этом выражении используются четыре операнда: 100, 20, 10, 2, и три оператора: «вычитание», «умножение», «деление». Первым действием выполняется вычитание $(100-20)$, поэтому мы задаем оператору «вычитание» высокий приоритет – Н. Вторым действием выполняется умножение $(10 \cdot 2)$, поэтому мы задаем оператору «умножение» средний приоритет – М. Третьим действием выполняется деление $(80/20)$, поэтому мы задаем оператору «деление» низкий

приоритет – L. В выпадающем окне, которое появляется после щелчка по блоку B001, нужно задать операнды, операторы и приоритеты, как показано на рис. 4.44. В результате выполнения функции мы получаем число 4, которое появляется под функцией *Аналоговые вычисления*, как показано на рис. 4.43.

Вход En можно настроить таким образом, что при подаче «0» на этот вход, на выходе будет «0» (эта настройка показана на рис. 4.44), а можно настроить так, что при подаче «0» на этот вход, выходной сигнал будет сохранять последнее значение.

При делении на 0 или переполнении на выходе AQ будет установлено максимальное значение 32767, а при отрицательном переполнении будет установлено минимальное значение -32768.

Параметр Point – p (число разрядов после десятичной точки) относится к отображению значений V1, V2, V3, V4 и AQ в текстах сообщений.

Пример 4.3. Вычислим выражение: $10/3=?$

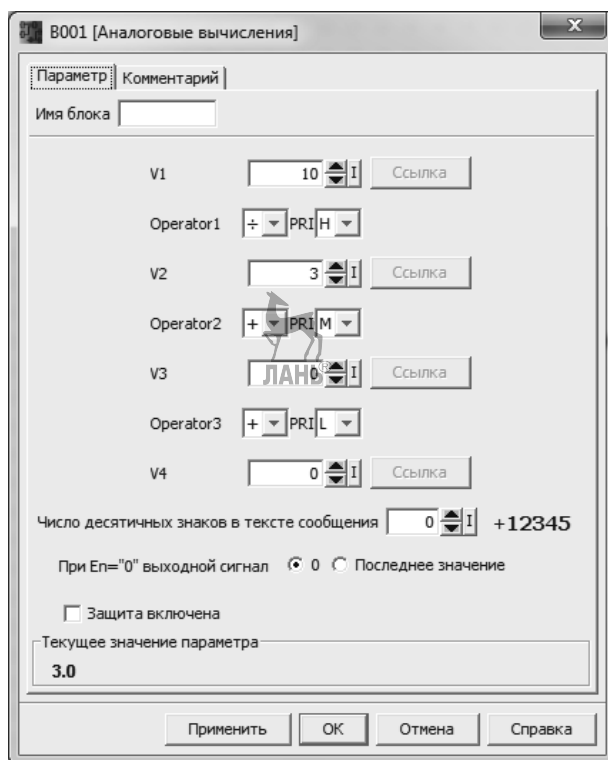


Рис. 4.45. Окно для задания параметров вычислений

В этом примере всего два операнда и один оператор. Недостающие два операнда дополняем нулевыми значениями и тогда параметры в выпадающем окне запишутся, как показано на рис. 4.45. В результате вычислений получаем 3.0, т.е. результат округлился до целого числа.

Аналоговый компаратор.

Аналоговый компаратор оснащен двумя входами, на каждый из которых подается входной сигнал. Компаратор определяет разность между сигналами. Разность является положительной, если $A_x > A_y$, и отрицательной, если $A_x < A_y$. Если разность превышает предварительно заданное значение, то на выходе устанавливается «1». При превышении порога выключения, на выходе устанавливается «0». Временная диаграмма, поясняющая работу, показана на рис. 4.46.

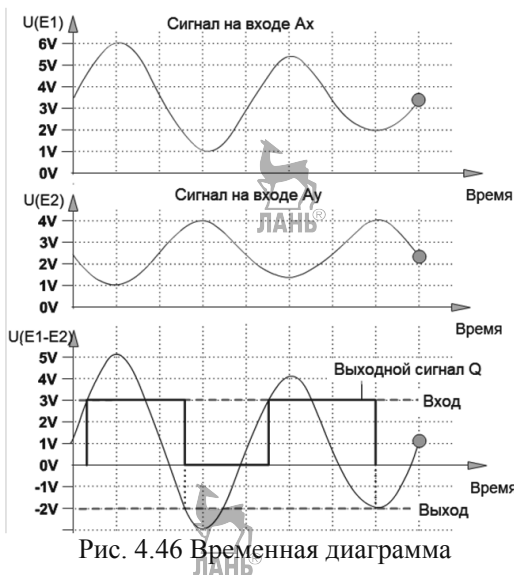


Рис. 4.46 Временная диаграмма

Блок-схема с функцией *Аналоговый компаратор* представлена на рис. 4.47. Выход этой функции устанавливается и сбрасывается в зависимости от разности двух конфигурируемых пороговых значений: порога включения (On) и порога отключения (Off).

Параметры функции задаются в выпадающем окне, представленном на рис. 4.48, где установлены следующие значения:

- порог включения (On) – 40 (соответствует температуре 40°);
- порог отключения (Off) – 30 (соответствует температуре 30°);
- усиление (Gain) – 0,15;
- смещение (Offset) – (-50).

Эти параметры отображаются около блока *Аналоговый компаратор* (B001).

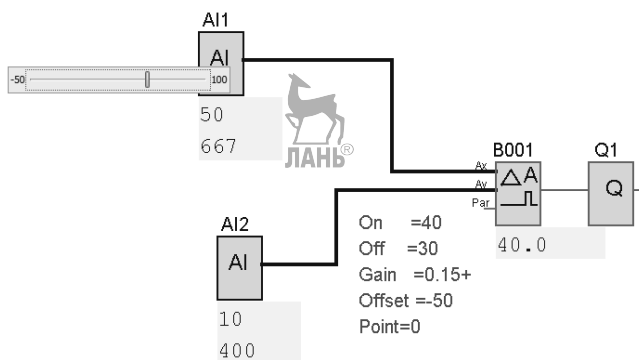


Рис. 4.47. Блок-схема с функцией *Аналоговый компаратор*

Заданным параметрам усиления и смещения соответствует диапазон изменения аналоговой величины от (-50) до (+100), что соответствует исходному температурному диапазону. На блок-схеме рис. 4.47 значения аналоговых и нормализованных величин отображаются под аналоговыми входами AI1 и AI2. Верхняя цифра соответствует аналоговой величине, нижняя цифра соответствует нормализованной величине.

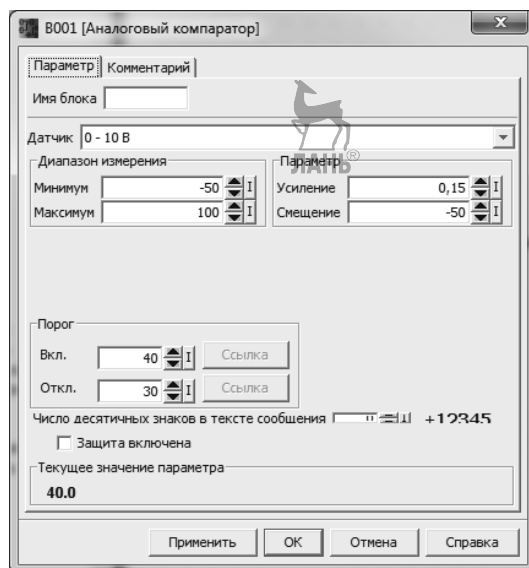


Рис. 4.48. Окно для задания параметров функции *Аналоговый компаратор*

Блок-схема на рис. 4.47 работает следующим образом. Значение аналоговой величины на входе AI2 равно 10. Если с помощью движка увеличивать значение аналоговой величины на входе AI1, то при достижении значений ≥ 50 разность двух аналоговых значений становится $A_x - A_y \geq 40$. При этом сигнал проходит

на выход, на выходе появляется «1». Если уменьшать значение аналогового сигнала на входе AI1, то при достижении значений $(A_x - A_y) \leq 30$ выход Q1 отключится, на выходе «0». Разность значений аналоговых величин на входах (AI1 – AI2) отображается под блоком B001 – Аналоговый компаратор.

Аналоговый пороговый выключатель.

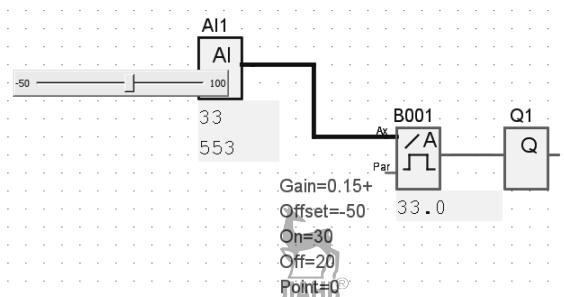


Рис. 4.49. Блок-схема с функцией Аналоговый пороговый выключатель

Пороговый выключатель также называется триггером Шмитта. Блок – схема с функцией Аналоговый пороговый выключатель представлена на рис. 4.49. Блок-схема работает следующим образом. Когда значение аналоговой величины превысит порог включения (On=30) на выходе появляется «1», которая сохраняется до тех пор, пока значение аналоговой величины не станет меньше порога выключения (Off=20). Значения порогов включения и отключения, а также параметров Gain (усиление) и Offset (смещение) устанавливаются в выпадающем окне, появляющемся после щелчка левой кнопкой мыши по блоку Аналоговый пороговый выключатель. Временная диаграмма работы порогового выключателя показана на рис. 4.50.



Рис. 4.50. Временная диаграмма

Аналоговый усилитель.

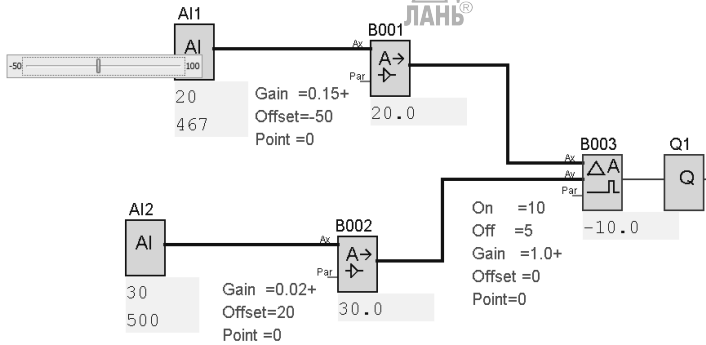


Рис. 4.51. Блок-схема с функцией *Аналоговый усилитель*

Функция *Аналоговый усилитель* предназначена для преобразования аналогового сигнала (измеренного значения физической величины) к нормализованному значению, лежащему в диапазоне 0...1000, и передачи этого значения на аналоговый выход. Блок-схема с двумя блоками *Аналоговый усилитель* представлена на рис. 4.51. Аналоговыми усилителями на схеме являются блоки B001 и B002.

Блок B001 за счет коэффициентов усиления и смещения преобразует диапазон изменения физической величины (-50)...(+100) в нормализованный диапазон 0...1000, как показано на рис. 4.52.

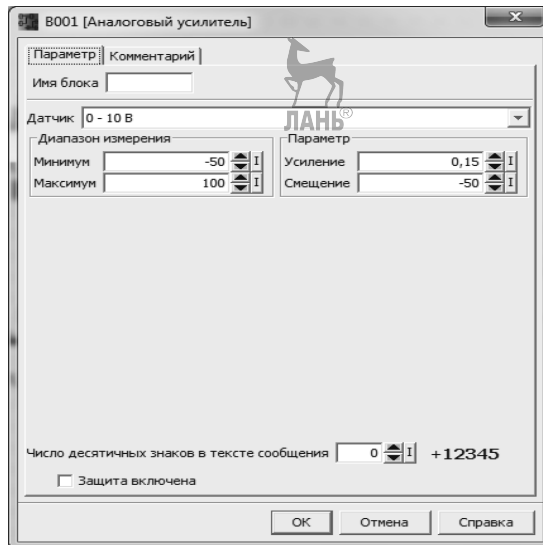


Рис. 4.52. Окно для задания параметров блока B001

Блок В002 преобразует диапазон измерения физической величины 20...40 в нормализованный диапазон 0...1000, как показано на рис. 4.53.

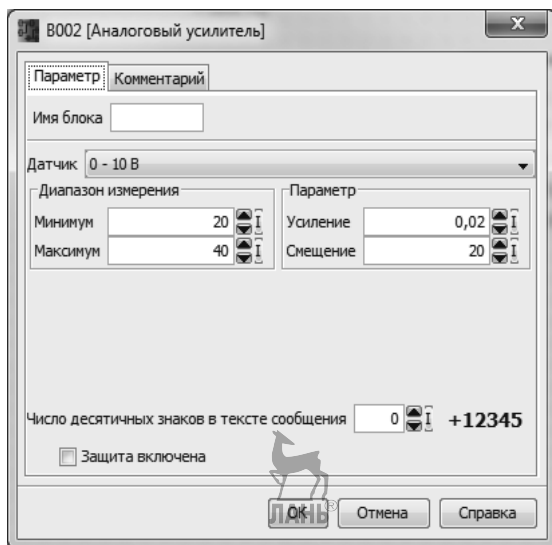


Рис. 4.53. Окно для задания параметров блока В002

После аналоговых усилителей В001 и В002 сигналы подаются на входы аналогового компаратора В003.

Блок-схема работает следующим образом. На аналоговом входе А12 значение измеренной аналоговой величины равно 30. Это значение передается на вход А_у аналогового компаратора. С помощью ползунка будем менять значение входной аналоговой величины на входе А11 и передавать это значение на вход А_х аналогового компаратора. Когда разность значений (А_х – А_у) достигнет порога включения (Оп=10), сигнал пройдет на выход аналогового компаратора, и выход примет значение «1». Значение «1» будет сохраняться на выходе до тех пор, пока при уменьшении аналоговой величины А_х разность сигналов (А_х – А_у) не достигнет значения, равного порогу отключения (Off=5).

Контроль аналоговых значений.

Блок – схема с функцией *Контроль аналоговых вычислений* представлена на рис. 4.54. Эта специальная функция сохраняет текущее значение аналогового сигнала на входе в памяти и переводит выход в состояние «1», если отклонение выходной переменной от сохраненного значения превышает заданную величину.

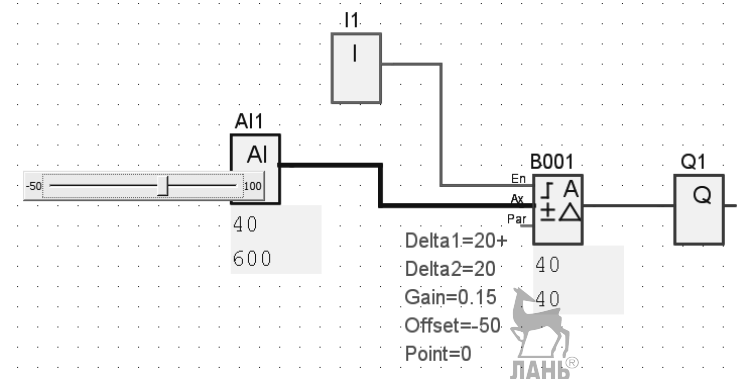


Рис. 4.54. Блок-схема с функцией *Контроль аналоговых вычислений*

В блок-схеме на рис. 4.54 для функции *Контроль аналоговых вычислений* установлены следующие параметры:

- порог 1(Delta 1) – 20;
- порог 2 (Delta 2) – 20;
- усиление (Gain) – 0,15;
- смещение (Offset) – (-50).

Блок-схема работает следующим образом. При переходе из 0 в 1 на входе En сохраняется значение сигнала, установленное на аналоговом входе Ax. Это сохраненное текущее значение обозначается «Aen» (на схеме Aen=40). Выход Q устанавливается в «1», когда сигнал на входе En = 1, и если фактическое значение на входе Ax выходит за пределы Aen + Порог 1 / Aen - Порог 2 (на схеме $40 + 21/40 - 20$). Поскольку Delta 1 задано, как (20+), то чтобы на выходе появилась «1», необходимо к исходной аналоговой величине 40 прибавить не 20, а 21. Выход Q сбрасывается в «0», когда фактическое значение на входе Ax выходит за пределы Aen + Порог 1 / Aen - Порог 2, или когда сигнал на входе En изменяется на «0».

Аналоговый дифференциальный выключатель.

Блок – схема с функцией *Аналоговый дифференциальный выключатель* представлена на рис. 4.55. Выход устанавливается и сбрасывается в зависимости от настраиваемого порога включения (On) и значения разности (Delta).

В блок-схеме на рис. 4.55 для функции *Аналоговый дифференциальный выключатель* установлены следующие параметры:

- усиление (Gain) – 0,15;
- смещение (Offset) – (-50);
- порог включения (On1) – 20;
- разность (Delta) – 5.

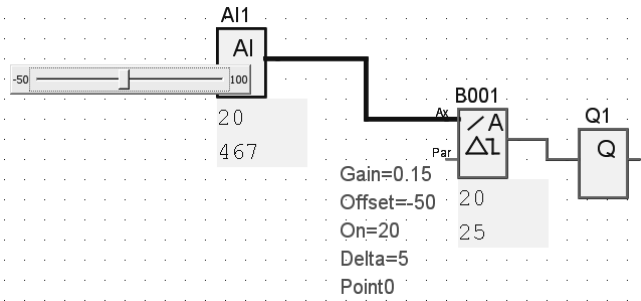


Рис. 4.55. Блок-схема с функцией *Аналоговый дифференциальный выключатель*

Выход Q устанавливается в «1» или сбрасывается в «0» в зависимости от заданного порогового значения (On) и значений разности (Delta). Функция автоматически вычисляет параметр Off. $Off = On + Delta$, где Delta может быть положительной или отрицательной величиной. Внутри диапазона $On \pm Delta$ на выходе имеем состояние «1», за пределами диапазона $On \pm Delta$ на выходе состояние «0».

Давайте сравним предыдущую функцию *Контроль аналоговых вычислений* с текущей функцией *Аналоговый дифференциальный выключатель*. В предыдущем случае в пределах диапазона $A_{en} + Delta$ / $A_{en} - Delta$ 2 на выходе сохранялось состояние «0», и за пределами этого диапазона состояние выхода переключалось на «1». В данном случае в пределах диапазона $On \pm Delta$ на выходе имеем состояние «1» и за пределами этого диапазона выход переключается в состояние «0», т.е. предыдущая и настоящая функции реализуют противоположные ситуации.

Аналоговый мультиплексор.

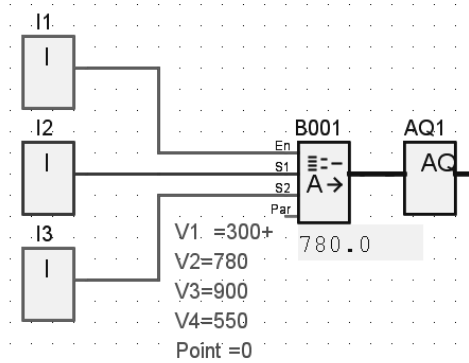


Рис. 4.56. Блок-схема с функцией *Аналоговый мультиплексор*

Блок – схема с функцией *Аналоговый мультиплексор* представлена на рис. 4.56. В случае установки входа En в состояние «1», функция выдает одно из 4 возможных аналоговых значений V1... V4 на выход AQ в зависимости от

значений параметров S1 и S2. Если вход En не установлен в состояние «1», функция выводит аналоговое значение «0» на выходе AQ. Возможные варианты установок входов I1...I3 и значений выхода AQ1 приведены в таблице 4.4.

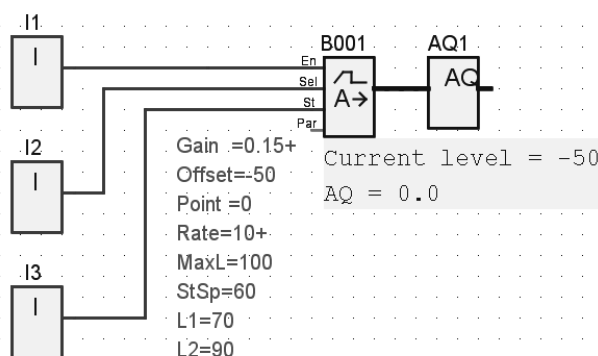
Таблица 4.4.

Вариант	I1 (En)	I2 (S1)	I3 (S3)	Параметр V (Выход AQ1)
1	0	любое	любое	0
2	1	0	0	V1
3	1	0	1	V2
4	1	1	0	V3
5	1	1	1	V4

На блок-схеме рис. 4.56 реализован вариант 3. Диапазон аналоговых значений V1...V4 может принимать значения: -32768...+32767. Соответственно диапазон значений AQ также: -32768...+32767.

Надо иметь в виду, что контроллер может работать с числами от 0 до 1000. Поэтому, следует подключить дополнительный аналоговый усилитель после аналогового выхода мультиплексора, чтобы добиться стандартизации выходного аналогового сигнала мультиплексора в пределах значений 0...1000 при значениях физической аналоговой величины больше 1000.

Линейно нарастающий аналоговый сигнал.

Рис. 4.57. Блок-схема с функцией *Линейно нарастающий аналоговый сигнал*

Блок – схема с функцией *Линейно нарастающий аналоговый сигнал* представлена на рис. 4.57. Функция позволяет изменять выходное значение аналогового сигнала от начального уровня пуска/останова до конечного уровня Level 1 или Level 2 с заданной скоростью (Rate). Термин «пуск/останов» означает,

что сигнал начинает линейно возрастать и затем линейно уменьшаться до одного и того же уровня, определяемого, как сумма величин B (Offset) + $StSp$ (Start/Stop).

Назначение входов функции:

Вход En: при переключении входа с «0» на «1» через время задержки 100 мс запускается линейное возрастание сигнала от начального уровня до конечного уровня. Начальный уровень задается, как B (Смещение) + $StSp$. Смещение и значение $StSp$ устанавливаются в выпадающем окне, показанном на рис. 4.58. В этом окне параметр $StSp$ обозначен, как *Смещение пуска/останова* и задается равным 60, *Смещение* равно (-50). Обратное переключение входа En с «1» на «0» сбрасывает функцию к исходному состоянию, т.е. устанавливает текущее значение аналоговой величины, равное *Смещению*, а на выходе AQ устанавливается «0», как показано на рис. 4.57 (Current level = -50, AQ = 0.0). Термин Current level означает текущее значение измеренной аналоговой величины, которое изменяется от -50 до 100. Нормализованное значение аналоговой величины изменяется от 0 до 1000.

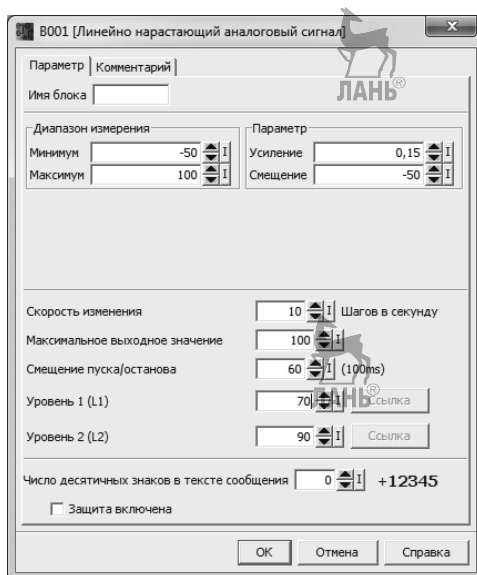


Рис. 4.58. Окно выбора параметров

Вход Sel (Select). При выборе $SeI = \langle 0 \rangle$ сигнал линейно возрастает до уровня Level 1. При выборе $SeI = \langle 1 \rangle$ сигнал возрастает до уровня Level 2.

Вход St. Изменение состояния с «0» на «1» на входе St вызывает уменьшение аналоговой величины до уровня пуска/останова, т.е. до уровня *Смещение* + $StSp$. Уровень пуска / останова держится в течение 100 мс, после чего скачком уменьшается до значения, равного *Смещению*, а на выходе AQ устанавливается «0».

Назначение параметров, задаваемых в окне на рис. 4.58:

Скорость изменения (Rate) – скорость достижения значений параметров Level 1 и Level 2. Задается числом шагов в секунду.

Максимальное выходное значение (MaxL) – максимальное значение сигнала, которое не должно превышаться.

Смещение пуска/останова (StSp) – величина, добавляемая к *Смещению* для задания уровня пуска/останова.

Уровень 1 (L1) и *Уровень 2 (L2)* – уровни, которые должны быть достигнуты при линейном возрастании сигнала.

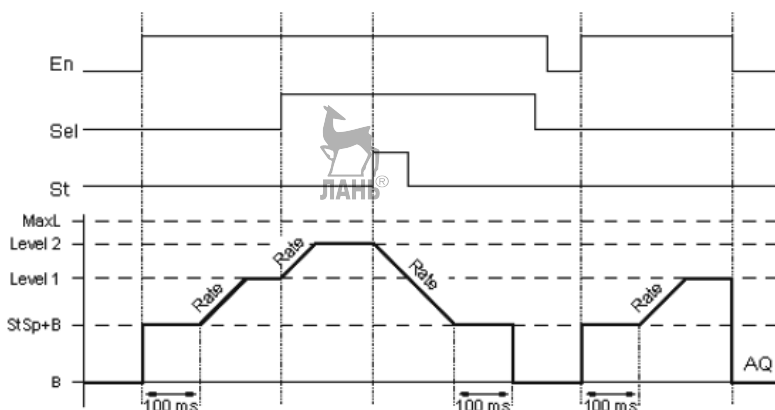


Рис. 4.59. Циклограмма работы функции *Линейно нарастающий аналоговый сигнал*

Работа функции поясняется циклограммой, представленной на рис. 4.59. Через 100 мс после подачи «1» на вход En функция скачком достигает начального значения ($B + StSp$). В нашем случае ($-50+60=10$). От значения 10 начинается линейное возрастание функции. При этом, если на входе Sel установлен «0», то возрастание будет до уровня Level 1 ($L1=70$), если установлена «1», то возрастание будет до уровня Level 2 ($L2=90$). Отметим, что «1» на вход Sel можно установить не сразу, а после достижения уровня Level 1, тогда получим ступенчатое возрастание функции, как показано на циклограмме. Далее, подавая «1» на вход St, линейно уменьшаем аналоговую величину до значения ($B+StSp$). В нашем случае – это 10. Через 100 мс аналоговое значение скачкообразно уменьшается до значения, равного *Смещению* «B». В нашем случае (-50). Подача сигнала «0» на вход En сбрасывает функцию в исходное состояние.

Нормализованное значение аналоговой величины на выходе AQ

$$N = \frac{A (\text{Current level}) - b (\text{Offset})}{k (\text{Gain})}$$

где A – реальное значение аналоговой величины;

B – Смещение (offset).

K – Усиление (gain).

Для $L1 = 70$: $N = (70 - (-50))/0,15 = 800$.

Для $L2 = 90$: $N = (90 - (-50))/0,15 = 933$.

ПИ-регулятор.

Блок-схема с ПИ-регулятором представлена на рис. 4.60.

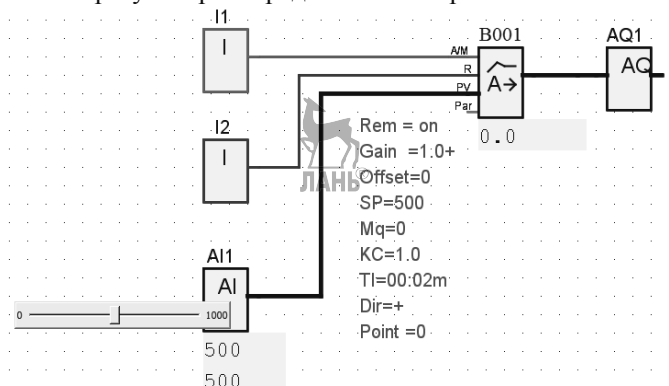


Рис. 4.60. Блок-схема с ПИ-регулятором

Входы и выходы блока «ПИ-регулятор».

– Вход А/М: вход используется для установки режима регулятора:

1: автоматический режим;

0: ручной режим.

– Вход R: вход используется для сброса выхода AQ. Пока этот вход установлен, вход А/М отключен. Выход AQ установлен равным 0.

– Вход PV: на этот вход подается значение управляемого параметра (значение технологической переменной).

– Выход AQ1: на выход подается регулируемая переменная M (например, выходной сигнал управления температурой), пропорциональная ошибке контура $e = SP - PV$.

Параметры блока «ПИ-регулятор».

Параметры блока «ПИ-регулятор» устанавливаются в выпадающем окне, представленном на рис. 4.61. Заданные в этом окне параметры затем можно увидеть на блок-схеме на рис. 4.60:

– Rem=on: сохранение включено;

– Gain = 1.0 (усиление);

– Offset = 0 (смещение);

– SP = 500: заданное значение управляемого параметра (уставка);

– Mq = 0: значение для AQ в ручном режиме, т.е. это значение регулируемой переменной, которое в ручном режиме остается постоянным. В автоматическом режиме значение регулируемой переменной зависит от ошибки контура «e» и автоматически устанавливается ПИ-регулятором.

- $KC = 1.0$: коэффициент усиления, одинаковый для составляющих «П» и «И» ПИ-регулятора ($k_P = k_I = KC$). В случае выбора $KC=0$, составляющая «П» ПИ-регулятора отключается. При этом коэффициент k для части «И» автоматически устанавливается в «1»;
- $TI = 00:02m$ (2 сек): интегральное время;
- Dir : параметр Dir задает направление действия регулятора. **Положительное значение** означает: если уставка **больше** значения технологической переменной, то значение технологической переменной увеличивается; если уставка **меньше** значения технологической переменной, то значение технологической переменной уменьшается. **Отрицательное значение** означает: если уставка **больше** значения технологической переменной, то значение технологической переменной уменьшается; если уставка **меньше** значения технологической переменной, то значение технологической переменной увеличивается. Например, при регулировании системы отопления, если уставка превышает значение технологической переменной (в помещении слишком холодно), то регулируемая переменная увеличивает значение технологической переменной;
- $Point = 0$: число десятичных знаков: 0, 1, 2, 3.

Уточнить число переменных и их значения всегда можно в файле Help, который вызывается либо из главного меню, либо клавишей F1.

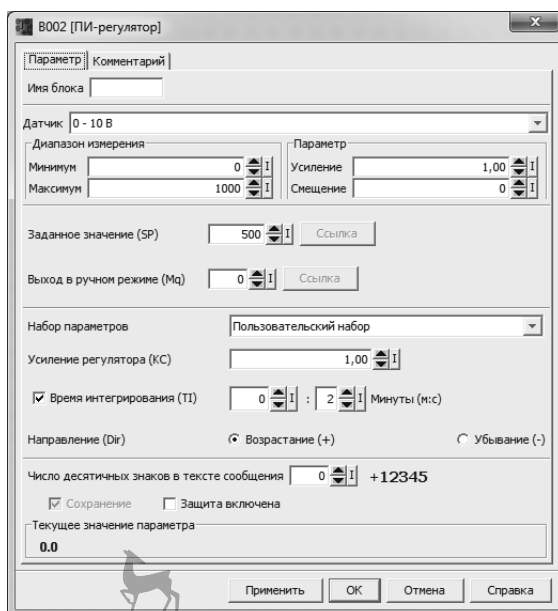


Рис. 4.61. Окно для установки параметров ПИ-регулятора

Характер поведения параметров SP, PV, AQ можно видеть на графике рис. 4.62, который появляется при запуске моделирования.

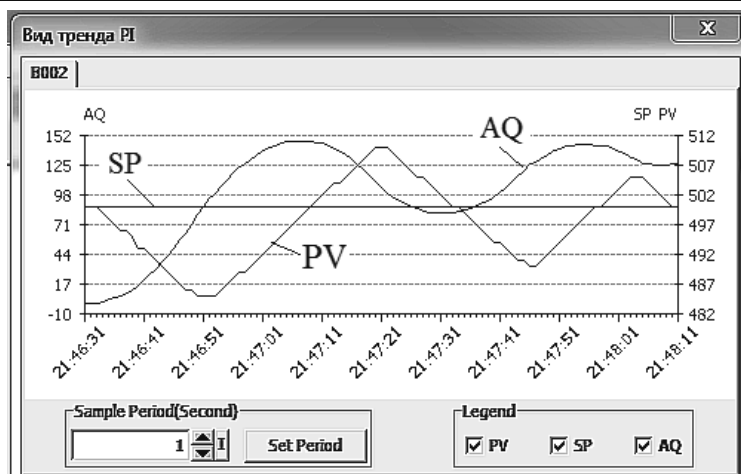


Рис. 4.62. График изменения переменных SP, PV, AQ

Из графика видно, что при запуске моделирования параметр SP сохраняет постоянное заданное значение – 500. Если при помощи движка уменьшать фактическое значение аналоговой величины PV, то появится ошибка контура, и будет вырабатываться регулируемый сигнал (сигнал управления температурой) AQ, пропорциональный ошибке контура.

Для упрощения использования ПИ-регулятора в нем запрограммированы наборы параметров КС, ТИ и Dir, которые устанавливаются в строке *Набор параметров* в окне на рис. 4.61. Возможные наборы параметров и соответствующие им значения переменных КС, ТИ и Dir перечислены в таблице 4.5.

Таблица 4.5

Набор параметров	Пример применения	Пара метр КС	Пара метр ТИ (с)	Пара метр Dir
Температура – быстро	Температура, управление охлаждением небольших пространств, небольших объемов	0,5	30	+
Температура – медленно	Отопление, вентиляция, температура, управление охлаждением больших пространств, больших объемов	1,0	120	+
Давление 1	Быстрое изменение давления, управление компрессором	3,0	5	+

Давление2	Медленное изменение давления, дифференциальное управление давлением (регулятор расхода)	1,2	12	+
Заполнение до уровня 1	Заполнение емкости без слива	1,0	99:59	+
Заполнение до уровня 2	Заполнение емкости со сливом	0,7	20	+

Блок широтно-импульсного модулятора (PWM)

Блок-схема с широтно-импульсным модулятором приведена на рис. 4.63.

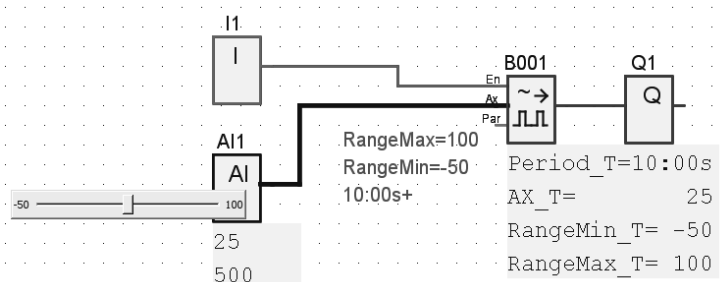


Рис. 4.63. Блок-схема с широтно-импульсным модулятором

Широтно-импульсный модулятор (PWM – Pulse Width Modulator) предназначен для преобразования заданного значения Ax аналогового входного сигнала в импульсный цифровой выходной сигнал. При этом в течение периода времени T генерируется прямоугольный импульс длительностью T_1 , в течение которого на выходе Q имеем сигнал «1», и остальное время $T_2 = T - T_1$ сигнал на выходе Q равен «0», как показано на рис. 4.64. Затем ситуация повторяется до тех пор, пока на входе «En» установлен высокий уровень. Длительность временных промежутков T_1 и T_2 определяется значением аналоговой величины, подаваемой на вход Ax .

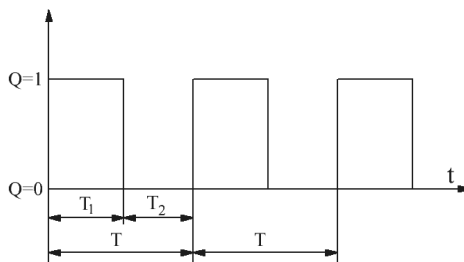


Рис. 4.64. Преобразование аналогового сигнала в цифровые сигналы

Параметры, задаваемые для функции, показаны в окне на рис. 4.65. Рассмотрим конкретный пример. Пусть фактический диапазон аналоговой величины изменяется от (-50°C) до (+100°C). Тогда следует задать:

- Усиление: $k = 0,15$.
- Смещение: $b = -50$.

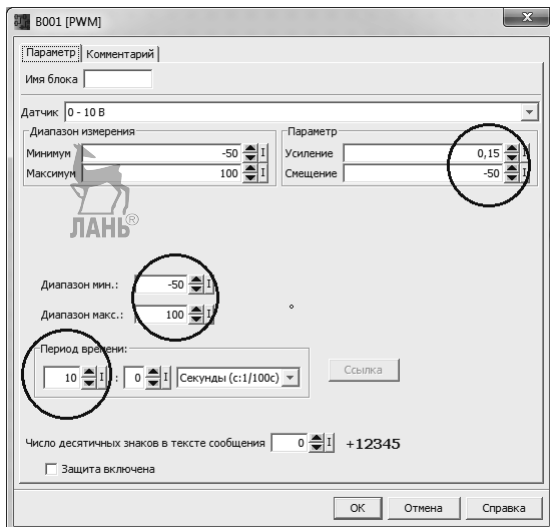


Рис.4.65. Окно выбора параметров

Усиление и смещение задаются таким образом, чтобы согласовать фактический диапазон изменения аналоговой величины от (-50°C) до (+100°C) с диапазоном значений 0...1000, в котором работает контроллер LOGO!

Далее в окне на рис. 4.65 задаются:

- Диапазон мин: $A_{min} = -50$.
- Диапазон макс.: $A_{max} = 100$.
- Период времени: $T = 10$ сек.

Указанные в окне на рис.4.65 параметры можно видеть около блока широтно-импульсного модулятора В001 на рис. 4.63.

Значения T_1 и T_2 вычисляются по формулам

$$T_1 = \frac{A_x - A_{min}}{A_{max} - A_{min}} \cdot T \quad (4.1)$$

$$T_2 = T - T_1 \quad (4.2)$$

Для нашего случая $A_x = 25$, тогда из (4.1) и (4.2) находим

$$T_1 = \frac{25 - (-50)}{100 - (-50)} \cdot 10 = 5 \text{ с}$$

$$T_2 = 10 - 5 = 5 \text{ с}$$

Таким образом, при подаче на аналоговый вход аналоговой величины $A_x = 25$ на выходе Q в течение 5 с будет сигнал «1» и в течение следующих 5 с – сигнал «0» и так далее, пока на входе «En» установлен высокий уровень. При установке «0» на входе «En» происходит сбрасывание функции.

4.5.7. Тексты сообщений.

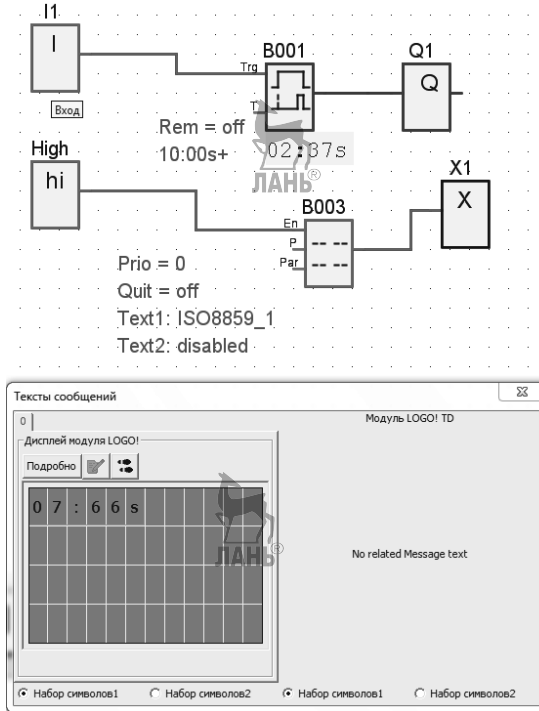


Рис. 4.66. Блок-схема с выводом текста сообщения

На рис. 4.66 приведена блок-схема, на которой присутствует дополнительная цепочка с блоком *Тексты сообщений* (блок B003). При запуске эмуляции рядом с блок-схемой автоматически появится окно, в котором будет отображаться текст сообщения. Это окно называется «Тексты сообщений».

Блок текста сообщения имеет следующие входы:

вход En: переход из 0 в 1 на входе En (разрешение) вызывает вывод текста сообщения;

вход P: определяет приоритет текста сообщения,
0 – минимальный приоритет, 127 – самый высокий приоритет.

вход Par: используется для задания параметров настройки.

Окно для задания параметров настройки представлено на рис. 4.67. В данном случае в этом окне в качестве текста сообщения задано оставшееся время до срабатывания блока задержки включения (B001).

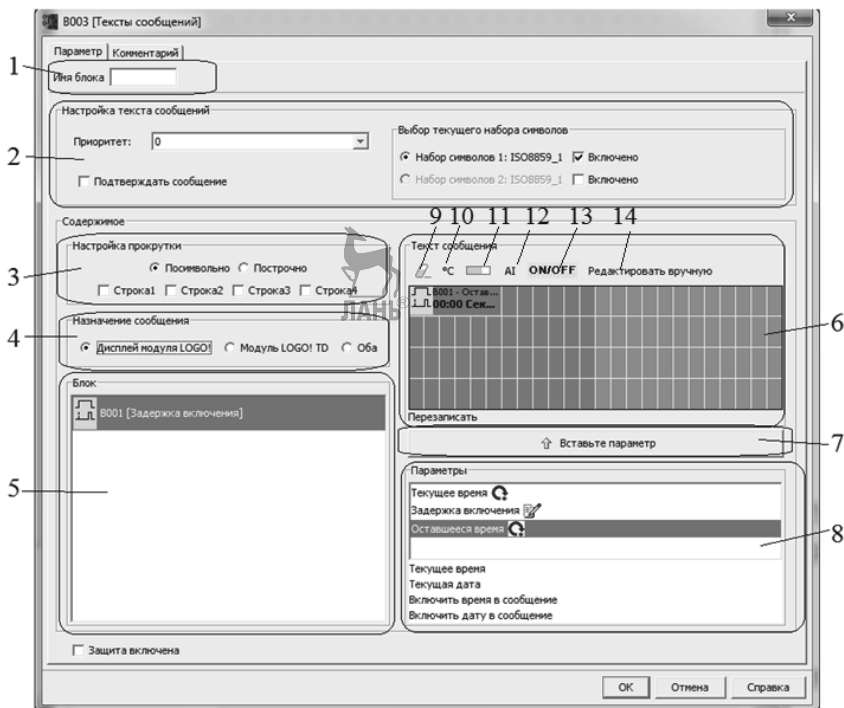


Рис. 4.67. Окно выбора параметров блока *Тексты сообщений*

1. Область имени блока.

2. Область настроек.

Можно задать:

- приоритет текста сообщения;
- подтверждение сообщения (в случае выбора параметра потребуется подтверждение сообщения для его закрытия);
- выбор набора символов для текста сообщения.

3. Область прокрутки.

Здесь задаются параметры прокрутки текста:

- посимвольный формат прокрутки;
- построчный формат прокрутки;
- блок выбора разрешения прокрутки для каждой строки дисплея.

4. Область назначения сообщения.

Здесь можно задать устройство для отображения сообщений: или дисплей LOGO!, или дисплей LOGO! TD, или оба устройства вместе.

5. Зона блоков.

В этой зоне выполняется выбор отдельных блоков, из всей совокупности блоков коммутационной программы, для отображения текста сообщения.

6. **Зона сообщений.** Введенная в этой зоне информация соответствует тому, что будет отображаться на дисплее модуля LOGO! или LOGO! TD.
7. **Клавиши вставки параметра.** Эта клавиша служит для вставки выбранного параметра блока в текст сообщения.
8. **Зона параметров блока.** В этой зоне выполняется выбор параметров, подлежащих отображению в тексте сообщения.
9. **Клавиша Удалить.** Для удаления записей из области сообщений.
10. **Клавиша специальных символов.** Для вставки специальных символов в область сообщений.
11. **Клавиша гистограммы.** Для вставки горизонтальной или вертикальной гистограммы в область сообщения.
12. **Клавиша AI.** Для вставки аналогового входного значения в область сообщения.
13. **Клавиш включения и отключения.** Для задания двух строк, представляющих цифровые значения, соответствующие состоянию 0 и состоянию 1, например, «ВЫКЛ» и «ВКЛ».
14. **Клавиша ручного редактирования.** Для использования статического редактора с целью перемещения, добавления или изменения элементов текста сообщения без изменения положения других элементов.

В случае срабатывания нескольких блоков текстовых сообщений, отображается сообщение с самым высоким приоритетом.

Если коммутационная программа использует флаг M27, то в случае, если $M27=0$

(низкий уровень) отображаются только те тексты сообщений, которые составлены из символов основного набора (набор символов 1). Когда $M27=1$ (высокое значение), отображаются тексты сообщений только вспомогательного набора символов (набор символов 2).

После отключения или подтверждения текста сообщения дисплей автоматически

показывает ранее отображавшийся текст сообщения, имеющий наивысший приоритет.

В тексте сообщения могут отображаться:

- **оставшееся время таймера.** Параметр оставшегося времени отображает время, оставшееся до истечения времени таймера. Для таймеров с многочисленными значениями времени (например, с временем задержки включения, временем задержки выключения) можно отображать в тексте сообщения оставшееся время для каждого параметра;
- **отображение аналоговых входов.** Можно выбрать аналоговые входы, подлежащие отображению в тексте сообщения. Для вставки конкретного AI в область текста сообщения служит кнопка AI. Периодичность обновления информации можно задать 100 мс, 200 мс, 400 мс, 800 мс и 1000 мс. Если в тексте сообщения используется больше одного аналогового входа, периодичность обновления используется для всех входов.

Определим **последовательность действий**, необходимую для вывода на дисплей текста сообщений.

- ✓ Составить блок-схему, в которой используется блок текста сообщений.
- ✓ Щелкнуть по блоку *Тексты сообщений*, чтобы появилось окно для задания параметров этого блока.
- ✓ В области 5 окна задания параметров выбрать блок, для которого нужно вывести сообщение. Для выбора блока щелкнуть по нему левой кнопкой мыши, при этом блок будет подсвечен синим цветом.
- ✓ В области 8 выбрать параметр, который должен выводиться в окне «Тексты сообщений».
- ✓ Щелкнуть по кнопке *Вставьте параметр*, чтобы выбранный параметр переместился в область 6. В области 6 появится прямоугольник желто-коричневого цвета. Этот прямоугольник можно перемещать мышкой и располагать в любом месте пространства, состоящего из зеленых прямоугольников. Зеленые прямоугольники ограничивают область видимости дисплея модуля LOGO! Можно располагать информацию и за пределами зеленого поля, но тогда потребуются функция прокрутки области видимости дисплея.
- ✓ Запустить эмуляцию и наблюдать текст в автоматически появившемся окне «Тексты сообщений».

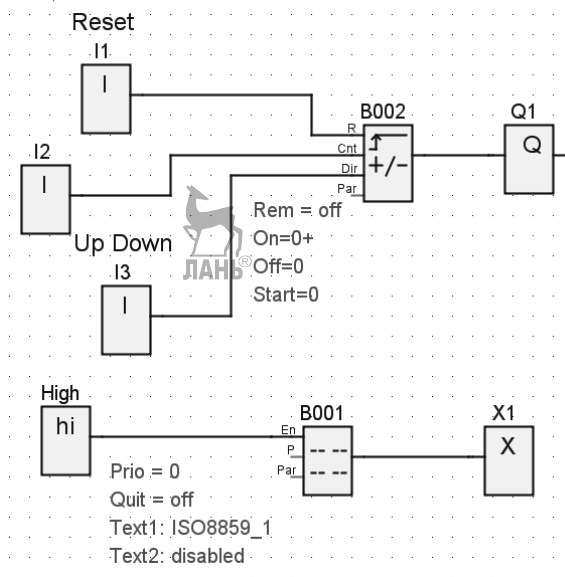


Рис. 4.68. Блок-схема с реверсивным счетчиком и блоком текстовых сообщений

Далее рассмотрим, как вывести столбиковую диаграмму в окне «Тексты сообщений». Составим блок-схему, содержащую блок «Реверсивный счетчик» и дополнительную цепочку, содержащую блок «Тексты сообщений», как показано на рис. 4.68.

Щелкнем по блоку «Тексты сообщений», чтобы появилось окно настройки параметров, показанное на рис. 4.69. Выберем в области 8 параметр «Счетчик» и переместим его в область 6. В области 6 укажем место расположения столбиковой

диаграммы. Для этого щелкнем по одному из зеленых прямоугольников, который поменяет при этом цвет и из зеленого станет синим. Этот прямоугольник надо расположить правее желто-коричневого прямоугольника, чтобы он не помешал столбиковой диаграмме, которая пойдет вверх, как показано стрелкой на рис. 4.69.

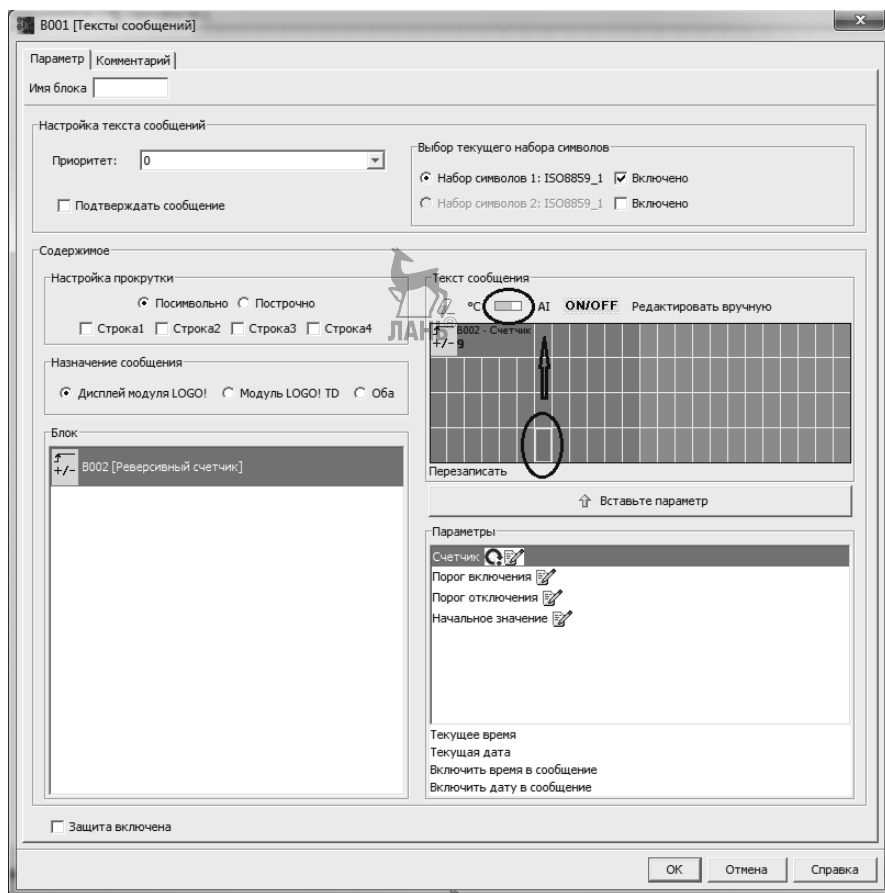


Рис. 4.69. Окно *Тексты сообщений*

Теперь щелкнем по клавише 11 «Клавиша гистограммы». Появится выпадающее окно, показанное на рис. 4.70. Установим в этом окне следующие параметры:

- в строке «Блок» установим «Реверсивный счетчик»;
- в строке Текущее значение блока установим «Счетчик»;
- в строке «MinValue» установим 0, в строке «MaxValue» установим 10;
- в строке «Направление» установим «Вертикальная»;
- в строке «Высота» установим 4.

Щелкнем по кнопке ОК.

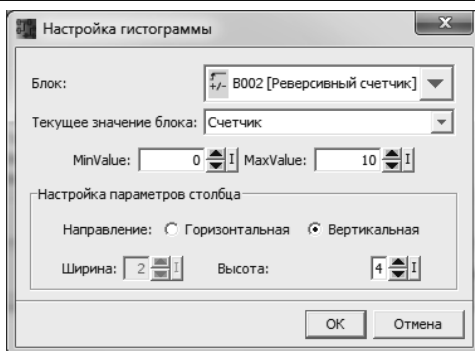


Рис. 4.70. Окно *Настройка гистограммы*

Далее запустим эмуляцию, и в окне «Тексты сообщений» на рис. 4.71 будем наблюдать переменные показания счетчика (цифра 4) и переменную столбиковую диаграмму.

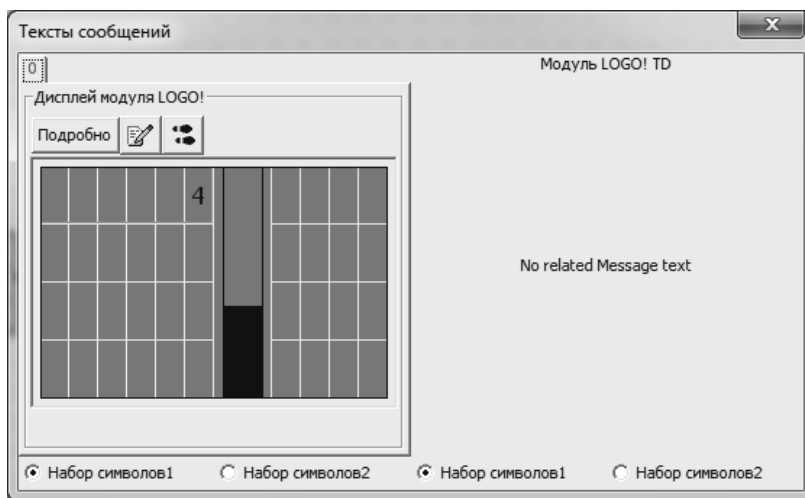


Рис. 4.71. Окно *Тексты сообщений*

4.5.8. Реализация циклического подключения выходов

В программах с циклическим подключением выходов широко используются блоки «Реле с блокировкой» и «Импульсное реле». Поэтому вначале рассмотрим работу этих блоков, а затем перейдем к примерам программ.

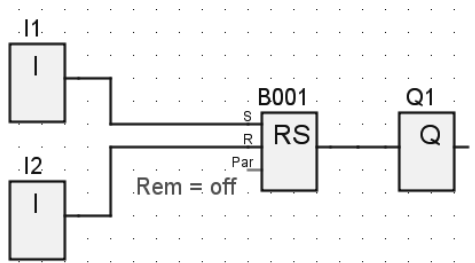
Реле с блокировкой.

Рис. 4.72. Блок-схема с функцией Реле с блокировкой

Блок-схема с функцией *Реле с блокировкой* представлена на рис. 4.72. Сигнал «1» на входе S устанавливает «1» на выходе Q. Если затем подать «0» на вход S, то на выходе остается «1» (т.е. состояние выхода остается без изменения, в этом и заключается суть блокировки). Сигнал «1» на входе R сбрасывает выход Q в «0».

Фактически, реле с блокировкой представляет собой простое двоичное логическое запоминающее устройство. Выходное значение зависит от состояния входов и от предыдущего состояния выхода. Различные комбинации состояний входов и выхода приведены в таблице 4.6.

Таблица 4.6

S	R	Q	Примечание
0	0	x	Состояние выхода сохраняется
0	1	0	Сброс реле
1	0	1	На выходе устанавливается «1»
1	1	0	Сброс реле

Импульсное реле.

Блок-схема с функцией *Импульсное реле* представлена на рис. 4.73. Если оба входа S и R находятся в нулевом состоянии, то выход Q последовательно устанавливается и сбрасывается (включается и выключается) подачей «1» на вход Trg. Первая подача «1» на вход Trg переводит выход Q в «1». Вторая подача «1» на вход Trg переводит выход Q в «0» и т.д.

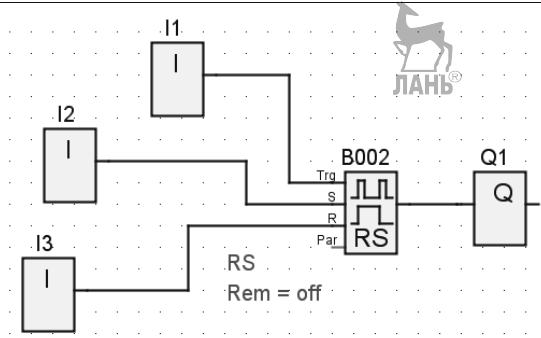


Рис. 4.73. Блок-схема с функцией *Импульсное реле*

Подача «1» на вход S переводит выход Q в «1». Последующая подача «1» на вход R сбрасывает выход Q в «0».

В свойствах блока *Импульсное реле* можно установить опции:

1. Вход R имеет приоритет над входом S.
2. Вход S имеет приоритет над входом R.

В первом случае при включенном входе S вход R устанавливает и сбрасывает (включает и выключает) выход Q. При этом около блока появляется запись RS, как показано на рис. 4.73. Во втором случае при включенном входе R вход S устанавливает и сбрасывает выход Q, а около блока появляется запись SR.

Если Trg = 0 и установлен приоритет RS, то функциональный блок *Импульсное реле* соответствует функциональному блоку *Реле с блокировкой*.

Пример 4.4. Циклическое включение и отключение выхода (рис. 4.74). При переводе входа I1 из «0» в «1» выход Q1 начинает автоматически включаться и отключаться через время задержки, определяемое блоком B002 «Задержка включения и отключения».

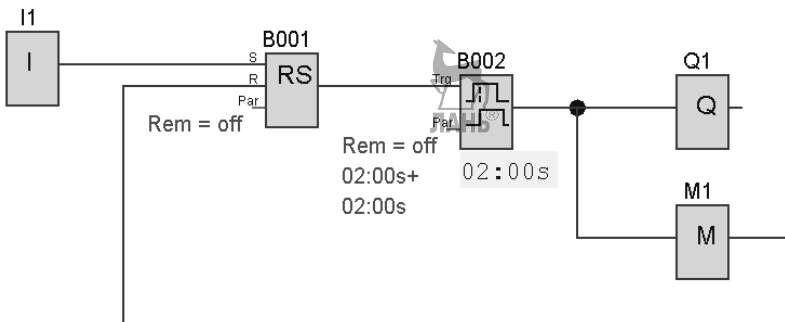


Рис. 4.74. Циклическое включение/отключение выхода

Пример 4.5. Циклическое переключение двух выходов (рис. 4.75). При переводе входа I1 из «0» в «1» выходы Q1 и Q2 начинают автоматически

переключаться через время задержки, определяемое блоком B002 «Задержка включения и отключения».

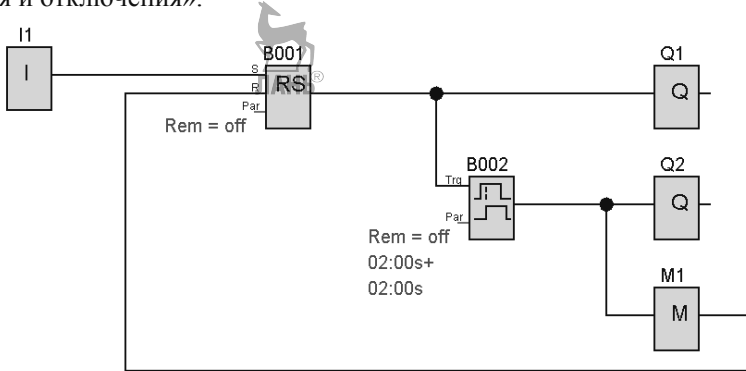


Рис. 4.75. Циклическое переключение выходов

Пример. 4.6. Попеременное включение двух выходов (рис. 4.76). Управление двумя выходами (попеременное включение двух выходов, например, двух станков) осуществляется пусковой кнопкой, связанной с входом I1. Кнопка 2, связанная с входом I2, останавливает работу станков. При поступлении сигнала «1» на вход I1 запускается станок №1, связанный с выходом Q1. При вторичной подаче «1» на вход I1 запускается станок №2, связанный с выходом Q2, а станок №1 отключается. Если на вход I1 еще раз подать «1», то станок №1 опять подключается, а станок №2 отключается.

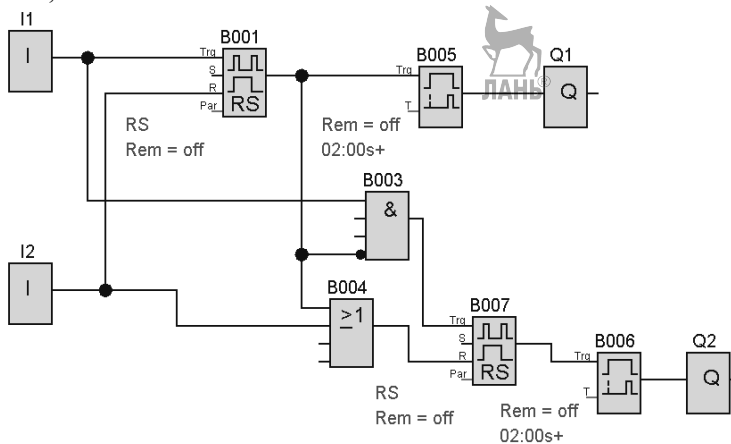


Рис. 4.76. Попеременное включение двух выходов

Таким образом, при каждой подаче «1» на вход I1 в работу поочередно включаются станки №1 и №2. Включение и отключение станков происходит с некоторой задержкой, определяемой блоком «Задержка включения». Оба станка отключаются при подаче «1» на вход I2.

Пример 4.7. Циклическое включение выходов (рис. 4.77). При переводе входа I1 из «0» в «1» выходы Q1...Q3 начинают последовательно включаться через время задержки, определяемое блоками B002, B003, B004 «Задержка включения». Через время задержки, определяемое блоком B005, все выходы одновременно отключаются. Далее выходы начинают опять последовательно включаться и процесс циклически повторяется. При переводе входа I1 из «1» в «0» работа программы останавливается.

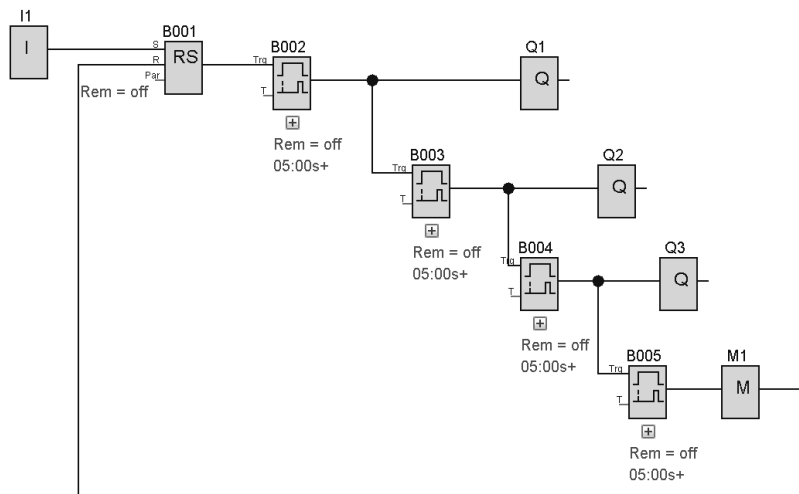


Рис. 4.77. Циклическое включение выходов

Пример 4.8. Циклическое отключение выходов (рис. 4.78). При переводе входа I1 из «0» в «1» все выходы Q1...Q3 одновременно включаются и затем начинают последовательно отключаться через время задержки, определяемое блоками B002, B003, B004 «Задержка отключения». Через время задержки, определяемое блоком B005 все выходы опять одновременно включаются и начинают последовательно отключаться и процесс циклически повторяется. При переводе входа I1 из «1» в «0» работа программы останавливается.



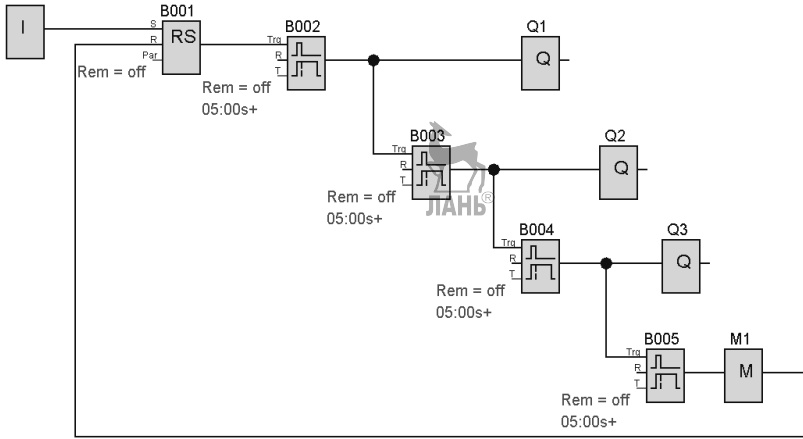


Рис. 4.78. Циклическое отключение выходов.

Пример 4.9. Последовательное переключение выходов (рис. 4.79). При каждом переводе входа I1 из «0» в «1» последовательно подключаются выходы Q1...Q4. При подключении последующего выхода, предыдущий выход отключается. После отключения выхода Q4 подключается выход Q1 и процесс повторяется циклически.

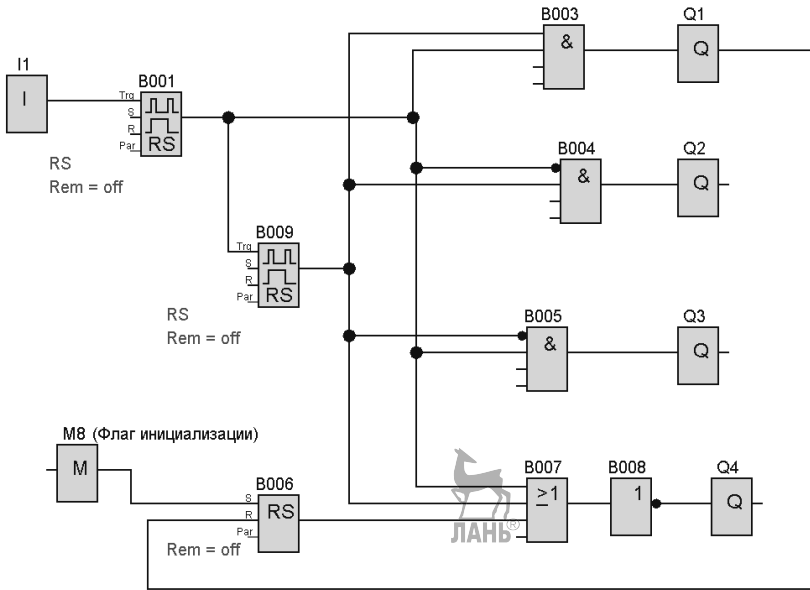


Рис. 4.79. Последовательное включение выходов

Пример 4.10. Последовательное включение и отключение выходов (рис. 4.80). При переводе входа I1 из «0» в «1» последовательно включаются выходы Q1...Q4 сверху вниз через время задержки, определяемое блоками B006, B007, B001 «Задержка включения». При последующем переводе I1 из «1» в «0» последовательно отключаются выходы Q4...Q1 снизу вверх через время, определяемое блоками B008, B004, B009, B012 «Задержка включения».

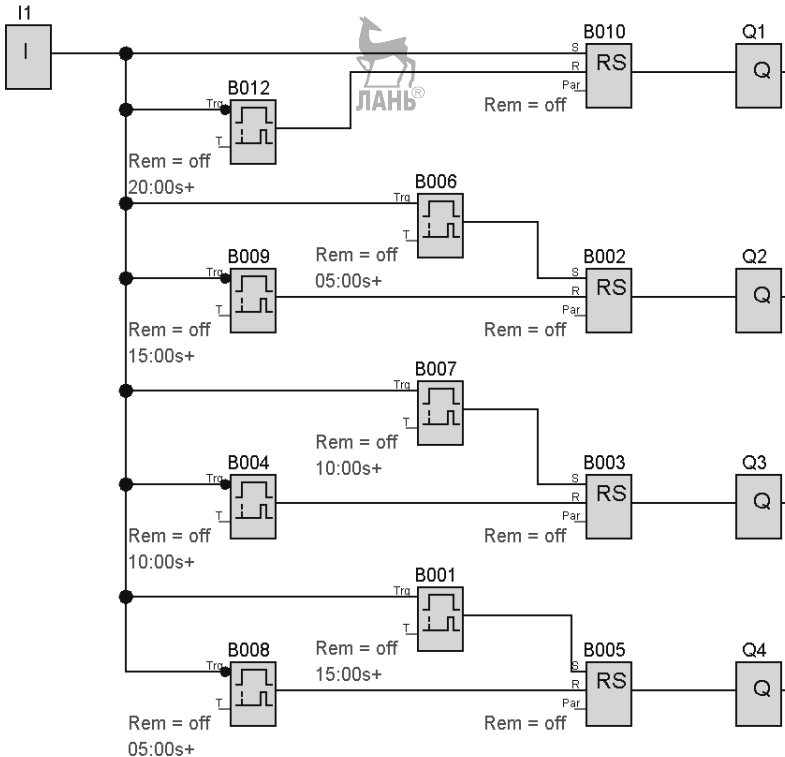


Рис. 4.80. Последовательное включение и отключение выходов.

Рассмотрим также в этом подразделе функцию **Программный выключатель**, которая может использоваться в программах с циклическим подключением выходов.

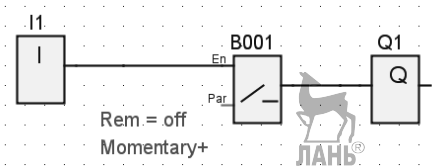


Рис. 4.81. Блок-схема с программным выключателем

Блок-схема с программным выключателем приведена на рис. 4.81, а окно настройки параметров блока на рис. 4.82.

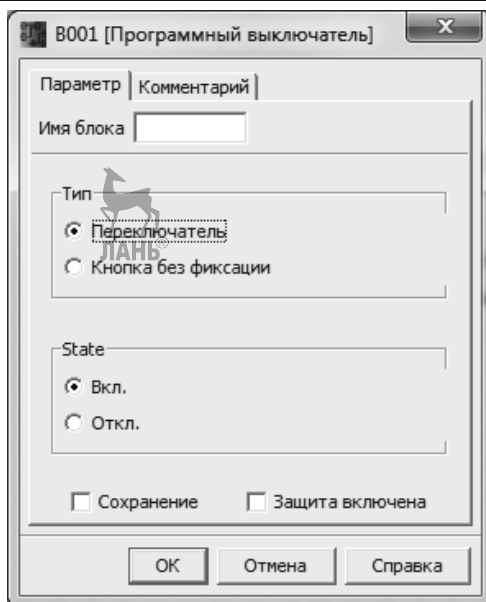
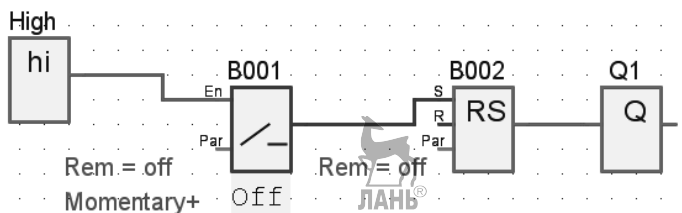


Рис. 4.82. Окно настройки параметров

При выборе опции *Переключатель* (Тип) и установке состояния (State) в режим «Вкл» блок работает, как обычный выключатель, т.е. при переходе из «0» в «1» на входе En выход Q включается, при переходе из «1» в «0» выход Q отключается.

При выборе опции *Кнопка без фиксации* (Тип) и установке состояния (State) в режим «Вкл» выход Q остается подключенным только в течение одного цикла и затем сразу сбрасывается в 0. Увидеть состояние «1» на выходе Q для этого режима работы можно с помощью блок-схемы, представленной на рис. 4.83.

Рис. 4.83. Блок-схема для обнаружения срабатывания программного выключателя в режиме *Кнопка без фиксации*

4.5.9. Примеры управляющих программ в LOGO! Soft Comfort

Пример 4.11. Автоматическое управление освещением.

Для автоматического управления включением и отключением освещения можно использовать коммутационную программу, показанную на рис. 4.84.

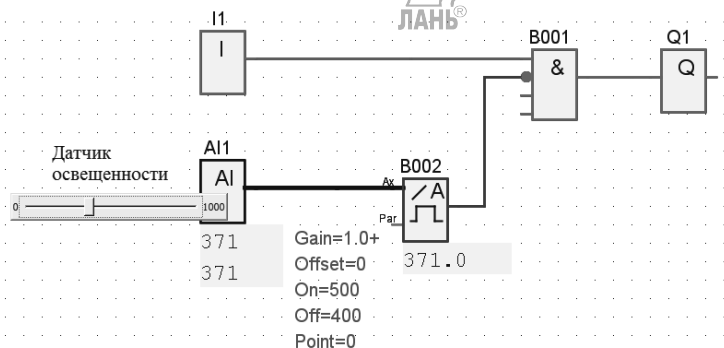


Рис. 4.84. Коммутационная программа для автоматического управления освещением

В программе используется логическая функция «И» (блок B001) и аналоговый пороговый выключатель (блок B002). Точка около второго входа блока B001 означает инверсию.

Схема работает следующим образом. Пороговый выключатель задает порог включения и отключения освещения. Включение схемы осуществляется переключателем I1. При наличии сигнала «1» на первом входе элемента «И» и отсутствии сигнала на втором входе, сигнал «1» проходит на выход Q. Это означает, что при освещенности ниже порогового уровня освещение включается. В дальнейшем, при увеличении освещенности, срабатывает пороговый выключатель (B002) и освещение отключается.

В качестве управляющего сигнала на аналоговом входе AI1 может использоваться датчик освещенности, выдающий напряжение от 0 до 10 вольт. Этому диапазону напряжений соответствуют внутренние значения контроллера LOGO! от 0 до 1000. Заданным уровням порогов срабатывания аналогового выключателя (500 и 400) соответствуют пороговые напряжения 5В и 4В датчика освещенности.

Пример 4.12. Управляемые жалюзи.

На рис. 4.85 представлена коммутационная программа для управления рулонными жалюзи. В программе предусмотрены ручной и автоматический режимы работы. В зависимости от времени суток жалюзи либо подняты (свернуты) вверх, либо опущены вниз.

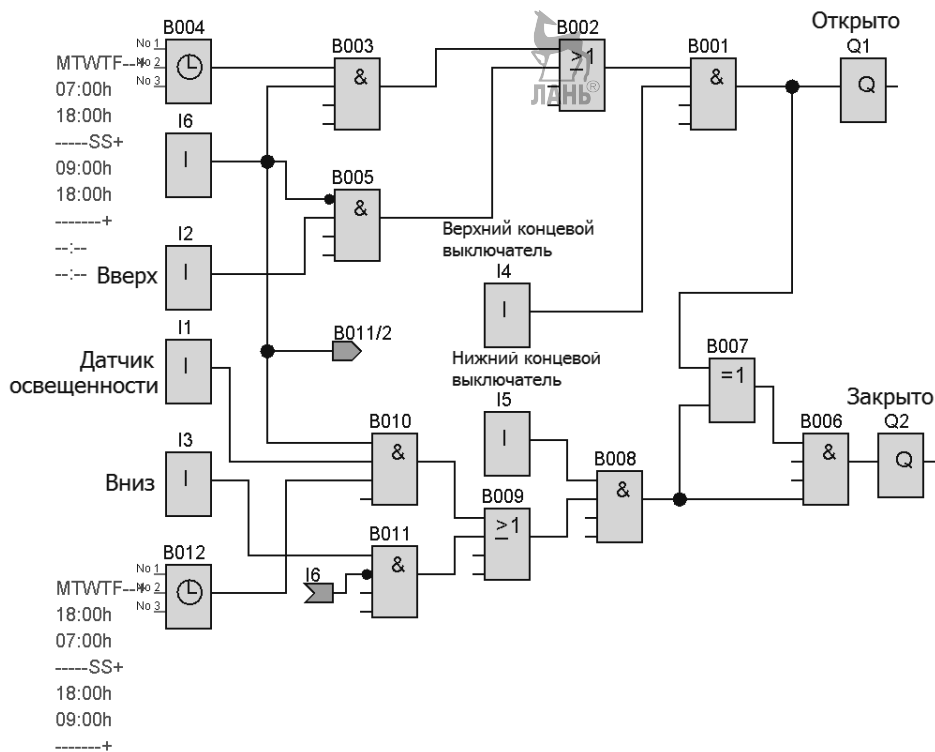


Рис. 4.85. Коммутационная программа для управления работой жалюзи

Используемые компоненты:

- I1 – датчик освещенности (NO контакт, используется при работе системы в автоматическом режиме);
- I2 – ручной переключатель, движение вверх (NO контакт);
- I3 – ручной переключатель, движение вниз (NO контакт);
- I4 – концевой выключатель верхнего положения жалюзи (NC контакт). Когда жалюзи находятся в нижнем положении, контакт замкнут и размыкается, когда жалюзи достигнут верхнего положения (откроются).
- I5 – концевой выключатель нижнего положения жалюзи (NC контакт). Когда жалюзи находятся в верхнем положении, контакт замкнут и размыкается, когда жалюзи достигнут нижнего положения (закроются).
- I6 – переключатель режима работы (автоматический/ручной);
- Q1 – открывание жалюзи;
- Q2 – закрывание жалюзи.

Программа работает следующим образом.

Ручной режим. С помощью переключателей I2 (вверх) и I3 (вниз) жалюзи можно вручную открывать и закрывать при условии, что переключатель I6 не установлен в автоматический режим (не включен). Предположим, жалюзи находятся в нижнем положении. Контакт I4 замкнут. При замыкании контакта I2 сигнал

проходит на выход Q1, жалюзи начнут подниматься вверх. Когда жалюзи достигнут верхнего положения, концевой выключатель I4 размыкается и обесточивается цепь электродвигателя, поднимающего жалюзи (выход Q1 отключается). При симуляции контакт I4 отключается вручную.

При опускании вниз контакт I5 находится в замкнутом положении. При замыкании контакта I3 сигнал проходит на выход Q2, жалюзи начинают опускаться вниз. В нижнем положении концевой выключатель I5 размыкается и обесточивает электродвигатель (выход Q2 отключается).

Автоматический режим. Для работы в автоматическом режиме переключатель I6 должен быть включен. Если датчик освещенности I1 активирован (включен), то жалюзи закрыты в с 18:00 вечера до 7:00 утра по будням и с 18:00 до 9:00 в субботу и воскресенье. Если датчик освещенности I1 выключен, то жалюзи открыты с 7:00 утра до 18:00 вечера по будням и с 9:00 до 18:00 в субботу и воскресенье. Блок *Семидневный таймер* автоматически отслеживает время суток.

Блок V004 отслеживает работу жалюзи в дневное время. Щелчком левой кнопкой мыши по блоку V004 и установим в выпавшем окне на вкладке *Переключатели 1* для рабочих дней недели время включения 7.00 и время отключения 18.00, как показано на рис. 4.86. На вкладке *Переключатели 2* задаем время работы в субботу и воскресенье, как показано на рис. 4.87.

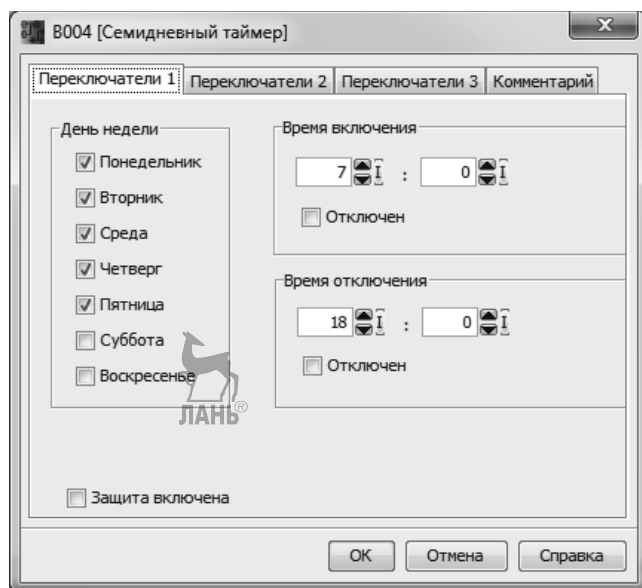


Рис. 4.86. Окно выбора параметров в будние дни

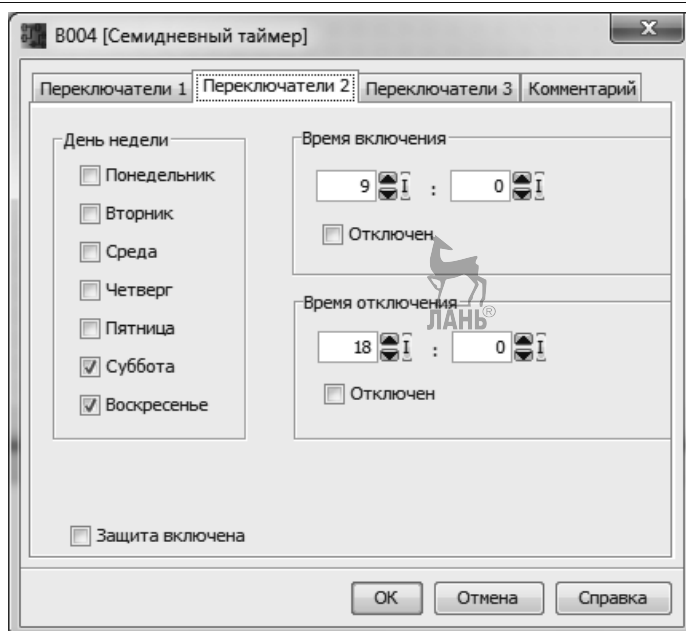


Рис. 4.87. Окно выбора параметров в субботу и воскресенье

Блок В012 отслеживает работу жалюзи в ночное время. Установим параметры работы жалюзи в ночное время. Щелкнем левой кнопкой мыши по блоку и установим в выпавшем окне во вкладке *Переключатели 1* время включения 18.00 и время отключения 7.00 для рабочих дней недели. Для субботы и воскресенья во вкладке *Переключатели 2* установим время включения 18.00 и время отключения 9.00.

Запустим программу на симуляцию. Чтобы проверить работу жалюзи в автоматическом режиме в дневное и ночное время, не обязательно ждать наступления дня или ночи. Можно смоделировать время с помощью пиктограммы на панели инструментов, на которой изображены часы (поз.13, рис. 4.10). После щелчка левой кнопкой мыши по этой пиктограмме появляется выпадающее окно, в котором можно установить дату и время.

Пример 4.13. Освещение охраняемой территории.

Коммутационная программа на рис. 4.88 предназначена для автоматического управления освещением охраняемой территории. Освещение может включаться вручную или автоматически. Автоматический режим освещения работает в ночное время суток. Для экономии электроэнергии ночное освещение подразделяется на дежурное и дополнительное (усиленное) освещение. Дежурное освещение горит постоянно все ночное время. Дополнительное освещение подключается кратковременно при срабатывании датчика движения.

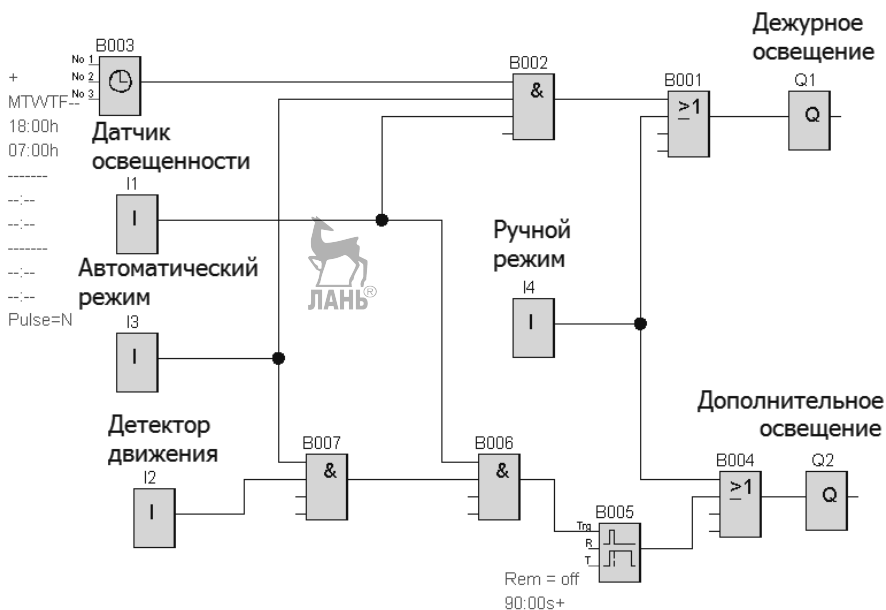


Рис. 4.88. Коммутационная программа для управления освещением

Используемые компоненты:

- I1 – датчик освещенности (NO контакт);
- I2 – детектор движения (NO контакт);
- I3 – выключатель автоматического режима (NO контакт);
- I4 – выключатель ручного режима (NO контакт);
- Q1 – дежурное освещение;
- Q2 – дополнительное освещение.

Программа работает следующим образом.

Ручной режим. Ручной режим управляется переключателем I4. В ручном режиме одновременно включается и дежурное и дополнительное освещение.

Автоматический режим. Время дежурного освещения в автоматическом режиме устанавливается семидневным таймером (блок B003). Переключатель автоматического режима I3 – включен. Переключатель I1 (датчик освещенности) – включен. Переключатель ручного режима I4 – выключен. Выход Q1 при этом активируется и работает дежурное освещение. При срабатывании датчика движения I2 выход Q2 активируется и к дежурному освещению подключается усиленное дополнительное освещение. По заднему фронту сигнала на датчике I2 запускается отсчет времени выключения выхода Q2 и, например, через 90 с выход Q2 отключается.

Пример 4.14. Гаражные ворота. Гаражные ворота представляют собой одну створку, которая при открытии ворот сдвигается в сторону. На рис. 4.89 представлена коммутационная программа для управления гаражными воротами.

Используемые компоненты:

I1 – пусковая кнопка «Открыть» (NO контакт);

I2 – пусковая кнопка «Закрыть» (NO контакт);

I3 – кнопка «Стоп» (NO контакт);

I4 – концевой выключатель закрытого положения ворот (NC контакт). Контакт при закрытой створке ворот замкнут и размыкается, когда створка полностью сдвинется в сторону;

I5 –концевой выключатель открытого положения ворот (NC контакт). Контакт при открытой створке ворот замкнут и размыкается, когда створка полностью перекроет проезд;

I6 – предохранительная планка. Если ворота в момент закрытия встречают препятствие, то срабатывает нажимной контакт под предохранительной планкой и закрытие останавливается (NO контакт);

Q1 – при наличии сигнала на выходе запускается электродвигатель, открывающий ворота;

Q2 – при наличии сигнала на выходе запускается электродвигатель, закрывающий ворота;

Q3 – мигающий предупредительный сигнал.

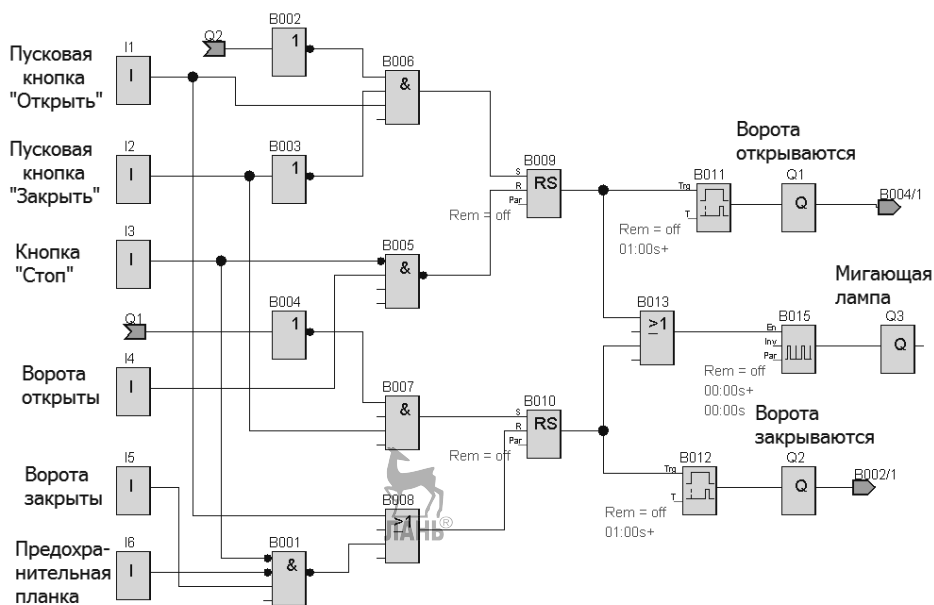


Рис. 4.89. Коммутационная программа для управления гаражными воротами

Программа работает следующим образом.

Предположим, ворота закрыты. Контакт I4 замкнут, т.е. находится в состоянии «1». При нажатии кнопки *Открыть* (подаем «1» на вход I1), сигнал проходит на выход Q1 и через заданное время задержки, устанавливаемое в блоке B011,

электродвигатель начинает открывать ворота (сдвигать створку в сторону). Когда ворота полностью откроются, концевой выключатель I4 выключается (переходит в состояние «0»), а концевой выключатель I5 включается (переходит в состояние «1»). В режиме симуляции это делается вручную. Чтобы закрыть ворота, нажимаем кнопку *Закрыть* (подаем «1» на вход I2). Если кнопка I1 при этом остается нажатой, то ничего не происходит. Таким образом, в программе блокируется одновременное нажатие двух кнопок. Сигнал «1» от кнопки I2 проходит на выход Q2 и через заданное время задержки начинает работать электродвигатель, закрывающий ворота. Когда ворота полностью закроются, размыкается контакт I5 и электродвигатель останавливается. Если ворота в процессе закрытия встречают препятствие, то срабатывает контакт I6 и закрытие ворот прекращается. Кроме того, открытие и закрытие ворот можно остановить в любой момент кнопкой I3 «Стоп». Процесс закрытия и открытия ворот сопровождается миганием лампочки (выход Q3).

Контрольные вопросы и задания.

1. Для решения каких задач следует выбирать логические контроллеры Siemens Simatic S7, и для каких задач – логические модули Siemens LOGO?
2. Аналоговые датчики могут иметь выходы 0...10 В или 0/4...20 мА. Какие из них можно подключать к базовому логическому модулю LOGO?
3. Что означает буква R в обозначении логических модулей LOGO?
4. К каким базовым логическим модулям LOGO! можно подсоединить аналоговые модули расширения?
5. Какие модели базовых модулей LOGO! имеют интерфейс Ethernet?
6. Какая модель базовых модулей LOGO! допускает управление через Интернет?
7. Какие три способа программирования можно использовать для логических модулей LOGO?
8. Как называется интегрированная среда программирования для логических модулей LOGO?
9. Можно ли отладить на компьютере коммутационную программу, не имея в наличии реального логического модуля?
10. Перечислить группы блоков, которые используются в интегрированной среде программирования LOGO!Soft Comfort.
11. Какие языки программирования можно использовать в LOGO!Soft Comfort?
12. Привести условно-графические обозначения для блоков *Задержка включения*, *Задержка отключения*, *Задержка включения/отключения* в языке программирования FBD.
13. Построить блок-схему и коммутационную программу для логического выражения

$$Y = AB + (\bar{A} + \bar{B}) + \overline{ABC}$$

14. Разработать коммутационную программу для отслеживания рабочего режима вращения вала электродвигателя. Вал электродвигателя может вращаться с угловой скоростью от 0 до 3000 об/мин. Рабочий режим

вращения вала располагается в диапазоне угловых скоростей от 1880 до 2200 об/мин. Если скорость вращения выходит за пределы указанного диапазона, то должна загораться тревожная лампа. Угловая скорость вращения измеряется аналоговым датчиком.

15. Разработать программу охранной сигнализации, работающей по следующему алгоритму. В ночное время суток (от 18.00 вечера до 7.00 утра) по датчику движения включается свет на 10 с и одновременно включается прерывистый звуковой сигнал на 5 с. Эта последовательность повторяется три раза с перерывом 5 с.



ГЛАВА 5. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ONI

5.1. Номенклатура контроллеров ONI

Торговая марка ONI принадлежит российской группе компаний IEK Group. Номенклатура контроллеров ONI делится на базовые контроллеры и программируемые реле. Программируемые реле, в свою очередь, подразделяются на микро ПЛК и логические реле.

Базовые контроллеры ONI PLC-S относятся к контроллерам старшего уровня. Они имеют модульную конструкцию и используются для построения систем управления средней и малой сложности. Габариты модулей ONI PLC-S имеют небольшие размеры, не более 91×63×30 мм. Несмотря на такую компактность, они обладают высокой надежностью и производительностью. Для заказа доступно 4 варианта исполнения модулей центрального процессора, к каждому из которых можно дополнительно установить до 11 модулей расширения. Все модули монтируются на стандартную DIN рейку шириной 35 мм. Программное обеспечение **ONI CICON** для контроллеров предоставляется бесплатно. Оно поставляется с широким набором готовых функциональных блоков, что позволяет существенно ускорить процесс разработки и отладки программ. Для разработки программ доступно 4 языка: LD, IL, SFC, FBD. Эти языки соответствуют международному стандарту IEC (МЭК) 61131-3. Загрузка разработанных проектов в контроллер осуществляется посредством стандартных кабелей mini USB и Ethernet, либо с помощью SD карты. Имеется возможность конфигурирования коммуникационных протоколов по каналам RS232, RS485 и Ethernet.

Микро ПЛК ONI PLR-M относятся к группе контроллеров среднего уровня. Они обладают отличным функционалом и могут быть использованы для построения малых и средних систем автоматизации. Развитые сетевые возможности микро ПЛК позволяют организовывать обмен данными в наиболее популярных промышленных сетях. Это достигается благодаря встроенным во все модули ЦПУ интерфейсам Ethernet, RS485 и RS232 с протоколами связи Modbus RTU/TCP/ASCII. Микро ПЛК ONI PLR-M позволяют организовать сбор и архивацию данных на микро SD карту. Наличие универсальных входов позволяет использовать контроллер для обработки как дискретных, так и аналоговых сигналов диапазонов 0-10 В и 0-20 мА. Интегрированный дисплей позволяет выводить информационные и аварийные сообщения, а также изменять параметры работы автоматизированного оборудования. Интегрированный WEB сервер позволяет получать информацию о состоянии оборудования через Интернет, используя стандартный WEB браузер.

Любой модуль ЦПУ ONI PLR-M позволяет подключить до 16 модулей расширения младшего семейства ONI PLR-S, что увеличивает возможности по его

использованию. Применение модулей расширения позволяет оптимизировать стоимость оборудования и унифицировать применяемые модули расширения.

Программируемые логические реле ONI PLR-S относятся к семейству контроллеров младшего уровня. Логические реле применяются преимущественно для построения автоматизированных систем малой сложности. Логические реле ONI PLR-S являются устройствами «все в одном». Уже в модуле ЦПУ есть полнофункциональный набор входов и выходов, а также клавиши управления и встроенный дисплей, позволяющие проводить настройку параметров работы оборудования без применения программаторов и персональных компьютеров. Несмотря на то, что ONI PLR-S относятся к классу логических реле, они обладают высокой надежностью и производительностью при сравнительно невысокой цене. Для заказа доступно 3 варианта модулей ЦПУ, к каждому из которых можно дополнительно установить до 16 модулей расширения, тем самым увеличив количество каналов ввода/вывода до 280.

Программирование ONI PLR-M и ONI PLR-S осуществляется с помощью программного обеспечения **ONI PLR Studio**, которое предоставляется бесплатно. Оно обладает интуитивно понятным интерфейсом и поставляется с широким набором готовых функциональных блоков и специальных программ, что позволяет существенно ускорить процесс программирования. Емкость программы составляет 1024 блока.

Программируемые логические реле ONI PLR-S имеют интерфейс RS485 с широко распространенным протоколом связи Modbus RTU. При этом они способны работать как в режиме Master, так и в режиме Slave.

Наличие протокола Modbus RTU позволяет обеспечить обмен данными с разнообразным оборудованием автоматизации, например, с панелями оператора, с частотными преобразователями и др. Также можно использовать программируемые логические реле серии ONI PLR-S в качестве станций удаленного ввода-вывода. Номенклатура контроллеров ONI представлена в таблице 5.1.

Таблица 5.1

Наименование	Базовые контроллеры PLC-S	Программируемые реле	
		Микро ПЛК PLR-M	Логические реле PLR-S
Питание	24V DC	24V DC, 220V AC	24V DC
Программное обеспечение	ONI CICON, драйвер USB	ONI PLR Studio, драйвер USB	ONI PLR Studio, драйвер USB
Состав	4 модуля ЦПУ, до 11 модулей расширения	8 модулей ЦПУ, до 16 модулей расширения	3 модуля ЦПУ, до 16 модулей расширения

5.2. Базовые контроллеры.

В номенклатуру базовых контроллеров ONI входят:

1. Модули ЦПУ;
2. Модули расширения AI;
3. Модули расширения DI;
4. Модули расширения AO;
5. Модули расширения DO;
6. Модули расширения AI/AO;
7. Модули расширения DI/DO;
8. Коммуникационные модули;
9. Аксессуары.

Базовые контроллеры имеют четыре типа модулей центрального процессора, которые можно условно разделить на две подгруппы:

- ЦПУ со встроенными дискретными входами DI и релейными выходами DO;
- ЦПУ со встроенными дискретными входами DI и транзисторными выходами DO.

Внешний вид модулей представлен на рис 5.1



Модули ЦПУ с DO^R

PLC-S-CPU-0808 / PLC-S-CPU-0806



Модули ЦПУ с DO^T

PLC-S-CPU-1616 / PLC-S-CPU-1616-SD

Рис. 5.1. Модули ЦПУ

Характеристики модулей приведены в таблице 5.2.

Таблица 5.2.

Артикул	PLC-S-CPU-0808	PLC-S-CPU-0806	PLC-S-CPU-1616	PLC-S-CPU-1616-SD
Вход	8	8	16	16
Выход	8 реле	6 реле	16x24 В DC (транзисторы)	16x24 В DC (транзисторы)

RS232C	есть	есть	есть	есть
RS485	нет	есть	нет	есть
Ethernet	нет	есть	есть	есть
SD/MMC	нет	нет	нет	есть

Расположение индикаторов, разъемов и переключателей на передней панели ПЛК показано на рис. 5.2.

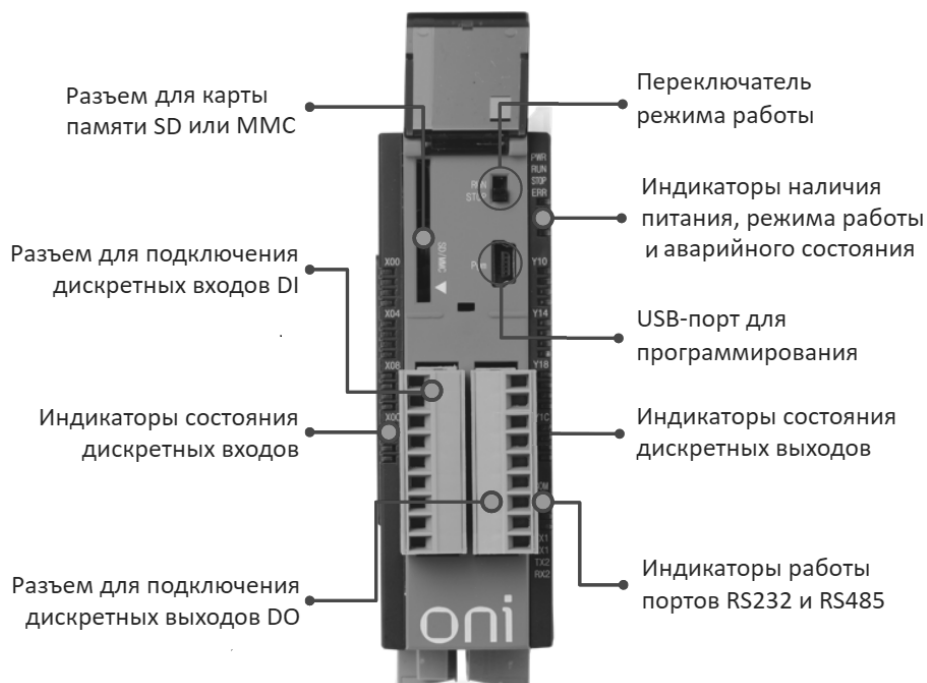


Рис. 5.2. Расположение разъемов, индикаторов и переключателей модуля ЦПУ

Основные характеристики ЦПУ:

- разрядность процессора – 32;
- напряжение питания – 24 В DC;
- поддержка модулей расширения – 11 модулей в различной комбинации;
- температура эксплуатации – (-10...+65°C)

Технические характеристики дискретных входов и выходов приведены в таблице 5.3.

Таблица 5.3

Параметр	Входы	Выходы	
		Релейный	Транзисторный
Номинальное напряжение	24 В DC	230 В AC/24 В DC	24 В DC
Номинальный ток	4 мА	Выход до 2 А (COM 5A)	Выход до 0,2 А (COM 2A)
Логическая единица	>19 В DC/3 мА	–	–
Логический ноль	<6 В DC/1 мА	–	–
Скорость реакции	<3 мс	<10 мс	<1 мс
Индикатор работы	Логич. единица – светодиод вкл.	Контакт замкнут – светодиод вкл.	Транзистор открыт – светодиод вкл.
Гальваническая развязка	оптопара	реле	оптопара
Тип входа	sink	–	–
Тип выхода	–	релейный	открытый коллектор

Для подключения цепей контроля и управления к модулям ЦПУ PLC-S-CPU-1616-х необходимо использовать дополнительные аксессуары (рис. 5.3): терминальный блок PLC-TB и кабель PLC-TB-CABLE-16.

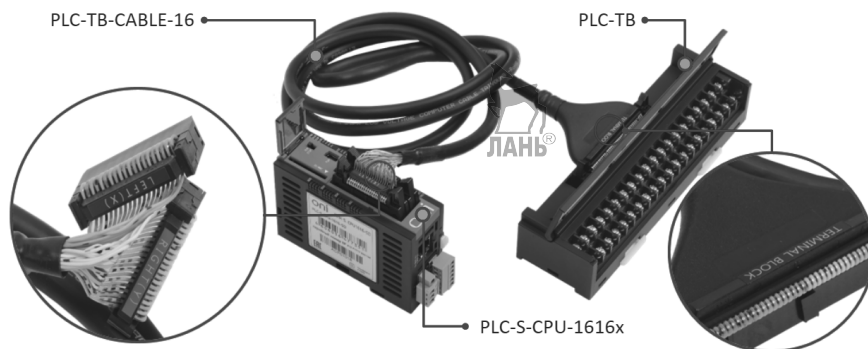


Рис. 5.3. Дополнительные аксессуары модулей ЦПУ PLC-S-CPU-1616-х

Номенклатура модулей расширения с дискретными входами/выходами содержит пять позиций, представленных в таблице 5.4.

Таблица 5.4

Артикул	PLC-S-EXD-3200	PLC-S-EXD-0032	PLC-S-EXD-1616	PLC-S-EXD-0016	PLC-S-EXD-0808
Вход	32x24 В DC	–	16x24 В DC	–	8x24 В DC
Выход	–	32x24 В DC (транзисторы)	16x24 В DC (транзисторы)	16 реле	8 реле

Номенклатура модулей расширения с аналоговыми входами/выходами содержит четыре позиции, представленные в таблице 5.5.

Таблица 5.5

Артикул	PLC-S-EXA-0400	PLC-S-EXA-0202	PLC-S-EXA-0004	PLC-S-RTD
Вход	напряжение/ ток 4x14(16) бит	напряжение/ ток 2x14(16) бит		термо- сопротивление (4)
Выход		напряжение/ ток 2x14(16) бит	ток 4x14 бит	

5.3. Микро ПЛК ONI PLR-M

Модельная линейка Микро ПЛК ONI PLR-M насчитывает восемь модулей центрального процессора. Общими конструктивными особенностями модулей ЦПУ являются:

- наличие текстового дисплея;
- наличие клавиатуры на передней панели;
- возможность подключения ко всем модулям ЦПУ модулей расширения;
- наличие у всех модулей ЦПУ интерфейсов RS232 и Ethernet;
- наличие у всех модулей ЦПУ цифровых входов. У большинства модулей есть также универсальные цифроаналоговые входы, но чисто аналоговых входов нет ни у одной модели;
- наличие у всех моделей цифровых выходов. Только у одной модели есть дополнительно аналоговые выходы.

Характеристики модулей ЦПУ микро ПЛК ONI PLR-M представлены в таблице 5.6.

Таблица 5.6

Артикул	Входы		Выходы		Интерфейс RS485	Питание
	Цифровые	Универсальные	Цифровые	Аналоговые		
PLR-M-CPU-12R00ADC	4	4	4R		1	DC
PLR-M-CPU-18R00ADC	4	8	6R		1	DC
PLR-M-CPU-18T00ADC	4	8	6T		1	AC
PLR-M-CPU-18R00AAC	12		6R			DC
PLR-M-CPU-26R00AAC	4	12	8R	2	2	AC
PLR-M-CPU-26R02ADC	4	12	8R+2T		2	DC
PLR-M-CPU-26U00ADC	16		10R		1	DC
PLR-M-CPU-26UGSMDC	4	12	8R+2T		2	AC

U – вход/выход напряжения;

T – выход транзисторный (открытый коллектор);

R – вход для подключения термосопротивлений/выход релейный.

5.4. Программируемые реле ONI PLR-S

В ассортимент программируемых логических реле ONI PLR-S входят 3 типа модулей центрального процессора, к двум из которых можно подсоединить модули расширения. Основные характеристики модулей ЦПУ приведены в таблице 5.7.

Таблица 5.7

Наименование	PLR-S-CPU-0804	PLR-S-CPU-1206	PLR-S-CPU-1410
Питание	12...24 В DC	12...24 В DC	12...24 В DC
Количество входов DI	4	6	8
Количество универсальных входов AI/DI	4 (0...10 В)	6 (0...10 В)	6 (0...10 В)
Количество	4 (реле 10 А)	6 (реле 10 А)	10 (реле 10 А)

выходов DO			
Дисплей	Нет	Есть	Есть
Часы реального времени	Есть	Есть	Есть
RS232C	Есть (требуется кабель PLR-S-CABLE-RS232)	Есть (требуется кабель PLR-S-CABLE-RS232)	Есть (требуется кабель PLR-S-CABLE-RS232)
RS485	Нет	RS485 (Modbus RTU) требуется модуль PLR-S-EMS-RS485	RS485 (Modbus RTU)
Расширение	Нет	до 16 модулей	до 16 модулей

Общий вид ЦПУ модуля PLR-S-CPU-1206 показан на рис. 5.4.

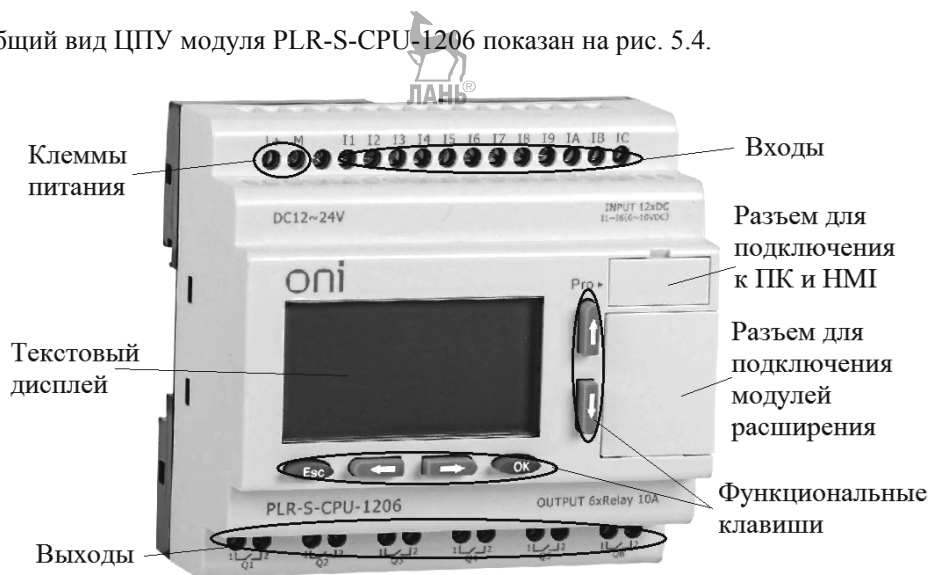


Рис. 5.4. Общий вид модуля PLR-S-CPU-1206

Дискретные входы реле ONI регистрируют сигналы логической единицы при уровне напряжения $> 8 \text{ В DC}$ и логического нуля – при уровне напряжения $< 5 \text{ В DC}$.

Технические данные модуля PLR-S-CPU-1206 представлены в таблице 5.8.

Таблица 5.8

Характеристика	Значение
Дисплей	4 строки по 16 символов

Объем памяти программы	1024 функциональных блока	
Погрешность часов реального времени	2 сек/день	
Поддержка модулей расширения	16 модулей	
Входы DI	(I1...I8)	DC 0...28,8 В; 4 Гц
	(I9...I16)	DC 0...28,8 В; 60 кГц
Входы AI (I1...I6)	DC 1...10 В, 10 бит	
Выходы DO (Q1...Q6)	до 250 В АС, 10 А (реле)	
Поддерживаемые протоколы	Modbus RTU/ASCII	
Коммутационные интерфейсы	RS232/RS485 (опция)	

Для подключения модуля ЦПУ к USB разъему компьютера используется специальный кабель-адаптер PLR-S-CABLE-USB (рис. 5.5)

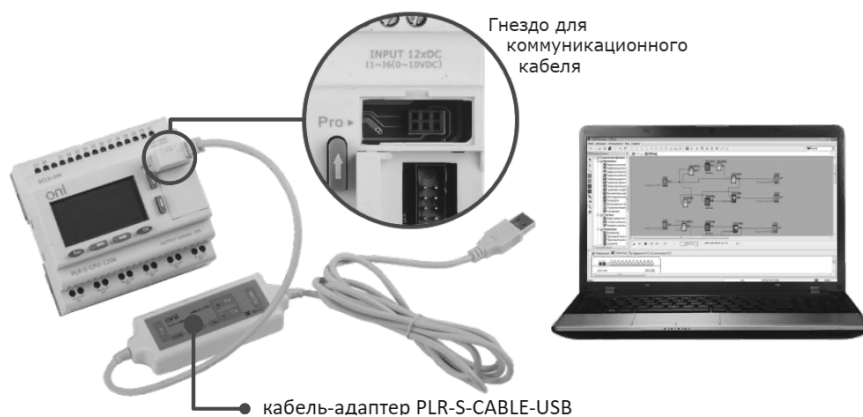
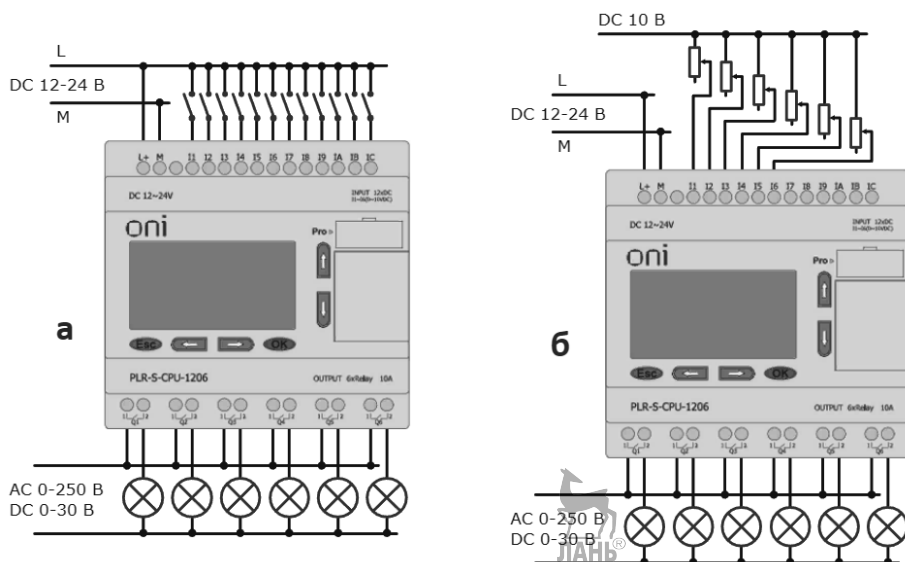


Рис. 5.5. Подключение модуля PLR-S-CPU-1206 к компьютеру

Схемы подключения внешних устройств к входам и выходам модуля ЦПУ PLR-S-CPU-1206 показаны на рис. 5.6.





Входные сигналы: а – дискретные; б - аналоговые

Рис. 5.6. Схемы подключения входов и выходов модуля ЦПУ PLR-S-CPU-1206.

Модули расширения выпускаются в 5-ти вариантах, показанных в таблице 5.9.

Таблица 5.9

Наименование	PLR-S-EMD-0808	PLR-S-EMA-0400	PLR-S-EMA-0002	PLR-S-EMA-PT100	PLR-S-EMA-RS485
Входы/выходы	дискретн. вход/выход	аналогов. вход	аналогов. выход	термосопротивление	коммуникации
Количество	входы DI	4			
	входы AI/DI	4 (0...10 В)			
	входы AI		4 (0...20 мА)		
	выходы DO	8 реле			
	выходы AI			2(0...10 В / 0...20 мА)	
	выходы PT100				3
RS485					Modbus RTU/ASCII

При монтаже модули расширения устанавливаются справа от модуля ЦПУ и могут размещаться в произвольном порядке. Однако обязательным условием при использовании коммуникационного модуля расширения PLR-S-EMA-RS485 является его установка в крайнюю правую позицию, как показано на рис. 5.7. Для правильной работы сборки ЦПУ с модулями расширения каждому модулю расширения должен быть задан уникальный адрес, начиная с первого. Адрес задается установкой микропереключателей в позицию, соответствующую желаемому адресу. Микропереключатели находятся под защитной заглушкой на внешней панели реле.

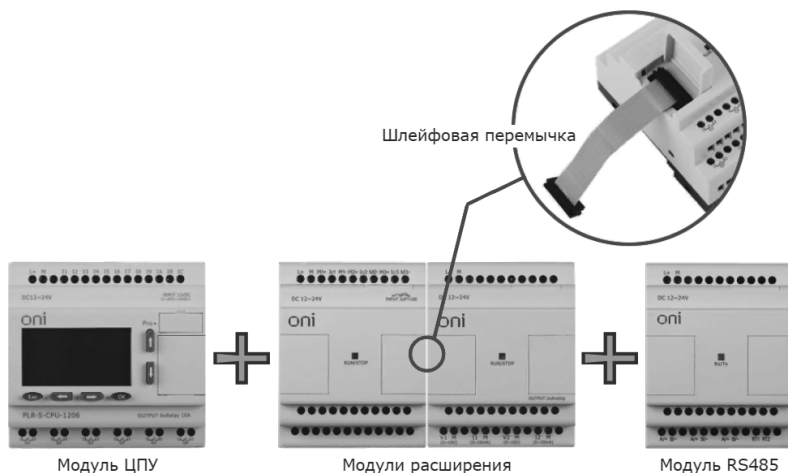


Рис. 5.7. Установка модулей расширения

5.5. Программное обеспечение ONI PLR Studio

5.5.1. Пользовательский интерфейс

Программное обеспечение ONI PLR Studio предназначено для разработки и отладки прикладных программ для логических реле ONI PLR-S и микро ПЛК ONI PLR-M. В программе используется графический язык функциональных блоков FBD. ONI PLR Studio является бесплатной программой, ее можно скачать с сайта www.oni-system.com. Программа имеет встроенный симулятор, который позволяет проводить отладку прикладных программ перед загрузкой их в контроллер. Интерфейсное окно программы показано на рис. 5.8

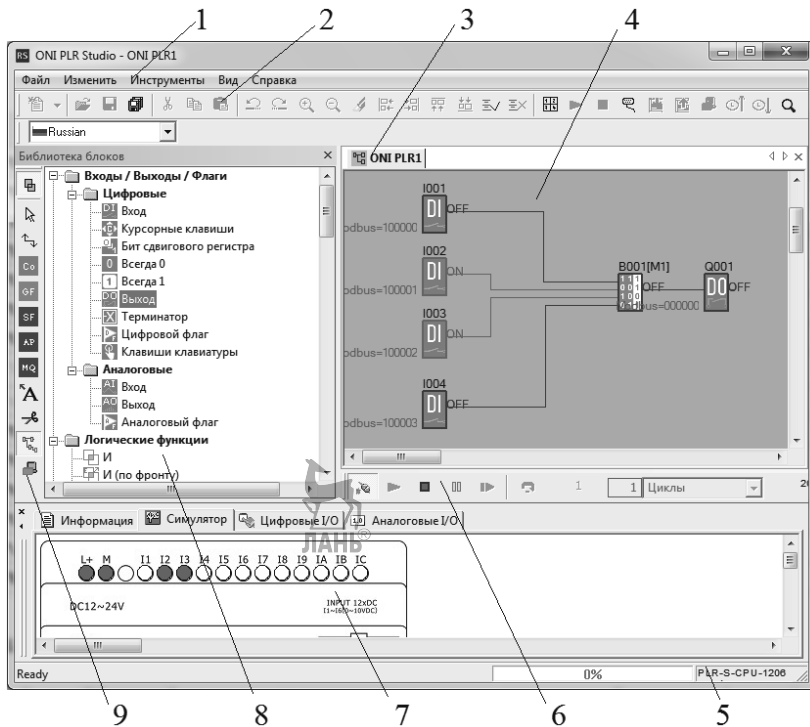


Рис. 5.8. Интерфейсное окно программы ONI PLR Studio

- | | |
|----------------------------------|--|
| 1. Главное меню. | 6. Строка управления работой симулятора. |
| 2. Основная панель инструментов. | 7. Окно информации. |
| 3. Панель закладок. | 8. Библиотека функциональных блоков. |
| 4. Рабочая область. | 9. Панель инструментов редактора. |
| 5. Строка состояния | |

Главное меню служит для доступа ко всем функциям и настройкам программы. Например, если щелкнуть по вкладке *Вид* и поставить в открывшемся окне галочку около строки *Сетка*, то на рабочей области появится сетка. Если в той же вкладке *Вид* щелкнуть по строке *Оформление*, то можно настроить цветовую гамму интерфейсного окна, в том числе цвет рабочей области. Во вкладке *Справка* имеется обширная справочная информация не только по программному обеспечению, но и по оборудованию, в том числе по модулям расширения, по монтажу оборудования, по схемам подключения входов и выходов, по созданию и редактированию проектов и т.д.

Программное обеспечение ONI PLR Studio позволяет вести одновременную работу над несколькими проектами. При этом все открытые проекты отображаются в виде отдельных окон. Для перехода между окнами служит панель закладок (поз. 3).

После того, как программа запущена на симуляцию, появляется *Строка управления работой симулятора* (поз. 6). С ее помощью программу можно запустить, остановить, поставить на паузу, запустить пошаговое исполнение программы, можно управлять часами в программе и т.д.

В окне информации (поз. 7) имеются четыре вкладки: *Информация*, *Симулятор*, *Цифровые I/O*, *Аналоговые I/O* (рис. 5.9). Вкладка *Информация* используется для вывода системных сообщений об ошибках и результатах операций в программе. На вкладке *Симулятор* моделируется выбранное оборудования и его состояние в процессе отладки проекта. Клавиши на лицевой панели, а также входы и выходы устройств позволяют моделировать входные воздействия и отслеживать состояние выходов. Если в программе имеются блоки вывода экранных сообщений, то сообщения будут выведены на экран виртуального устройства.

Вкладка *Цифровые I/O* отображает цифровые входы и выходы, задействованные в проекте, и позволяет наблюдать состояния входов и выходов при отладке программы в симуляторе.

Вкладка *Аналоговые I/O* отображает аналоговые входы и выходы, задействованные в проекте, и позволяет наблюдать состояния входов и выходов при отладке программы в симуляторе.

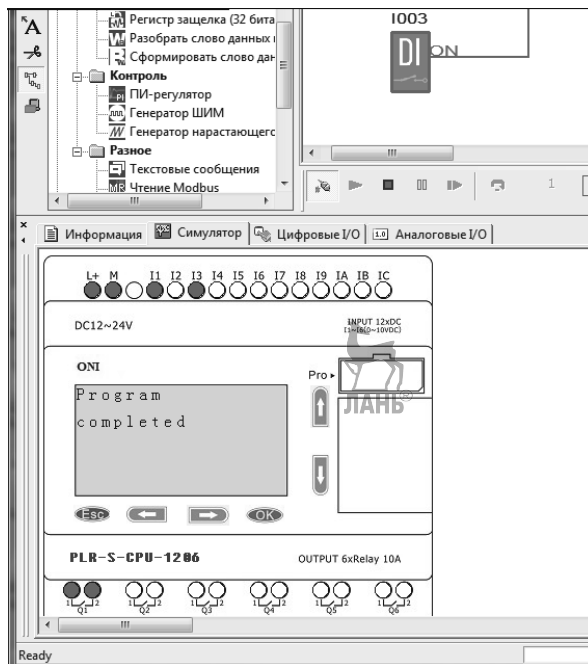


Рис. 5.9. Нижняя часть интерфейсного окна ONI PLR Studio

В окне библиотеки блоков (поз. 8) представлены все доступные функциональные блоки, применяемые при создании управляющих программ. Для удобства работы все блоки здесь сгруппированы по функциональному признаку.

5.5.2. Примеры управляющих программ в ONI PLR Studio

Подробное изучение всех блоков программы ONI PLR Studio выходит за рамки данного учебного пособия, поэтому рассмотрим только несколько примеров, поясняющих работу программы. Тем более, что работа с ONI PLR Studio практически идентична работе с LOGO! Soft Comfort, описанной в предыдущей главе.

Пример 5.1. Использование в программе блока *Текстовые сообщения*.

При запуске программного обеспечения ONI PLR Studio необходимо выбрать тип программируемого реле. Для этого надо щелкнуть левой кнопкой мыши по вкладке главного меню *Файл*, выбрать в выпадающем окне строку *Создать* и выбрать строку *Функциональная блок-схема (FBD)*. Появится окно *Информация*, показанное на рис. 5.10. На вкладке *Оборудование* этого окна можно выбрать модель программируемого реле, например, PLR-S-CPU-1206, и нажать ОК.

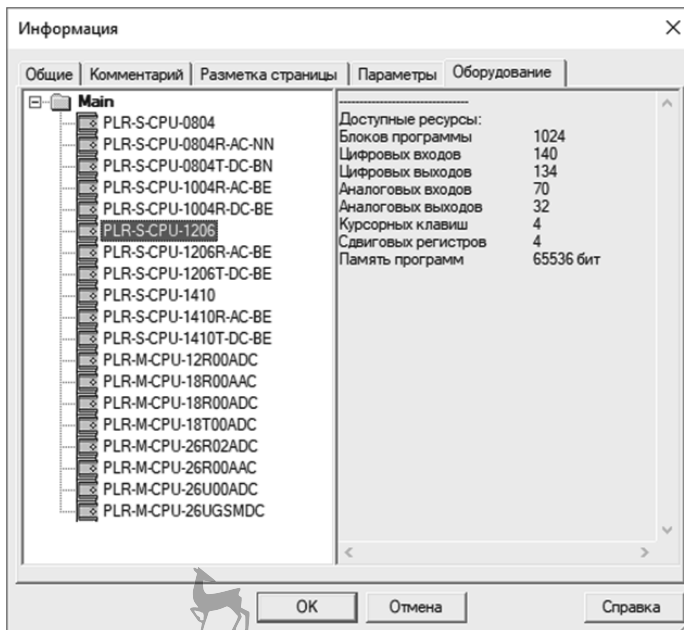


Рис. 5.10. Окно для выбора модели программируемого реле

Рассмотрим, как применить в программе блок *Текстовые сообщения*. Наберем в рабочем поле программу, показанную на рис. 5.11. Для этого

вытаскиваем на рабочее поле блоки: цифровые входы DI, цифровой выход DO, блок «И» (B005), блок *Задержка включения* (B003) и блок *Текстовые сообщения* (B001). Связываем блоки соединительными линиями.

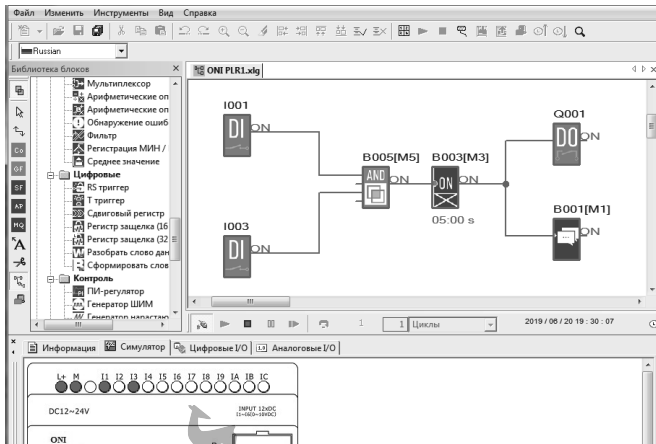


Рис. 5.11. Коммутационная программа

После того, как программа (схема) собрана, щелчком левой кнопки мыши по блоку *Текстовые сообщения*. Появится выпадающее окно, показанное на рис. 5.12. В этом окне набираем текст Program completed (программа завершена). Затем запускаем симулятор. Запуск симулятора можно выполнить либо через вкладку главного меню *Инструменты*, либо щелчком левой кнопки мыши по кнопке *Симулятор* (вторая снизу) в боковой панели инструментов, либо нажатием клавиши F3.

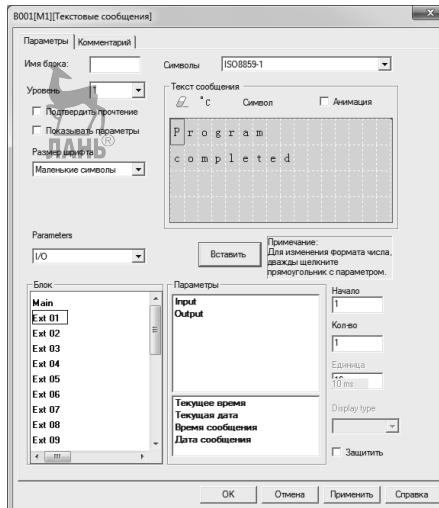


Рис. 5.12. Окно параметров блока *Текстовые сообщения*

Если в программе используются модули расширения, то в левом нижнем углу окна рис. 5.12 указывается тип этих модулей. Имеется и другой способ задания модулей, а именно, во вкладке меню *Инструменты* выбрать строку *Определить тип модулей для симулятора*.

После окончания работы программы на дисплее виртуального реле появляется набранный текст, как показано на рис. 5.13.

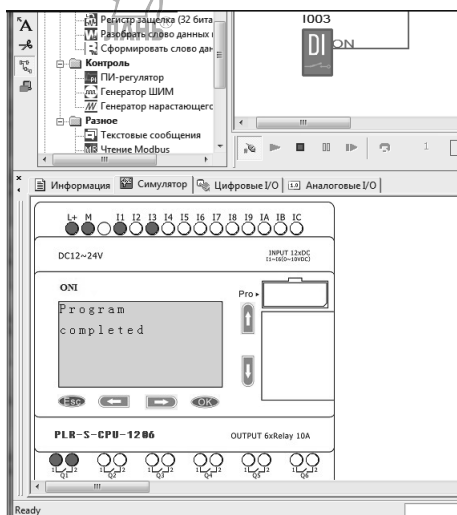


Рис. 5.13. Виртуальное реле с дисплеем

Пример 5.2. Рассмотрим логику работы блоков: *Задержка включения/выключения* (блок B001), *Задержка включения* (блок B002), *RS триггер* (блок B004). Для этого соберем схему, показанную на рис. 5.14. Кроме иллюстрации работы перечисленных блоков, данная схема позволяет организовать циклический процесс, что часто требуется в прикладных программах.

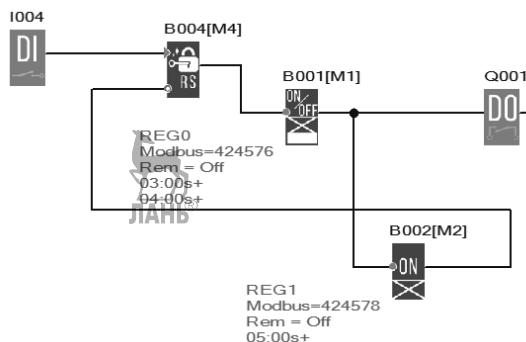


Рис. 5.14. Коммутационная программа

Установим для блока V001 задержку включения 3 с и задержку выключения 4 с, как показано на рис. 5.15. Для блока V002 установим задержку включения 5 с.

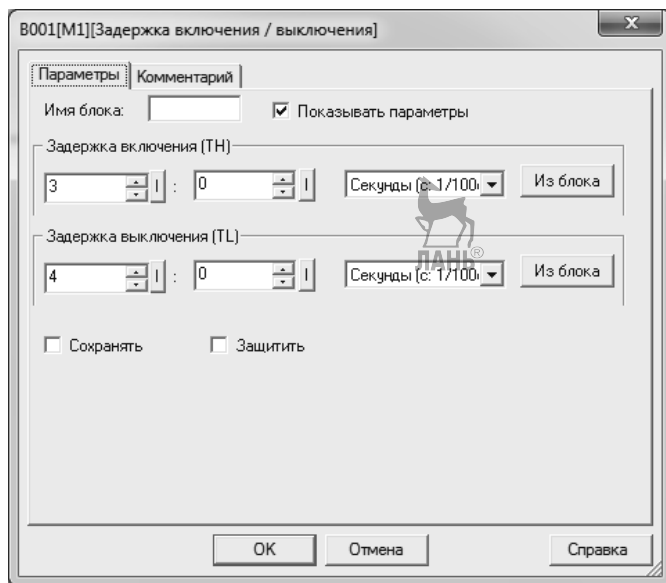


Рис. 5.15. Окно параметров блока *Задержка включения/выключения*

Алгоритм работы программы следующий. При подаче «1» на вход «S» триггера сигнал «1» проходит на блок V001, запуская отсчет времени задержки включения. Через 3 сек на выходе блока V001 появляется «1», которая проходит на выход схемы Q001 и одновременно на вход блока V002. В блоке V002 запускается отсчет времени задержки включения, и через 5 сек блок выдает «1» на вход «R» триггера. Триггер сбрасывается в «0», сигнал «0» с выхода триггера проходит на блок V001, запуская отсчет времени задержки выключения. Через 4 сек. выход блока V001 отключается, т.е. на выходе блока V001 появляется «0», который проходит на выход Q001 и одновременно через блок V002 проходит на вход «R» триггера. Поскольку на входе «S» триггера постоянно присутствует «1», то выход триггера переключается в «1» и цикл повторяется снова.

Таким образом, чтобы сработал блок задержки включения/ выключения, сигнал на него должен поступить дважды. При поступлении первого сигнала («1») запускается задержка включения, при поступлении второго сигнала («0») запускается задержка отключения.

Вход R триггера имеет приоритет над входом S. Это означает, что если на входе «S» установлена «1», то выход триггера будет переключаться между «0» и «1» всякий раз, когда между «0» и «1» будет переключаться вход «R».

Циклический процесс в схеме реализуется в том случае, если цифровой вход (I004) установлен в режим «Переключатель» и все время выдает «1» на вход «S» триггера. Щелкнем левой кнопкой мыши по входному блоку I004. Появится окно, показанное на рис. 5.16. Установим во вкладке *Симулятор* этого окна режим «НО

контакт». Это равносильно тому, что цифровой вход будет использоваться в режиме кнопки без фиксации. В этом случае программа выполнится однократно и остановится. Циклического режима не будет.

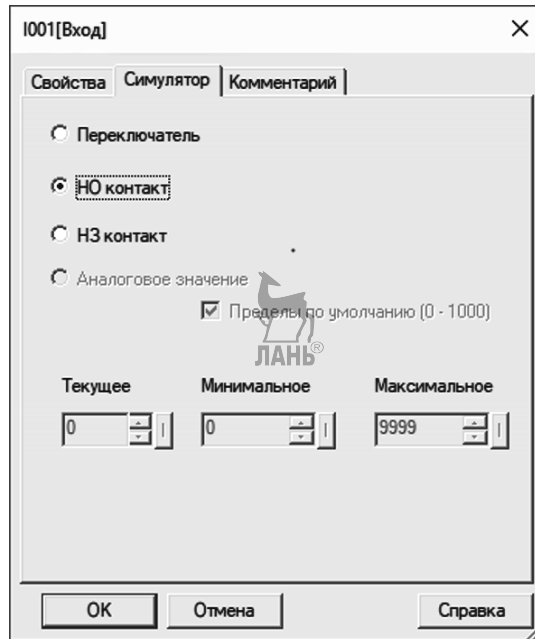


Рис. 5.16. Окно параметров блока *Цифровой вход*

Теперь можно запустить симулятор и убедиться, что схема работает в соответствии с описанным алгоритмом.

Пример 5.3. Использование в программе блока *Настраиваемая булева логика*.

В ПО ONI PLR Studio появился новый, по сравнению с LOGO! Soft Comfort, блок *Настраиваемая булева логика*. Для понимания работы этого блока соберем схему, показанную на рис. 5.17. В этой схеме использованы цифровые входы и выходы, блоки «И», «ИЛИ» и «Настраиваемая булева логика».

Щелчком левой кнопкой мыши по блоку *Настраиваемая булева логика*, появится выпадающее окно параметров блока с таблицей, показанное на рис. 5.18. Поскольку у блока задействованы все четыре входа, таблица будет иметь 16 активных строк. Можно задействовать не все входы, тогда активная область в таблице будет меньше.

Предположим, что на выходе блока должна появляться «1» всякий раз, когда на любые три входа блока подается «1». Для этого в крайнем правом столбце таблицы «Выход» ставим «1» в тех строчках, где на любых трех входах имеется «1». Это 8, 12, 14, 15 строчки таблицы. Чтобы на месте «0» появилась «1», щелкаем двойным щелчком левой кнопки мыши по «0».

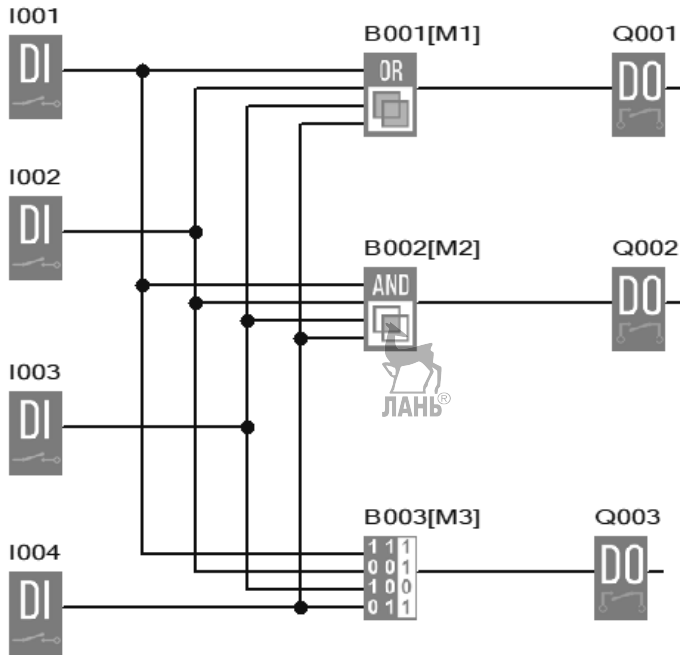


Рис. 5.17. Программа к примеру 5.3.

Теперь можно запускать симулятор. На выходе Q001, в соответствии с логикой работы блока «ИЛИ», будет появляться «1», когда на любом из четырех цифровых входов будет «1». На выходе Q002, в соответствии с логикой работы блока «И», будет появляться «1», когда на всех четырех цифровых входах будет «1». И на выходе Q003 будет появляться «1» только тогда, когда одновременно на любых трех входах будет «1». Понятно, что для блока *Настраиваемая булева логика* можно установить любую рабочую комбинацию входов и выходов, и, таким образом, этот блок является универсальным и может заменить многие логические функции.



В003[МЗ][Настраиваемая булева логика] X

Параметры | Комментарий

Имя блока: Показывать параметры

Варианты

Выход 0 если результат TRUE

Выход 1 если результат TRUE ЛАНЬ®

Индекс	Вход 1	Вход 2	Вход 3	Вход 4	Выход
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	1	1	0	0	0
5	0	0	1	0	0
6	1	0	1	0	0
7	0	1	1	0	0
8	1	1	1	0	1
9	0	0	0	1	0
10	1	0	0	1	0
11	0	1	0	1	0
12	1	1	0	1	1
13	0	0	1	1	0
14	1	0	1	1	1
15	0	1	1	1	1
16	1	1	1	1	0

OK Отмена Справка

Рис. 5.18. Окно параметров блока *Настраиваемая булева логика***Пример 5.4.** Функционирование блока «И (по фронту)».

Выход блока «И (по фронту)» переключается в состояние логической единицы на один цикл программы, только если логическая единица действует на всех входах блока одновременно, и при условии, что, по крайней мере, один вход был в состоянии логического нуля в предыдущем цикле программы. Временная диаграмма работы блока представлена на рис. 5.19.



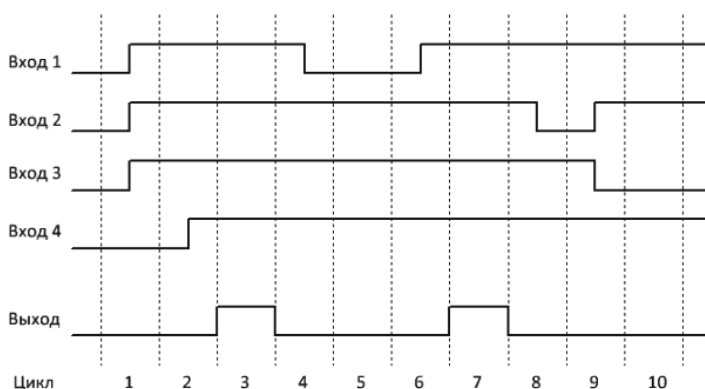


Рис. 5.19. Временная диаграмма блока «И (по фронту)»

Из диаграммы следует, что выход блока переключается в состояние «1» на 3 и 7 цикле. На 3-ем цикле все входы находятся в состоянии «1», но в предыдущем 2-ом цикле вход 4 был в состоянии «0». На 7-ом цикле все входы также в состоянии «1», но на предыдущем цикле вход 1 был в состоянии «0».

Соберем схему, представленную на рис. 5.20.

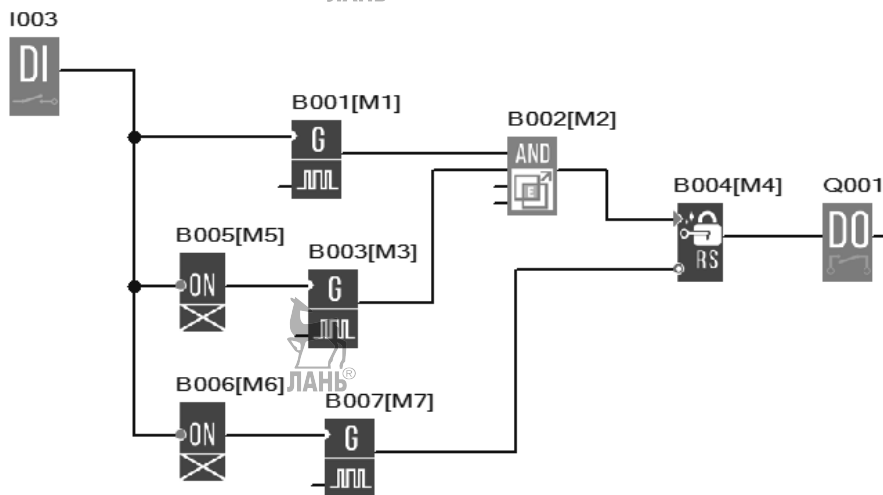


Рис. 5.20. Коммутационная программа

Схема состоит из трех генераторов прямоугольных импульсов (B001, B003, B007). На двух генераторах (B001, B003) установлена одинаковая длительность импульса (5 с) и одинаковый интервал между импульсами (5 с). На блоке задержки B005 установлена задержка 1 с, на блоке B006 установлена задержка 2 с. Кроме того, в схему входит RS триггер (B004), который нужен для того, чтобы визуализировать короткий импульс с блока «И (по фронту)». RS триггер

сбрасывается генератором В007, для которого установлен интервал между импульсами 5 с, а длительность импульса 10 мс. Диаграмма работы схемы представлена на рис. 5.21.

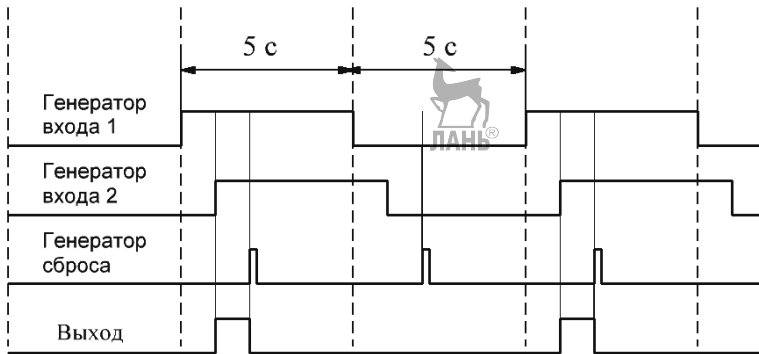


Рис. 5.21. Временная диаграмма

В результате работы программы на выходе получаем импульсы, длительность которых регулируется задержкой, устанавливаемой в блоке В006. Если вместо блока «И (по фронту)» установить блок «И», то получим уже другую программу с другой логикой работы выхода.

Пример 5.5. Импульсное регулирование температуры воздуха в помещении.

Алгоритм работы программы следующий. Имеется датчик температуры, который измеряет реальную температуру воздуха в помещении. Эта температура, в соответствии с температурой наружного воздуха, может изменяться, предположим, от -40 до $+50^{\circ}\text{C}$. Этим значениям температуры соответствует аналоговый вход А1001. Заданное (допустимое) значение температуры воздуха в помещении должно оставаться в пределах комфортных значений от 20 до 30°C . Этим значениям температуры соответствует аналоговый вход А1002. Если температура в помещении опускается ниже заданного значения, то подключается выход программы Q001, связанный с обогревателем. Если температура в помещении повышается и превышает порог отключения, то выход Q001 (обогреватель) отключается. Коммутационная программа представлена на рис. 5.22.

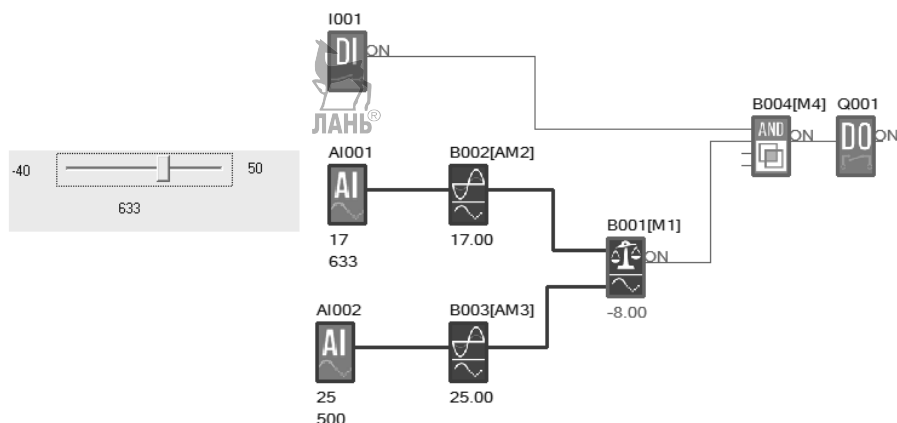


Рис. 5.22. Коммутационная программа

Программа имеет следующие блоки:

I001 – цифровой вход, который управляет включением/выключением системы;

AI001 – аналоговый вход (на него подается реальная температура);

AI002 – аналоговый вход (на него подается допустимая температура);

B002 – аналоговый усилитель. Логическое реле ONI работает с числами в диапазоне от 0 до 1000, поэтому диапазон реальных температур датчика должен быть согласован с этим диапазоном. Эту функцию выполняет аналоговый усилитель. Для согласования используется формула

$$A = k \cdot N + b$$

где A – реальное значение аналоговой величины;

N – нормализованная величина;

k – усиление;

b – смещение.

Подставляя в формулу наибольшее и наименьшее значения реальной температуры (-40 и +50) и наибольшее и наименьшее значения нормализованной величины (0 и 1000) найдем коэффициент усиления $k=0,09$ и смещение $b=(-40)$. Зададим эти коэффициенты в параметрах усилителя B002, как показано на рис. 5.23. Методика согласования реального диапазона значений аналоговой величины с диапазоном нормализованных значений подробно изложена в главе 3.

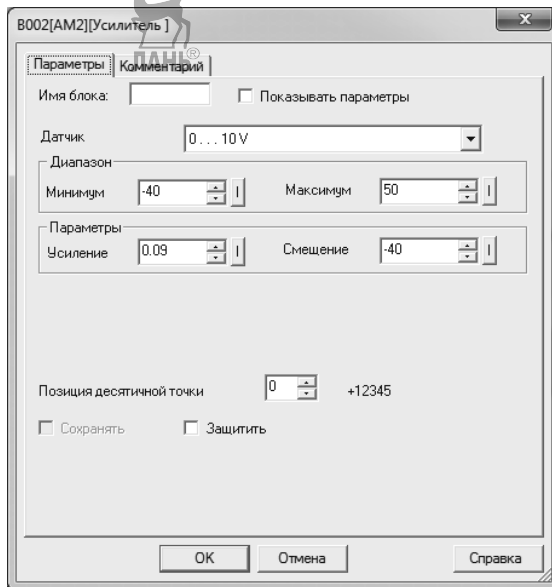


Рис. 5.23. Окно параметров усилителя V002

V003 – аналоговый усилитель, который должен согласовать диапазон допустимых температур в помещении 20...30°C с диапазоном 0...1000. Действуя по аналогии с предыдущим усилителем, находим $k=0,01$, $b=20$, как показано на рис. 5.24.

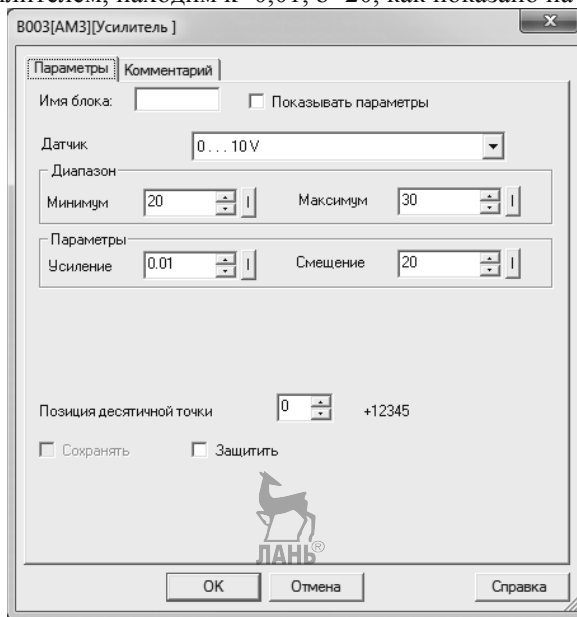


Рис. 5.24. Окно параметров усилителя V003

B001 – аналоговый компаратор. Установим в окне параметров аналогового компаратора коэффициент усиления $k=1$ и смещение $b=0$. Эти коэффициенты соответствуют диапазону изменения внутренних нормализованных чисел контроллера от 0 до 1000. Установим порог включения выхода (-40), порог выключения 2, как показано на рис. 5.25.

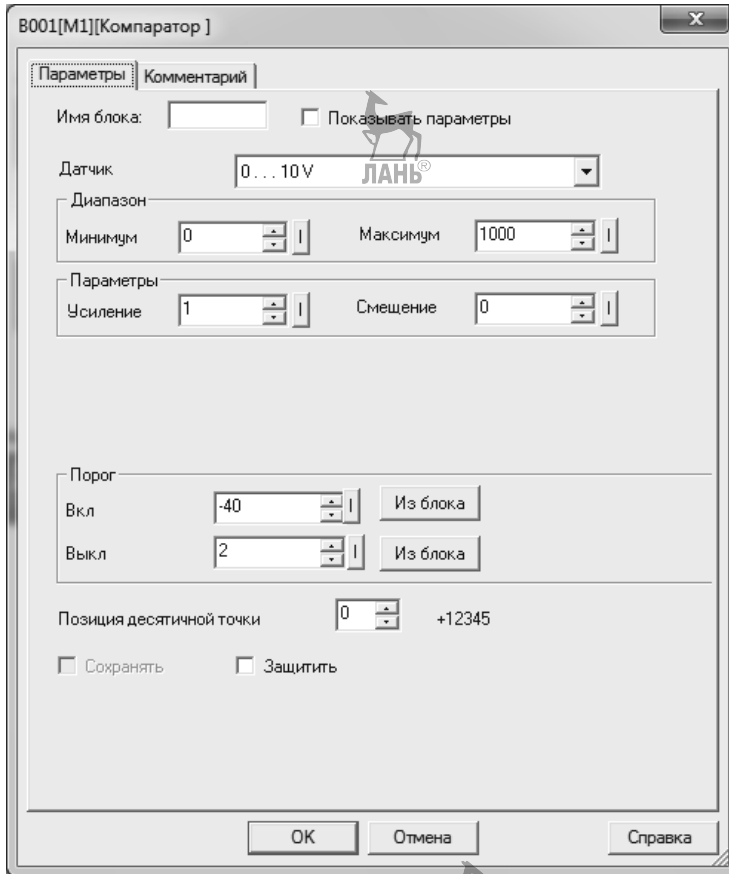


Рис. 5.25. Окно параметров компаратора

Запустим симулятор. Зададим уставку (заданное значение температуры в помещении) $+25^{\circ}\text{C}$. Для этого щелкнем левой кнопкой мыши по блоку AI002 и на появившемся движке установим значение 25. Верхняя цифра соответствует реальной температуре, нижняя цифра соответствует нормализованному значению. Подадим «1» на вход I001. Значение реальной температуры в помещении будем моделировать перемещением движка, который появляется после щелчка левой кнопкой мыши по блоку AI001. Аналоговый компаратор вычисляет разность заданного и измеренного (реального) значения температуры. Если температура снижается, и разность температур превышает порог включения, то выход Q001

устанавливается в «1» и включает нагреватель. Если измеренная температура повышается и разность ($A_x - A_y$) превышает порог отключения, то выход Q001 отключается.

Пример 5.6. Управление насосной парой.

Данная программа позаимствована из альбома примеров для контроллера ONI. Суть программы в следующем. Для повышения надежности работы и повышения долговечности, применяют не один насос, а насосную пару. При получении сигнала *Пуск* происходит запуск первого насоса. В случае, если не приходит подтверждающего сигнала с датчика потока (реле потока), то происходит запуск второго насоса. Если же и в этом случае нет сигнала с датчика потока, то формируется сигнал аварии, и дальнейшие попытки пуска насосов блокируются. В случае нормальной работы (если нажать клавишу блока I002 до истечения, заданного в блоке V016 времени) происходит циклическая смена насосов через заданный интервал времени. При этом один насос останавливается и одновременно с ним запускается второй.

Программа, управляющая работой насосов, показана на рис. 5.26. В программе используются следующие блоки:

- цифровые входы;
- цифровые выходы;
- задержка включения (блоки V001, V013, V016)
- И по фронту (блоки V008, V015);
- И (блоки V006, V007, V009, V011, V012);
- XOR (блок V002);
- RS триггер (V003, V018);
- блок НЕ;
- блок Всегда 1;
- блок Текстовые сообщения (V020, V021).

Зададим, для демонстрационных целей, время задержки на блоке V001 – 20 с, время задержки на блоке V013 – 10 с, и время задержки на блоке V016 – 30 с. При щелчке левой кнопкой мыши по цифровому входу I001, запускается первый насос (блок Q001). Если в течение 10 с не придет сигнала подтверждения с реле потока, т.е. с цифрового входа I002, (при симуляции работы программы нужно вручную щелкнуть левой кнопкой мыши по входу I002), то запустится второй насос (блок Q002), а первый насос остановится. Если и дальше не придет сигнала подтверждения с реле потока (входа I002) то сработает режим аварии, и сигнал поступит на выход Q003. Оба насоса при этом остановятся. Если же в течение 30 с от момента подачи команды *Пуск* сработает реле потока (цифровой вход I002), то схема перейдет в режим циклического переключения насосов через время, заданное в блоке V001.

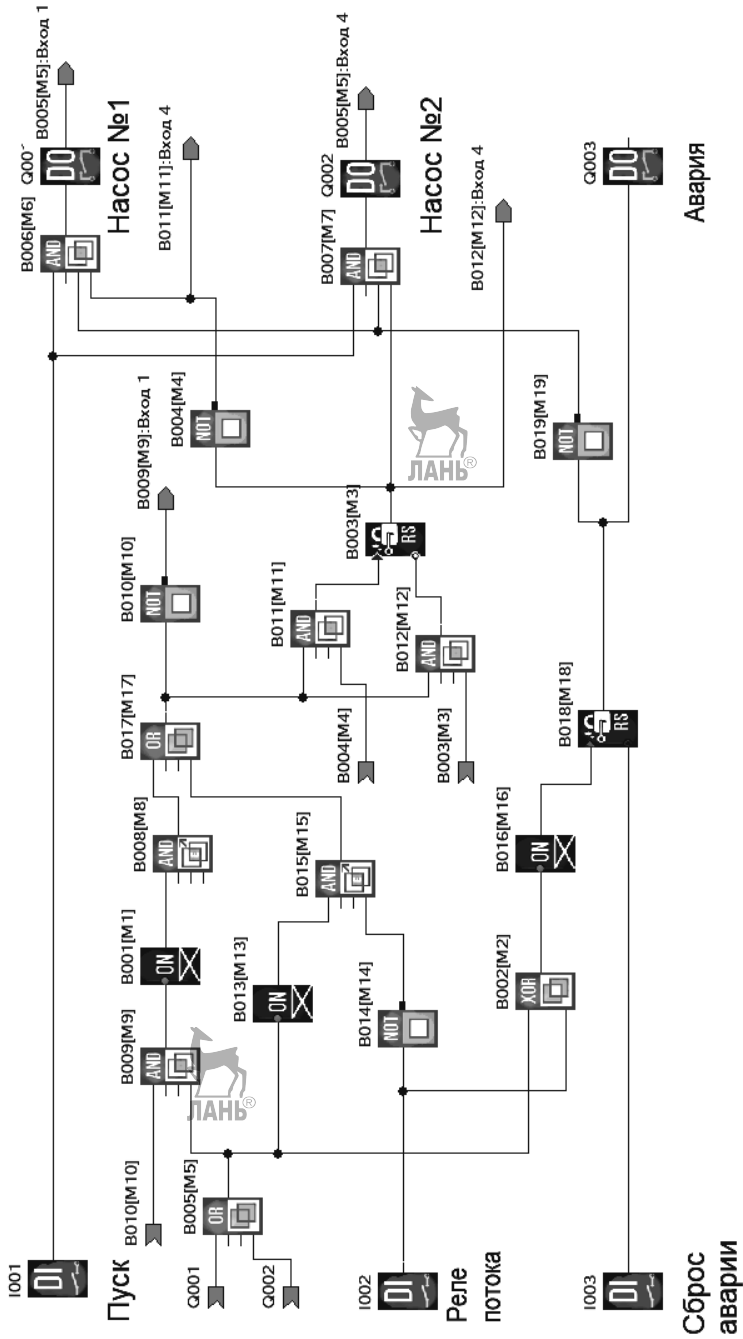


Рис. 5.26. Программа управления насосной парой

Контрольные вопросы и задания

1. На какие группы подразделяются контроллеры ONI?
2. Какое дополнительное устройство необходимо использовать, чтобы подключить электродвигатель с большим потреблением тока к выходу контроллера?
3. Для какой цели в контроллерах ONI используются интерфейсы RS232, RS485?
4. Как подключить модуль расширения к модулю ЦПУ контроллера ONI?
5. К каким входам логического реле надо подключать внешние кнопки?
6. Какое программное обеспечение используют для программирования базовых контроллеров ONI?
7. Какое программное обеспечение используют для программирования логических реле ONI?
8. Как убрать сетку с рабочего поля при составлении программы?
9. Как изменить цвет рабочего поля при составлении программы?
10. Какое усиление и смещение надо выбрать, чтобы преобразовать диапазон напряжений 0...10 В на выходе аналогового датчика в диапазон нормализованных значений 300...700 логического реле?
11. На какой вход, цифровой или аналоговый, надо подавать сигнал от датчика давления?
12. На какой вход, цифровой или аналоговый, надо подавать сигнал от датчика уровня жидкости?
13. Какие режимы работы можно устанавливать в программе для цифрового входа?
14. Каким образом в ONI PLR Studio можно управлять работой симулятора?
15. Как смоделировать в программе изменение сигнала от аналогового датчика при работе с аналоговым входом в режиме симулятора?
16. Применить программу примера 5.5 для импульсного регулирования температуры в холодильной камере с уставкой $+6^{\circ}\text{C}$ и диапазоном изменения допустимых значений уставки от $+3$ до $+9^{\circ}\text{C}$. Диапазон изменения реальных температур от 0 до 50°C . Вместо обогревателя должен использоваться холодильный агрегат, включающийся при превышении температуры в холодильной камере выше порогового значения.
17. Составить программу, состоящую из цифровых входов, цифрового выхода, блока *Настраиваемая булева логика*, моделирующей работу логической функции «ИЛИ-НЕ».
18. Найти лишние блоки, которые не влияют на алгоритм работы программы, показанной на рис. 5.27.

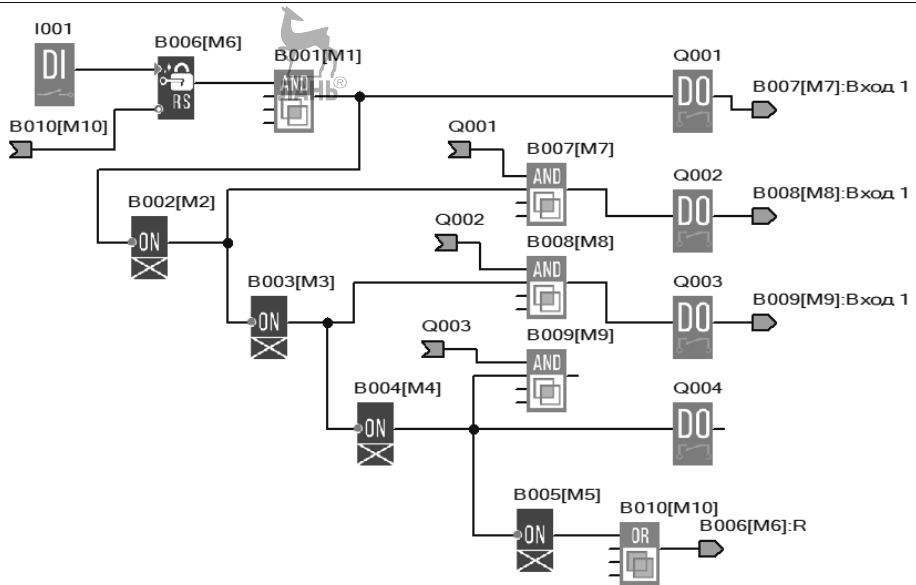


Рис. 5.27. Программа с лишними блоками



ГЛАВА 6. ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ОВЕН

6.1. Общая характеристика программируемых логических контроллеров

Полное представление о технических характеристиках того или иного контроллера можно получить после изучения технической документации (паспорт прибора или Datasheet, руководства по эксплуатации, руководства пользователя и т.д.). Приводимая здесь информация служит лишь для того, чтобы сориентироваться в многочисленных моделях и сделать первоначальный выбор устройства.

Линейка программируемых логических контроллеров ОВЕН включает в себя несколько модификаций.

- Контроллеры для локальных систем автоматизации с человеко-машинным интерфейсом HMI (Human Machine Interface) – **ОВЕН ПЛК63 / ПЛК73**.

Контроллеры предназначены для применения в системах управления климатическим оборудованием в сфере ЖКХ, в системах управления оборудованием хранения и упаковки, в системах управления промышленными станками и механизмами, в системах HVAC. Системы HVAC (Heating, Ventilation, Air Conditioning) – системы отопления, вентиляции и кондиционирования. Отличительные особенности данной линейки контроллеров:

- возможность управления технологическим процессом непосредственно с лицевой панели контроллера;
- широкие возможности самодиагностики (контроль работы датчиков, правильности пользовательских программ, контроль зависания ПЛК);
- возможность создания пользовательских программ с привязкой к реальному времени (в контроллере имеются встроенные часы реального времени с автономным питанием);
- поддержка технологии OwenCloud.

Конструктивно контроллеры выполнены либо для крепления на DIN-рейку (ПЛК63), либо для крепления на лицевую панель щита (ПЛК73).

- Контроллеры для малых систем автоматизации – **ОВЕН ПЛК100 / ПЛК150 / ПЛК154**.

Линейка контроллеров предназначена для построения распределенных систем управления и диспетчеризации с использованием как проводных, так и беспроводных технологий в системах HVAC в сфере ЖКХ, в области управления малыми станками и механизмами, в области автоматизации технологических

процессов в сфере производства строительных материалов и др. Отличительные особенности данной линейки:

- высокая производительность. Контроллеры имеют процессор ARM9 RISC-архитектуры, 8 Мбайт оперативной памяти; 4 Мбайт постоянной Flash-памяти;
- широкие возможности самодиагностики (контроль работы датчиков, правильности пользовательских программ, контроль зависания ПЛК);
- большое количество встроенных интерфейсов: последовательные RS-232/RS-485, а также порт Ethernet;
- возможность работы со стандартными (Modbus RTU/ASCII/TCP, DCON) и нестандартными протоколами обмена;
- возможность создания пользовательских программ с привязкой к реальному времени;
- наличие на борту дискретных и аналоговых входов для подключения широкого спектра датчиков (термосопротивлений, термопар, резистивных датчиков и др).

• Моноблочные контроллеры для средних систем автоматизации – **ПЛК110 / ПЛК160.**

Контроллеры данной линейки рекомендуются для торгового оборудования; линий дерево- и металлообработки; станков по дозированию, упаковке и переработке; для котельных установок средней мощности; систем вентиляции и теплоснабжения; систем водоподготовки. Отличительные особенности данной линейки:

- мощные вычислительные ресурсы и большой объем памяти;
- широкие коммуникационные возможности: до 4 последовательных портов RS-232, RS-485, наличие порта Ethernet для включения в локальные или глобальные сети верхнего уровня, поддержка протоколов обмена Modbus (RTU, ASCII, TCP), ОВЕН, DCON, возможность подключения внешних устройств с нестандартными протоколами;
- наличие Flash-памяти для архивирования данных;
- до 60 точек ввода/вывода «на борту» контроллера;
- подключение счетчиков и энкодеров;
- широкие возможности самодиагностики (контроль работы датчиков, правильности пользовательских программ, контроль зависания ПЛК);
- встроенные часы для создания систем управления с учетом реального времени;
- поддержка OwenCloud.

• Коммуникационные контроллеры – **ПЛК304 / ПЛК323.**

Контроллеры данной линейки предназначены для распределенных систем управления и диспетчеризации. Контроллеры рекомендуется использовать в автоматизированных системах контроля и учета энергоресурсов, в системах телеметрии, в устройствах сбора и передачи данных. Контроллеры позволяют объединить устройства с различными интерфейсами и протоколами связи в единую коммуникационную сеть. Отличительные особенности данной линейки:

- открытая архитектура на основе ОС Linux;
- большое количество последовательных интерфейсов;
- порт Ethernet для включения в локальные или глобальные сети верхнего уровня;
- возможность работы по нестандартным протоколам, что позволяет подключать такие устройства, как электрические, газовые и водосчетчики, считыватели штрих-кодов и т.д.;
- расширенный температурный диапазон: $-25 \dots +70 \text{ }^\circ\text{C}$;
- USB-host для подключения внешних накопителей информации;
- встроенный разъем для SD-карт памяти объемом до 32 Гб;
- возможность создания пользовательских программ с привязкой к реальному времени – встроенные часы реального времени (RTC);
- возможность встраивания в вертикально интегрированные SCADA системы;
- поддержка OwenCloud.

На рис. 6.1 показаны модели моноблочных контроллеров с дискретными входами/выходами на борту – ПЛК110-MS4 [M02] с исполнительной средой MasterSCADA 4D. MasterSCADA 4D – кросс-платформенная SCADA-система, позволяющая создавать проекты автоматизации с использованием языков стандарта МЭК61131-3 с динамическим web-интерфейсом. Масштабируемая векторная графика дает возможность управлять технологическим процессом с компьютера, планшета или смартфона с любыми операционными системами. Контроллеры оптимальны для построения систем автоматизации среднего уровня и распределенных систем управления.



Рис. 6.1. Общий вид контроллеров ПЛК110-30 и ПЛК110-60

В таблице 6.1 приведены технические характеристики контроллеров ПЛК110-30 и ПЛК110-60 с MasterSCADA 4D.

Таблица 6.1

Параметр	ПЛК110-30 [M02]	ПЛК110-60 [M02]
Габаритные размеры, мм	(140×114×83)±1	(208×114×83)±1
Центральный процессор	RISC-процессор Texas Instruments Sitara AM1808	
Объем энергонезависимой памяти (тип памяти)	128 Мб для хранения файлов и архивов	
Количество цифровых входов	18	36
Количество выходов (электромагнитное реле или транзисторный n-p-n ключ)	12	24
RS-485	2	2
RS-232	1	1
RS-232-Debug	1	1
Ethernet 100 Base-T	1	1
USB B - USB-RNDIS	1	1
Напряжение питания: =24 В; ~ 220 В	от 9 до 30 В постоянного тока при $T > \text{минус } 20\text{ }^{\circ}\text{C}$, от 9 до 26 В постоянного тока при $\text{минус } 40\text{ }^{\circ}\text{C} > T > \text{минус } 20\text{ }^{\circ}\text{C}$ (номинальное 12 или 24 В); от 90 до 264 В переменного тока, либо постоянного тока (номинальное 120/230 В)	
Подключаемые входные устройства	<ul style="list-style-type: none"> – коммутационные устройства (контакты кнопок, выключателей, герконов, реле и т.п.) – трехпроводные датчики, имеющие на выходе транзистор n-p-n или p-p-n – типа с открытым коллектором; – дискретные сигналы с напряжением до 30 В 	
Дополнительное оборудование	<ul style="list-style-type: none"> – Часы реального времени с собственным батарейным питанием. – Встроенный источник выдачи звукового сигнала; – Трехпозиционный переключатель на передней панели контроллера. 	

Программирование логических контроллеров осуществляется с помощью интегрированной среды программирования Codesys, в которую в соответствии со стандартом МЭК (IEC) 61131-3, входит пять языков программирования.

6.2. Общая характеристика логических реле (ПЛР) ОВЕН

Разновидностью программируемых логических контроллеров ОВЕН являются программируемые логические реле ОВЕН, которые применяются в системах малой и средней автоматизации. Программируемые реле отличаются от полноценных ПЛК меньшим числом каналов ввода-вывода, меньшим объемом памяти программ, невозможностью исполнения сложных математических операций, зачастую моноблочной конструкцией. Коммуникационные возможности ОВЕН ПЛР ограничены, как правило, одним интерфейсом для загрузки программы или связи с АСУ верхнего уровня. Основные типы интерфейсов: RS-485 или Ethernet.

Алгоритм работы программируемого реле формируется непосредственно пользователем, что делает прибор универсальным и дает возможность широко использовать его в различных областях промышленности, сельском хозяйстве, ЖКХ и на транспорте.

Линейка программируемых логических реле включает в себя 4 модели: **ПР110, ПР114, ПР100, ПР200**. Общий вид логического реле ПР200 приведен на рис. 6.2.



Рис. 6.2. Общий вид логического реле ПР200

Программирование ПЛР не требует специальных навыков, поскольку осуществляется с помощью интуитивно понятной среды программирования собственной разработки Owen Logic. Линейка программируемых реле ОВЕН широко используется для построения локальных автоматизированных систем управления, например:

- управление наружным и внутренним освещением, освещением витрин;
- управление технологическим оборудованием (насосами, вентиляторами, компрессорами, прессами);
- системы управления воротами, дверями, жалюзи;
- системы управления конвейерами.

управление подъемниками, парковочными автоматами и т.д.

Основные особенности:

различные виды исполнения (по питанию, по типу входов/выходов, по количеству входов/выходов);

компактный корпус на DIN-рейку;

широкий климатический диапазон: -20...+55 °С;

наличие часов реального времени (в зависимости от модификации);

возможность интеграции в сети RS-485;

простая, интуитивно понятная среда программирования с широкими возможностями;

возможность создания и отладки проекта без прибора.

Технические характеристики логических реле ОВЕН представлены в табл.6.2

Таблица 6.2

	ПР110	ПР114	ПР100	ПР200
Каналы ввода / вывода				
Количество дискретных входов	8 или 12	8 (12)	До 12	8
Количество аналоговых входов	Нет	До 4 (переключаемый AI/DI)	До 4 (переключаемый AI/DI)	До 4 (переключаемый AI/DI)
Тип выходных сигналов	–	0...10 В; 4...20 мА; дискретный	0...10 В; 4...20 мА; дискретный	0...10 В; 4...20 мА; 0...4 кОм; дискретный
Количество дискретных выходов	4 или 8 (P)	4 – 8 (P)	До 8 (P)	До 8 (P или K)
Количество аналоговых выходов	Нет	До 4 (P, K, C, T, И, У по заказу)	Нет	2 (И – 4...20 мА) или 2 (У – 0...10 В)
Модули расширения	Нет	Нет	Нет	2
Питание прибора				
Напряжение питания	=24 В или ~220 В			
Программирование				
Среда программирования	Owen Logic			
Количество функциональных блоков, среднее	63	500		

Время цикла, мс, не менее	3	1	1	
Интерфейс программирования	Программирование возможно только при использовании комплекта для программирования ПР-КП10 или ПР-КП20.		USB	miniUSB
Конструкция				
Тип крепления	На DIN-рейку или настенное			
ЖК дисплей	Нет	Нет	Нет	Есть
Габариты, мм	110×73×63 110×73×96	110×73×96		123×90×58
Коммутационные интерфейсы				
Интерфейс	RS-485 (модуль ПР-МИ485)		–	2xRS485 (плата ПР-ИП485)
Протокол	Modbus RTU/ASCII		–	Modbus RTU/ASCII

Технические характеристики выходных устройств логических реле ОВЕН представлены в табл. 6.3.

Таблица 6.3

Обозначение	Тип выходного устройства	Электрические характеристики
Р	Электромагнитное реле	Максимальный ток нагрузки – 1 А (для ПИД-регуляторов), 8 А (для сигнализации) при 220 В 50...60 Гц или 30 В пост. тока
К	Транзисторная оптопара структуры прп-типа	Максимальный ток нагрузки – 400 мА при 60 В пост. тока
С	Симисторная оптопара	Максимальный ток нагрузки – 50 мА при 240 В (постоянно открытый симистор) или 0,5 А (симистор включается с частотой не более 50 Гц и тимп. = 5 мс)
И	Цифроаналоговый преобразователь «параметр–ток 4...20 мА»	Номинальное сопротивление нагрузки – 0...1000 Ом, напряжение питания 10...30 В пост. тока
У	Цифроаналоговый преобразователь «параметр–напряжение 0...10 В»	Номинальное сопротивление нагрузки – не менее 2 кОм, напряжение питания 15...32 В

Т	Выход для управления твердотельным реле	Выходное напряжение 4...6 В, максимальный выходной ток 50 мА
СЗ	Три симисторные оптопары (для трехфазной нагрузки)	Максимальный ток нагрузки – 50 мА при 240 В (постоянно открытый симистор) или 0,5 А (симистор включается с частотой не более 50 Гц и тимп. = 5 мс)

6.3. Монтаж электрических цепей

Подключение логических реле к сети переменного тока следует осуществлять от сетевого фидера, не связанного непосредственно с питанием мощного силового оборудования, чтобы избежать повреждений ПЛР от чрезмерно больших токов. Во внешней цепи рекомендуется установить выключатель, обеспечивающий отключение прибора от сети. Для обеспечения надежности электрических соединений рекомендуется использовать кабели с медными многопроволочными жилами, сечением не более 0,75 мм². Релейные выходы логического реле можно применять для коммутации цепей переменного тока напряжением не более 250 В и рабочим током не более 5 А.

На рис. 6.3 приведены схемы подключения дискретных входов логического реле ПР200.

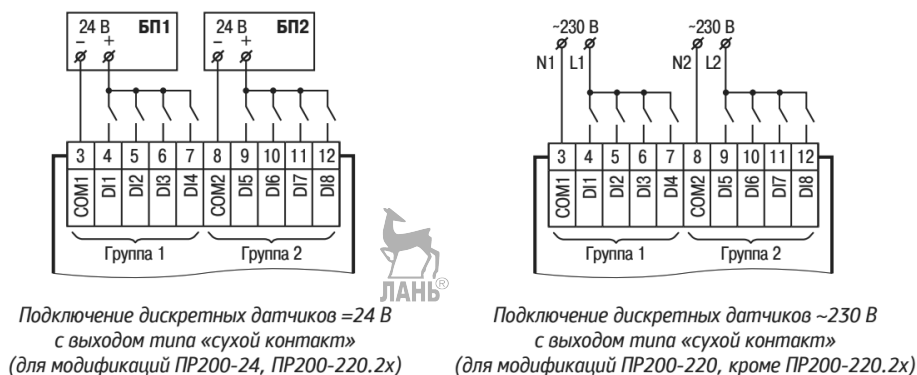
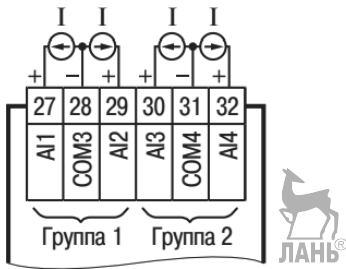
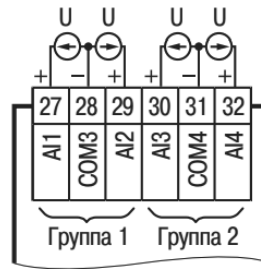


Рис. 6.3. Подключение дискретных входов реле ПР200

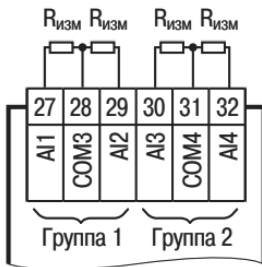
На рис. 6.4 приведены схемы подключения аналоговых входов логического реле ПР200.



Подключение активных датчиков с выходом «ток 4...20 мА» (встроенное шунтирующее сопротивление $R_{ш}$)



Подключение активных датчиков с выходом «напряжение 0...10 В»

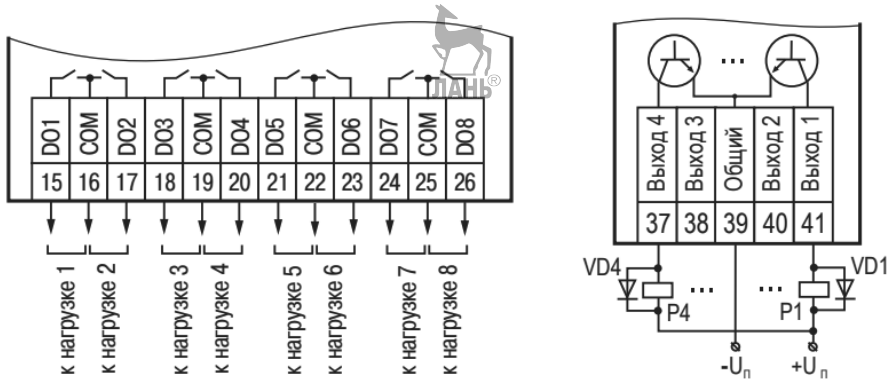


Подключение сопротивления 0...4000 Ом

Рис. 6.4. Подключение аналоговых входов реле ПР200.

Аналоговые входы могут также работать в дискретном режиме. Тип аналогового входа определяется установкой перемычек на плате и выбором типа в среде OwenLogic.

На рис. 6.5 приведены схемы подключения дискретных выходов логического реле ПР200.

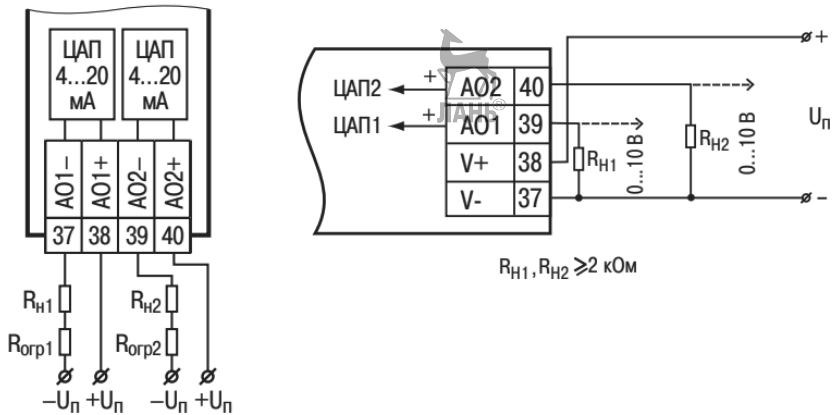


Подключение дискретных выходов типа P

Подключение дискретных выходов типа К
 ПР200-х.5, ПР200-220.25

Рис. 6.5. Подключение дискретных выходов реле ПР200

На рис. 6.6 приведены схемы подключения аналоговых выходов логического реле ПР200.



Подключение аналоговых выходов типа И
 ПР200-х.2, ПР200-220.22

Подключение аналоговых выходов типа У
 ПР200-х.4, ПР200-220.24

Рис. 6.6. Подключение аналоговых выходов реле ПР200

6.4. Среда программирования OWEN LOGIC

6.4.1. Пользовательский интерфейс

Для программирования логических реле ОВЕН используется среда программирования Owen Logic. Интерфейсное окно Owen Logic представлено на рис. 6.7.

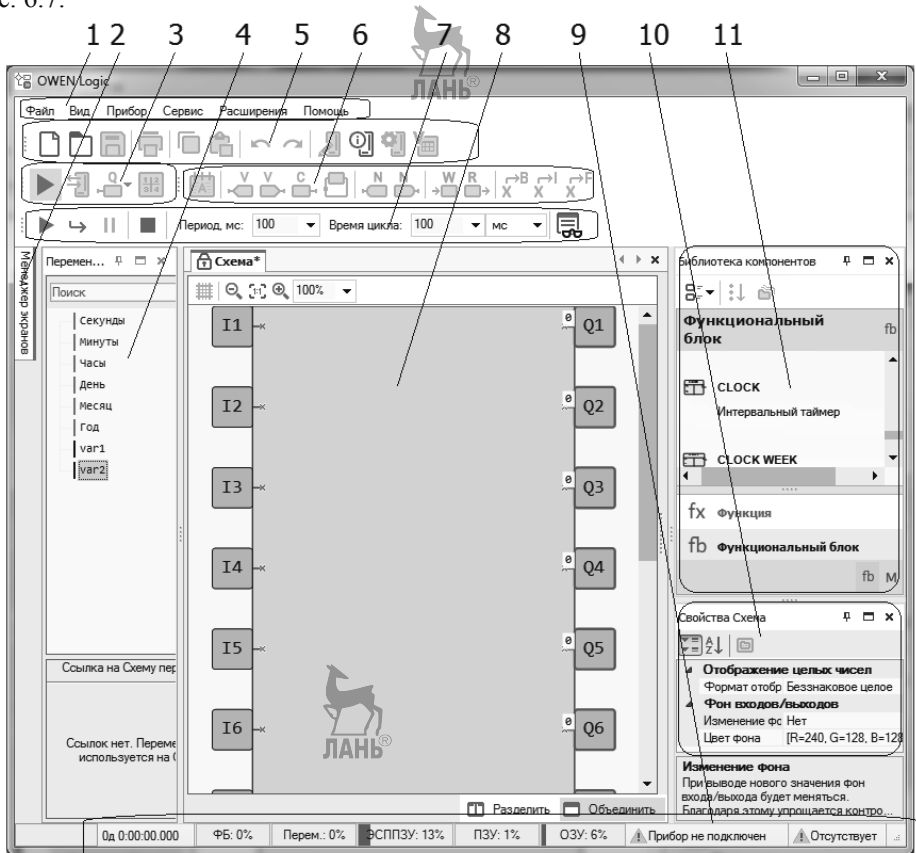


Рис. 6.7. Интерфейсное окно Owen Logic

- 1 – Главное меню.
- 2 – Менеджер экранов (присутствует только для тех моделей ПЛР, у которых есть дисплей).
- 3 – Панель отладки.
- 4 – Панель *Переменные*.
- 5 – Панель инструментов.
- 6 – Панель вставки.
- 7 – Панель симуляции (появляется в режиме симуляции).

8 – Рабочая область (холст).

9 – Строка состояния.

10 – Панель *Свойства*.

11 – Панель *Библиотека компонентов*.

Включить/выключить отображение панелей можно в главном меню *Вид*. На панели *Свойства* отображаются и редактируются:

параметры элементов программы;

размеры холста;

свойства входов и выходов;

комментарии.

Если элемент не выбран, то на панели *Свойства* отображаются параметры холста: ширина и высота, которые можно здесь изменить. Для отображения параметров элемента следует нажать на него левой кнопкой мыши.

Чтобы разместить панели *Свойства*, *Библиотека компонентов*, *Переменные* в определенном месте интерфейсного окна (например, как на рис. 6.7), надо навести курсор на заголовок в верхней части панели, и нажать левую кнопку мыши. При этом появятся маленькие квадратики с треугольными стрелочками и большой синий прямоугольник, как показано на рис. 6.8. Далее, не опуская левую кнопку мыши, совместить курсор с тем квадратиком, который указывает направление возможного перемещения панели. И тогда панель займет соответствующее положение в интерфейсном окне: сверху, снизу, справа сверху, справа снизу и т.д.

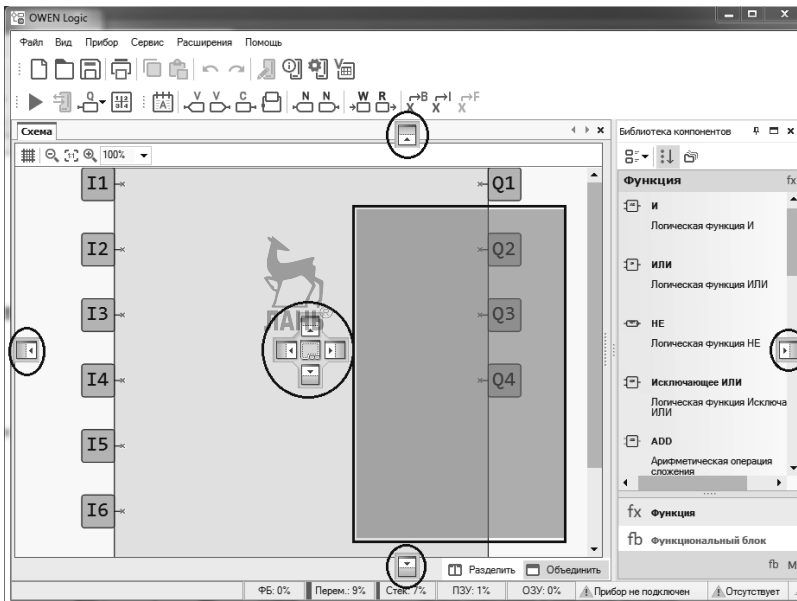


Рис. 6.8. Выбор места размещения панелей в интерфейсном окне

С левой стороны рабочей области расположены входы: Ix – дискретные, AIx – аналоговые. С правой стороны расположены выходы: Qx – дискретные (реле), AOx – аналоговые, Fx – светодиоды. Входы и выходы можно перемещать по вертикали методом drag&drop (т.е. путем захвата левой кнопкой мыши). Цифры в обозначениях входов и выходов соответствуют номерам физических входов и выходов прибора.

Под заголовком "Схема" в верхней части рабочей области располагаются кнопки: *Включить/Выключить сетку*. Размещаемые в рабочем поле компоненты и линии связи привязываются к сетке.

Уменьшить масштаб. Уменьшение масштаба на 10% от первоначального.

Оригинальный размер. Возврат к исходному масштабу.

Увеличить масштаб. Увеличение масштаба на 10% от первоначального.

Задать произвольный масштаб можно с помощью крайней правой кнопки.

В нижней части рабочей области находятся кнопки "Разделить" и "Объединить". Кнопка "Разделить" используется для одновременного отображения двух областей одной схемы в разных окнах, кнопка "Объединить" – для отображения схемы в одном окне.

В нижней части панели *Библиотека компонентов* показаны заголовки разделов: *Функции*, *Функциональные блоки*, *Макросы*. Нужно выбрать необходимый заголовок раздела для отображения его содержимого. На рис. 6.9 выбран и отображается раздел *Функция*.



Рис. 6.9. Отображение раздела *Функции*

Если щелкнуть левой кнопкой мыши по кнопке *Группировка по папкам* в верхней части *Библиотеки компонентов*, то все функции сгруппируются по папкам, как показано на рис. 6.10. Папка открывается двойным щелчком мыши.

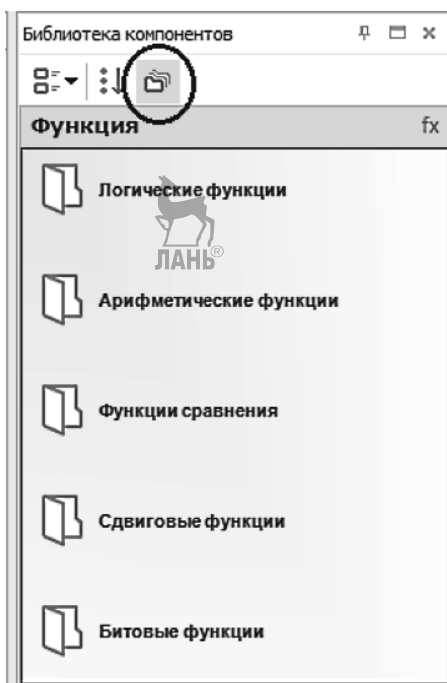


Рис. 6.10. Группировка функций по папкам

С помощью вкладки *Менеджер экранов* можно настроить вывод информации из выполняемой коммутационной программы на дисплей логического реле. Для этого надо зайти в *Редактор экранов*, который открывается по двойному щелчку на изображении экрана в *Менеджере экранов*.

На рис. 6.11 показана *Статусная строка*. В статусной строке выводится информация о доступных ресурсах ПЛР и его подключении.

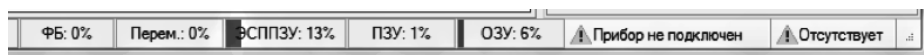


Рис. 6.11. Статусная строка

При подключенном приборе в статусной строке отображается информация:

ФБ - доступное количество функциональных блоков.

Перем. - доступное количество переменных.

ЭСПЗУ - доступное количество энергонезависимой памяти для Retain переменных.

ПЗУ - доступная память ПЗУ в процентах от общего объема: « 0...100 %».

ОЗУ - доступная память ОЗУ в процентах от общего объема: « 0...100 %». Чем больше используется в программе блоков и функций, тем больше требуется памяти. OWEN Logic автоматически рассчитывает доступную память и в случае критического значения выводит соответствующее предупреждение.

ПРх-х (на рис. 6.11 отсутствует) – информация о наличии связи между программным обеспечением OWEN Logic и логическим реле, при наличии связи указывается модель подключенного ПЛР.
СОМх (на рис. 6.11 отсутствует) – номер выбранного пользователем порта.

6.4.2. Создание нового проекта

Чтобы создать новый проект, надо выбрать *Файл > Новый проект*. В открывшемся окне (рис. 6.12) следует выбрать модель ПЛР, для которой создается проект. После этого нажать "ОК".

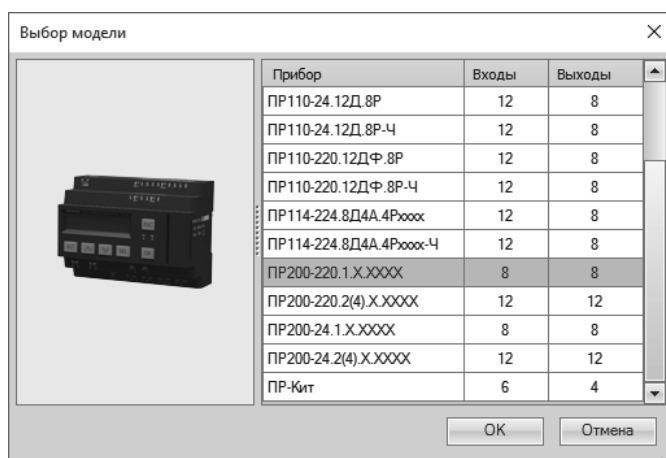


Рис. 6.12. Выбор модели логического реле

В главном окне программы отобразится рабочее поле (холст) и станут активными статусная строка и панели интерфейсного окна. При наличии у выбранного прибора дисплея отобразится вкладка *Менеджер экранов*, где можно настроить вывод информации на дисплей. Далее можно приступить к созданию коммутационной программы на языке FBD с помощью функций и функциональных блоков панели *Библиотека компонентов*.

Моделирование работы коммутационной программы и ее отладка проводятся в режиме симуляции. После того, как коммутационная программа прошла процесс отладки и ее работоспособность проверена, она загружается в прибор. Но предварительно прибор надо подключить к компьютеру. При подключении ПЛР к компьютеру через порт USB необходимо установить драйвер USB с CD-диска, входящего в комплект поставки прибора, либо скачать нужный драйвер на сайте owen.ru. После подключения прибора надо указать номер эмулируемого СОМ-порта в окне *Настройка порта*. Узнать номер порта можно в *Диспетчере устройств* компьютера, как показано на рис. 6.13.

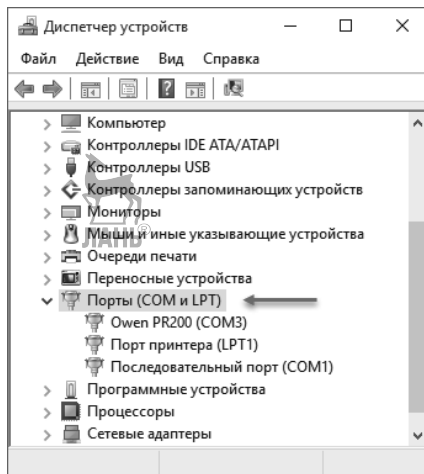
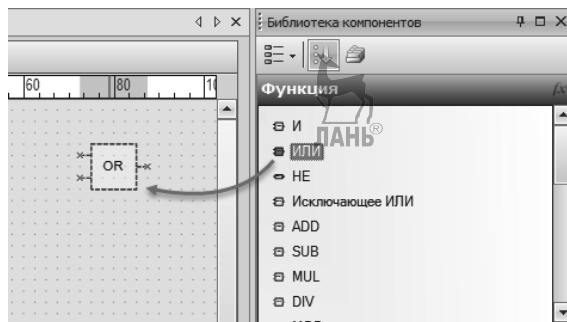


Рис. 6.13. Диспетчер устройств

На рис. 6.13 номер COM-порта прибора ПР200 определен, как COM3. Чтобы открыть окно *Настройка порта*, нужно щелкнуть левой кнопкой мыши по вкладке главного меню *Прибор* и выбрать строку *Настройка порта*.

6.4.3. Размещение компонентов на рабочем поле и создание коммутационной программы

При создании коммутационной программы компоненты перетаскиваются на рабочее поле из *Библиотеки компонентов* методом "drag&drop" (рис. 6.14).

Рис. 6.14. Перенос компонентов из *Библиотеки* на *Рабочее поле*

Входы и выходы компонентов необходимо связать соединительной линией, как показано на рис. 6.15.

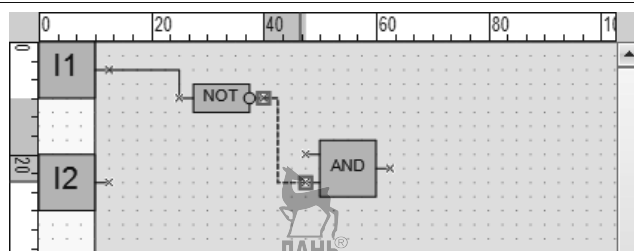


Рис. 6.15. Соединение компонентов

Настройка параметров компонентов осуществляется на панели Свойства. Для этого необходимо предварительно выделить компонент и затем задать его параметры в окне панели, как показано на рис. 6.16.

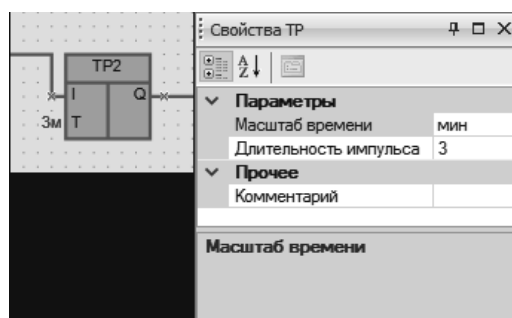
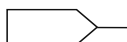
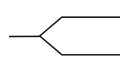


Рис. 6.16. Настройка параметров компонентов

На рабочее поле можно добавлять не только компоненты из *Библиотеки компонентов*, но и переменные и константы в виде отдельных блоков. Чтобы добавить блок переменной на рабочее поле, следует выбрать нужный тип блока на панели вставки и затем щелкнуть левой кнопкой в том месте рабочего поля, где должен располагаться блок, как показано на рис. 6.17. Входные и выходные блоки имеют разное направление:

 блок входной переменной (для передачи значения в программу);

 Блок выходной переменной (для записи значения из программы).

В OWEN Logic используются три типа переменных:

булевский (двоичный);

целочисленный;

с плавающей запятой (вещественный).

Значения от одной переменной к другой могут передаваться только при совпадающих типах переменных. Линия связи может быть создана только для блоков одного и того же типа: булевского, целочисленного или с плавающей запятой. Для создания линий связи между входами и выходами блоков разных

типов следует использовать блоки преобразования, которые расположены справа в панели вставки.

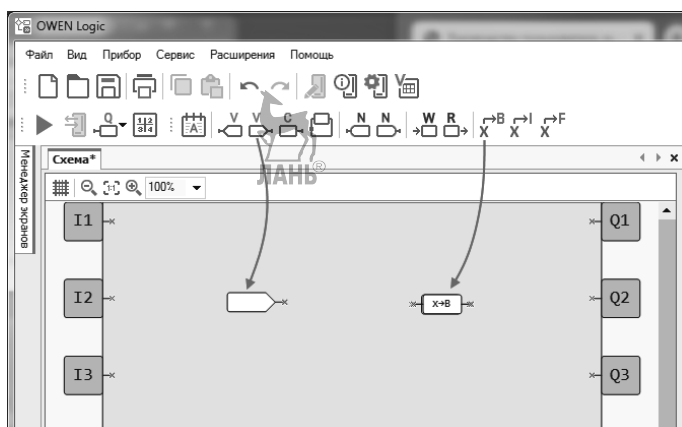
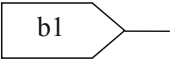
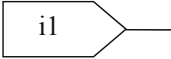
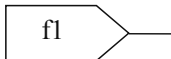


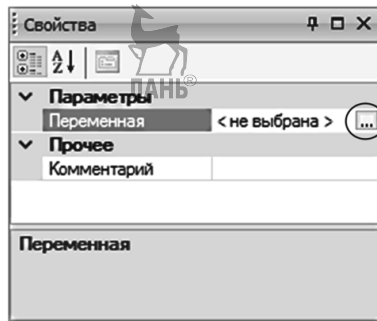
Рис. 6.17. Добавление блока переменной и блока преобразования на рабочее поле

Обозначения переменных различного типа приведены в табл. 6.4.

Таблица 6.4.

Булевский тип	Целочисленный тип	Тип с плавающей запятой
		
<p>Переменная булевого типа может принимать одно из двух значений: «1» или «0». На схеме переменные булевого типа соединяются черными линиями</p>	<p>Переменная целочисленного типа может принимать значение целого числа в диапазоне от 0 до 4294967295. На схеме переменные целочисленного типа соединяются красными линиями:</p>	<p>Переменная типа с плавающей запятой может принимать значение вещественного числа в диапазоне от (-3,402823466E+38) до (+3,402823466E+38). На схеме переменные типа с плавающей запятой соединяются фиолетовыми линиями:</p>

Параметры блока переменной задаются в панели *Свойства*. Для этого надо выделить блок переменной в рабочем поле, затем нажать кнопку *Переменная* в панели *Свойства* и щелкнуть левой кнопкой мыши по значку (...), как показано на рис. 6.18.

Рис. 6.18. Панель *Свойства*

После нажатия кнопки (...) откроется окно *Таблица переменных* (рис. 6.19), в которой нужно выбрать существующую или создать новую переменную и нажать "ОК". Выбранная переменная будет привязана к добавленному блоку и будет отображаться в панели *Переменная* в интерфейсном окне.

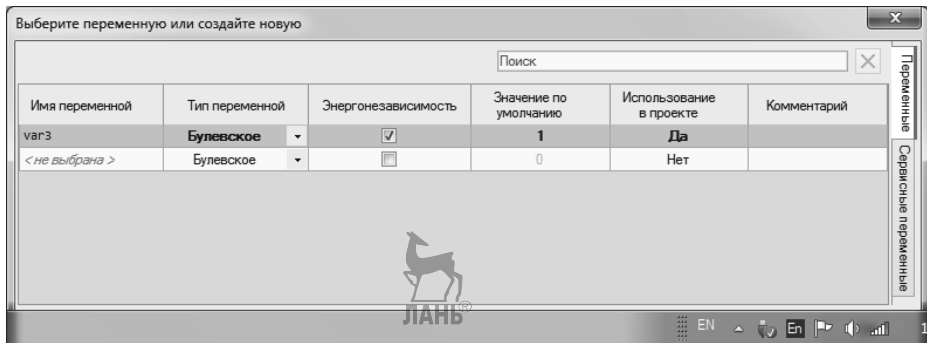


Рис. 6.19. Таблица переменных

Если в процессе создания блока переменной произошла ошибка, то блок переменной будет выделен красным цветом.

Особенностью работы блоков логических функций является их самонастройка на тип данных. Если к входу блока логической функции была подсоединена целочисленная переменная, то блок автоматически перестраивается на работу с целочисленными значениями.

Для функций «И» и «ИЛИ» следует учитывать, что неподключенные входы блоков будут иметь следующие состояния:

для функции «И» – логическая «1»;

для функции «ИЛИ» – логический «0».

В этом случае блоки выполняют функцию повторителя сигнала.

6.4.4. Работа с панелью симуляции

Режим симуляции используется для отладки коммутационной программы и позволяет проанализировать состояние сигналов внутри программы. Для перехода в режим симуляции следует нажать кнопку в виде треугольника на панели инструментов или выбрать в главном меню *Сервис > Режим симулятора*. Откроется панель симуляции, показанная в верхней части рис. 6.20. На этом же рисунке приведена таблица с расшифровкой обозначений, которые используются в панели симуляции.

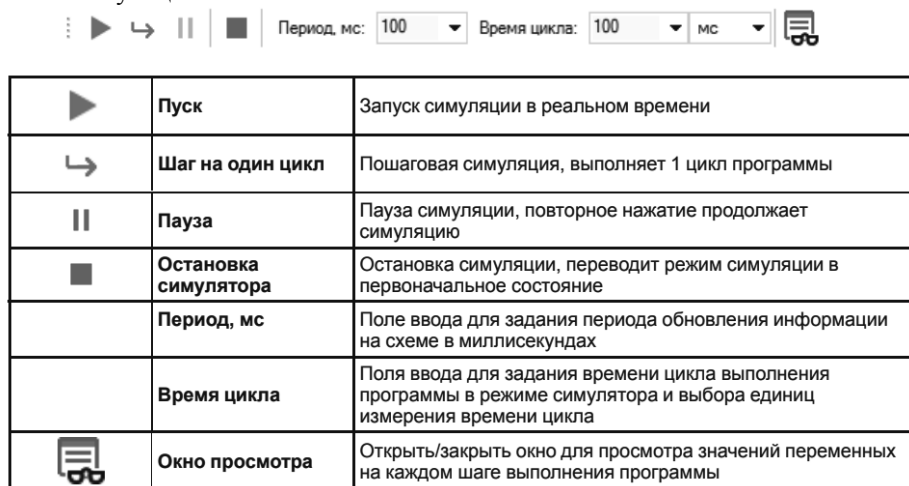


Рис. 6.20. Панель симуляции и таблица с расшифровкой обозначений

Следует различать *время цикла в режиме симуляции* и *время выполнения рабочего цикла прибора*.

Время выполнения рабочего цикла прибора складывается из следующих составляющих:

опрос состояния физических входов прибора и копирование их значений в ячейки памяти;

обработка программы;

чтение/запись сетевых переменных программы;

запись результатов работы программы в физические выходы прибора.

По умолчанию время цикла равно 1 мс. Но это не постоянная величина. Прибор сам подстраивает время цикла в зависимости от сложности программы. К увеличению времени цикла приводят следующие факторы:

сложность алгоритма (задействовано большое количество функциональных блоков и макросов);

в программе используется большое количество сетевых переменных;

используется большое количество элементов управления данными с помощью дисплея прибора.

Пользователь не может задавать время выполнения рабочего цикла. Если прибор оснащен дисплеем, то текущее время цикла можно посмотреть в системном меню прибора. Если прибор подключен к ПК, то время цикла можно посмотреть в окне *Информация о приборе*.

Время цикла в режиме симуляции может быть выбрано произвольно и требуется для симуляции работы временных функциональных блоков: TON, TOF, BLINK и др. Все уставки временных функциональных блоков будут отработаны в промежуточных шагах симулятора.

Порядок симуляции программы:

- Запуск симулятора в режиме реального времени или в пошаговом режиме.
- Задание значений входных сигналов на блоках программы:
- Подбор значений параметров *Период* и *Время цикла* для удобства симуляции.
- Выход из режима симуляции для корректировки программы.

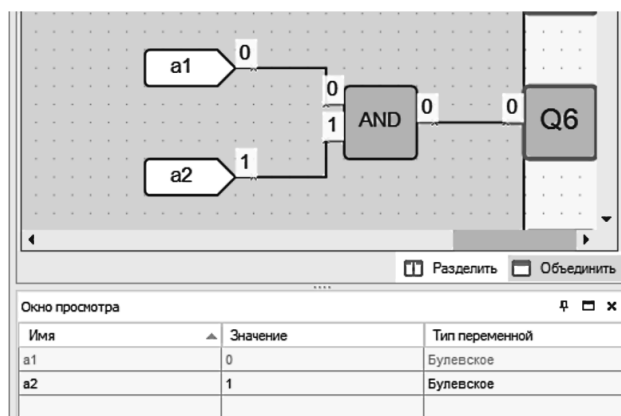


Рис. 6.21. Окно просмотра панели симуляции

В режиме симуляции необходимо соблюдать следующие правила:

- программа и макросы моделируются по отдельности;
- работа блоков, не имеющих связи ни с одним из выходов прибора или выходным блоком сетевой переменной, не симулируется;
- симуляция не будет работать для энергозависимых и некорректно привязанных переменных.

Окно просмотра панели симуляции (показано в нижней части рис. 6.21) предназначено для просмотра значений переменных или входов/выходов на каждом шаге выполнения программы. До запуска симуляции оно пустое. Для добавления переменной для отображения следует нажать в пустое поле в колонке *Имя*, затем на кнопку (...). Откроется таблица переменных, в которой могут быть выбраны переменные проекта и входы/выходы. Выбранные переменные добавятся в окно просмотра.

6.4.5. Библиотека компонентов

В среде программирования Owen Logic (версия 1.14.194.18756) в состав библиотеки компонентов входят следующие функции: (таблица 6.5):

Таблица 6.5

И – логическая функция И	ИЛИ – логическая функция ИЛИ
НЕ – логическая функция НЕ	Исключающее ИЛИ – логическая функция Исключающее ИЛИ–
ADD – арифметическая операция сложения	SUB – арифметическая операция вычитания
MUL – арифметическая операция умножения	DIV – арифметическая операция деления
MOD – арифметическая операция деления с остатком	EQ – операция сравнения
GT – больше	SEL – тернарная условная операция сравнения
SHL – сдвиговый регистр влево	SHR – сдвиговый регистр вправо
EXTRACT – чтение бита	PUTBIT – запись бита
DC32 – дешифратор	CD32 – шифратор
fADD – арифметическая операция сложения чисел с плав. запятой	fSUB – арифметическая операция вычитания с плав. запятой
fMUL – арифметическая операция умножения с плав. запятой	fDIV – арифметическая операция деления с плав. запятой
fPOW – арифметическая операция возведения в степень	fABS – абсолютное значение числа с плав. запятой
fGT – больше для чисел с плав. запятой	fSEL – тернарная условная операция сравнения для чисел с плав. запятой

В состав библиотеки компонентов входят следующие функциональные блоки (таблица 6.6):

Таблица 6.6

RS – RS-триггер с приоритетом выключения	SR – SR-триггер с приоритетом включения
RTRIG – детектор переднего фронта	FTRIG – детектор заднего фронта
DTRIG – Д-триггер	TP – импульс включения заданной длительности
TON – таймер с задержкой включения	TOF – таймер с задержкой выключения
BLINK – генератор импульсов	CT – инкрементный счетчик с автосбросом
CTN – универсальный счетчик	CTU – инкрементный счетчик
CLOCK – интервальный таймер	CLOCK WEEK – интервальный таймер с маской недели
PID – ПИД регулятор	

В заключение рассмотрим примеры применения среды программирования Owen Logic для составления коммутационных программ.

6.4.6. Примеры управляющих программ в OWEN Logic

Пример 6.1. Резервуар с водой.

Схема резервуара с водой представлена на рис. 6.22.

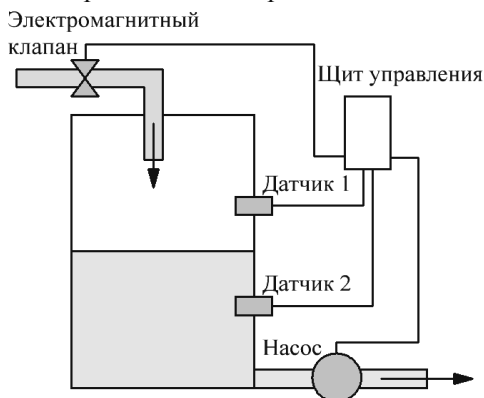


Рис. 6.22. Схема установки

Имеется резервуар, который должен обеспечивать водой потребителя. В верхней части резервуара имеется водопроводная труба, через которую в резервуар поступает вода. Труба открывается и закрывается электромагнитным клапаном. Вода откачивается из резервуара с помощью насоса. Внутри резервуара находятся два датчика уровня воды. Логика работы установки следующая. В начальный момент в резервуаре нет воды. При нажатии кнопки Пуск срабатывает электромагнитный клапан, открывает трубу, в резервуар поступает вода. При срабатывании нижнего датчика 2 запускается насос, который откачивает воду из резервуара. Труба остается открытой. Вода продолжает наполнять резервуар. Когда уровень воды достигнет датчика 1, срабатывает электромагнитный клапан и перекрывает подачу воды. Насос при этом продолжает работать. Уровень воды в резервуаре начинает понижаться. Когда уровень воды будет ниже датчика 1, сработает электромагнитный клапан и вновь откроет подачу воды. Если окажется, что уровень воды будет ниже не только датчика 1, но и ниже датчика 2, то насос отключится, и будет оставаться в отключенном состоянии, пока уровень воды не поднимется выше датчика 2. При нажатии кнопки Стоп работа установки прекращается.

Исходя из логики работы установки, составляем таблицу истинности. Имеются две входных переменных: датчик 1 и датчик 2, и две выходных переменных: электромагнитный клапан и насос. Используем следующие значения входных и выходных переменных.

Электромагнитный клапан:

0 – закрытое положение (вода перекрыта);
 1 – открытое положение (вода поступает в резервуар).

Насос:

0 – не работает;
 1 – работает.

Датчики уровня воды:

0 – датчик не сработал (уровень воды ниже датчика);
 1 – датчик сработал (уровень воды выше датчика).

Составим таблицу истинности 6.5.

Таблица 6.5

X1 (датчик 1)	X2 (датчик 2)	Q1 (клапан)	Q2 (насос)
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	1

Применим нормальную форму ИЛИ (см. главу 2) и получим для данной таблицы истинности следующие логические выражения

Клапан:

$$Q1 = \overline{X1} \cdot \overline{X2} + \overline{X1} \cdot X2$$

Насос:

$$Q2 = \overline{X1} \cdot X2 + X1 \cdot X2$$

Откроем программу OWEN Logic. Щелкнем по пиктограмме *Новый проект в Панели инструментов*. Появится выпадающее окно с перечнем моделей логических реле. Выберем, например, модель ПР200-220.2(4).X, как показано на рис. 6.23 и щелкнем по кнопке ОК.

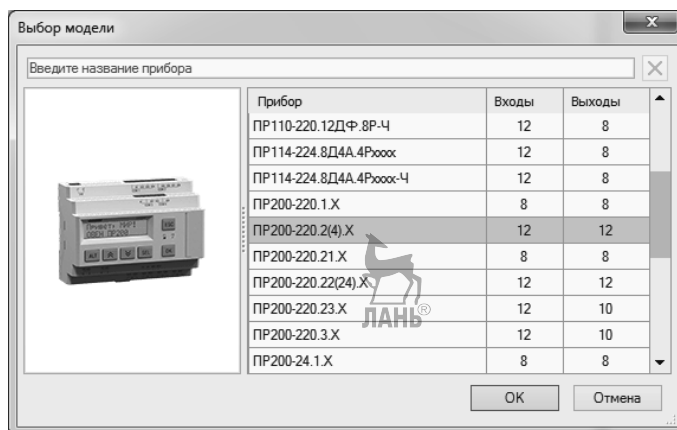


Рис. 6.23. Выбор модели логического реле

Появится интерфейсное окно, в котором подготовим коммутационную программу (рис. 6.24) в соответствии с полученными логическими выражениями. Программа состоит из двух ветвей: для клапана с выходом на Q1, и для насоса с выходом на Q2.

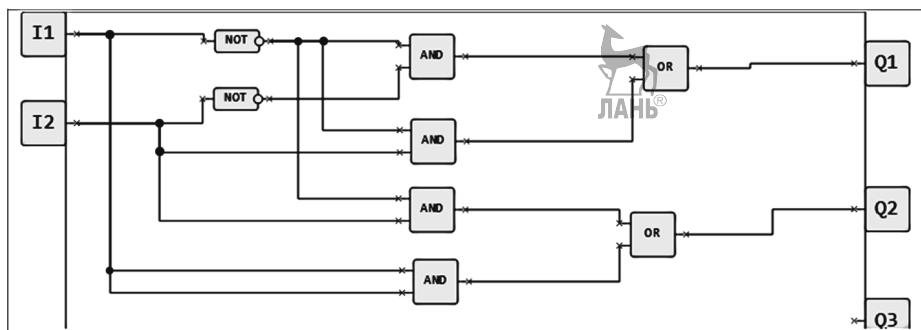


Рис. 6.24. Логическая схема

Далее добавим в схему кнопки Пуск и Стоп (кнопки без фиксации). Эти кнопки будут связаны с входами I3 и I4 соответственно. Кнопки будут управлять работой схемы с помощью SR-триггера и двух дополнительных блоков «И» в каждой ветви.

Кроме того, в схему необходимо добавить блок задержки включения насоса. Это нужно сделать для того, чтобы уровень воды в момент запуска насоса был выше уровня датчика 2. В противном случае можем получить непрерывное импульсное включение и отключение насоса около уровня датчика 2. Окончательный вид коммутационной программы приведен на рис. 6.25.

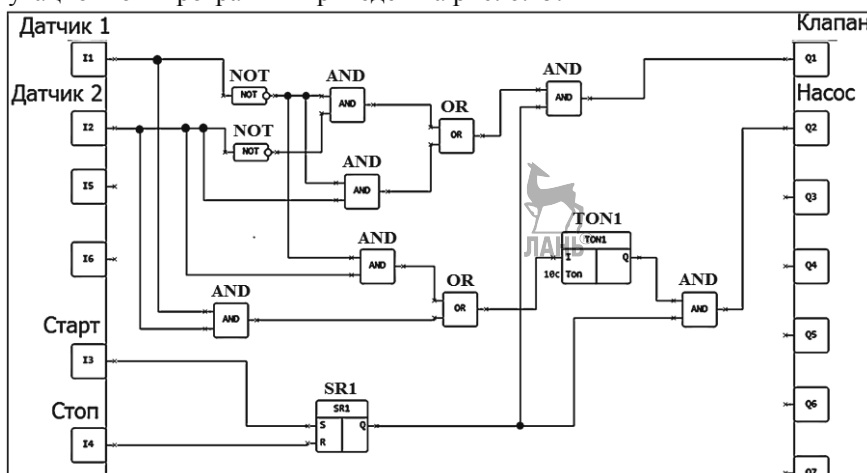


Рис. 6.25. Коммутационная программа

Пример 6.2. Вычисление значений арифметических выражений.

Составить коммутационную программу для вычисления выражения

$$(1,5 + 2,2)^3 = ?$$

Создание коммутационной программы. Поскольку в арифметическом выражении присутствуют вещественные числа, то для констант необходимо задать тип с плавающей запятой и использовать в коммутационной программе блоки арифметических операций с плавающей запятой: fADD и fPOW. Коммутационная программа показана на рис. 6.26.

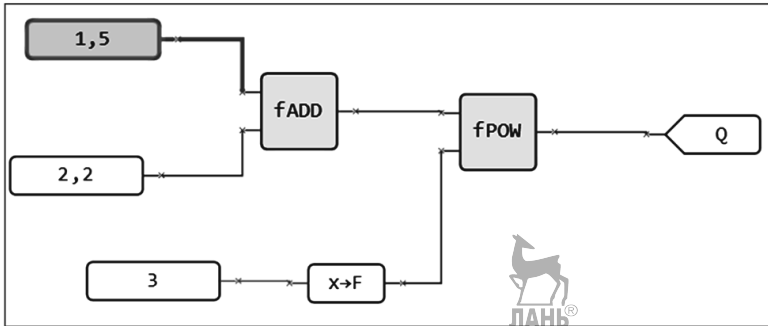


Рис. 6.26. Коммутационная программа

Чтобы задать тип константы с плавающей запятой (например, для числа 1,5) надо выделить это число на схеме и в панели Свойства, (в правом нижнем углу) задать тип константы *С плавающей запятой* и задать значение константы (1,5), как показано на рис. 6.27.

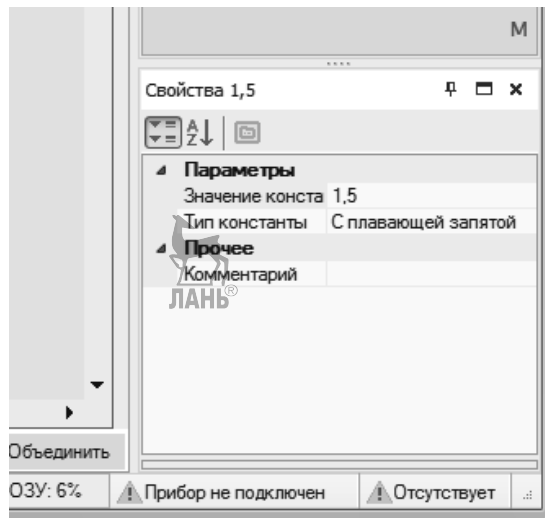


Рис. 6.27. Задание типа и значения константы

Для показателя степени выбран тип *Целочисленный*, но поскольку все числа в арифметическом выражении должны иметь один тип, то показатель степени надо перевести из типа *Целочисленный* в тип *С плавающей запятой*. Для этого в программе используется блок-преобразователь ($x \rightarrow F$).

Для переменной на выходе задаем название Q, тип *С плавающей запятой*, оставляем значение по умолчанию 0, как показано на рис. 6.28.

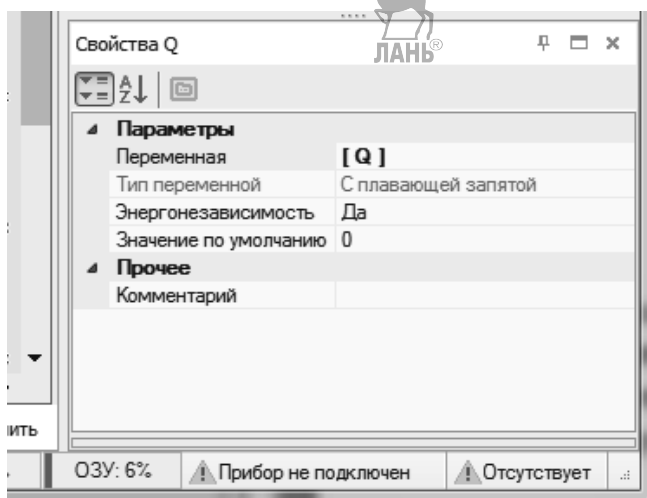


Рис. 6.28. Задание параметров переменной на выходе

После того, как все вспомогательные операции проделаны, запускаем симуляцию и получаем результат (50,653), показанный на рис. 6.29.

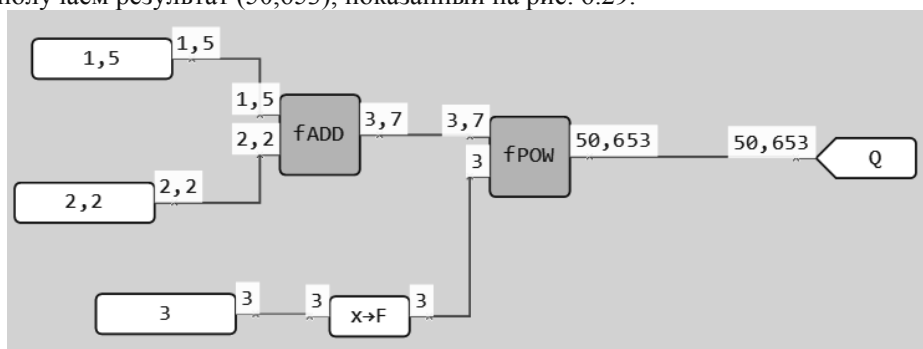


Рис. 6.29. Симуляция арифметического выражения

Пример 6.3. Выключатель света с автоматическим отключением.

Этот пример приведен в Руководстве пользователя Owen Logic. В подсобных помещениях и коридорах электрический свет нужен ограниченное время. После включения освещения, его нередко забывают отключить, что приводит к излишнему расходу электроэнергии. Задача обеспечить включение света на

заданный интервал времени, например, у входной двери в квартиру, по следующему алгоритму:

перед входной дверью установлен датчик освещения (F1) и кнопка включения света ТАЙМЕР (SB1);

при кратковременном нажатии на кнопку ТАЙМЕР при недостаточном естественном освещении светильник должен включаться на интервал времени 1 мин – этого времени достаточно, чтобы открыть ключом дверь;

при удерживании нажатой кнопки ТАЙМЕР в течение 2 с светильник должен включаться на интервал времени 3 мин независимо от внешнего освещения – этот режим может потребоваться при уборке коридора;

предусмотреть возможность управления работой светильника по командам от внешних управляющих устройств или при помощи выключателя СВЕТ (SA1), независимо от внешнего освещения;

предусмотреть возможность включения светильника только в вечернее и ночное время.

Выбор программируемого прибора.

Для реализации данной задачи логическое реле должно иметь встроенные часы реального времени, три входа и один выход, что обеспечивают приборы из серии ПР110, имеющие последнюю букву «Ч» в обозначении.

При выполнении автоматике на основе ПР110-24.8Д.4Р-Ч можно воспользоваться схемой (рис. 6.30):

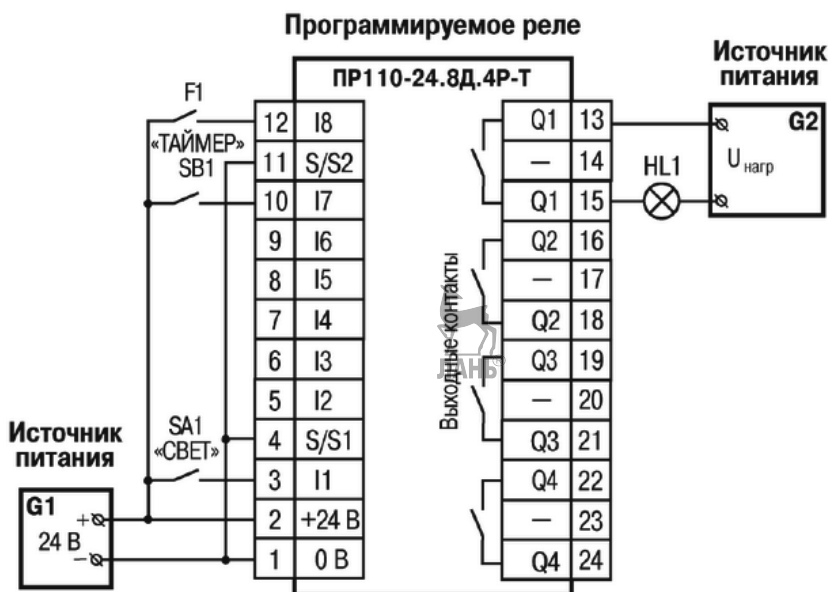


Рис. 6.30. Электрическая схема подключения ПЛР, SA1 - переключатель OFF-ON; SB1 - кнопка без фиксации OFF-ON; HL1 - световая лампа

Создание коммутационной программы.

На этапе планирования составляется схема реализации поставленной задачи на основе логических элементов и функциональных блоков, доступных в программе. Функциональная блок-схема имеет вид, показанный на рис. 6.31.

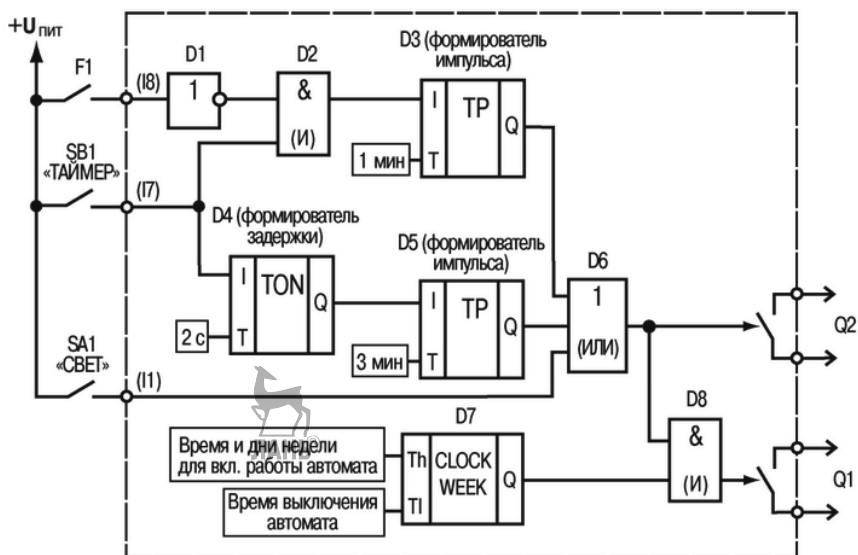


Рис. 6.31. Функциональная блок-схема

Выход Q2 схемы используется как контрольный для проверки функционирования логической части (элементов D1–D6). Выход Q1 является основным и может включиться только в заданные таймером D7 интервалы времени при соблюдении логических условий, обеспечиваемых элементами D1–D6.

Описание работы схемы:

При кратковременном (менее 2 с) нажатии на кнопку ТАЙМЕР (SB1), логическая «1» поступает на вход логического элемента «И» (D2). Так как на втором входе D2 также присутствует логическая «1», на выходе D2 появится логическая «1», которая запустит формирователь импульса (D3) длительностью 1 мин. Этот импульс через элемент «ИЛИ» (D6) поступит на выход Q2.

Контакты датчика F1 замыкают цепь только при хорошем естественном освещении – в этом случае логическая «1» на входе D1 преобразуется в логический «0» на выходе и поступает на вход логического элемента «И» (D2), что блокирует работу D2. При этом нажатие на кнопку ТАЙМЕР не включает выход Q2.



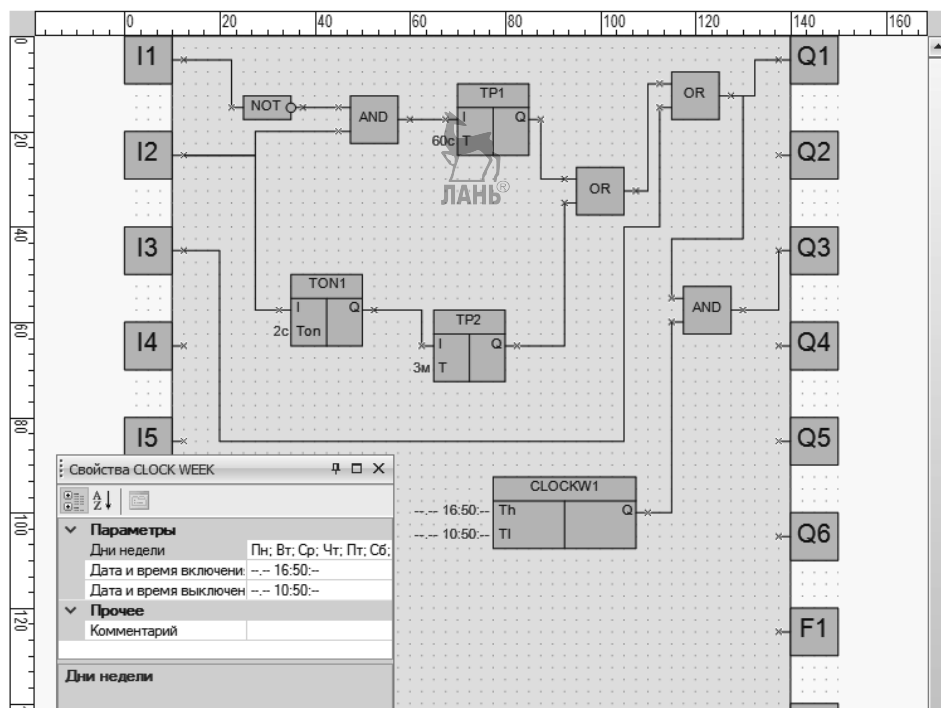


Рис. 6.32. Коммутационная программа

При удержании нажатой кнопки ТАЙМЕР более 2 с, на выходе блока D4 появится логическая «1», которая запустит формирователь импульса (D5) длительностью 3 мин. Этот импульс через элемент «ИЛИ» (D6) поступит на выход Q2 (интервал действия импульса не зависит от состояния датчика F1).

При включении тумблера «СВЕТ» (SA1) логическая «1» через элемент «ИЛИ» (D6) поступит на выход Q2 (на время включения SA1).

Сигнал логическая «1» с выхода Q2 поступает на вход логического элемента И (D8), а если на второй вход этого элемента также поступает логическая «1» (от таймера реального времени D7) – на выходе Q1 появится логическая «1» (выходное реле Q1 включится).

Коммутационная программа, обеспечивающая работу логического реле, показана на рис. 6.32.

Пример 6.4. Автоматическое управление электромотором мешалки.

Этот пример приведен в Руководстве пользователя Owen Logic. При производстве пищевых продуктов часто требуется перемешивать компоненты (например, молоко или сливки на молочной ферме) в течение определенного времени.

Задача обеспечить работу установки по следующему алгоритму:

необходимы два режима работы: «Автоматический» и «Ручной», переключаемых тумблером «РЕЖИМ» (SA1);

в режиме «Автоматический», при включении оператором установки кнопкой ПУСК (SB1), производится автоматическое включение и отключение электромотора через заданные интервалы времени (15 с – включен, 10 с – отключен). Отключение установки производится через интервал 5 мин или оператором при помощи кнопки СТОП (SB2);

в режиме «Ручной» производится прямое управление работой электромотора от кнопок ПУСК и СТОП (без временных интервалов отключения);

при перегрузке электромотора (на котором устанавливается соответствующий датчик – F1) должно срабатывать автоматическое отключение установки с индикацией режима «Неисправность» лампой (HL1) и звуковым прерывистым сигналом (интервал повторения звукового сигнала 3 с);

звуковой сигнал должен отключаться кнопкой СБРОС (SB3);

с помощью кнопки КОНТРОЛЬ (SB4) проверяется исправность элементов сигнализации – работа лампы и звукового сигнала.

Выбор программируемого прибора

Для реализации данной задачи управляющий прибор должен иметь шесть входов (по числу управляющих сигналов) и три выхода, что обеспечивает любая модель прибора из серии ПР110.

При выполнении автоматики на основе ПР110-24.8Д.4Р можно воспользоваться схемой:

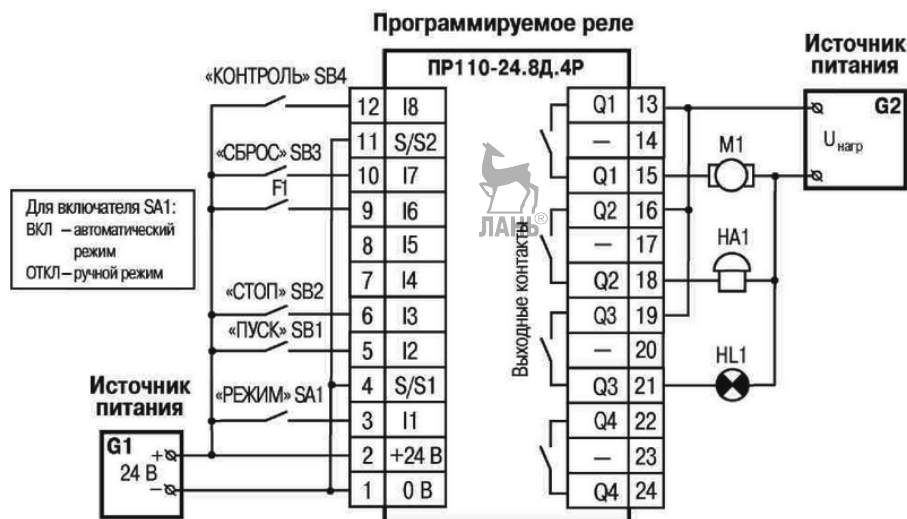


Рис. 6.33. Электрическая схема подключения

Электрическая схема подключения ПЛР, где SA1 – переключатель OFF-ON; SB1–SB4 – кнопки без фиксации OFF-ON; M1 – электромотор; HA1 – звонок; HL1 – индикаторная лампа.

Работу программы прибора можно представить в виде цепей схемы, приведенной ниже (рис. 6.30), (выходы Q1–Q3 соответствуют двум контактам клеммника ПЛП).

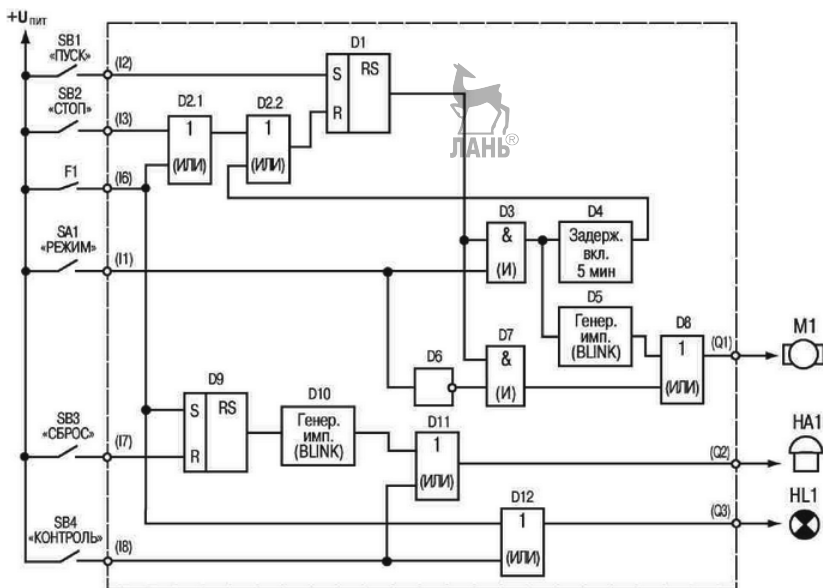


Рис. 6.34. Функциональная блок-схема

Описание работы схемы.

Цепь входа I2 (включение установки): при нажатии кнопки ПУСК (SB1) на входе S (D1) появляется логическая «1» – на выходе RS-триггера D1 установится логическая «1». Этот сигнал поступает дальше, в зависимости от состояния выключателя SA1:

при разомкнутых контактах SA1 (РЕЖИМ – Ручной), логическая «1» проходит через элементы D8, D9 и поступает на выход Q1 (контактами выходного реле электромотор M1 включится);

при замкнутых контактах SA1 (РЕЖИМ – Автоматический), логическая «1» проходит только через элемент D3 для запуска работы блоков D4, D5.

Цепь входа I3 (отключение установки): при нажатии кнопки СТОП (SB2) или срабатывании датчика F1 на входе R (D1) появляется логическая «1» – на выходе RS-триггера (D1) установится логический «0» (включение выхода Q1 блокируется).

Цепь входа I1 (формирование интервалов работы электромотора): при включенном триггере D1 и замкнутых контактах выключателя SA1 (РЕЖИМ – Автоматический):

Сигнал логическая «1» от SA1 проходит через элемент D3 и поступает на D4 (формирователь импульса с задержкой 5 мин). Этот импульс, проходя через элемент ИЛИ (D2), поступит на вход R триггера D1 – установка отключится.

Сигнал логическая «1» с выхода D3 поступает на D5 – генератор импульсов с параметрами: логическая «1» – 15 с, логический «0» – 10 с. Эти импульсы с

выхода генератора проходят через элемент ИЛИ (D8) и поступают на выход Q1 для управления работой электромотора M1.

При разомкнутых контактах выключателя SA1 (РЕЖИМ – Ручной) логический элемент D3 заблокирован и сигнал с выхода триггера D1 через элемент D6 сразу поступает на выход Q1, т. е. в этом режиме состояние выхода триггера непосредственно управляет работой электромотора M1.

Цепь входа I6 (включение звуковой сигнализации): при срабатывании датчика F1 на входе S (D9) появляется логическая «1». На выходе RS-триггера D9 установится логическая «1», которая включает генератор D10 с параметрами: логическая «1» – 3 с, логический «0» – 3 с. Эти импульсы проходят через элемент ИЛИ (D12) и поступают на выход Q2 для управления работой звонка HA1.

Цепь входа I7 (отключение звуковой сигнализации): при нажатии кнопки СБРОС (SB3) на входе R (D9) появляется логическая «1» – на выходе RS-триггера установится логический «0» – генератор D10 отключится.

Цепь входа I8 (контроль работы лампы и звуковой сигнализации): при нажатой кнопке КОНТРОЛЬ (SB4) логическая «1» через элемент «ИЛИ» (D11) поступает на включение выхода Q3 – лампа HL1 будет непрерывно светиться. Выход Q3 включится также, если контакты датчика F1 замкнуты. При нажатой кнопке КОНТРОЛЬ (SB4) логическая «1» через элемент «ИЛИ» (D12) поступает на включение выхода Q2 – звонок HA1 будет непрерывно работать.

Коммутационная программа, обеспечивающая работу ПЛР, имеет вид (рис. 6.35):

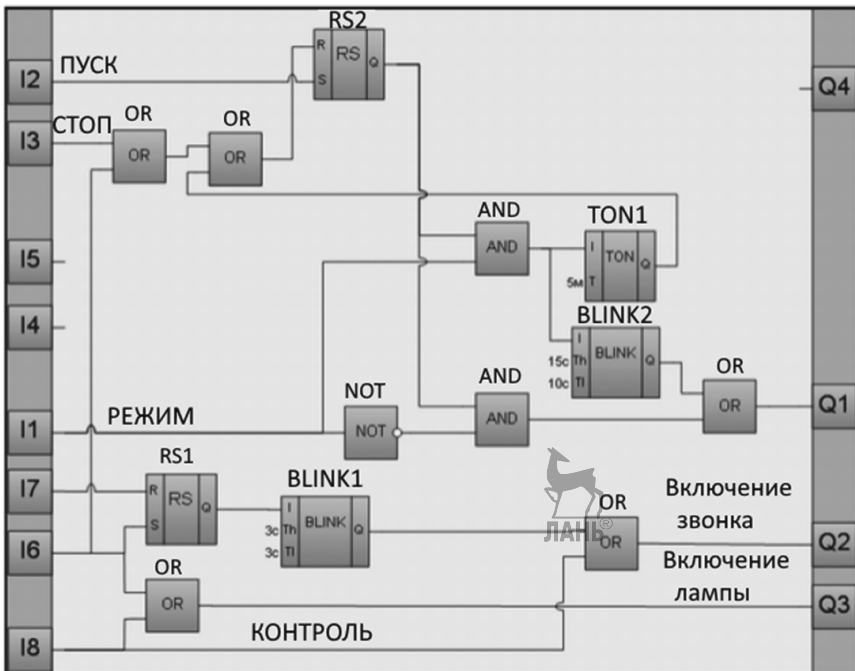


Рис. 6.35. Коммутационная программа

Примечание. Оставшиеся свободные два входа и один выход, при необходимости, можно использовать для введения дополнительных функций. Например, переключать четыре разных интервала длительности автоматической работы электромотора или изменять другие рабочие параметры установки. Технологический цикл работы установки можно сделать полностью автоматическим, если схему дополнить инкрементным счетчиком (СТ), выходным сигналом которого можно выключать триггер D1.

Контрольные вопросы и задания

1. Какие модели логических реле ОВЕН имеют ЖК дисплей?
2. Какие модели логических реле ОВЕН не требуют комплекта программирования ПР-КП10/ПР-КП20?
3. Что означают буквы Р, К, С, И, У в обозначении выходных цепей реле ОВЕН?
4. Какие напряжения и токи могут выдавать аналоговые выходы логического реле ОВЕН?
5. На какое напряжение питания рассчитаны логические реле?
6. Какой максимальный ток могут выдержать релейные выходы реле ПР200?
7. Какое программное обеспечение используется для программирования ПЛК ОВЕН и ПЛР ОВЕН?
8. С помощью какой вкладки главного меню можно показать или скрыть панель *Библиотека компонентов* в интерфейсном окне?
9. Что надо сделать, чтобы расположить панель *Свойства* в нижнем правом углу интерфейсного окна?
10. Как задать параметры функций и функциональных блоков в OWEN Logic?
11. Где расположена панель симуляции в интерфейсном окне?
12. Каким образом организовать пошаговое выполнение коммутационной программы?
13. Как убрать сетку с рабочего поля интерфейсного окна?
14. Какой кабель используется для подключения логического реле ОВЕН к компьютеру?
15. Как определить номер COM-порта при подключении логического реле к компьютеру?
16. Какие переменные могут использоваться в OWEN Logic?
17. Как задать параметры блока переменной?
18. Как подать три сигнала на вход блока «И», если блок «И» имеет только два входа?
19. Какие временные блоки (таймеры) используются в OWEN Logic?
20. Запустить OWEN Logic, составить коммутационную программу с блоком *Исключающее ИЛИ* и описать его работу.

21. Запустить OWEN Logic, составить коммутационную программу с блоком TP и описать его работу.
22. Запустить OWEN Logic, составить коммутационную программу с блоком RTRIG и описать его работу.
23. Запустить OWEN Logic и составить программы для вычисления арифметических выражений

$$(15 - 7) / 3 = ?$$

$$(0,5 + 2,7) \cdot 2 = ?$$



ГЛАВА 7. ПРОГРАММИРОВАНИЕ ОБЕН В CODESYS

7.1. Установка Codesys на компьютер

Codesys (Controllers Development System) – это интегрированная среда программирования логических контроллеров, разработанная компанией 3S-Smart Software Solutions GmbH (Германия). Получить полную информацию о Codesys можно на сайтах www.codesys.com и www.codesys.ru.

Codesys – это инструмент программирования промышленных контроллеров, опирающийся на международный стандарт МЭК 61131-3. Основу Codesys составляют три компонента:

- специализированные редакторы для ввода прикладных программ;
- встроенные компиляторы, генерирующие исполняемый машинный код;
- широкий набор средств отладки и сопровождения.

Используемые редакторы и отладочные средства базируются на известных принципах, знакомых по другим популярным средам профессионального программирования (например, Visual C++).

Прежде, чем переходить к практической работе с пакетом Codesys, необходимо установить это программное обеспечение на компьютер. Дистрибутив Codesys можно бесплатно скачать с уже упоминавшихся сайтов www.codesys.com, www.codesys.ru или с сайта ОБЕН www.owen.ru. Поскольку далее codesys будет рассматриваться применительно к программированию контроллеров ОБЕН, то скачаем его с сайта www.owen.ru. Для этого на сайте надо пройти по ссылкам [Каталог продукции](#) > [Среда программирования CODESYS](#) > [CODESYS v.2](#). Процесс установки ПО Codesys на компьютер не вызывает затруднений, надо просто следовать указаниям мастера установки.

Следующим этапом будет установка таргет-файлов. Таргет-файлы – это файлы с описанием конфигурации конкретной модели ПЛК. Таргет-файл подсказывает Codesys, какое количество и какие входы/выходы имеет выбранный ПЛК. Загрузить таргет-файлы можно с уже упоминавшейся страницы на сайте ОБЕН. Для установки таргет-файла надо проделать следующую последовательность действий:

1. Скачать Таргет-файл.
2. Распаковать содержимое архива.
3. Открыть папку Таргет-файла для соответствующей модели контроллера.
4. Запустить файл `InstallTarget.bat`.
5. Установка завершится автоматически.

7.2. Пользовательский интерфейс

Итак, Codesys установлен, таргет-файлы тоже установлены. Теперь запускаем программу. После запуска программы появляется окно «Настройки целевой

платформы» (рис. 7.1), в котором надо выбрать контроллер, для которого будет создаваться программа.

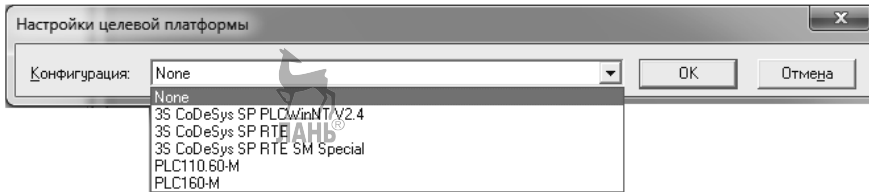


Рис. 7.1. Окно выбора контроллера

После этого появляется другое окно (рис. 7.2), в котором надо выбрать язык программирования

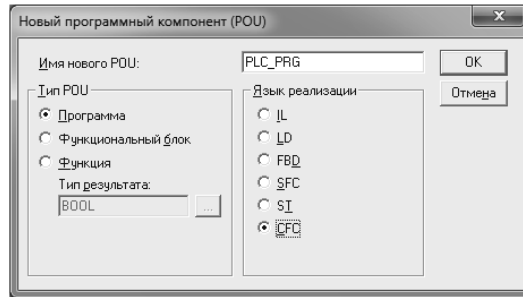


Рис. 7.2. Окно выбора языка программирования

И только после этого появляется интерфейсное окно среды программирования Codesys, представленное на рис. 7.3.

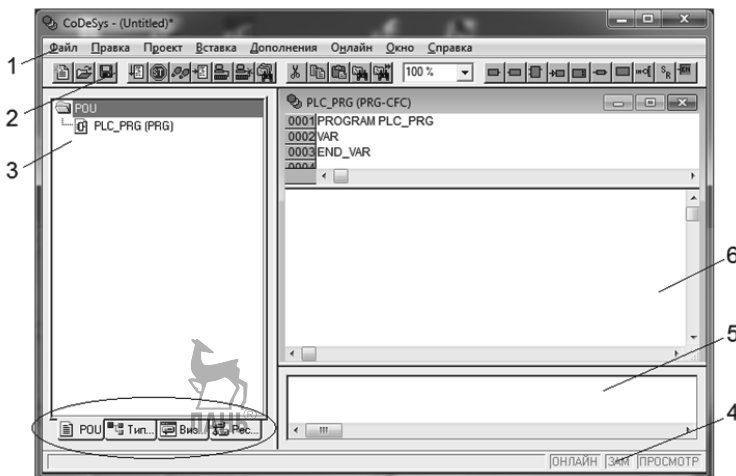


Рис. 7.3. Интерфейсное окно программы Codesys

Цифрами на этом рисунке обозначены следующие элементы интерфейсного окна:

1 – Меню.

2 – Панель инструментов.

3 – Организатор (менеджер) объектов, имеющий внизу вкладки:

POU (Program Organization Unit).

Типы данных (Data type).

Визуализация (Visualization).

Ресурсы (Resources).

4 – Статусная строка, содержащая информацию о текущем состоянии проекта.

5 – Окно сообщений, в котором появляются сообщения компилятора. При сообщении об ошибках двойной щелчок мышкой по сообщению перенаправит нас к объекту, к которому относится данное сообщение.

6 – Рабочая область (область создания и редактирования программы). В верхней части рабочей области выше разделительной линии находится область объявления переменных. Переменные объявляются между строчками `VAR ... END_VAR`, например,

`VAR`

`a20:INT;` (объявление имени и типа входной переменной)

`T5:TIME:=T#5s;` (объявление переменной времени с одновременным заданием продолжительности временного промежутка)

`in2:BOOL;` (объявление булевой переменной)

`END_VAR`

7.3. Константы и переменные.

Целочисленные константы

Программирование начинается с объявления переменных. При объявлении переменной необходимо указать имя переменной, тип и можно сразу задать начальное значение, например,

`a:int:=100;`

`a` – имя переменной,

`int` – тип переменной,

`100` – значение переменной.

Имя переменной (идентификатор) должно начинаться с букв английского алфавита, цифры в начале имени не допускаются. Имя не должно содержать пробелов и спецсимволов, не должно объявляться более одного раза и не должно совпадать с ключевыми словами, например, `var`, `program`, `and`, `word`, `bool` и т.д. Регистр символов не учитывается, это означает, что `VAR1`, `Var1` и `var1` - это одна и та же переменная. Символ подчеркивания является значимым, т.е. `A_BCD` и `AB_CD` – это разные имена. Имя должно включать не более одного символа подчеркивания. Ограничений на длину имени нет. Область применения переменной задается ее типом.

Список всех типов переменных в Codesys доступен через *Ассистент ввода*. Чтобы вызвать *Ассистент ввода*, необходимо нажать комбинацию клавиш Shift-F2. При этом появится выпадающее окно, показанное на рис. 7.4.

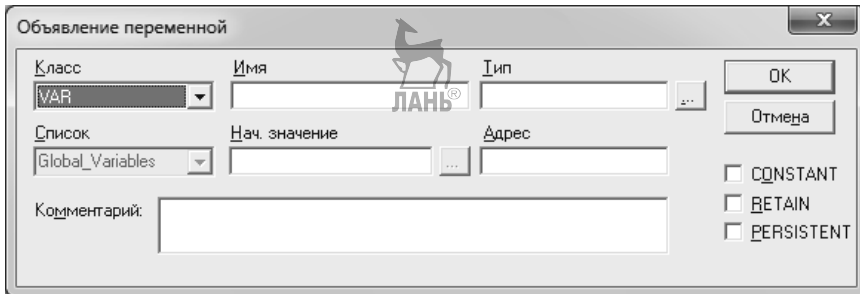


Рис. 7.4. Окно объявления переменной

Далее, если нажать на кнопку (...) справа от поля Тип, то появится окно (Рис. 7.5), в котором перечислены все типы переменных

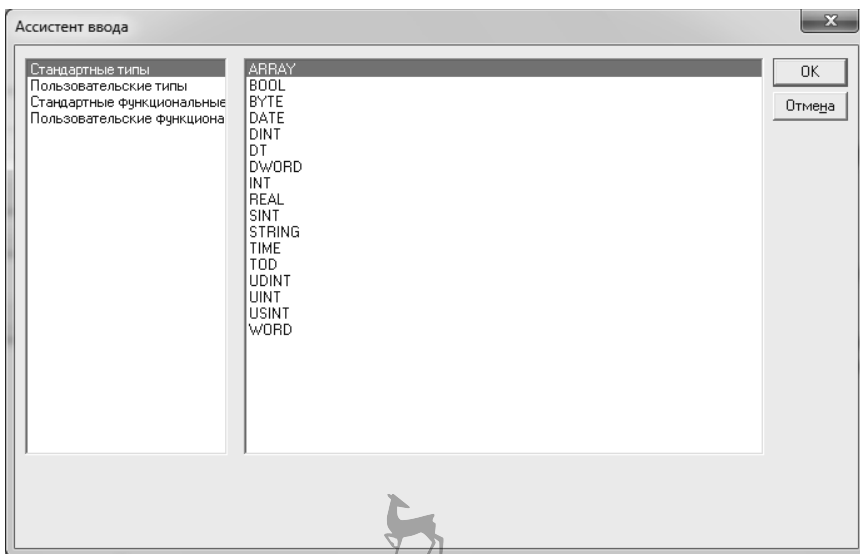


Рис. 7.5. Типы переменных

Переменные подразделяются на несколько типов.

Целочисленные типы: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT. Перечисленные переменные различаются диапазоном сохраняемых данных и, естественно, различными требованиями к памяти. Подробно данные характеристики представлены в таблице 7.1.

Таблица 7.1

Тип	Нижний предел	Верхний предел	Размер памяти
BYTE	0	255	8 Бит
WORD	0	65535	16 Бит
DWORD	0	4294967295	32 Бит
SINT:	-128	127	8 Бит
USINT:	0	255	8 Бит
INT:	-32768	32767	16 Бит
UINT:	0	65535	16 Бит
DINT:	-2147483648	2147483647	32 Бит
UDINT:	0	4294967295	32 Бит

Присвоение данных меньшего типа (BYTE , SINT) переменной большего типа может приводить к потере информации. С другой стороны, присвоение данных очень большого типа (DINT, UDINT) для небольших величин приводит к завышенному расходу памяти и снижению быстродействия при решении задачи.

Переменные типа REAL/LREAL. Переменные REAL и LREAL представляются в формате с десятичной точкой либо в экспоненциальном формате. Запятая вместо точки не допускается:

7.4 но не 7,4

1.64e+009 но не 1,64e+009

Переменные типа STRING. Переменные типа STRING представляются в виде набора символов, заключенных в одинарные кавычки. Строка может содержать пробелы и специальные символы. Символы, не имеющие печатного образа, могут быть заданы шестнадцатеричным кодом в виде двух цифр, следующих за знаком доллара (\$).

Пример переменной string: 'Резервуар основной'.

Переменные типа TIME. Переменные типа TIME в CoDeSys всегда начинаются с префикса "t" или "T" и знака # (решетка). Далее следует время, которое может задаваться в днях "d", часах "h", минутах "m", секундах "s" и миллисекундах "ms". Нет необходимости обязательно определять все составляющие времени, но присутствующие поля обязаны следовать именно в таком порядке (d, затем h, затем m, затем s, затем m, затем ms).

Правильно:

TIME1:= T#14ms;

TIME1:= T#100S12ms;

TIME1:= t#12h34m15s.

Неправильно:

TIME1:= t#5m68s; (младший компонент вышел за верхний предел);

TIME1:= 15ms; (T# пропущено);

TIME1:= t#4ms13d; (неправильная последовательность).

DATE константы. Константы типа DATE начинаются с префикса "d", "D", "DATE" или "date" и последующего "#". Даты задаются в формате Год-Месяц-День.

Например,
DATE#1996-05-06;
d#1972-03-29.

Массивы. Элементарные типы данных могут образовывать одно-, двух-, и трехмерные массивы. Массивы могут быть объявлены в разделе объявлений POU или в списке глобальных переменных. Путем вложения массивов можно получить многомерные массивы, но не более 9-мерных ("ARRAY[0..2] OF ARRAY[0..3] OF ...").

Синтаксис: <Имя_массива>: ARRAY [<ll1>..

где ll1, ll2, ll3(low level) указывают нижний предел индексов; ul1, ul2 и ul3 (upper level) указывают верхние пределы.

Индексы должны быть целого типа. Нельзя использовать отрицательные индексы.

Например: Card_game: ARRAY [1..13, 1..4] OF INT.

Пример инициализации простых массивов:

arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;

arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (сокращение для 1,7,7,7)

arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (сокращение для 0,0,4,4,4,2,3)

Доступ к элементам массива. Для доступа к элементам двухмерного массива используется следующий синтаксис:

<Имя_массива>[Индекс1,Индекс2]

Например: Card_game [9,2]

7.4. Библиотеки

Проект может использовать несколько библиотек, в которые входят различные типы данных и глобальные переменные. Библиотеки "standard.lib" и "util.lib" обязательно входят в стандартный комплект поставки.

Чтобы посмотреть, какие библиотеки присутствуют в Codesys, надо щелкнуть по вкладке *Ресурсы* в нижней части *Организатора объектов* (поз 3, рис. 7.3) и затем щелкнуть по строке *Менеджер библиотек*. Если выделить курсором определенную библиотеку, то можно увидеть, какие функции и функциональные блоки входят в состав библиотеки, как показано на рис. 7.6.

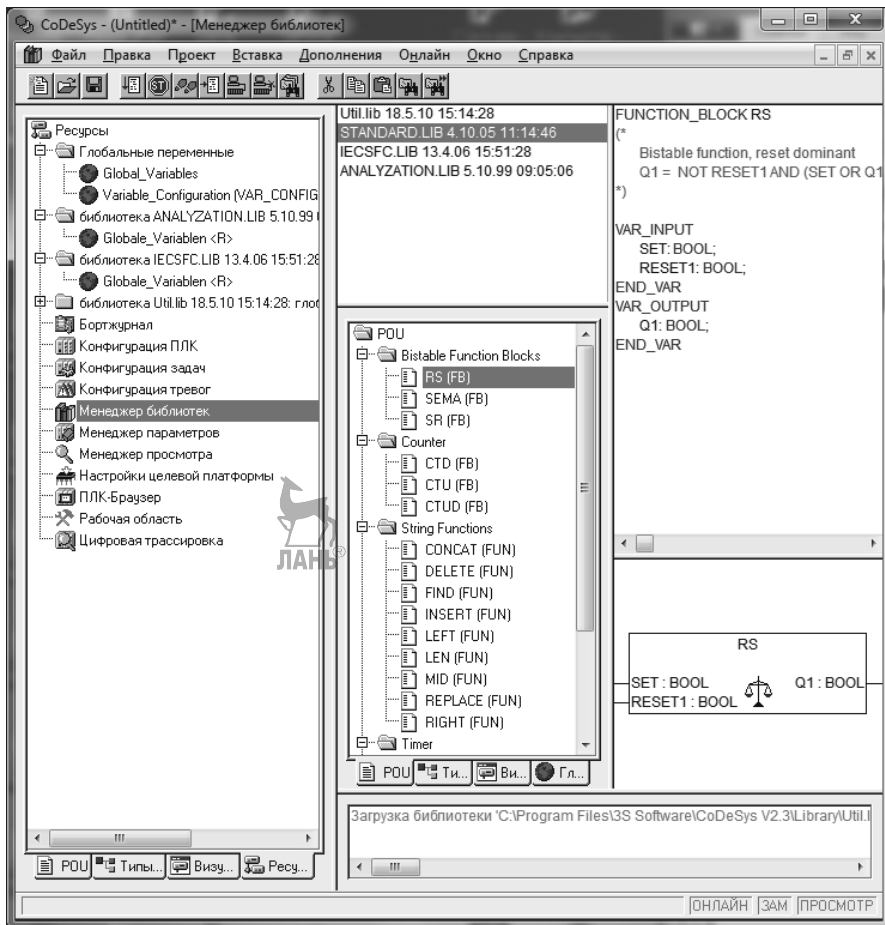


Рис. 7.6. Менеджер библиотек

Чтобы добавить в Codesys библиотеку, надо выбрать опции *Вставка > Добавить библиотеку*.

7.5. Обзор языков программирования

7.5.1. Язык ST

Для разработки прикладных программ контроллеров ОВЕН ПЛК используется интегрированная среда программирования Codesys. Она распространяется бесплатно и может быть без ограничений установлена на нескольких рабочих местах. В Codesys для программирования доступны следующие языки, определяемые стандартом IEC 61131-3 (МЭК 61131-3).

IL (Instruction List – *список инструкций*) — Ассемблер-подобный язык. По синтаксису напоминает ассемблер. Каждая инструкция начинается с новой строки и содержит оператор и, в зависимости от типа операции, один и более операндов, разделенных запятыми.

ST (Structured Text – *структурированный текст*) – Pascal-подобный язык. По структуре и синтаксису ближе всего к языку программирования Паскаль. Удобен для написания больших программ и работы с аналоговыми сигналами и числами с плавающей запятой.

LD (Ladder Diagram – *язык релейных диаграмм*) – графический язык релейных схем. Применяются также названия: язык релейно-контактной логики, релейные диаграммы, релейно-контактные схемы (РКС). Синтаксис языка удобен для описания логических схем, выполненных на релейной технике. Ориентирован на инженеров по автоматизации, работающих на промышленных предприятиях. Обеспечивает наглядный интерфейс логики работы контроллера.

FBD (Function Block Diagram – *язык функциональных блоковых диаграмм*) – графический язык функциональных блоков. Программа образуется путем создания цепей, выполняемых последовательно сверху вниз. При программировании используются наборы библиотечных блоков и собственные блоки программиста. Блоками могут быть подпрограмма, функция, функциональные блоки, логические функции И, ИЛИ, НЕ, триггеры, таймеры, счётчики, блоки обработки аналогового сигнала, математические операции и др.

SFC (Sequential Function Chart – *язык последовательных функциональных схем*) – графический язык, который позволяет описать хронологическую последовательность различных действий в программе. С этой целью, действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами.

CFC (Continuous Function Chart – *язык непрерывных функциональных схем*) – графический язык, который является вариантом языка FBD. Позволяет произвольно задавать порядок выполнения блоков. В отличие от FBD не использует цепи и дает возможность свободно размещать компоненты и соединения, что позволяет создавать обратные связи. Возможность создавать обратные связи является основным преимуществом CFC перед FBD.

Наиболее распространенными при программировании контроллеров являются языки ST, LD и CFC. В дальнейшем будут рассматриваться только эти языки.

Ознакомление с языками начнем с языка ST. Интерфейсное окно Codesys на языке ST показано на рис. 7.7. В зависимости от выбранного языка интерфейсное окно может отличаться видом панели инструментов. Панель инструментов языка ST приведена на рис. 7.8, а расшифровка кнопок панели в таблице 7.2.

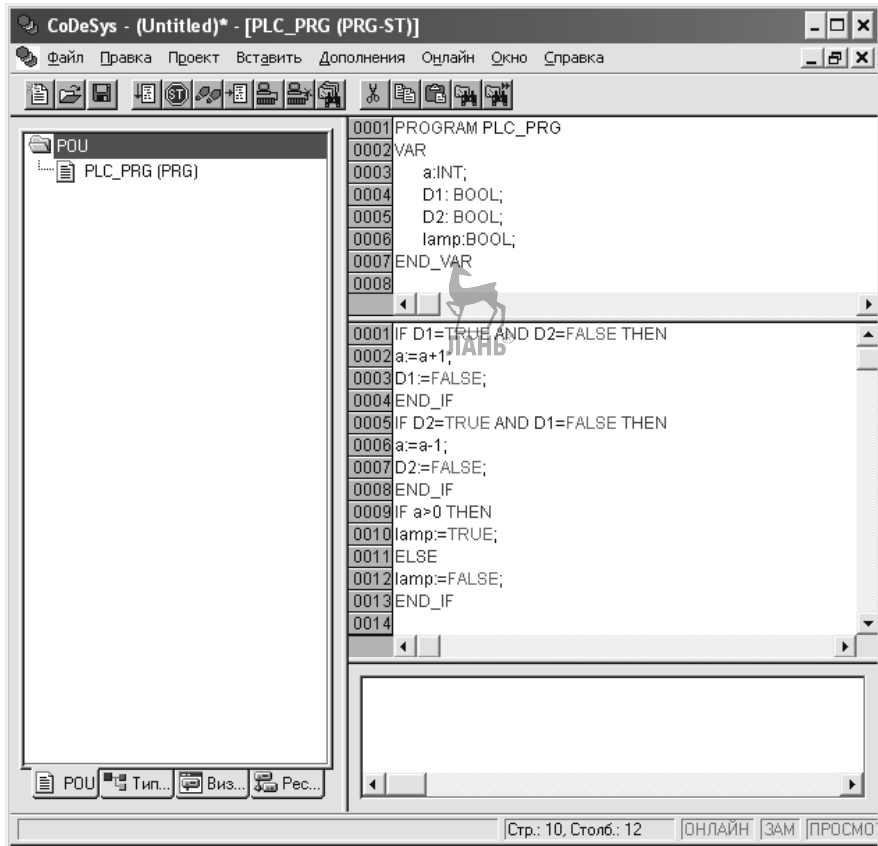


Рис. 7.7. Пример программы на языке ST

Поскольку программный код внутри рисунков плохо читается, будем повторять его под рисунком более крупным шрифтом.

```

PROGRAM PLC_PRG
VAR
    a:INT;
    D1:BOOL;
    D2:BOOL;
    lamp:BOOL;
END_VAR

IF D1=TRUE AND D2=FALSE THEN
a:=a+1;
D1:=FALSE;
END_IF
IF D2=TRUE AND D1=FALSE THEN

```

```
a:=a+1;
D2:=FALSE;
END_IF
IF a>0 THEN
lamp:=TRUE;
ELSE
lamp:=FALSE;
END_IF
```



Рис. 7.8. Панель инструментов языка ST

Таблица 7.2

1- Создать	9 - Отключение
2 - Открыть	10 – Глобальный поиск
3 - Сохранить	11 - Вырезать
4 - Старт	12 - Копировать
5 - Стоп	13 - Вставить
6 – Шаг поверху	14 - Найти
7 – Переключить точку останова	15 – Найти далее
8 - Подключение	

ST представляет собой аналог языка высокого уровня. В состав языка входят выражения, условные операторы if, операторы цикла for и while, оператор case и др.

Вычисление выражений. Вычисление выражений выполняется согласно правилам приоритета. Оператор с самым высоким приоритетом выполняется первым, оператор с более низким приоритетом - вторым и т.д. Операторы с одинаковым приоритетом выполняются слева направо.

В таблице 7.3. приведен список ST операторов, расположенных в порядке убывания приоритета сверху вниз.

Таблица 7.3

Операция	Обозначение
Вызов функции	Имя функции (список параметров)
Возведение в степень	EXPT(a,b) a – основание b – степень
Квадратный корень	SQRT(a)
Замена знака	/ +
Числовое дополнение	NOT
Умножение	*
Деление	/
Абсолютная величина	MOD
Сложение	+ ЛАНЬ®
Вычитание	–
Сравнение	<; >; <=; >=
Равенство	=
Неравенство	<; >
Логическое И	AND
Логическое исключаяющее ИЛИ	XOR
Логическое ИЛИ	OR

Присваивание значений в выражениях осуществляется с помощью знаков двоеточие и равно, например, A:=B; CV := CV + 1; C:=SIN(X); В конце выражения необходимо ставить точку с запятой.

Оператор IF. Используя инструкцию IF, можно проверить условие, и в зависимости от этого условия выполнить какие-либо действия. Например, фрагмент программы с оператором if запишется следующим образом

```
a:=10;
b:=20;
IF a<b THEN
c:= a+b;
d:=a-b;
ELSE
c :=a*b;
d:=b/a;
END_IF
```

Как видно из примера, оператор IF должен заканчиваться строчкой END_IF. Точку с запятой в начальной и конечной строке оператора ставить не надо. Программу

можно набирать как большими, так и малыми буквами. Если программный код, следующий за оператором `if`, состоит из нескольких строк, то его выделять фигурными скобками, как в языке C, не надо. Программный код будет выполняться до строчки, содержащей оператор `END_IF`.

Цикл FOR. С помощью оператора FOR можно программировать повторяющиеся процессы. Записать фрагмент программы с оператором цикла можно следующим образом:

```
sum:=0;
FOR i:=0 TO 10 BY 1 DO
sum:=sum+a;
END_FOR
```



здесь `i=0` – начальное значение переменной цикла,

`10` – конечное значение переменной цикла,

`1` – шаг цикла. Шаг цикла может принимать любое целое значение. По умолчанию шаг устанавливается равным 1. И в этом случае оператор цикла можно записать более коротким образом:

```
FOR i:=0 TO 10 DO
```

Оператор цикла должен заканчиваться строчкой `END_FOR`.

В приведенном фрагменте вычисляется сумма чисел от 0 до 10. Значение суммы записывается в переменную `sum`. В конце цикла переменная цикла `i` принимает значение 11, переменная `sum` принимает значение 55. При этом выполнение оператора цикла заканчивается, и программа выходит из цикла.

Программа с оператором цикла, записанная в среде Codesys, представлена на рис. 7.9, а результат выполнения программы на рис. 7.10.

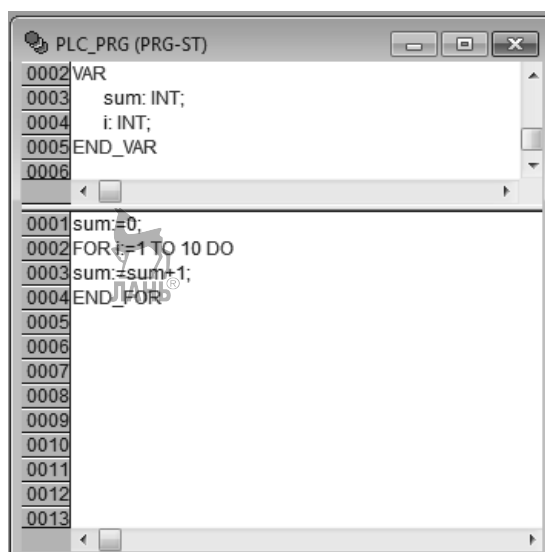


Рис. 7.9. Программа с оператором цикла

Чтобы получить результат, программу вначале пускаем на компиляцию *Проект > Компилировать (F11)*, затем на исполнение *Онлайн > Режим эмуляции > Подключение (Alt+F8) > Старт (F5)*.

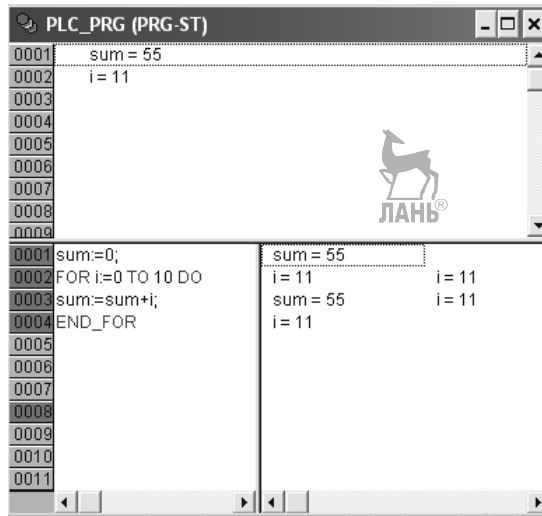


Рис. 7.10. Результат выполнения программы с оператором цикла

Если внутри цикла вставлен оператор **EXIT**, то цикл заканчивает свою работу независимо от значения условия выхода, как показано на рис. 7.11. При этом значение *i*, включенное в условие окончания вычислений, учитывается в расчетах, т.е. является последним значением цикла.

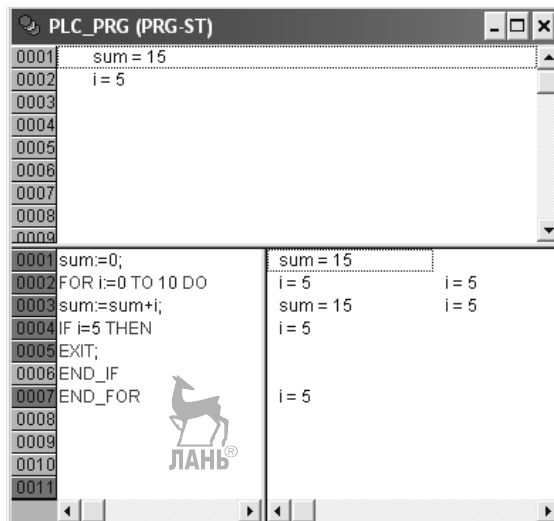


Рис. 7.11. Результат выполнения программы с оператором Exit

```

Текст программы:
sum:=0;
FOR i:=0 TO 10 DO
sum:=sum+i;
IF i=5 THEN
EXIT;
END_IF
END_FOR

```

Приведем пример заполнения массива значениями с помощью оператора цикла. Программа показана на рис. 7.12, а результат выполнения программы на рис. 7.13. Массив состоит из шести элементов, первый элемент массива имеет значение 1, остальные элементы задаются с шагом =2. Программа выходит из цикла, когда переменная цикла *i* получает значение =6.

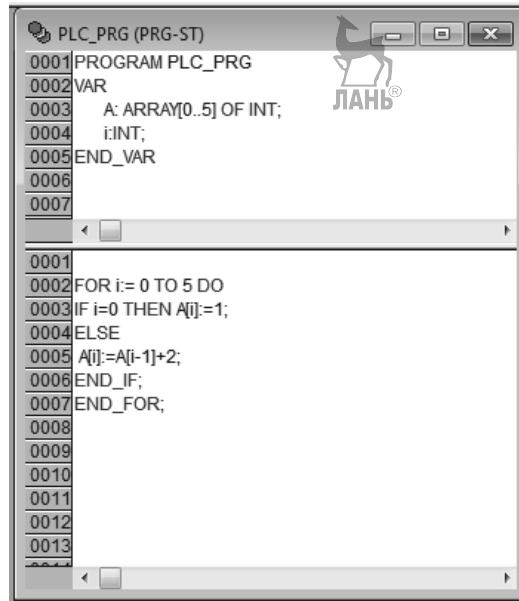


Рис. 7.12. Программа заполнения массива

```

PROGRAM PLC_PRG
VAR
    A:ARRAY[0..5] OF INT;
    i:INT;
END_VAR

FOR i:=0 TO 5 DO

```

```

IF i=0 THEN A[i]:=1;
ELSE
A[i]:=A[i-1]+2;
END_IF
END_FOR

```

Точку с запятой после операторов END_IF и END_FOR можно не ставить. Пробелы в операторах END_IF и END_FOR не допускаются.

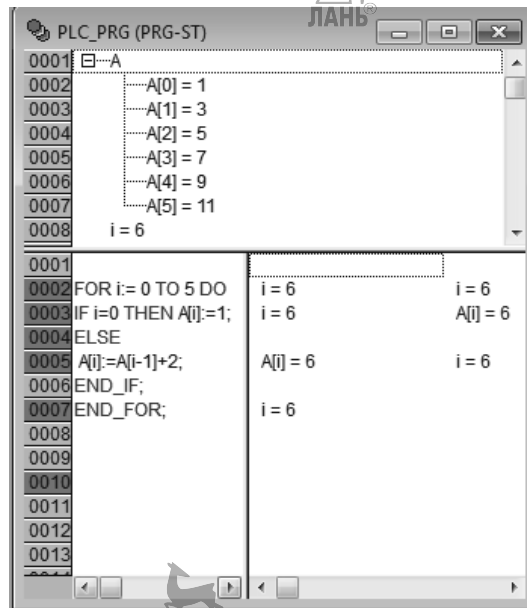


Рис. 7.13. Результат выполнения программы заполнения массива

Цикл WHILE. Цикл WHILE может использоваться, как и цикл FOR, с тем лишь различием, что условие выхода определяется логическим выражением. Это означает, что цикл выполняется, пока выполняется заданное условие.

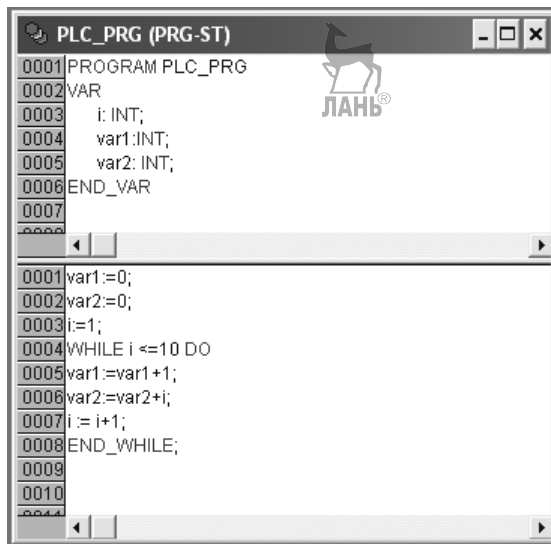


Рис. 7.14. Программа с циклом While

```
PROGRAM PLC_PRG
VAR
    i:=INT;
    var1:=INT;
    var2:=INT;
END_VAR
```

```
var1:=0;
var2:=0;
i:=1;
WHILE i<=10 DO
var1:=var1+1;
var2:=var2+i;
i:=i+1;
END_WHILE
```



```

0001 i = 11
0002 var1 = 10
0003 var2 = 55
0004
0005
0006
0007
0008
0009
0010
0011
0012

```

```

0001 var1:=0; var1 = 10
0002 var2:=0; var2 = 55
0003 i:=1; i = 11
0004 WHILE i<=10 DO i = 11
0005 var1:=var1+1; var1 = 10
0006 var2:=var2+i; var2 = 55 i = 11
0007 i:=i+1; i = 11
0008 END_WHILE
0009
0010
0011
0012

```

Стр.: 4, Столб.: 10 ОНЛАЙН ЭМУЛ. ЗАПУЩЕН

Рис. 7.15. Результат выполнения программы с циклом While

Синтаксис оператора цикла WHILE:

```

WHILE <Boolean expression>
<Instructions>
END_WHILE

```

Раздел <Instructions> выполняется циклически до тех пор, пока <Boolean_expression> дает TRUE. Если <Boolean_expression> равно FALSE уже при первой итерации, то раздел <Instructions> не будет выполнен ни разу. Если <Boolean_expression> никогда не примет значение FALSE, то раздел <Instructions> будет выполняться бесконечно.

Чтобы цикл не сделался бесконечным необходимо в теле цикла менять значение входящей в условие переменной, например, $i:=i+1$. Программа с циклом WHILE представлена на рис. 7.14, а результат выполнения программы представлен на рис. 7.15. Цикл выполняется до тех пор, пока не выполнится условие $i \leq 1$, т.е. пока переменная цикла не примет значение $i=11$, при этом переменная var1 принимает значение =10, а var2 принимает значение =55.

Оператор CASE. С помощью оператора CASE можно различным значениям целочисленной переменной сопоставить различные блоки программы.

Синтаксис оператора Case:

```

CASE <Var1> OF
<Value1>: <Instruction 1>
<Value2>: <Instruction 2>
<Value3, Value4, Value5>: <Instruction 3>
<Value6 .. Value10>: <Instruction 4>
...

```

```

<Value n>: <Instruction N>
ELSE <Instruction M>
END_CASE;

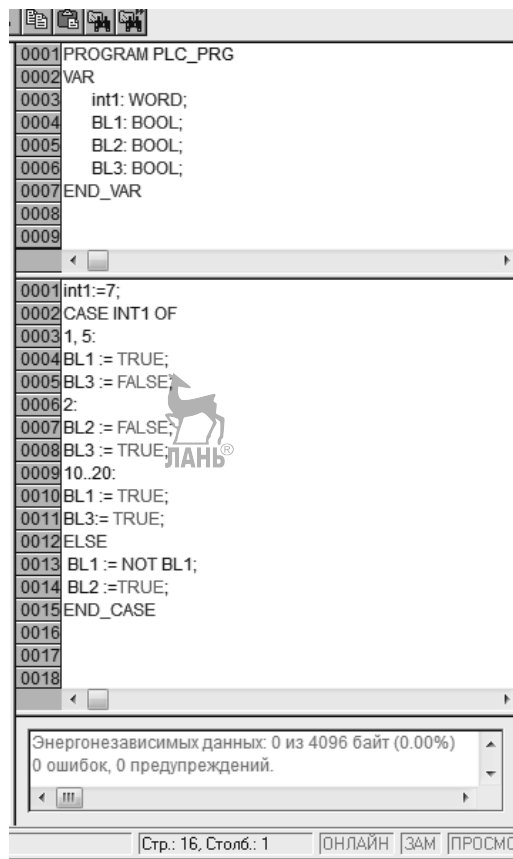
```

Представленный фрагмент программы работает следующим образом. Если переменная <Var1> имеет значение <Value i>, то выполняется инструкция <Instruction i>. Если <Var1> не принимает ни одного из указанных значений, то выполняется <Instruction M>.

Чтобы одна и та же инструкция выполнялась при различных значениях переменной <Var1>, необходимо перечислить эти значения через запятую.

Чтобы одна и та же инструкция выполнялась для целого диапазона значений, необходимо указать начальное и конечное значения, разделенные двумя точками.

Пример программы с оператором CASE показан на рис. 7.16.



```

0001 PROGRAM PLC_PRG
0002 VAR
0003   int1: WORD;
0004   BL1: BOOL;
0005   BL2: BOOL;
0006   BL3: BOOL;
0007 END_VAR
0008
0009
0010 int1:=7;
0011 CASE INT1 OF
0012 1, 5:
0013   BL1 := TRUE;
0014   BL3 := FALSE;
0015 2:
0016   BL2 := FALSE;
0017 10..20:
0018   BL1 := TRUE;
0019   BL3 := TRUE;
0020 ELSE
0021   BL1 := NOT BL1;
0022   BL2 := TRUE;
0023 END_CASE
0024
0025
0026
0027

```

Энергонезависимых данных: 0 из 4096 байт (0.00%)
0 ошибок, 0 предупреждений.

Стр.: 16, Столб.: 1 | ОНЛАЙН | ЗАМ | ПРОСМО

Рис. 7.16. Программа с оператором CASE

```

PROGRAM PLC_PRG
VAR

```

```
int1:=WORD;
BL1:BOOL;
BL2:BOOL;
BL3:BOOL;
END_VAR

int1:=7;
CASE INT1 OF
1,5:
BL1:=TRUE;
BL3:=FALSE;
2:
BL1:=FALSE;
BL3:= TRUE;
10..20:
BL1:=TRUE;
BL3:= TRUE;
ELSE
BL1:=NOT BL1;
BL2:=TRUE;
END_CASE
```



При `int1:=7` будет выполняться последний блок программы после оператора `ELSE`. Так как программа выполняется циклически, то переменная `BL1` будет принимать поочередно значения `TRUE` и `FALSE`, переменная `BL2` будет всегда иметь значение `TRUE`, переменная `BL3` будет всегда иметь значение `FALSE`.

Чтобы проследить выполнение программы, можно воспользоваться точками останова и пошаговым выполнением программы. Точку останова можно назначить после перехода в режим *Подключение*. Для этого надо щелкнуть левой кнопкой мыши в левом столбце по нужной строке, либо назначить строку останова в меню *Онлайн > Диалог точек останова >* и в выпадающем окне выбрать строку *Расположение*. Далее запустить программу на исполнение командой *Старт*. Выполнение программы остановится перед указанной строкой останова. Далее можно продолжить выполнение программы по шагам, нажимая клавишу `F8`.

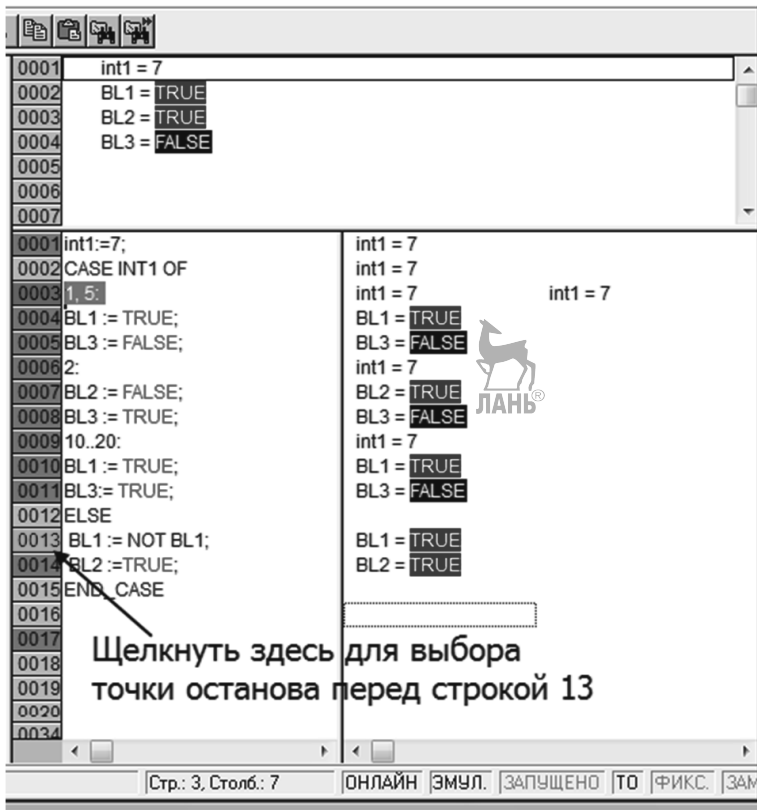


Рис. 7.17. Выбор точки останова

Удалить точку останова можно в выпадающем окне, пройдя к нему по пунктам меню *Онлайн > Диалог точек останова*.

7.5.2. Примеры управляющих программ на языке ST

Пример 7.1. Напишем на языке ST программу для вычисления корней квадратного уравнения

$$2x^2 + 12x + 10 = 0$$

Программа показана на рис. 7.18.



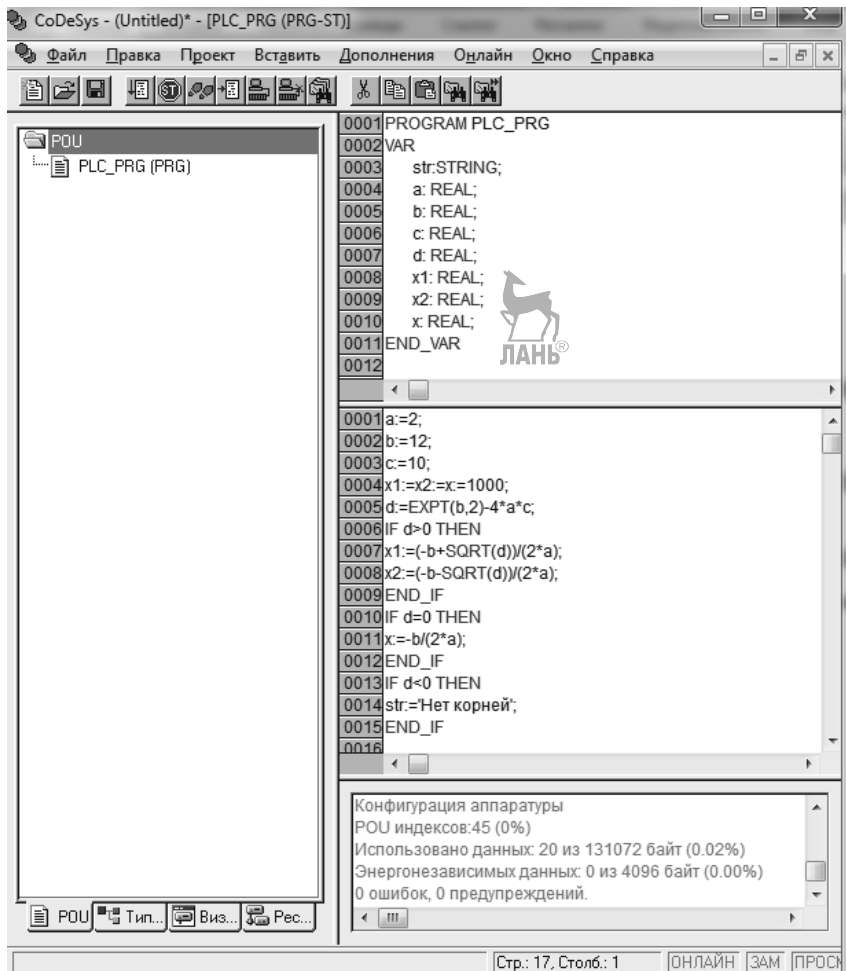


Рис. 7.18. Программа на языке ST для вычисления корней квадратного уравнения

```
PROGRAM PLC_PRG
VAR
  str:STRING;
  a:REAL;
  b:REAL;
  c:REAL;
  d:REAL;
  x1:REAL;
  x2:REAL;
  x:REAL;
END_VAR
```




```

a:=2;
b:=12;
c:=10;
x1:=x2:=x:=1000;
d:=EXPT(b,2)-4*a*c;
IF d>0 THEN
x1:=(-b+SQRT(d))/(2*a);
x2:=(-b-SQRT(d))/(2*a);
END_IF
IF d=0 THEN
x:=-b/(2*a);
END_IF
IF d<0 THEN
str:='Нет корней';
END_IF

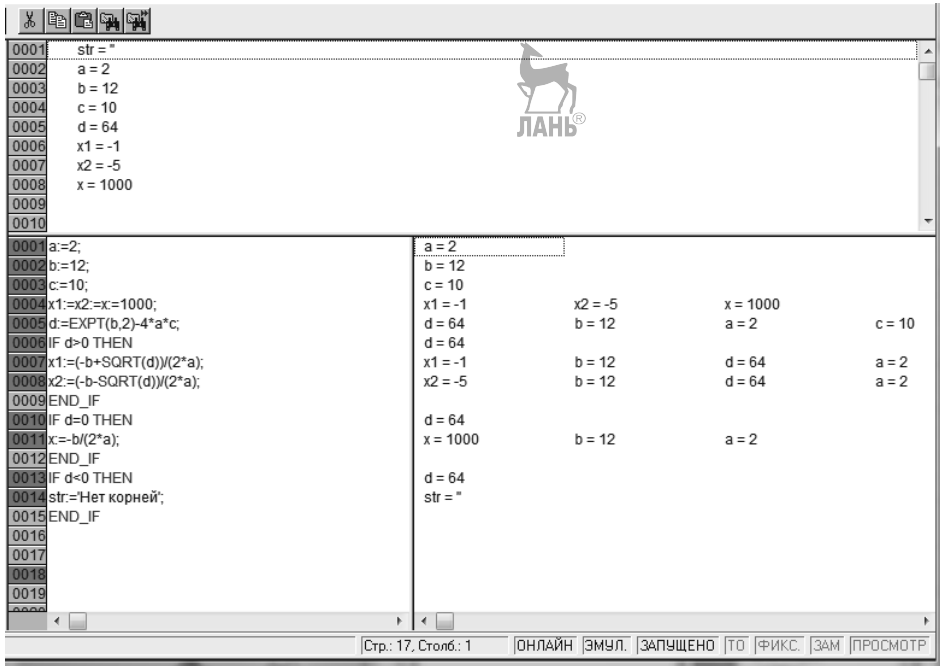
```



При написании программы должна быть заполнена верхняя часть рабочей области – область определения переменных. Но лучше начинать создание программы не с заполнения этой области, а сразу с написания текста программы, потому что в процессе написания текста программы эта область будет заполняться автоматически, когда будет вводиться новая переменная в программе. Чтобы задать тип переменной в программе, надо каждый раз при вводе переменной вызывать *Ассистент ввода* комбинацией клавиш Shift+F2, при этом будет появляться окно, в котором надо задавать тип переменной.

После того, как программа написана, ее нужно скомпилировать. Для этого надо выбрать *Проект > Компилировать*. Если в программе есть ошибки, они отобразятся в *Окне сообщений* (поз. 5, рис. 7.3). Можно дважды щелкнуть по ошибке левой кнопкой мыши, тогда курсор перескочит на строчку программы, в которой сделана ошибка. Если ошибок в программе нет, то в окне сообщений появится запись «0 ошибок, 0 предупреждений», как показано на рис. 7.18. После этого надо зайти во вкладку меню *Онлайн* и выбрать строки *Режим эмуляции, Подключение, Старт (F5)*. Появится окно с результатами вычислений (рис. 7.19), а в статусной строке этого окна будет присутствовать запись *Запущено*. При заданных в программе коэффициентах (a, b, c) дискриминант будет положительным, сработает первый условный оператор (остальные будут пропущены) и будет вычислено два корня: $x1 = -1$, $x2 = -5$. Для переменных (x, x1, x2) было задано начальное значение =1000. Если не задать никаких начальных значений, то, как правило, переменная имеет начальное значение =0.





```

0001 str = ""
0002 a = 2
0003 b = 12
0004 c = 10
0005 d = 64
0006 x1 = -1
0007 x2 = -5
0008 x = 1000
0009
0010
0001 a:=2;
0002 b:=12;
0003 c:=10;
0004 x1:=x2=-x=1000;
0005 d:=EXPT(b,2)-4*a*c;
0006 IF d>0 THEN
0007 x1:=(-b+SQRT(d))/(2*a);
0008 x2:=(-b-SQRT(d))/(2*a);
0009 END_IF
0010 IF d=0 THEN
0011 x=-b/(2*a);
0012 END_IF
0013 IF d<0 THEN
0014 str:="Нет корней";
0015 END_IF
0016
0017
0018
0019

```

a = 2				
b = 12				
c = 10				
x1 = -1	x2 = -5	x = 1000		
d = 64	b = 12	a = 2	c = 10	
d = 64				
x1 = -1	b = 12	d = 64	a = 2	
x2 = -5	b = 12	d = 64	a = 2	
d = 64				
x = 1000	b = 12	a = 2		
d = 64				
str = ""				

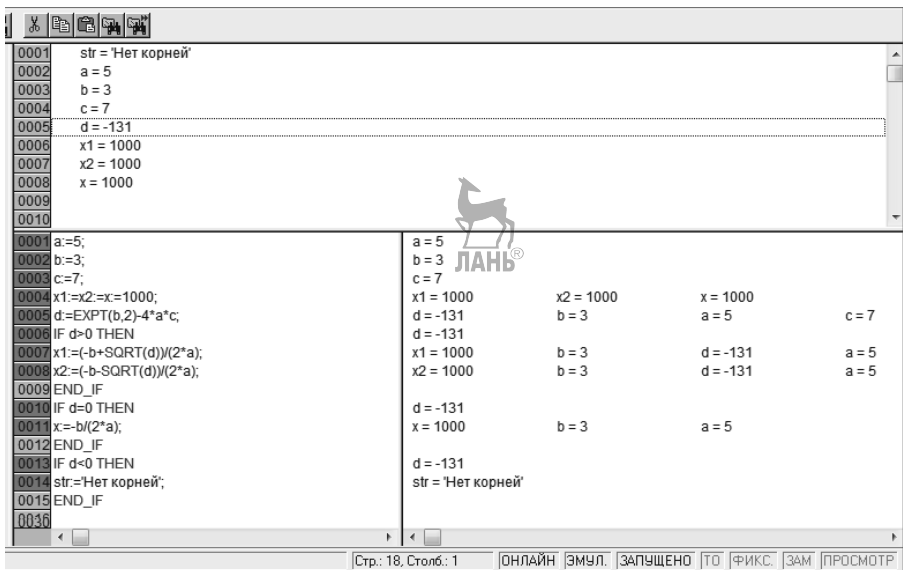
Стр.: 17, Столб.: 1 ОНЛАЙН ЭМУЛ. ЗАПУЩЕНО ТО ФИКС. ЗАМ ПРОСМОТР

Рис. 7.19. Результат выполнения программы с дискриминантом больше нуля

Если назначить коэффициентам (a, b, c) соответственно значения (5, 3, 7), то дискриминант будет отрицательным, сработает третий условный оператор и будет выведено значение переменной типа STRING ('нет корней'), как показано на рис. 7.20.

Чтобы закончить выполнение программы, надо выбрать опции *Онлайн > Отключение*.





```

0001 str = 'Нет корней'
0002 a = 5
0003 b = 3
0004 c = 7
0005 d = -131
0006 x1 = 1000
0007 x2 = 1000
0008 x = 1000
0009
0010
0001 a=5;
0002 b=3;
0003 c=7;
0004 x1:=x2:=x:=1000;
0005 d:=EXPT(b,2)-4*a*c;
0006 IF d>0 THEN
0007 x1:=(-b+SQRT(d))/(2*a);
0008 x2:=(-b-SQRT(d))/(2*a);
0009 END_IF
0010 IF d=0 THEN
0011 x:=b/(2*a);
0012 END_IF
0013 IF d<0 THEN
0014 str:='Нет корней';
0015 END_IF
0036

```

Стр.: 18, Столб.: 1 | ОНЛАЙН | ЭМУЛ. | ЗАПУЩЕНО | ТО | ФИКС. | ЗАМ | ПРОСМОТР

Рис. 7.20. Результат выполнения программы с дискриминантом меньше нуля

Пример 7.2. Вернемся к задаче «Резервуар с водой», рассмотренной в предыдущей главе, и напишем для нее программу на языке ST. Реализация этой задачи с помощью языка ST позволяет более точно отразить логику работы установки.

Итак, имеется резервуар, в который вода поступает через верхнюю трубу, и откачивается из резервуара через нижнюю трубу (рис. 7.21).

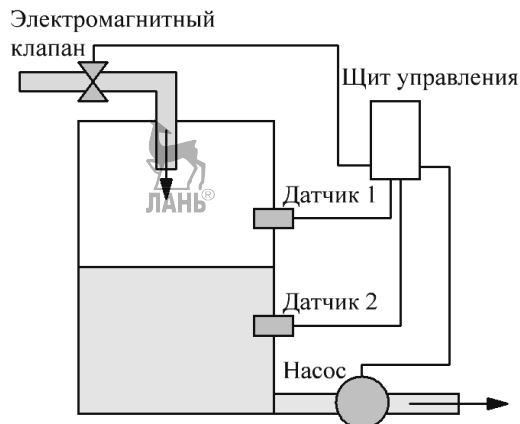


Рис. 7.21. Схема установки

Верхняя труба открывается и закрывается электромагнитным клапаном. Внутри резервуара находятся два датчика уровня воды. Логика работы установки следующая. В начальный момент в резервуаре нет воды. При нажатии кнопки

Пуск электромагнитный клапан открывает трубу, в резервуар поступает вода. Сначала срабатывает нижний датчик Д2 и переходит из состояния 0 в состояние 1. Верхняя труба при этом остается открытой, вода продолжает поступать в резервуар. Когда вода поднимется выше уровня датчика Д1, датчик Д1 срабатывает и переходит из состояния 0 в состояние 1. Если оба датчика Д1 и Д2 находятся в состоянии 1, срабатывает электромагнитный клапан (переходит из состояния 0 в состояние 1) и перекрывает верхнюю трубу. Подача воды в резервуар прекращается. Поскольку происходит постоянный расход воды из резервуара, уровень воды начинается понижаться. Сначала уровень воды опустится ниже датчика Д1 и датчик Д1 перейдет из состояния 1 в состояние 0. Затем вода опустится ниже датчика Д2 и датчик Д2 перейдет из состояния 1 в состояние 0. Когда оба датчика Д1 и Д2 будут находиться в состоянии 0, верхняя труба откроется, и вода вновь начнет поступать в резервуар. Далее процесс циклически повторяется. При нажатии кнопки *Смон* работа установки прекращается.

Различают комбинационные и последовательностные логические устройства. В комбинационных логических устройствах значения выходных сигналов зависят только от комбинации входных сигналов в данный момент времени. В последовательностных логических устройствах выходные сигналы зависят от значений входных сигналов не только в данный момент времени, но и в предыдущие моменты времени. В состав этих устройств входят элементы памяти – триггеры, в которых хранится информация о состоянии системы в предыдущие моменты времени.

Резервуар с водой относится к последовательностным логическим устройствам. Покажем это. Сигнал от датчика Д1 обозначим «А», сигнал от датчика Д2 обозначим «В», сигнал к электромагнитному клапану от логического устройства обозначим Y. Переменная «А» имеет значение 0 (False), когда уровень воды ниже датчика Д1, и имеет значение 1 (True), когда вода выше уровня датчика Д1. Аналогично переменная «В» имеет значение 0 (False), когда вода ниже уровня датчика Д2, и имеет значение 1 (True), когда вода выше уровня датчика. Выходная переменная Y принимает значение 0 (False), когда верхняя труба открыта, и принимает значение 1 (True), когда верхняя труба закрыта.

Значения логических переменных при изменении уровня воды в резервуаре, начиная от минимального, можно представить в виде таблицы 7.4.

Таблица 7.4

А	0	0	1	0
В	0	1	1	1
Y	0	0	1	1

Из таблицы 7.4 видно, если уровень воды находится между датчиками Д2 и Д1 («А»=0, «В»=1), то значение Y=0 при повышении уровня воды, и Y=1 при понижении уровня воды. Таким образом, выходная переменная Y принимает разные значения при одинаковых значениях «А» и «В». Чтобы различать эти два состояния, введем дополнительную логическую переменную «Z», значение которой «Z»=0 (False) при повышении уровня воды и «Z»=1 (True) при понижении

уровня воды. Дополним таблицу 7.4 строкой со значениями «Z» и получим таблицу 7.5.

Таблица 7.5

A	0	0	1	0
B	0	1	1	1
Y	0	0	1	1
Z	0	0	1	1

Программа имеет вид, представленный на рис. 7.22.

```

0001 PROGRAM PLC_PRG
0002 VAR
0003   A:BOOL;
0004   B:BOOL;
0005   y: BOOL;
0006   Z: BOOL;
0007 END_VAR
0008
0001 IF Z=FALSE THEN
0002 IF (A=FALSE AND B=FALSE) OR (B=TRUE AND A=FALSE) THEN
0003 y:=FALSE;
0004 Z:=FALSE;
0005 END_IF
0006 END_IF
0007 IF A=TRUE AND B=TRUE THEN
0008 y:=TRUE;
0009 Z:=TRUE;
0010 END_IF
0011 IF Z=TRUE THEN
0012 IF B=TRUE AND A=FALSE THEN
0013 y:=TRUE;
0014 END_IF
0015 END_IF
0016 IF Z=TRUE THEN
0017 IF A=FALSE AND B=FALSE THEN
0018 Z:=FALSE;
0019 END_IF
0020 END_IF
0021
0022

```

Рис. 7.22. Программа «Резервуар с водой»

```

PROGRAM PLC_PRG
VAR
  A:BOOL;
  B:BOOL;
  y: BOOL;
  Z: BOOL;
END_VAR

```

```
IF Z=FALSE THEN
IF (A=FALSE AND B=FALSE) OR (B=TRUE AND A=FALSE) THEN
y:=FALSE;
Z:=FALSE;
END_IF
END_IF
IF A=TRUE AND B=TRUE THEN
y:=TRUE;
Z:=TRUE;
END_IF
IF Z=TRUE THEN
IF B=TRUE AND A=FALSE THEN
y:=TRUE;
END_IF
END_IF
IF Z=TRUE THEN
IF A=FALSE AND B=FALSE THEN
Z:=FALSE;
END_IF
END_IF
```



Для того, чтобы проследить, как работает программа, будем выполнять ее по шагам. Для этого проставим точки останова во всех строчках программы (кроме строчек END_IF), щелкая левой кнопкой мыши по номерам строк в левом столбце. Точки останова можно назначать после перехода программы в режим *Онлайн > Подключение*. Далее запустим программу на исполнение командой *Старт*.



```

0001 A = FALSE
0002 B = TRUE
0003 y = TRUE
0004 Z = TRUE
0005
0006
0007
0001 IF Z=FALSE THEN
0002 IF (A=FALSE AND B=FALSE) OR (B=TRUE AND A=FALSE) THEN
0003 y:=FALSE;
0004 Z:=FALSE;
0005 END_IF
0006 END_IF
0007 IF A=TRUE AND B=TRUE THEN
0008 y:=TRUE;
0009 Z:=TRUE;
0010 END_IF
0011 IF Z=TRUE THEN
0012 IF B=TRUE AND A=FALSE THEN
0013 y:=TRUE;
0014 END_IF
0015 END_IF
0016 IF Z=TRUE THEN
0017 IF A=FALSE AND B=FALSE THEN
0018 Z:=FALSE;
0019 END_IF
0020 END_IF
0021
0022
0023

```

Variable values during execution:

- Z = TRUE
- A = FALSE
- y = TRUE
- Z = TRUE
- B = TRUE
- A = FALSE
- y = TRUE
- Z = TRUE
- Z = TRUE
- B = TRUE
- A = FALSE
- y = TRUE
- Z = TRUE
- A = FALSE
- B = TRUE
- Z = TRUE

Рис. 7.23. Результат выполнения программы

Выполнение программы будет происходить построчно. Очередной пуск программы после останова осуществляется нажатием клавиши F8. Чтобы управлять работой программы, надо менять состояние датчиков Д1 и Д2. Для этого дважды щелкаем в верхней области программы (в области задания переменных) по переменной «А» или «В». В соответствии с логикой работы установки, сначала переменные «А» и «В» находятся в состоянии False. При этом открывается верхняя труба и резервуар начинает наполняться. Нижний датчик Д2 (переменная «В») переходит в состояние True. Чтобы перевести переменную «В» в состояние True, дважды щелкаем по ней левой кнопкой мыши. Правее появляется значение TRUE. Чтобы записать это значение в программу, во вкладке *Онлайн* выбираем строчку *Записать значение* (или нажимаем комбинацию клавиш Ctrl+F7). Далее выполняем программу построчно и наблюдаем за значениями переменных. Затем переводим в состояние True переменную «А», выполняем программу построчно и наблюдаем за значениями переменных и т.д. Убеждаемся, что программа работает в соответствии с заданным алгоритмом, т.е. верхняя труба открывается, когда уровень воды ниже датчика Д2 и закрывается, когда уровень воды поднимается выше датчика Д1. Для корректной работы программа должна стартовать, когда обе переменные «А» и «В» имеют значение False и далее меняются в соответствии с логикой работы установки. На рис. 7.23 показан результат работы программы, когда идет снижение уровня воды.

После того как работа программы проверена на симуляторе, программа может быть загружена в память контроллера.

Пример 7.3. Пусть имеется конвейер (рис. 7.24), по которому перемещаются детали разной высоты. Высоту деталей контролируют два датчика D1 и D2. Датчик D1 определяет минимально допустимую высоту и датчик D2 – максимально допустимую высоту. Если высота детали выходит за пределы допустимого диапазона, то срабатывает боковой толкатель и сбрасывает деталь в ящик с браком.

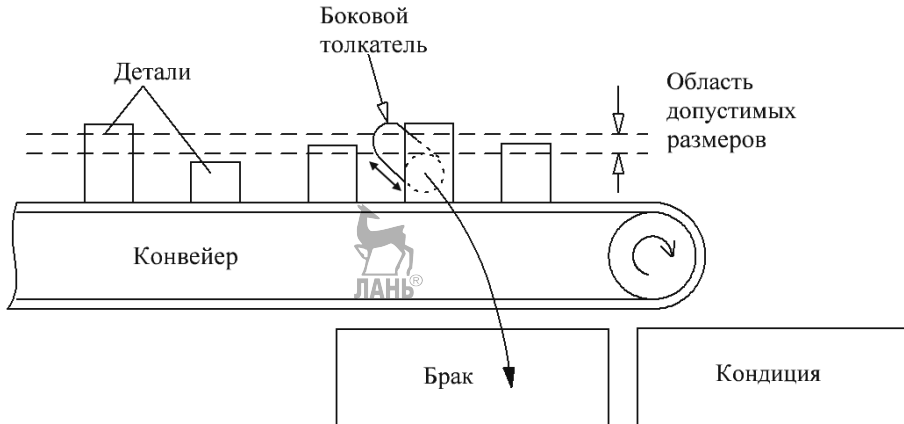


Рис. 7.24. Схема установки

Свяжем датчик D1 с входом In1 контроллера, датчик D2 с входом In2 и выход контроллера Out1 свяжем с толкателем. Для этого перейдем в левой части интерфейсного окна на вкладку *Ресурсы* и для выбранного контроллера присвоим входам и выходам наименования In1, In2, Out1, как показано на рис. 7.25.

Итак, имеем:

- 1) Датчик D1 (вход IN1): если D1=False, то деталь меньше нижней границы допустимой области, D1=TRUE – деталь выше нижней границы.
- 2) Датчик D2 (вход IN2): если D2=False, то деталь ниже верхней границы, D2=TRUE – деталь выше верхней границы.
- 3) Толкатель Out1: Out1=TRUE – толкатель сработал и столкнул деталь в брак, Out1=FALSE – толкатель пропустил нормальную деталь.



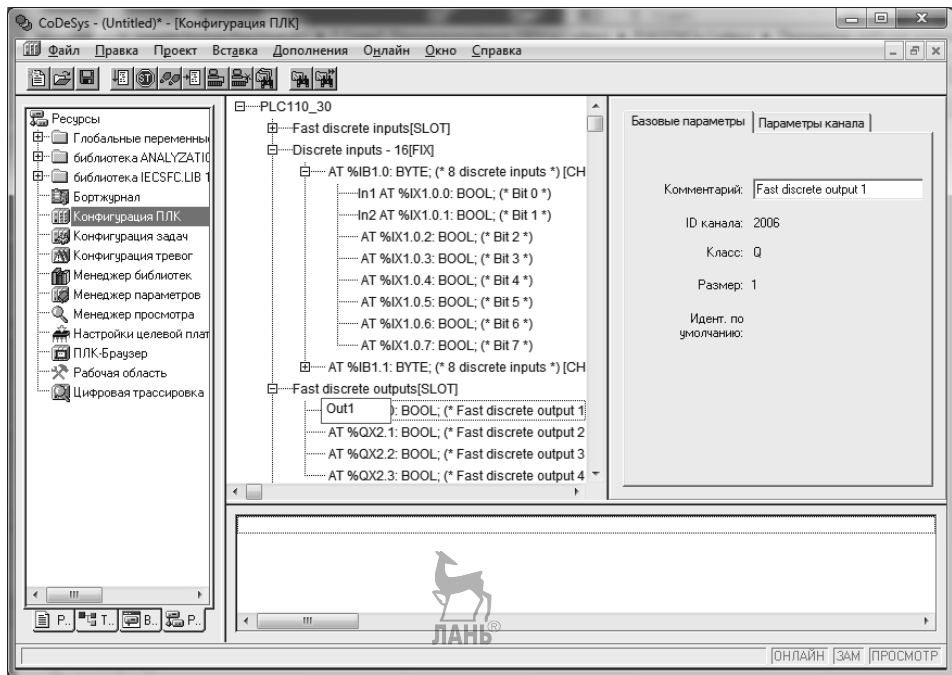


Рис. 7.25. Присваивание наименований переменных входам и выходам контроллера

Кроме переменных, связанных с входами и выходами контроллера, введем внутренние переменные программы, которые будут накапливать кол-во деталей соответствующего размера:

normal – кол-во нормальных деталей;
 low – кол-во деталей меньшей высоты;
 high – кол-во деталей большей высоты.

Понятно, что детали меньшей высоты – это уже неисправимый брак.

Программа показана на рис. 7.26.

После запуска программы на исполнение можно менять булевы значения датчиков и наблюдать за накоплением деталей разного вида.



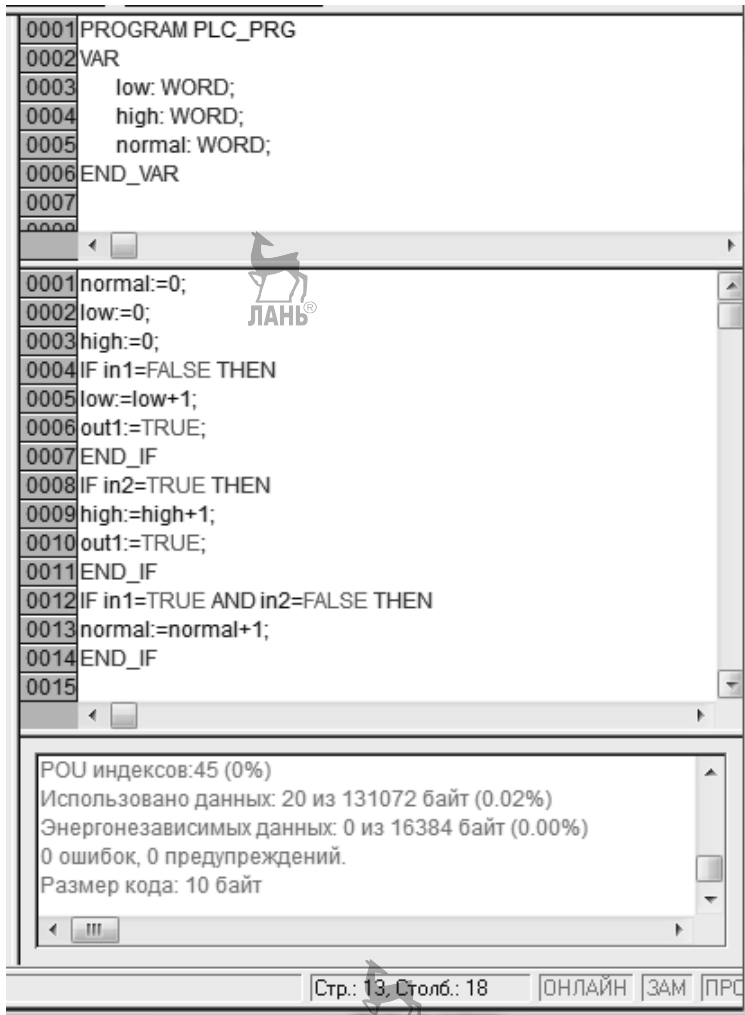


Рис. 7.26. Программа

```

PROGRAM PLC_PRG
VAR
    low:WORD;
    high:WORD;
    normal:WORD;
END_VAR

```

```

normal:=0;
low:=0;
high:=0;
IF in1=FALSE THEN

```

```

low:=low+1;
out1:=TRUE;
END_IF
IF in2=TRUE THEN
high:=high+1;
out1:=TRUE;
END_IF
IF in1=TRUE AND in2=FALSE THEN
normal:=normal+1;
END_IF

```



7.5.3. Язык LD

Язык LD – релейных диаграмм или релейно-контактных схем – графический язык, реализующий структуры электрических цепей. Лучше всего LD подходит для построения логических переключателей, но достаточно легко можно создавать и сложные цепи. Диаграмма LD состоит из последовательности цепей, каждая из которых содержит логическое или арифметическое выражение, функциональный блок, переход или инструкцию возврата. Слева и справа схема ограничена вертикальными линиями - шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями. Слева любая цепь начинается набором контактов, которые посылают слева направо состояние "ON" или "OFF", соответствующие логическим значениям ИСТИНА или ЛОЖЬ, и заканчивается катушкой (обмоткой) реле. Каждому контакту и катушке соответствует логическая переменная, задаваемая в области определения переменных.

Идеология релейных схем подразумевает параллельную работу всех цепей. Ток во все цепи подается одновременно. Выполнение программы происходит последовательно слева направо и сверху вниз. Если реле в цепи изменит значение переменной, то цепи, расположенные ниже, получают новое значение переменной сразу. Цепи, расположенные выше, получают новое значение только в следующем цикле.

Интерфейсное окно Codesys для языка программирования LD приведено на рис. 7.27, а панель инструментов на рис. 7.28. Первые пятнадцать значков (кнопок) панели совпадают с соответствующими кнопками интерфейсного окна для языка ST, поэтому в таблице 7.4 приведены названия только последующих кнопок, начиная с шестнадцатой.



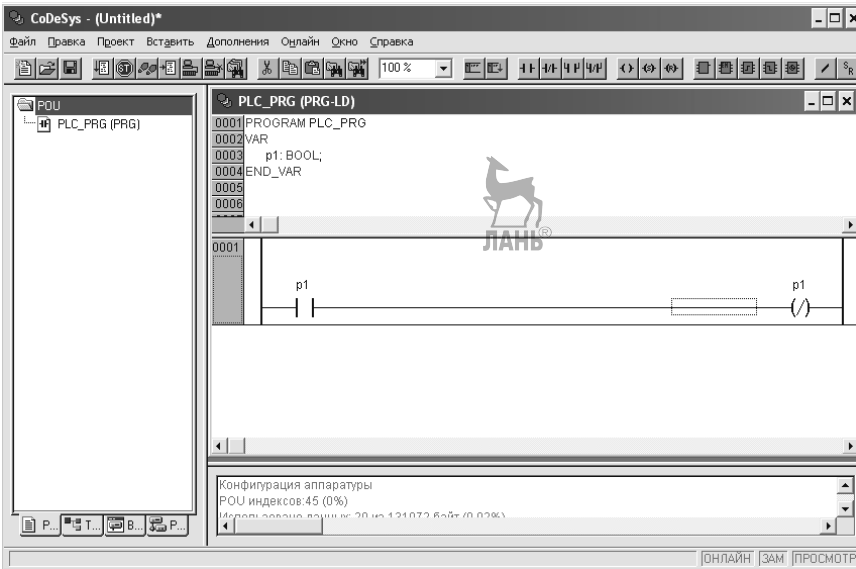


Рис. 7.27. Интерфейсное окно языка LD

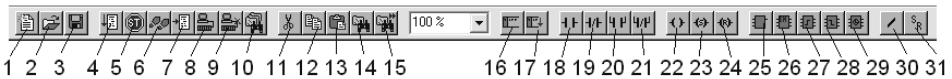


Рис. 7.28. Панель инструментов языка LD

Таблица 7.4

16 – Цепь перед	24 – Reset обмотка
17 – Цепь после	25 – Функциональный блок
18 – Контакт	26 – Элемент с EN
19 – Инверсный контакт	27 – Детектор переднего фронта
20 – Параллельный контакт	28 – Детектор заднего фронта
21 – Параллельный инверсный контакт	29 – Таймер (TON)
22 – Обмотка	30 – Инверсия
23 – Set обмотка	31 – Set/Reset

Порядок выполнения цепей диаграммы можно принудительно изменять, используя переходы. Используя переход, можно пропустить выполнение части диаграммы. Переход вверх позволяет создавать циклы.

Контакт обозначается двумя параллельными линиями (| |) и может иметь состояние ON или OFF. Состояние ON соответствует значению TRUE, состояние OFF соответствует значению FALSE. Контакты могут быть соединены параллельно или последовательно. Параллельное соединение контактов эквивалентно логическому элементу ИЛИ. Последовательное соединение контактов эквивалентно логическому элементу И.

Контакт может быть **инверсным**. Такой контакт обозначается с помощью символа (/). Состояние ON соответствует значению FALSE, состояние OFF соответствует значению TRUE. Работа контактов поясняется таблицей 7.5.

Таблица 7.5

Тип контакта		Состояние контакта	Логическое значение	Ток
	Нормально разомкнутый (NO)	Разомкнутый	FALSE	Не пропускает
		Замкнутый	TRUE	Пропускает
	Нормально замкнутый (инверсный) (NC)	Замкнутый	FALSE	Пропускает
		Разомкнутый	TRUE	Не пропускает

Таким образом, **начальное** состояние контакта – это всегда значение **FALSE** и **конечное** состояние – это всегда значение **TRUE**. Но разница между контактами в том, что в начальном состоянии контакт NO разомкнут и не пропускает ток, а контакт NC замкнут и пропускает ток.

Обмотки. В правой части схемы может находиться любое количество обмоток (реле), которые обозначаются круглыми скобками (). Они могут соединяться только параллельно. Обмотка может быть либо замкнутой (ON), т.е. находиться в состоянии TRUE, либо может быть разомкнутой (OFF), т.е. находиться в состоянии FALSE. Обмотки также могут быть инверсными, обозначаются (/). Инверсные контакты и обмотки равнозначны логической операции НЕ.

Для понимания логики работы контактов и обмоток создадим LD программу, представленную на рис. 7.29.



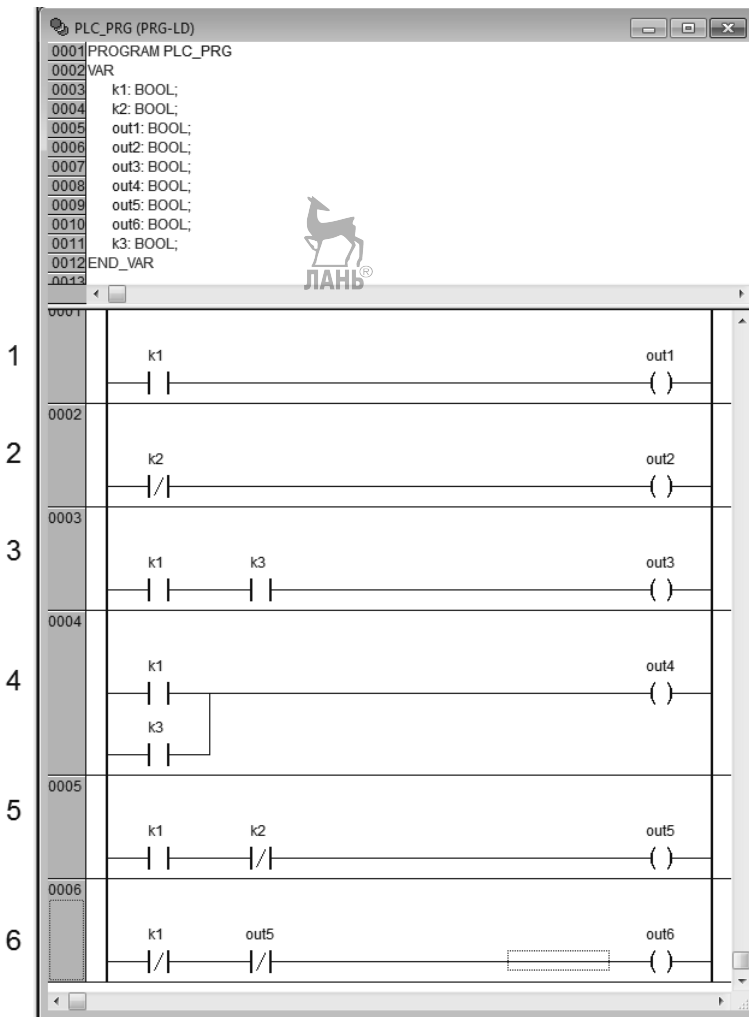


Рис. 7.29. Тестовая программа для изучения работы контактов

Программа представляет собой релейную диаграмму, состоящую из нескольких цепей. Чтобы добавить новую цепь на диаграмму, надо либо щелкнуть по соответствующей кнопке на панели инструментов либо выбрать *Вставить > Цепь (перед)/Цепь (после)*. Для добавления элемента на цепь, надо сначала выделить цепь, щелкнув по ней левой кнопкой мыши (на цепи должен появиться пунктирный прямоугольник), а затем с помощью панели инструментов или меню добавить необходимый элемент. Вместо вопросительных знаков над элементом надо записать имя переменной, например Var1, и нажать Enter. После этого появится окно *Объявление переменной*, в котором можно задать тип переменной и начальное значение. Затем нажать ОК. Окно объявления переменной можно также вызвать комбинацией клавиш Shift + F2.

Для активирования программы во вкладке меню *Онлайн* выбрать опции *Режим эмуляции*, *Подключение*, *Старт*. Получим рабочее состояние программы, представленное на рис. 7.30.

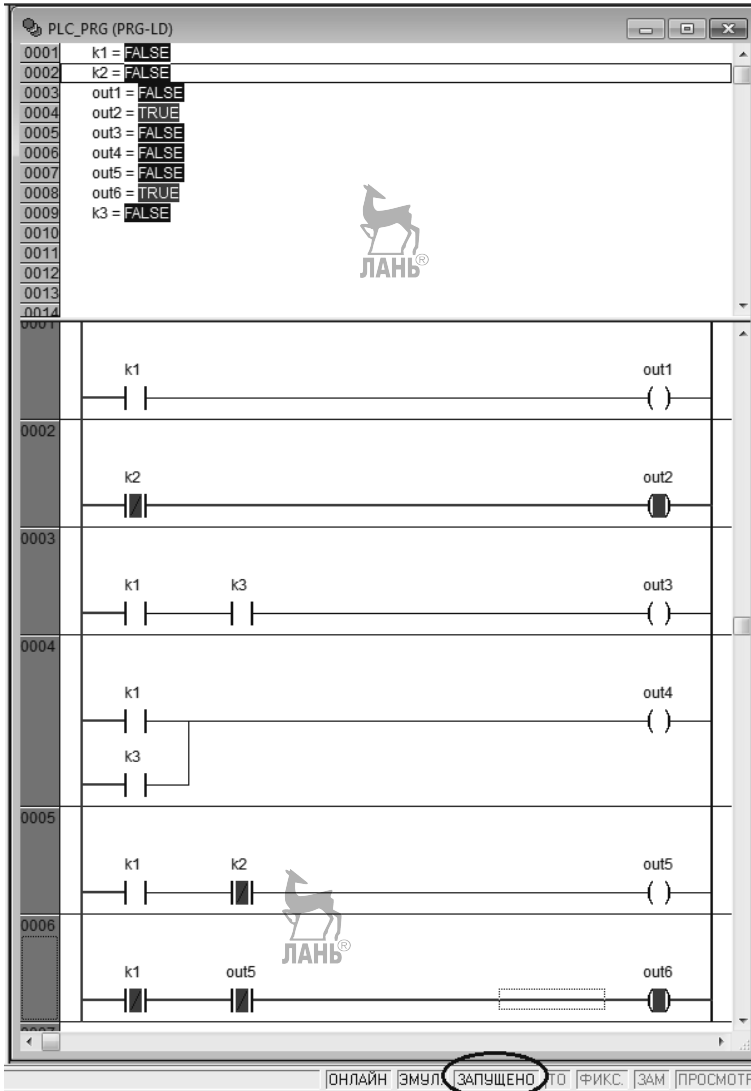


Рис. 7.30. Программа в режиме исполнения

Разберем работу каждой из цепей.

Цепь 1. Нормально разомкнутый контакт k1 находится в состоянии FALSE и не пропускает ток на выход «out1».

Цепь 2. Нормально замкнутый контакт k2 находится в состоянии FALSE и пропускает ток на выход «out2».

Цепь 3. Эта цепь является аналогом логического элемента И. Цепь пропускает ток, когда оба контакта находятся в состоянии «1» или TRUE.

Цепь 4. Эта цепь является аналогом логического элемента ИЛИ. Цепь пропускает ток, когда один из контактов находится в состоянии «1» или «TRUE».

Цепь 5. Нормально разомкнутый контакт k1 находится в состоянии FALSE и не пропускает ток. Нормально замкнутый контакт k2 находится в состоянии FALSE и пропускает ток. Выход «out5» находится в состоянии «0» или «FALSE».

Цепь 6. Нормально замкнутый контакт k1 инвертирован, т.е. находится в состоянии TRUE и пропускает ток. Выход «out5» инвертирован и пропускает ток. На выходе «out6» имеем «1» или TRUE.

Фрагмент электрической схемы, соответствующей двум последним цепям, имеет вид (рис.7.31).

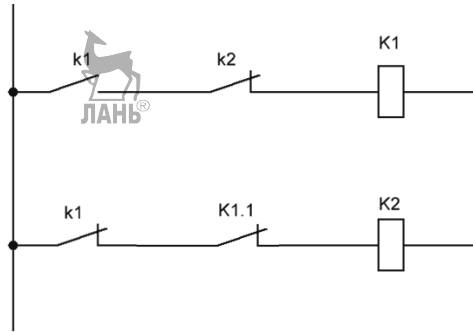


Рис.7.31. Фрагмент электрической схемы

Пример. 7.4. Пусть имеется двигатель и две управляющие двигателем кнопки. При нажатии на кнопку 1 двигатель вращается по часовой стрелке, а при нажатии на кнопку 2 – против часовой стрелки. Необходимо в системе управления двигателем предусмотреть блокировку, чтобы при случайном нажатии на обе кнопки сигнал на выходе схемы управления отсутствовал.

LD диаграмма для данной схемы управления представлена на рис. 7.32.

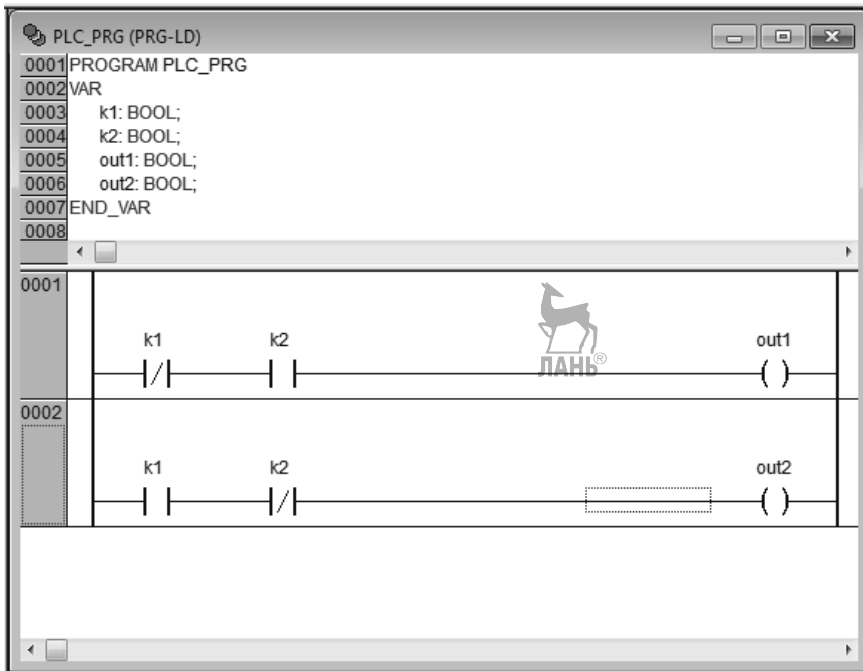


Рис. 7.32. LD диаграмма

Из схемы видно, что при замыкании контакта k1 (т.е. подаче на контакт значения TRUE) ток проходит на выход out2. При замыкании контакта k2 ток проходит на выход out1. При замыкании обоих контактов ток на выходах отсутствует. Замыкание и размыкание контактов осуществляется переводом контактов из состояния TRUE в состояние FALSE и затем записи этих значений с помощью комбинации клавиш Ctrl+F7.

В цепях могут использоваться обмотки с «самофиксацией» типов SET и RESET. Обмотки типа SET обозначаются буквой "S" внутри круглых скобок (S). Если соответствующая этой обмотке переменная принимает значение TRUE, то она сохраняет его до момента сброса (до установки R). Обмотки типа RESET обозначаются буквой R. Если соответствующая переменная принимает значение FALSE, то она сохраняет его до момента сброса (до установки S) сохраняет его.

Возможности LD программы можно расширить, вставляя функциональные блоки. Вставлять можно все стандартные функциональные блоки, которые содержатся в МЭК. Они могут использоваться подобно контактам. Очень часто в программу вставляются таймеры. Эти блоки входят в стандартную библиотеку (standard.lib). К таймерам относятся четыре блока: TP, TON, TOF, RTC. Рассмотрим работу первых трех блоков, которые представлены на рис. 7.33.

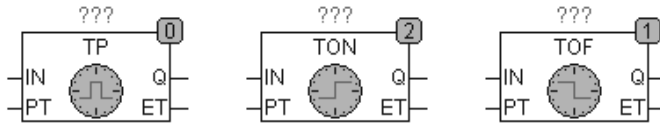


Рис. 7.33. Таймеры

Блоки имеют по два входа и два выхода. Вход IN и выход Q имеют тип BOOL. Вход PT и выход ET имеют тип TIME. Переменные и константы типа TIME в CoDeSys начинаются с префикса "T" и знака "#". Далее следует время, которое может включать дни "d", часы "h", минуты "m", секунды "s" и миллисекунды "ms". Нет необходимости обязательно определять все составляющие времени, но присутствующие поля обязаны следовать именно в таком порядке (d, затем h, затем m, затем s, затем m, затем ms).

Блок TP. Схема работы блока TP построена следующим образом. Пока IN равен FALSE, выход Q тоже FALSE, выход ET = 0. При переходе входа IN в TRUE выход Q сразу устанавливается в TRUE и находится в этом состоянии, пока таймер не отсчитает время, заданное на входе PT. После этого выход возвращается в состояние FALSE. Таким образом, выход Q генерирует импульс длительностью PT.

Блок TON. Схема работы блока TON построена следующим образом. Пока IN равен FALSE, выход Q = FALSE, выход ET = 0. Как только IN становится TRUE, начинается отсчет времени (в миллисекундах) на выходе ET до значения, равного PT. По окончании отсчета времени выход Q переходит в состояние TRUE и остается в дальнейшем в этом состоянии. Таким образом, выход Q переходит в состояние TRUE с задержкой PT. Далее, если подать FALSE на вход IN, то выход Q возвратится в исходное состояние FALSE.

Блок TOF. Вход IN и выход Q должны находиться в состоянии TRUE, выход ET = 0. Как только IN переходит в FALSE, начинается отсчет времени (в миллисекундах) на выходе ET. При достижении заданной длительности выход Q отключается, т.е. переходит в состояние FALSE. Таким образом, выход Q сбрасывается с задержкой PT при переходе входа IN из состояния TRUE в состояние FALSE.

Таким образом, начальное состояние блоков TP и TON подразумевает значение FALSE на входе и на выходе. Начальное состояние блока TOF подразумевает TRUE на входе и на выходе. При переходе входов в противоположное состояние начинают работать таймеры. Это утверждение справедливо для прямых контактов. Если на входе инверсные контакты, то начальное значение контактов для таймеров TP и TON должно быть TRUE, а для таймера TOF входной контакт должен иметь начальное значение FALSE.

Изложенные схемы работы таймеров можно проверить на LD диаграмме, представленной на рис. 7.34.

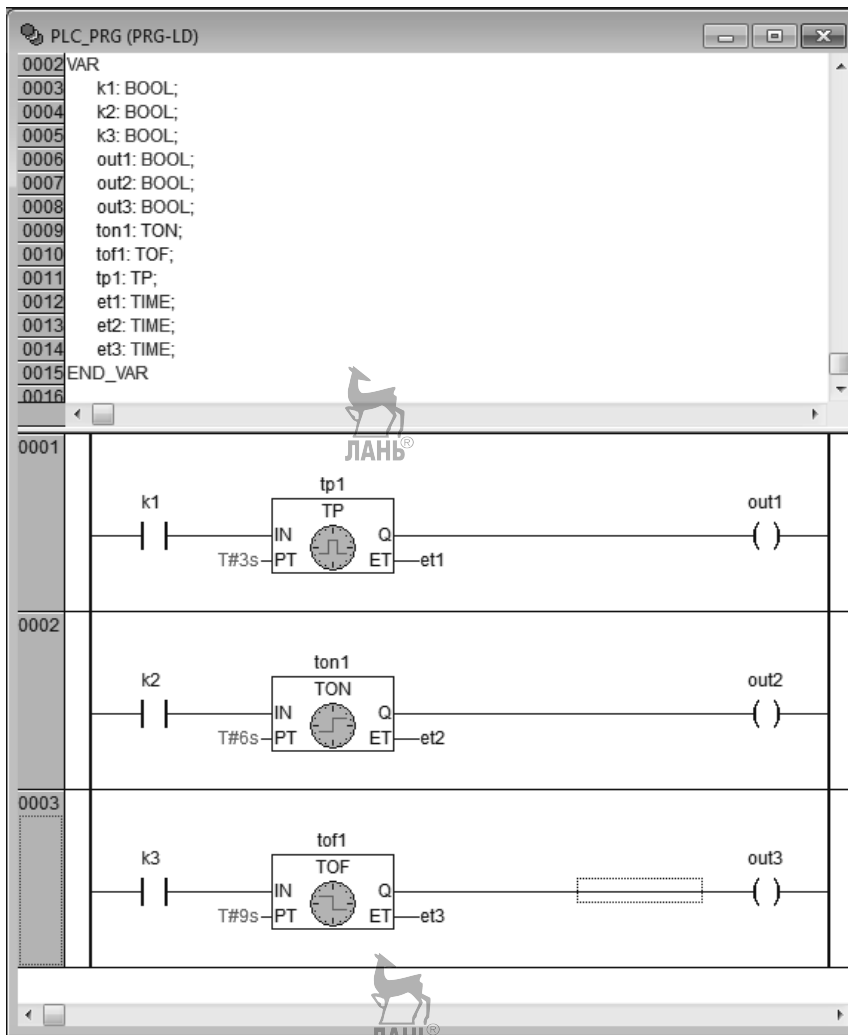


Рис. 7.34. LD диаграмма с таймерами

Над блоками таймеров на месте вопросительных знаков надо задать названия блоков. Это делается для того, чтобы различать блоки между собой, поскольку в программе может использоваться несколько одинаковых блоков, например, несколько блоков задержки в разных цепях. Переменные et1, et2, et3 отсчитывают текущее время и заканчивают свою работу в момент достижения значений, заданных на входе PT.

Пример 7.5. Составим LD диаграмму для схемы, представленной на рис. 7.35.

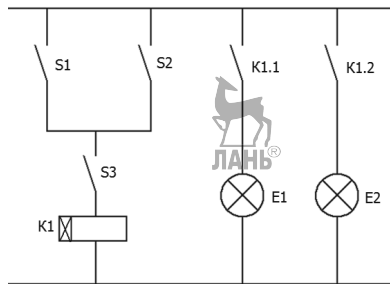


Рис. 7.35. Схема соединения

На этой схеме выходы E1, E2 включаются и отключается выключателями S1, S2, S3. Временное реле K1 срабатывает при выполнении условия S1 «ИЛИ» S2 «И» S3 с задержкой времени, например, 5 с.

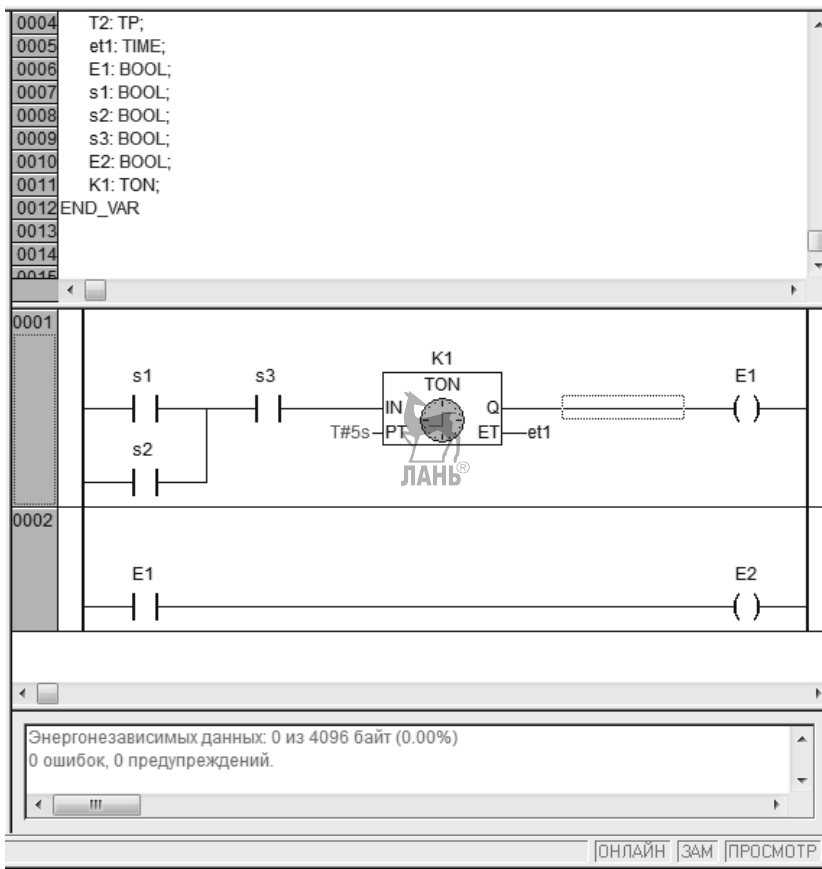


Рис. 7.36. LD диаграмма

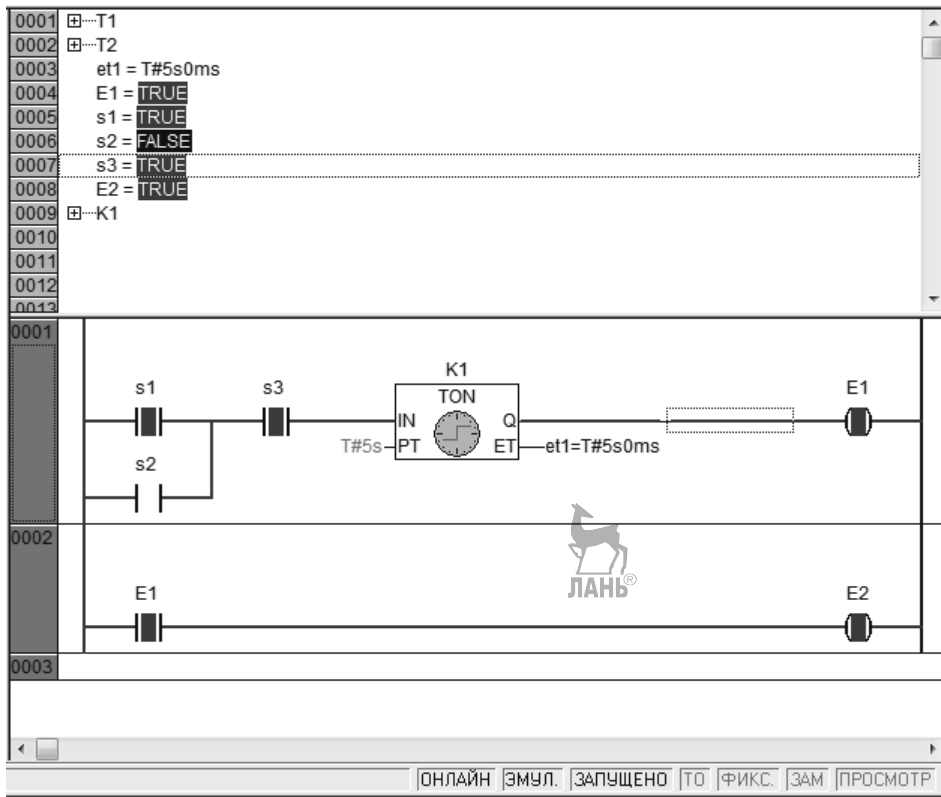


Рис. 7.37. Симуляция LD диаграммы

Обратите внимание, наименование катушки в первой цепи и контакта во второй цепи совпадают, т.е. катушка E1 управляет контактом E1. При замыкании катушки E1, контакт E1 также замыкается.

Пример 7.6. Циклический алгоритм. Для организации циклического процесса в LD диаграммах можно использовать схему, показанную на рис. 7.38. При $k1=TRUE$ выход $out1=FALSE$ (отключен). При $k1=FALSE$ выход $out1=TRUE$ (подключен). Длительность включения и отключения выхода $out1$ определяется временем, установленным в блоках T1 и T2.



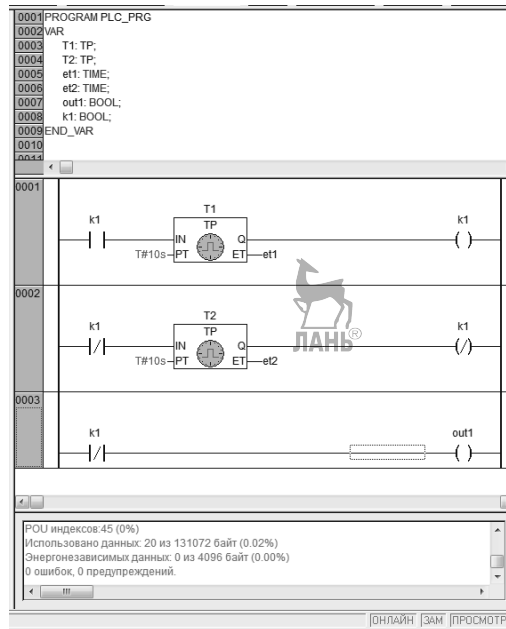


Рис. 7.38. Пример циклического алгоритма

7.5.4. Язык CFC

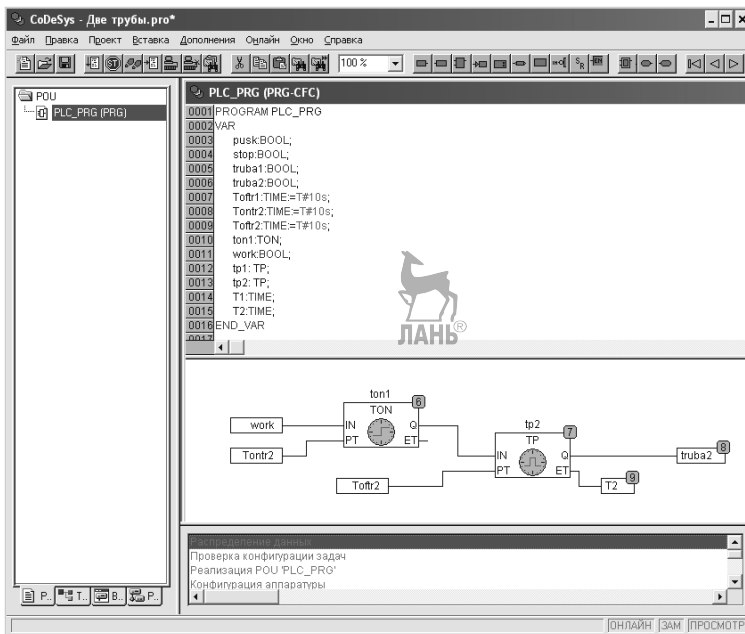


Рис. 7.39. Интерфейсное окно языка CFC

Интерфейсное окно языка CFC (Continuous Function Chart – редактор непрерывных функциональных схем) представлено на рис. 7.39, а панель инструментов на рис. 7.40. Расшифровка обозначений кнопок приведена в таблице 7.6. В отличие от языка FBD язык CFC не использует цепи и дает возможность свободно размещать компоненты и блоки. Вследствие этого в схемы CFC редактора можно добавлять линии обратной связи.

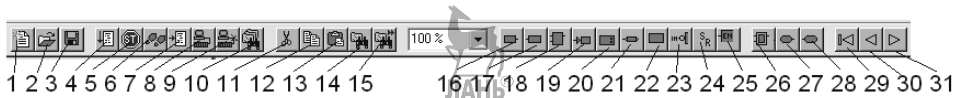


Рис. 7.40. Панель инструментов языка CFC

Таблица 7.6

16 – Вход	24 – Set/Reset
17 – Выход	25 – EN/ENO
18 – Элемент	26 – Создать макрос
19 – Переход	27 – Вход макро
20 – Метка	28 – Выход макро
21 – Возврат	29 – Перейти на верхний уровень
22 – Комментарий	30 – Вернуться на предыдущий уровень
23 – Инверсия	31 – Показать содержимое макроса

К элементам языка CFC относятся блоки, входы, выходы, возвраты, произвольные переходы, метки и комментарии. Входы и выходы этих элементов можно соединять, перетаскивая линии соединения мышкой. Эти линии будут перерисовываться автоматически при перемещении элементов. В случае, если линия соединения не может быть перерисована, то она становится красной, и как только переставить элемент так, чтобы можно было соединить вход и выход линией без пересечений с другими элементами, линия становится нормальной. Чтобы поместить элемент на рабочее поле, надо щелкнуть по значку элемента, расположенному в панели инструментов, левой кнопкой мыши, затем щелкнуть левой кнопкой мыши на рабочем поле. Команда может быть использована для вставки операторов, функций, функциональных блоков и программ. Сразу после ее выполнения появляется блок с именем "AND". Выбрав текстовое поле внутри этого блока, можно изменить его на имя любого другого оператора, функции, функционального блока или программы. Для этой цели удобно использовать *Ассистент ввода*, который вызывается нажатием клавиши F2.

Один или несколько элементов можно перемещать с помощью клавиш перемещения (кнопки со стрелками на клавиатуре), с зажатой клавишей <Shift>. Элемент необходимо предварительно выделить, зажав левую клавишу мыши и очертив вокруг элемента прямоугольник. Перемещать выделенный элемент можно также с помощью зажатой левой клавиши мыши, захватив элемент за пунктирный прямоугольник в области элемента, который появляется после выделения

элемента. Элементы перемещаются до тех пор, пока они не перекрывают другие элементы или не заходят за пределы экрана.

Вход одного элемента можно соединять с выходом другого. Выход одного элемента может соединяться сразу с несколькими входами других элементов. Для соединения элементов надо поместить указатель мыши на выход одного элемента, нажать левую кнопку мыши и, удерживая ее, переместить курсор мыши на вход другого элемента, и отпустить кнопку мыши. Можно также создать линию соединения другим способом. Нужно переместить один из элементов так, чтобы его вход (выход) соприкоснулся с выходом (входом) другого. Теперь можно как угодно перемещать элементы, и при этом они останутся соединенными.

Чтобы инвертировать вход, надо нажать на него правой клавишей мыши и выбрать в выпадающем окне опцию *Инверсия*. На инвертированном входе появляется кружок. Инверсию можно снять, выполнив команду еще раз.

Есть несколько способов удаления линии, соединяющей выход элемента 1 и вход элемента 2. Способ 1: выбрать выход элемента 1 или вход элемента 2 на концах соединительной линии и нажать Delete или выполнить команду *Правка > Очистить*. Если выход элемента 1 связан с несколькими входами, то будут удалены все соединения. Способ 2: поместить указатель мыши на вход элемента 2 и, удерживая левую клавишу мыши, переместить его на свободную область экрана. Соединение будет удалено, как только будет отпущена кнопка мыши.

Порядок выполнения схемы определяется потоком данных, а не позициями элементов. Чтобы расставить элементы в соответствии с потоком данных, необходимо нажать правой клавишей мыши в рабочем поле и выбрать опции *Порядок > В соответствии с потоком данных*.

Пример 7.7. Мигание светодиода. Изучение программирования любого контроллера начинается с создания простейшей программы мигания светодиодом. Не будем нарушать традицию и рассмотрим пример создания такой программы с помощью стандартного функционального блока BLINK. Чтобы разместить BLINK на рабочем поле выберем опцию *Менеджер библиотек* во вкладке меню *Окно*. Откроется окно, представленное на рис. 7.41.

В области 3 этого окна представлены блоки стандартной библиотеки Standard.lib. Среди них нет блока BLINK, так как этот блок находится в библиотеке утилит Util.lib. Подгрузим эту библиотеку. Для этого щелкнем правой кнопкой мыши в области 1 и выберем в выпадающем окне опцию *Добавить библиотеку*. Появится окно, представленное на рис. 7.42.



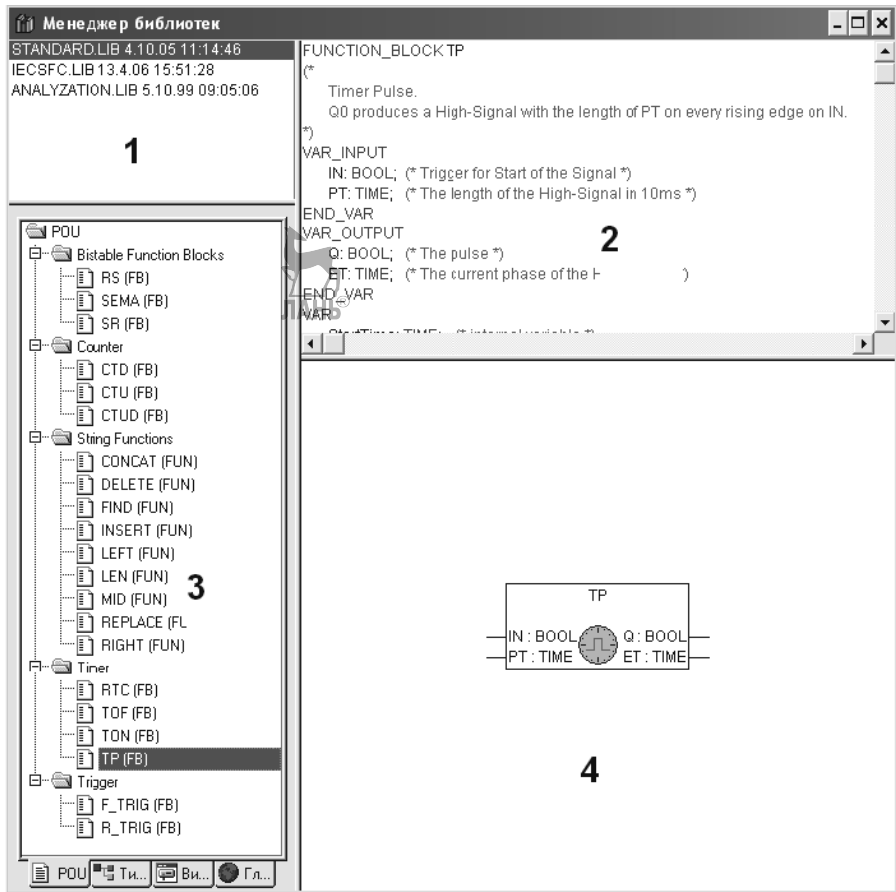


Рис. 7.41. Окно Менеджера библиотек

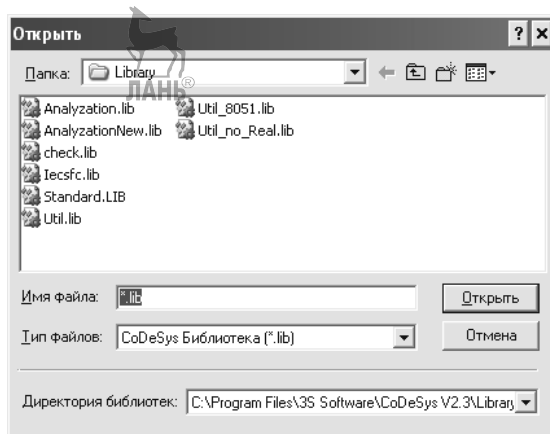


Рис. 7.42. Перечень библиотек

Выберем в этом окне строку Util.lib и нажмем *Открыть*. В область 3 добавится библиотека утилит, в которой найдем функциональный блок BLINK. Свернем окно рис. 7.41 и приступим к созданию программы. Щелкнем правой кнопкой мыши в области рабочего поля и в выпадающем окне выберем строку *Элемент*. На рабочем поле появится элемент AND. Выделим надпись и нажмем клавишу F2. Появится окно *Ассистент ввода*, представленное на рис. 7.43.

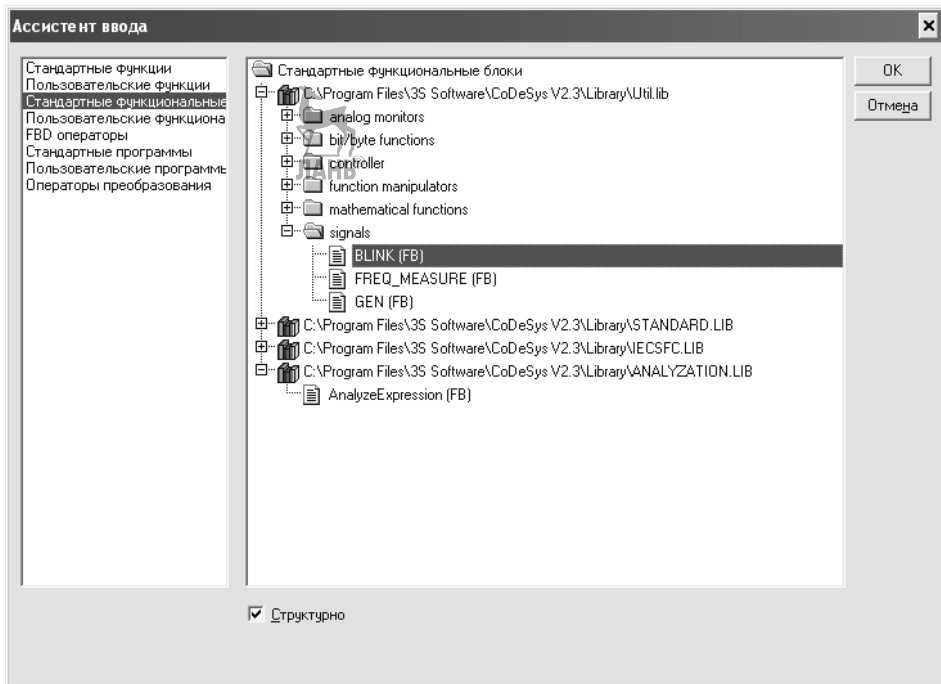
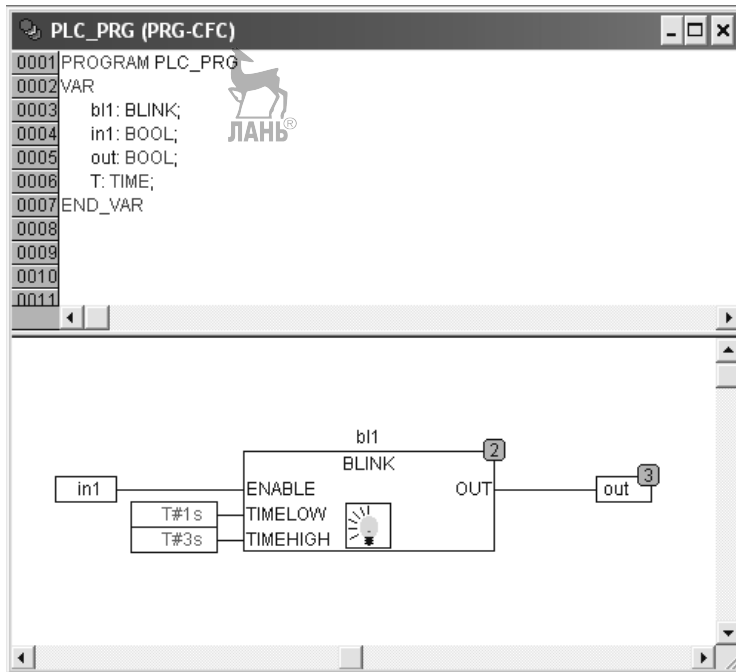
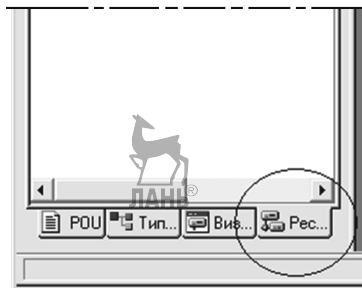


Рис. 7.43. Окно *Ассистент ввода*

В этом окне к стандартной библиотеке Standard.lib уже будет добавлена библиотека Util.lib. Найдем здесь блок BLINK, выделим его и нажмем ОК. На месте вопросительных знаков над блоком зададим название переменной для этого блока – bl1. Щелкнем в рабочем поле правой кнопкой мыши и добавим на схему вход in1, выход out, как показано на рис. 7.44. Чтобы перемещать блоки входа и выхода по рабочему полю, надо захватить блок левой кнопкой мыши со стороны, противоположной хвостику. Установим для блока BLINK временные промежутки 1 с и 3 с, т.е. в течение трех секунд светодиод будет гореть (TIME HIGH), и в течение одной секунды будет погашен (TIME LOW).

Рис. 7.44. Блок *Blink*

Чтобы запустить программу на исполнение, в меню *Онлайн* выбираем *Режим эмуляции*, щелкаем по строке *Подключение* (можно нажать комбинацию клавиш Alt+F8) и щелкаем по строке *Старт* (можно нажать клавишу F5). Далее переводим вход in1 из состояния FALSE в состояние TRUE и наблюдаем периодическое переключение выхода из состояния TRUE в состояние FALSE. Работу выхода out программы можно посмотреть с помощью цифрового трассировщика. Перейдем на вкладку *Ресурсы* в нижнем левом углу интерфейсного окна (рис. 7.45).

Рис. 7.45. Расположение вкладки *Ресурсы*

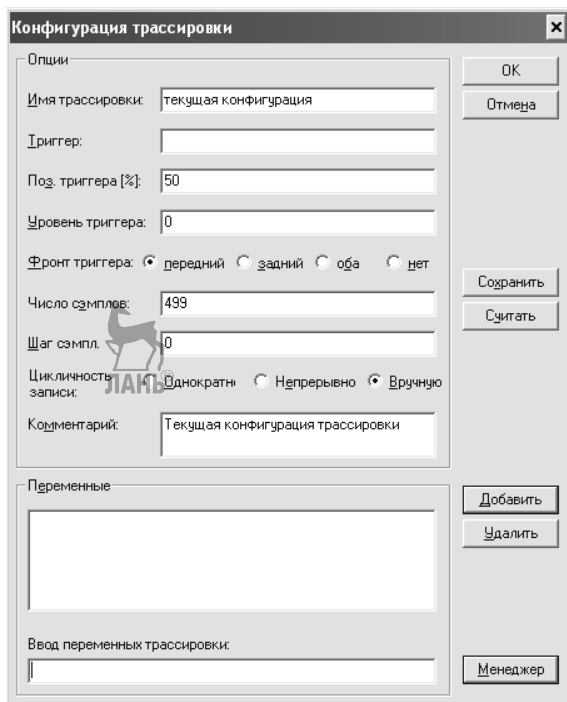


Рис. 7.46. Окно Конфигурация трассировки

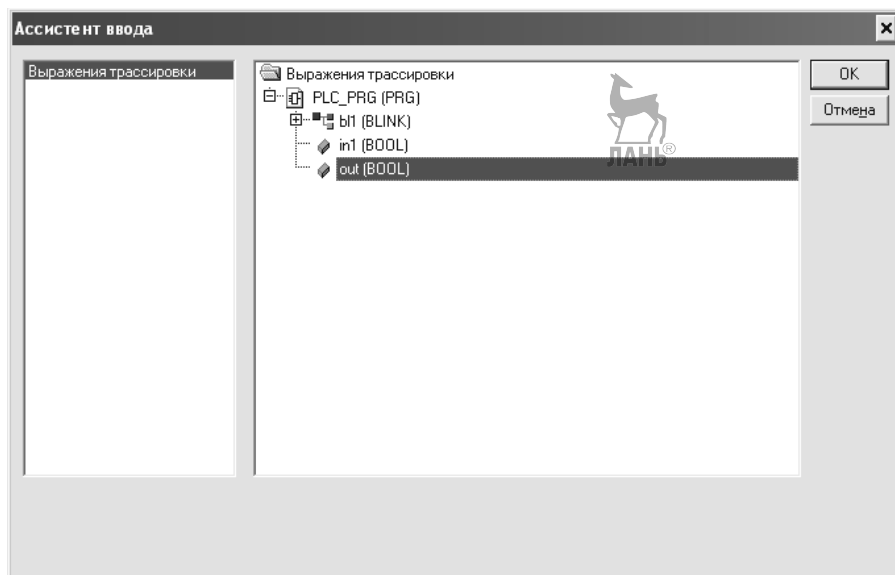


Рис. 7.47. Окно Ассистент ввода

Дважды щелкнем по строке *Цифровая трассировка*. В появившемся окне щелкаем по свободному полю правой кнопкой мыши и в выпадающем окне выбираем строку *Настройка трассировки*. Появится новое окно *Конфигурация трассировки* (рис. 7.46). Выбираем в этом окне цикличность записи *Вручную* и щелкаем по кнопке *Менеджер*. Появится окно *Ассистент ввода* (рис. 7.47). Выбираем в этом окне переменную out(BOOL) и щелкаем ОК. Еще раз щелкаем ОК и закрываем окно *Конфигурация трассировки*. Переходим на вкладку *POU*, запускаем задачу (*Онлайн > Подключение > Старт*) и переводим in1 в состояние TRUE. Программа работает. Переходим в окно *Цифровая трассировка*, выбираем *Перо*, щелкаем правой кнопкой мыши в свободном поле и выбираем в выпадающем окне опции *Начать трассировку* и *Автоматическое чтение*. Получим картинку, представленную на рис. 7.48.

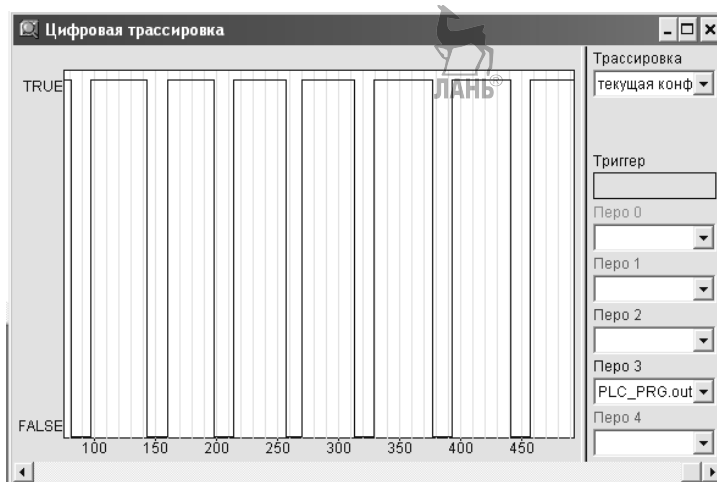


Рис. 7.48. Трассировка

Пример 7.8. Мигание группы светодиодов. От мигания одним светодиодом перейдем к миганию несколькими светодиодами. Пусть их будет шесть. Программа для мигания шестью светодиодами представлена на рис. 7.49. В данном варианте обеспечивается поочередное мигание трех верхних и трех нижних светодиодов.

Для наглядности выполним визуализацию проекта. Щелкнем по вкладке *Визуализации* в левом нижнем углу. Эта вкладка находится рядом со вкладкой *Ресурсы*. Вверху появится строка *Визуализации*. Щелкнем по ней правой кнопкой мыши и в появившемся окне выберем строку *Добавить объект*. Появится небольшое окно, в котором надо присвоить имя для вновь создаваемой визуализации. Пусть будет viz1, щелкнем ОК. Появится окно с таким названием сверху, в котором будем создавать визуализацию. Можно настроить параметры окна, если внутри (в области рабочего поля) щелкнуть правой кнопкой мыши и выбрать в появившемся окне опцию *Настройки*. В частности, можно настроить параметры сетки в этом окне.

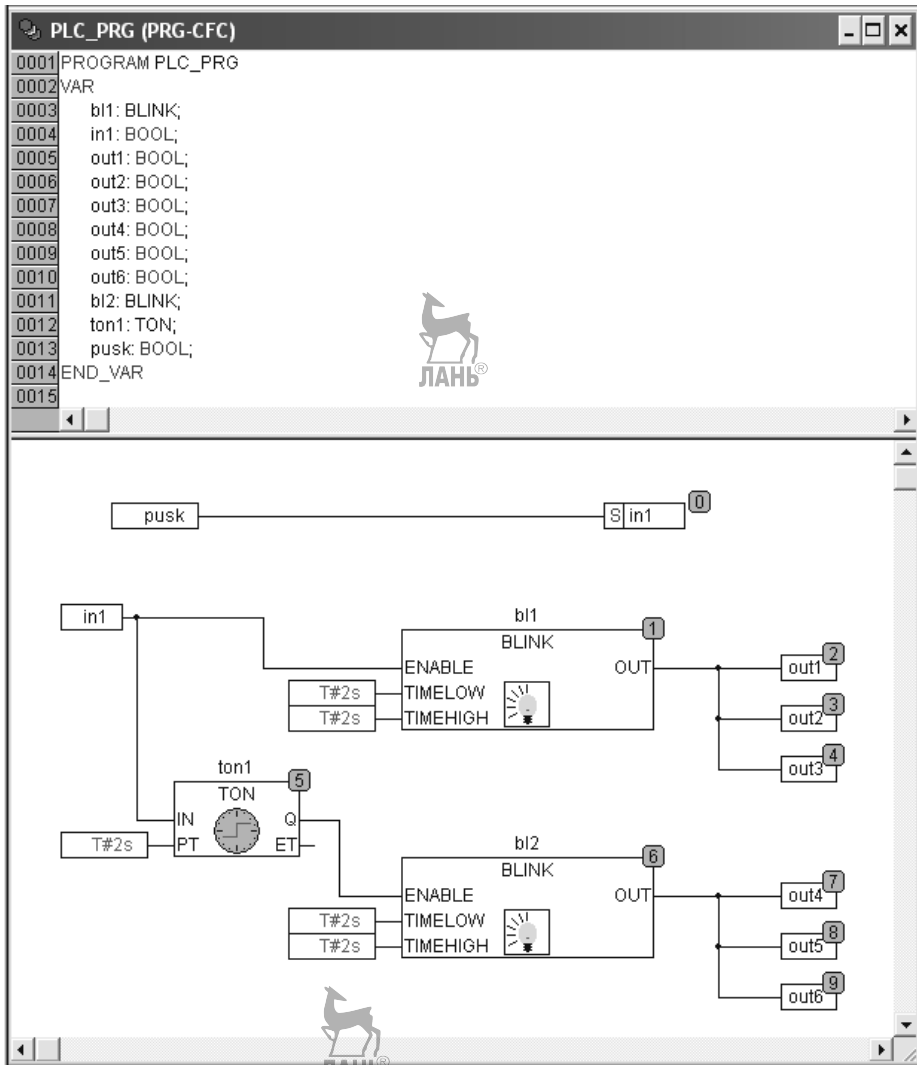


Рис. 7.49. Программа для мигания светодиодами

```

PROGRAM PLC_PRG
VAR
    bl1:BLINK;
    in1:BOOL;
    out1:BOOL;
    out2:BOOL;
    out3:BOOL;
    out4:BOOL;
    out5:BOOL;

```

```

out6:BOOL;
bl2:BLINK;
ton1:TON;
pusk:BOOL;
END_VAR

```



В режиме визуализации под строкой меню в интерфейсном окне появится новая панель инструментов, представленная на рис. 7.50.



Рис. 7.50. Панель инструментов визуализации

Первые тринадцать значков (иконок) этой панели встречались ранее, а вот остальные значки появились впервые. Их наименования приведены в таблице 7.7.

Таблица 7.7

14 – Режим выбора	24 – Кнопка
15 – Прямоугольник	25 – WMF-файл
16– Скругленный прямоугольник	26 – Таблица
17 – Эллипс	27 – Тренд
18 – Полигон	28 – Таблица тревог
19 – Ломаная линия	29 – Элемент ActiveX
20 – Кривая	30 – Ползунок
21 – Сектор	31 – Индикатор
22 – Растровый рисунок	32 – Столбиковый указатель
23 – Визуализация	33 – Гистограмма

Щелкнем по значку 15 и нарисуем прямоугольник. Щелкнем по значку 17 и нарисуем окружность. Причем достаточно нарисовать одну окружность, а остальные получим копированием с помощью значков 12, 13. Щелкнем по значку 24 и нарисуем в нижней части большого прямоугольника маленький прямоугольник, символизирующий кнопку запуска. В итоге получим рисунок, представленный на рис. 7.51.





Рис. 7.51. Рисунок визуализации

Теперь необходимо связать элементы рисунка с переменными программы. Дважды щелкнем по маленькому прямоугольнику. Появится окно *Конфигурирование элемента*. Слева в этом окне выберем опцию *Текст* и напишем *Пуск*, как показано на рис. 7.52.

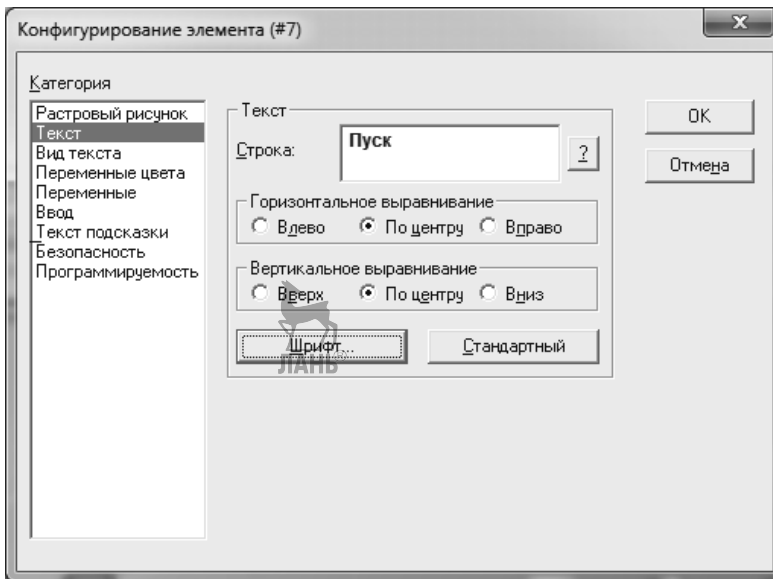


Рис. 7.52. Конфигурирование кнопки Пуск

Затем в этом же окне (рис. 7.52) щелкнем по строке *Ввод* и поставим галочку около строки *Пер-я переключения* в появившемся окне. Далее поместим курсор внутри прямоугольной полоски правее галочки и нажмем клавишу F2. Появится окно *Ассистент ввода*, представленное на рис. 7.53.

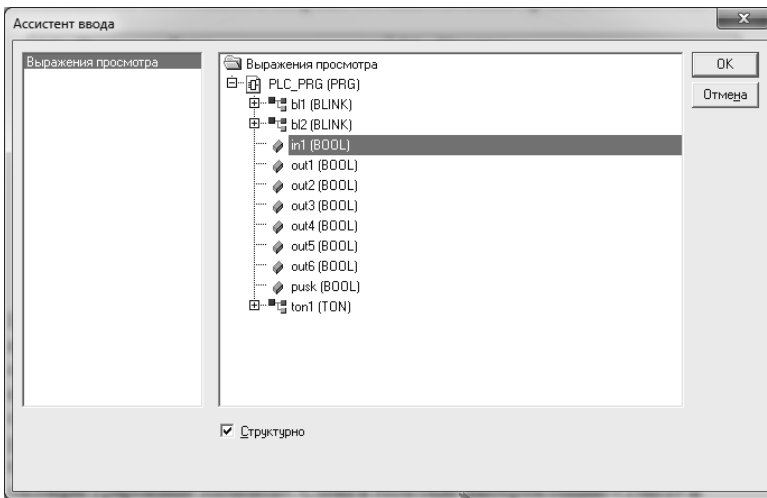


Рис. 7.53. Окно *Ассистент ввода*

Выберем в этом окне переменную `in1 (BOOL)` и нажмем **OK**. Окно *Конфигурирование элемента* примет вид, показанный на рис. 7.54. Нажмем **OK** и закроем окно. Таким образом, мы связали кнопку Пуск с переменной `in1`.

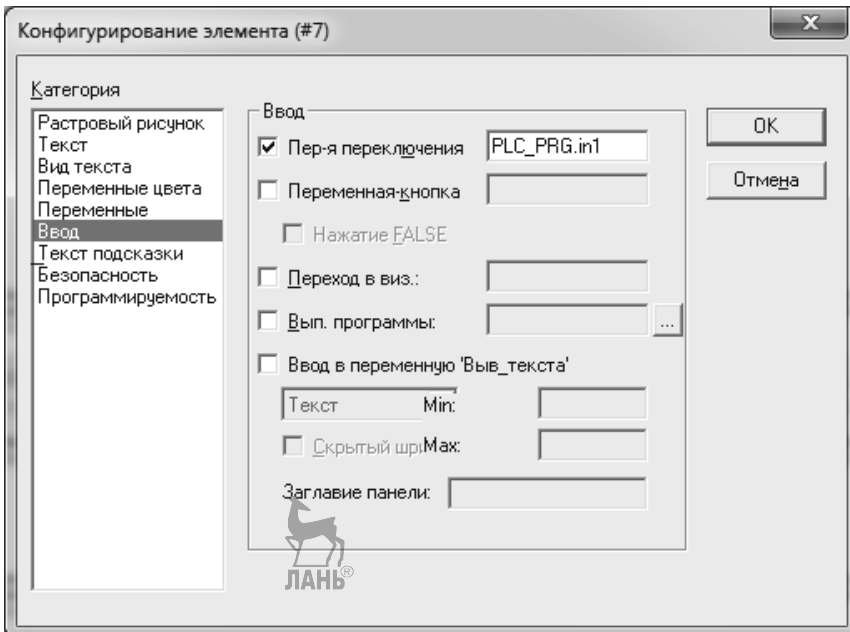


Рис. 7.54. Конфигурирование переменной входа

Кружочки на рис. 7.51 соответствуют светодиодам, поэтому их надо связать с переменными out1...out6. Прделаем это. Дважды щелкнем по верхнему левому кружочку, появится окно, показанное на рис. 7.55. Выберем слева опцию *Цвета*. Появится окно рис. 7.56. Щелкая в этом окне по верхней и нижней кнопке *Заливка*, установим начальный цвет белый, а тревожный цвет красный. Далее перейдем в этом окне на строку *Переменные*. Появится окно рис. 7.57. Поставим курсор внутри прямоугольной полоски *Изм. цвета* и нажмем клавишу F2. В появившемся окне *Ассистент ввода* выберем переменную out1 и нажмем ОК. В окне на рис. 7.57 в строке *Изм. цвета* появится запись PLC_PRG.out1.

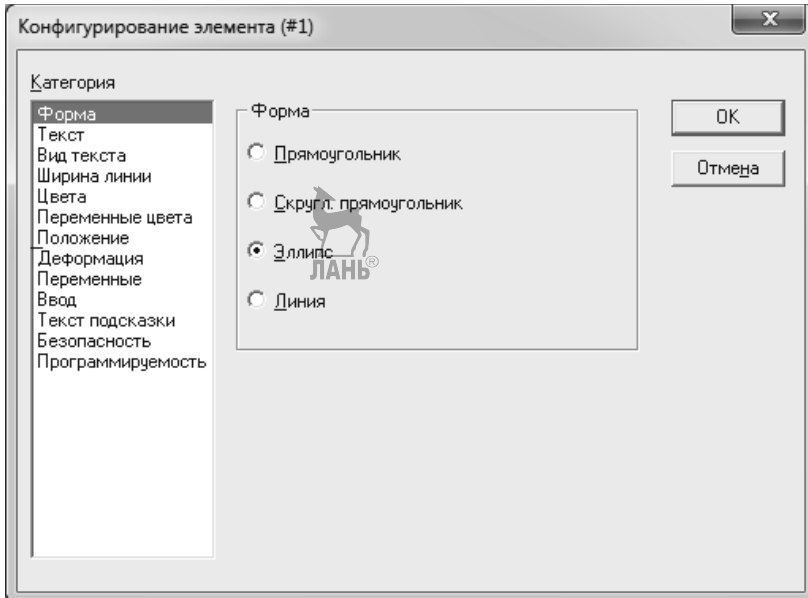


Рис. 7.55. Конфигурирование переменных выхода



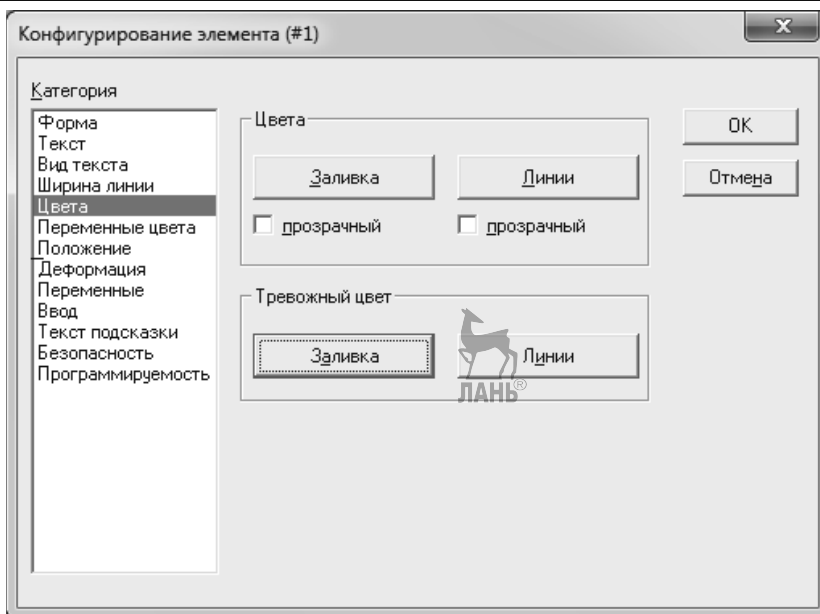


Рис. 7.56. Окно назначения цвета

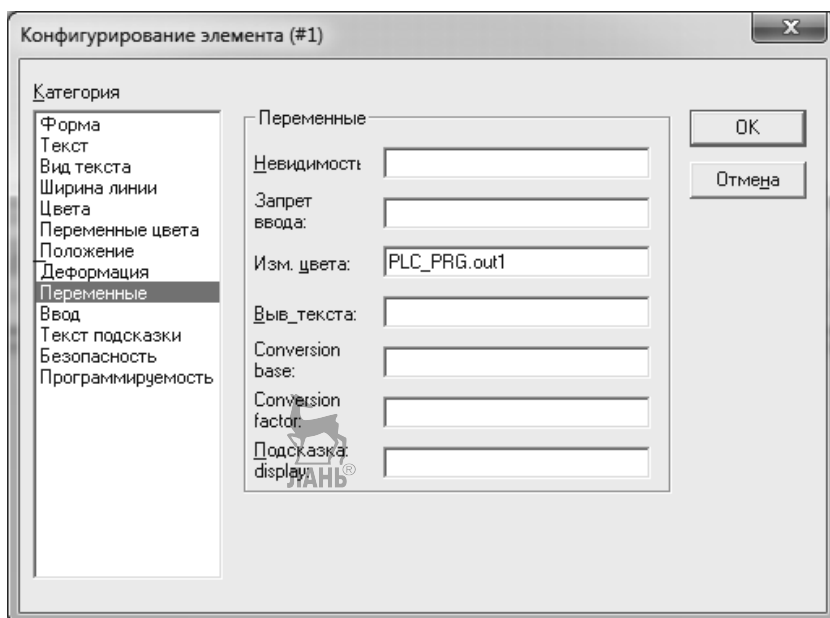


Рис. 7.57. Конфигурирование переменных выхода

Проделав аналогичные операции, свяжем остальные кружочки с переменными out2...out6. При этом светодиодам верхнего ряда назначим тревожный цвет

красный, а светодиодам нижнего ряда синий. Итак, мы связали все элементы нашего рисунка с переменными. Теперь можно запускать программу. В нижней левой части интерфейсного окна переходим на вкладку *POU*. В меню *Онлайн* выбираем *Подключение > Старт*. Нажимаем кнопку *Пуск* на рисунке визуализации и получаем попеременное мигание красных и синих светодиодов. Чтобы окно программы не загромождало всю рабочую область, его надо уменьшить и добиться того, чтобы в рабочей области были видны оба окна: окно программы (*PLC_PRG*) и окно визуализации (*viz1*).

Пример 7.9. Дозированное наполнение. Предположим, что из бака через нижнюю трубу 2 равными порциями откачивается жидкость. Через верхнюю трубу 1, также равными порциями, в бак закачивается жидкость, поддерживая в баке постоянный уровень жидкости. Такая задача может быть актуальна, например, для автоматических линий розлива жидкости в стеклянную или пластмассовую тару. Цикл начинается с того, что открывается труба 1 и через нее в бак поступает жидкость. Продолжительность заправки жидкости в бак задается переменной *Tofttr1*. После этого труба 1 перекрывается. Через время задержки *Tonttr2* после открывания верхней трубы 1 открывается труба 2, через которую вода откачивается из бака. Продолжительность откачивания воды задается временем *Tofttr2*. После этого труба 2 перекрывается и вновь открывается труба 1, через которую в бак поступает вода. Далее процесс повторяется. Схема установки приведена на рис. 7.58, а временная диаграмма работы установки приведена на рис. 7.59.

Программа работы установки приведена на рис. 7.60.

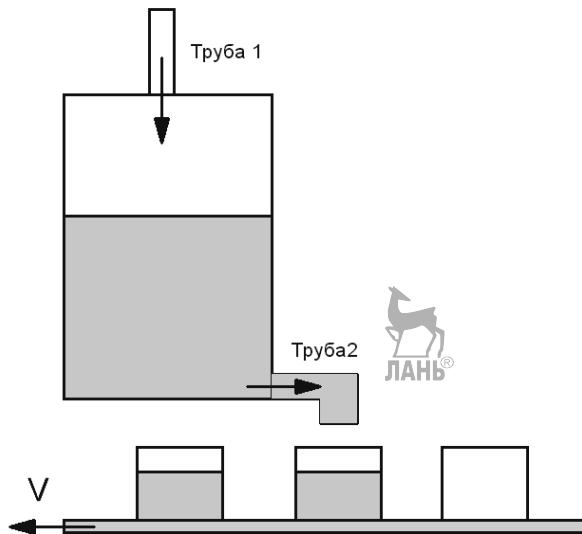


Рис. 7.58. Схема установки

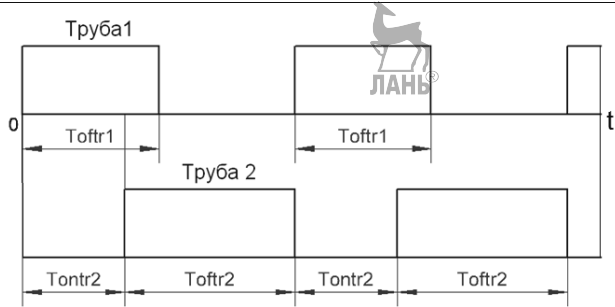


Рис. 7.59. Временная диаграмма

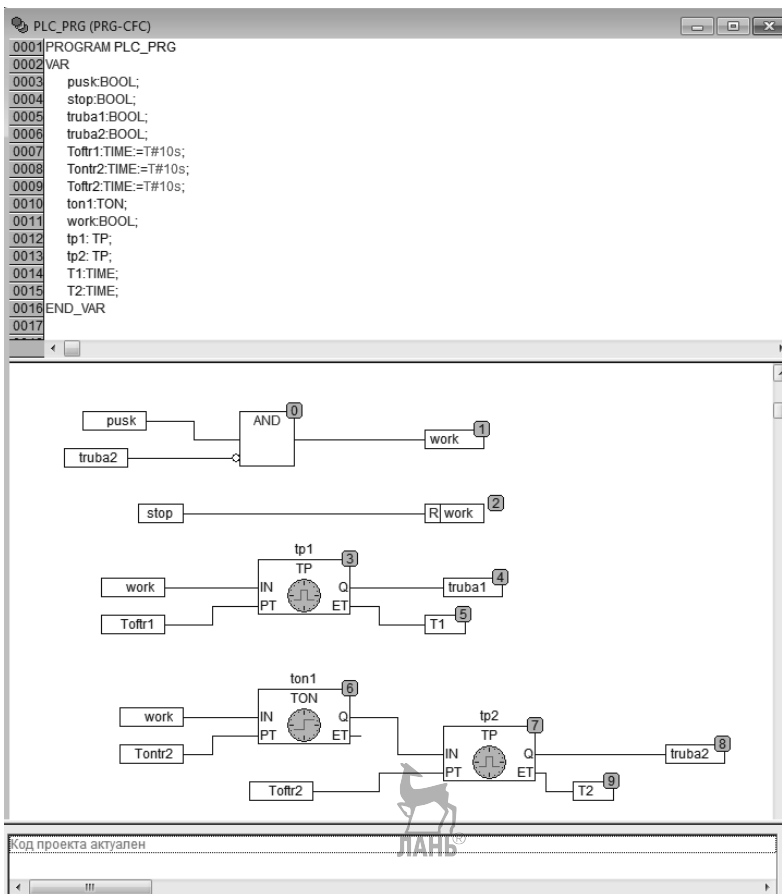


Рис. 7.60. Программа «Дозированное наполнение»

```

PROGRAM PLC_PRG
VAR
  pusк:BOOL;

```

```

stop:BOOL;
truba1:BOOL;
truba2:BOOL;
Toftr1:TIME:=T#10s;
Tontr2:TIME:=T#10s;
Toftr2:TIME:=T#10s;
ton1:TON;
work:BOOL;
tp1:TP;
tp2:TP;
T1:TIME;
T2:TIME;
END_VAR

```



Контрольные вопросы и задания

1. Для программирования какой серии контроллеров ОВЕН используется среда программирования Codesys?
2. Что такое таргет-файлы и зачем они нужны?
3. Какие языки программирования можно использовать в Codesys?
4. Какие вкладки расположены в нижней части Организатора (менеджера) объектов?
5. Как объявить переменную в Codesys?
6. Привести пример объявления булевой переменной?
7. Как задать значение переменной при ее объявлении?
8. Как правильно задать переменную времени с значением времени $t=5$ сек?
9. Какую комбинацию клавиш надо использовать при задании типа переменной?
10. Какие значения может принимать переменная типа word, и сколько памяти она занимает?
11. Какие библиотеки должны обязательно присутствовать в стандартной поставке Codesys?
12. Как подключить в Codesys отсутствующую библиотеку?
13. Написать на языке ST фрагмент программы с оператором if:

$$\text{если } a < b \text{ и } (a + b) > c, \text{ то}$$

$$y = ax^2 + bx + c$$
14. Написать на языке ST программу для вычисления суммы чисел от 1 до 20.
15. Как осуществить на языке ST выполнение задачи по шагам?
16. Написать на языке LD программу для электрической схемы на рис. 7.61

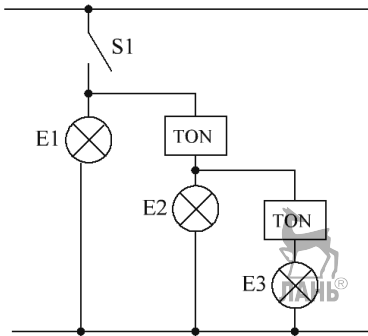


Рис. 7.61. Электрическая схема

17. Программу **Примера 7.2** переделать с учетом привязки к входам и выходам реального контроллера.
18. Переделать программы **Примеров 7.2 и 7.3** с языка ST на язык CFC.
19. Сделать визуализацию к **Примеру 7.3**. Визуализация должна содержать три прямоугольника: 1) высотой меньше области допустимых размеров, 2) высотой в пределах допустимой области и 3) высотой выше допустимой области. Если щелкнуть левой кнопкой мыши по прямоугольникам, которые выходят за пределы допустимой области, то должен на короткое время загораться светодиод, сигнализирующий о срабатывании бокового толкателя.



ГЛАВА 8. ПРОЕКТИРОВАНИЕ СИСТЕМЫ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ



8.1. Планирование системы логического управления

Рассмотрим загородный дом в качестве объекта, который необходимо оборудовать системой управления освещением и охранными датчиками. Для построения системы управления будем использовать программируемое логическое реле. Разработку проекта системы логического управления можно разбить на следующие этапы:

- Планирование системы логического управления.
- Разработка логической схемы.
- Разработка управляющей программы и ее моделирование.
- Разработка электрической схемы.

Пусть дом имеет систему наружного освещения, которая состоит из четырех дежурных светильников, установленных на столбах по четырем углам дома, как показано на рис. 8.1. Дежурные светильники оснащены датчиками освещенности, которые являются автономными и не связанными с центральной системой управления. По командам датчиков освещенности дежурное освещение включается в темное время суток и отключается в светлое время суток. Кроме дежурных светильников на столбах установлены дополнительные светильники, которые включаются по сигналу тревоги охранных датчиков. Датчики охранной сигнализации включают в себя:

- контактные датчики (герконы), установленные в дверных проемах и срабатывающие при открывании двери;
- инфракрасные датчики движения, сигнализирующие о движении снаружи дома в районе входных дверей;
- микроволновый датчик движения, установленный внутри дома и сигнализирующий о проникновении во внутренние помещения дома. Необходимость такого датчика обусловлена тем, что нарушитель может проникнуть в дом через окно, поэтому нужен датчик, который контролирует внутреннее пространство дома, невзирая на стены и перегородки. Эту функцию может выполнить микроволновый датчик.

Логика работы системы следующая. При срабатывании инфракрасного датчика движения включается усиленное наружное освещение на всех четырех столбах и сигнальная лампа внутри дома. При срабатывании контактного датчика, или при совместном срабатывании контактного и инфракрасного датчиков включается усиленное освещение, звуковая сирена и сигнальная лампа. При срабатывании микроволнового датчика, или при срабатывании микроволнового датчика в любой комбинации с другими охранными датчиками, включается

усиленное освещение, звуковая сирена, сигнальная лампа и передается сигнал тревоги в центральный пост охраны.

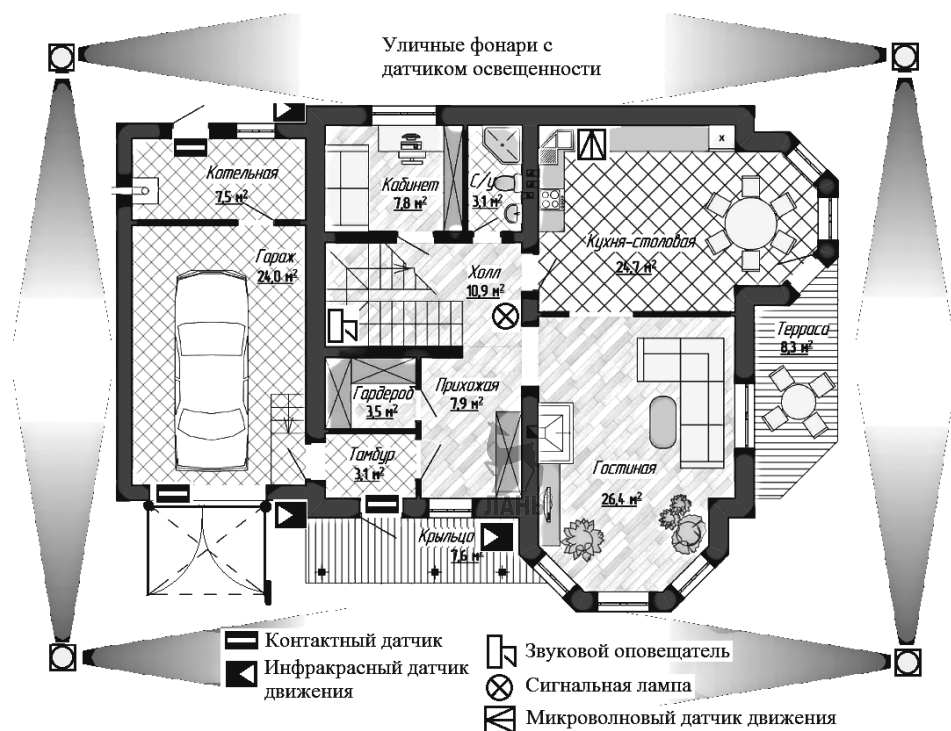


Рис. 8.1. План первого этажа загородного дома



8.2. Разработка логической схемы.

Составим таблицу истинности исходя из логики работы системы. Имеем три входных логических переменных: А – контактный датчик, В – инфракрасный датчик движения, С – микроволновый датчик движения. Имеем четыре выходных логических переменных: Y1 – усиленное освещение; Y2 – звуковой сигнал; Y3 – сигнальная лампа; Y4 – сигнал тревоги в центральный пост охраны. Таблица истинности будет иметь следующий вид

A	B	C	Y1	Y2	Y3	Y4
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	0
0	1	1	1	1	1	1

1	0	0	1	1	1	0
1	0	1	1	1	1	1
1	1	0	1	1	1	0
1	1	1	1	1	1	1

С помощью, инструмента *Логический преобразователь (Logic converter)* программы Multisim получим логические выражения для выходных переменных Y_1, Y_2, Y_3, Y_4 . Для этого поочередно заполняем правый столбец для каждой функции Y . Для Y_1 , например, логическое выражение, как видно из рис. 8.2, будет иметь вид

$$Y_1 = A + B + C$$

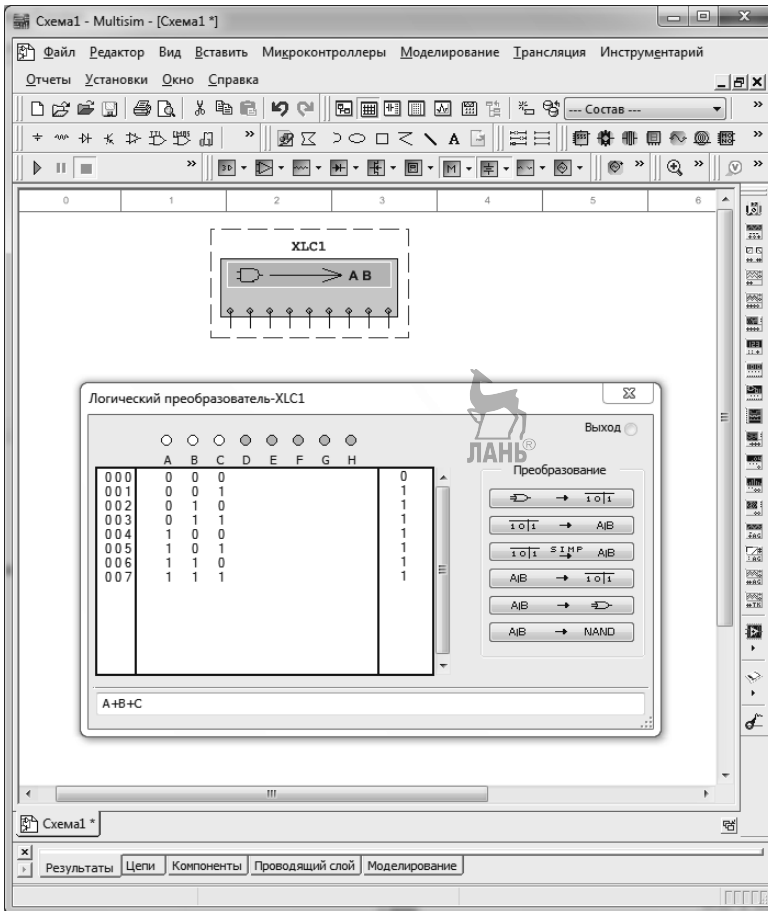


Рис. 8.2. Получение логических выражений с помощью Multisim

Для трех других функций, соответственно получим:

$$Y_2 = A + C$$

$$Y3 = A + B + C$$

$$Y4 = C$$

Логическая схема будет иметь вид (рис. 8.3)

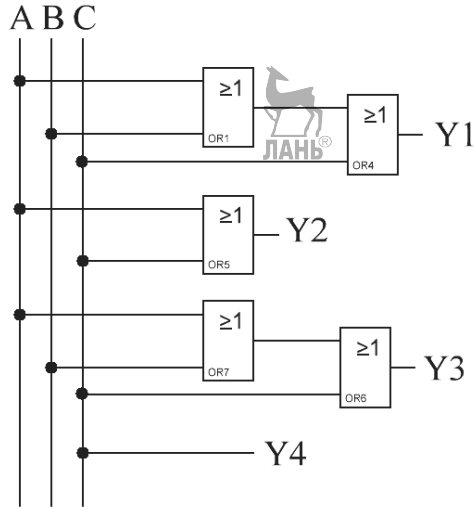


Рис. 8.3. Логическая схема

8.3. Разработка управляющей программы.

Рассмотрим построение управляющих программ для реле LOGO!, ONI и OWEN. Коммутационная программа для LOGO! в среде программирования LOGO! Soft Comfort будет иметь вид, как показано на рис. 8.4.

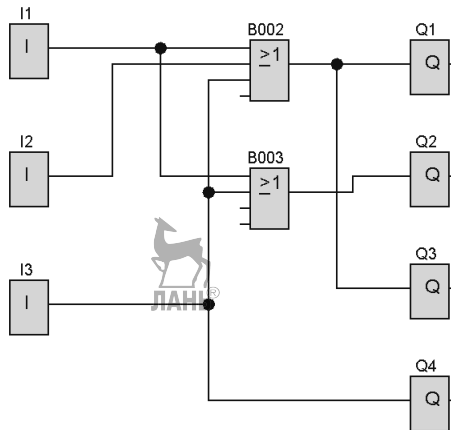


Рис. 8.4. Коммутационная программа

Полученную коммутационную программу необходимо доработать. Необходимо предусмотреть задержку отключения тревожных сигналов. При этом в цепь звукового оповещателя целесообразно поставить блок «Интервальное реле с запуском по фронту», а не блок «Задержка отключения». Это вызвано тем, что дверь при попытке проникновения в дом, может остаться открытой, и звуковой сигнал будет продолжаться слишком долго. Блок «Интервальное реле с запуском по фронту» будет отключать звуковой сигнал через заданный промежуток времени. Кроме того, для усиления эффекта тревоги, выходной сигнал лучше сделать прерывистым. Доработанная коммутационная программа представлена на рис. 8.5.

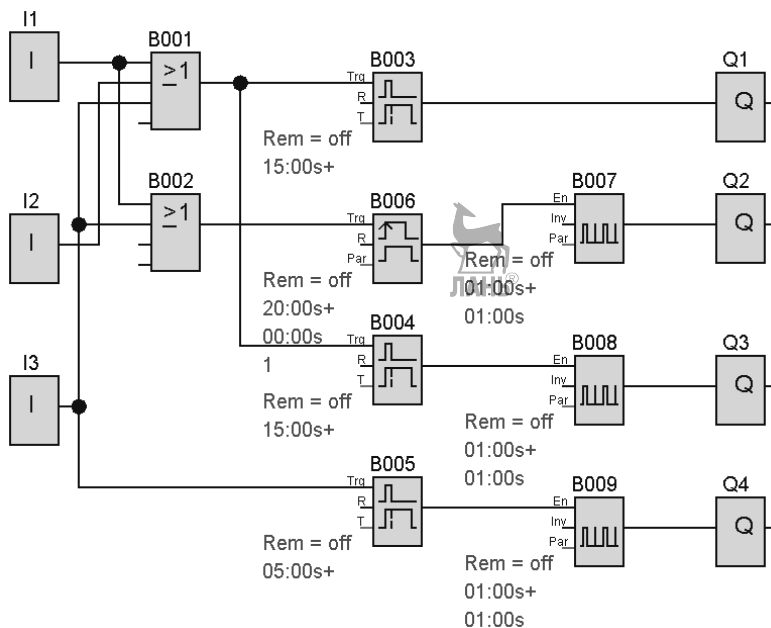


Рис. 8.5. Управляющая программа в LOGO! Soft Comfort

В программу входят следующие блоки:

- I1...I3 – цифровые входы;
- B001, B002 – функция ИЛИ;
- B003, B004, B005 – задержка отключения;
- B006 – интервальное реле с запуском по фронту;
- B007, B008, B009 – асинхронный генератор импульсов;
- Q1...Q4 – цифровые выходы.

Эта же программа, составленная в ONI PLR Studio, показана на рис.8.6. Видно, что программы в LOGO! Soft Comfort и ONI PLR Studio идентичны не только по содержанию, но и по форме. Обе программы имеют одинаковые блоки.

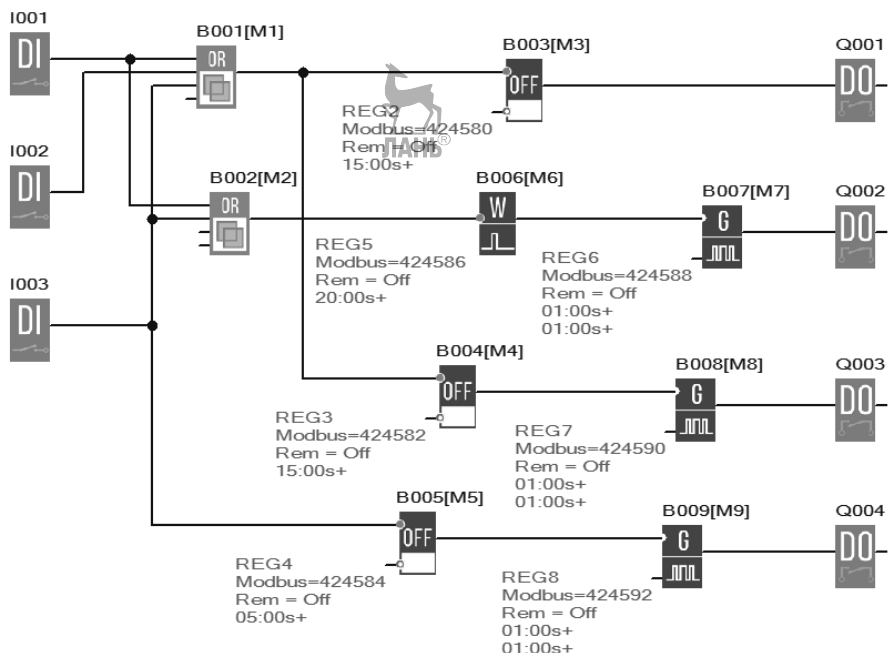


Рис. 8.6. Управляющая программа в ONI PLR Studio

Эта же программа, созданная в OWEN LOGIC, показана на рис.8.7. Здесь уже нет полной идентичности программ, как в предыдущем случае. В OWEN LOGIC нет блока «Интервальное реле с запуском по фронту». Для формирования сигнала заданной длительности (для звукового оповещателя) в OWEN LOGIC можно использовать блок TP. Кроме того, чтобы сложить три сигнала, в OWEN LOGIC надо брать два блока ИЛИ.

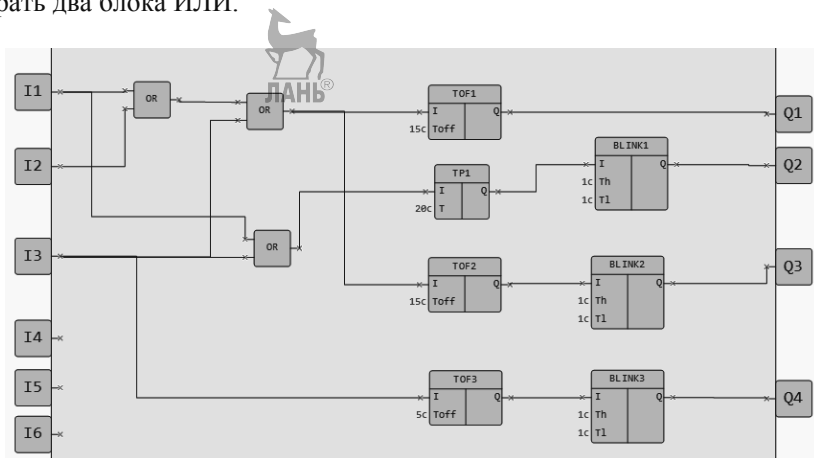


Рис. 8.7. Управляющая программа в OWEN LOGIC

8.4. Разработка электрической схемы.

Для проекта выбираем программируемое логическое реле LOGO! 230RCE с питанием 115...240 В переменного тока, 8 входов, 4 релейных выхода, ЖК-дисплей, встроенный таймер.

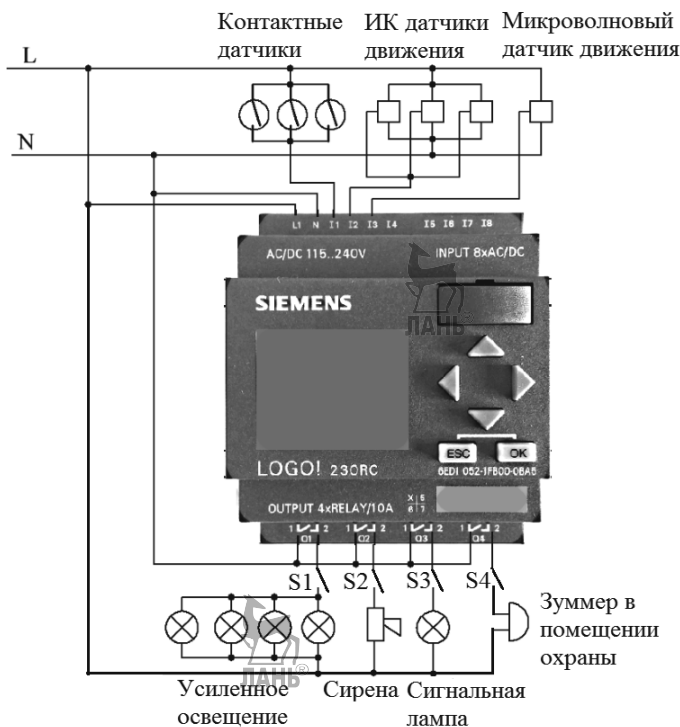


Рис. 8.8. Электрическая схема

Электрическая схема представлена на рис. 8.8. Контактные датчики и ИК датчики движения объединены параллельно в две группы. Таким образом, срабатывание любого контактного или ИК датчика приведет к подаче тревожного сигнала на контроллер. На выходах контроллера установлены ключи S1...S4, с помощью которых вручную можно отключать сигналы с выходов контроллера.

В соответствии с техническими характеристиками модуля LOGO!230RCE нагрузка на релейный выход по току не должна превышать 10 А. Если для усиленного освещения будут выбраны мощные лампы, то нагрузка на релейный выход может превысить допустимую. Тогда лампы надо подключать к выходу не напрямую, а через контактор. Но рекомендация не превышать ток на релейном выходе в 10 А не означает, что на каждом релейном выходе может быть по 10 А. Для защиты выходной цепи контроллера следует применять общий автоматический выключатель на 16А.

ГЛАВА 9. ЛАБОРАТОРНЫЕ РАБОТЫ

В лабораторных работах используется следующее оборудование:

- лабораторный стенд Инженерно-производственного центра (ИПЦ) «Учебная техника» (г. Челябинск);
- стационарные лабораторные стенды ГБПОУ ОКГ «Столица»;
- макетные лабораторные стенды, которые собираются студентами из отдельных компонентов в процессе выполнения лабораторной работы.

Из опыта проведения лабораторных работ следует, что наибольшее количество ошибок студенты совершают по следующим причинам:

- несоответствие входов и выходов в коммутационной программе и в подключенном контроллере;
- ошибки, связанные с переносом программы из компьютера в контроллер. В связи с этим приведено подробное описание процесса переноса коммутационной программы из компьютера в контроллер для модулей LOGO! шестой серии, модулей LOGO! восьмой серии, модулей ONI, модулей ОВЕН, которые изложены в подразделах 9.1.1, 9.1.7, 9.2.1, 9.3.1 соответственно.

Рекомендации по оформлению отчета

Отчет должен содержать:

- Титульный лист.
- Цель и краткое описание лабораторной работы (цель и средства для достижения цели).
- Коммутационную программу.
- Перечень блоков коммутационной схемы.
- Фото лабораторного стенда.
- Перечень приборов и компонентов лабораторного стенда.
- Электрическую схему соединений лабораторного стенда, выполненную студентом в какой-либо графической программе, например, AutoCAD или sPlan.
- Коммутационную программу дополнительного задания.
- Ответы на контрольные вопросы.

По усмотрению преподавателя, из описания лабораторной работы можно исключить коммутационную программу и электрическую схему с тем, чтобы студенты самостоятельно разрабатывали эти схемы.

Предупреждение. В некоторых лабораторных работах используются контроллеры с напряжением питания 220 вольт. Это сделано намеренно, чтобы студенты привыкали работать с реальным оборудованием. Напряжение 220 вольт является опасным для жизни, поэтому все работы по сборке электрических схем

проводятся под контролем преподавателя. Собранные схемы подключаются к источнику напряжения 220 вольт только после проверки преподавателем.

9.1. Лабораторные работы с контроллером Siemens LOGO!

9.1.1. Описание лабораторного стенда

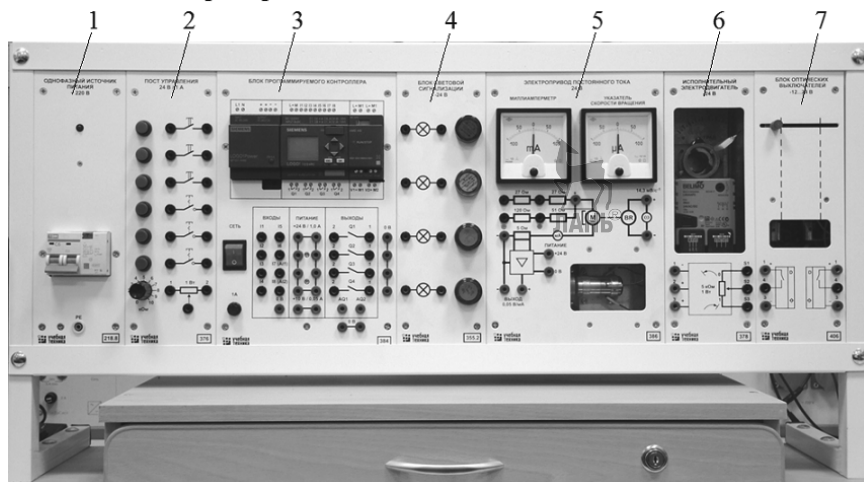


Рис. 9.1. Лабораторный стенд с контроллером Siemens LOGO!

Лабораторные работы с контроллером Siemens LOGO! шестой серии выполняются на учебно-лабораторных стендах модели АПК1-Н-Р ИПЦ «Учебная техника» (рис. 9.1). Лабораторный стенд АПК1-Н-Р состоит из отдельных модулей, которые могут вставляться и выниматься из рамы в зависимости от требований конкретной лабораторной работы. Наименования модулей, а также датчиков, используемых вместе с модулями, приведены в таблице 9.1.

Таблица 9.1

№ п/п	Наименование сменных модулей	Назначение и параметры	Обозначение
1	Однофазный источник питания.	~220 В/16 А	G1
2	Пост управления 24 В/1 А.	Предназначен для коммутации электрических цепей, имеет 3 кнопки без фиксации, 3 кнопки с фиксацией, потенциометр.	A2
3	Узел программируемого контроллера.	- Преобразователь LOGO! Power (6EP1331-1SH03), преобразует входное напряжение AC 100-240 В	A1

	Состоит из трех модулей: а) преобразователь напряжения LOGO! Power, б) контроллер LOGO!12 / 24RC шестой серии, в) аналоговый модуль AM2 AQ.	в выходное напряжение DC 24 В. - Контроллер Siemens LOGO!12/24RC (6ED1 052-1MD00-0BA6), имеет 6 цифровых входов, 2 цифровых (аналоговых) входа, 4 релейных выхода. - Аналоговый модуль AM2 AQ (6ED1 055-1MM00-0BA1) с двумя аналоговыми выходами: входное напряжение – DC 24 В; диапазон напряжения выходов – 0...10 В; нагрузка по напряжению ≥ 5 кОм; диапазон тока выхода – 0/4...20 мА.	
4	Блок световой сигнализации.	4 светодиодных лампы DC 24 В.	A3
5	Электропривод постоянного тока.	Электродвигатель DC 24 В, 8000 об/мин, в комплекте тахогенератор.	A10
6	Исполнительный электродвигатель (BELIMO LM24AP5).	DC 24 В, направление вращения – реверсивное, максимальный угол поворота – 95 град., время поворота – 150 с.	A7
7	Блок оптических выключателей.	DC 12...24 В, 2 оптических выключателя.	A4
8	Модуль отапливаемого помещения.	Состоит из лампы накаливания 24 В, используемой в качестве нагревателя, и датчика температуры 0,1 В/°C (TC125-50M.B2.60).	A8 (на рис. 9.1 не показан)
9	Преобразователь постоянного напряжения.	Преобразует входное напряжение DC 24 В в выходное напряжение DC 1,0...22 В.	A9 (на рис. 9.1 не показан)
10	Датчик освещенности (TSL250R).	Напряжение питания 2,7...5,5 В.	A5 (на рис. 9.1 не показан)
11	Зуммер.	24 В / 20 мА.	A6 (на рис. 9.1 не показан)

В данном подразделе частично используются лабораторные работы, изложенные в [3]. Принципиальным отличием является то, что в [3] лабораторные работы выполняются в режиме ручного программирования контроллера через

ЖК-дисплей, а в данном учебном пособии лабораторные работы выполняются с помощью компьютерного программирования в LOGO! Soft Comfort. Программирование контроллера заключается в составлении коммутационной программы (схемы), которая управляет работой контроллера. Для передачи коммутационной программы из компьютера в память контроллера будем использовать кабель LOGO!USB-Cable, который требует установки драйвера CH341. Этот драйвер прилагается к кабелю на компакт-диске или его можно скачать из Интернета. Опишем процесс установки драйвера и процесс передачи программы из компьютера в контроллер. Запускаем файл setup.exe драйвера. В появившемся окне (рис. 9.2.) нажимаем кнопку Install.

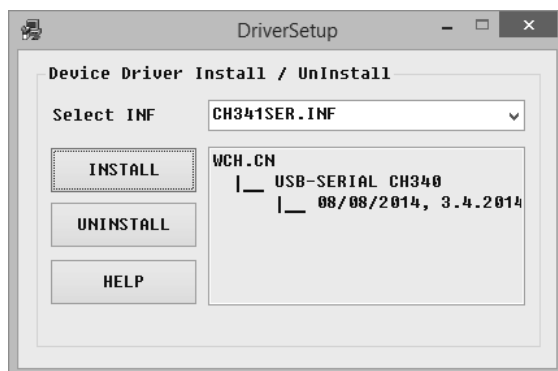


Рис. 9.2. Установка драйвера CH341

После установки драйвера подготавливаем стенд к лабораторной работе и коммутируем проводами необходимые входы и выходы на лабораторном стенде в соответствии с электрической схемой соединений, изложенной в лабораторной работе.

Внимание. Все коммутационные соединения на стенде выполняются при отключенном питании стенда.

Соединяем компьютер с контроллером с помощью LOGO!USB-Cable кабеля. После этого включаем стенд. С помощью курсора выбираем на экране ЖК-дисплея контроллера опцию *Program*. Чтобы передать программу из компьютера в контроллер, в панели инструментов программы LOGO! Soft Comfort нажимаем кнопку PC→LOGO! (или нажимаем комбинацию клавиш Ctrl + D). Получаем сообщение, показанное на рис. 9.3.

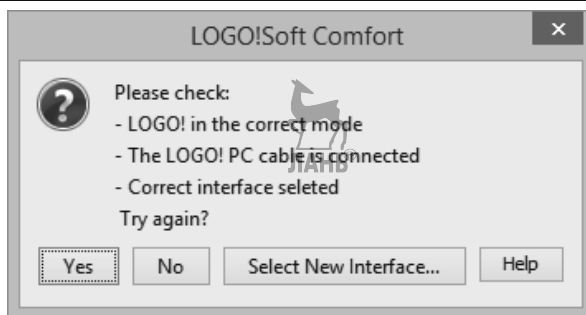


Рис. 9.3. Сообщение о необходимости выбора другого порта

Это сообщение означает, что надо выбрать другой COM-порт. Щелкаем в этом окне по клавише *Select New Interface*. Появляется окно рис. 9.4, в котором вместо COM1 выбираем COM3 (вместо COM3 может отобразиться другой порт, например, COM4) и щелкаем по кнопке ОК.

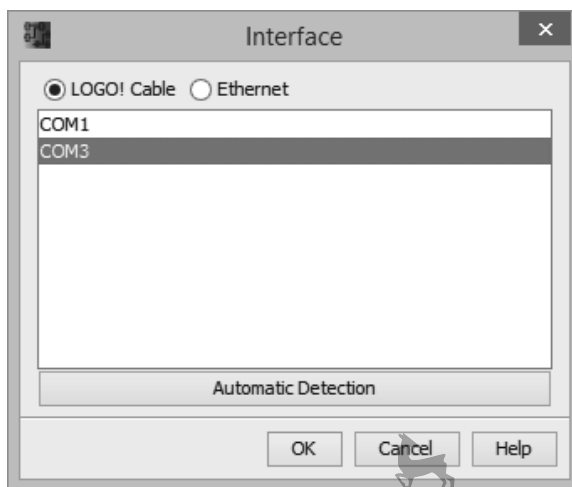


Рис. 9.4. Выбор другого порта

Появляется сообщение, представленное на рис. 9.5.

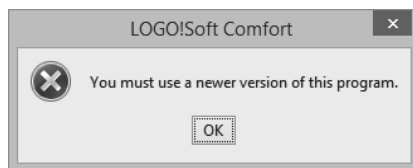


Рис. 9.5. Сообщение о необходимости выбора новой версии программы

Это сообщение означает, что компьютер затребовал более новую версию программы, нежели установленная на компьютере версия. Предположим, что на

компьютере была установлена версия LOGO!Soft Comfort V.7. Необходимо провести Upgrade программы до версии 8.1 или 8.2. Скачиваем с сайта компании Siemens программное обеспечение «Upgrade from LOGO! Soft Comfort: V1.0 / V2.0 / V3.x / V4.0 / V5.0 / V6.x / V7.0 to V8.2» применительно к установленной на компьютере версии Windows 32 Bit или 64 Bit. Устанавливаем ПО «Upgrade...» на компьютер. После установки автоматически запускается LOGO!Soft Comfort V8.2, в стандартной панели инструментов интерфейсного окна нажимаем кнопку PC → LOGO! Появляется окно, показанное на рис. 9.6. В верхней части этого окна около опции *Connect through* необходимо установить LOGO! Cable. Справа в соседней строчке вместо COM1 необходимо выбрать другой доступный порт, например, COM3 или COM4, как показано на рис 9.6. Щелкаем по кнопке ОК. Начинается процесс передачи программы из компьютера в контроллер LOGO! После окончания передачи можно запускать программу на лабораторном стенде. Для этого выбираем опцию *Start* на ЖК-дисплее контроллера и нажимаем клавишу ОК рядом с экраном.



Рис. 9.6. Установка параметров при передаче программы из компьютера в контроллер LOGO!

9.1.2. Лабораторная работа №1 «Тестирование таймеров»

Описание лабораторной работы.

В данной лабораторной работе тестируется работа таймеров:

- *Задержка включения;*
- *Задержка отключения;*
- *Задержка включения и отключения;*
- *Интервальное реле (импульсный выход);*

Порядок выполнения лабораторной работы.

1) Тестирование таймера *Задержка включения.*

- Запустить программу LOGO! Soft Comfort.
- Собрать коммутационную схему, показанную на рис. 9.7.

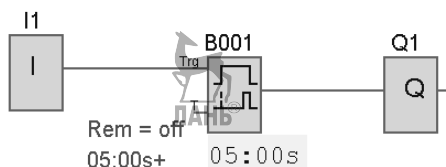


Рис. 9.7. Коммутационная схема

- Дважды щелкнуть по блоку B001 и установить задержку включения 5 с.
- Щелкнуть правой кнопкой мыши по блоку I1 и установить режим работы блока *Переключатель* (*Свойства > Эмуляция > Переключатель*). По умолчанию этот режим может быть уже установлен.
- Проверить работу коммутационной схемы. Для этого перевести программу в режим эмуляции *Сервис > Эмуляция* (или клавиша F3). Щелкнуть левой кнопкой мыши по блоку I1 и наблюдать отсчет времени под блоком B001. Через 5с сигнал проходит на выход Q1. Линия связи и контур блока Q1 меняют цвет с синего на красный. После этого можно переходить к работе со стендом.
- Собрать электрическую схему на лабораторном стенде, как показано на рис. 9.8.
- Соединить LOGO!USB-Cable кабелем компьютер с контроллером LOGO!
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер (например, *Сервис > Передача > ПК-> LOGO!*). Установить требуемый COM-порт. Программа будет загружена в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.
- По завершении эксперимента остановить коммутационную программу (ESC>Stop>Yes), **отключить выключатель «Сеть» в источнике питания G1.**



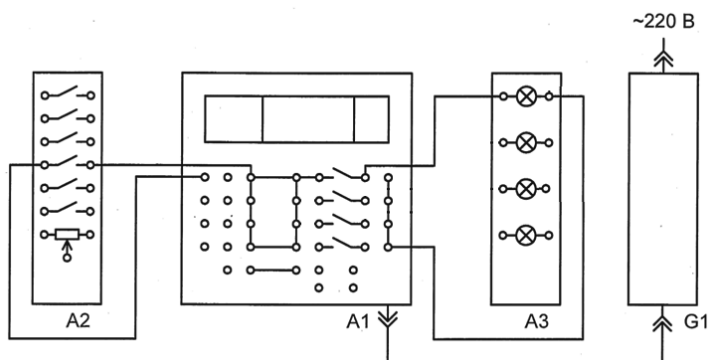


Рис. 9.8. Электрическая схема

2) Тестирование таймера *Задержка отключения.*

- Собрать коммутационную схему, показанную на рис. 9.9.

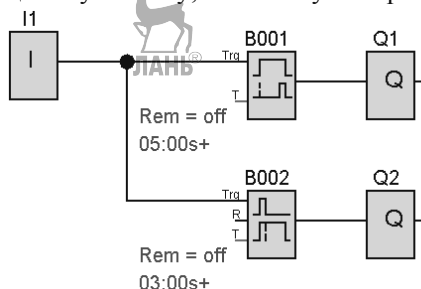


Рис. 9.9. Коммутационная схема

- Дважды щелкнуть по блоку B002 и установить задержку выключения.
- Проверить работу коммутационной схемы в режиме эмуляции.
- Собрать электрическую схему на лабораторном стенде, как показано на рис. 9.10.
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.
- По завершении эксперимента остановить коммутационную программу (ESC>Stop>Yes), **отключить выключатель «Сеть» в источнике питания G1.**

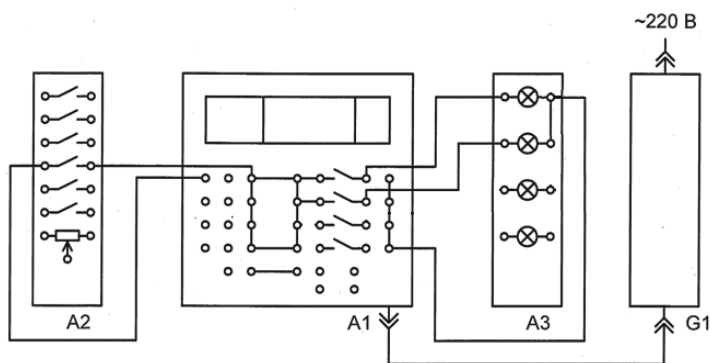


Рис. 9.10. Электрическая схема

3) Тестирование таймера *Задержка включения и отключения.*

- Собрать коммутационную схему, показанную на рис. 9.11.

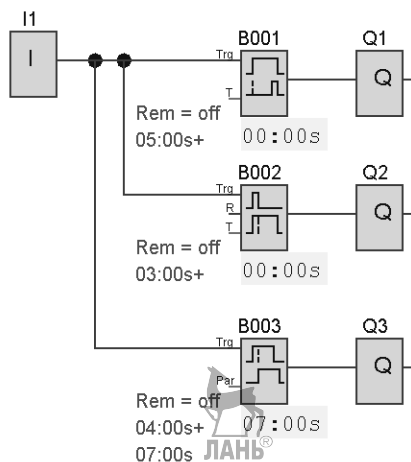


Рис. 9.11. Коммутационная схема

- Дважды щелкнуть по блоку B003 и установить задержку включения и отключения.
- Проверить работу коммутационной схемы в режиме эмуляции.
- Собрать электрическую схему на лабораторном стенде, как показано на рис. 9.12.
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.

- По завершении эксперимента остановить коммутационную программу (ESC>Stop>Yes), **отключить выключатель «Сеть» в источнике питания G1.**

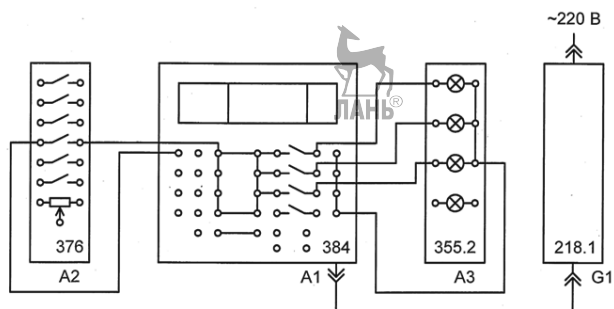


Рис. 9.12. Электрическая схема

4) Тестирование таймера *Интервальное реле (импульсный выход).*

- Собрать коммутационную схему, показанную на рис. 9.13.

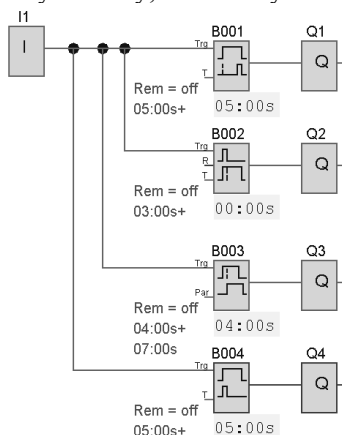


Рис. 9.13. Коммутационная схема

- Дважды щелкнуть по блоку B004 и установить время отключения.
- Проверить работу коммутационной схемы в режиме эмуляции.
- Собрать электрическую схему на лабораторном стенде, как показано на рис. 9.14.
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.

- По завершении эксперимента остановить коммутационную программу (ESC>Stop>Yes), **отключить выключатель «Сеть» в источнике питания G1.**

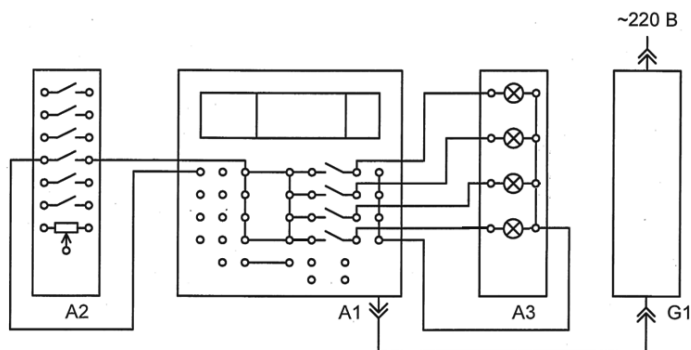


Рис. 9.14. Электрическая схема

Контрольные вопросы

1. Какие элементы подключаются к входам контроллера Siemens LOGO! в лабораторной работе?
2. Какие элементы подключаются к выходам контроллера Siemens LOGO! в лабораторной работе?
3. Каким образом коммутационная схема загружается в память контроллера?
4. Как запустить программу на исполнение после загрузки её в контроллер?
5. Привести условно-графические обозначения блоков: *Задержка включения, Задержка отключения, Задержка включения и отключения, Интервальное реле (импульсный выход).*
6. Что происходит в коммутационной схеме при подаче «1» на вход блока *Задержка включения?*
7. Что происходит в коммутационной схеме при подаче «1» на вход блока *Задержка отключения?*
8. С помощью какого таймера можно обеспечить автоматическое отключение выхода через заданное время после подачи «1» на вход схемы?

9.1.3. Лабораторная работа №2 «Тестирование аналоговых функций»

Описание лабораторной работы.

В данной лабораторной работе тестируется работа аналоговых функций:

- Аналоговый пороговый выключатель;
- Аналоговый дифференциальный выключатель;
- Широтно-импульсный модулятор (PWM).

Порядок выполнения лабораторной работы.

1) Тестирование функции Аналоговый пороговый выключатель.

Функция работает следующим образом. Когда значение аналоговой величины превысит *Порог включения* (например, On=300) на выходе появляется «1», которая сохраняется до тех пор, пока значение аналоговой величины будет меньше *Порога выключения* (например, Off=700). Значения порогов включения и отключения, а также параметров *Усиление* (Gain) и *Смещение* (Offset) устанавливаются в выпадающем окне, появляющемся после двойного щелчка левой кнопкой мыши по блоку *Аналоговый пороговый выключатель* (B001).

- Запустить программу LOGO! Soft Comfort.
- Собрать коммутационную схему, показанную на рис. 9.15.

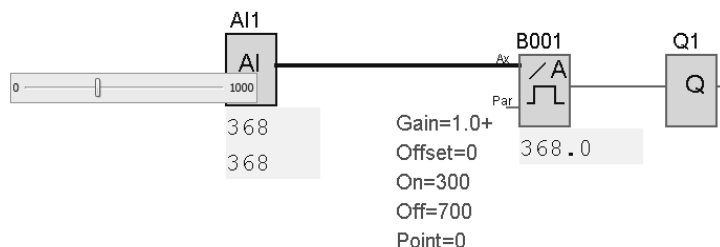


Рис. 9.15. Коммутационная схема

- Дважды щелкнуть по блоку B001 и установить в выпадающем окне значения параметров (рис. 9.16):
 - Усиление (Gain) – 1,00;
 - Смещение (Offset) – 0;
 - Порог включения (On) – 300;
 - Порог отключения (Off) – 700.

На выходе реального аналогового датчика существует напряжение 0...10 В. Поскольку это напряжение масштабируется к внутренним значениям контроллера 0...1000, то внутренние значения контроллера 300 и 700 будут соответствовать показаниям датчика 3 и 7 В. Напряжение 0...10 В в лабораторном стенде моделируется переменным резистором в модуле A2.



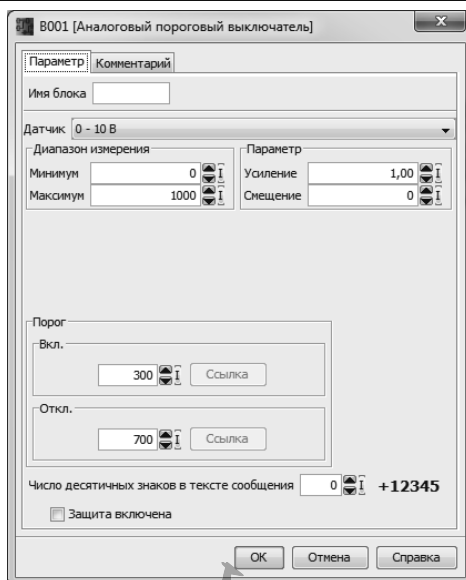


Рис. 9.16. Окно для задания параметров блока B001

- Проверить работу коммутационной схемы в режиме эмуляции.
- Собрать электрическую схему на лабораторном стенде, как показано на рис. 9.17.

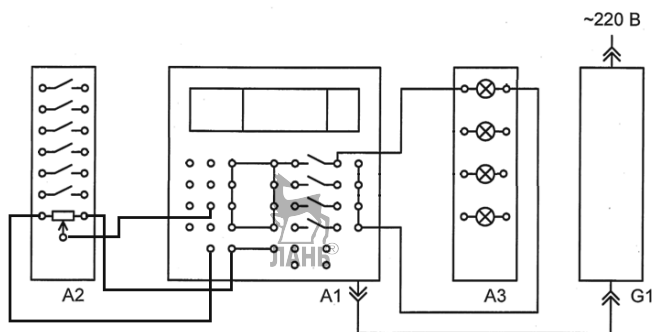


Рис. 9.17. Электрическая схема

- Соединить LOGO!USB-Cable кабелем компьютер с контроллером LOGO!
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.

- Вращая рукоятку переменного резистора поста управления A2, наблюдать загорание индикаторной лампы при положении рукоятки в пределах диапазона 3...7 В.
- По завершении эксперимента отключить выключатель «Сеть» в источнике питания G1 (выключить стенд).

2) Тестирование функции Аналоговый дифференциальный выключатель.

Функция работает следующим образом. Когда значение аналоговой величины превысит *Порог включения* ($On=300$) на выходе появляется «1», которая сохраняется до тех пор, пока значение аналоговой величины будет меньше значения ($On+Delta$), в нашем случае =400. Значения параметров On и $Delta$, а также параметров *Усиление* ($Gain$) и *Смещение* ($Offset$) устанавливаются в выпадающем окне, появляющемся после двойного щелчка левой кнопкой мыши по блоку *Аналоговый дифференциальный выключатель*.

- Собрать коммутационную схему, показанную на рис. 9.18.

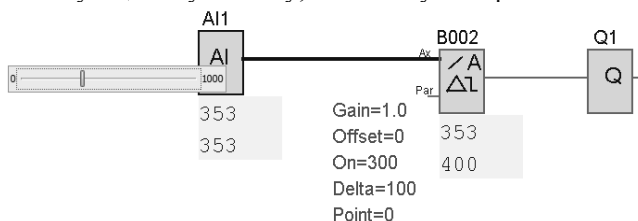


Рис. 9.18. Коммутационная схема

- Дважды щелкнуть по блоку B002 и установить параметры:
 - Усиление – 1,00;
 - Смещение – 0;
 - Вкл. (On) – 300;
 - Дифференциальный ($Delta$) – 100.
- Проверить работу коммутационной схемы в режиме эмуляции.
- Электрическую схему (рис. 9.17) оставить без изменений.
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.
- Вращая рукоятку переменного резистора модуля A2, наблюдать загорание индикаторной лампы при положении рукоятки в пределах диапазона напряжений 3...4 В.
- По завершении эксперимента отключить выключатель «Сеть» в источнике питания G1 (выключить стенд).

3) Тестирование функции *Широтно-импульсный модулятор (PWM)*.

Широтно-импульсный модулятор (PWM – Pulse Width Modulator) предназначен для преобразования аналогового входного сигнала в импульсный цифровой выходной сигнал. Ширина импульса цифрового сигнала пропорциональна значению A_x аналогового входного сигнала. Чем больше значение A_x , тем шире импульс.

В течение периода времени T генерируется прямоугольный импульс длительностью T_1 , в течение которого на выходе Q имеем сигнал «1», и остальное время $T_2 = T - T_1$ сигнал на выходе Q равен «0», как показано на рис. 9.19. Затем ситуация повторяется до тех пор, пока на входе «En» установлен высокий уровень. Длительность T_1 и T_2 , как уже отмечалось, определяется значением A_x .

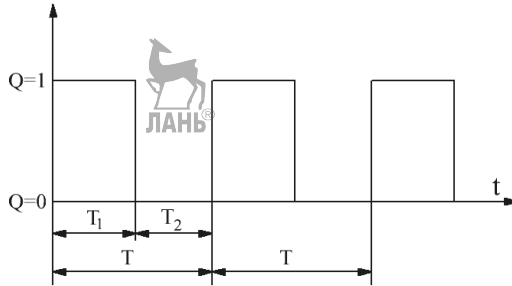


Рис. 9.19. Преобразование аналогового сигнала в цифровой сигнал

Рассмотрим конкретный пример. Пусть фактический диапазон аналоговой величины изменяется от -50 до 100°C . Указываем эти значения в секции *Диапазон измерения* на рис. 9.20. При этом контроллер автоматически вычисляет значения усиления (0,15) и смещения (-50).

Далее в окне на рис. 9.20 устанавливаются:

– *Диапазон мин.*: $A_{\min} = -50$ (минимальное значение измеряемой физической величины).

– *Диапазон макс.*: $A_{\max} = 100$ (максимальное значение измеряемой физической величины).

– *Период времени*: $T = 10$ сек.

Значения T_1 и T_2 вычисляются по формулам

$$T_1 = \frac{A_x - A_{\min}}{A_{\max} - A_{\min}} \cdot T \quad (9.1)$$

$$T_2 = T - T_1 \quad (9.2)$$

Пусть $A_x = 25$, тогда из (9.1) и (9.2) находим

$$T_1 = \frac{25 - (-50)}{100 - (-50)} \cdot 10 = 5 \text{ с}$$

$$T_2 = 10 - 5 = 5 \text{ с}$$

где A_x – текущее значение измеряемой физической величины.

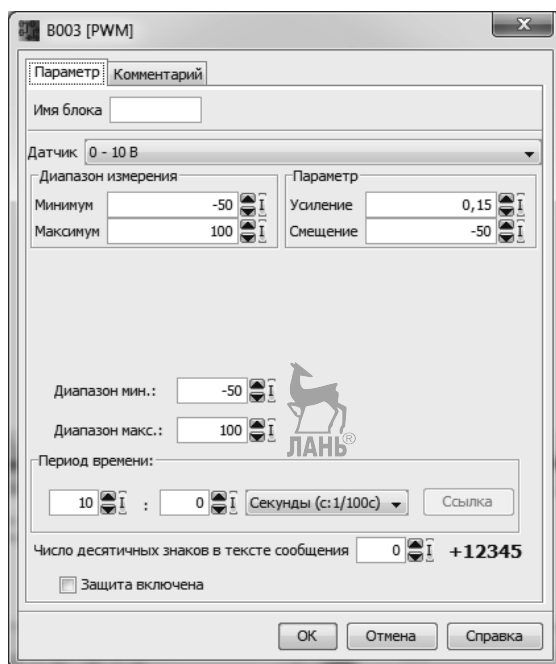


Рис. 9.20. Окно выбора параметров

Таким образом, при существовании на аналоговом входе значения $A_x = 25$ на выходе Q в течение 5 с будет сигнал «1» и в течение следующих 5 с – сигнал «0» и так далее (циклический процесс), пока на входе «En» установлен высокий уровень. При установке «0» на входе «En» происходит сбрасывание функции.

- Собрать коммутационную схему, показанную на рис. 9.21.
- Дважды щелкнуть по блоку B003 и установить параметры, указанные на рис. 9.20.
- Проверить работу коммутационной схемы в режиме эмуляции.
- Электрическую схему (рис. 9.17) оставить без изменений.
- Включить тумблер «Сеть» в источнике питания G1.
- Перевести контроллер с помощью клавиш в режим отображения «Главного меню» и выбрать команду «Program».
- Передать программу из компьютера в контроллер.
- Запустить программу на исполнение с помощью пункта «Start» Главного меню. Убедиться в правильной работе системы.
- Вращая рукоятку переменного резистора модуля A2, наблюдать изменение частоты мигания индикаторной лампы.

По завершении эксперимента отключить выключатель «Сеть» в источнике питания G1 (выключить стенд).

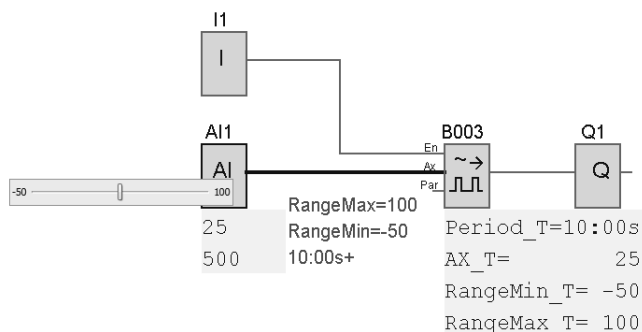


Рис. 9.21. Коммутационная схема

Контрольные вопросы

1. Какие элементы подключаются к входам контроллера Siemens LOGO! в лабораторной работе?
2. Какие элементы подключаются к выходам контроллера Siemens LOGO! в лабораторной работе?
3. Привести условно-графические обозначения блоков: *Аналоговый пороговый выключатель*, *Аналоговый дифференциальный выключатель*, *Широтно-импульсный модулятор*.
4. Как изменится алгоритм работы функции *Аналоговый дифференциальный выключатель*, если для Delta задать значение Delta = -100?
5. Для какой цели используется функция *Широтно-импульсный модулятор*?
6. От какого параметра зависит ширина импульса в широтно-импульсном модуляторе?
7. Какой сигнал получим на выходе функции *Широтно-импульсный модулятор*, если значение Ax на входе будет равно максимальному значению измеряемой физической величины?
8. Каким образом происходит согласование реального диапазона физической величины, измеряемой датчиком, с диапазоном значений электрического сигнала на выходе датчика?
9. Какой диапазон чисел используется внутри контроллера?
10. Какому реальному значению физической величины будет соответствовать ширина прямоугольного импульса в 2,5 с для схемы на рис. 9.21?



9.1.4. Лабораторная работа №3 «Автоматическая система импульсного регулирования температуры воздуха в помещении»

Описание лабораторной работы.

В данной лабораторной работе моделируется система импульсного регулирования температуры воздуха в помещении. Для этой цели на лабораторном стенде задействуется модуль отопляемого помещения. Модуль отопляемого помещения представляет собой замкнутую коробку (блок А8 на рис. 9.23), в которой в качестве нагревателя используется обычная лампа накаливания. Кроме того, внутри установлен температурный датчик, отслеживающий реальную температуру внутреннего пространства модуля (коробки). При отключении лампы происходит охлаждение внутреннего пространства модуля естественным путем до температуры окружающего воздуха.

Коммутационная программа представлена на рис. 9.22.

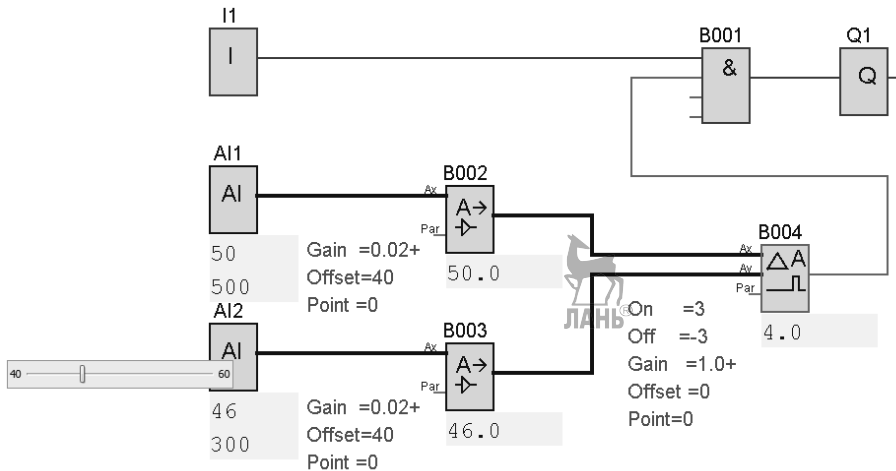


Рис 9.22. Коммутационная программа

В коммутационной программе использованы следующие функции:
 I1 – цифровой вход. Управляет включением / выключением системы.

AI1 – аналоговый вход контроллера. На вход AI1 подается заданное значение температуры (уставка) с переменного сопротивления (0...+10 В) лабораторного стенда.

AI2 – аналоговый вход контроллера. На вход AI2 подается реальная температура с датчика температуры модуля отопляемого помещения.

B001 – Функция «И».

B002, B003 – Аналоговые усилители. Осуществляют согласование физического диапазона температур с диапазоном допустимых внутренних значений контроллера (0...1000).

В004 – Аналоговый компаратор. Осуществляет сравнение аналоговых сигналов на входах A_x и A_y и управляет работой выхода Q1.

Q1 – выход программируемого контроллера. Контакты выхода включают/отключают нагреватель (лампу).

Программа работает следующим образом. В схеме на рис. 9.22 задействованы два датчика температуры: датчик AI1 задает поддерживаемое постоянное значение температуры воздуха в модуле отапливаемого помещения (уставку). На лабораторном стенде датчик AI1 моделируется потенциометром поста управления. Датчик AI2 измеряет текущую температуру воздуха в модуле отапливаемого помещения.

Программа переходит в режим регулирования температуры при установке «1» на входе П1. Коэффициент усиления 0,02 и смещение 40 на аналоговых усилителях (В002, В003) определяет физический диапазон изменения температур в модуле от 40 до 60°C. Соответствующий диапазон изменения нормализованных чисел в контроллере находится в пределах от 0 до 1000. Это следует из формулы

$$N = \frac{AI - b}{k}$$

Подставим в формулу значения $k = 0,02$; $b = 40$. Тогда при $AI = 40$ получим $N = 0$, при $AI = 60$ получим $N = 1000$. Таким образом, аналоговые усилители обеспечивают согласование реального диапазона температур (40...60°C) с диапазоном чисел (0...1000), с которым работает контроллер. Верхняя цифра под аналоговыми входами AI1, AI2 на рис. 9.22 соответствует реальной температуре, нижняя цифра соответствует нормализованному числу контроллера в диапазоне 0...1000. Заданное значение уставки 50°C (рис. 9.22, аналоговый вход AI1) соответствует нормализованному числу 500 в контроллере.

Выбранные значения диапазона температур (40...60°C) и уставки 50°C требуют пояснения. Возникает вопрос, почему выбрано такое высокое значение уставки. Это же не соответствует реальной действительности. Данный выбор уставки сделан исключительно в целях эксперимента, а не для отображения реальной действительности. Дело в том, что охлаждение модуля отапливаемого помещения происходит естественным путем до температуры окружающего воздуха. А поскольку лабораторный стенд находится в учебном кабинете, где реальная температура воздуха составляет 25°C, то если установить в модуле отапливаемого помещения в качестве уставки значение температуры также 25°C, то модуль не будет охлаждаться, и система работать не будет.

Аналоговый компаратор вычисляет разность заданного и измеренного значений температур ($A_x - A_y$). Если разность больше или равна порогу включения ($On \geq 3$), то выход компаратора устанавливается в «1». Сигнал «1» проходит на выход Q1 и включает нагреватель. Когда разность температур меньше или равна порогу отключения ($Off \leq -3$), выход компаратора переходит в состояние «0». Сигнал «0» проходит на выход Q1 и отключает нагреватель. Процесс повторяется циклически.

Схема электрических соединений представлена на рис. 9.23.

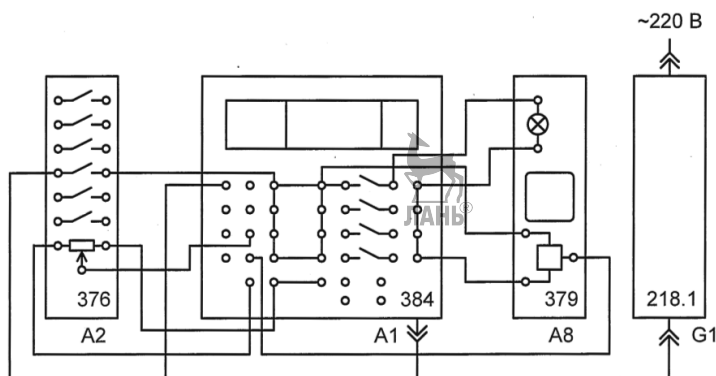


Рис. 9.23. Схема электрических соединений

Однофазный источник питания G1 предназначен для безопасного питания блока программируемого контроллера A1.

Кнопка с фиксацией поста управления предназначена для включения (отключения) системы.

Переменный резистор поста управления A2 используется для формирования регулируемого аналогового сигнала 0...+10 В, пропорционального уставке температуры. Чтобы изменить исходное значение уставки, надо вращать ручку потенциометра, расположенную слева от переменного резистора на блоке A1.

Датчик температуры и лампа накаливания внутри модуля отапливаемого помещения (A8) являются элементами контура управления. С их помощью поддерживается заданная температура в помещении.

Порядок выполнения работы.

- Запустите на компьютере программу LOGO!Soft Comfort и наберите коммутационную схему (рис. 9.22). Задайте параметры блоков B002, B003, B004, как указано на рис. 9.22.
- Запустите режим «Эмуляция» на панели программирования ПО LOGO!Soft Comfort и проверьте правильность работы коммутационной схемы.
- Убедитесь, что лабораторный стенд отключен от сети электропитания.
- Соедините проводами разъемы на лицевой части стенда в соответствии с электрической схемой соединений (рис. 9.23).
- Включите источник питания G1 стенда.
- Выберите строку «Program» на ЖК-дисплее контроллера.
- Загрузите в контроллер коммутационную схему из компьютера.
- Запустите программу на исполнение с помощью пункта «Start» на ЖК-дисплее. С помощью Поста управления подайте на вход I1 «1» (для этого замкните на Посте управления соответствующую кнопку). Убедитесь в правильной работе системы.

По завершении эксперимента остановите коммутационную программу (ESC>Stop>Yes), **отключите выключатель «Сеть» в источнике питания G1.**

Контрольные вопросы



1. Какие элементы подключаются к входам контроллера Siemens LOGO! в лабораторной работе?
2. Какие элементы подключаются к выходам контроллера Siemens LOGO! в лабораторной работе?
3. Почему автоматическая система управления температурой в данной лабораторной работе называется импульсной?
4. Что означает термин «уставка»?
5. Какую функцию выполняет аналоговый усилитель?
6. Как работает аналоговый компаратор?

9.1.5. Лабораторная работа №4 «Автоматическая система регулирования температуры воздуха в помещении с помощью ПИ-регулятора»

Описание лабораторной работы.

В данной лабораторной работе моделируется система непрерывного регулирования температуры воздуха в помещении с помощью ПИ-регулятора. В эксперименте участвует модуль отапливаемого помещения, который использовался и в предыдущей лабораторной работе.

Коммутационная программа представлена на рис. 9.24.

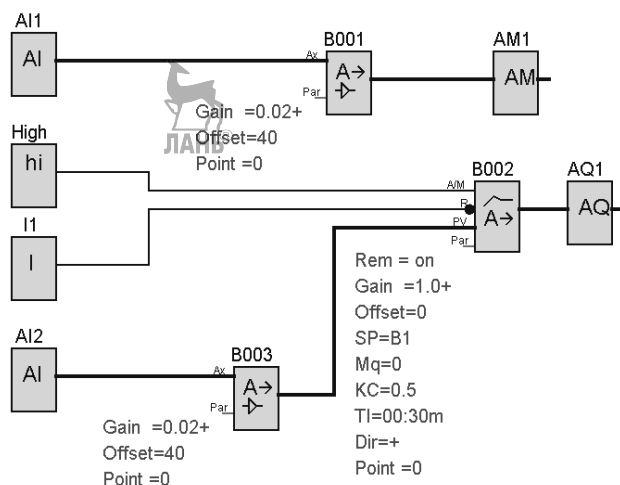


Рис. 9.24. Коммутационная программа

В коммутационной программе использованы следующие функции:
 I1 – цифровой вход. Управляет включением / выключением системы.

AI1 – аналоговый вход контроллера. На вход AI1 подается заданное значение температуры (уставка) с переменного сопротивления (0...+10 В) лабораторного стенда.

AI2 – аналоговый вход контроллера. На вход AI2 подается реальная температура с датчика температуры, расположенного внутри модуля отапливаемого помещения.

V001, V003 – Аналоговые усилители. Осуществляют согласование физического диапазона значений температуры с диапазоном допустимых внутренних значений контроллера (0...1000).

V002 – ПИ-регулятор.

AM1 – аналоговый флаг (переменная программы). Необходим для завершения цепочки блоков, не имеющей подключения к выходу контроллера.

AQ1 – аналоговый выход программируемого контроллера. Напряжение выхода управляет нагревателем (лампой).

В схеме на рис. 9.24 вместо аналогового компаратора используется ПИ-регулятор, который, в отличие от аналогового компаратора, осуществляет непрерывную коррекцию температуры.

Программа работает следующим образом. Датчик AI1 задает постоянное значение температуры воздуха в помещении (уставку), датчик AI2 измеряет текущую температуру воздуха в помещении. (На лабораторном стенде датчик AI1 моделируется потенциометром поста управления). Выход AQ1 включает и выключает нагреватель (лампу).

Сигнал «0» с входа контроллера П поступает на вход сброса «R» ПИ-регулятора V002 и блокирует его работу независимо от состояния других входов этого блока. На выходе ПИ-регулятора и аналоговом выходе контроллера AQ1 устанавливается сигнал «0». Нагреватель отключен.

При подаче «1» на вход П снимается блокировка с ПИ-регулятора. Сигнал «1» (блок High) на входе А/М блока V002 задает режим автоматической работы ПИ-регулятора. Программа переходит в режим регулирования температуры.

Сигнал AI2 поступает на вход сигнала обратной связи (PV) ПИ-регулятора. ПИ-регулятор определяет разность заданного (SP) и измеренного (PV) значений температуры и вычисляет выходной сигнал управления нагревателем, поступающий на аналоговый выход контроллера AQ1. Заданное значение температуры (SP) передается в блок ПИ-регулятора косвенно, как ссылка на номер блока усилителя V001, вычисляющего эту величину. Ссылка на блок V001 указывается в параметрах ПИ-регулятора в выпадающем окне, которое показано на рис. 9.25

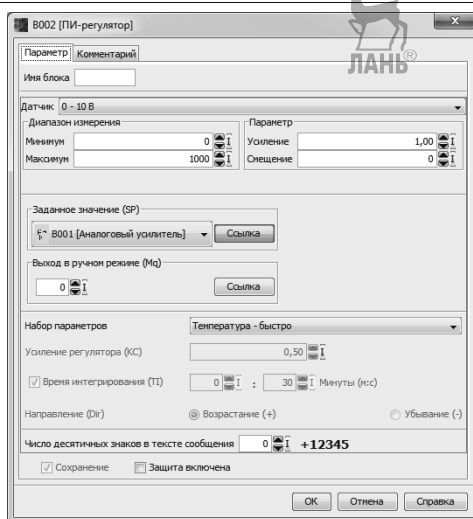


Рис. 9.25. Окно для настройки параметров ПИ-регулятора

Параметры ПИ-регулятора (блока В002) означают следующее:

$SP=B1$. Требуемое значение температуры. Задано, как ссылка на блок усилителя (блок В001), вычисляющего это значение.

$KC=0,5$. Усиление ПИ-регулятора.

$Ti=00:30$. Время интегрирования (30 с).

$Dir=+$. Направление действия регулятора (знак «+» означает, что выходной сигнал регулятора увеличивается, если текущее значение температуры ниже заданной величины).

$Mq=0$. Значение сигнала на выходе АQ при ручном режиме ($A/M=0$). В рассматриваемой программе ручной режим ПИ-регулятора не используется.

$r=0$ – число десятичных знаков в тексте сообщения.

Схема электрических соединений представлена на рис. 9.26.

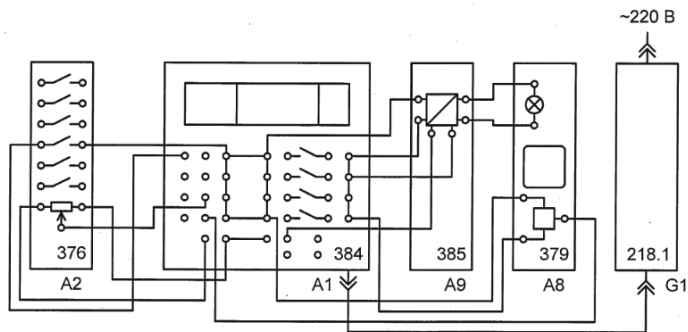


Рис. 9.26. Схема электрических соединений

Кнопка с фиксацией поста управления предназначена для включения (отключения) системы. При отключенной системе выходной сигнал ПИ-регулятора (аналоговый выход АQ1) равен 0.

Переменный резистор поста управления используется для формирования регулируемого аналогового сигнала $0...+10$ В, пропорционального уставке температуры воздуха в помещении в диапазоне $40...60^{\circ}\text{C}$. Чтобы изменить исходное значение поддерживаемой температуры (уставки), надо вращать ручку потенциометра, расположенную слева от переменного резистора.

Датчик температуры внутри модуля (А8) отапливаемого помещения формирует сигнал, пропорциональный температуре воздуха в помещении, т.е. этот датчик отображает текущую температуру воздуха в помещении.

Лампа накаливания в модуле отапливаемого помещения играет роль нагревательного элемента.

При включении системы выходной сигнал ПИ-регулятора поступает на аналоговый выход АQ1 контроллера и через преобразователь постоянного напряжения А9 задает напряжение питания лампы (нагревателя) в блоке А8.

Порядок выполнения работы.

- Запустить на компьютере программу LOGO!Soft Comfort и набрать коммутационную схему (рис. 9.24). Задать параметры блоков В001, В002, В003 как указано на рисунке.
- Запустить режим «Эмуляция» на панели программирования ПО LOGO!Soft Comfort и проверить правильность работы коммутационной схемы.
- Убедиться, что лабораторный стенд отключен от сети электропитания.
- Соединить проводами разъемы на лицевой части стенда в соответствии с электрической схемой соединений (рис. 9.26).
- Включить источник питания G1 стенда.
- Выбрать строку «Program» на ЖК-дисплее контроллера.
- Загрузить в контроллер коммутационную схему из компьютера.
- Запустить программу на исполнение с помощью пункта «Start» на ЖК-дисплее контроллера.
- Проверить состояние кнопки включения/отключения системы (кнопка с фиксацией поста управления). Установить её в состояние «замкнуто» – на вход П1 подан высокий уровень, система включена. Заданное и измеренное значение температуры отображаются на экране состояния аналоговых входов АI дисплея контроллера. Для перехода к указанному экрану состояния необходимо нажать кнопку управления курсором на передней панели контроллера (▶) до появления на дисплее изображения, аналогичного изображению на рис. 9.27. (Отображаемые значения показаны условно, они могут быть другими.)

AI:	
1 :	00600
2 :	00504
3 :	00000

Рис. 9.27. ЖК-дисплей контроллера с показаниями значений аналоговых величин.

В строке «1:» отображается аналоговая величина AI1, соответствующая заданной температуре. Значение 00600 соответствует температуре 60°C. В строке «2:» отображается аналоговая величина AI2, соответствующая текущей измеренной температуре. Значение 00504 соответствует температуре 50,4°C. Вращая переменный резистор поста управления, зададим требуемое значение температуры в помещении, например, SP = 50°C. Этому значению температуры соответствует внутреннее нормализованное значение контроллера 500. Установим с помощью переменного резистора в строке «1:» значение 00500. В строке «2:» будет отображаться значение текущей температуры. Так как в начальный момент текущая температура в модуле отапливаемого помещения меньше температуры 50°C, то в модуле загорится лампа (что равносильно включению нагревателя). Температура в помещении начнет повышаться. По инерции температура проскочит заданное значение температуры 50°C и поднимется выше. Лампа погаснет, и температура начнет понижаться. По инерции температура опять проскочит значение 50°C и опустится ниже. Лампа (нагреватель) включится, и температура внутри помещения начнет повышаться. Затем этот колебательный процесс будет повторяться.

- По завершении эксперимента остановить коммутационную программу (ESC>Stop>Yes), **отключить выключатель «Сеть».**

Контрольные вопросы

1. Какие элементы подключаются к входам контроллера Siemens LOGO! в лабораторной работе?
2. Какие элементы подключаются к выходам контроллера Siemens LOGO! в лабораторной работе?
3. Каким датчикам соответствуют аналоговые величины AI1 и AI2?
4. Какое назначение ПИ-регулятора?
5. Каким параметрам системы соответствуют обозначения SP, PV?
6. Какие две составляющие физической величины отслеживает ПИ-регулятор?

9.1.6. Лабораторная работа №5 «Система автоматического регулирования скорости двигателя постоянного тока с помощью П-, И-, ПИ-регуляторов»

Описание лабораторной работы.

В данной лабораторной работе моделируется система непрерывного регулирования скорости двигателя постоянного тока с помощью П-, И-, ПИ-регуляторов.

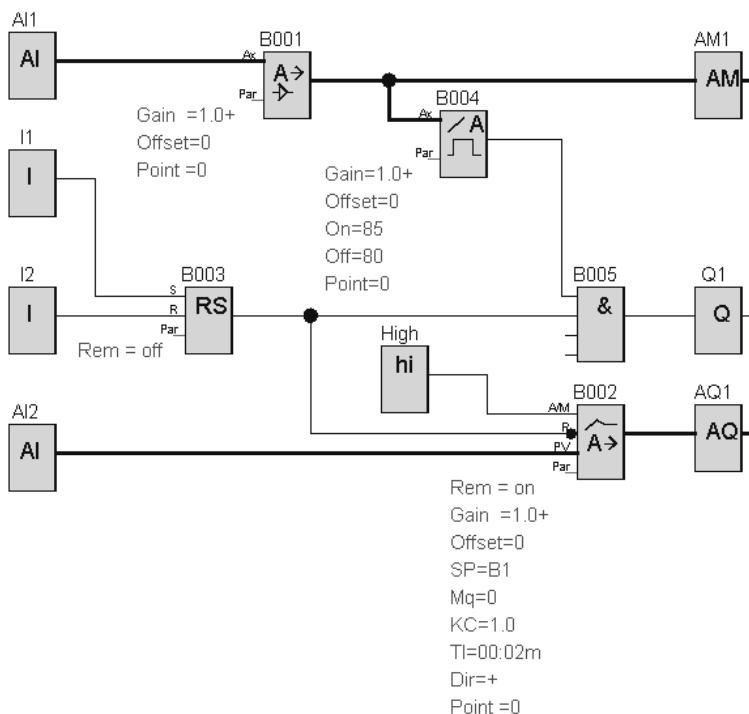


Рис. 9.28. Коммутационная программа

Коммутационная программа представлена на рис. 9.28. В программе использованы следующие функции:

I1, I2 – цифровые входы. Управляют включением / выключением системы.

AI1, AI2 – аналоговые входы контроллера. На вход AI1 подается сигнал задания скорости вращения двигателя с переменного резистора (0...+10 В). На вход AI2 – сигнал тахогенератора.

B001 – аналоговый усилитель. Для блока установлен коэффициент усиления Gain=1 и смещение Offset=0.

B002 – ПИ-регулятор. Вход сброса «R» (второй сверху) инвертирован. При сигнале 0 на инвертированном входе «R» регулятор отключен, на его выходе «0».

AM1 – аналоговый флаг (переменная программы). Необходим для завершения цепочки блоков, не имеющей подключения к выходу контроллера.

AQ1 – аналоговый выход программируемого контроллера. Напряжение выхода управляет напряжением питания двигателя.

B003 – Реле с блокировкой (RS-триггер). Сигнал S=1 (верхний вывод) устанавливает «1»

B003 – Реле с блокировкой (RS-триггер). Сигнал S=1 (верхний вывод) устанавливает «1»

B004 — аналоговый пороговый выключатель. Для блока установлен порог включения $Op=85$ (0,85 В), выключения $Off=80$ (0,80 В).

B005 – функция «И».

Q1 – выход программируемого контроллера. Контакты выхода Q1 управляют цепью питания двигателя.

Программа работает следующим образом. При запуске программы на выходе триггера B003 устанавливается «0». Этот сигнал блокирует работу регулятора B002 и напряжение на аналоговом выходе контроллера AQ1 равно «0». Этот же сигнал устанавливает «0» на выходе B005, при этом контакты выхода Q1 размыкают цепь питания двигателя.

Кратковременное нажатие на кнопку «Пуск» приводит к появлению «1» на входе И1 и установке «1» на выходе триггера B003. Регулятор B002 разблокирован. Сигнал «1» (блок High) на входе А/М B002 задает режим автоматической работы регулятора.

Пока сигнал задания скорости на входе АИ1 не превысит 0,85 В выход аналогового порогового выключателя B004 остается в состоянии «0». Сигнал «0» сохраняется на выходе B005, контакты выхода Q1 разомкнуты и разорвана цепь питания двигателя. Это исключает вращение двигателя за счет постоянного остаточного напряжения на выходе преобразователя (А9).

Когда сигнал задания скорости превысит порог срабатывания выключателя B004 (≥ 85), то на обоих входах B005 будут установлены «1», замкнутся контакты Q1 и двигатель будет подключен к выходу преобразователя (А9). Система перейдет в режим регулирования скорости.

Сигнал задания скорости (SP) на входе контроллера АИ1 меняется от 0 до 10 В, что соответствует диапазону изменения сигнала $SP=0 \dots 1000$ на входе АИ1 коммутационной программы. Аналоговый усилитель (B001) с параметрами $Gain=1$ и $Offset=0$ не изменяет значения сигнала, и необходим лишь для передачи значения SP блоку регулятора B002. Для завершения печочки блоков АИ1, B001 выход усилителя подключен к АМ1 (аналоговый флаг).

Напряжение тахогенератора (датчика скорости) поступает на вход контроллера и передается на вход сигнала обратной связи (PV) регулятора B002. Напряжение тахогенератора 0...10 В преобразуется в контроллере в сигнал с диапазоном 0...1000.

Регулятор (B002) определяет разность заданного (SP) и измеренного (PV) значений скорости и вычисляет выходной сигнал управления двигателем, поступающий на аналоговый выход контроллера AQ1. Заданное значение скорости (SP) передается в блок регулятора косвенно, как ссылка на номер блока усилителя B001, вычисляющего эту величину ($SP=B1$).

На блоке ПИ-регулятора (В002) необходимо установить следующие параметры:

$SP=B1$ – требуемое значение скорости. Задано как ссылка на номер блока В001.

$KC=1,00$ (Усиление). Коэффициент усиления пропорционального и интегрирующего звеньев одинаков и равен KC . При $KC=0$ пропорциональное звено обнуляется, остается только интегрирующее звено, т.е. получаем И-регулятор.

$TI=00:02$ (2 с) – время интегрирования. (Для ПИ-регулятора $TI=2$ с.

Для И-регулятора аналогично $TI=2$ с. Для П-регулятора интегрирующее звено необходимо отключить. Для этого устанавливаем максимальное значение $TI=99:59$.)

Предпочтительные значения параметров для регуляторов различного типа приведены в таблице 9.2.

Таблица 9.2

Тип регулятора	П	И	ПИ
Усиление KC	1,0...2,0	0	1,0
Время интегрирования TI , с	99:59	00:02	00:02

$Dir=+$. Направление действия регулятора («+» выходной сигнал регулятора увеличивается, если текущее значение скорости ниже заданной величины).

$Mq=0$. Значение сигнала на выходе AQ при ручном режиме ($A/M=0$). В рассматриваемой программе ручной режим регулятора не используется.

$Min=0$. Минимальное значение для PV .

$Max=1000$ Максимальное значение для PV .

A (Gain) = 1,0+. Усиление PV , равное +1.

B (Offset) = 0. Смещение нулевой точки PV .

$r=0$ – количество знаков после запятой.

Кратковременное нажатие на кнопку «Стоп» приводит к появлению «1» на входе $I2$ и установке «0» на выходе триггера В003. При этом регулятор В002 заблокирован, система отключена.

Схема электрических соединений показана на рис. 9.29.

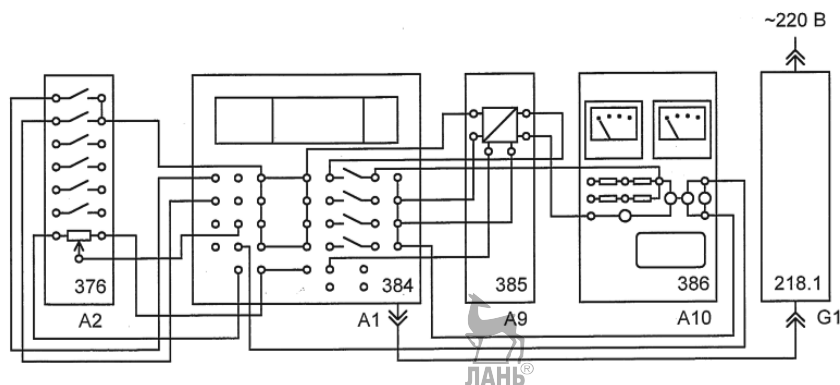


Рис.

9.29. Схема электрических соединений

Однофазный источник питания G1 предназначен для безопасного питания блока программируемого контроллера.

Кнопки поста управления предназначены для включения (отключения) системы и подключены, соответственно, к входам контроллера I1 и I2.

Переменный резистор в poste управления используется для формирования аналогового сигнала 0...+10 В, подключенного к входу AI1 контроллера и задающего скорость вращения двигателя.

Сигнал обратной связи (выход тахогенератора) подключен к входу AI2 контроллера.

Сигнал ПИ-регулятора с аналогового выхода контроллера AQ1 через преобразователь постоянного напряжения (блок A9) поступает в цепь питания двигателя. Преобразователь (блок A9) согласует диапазон выходных напряжений контроллера 0...+10 В с напряжением питания двигателя 0...+24 В. Двигатель подключен к выходу преобразователя через контакты выхода Q1 контроллера. Контакты Q1 размыкаются при нулевом сигнале управления, что исключает вращение двигателя из-за остаточного напряжения 1,7 В на выходе преобразователя.

Порядок выполнения работы.

- Запустите на компьютере программу LOGO!Soft Comfort и наберите коммутационную схему (рис. 9.28). Задайте параметры блоков B001, B002, B004, как указано на рисунке.
- Запустите режим «Эмуляция» на панели программирования ПО LOGO! Soft Comfort и проверьте правильность работы коммутационной схемы.
- Убедитесь, что лабораторный стенд отключен от сети электропитания.
- Соедините проводами разъемы на лицевой части стенда в соответствии с электрической схемой соединений (рис. 9.29).
- Включите источник питания G1 стенда.
- Выберите строку «Program» на ЖК-дисплее контроллера.
- Загрузите в контроллер коммутационную схему из компьютера.
- Запустите программу на исполнение с помощью пункта «Start».

- Кратковременно нажмите кнопку «Пуск» (верхняя кнопка с возвратом поста управления). Система включена. Заданное и фактическое значение аналоговых величин отображаются на экране состояния аналоговых входов AI контроллера, например, как показано на рис. 9.30. Для перехода к указанному экрану необходимо нажать кнопку управления курсором на корпусе контроллера (▶) до появления экрана состояния входов AI1, AI2.

AI:	
1:	00500
2:	00670
3:	00000

Рис. 9.30. ЖК-дисплей контроллера с показаниями значений аналоговых величин.

В строке «1:» на экране отображается напряжение на входе задания скорости AI1 (0. . .+10 В) умноженное на 100, т. е. диапазон значений AI1=0...1000.

В строке «2:» отображается напряжение на входе AI2, т.е.напряжение на выходе тахогенератора, пропорциональное текущему значению скорости двигателя. Это значение также умножено на 100, т. е. значение 00504 в таблице соответствует 5,04 В. Разность значений в строках «1:» и «2:» характеризует работу регулятора и всей системы в целом.

Далее следует проверить работу системы с П-, И-, ПИ-регуляторами. Эффективность работы различных регуляторов можно отслеживать по экрану на рис. 9.30, сравнивая заданное (AI1) и текущее (AI2) значения аналоговых величин.

- По завершении эксперимента остановите коммутационную программу (ESC>Stop>Yes), **отключите выключатель «Сеть» в источнике питания G1.**

Контрольные вопросы

1. Какие элементы подключаются к входам контроллера Siemens LOGO! в лабораторной работе?
2. Какие элементы подключаются к выходам контроллера Siemens LOGO! в лабораторной работе?
3. Как задать в программе связь блоков В001 и В002?
4. Какой регулятор П, И, или ПИ обеспечивает наиболее эффективное управление вращением двигателя?

9.1.7. Лабораторная работа №6 «Подсветка взлетной полосы»

Описание лабораторной работы.

Предположим на аэродроме есть несколько взлетных полос (рис. 9.31). Каждая полоса подсвечивается по бокам левым и правым рядом фонарей. Для того, чтобы пилоту самолета было проще отличить задействованную взлетную полосу от остальных, предлагается циклический алгоритм её подсвечивания. Например, оба ряда фонарей (левый и правый) одновременно мигают три раза, затем некоторое время горят постоянно. Затем опять мигают три раза и горят постоянно, т.е. цикл повторяется. Коммутационная программа, работающая по изложенному алгоритму, представлена на рис. 9.32.



Рис. 9.31. Взлетная полоса аэродрома

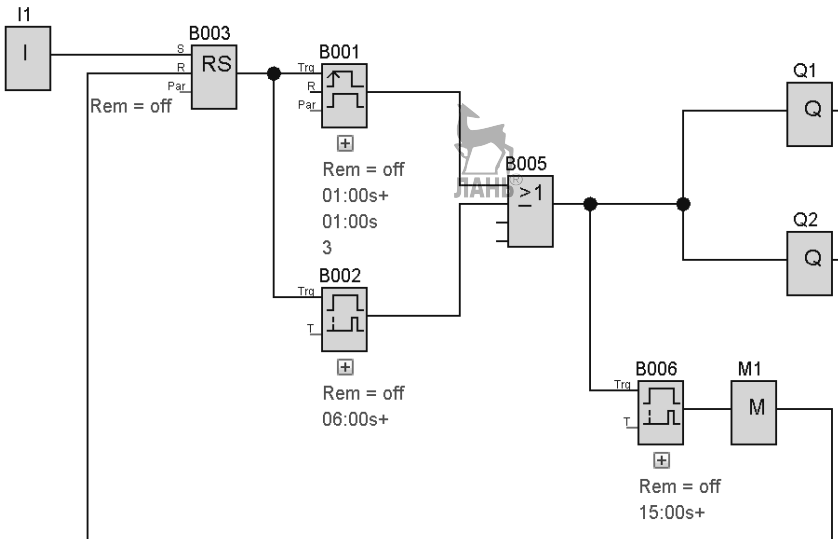


Рис. 9.32. Коммутационная программа

Для практической реализации коммутационной программы соберем макетный лабораторный стенд из отдельных компонентов. Компоненты, используемые для стенда, показаны на рис. 9.33.

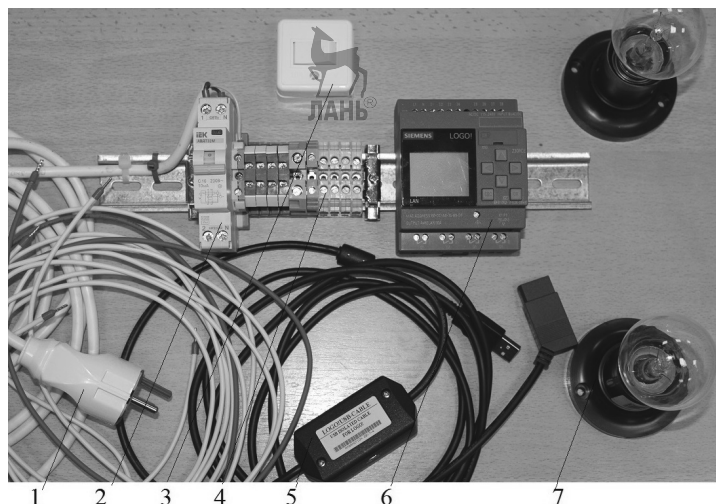


Рис. 9.33. Компоненты лабораторного макета

Цифрами на рис 9.33 обозначены:

1. Соединительные провода.
2. Автоматический выключатель дифференциального тока iEK АВДТ32М С16 10 мА, предназначен для защиты человека от поражения электрическим током при повреждении изоляции электроустановок, а также для защиты от перегрузки и короткого замыкания в сетях переменного тока напряжением 230 В, частотой 50 Гц.
3. Выключатель (ключ).
4. DIN-рейка с клеммными колодками. (Клеммная колодка «Земля» имеет металлический контакт с DIN-рейкой).
5. Соединительный кабель LOGO!USB-CABLE. (**Примечание.** Для программируемого реле восьмого поколения LOGO! 230RCE 6ED1052-1FB08-0BA0, который используется в лабораторной работе, соединительный кабель LOGO!USB-CABLE не применяется. Для переноса программы из компьютера в модуль LOGO! используется либо карта памяти Micro SD, либо кабель Ethernet).
6. Программируемое логическое реле восьмого поколения LOGO! 230RCE 6ED1052-1FB08-0BA0, напряжение питания/входов/выходов: 115 В/230 В/ реле, 8 DI/4DO, программное обеспечение LOGO! Soft Comfort версии 8.1 и выше.
7. Лампы 220 В, 40 Вт – 2 шт.

Электрическая схема лабораторного стенда представлена на рис. 9.34.

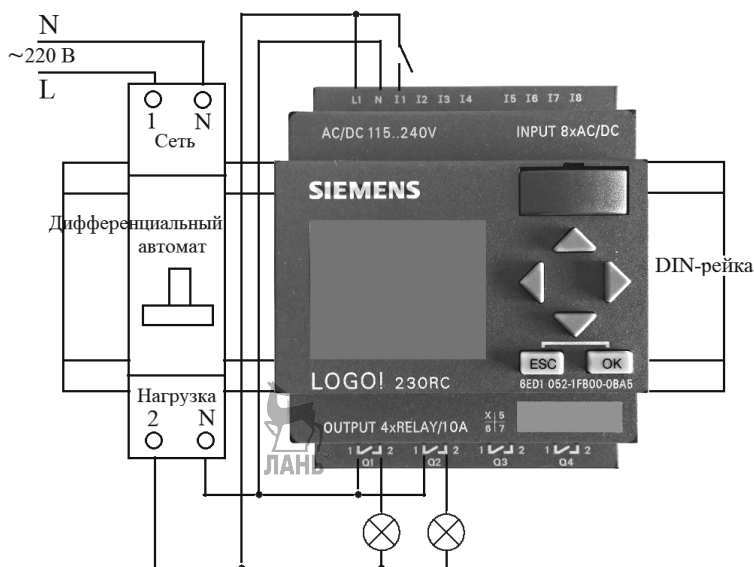


Рис. 9.34. Электрическая схема соединений

Порядок выполнения работы.

- Запустить на компьютере программу LOGO! Soft Comfort и подготовить коммутационную программу. Задать параметры блоков, как указано в коммутационной программе.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Собрать лабораторный стенд из компонентов в соответствии с электрической схемой соединений.
- Включить источник питания.
- Загрузить в контроллер коммутационную программу.
- Запустить программу на исполнение с помощью пункта «Start».
- По завершении эксперимента остановить коммутационную программу и **отключить выключатель «Сеть»**.

Рассмотрим способы переноса коммутационной программы из компьютера в модуль LOGO! восьмого поколения.

Загрузка программы в модуль LOGO! с помощью карты памяти Micro SD.

Необходимо сохранить коммутационную программу на Micro SD карту в формате Binary dump (*.bin), как показано на рис. 9.35. Модуль LOGO! поддерживает карты Micro SD с форматом файловой системы FAT32. Перед применением карту памяти желательно отформатировать в этом формате. При этом на карте памяти автоматически создается системный файл, который не препятствует нормальному переносу коммутационной программы в модуль LOGO!

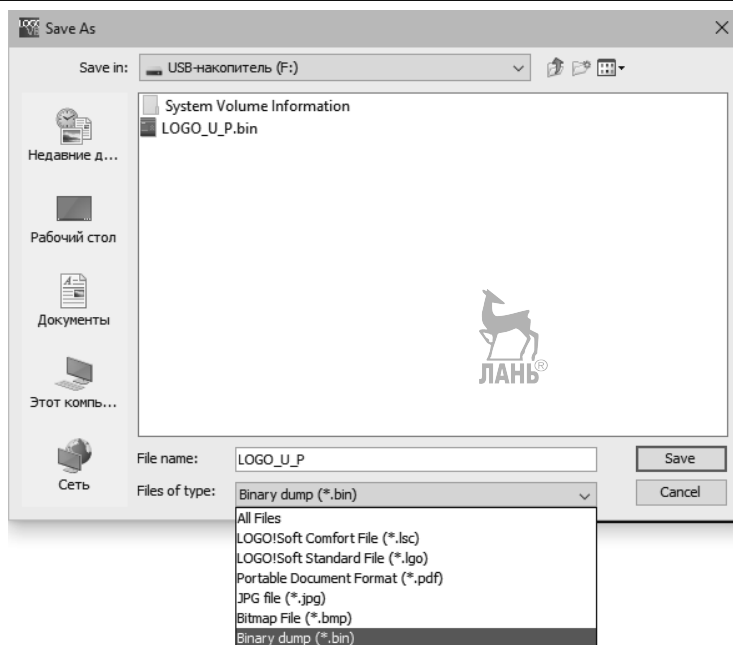


Рис. 9.35. Сохранение коммутационной программы на карту памяти

Приемный адаптер для карты памяти в модуле LOGO! показан на рис. 9.36. Вытянуть адаптер из приемного гнезда можно с помощью небольшой плоской отвертки.

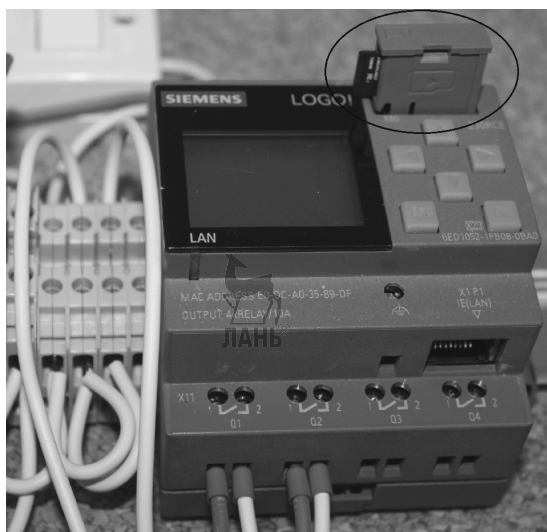


Рис. 9.36. Приемный адаптер модуля LOGO!

После установки карты памяти в приемный адаптер можно скопировать коммутационную программу в модуль LOGO! одним из следующих способов:

- а) автоматически при запуске модуля LOGO! (при включении питания);
- б) при помощи меню LOGO!

Чтобы **автоматически** скопировать коммутационную программу в модуль LOGO! нужно выполнить следующие действия:

- установить карту памяти в приемный адаптер и плотно задвинуть приемный адаптер в приемное гнездо модуля;
- включить питание модуля LOGO!

После завершения копирования на дисплее LOGO! отобразится главное меню, как показано на рис. 9.37.

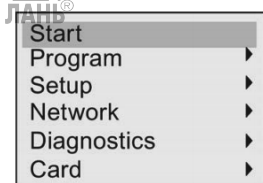


Рис. 9.37. Главное меню модуля LOGO!

Выбрав опцию Start и нажав клавишу ОК на лицевой панели модуля, можно запустить коммутационную программу на исполнение.

Чтобы скопировать программу с карты памяти в модуль LOGO! **посредством меню**, нужно выполнить последовательность действий, показанную на рис. 9.38.

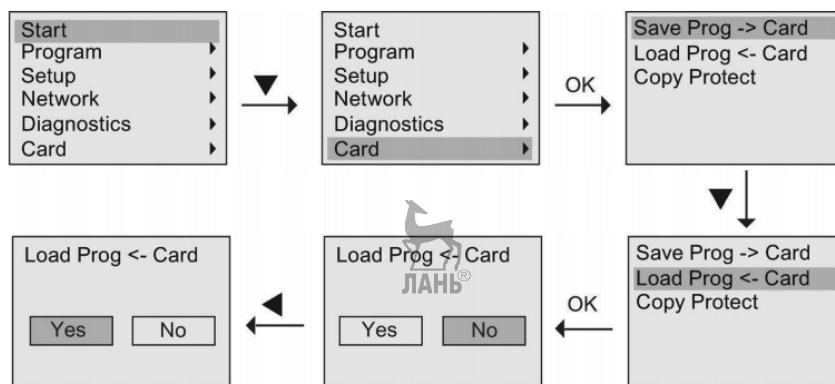


Рис. 9.38. Последовательность действий при копировании программы с карты памяти в модуль LOGO!

Загрузка программы через кабель Ethernet. Соединяем модуль LOGO! с компьютером обычным сетевым кабелем (не кросс) напрямую. Модуль LOGO! восьмого поколения предлагает команды меню для конфигурирования сетевых настроек. Для этого надо переключить меню в режим сетевых настроек, как показано на рис. 9.39.

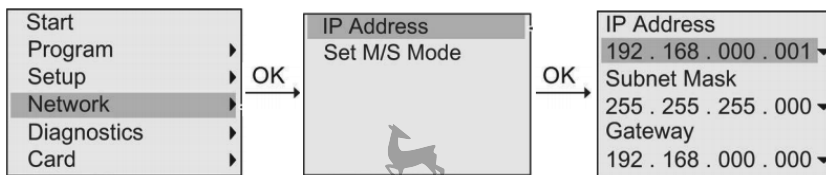


Рис. 9.39. Переход в режим сетевых настроек

На дисплее модуля отобразятся установленные по умолчанию сетевые параметры. Чтобы изменить IP адрес, надо нажать кнопку ОК. Появится курсор в виде сплошного прямоугольника. С помощью разнонаправленных кнопок на лицевой панели модуля LOGO! можно изменить IP адрес. Аналогичным образом можно изменить маску подсети и адрес шлюза. Установим на модуле LOGO! IP адрес 192.168.0.2, маску подсети оставим без изменений, шлюз установим 192.168.0.1. Перейдем в сетевые настройки компьютера и установим параметры, как показано на рис. 9.40.

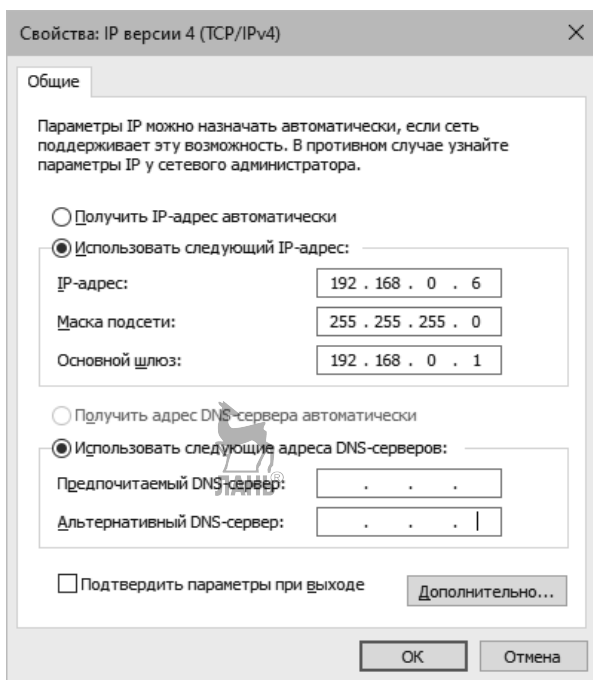


Рис. 9.40. Сетевые настройки компьютера

Предварительно переведем модуль LOGO! в режим Stop и запустим программное обеспечение LOGO! Soft Comfort. В левой панели интерфейсного окна перейдем на вкладку Network Project. Появится окно для сетевого подключения, показанное на рис. 9.41.

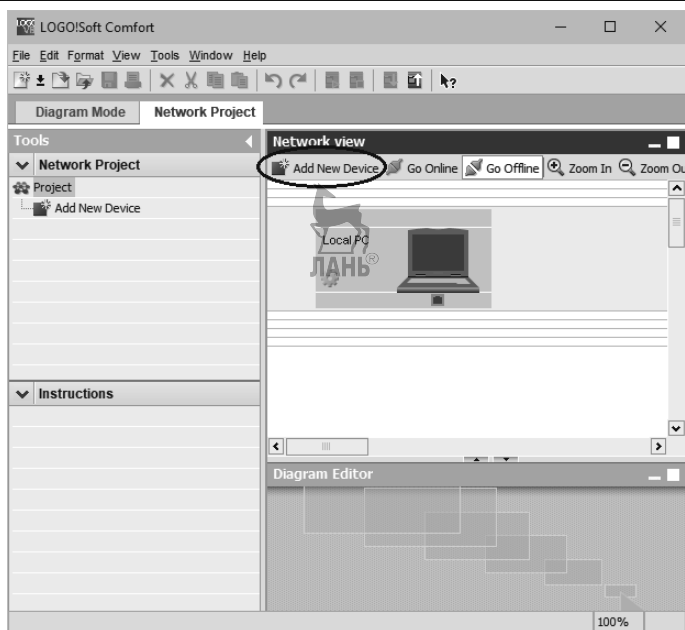


Рис. 9.41. Окно для сетевого подключения

Щелкнем в этом окне по вкладке Add New Device. Появится окно, показанное на рис. 9.42.

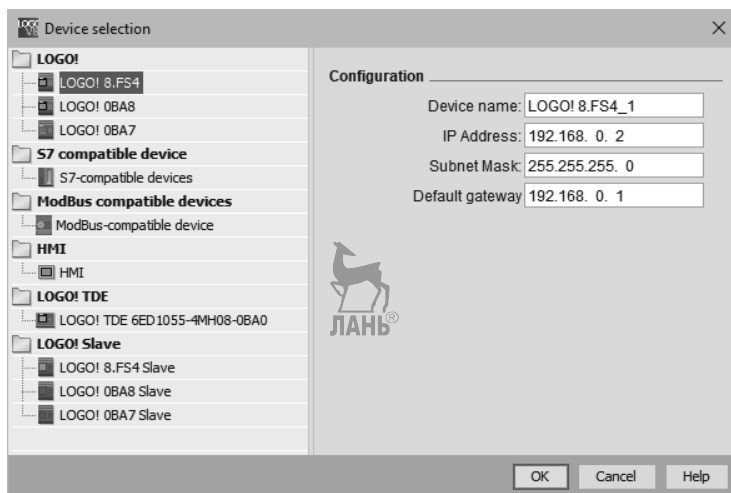


Рис. 9.42. Окно задания сетевых параметров модуля LOGO!

Установим в этом окне сетевые параметры модуля LOGO! и щелкнем ОК. Появится окно (рис. 9.43), в котором проложена связь между компьютером и модулем LOGO!. На модуле LOGO! отображается IP адрес 192.168.0.2.

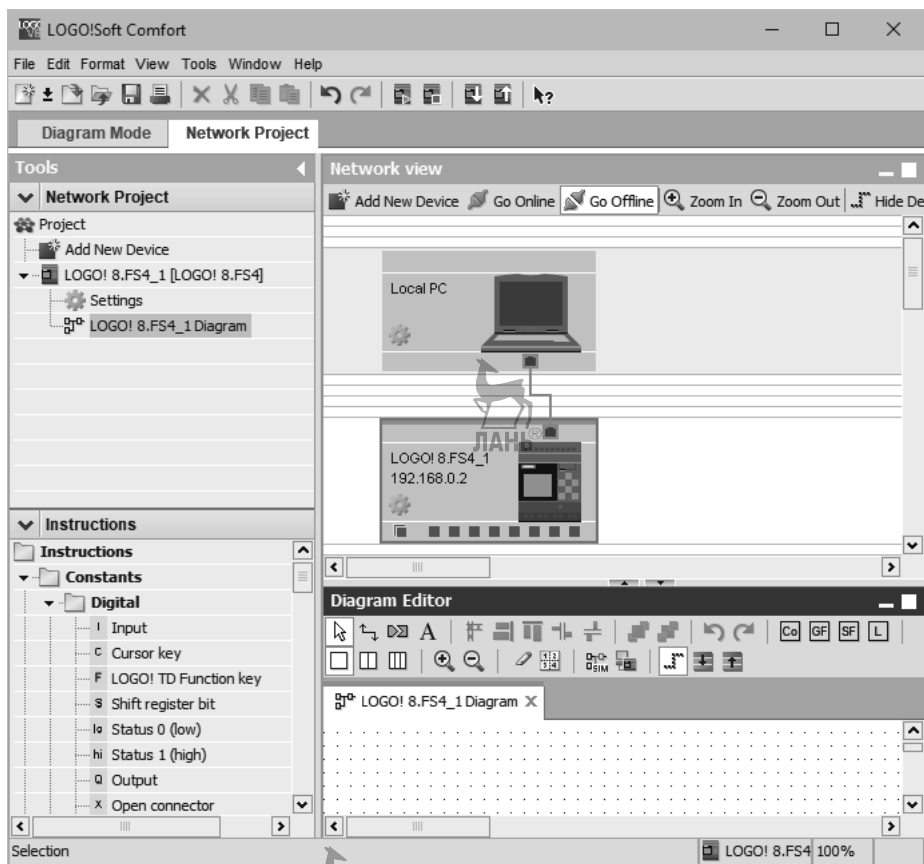


Рис. 9.43. Сетевой проект

После того, как завершена настройка сетевых адресов, можно переходить к передаче коммутационной программы из LOGO! Soft Comfort в модуль LOGO! с помощью команды PC → LOGO!. Но здесь могут возникнуть трудности. Во-первых, версии 32-bit и 64-bit компьютера и LOGO! Soft Comfort должны соответствовать друг другу. Во-вторых, в названии сетевой карты или сетевого соединения в реестре не должны присутствовать русские буквы.

Если проблем нет, то после щелчка левой кнопкой мыши по шестеренке на модуле LOGO! (нижнее изображение) появится окно, показанное на рис. 9.44.

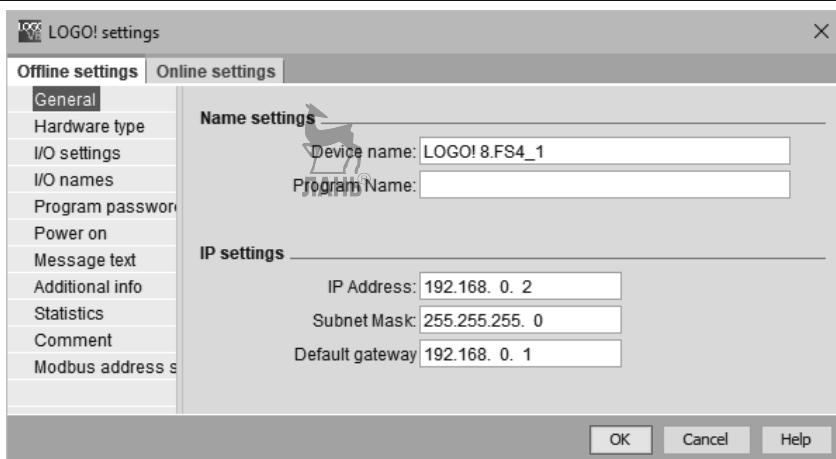
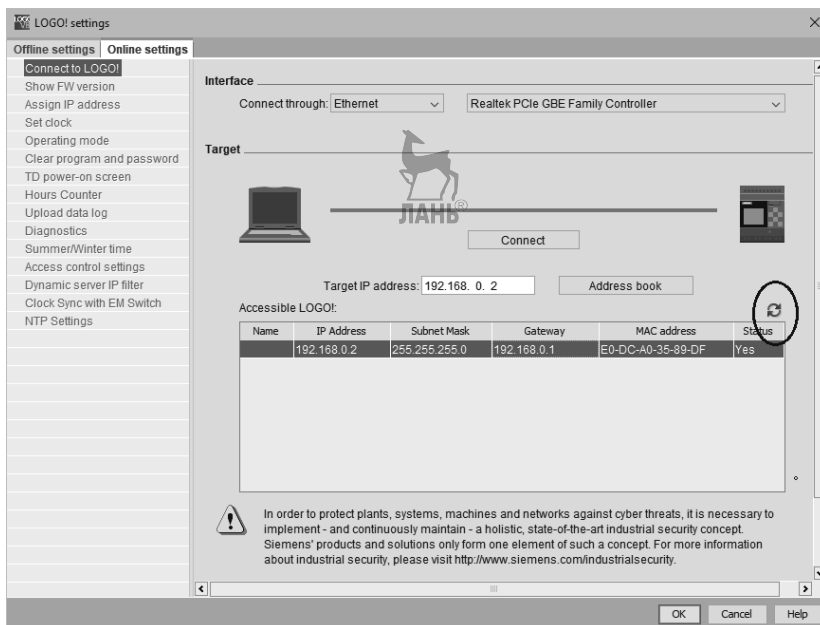


Рис. 9.44

Перейдем в этом окне на вкладку *Online settings*. Появится новое окно, показанное на рис. 9.45.

Рис. 9.45. Окно *Online settings*

В строке *Connect through* нужно установить *Ethernet* и щелкнуть по значку с двумя круговыми стрелочками (выделен овалом). В области *Accessible LOGO!* отобразится строка с сетевыми характеристиками модуля LOGO! Нужно выделить эту строку левой кнопкой мыши. При этом автоматически появится IP адрес в

строке *Target IP address*. Щелкнуть по кнопке *Connect*. Установится связь между компьютером и LOGO! Признаком установившейся связи будет галочка над линией связи между компьютером и LOGO, а сама линия станет зеленой. Закрыть окно, щелкнув по кнопке ОК и перейти на вкладку *Diagram Mode* в левой панели интерфейсного окна. Здесь надо создать коммутационную программу или загрузить уже готовую. Название коммутационной программы **не должно содержать русских букв**. Далее нажать кнопку PC → LOGO! или задать эту опцию через вкладку меню *Tools*. Появится окно, показанное на рис. 9.46. Щелкнуть в этом окне по круговому значку с двумя стрелочками, выделить строку с сетевыми параметрами LOGO! и щелкнуть по кнопке *Test*. Будет протестирована связь между компьютером и модулем LOGO! В случае положительного результата линия связи станет зеленой и над ней появится галочка (рис. 9.46). Нажать ОК.

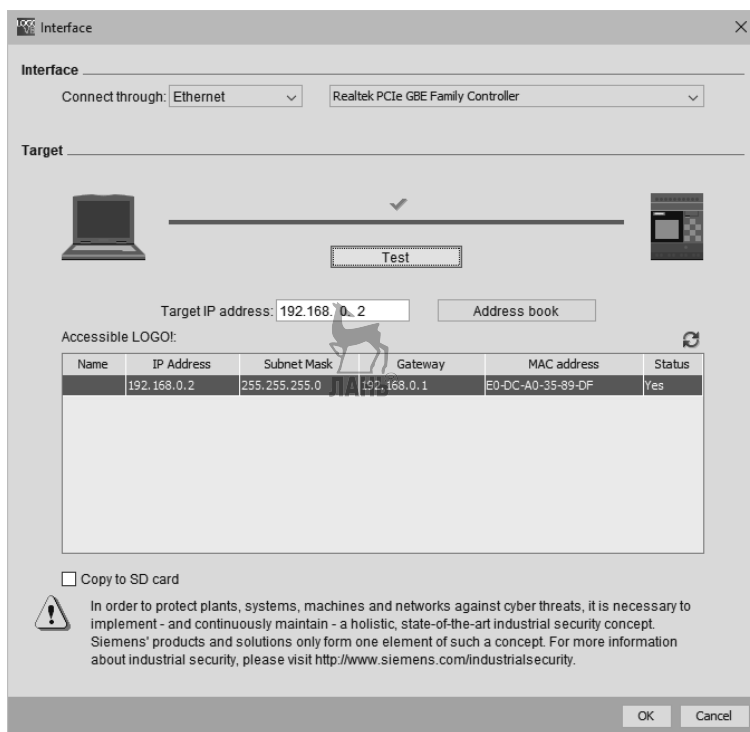


Рис. 9.46. Тестирование соединения PC – LOGO!

Программа загрузится в контроллер и появится небольшое окно (рис. 9.47) с сообщением *The device is in STOP mode. Change to RUN?* Нажать кнопку YES. Программа начнет выполняться.

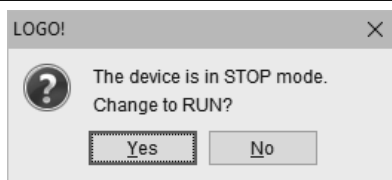


Рис. 9.47. Окно запуска программы

Приведем также описание подключения модуля **LOGO!** через *Dynamic DNS*. Этот способ можно применить, если в сетевом подключении встречаются русские буквы. Щелкнуть левой кнопкой мыши по шестеренке на модуле LOGO! (нижнее изображение). В появившемся окне в строке *Connect through* указать *Dynamic DNS*. В строке URL указать IP адрес модуля LOGO! 192.168.000.002 и щелкнуть по кнопке *Connect* (рис.9.48). Установится связь между компьютером и модулем LOGO! Линия связи поменяет цвет на зеленый. Закрыть окно, щелкнув по кнопке ОК, и перейти на вкладку *Diagram Mode*. Создать коммутационную программу или загрузить уже готовую. Далее нажать кнопку PC → LOGO! или задать эту опцию через вкладку меню *Tools*. Появится окно, аналогичное окну 9.46, с кнопкой *Test*. Нажать кнопку *Test* и затем ОК. Появится окно для введения пароля. Набрать в качестве пароля **LOGO** большими буквами. Нажать ОК. Появится сообщение *The device is in STOP mode. Change to RUN?* Нажать *YES*. Программа начнет выполняться.

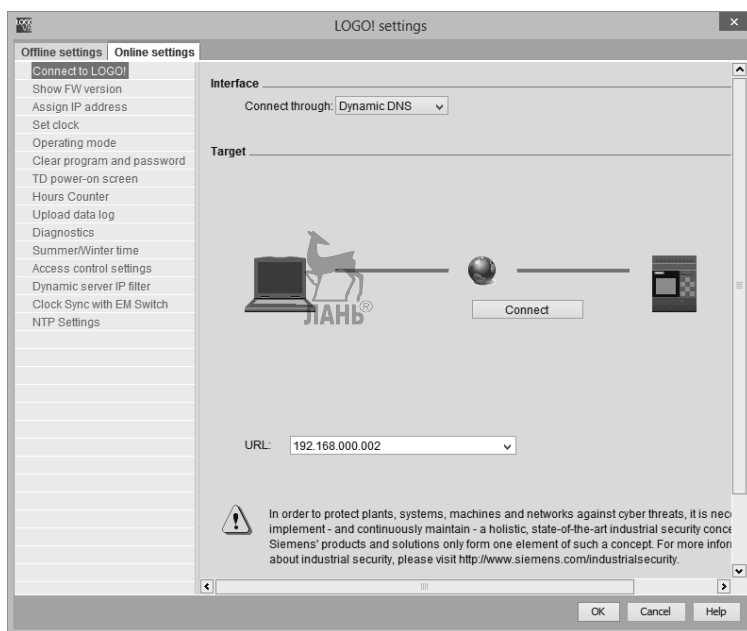


Рис. 9.48. Выбор режима DNS в окне LOGO! settings

На рис. 9.49 показан лабораторный стенд в процессе выполнения программы.

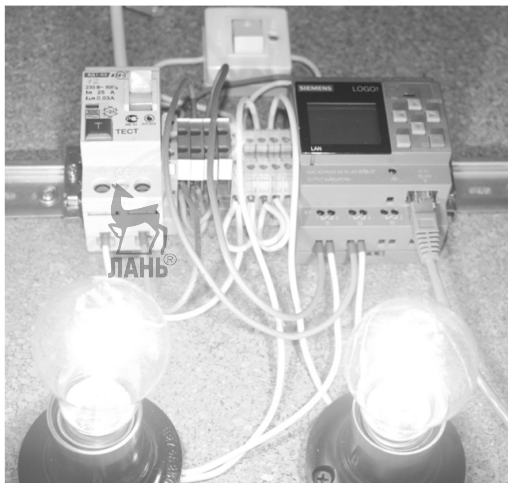


Рис. 9.49. Лабораторный стенд в процессе работы

Контрольные вопросы

1. Из каких блоков состоит коммутационная программа?
2. В чем состоит различие логики работы блоков «Асинхронный генератор импульсов» и «Интервальное реле с запуском по фронту»?
3. Для чего предназначен дифференциальный автомат?
4. Сколько и какие входы, и выходы имеет логическое реле LOGO! 230RCE?



9.2. Лабораторные работы с контроллером ONI

9.2.1. Лабораторная работа №7 «Многоканальный пожарный извещатель»

Описание лабораторной работы.

В лабораторной работе моделируется логика работы многоканального пожарного извещателя. Пожарный извещатель предназначен для выдачи тревожного сигнала о возгорании в помещении. Чем раньше будет обнаружен очаг возгорания, тем больше шансов потушить пожар, понеся при этом наименьшие материальные потери. Для более раннего обнаружения очага возгорания и для уменьшения вероятности ложных срабатываний используются многоканальные пожарные извещатели, реагирующие на различные факторы, сопровождающие процесс горения. Предположим, имеется пожарный извещатель, в состав которого входят 4 датчика: оптический датчик, реагирующий на свет от пламени пожара, СО-датчик, реагирующий на выделение угарного (СО) газа, тепловой датчик, реагирующий на повышение температуры при горении, дымовой датчик, реагирующий на наличие дыма при пожаре. Применение различных датчиков обусловлено тем, что различным очагам пожара сопутствуют различные существенные признаки горения. Например, при горении бензина, практически не выделяется дыма, но присутствует сильное световое излучение. При горении пластика (электропроводки) выделяется большое количество угарного газа. При горении нефтепродуктов выделяется большое количество дыма и т.д.

Обозначим перечисленные четыре типа датчиков логическими переменными X_1, X_2, X_3, X_4 . Эти переменные принимают значение «0», когда пожар отсутствует, и «1», когда произошло возгорание и превышен некоторый порог срабатывания датчиков. Пусть пожарный извещатель выдает сигнал «Пожар» (т.е. выдает на выходе значение «1») при срабатывании любых двух и более датчиков, т.е. когда две и более переменные принимают значение «1». Такая логика работы приведет к существенному росту вероятности обнаружения пожара, повысит универсальность и помехозащищенность пожарного извещателя. Например, при ярком солнечном освещении температура внутри помещения может существенно повышаться, но пожара нет. Этот пример характеризует помехозащищенность многоканального извещателя.

Составим таблицу истинности (табл. 9.3), которая будет иметь $2^4 = 16$ строк.

Таблица 9.3.

X_1	X_2	X_3	X_4	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1

0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

С помощью инструмента *Логический преобразователь* программы Multisim спроектируем логическую схему. Для этого запустим программу Multisim, вытащим на рабочее поле значок логического преобразователя и дважды щелкнем по нему левой кнопкой мыши. В открывшемся окошке активируем левой кнопкой мыши переменные A, B, C, D. Сформируется таблица истинности четырех переменных. Заполним крайний правый столбец таблицы в соответствии с таблицей 9.3, как показано на рис. 9.50

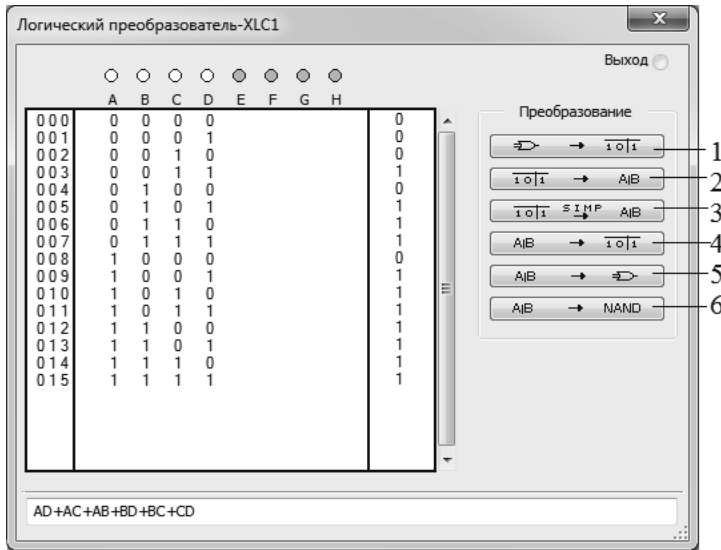


Рис. 9.50. Логический преобразователь

Далее щелкнем по клавише, обозначенной цифрой 3 на рис. 9.50. В нижней части окна (рис. 9.50) появится логическое выражение. Чтобы построить по этому выражению логическую схему в базисе И, ИЛИ, НЕ, щелкнем по клавише 5. Программа Multisim построит логическую схему, которая показана на рис. 9.51. Отметим, что для проектирования логической схемы можно также использовать *Нормальную форму «И»*, описанную в главе 2. В данном случае целесообразно

выбрать именно нормальную форму «И» (а не «ИЛИ»), поскольку нулей в таблице истинности меньше, чем единиц.

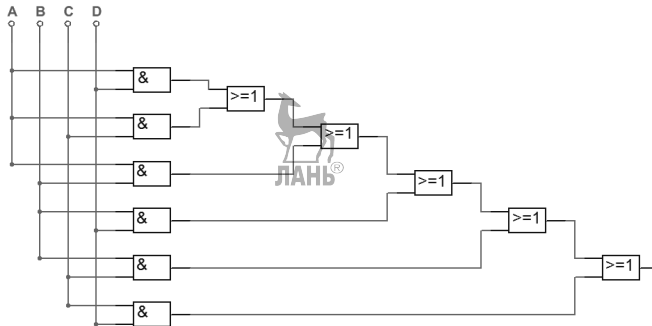


Рис. 9.51. Логическая схема 4-х канального пожарного извещателя

По логической схеме составим коммутационную программу, которая показана на рис. 9.52.

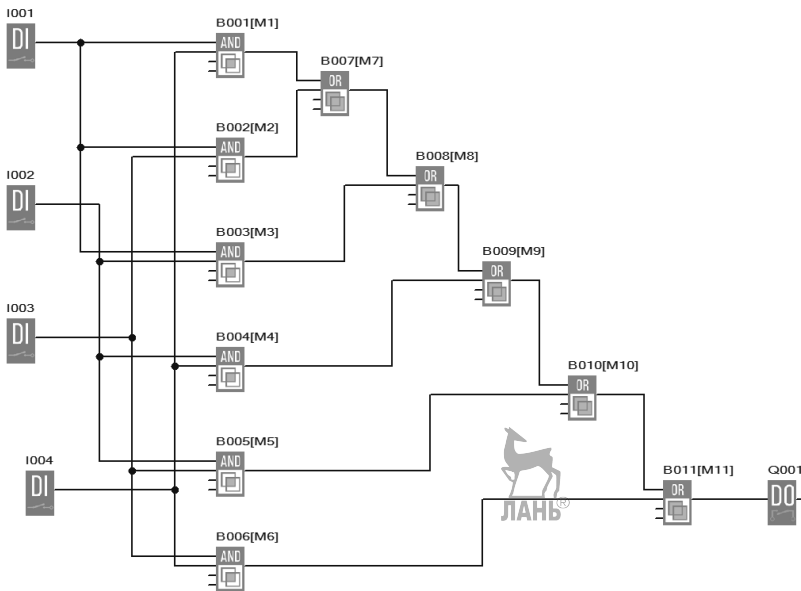


Рис. 9.52. Коммутационная программа

Лабораторная работа выполняется на стационарном стенде, представленном на рис. 9.53.



Рис. 9.53. Лабораторный стенд с программируемым реле ONI

Стенд состоит из шкафа управления, сигнальных ламп и кнопочного поста. В качестве контроллера ONI используется программируемое логическое реле ONI PLR-S-CPU-1206. Логическое реле имеет 6 цифровых входов, 6 универсальных входов и 6 цифровых выходов. Напряжение питания 12...24 В постоянного тока. Подключение логического реле к USB-разъему компьютера производится с помощью кабель-адаптера PLR-S-CABLE-USB. Чтобы загрузить программу в память логического реле, надо выбрать опции *Tools > Transfer > PC->PLR*.

Порядок выполнения работы.

- Запустить на компьютере программу ONI PLR Studio и подготовить коммутационную схему.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной схемы.
- Подать питание на стенд.
- Загрузить в контроллер коммутационную программу из компьютера.
- Запустить программу на исполнение. Включая и отключая кнопки кнопочного поста, убедиться в правильной работе системы. Сигнальная лампа должна загораться при нажатии любой комбинации двух и более кнопок.
- **По завершении эксперимента отключить питание стенда.**

Прежде, чем загружать программу в контроллер, надо установить драйвер. Установочный файл CH341SER.exe драйвера для программируемого реле ONI PLR-S-CPU-1206 можно скачать с официального сайта ONI. После запуска установочного файла появляется окно, показанное на рис. 9.54.

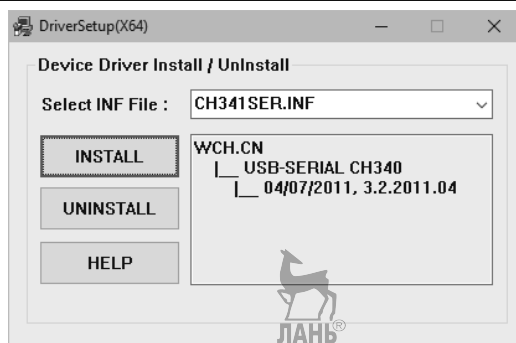


Рис. 9.54. Окно установки драйвера

Нажимаем кнопку **INSTALL**. В случае успешной установки драйвера появляется сообщение *Driver install success!* Подаем на контроллер питание. На экране контроллера появляется сообщение *Работает. Ошибок нет.* Щелкаем по вкладке меню *Инструменты* и далее щелкаем в выпадающем окне по строке *Подключение к PLR...* Появляется окно, показанное на рис. 9.55.

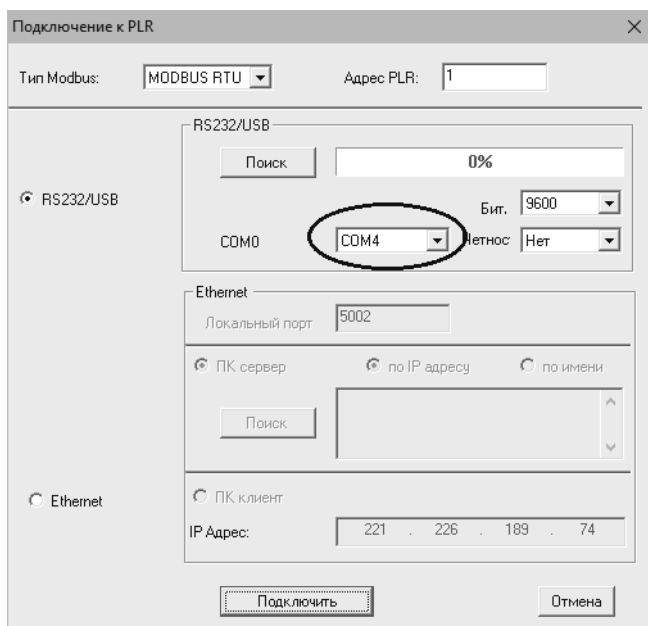


Рис. 9.55. Окно подключения к PLR

Устанавливаем COM-порт, например, COM4, как показано на рис. 9.55, и щелкаем по кнопке *Подключить*. Устанавливается связь между компьютером и контроллером. Теперь можно приступать к загрузке программы в контроллер. Во вкладке меню *Инструменты* выбираем опцию *Инструменты > Операция онлайн*

> *Загрузить в PLR.* Появляется сообщение *Остановить PLR для загрузки программы?* Нажимаем кнопку *Да.* В нижней части интерфейсного окна (рис. 9.56) можно наблюдать индикатор загрузки программы. После того, как загрузка программы составит 100%, можно переходить к работе со стендом.

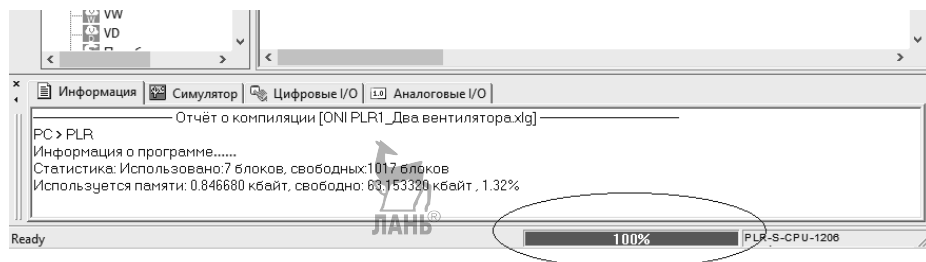


Рис. 9.56. Индикатор загрузки программы

Контрольные вопросы

1. Как называется программное обеспечение для разработки коммутационной программы логического реле ONI?
2. Привести таблицы истинности для базовых функций: AND, NAND, OR, NOR, XOR, NOT.
3. Как построить логическое выражение по известной таблице истинности, используя *Нормальную форму «И»* и *Нормальную форму «ИЛИ»*?
4. Как построить логическое выражение, используя программу Multisim?
5. Как загрузить коммутационную программу в логическое реле ONI?



9.2.2. Лабораторная работа №8 «Сигнал SOS»

Описание лабораторной работы.

В лабораторной работе программируется контроллер для выдачи сигнала SOS. Сигнал SOS состоит из трех коротких импульсов (буква S), трех длинных импульсов (буква O), трех коротких импульсов (буква S). Далее, через некоторое время молчания, цикл повторяется. Примем в лабораторной работе временную диаграмму для выдачи сигнала, как показано на рис. 9.57.

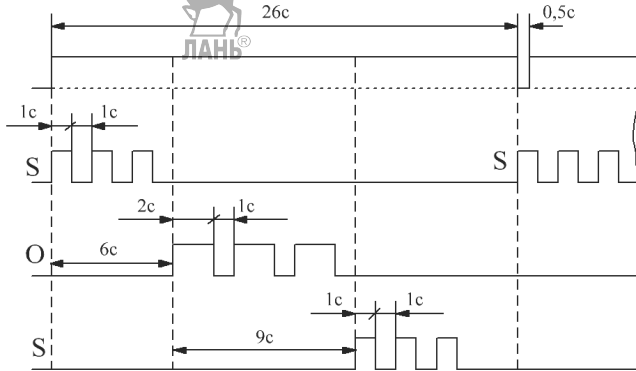


Рис. 9.57. Временная диаграмма

Коммутационная программа, составленная в соответствии с временной диаграммой, показана на рис. 9.58.

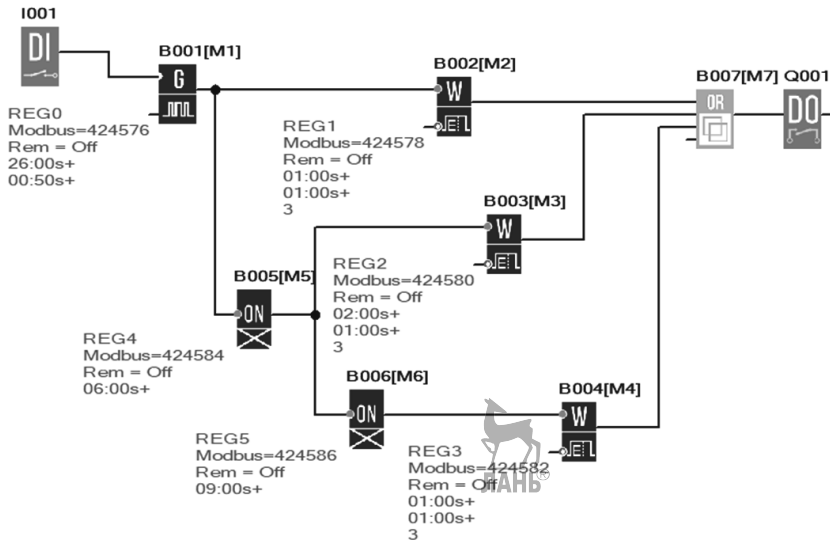


Рис. 9.58. Коммутационная программа

Коммутационная программа состоит из следующих блоков:

- I001 – цифровой вход;
- B001 – генератор импульсов;
- B002, B003, B004 – генератор серии импульсов;
- B005, B006 – задержка включения;
- B007 – логическая функция «ИЛИ»;
- Q001 – цифровой выход.

После проверки работоспособности коммутационная программа передается в программируемое логическое реле ONI PLR-S-CPU-1206. Для этого используется стенд, описанный в предыдущей лабораторной работе.

Порядок выполнения работы.

- Запустить на компьютере программу ONI PLR Studio и подготовить коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Подать питание на стенд.
- Загрузить в контроллер коммутационную программу из компьютера. (Процесс подключения контроллера к компьютеру и загрузки программы в контроллер описан в лабораторной работе №7).
- Запустить программу на исполнение. Убедиться в правильной работе программы. Сигнал SOS выдается световыми импульсами.

По завершении эксперимента отключите питание стенда.

Контрольные вопросы.

1. Какие функции используются в коммутационной программе?
2. Как называется в программе LOGO! Soft Comfort функция, аналогичная функции «Генератор серии импульсов» в программе ONI PLR Studio?
3. Какой кабель используется для связи компьютера и логического реле?
4. Сколько и какие входы и выходы имеет логическое реле ONI PLR-S-CPU-1206?
5. Какую последовательность опций надо выбрать в программном обеспечении ONI PLR Studio, чтобы загрузить программу в память логического контроллера?
6. Какое напряжение питания имеет логический контроллер ONI PLR-S-CPU-1206?



9.2.3. Лабораторная работа №9 «Управление двумя вентиляторами»

Описание лабораторной работы.

Имеются два вентилятора и две кнопки. При нажатии на кнопку 1 работают оба вентилятора. При нажатии на кнопку 2 вентиляторы работают попеременно, т.е. поток воздуха создает только один вентилятор. Попеременный режим работы в данном случае позволяет продлить срок службы вентиляторов.

Коммутационная программа показана на рис. 9.59.

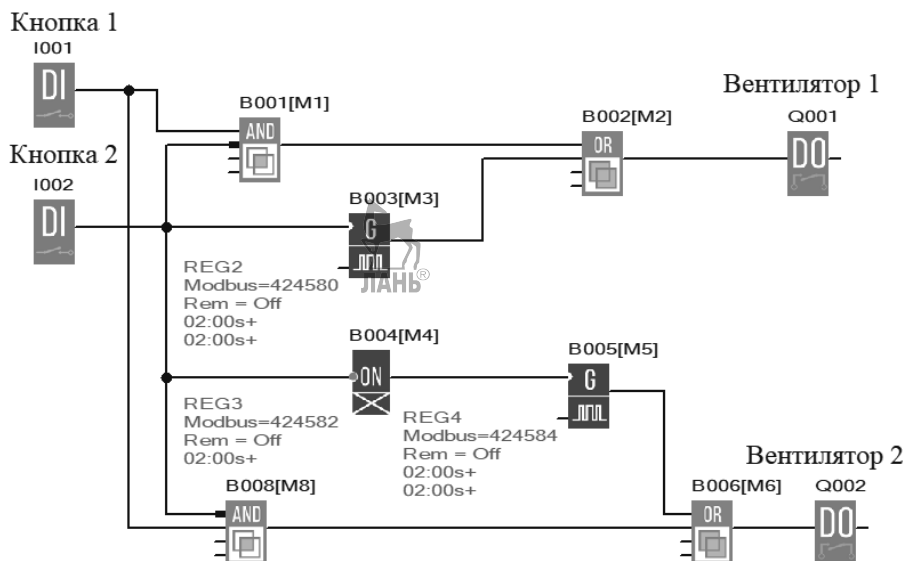


Рис. 9.59. Коммутационная программа

Коммутационная программа состоит из следующих блоков:

- I001, I002 – цифровые входы;
- B001, B008 – логическая функция «И»;
- B002, B005 – логическая функция «ИЛИ»;
- B003, B005 – генератор импульсов;
- B004 – задержка включения;
- Q001, Q002 – цифровой выход.



Для практической реализации программы соберем лабораторный стенд из отдельных компонентов. Компоненты, используемые для лабораторного стенда, показаны на рис. 9.60.

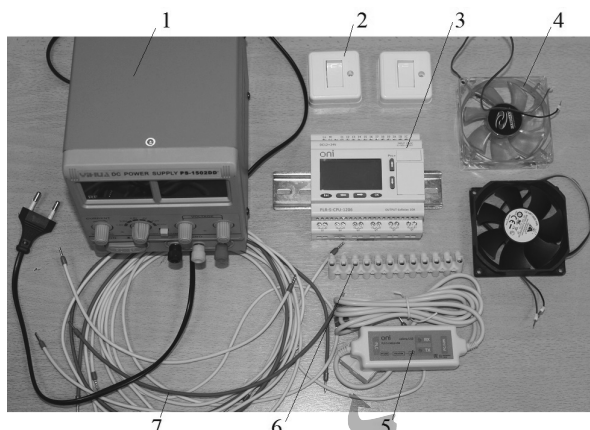


Рис. 9.60. Компоненты лабораторного стенда

Цифрами на рис 9.60 обозначены:

1. Лабораторный источник питания модели YINUA1502 DD, выходное напряжение $0 \dots 15$ В, выходной ток $0 \dots 2$ А, защита от короткого замыкания.
2. Выключатели.
3. Программируемое логическое реле ONI PLR-S-CPU-1206.
4. Мини вентиляторы, напряжение 12 В.
5. Соединительный кабель компьютер – логическое реле: ONI PLR-S-CABLE-USB.
6. Клеммная колодка.
7. Соединительные провода.

Электрическая схема лабораторного стенда представлена на рис. 9.61.

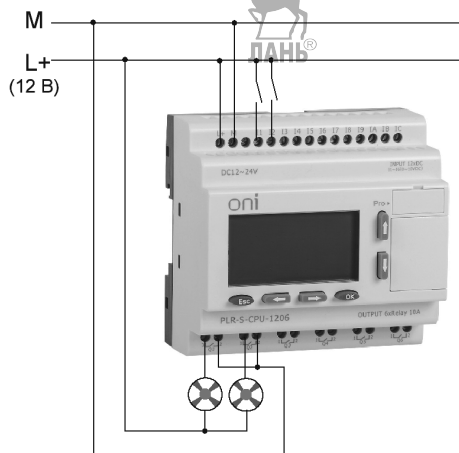


Рис. 9.61. Электрическая схема

Порядок выполнения работы.

- Запустить на компьютере программу ONI PLR Studio и подготовить коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Подать питание на стенд.
- Загрузить в контроллер коммутационную программу. (Процесс подключения контроллера к компьютеру и загрузки программы в контроллер описан в лабораторной работе №7).
- Запустить программу на исполнение. Убедиться в правильной работе лабораторного стенда в соответствии с описанным алгоритмом.
- **По завершении эксперимента отключить питание стенда.**

Лабораторный стенд в процессе выполнения программы показан на рис. 9.62.

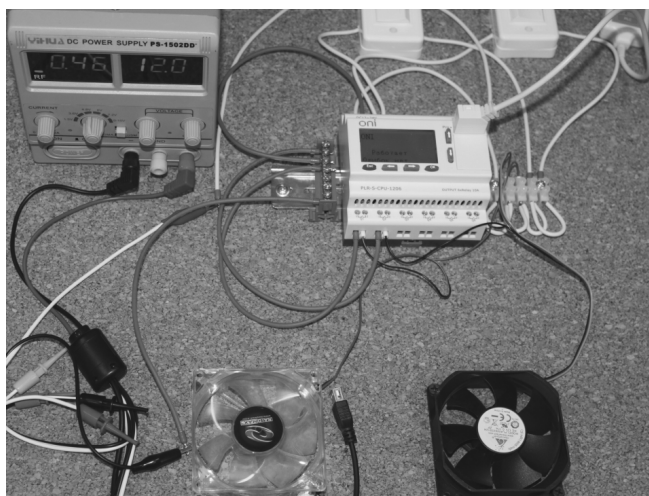


Рис. 9.62. Лабораторный стенд в процессе работы

Контрольные вопросы и задания

1. Как удалить сетку с рабочего поля программы ONI PLR Studio?
2. Какой кабель используется для связи компьютера и логического контроллера?
3. Какие выходы могут использоваться в логических реле, кроме релейных?
4. Откуда берут питание внешние компоненты, подключаемые к релейным выходам?
5. Разработать коммутационную программу, работающую по следующему алгоритму:
 - при включении переключателя I001 начинает циклически работать выход Q001,

- при включении переключателя I002 выход Q001 работает постоянно, выход Q002 не работает,
- при выключении переключателя I001 выход Q001 продолжает работать постоянно и начинает циклически работать выход Q002.

Изложенный алгоритм работы позволяет наращивать воздушный поток от вентиляторов с помощью двух переключателей.

Решение. Вариант коммутационной программы, работающей по изложенному алгоритму, представлен на рис. 9.63.

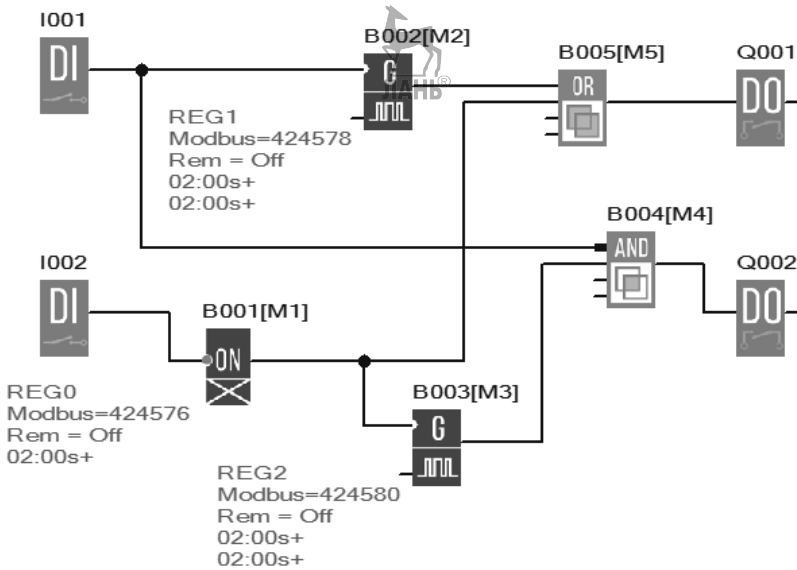


Рис. 9.63. Коммутационная программа



9.3. Лабораторные работы с контроллером ОВЕН

9.3.1. Лабораторная работа №10 «Светофор»

Описание лабораторной работы.

В лабораторной работе моделируется работа светофора. Для создания программы используется программное обеспечение OWEN Logic.

Коммутационная программа представлена на рис. 9.64. Программа переключает цвета красный–желтый–зеленый и обратно в соответствии с логикой работы обычного одностороннего светофора.

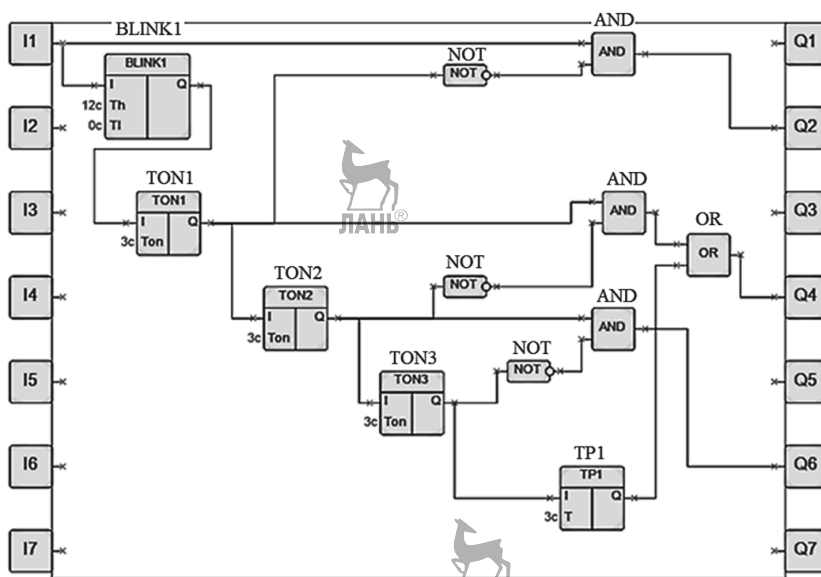


Рис. 9.64. Коммутационная программа

Коммутационная программа загружается в лабораторный стенд с помощью обычного USB-кабеля. Лабораторный стенд показан на рис. 9.65. В состав лабораторного стенда входят:

1. Устройство защитного отключения iEK АВДТ32, С16, 30 мА.
2. Двухполюсный автоматический выключатель С10.
3. Программируемое реле ПР200-220.2.1.0.
4. Индикаторная лампа МТ22-Д63.
5. Индикаторная лампа МТ22-Д64.
6. Индикаторная лампа МТ22-Д64.
7. Два выключателя. Левый выключатель служит для включения/отключения стенда. Верхний выключатель выполняет роль ключа, подключающего цифровой вход.

8. Клеммы WAGO для монтажа на DIN-рейку.
 9. Гибкий провод $S = 0,75 \text{ мм}^2$.
 10. Соединительный USB-кабель компьютер – логическое реле.
- Приборы зафиксированы на DIN-рейке.

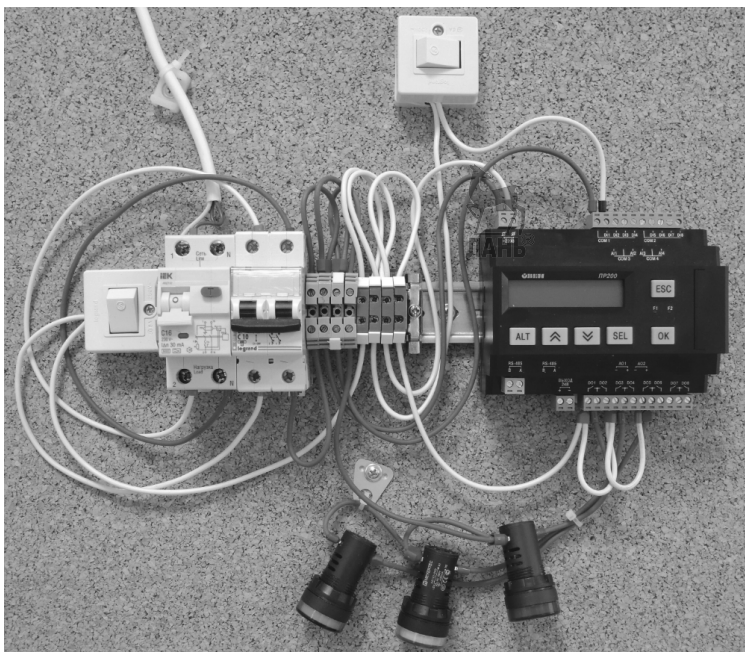


Рис. 9.65. Лабораторный стенд с программируемым реле OWEN

Схема электрических соединений приведена на рис. 9.66.

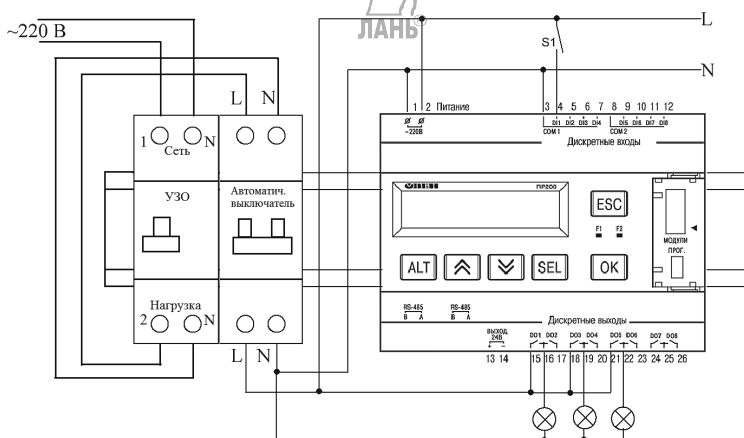


Рис. 9.66. Схема электрических соединений

Порядок выполнения работы.

- Запустить на компьютере программу OWEN Logic и подготовить коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Соединить компьютер и контроллер USB-кабелем.
- Подать питание на стенд.
- Загрузить коммутационную программу в контроллер.
- Запустить программу на исполнение. Убедиться в правильной работе лабораторного стенда в соответствии с логикой работы светофора.
- **По завершении эксперимента отключить питание стенда.**

Перед загрузкой программы в контроллер необходимо выполнить следующие действия.

- Установить на компьютер драйвер для контроллера ПР200.
- Настроить СОМ-порт. Для этого в диспетчере устройств компьютера в разделе *Порты (СОМ и LPT)* определить к какому порту подключился контроллер и установить этот порт в настройках подключения прибора. Для этого пройти по ссылкам *Прибор > Настройка порта*. Появится окно, показанное на рис. 9.67.

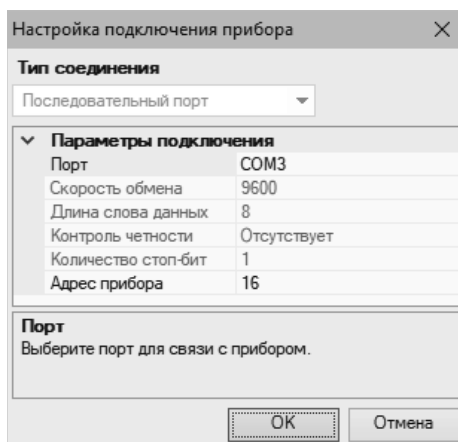


Рис. 9.67. Настройка СОМ-порта

Указать в этом окне нужный СОМ-порт, который надо выбрать, щелкнув по появившейся треугольной стрелочке в строке *Порт*.

- Настроить часы. Для этого пройти по ссылкам *Прибор > Настройка прибора > Часы*. Появится окно, показанное на рис. 9.68.



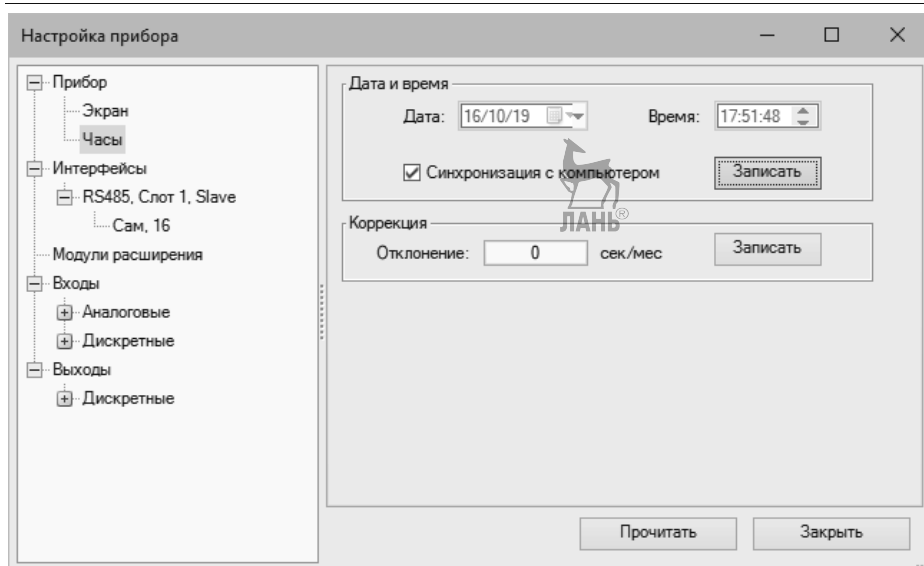


Рис. 9.68. Настройка часов

Здесь поставить галочку около опции *Синхронизация с компьютером* и щелкнуть по кнопке *Записать*.

- Чтобы загрузить программу в прибор надо выйти из режима эмуляции. Затем во вкладке меню *Прибор* щелкнуть левой кнопкой мыши по строке *Записать программу в прибор*.

Контрольные вопросы

1. Как называется программное обеспечение для составления коммутационной программы логического реле ОВЕН?
2. Как называется язык программирования?
3. Какие логические функции используются в коммутационной программе?
4. Как загрузить коммутационную программу в программируемое реле ПР200?
5. Для чего служит УЗО в лабораторном стенде?
6. Для чего служит автоматический выключатель в лабораторном стенде?
7. Как называются входы и выходы программируемого реле, задействованные в лабораторном стенде?



9.3.2. Лабораторная работа №11 «Элементы автоматики»

Описание лабораторной работы.

Целью лабораторной работы является ознакомление с элементами автоматики на примере автоматических одностворчатых гаражных ворот (рис. 9.69).

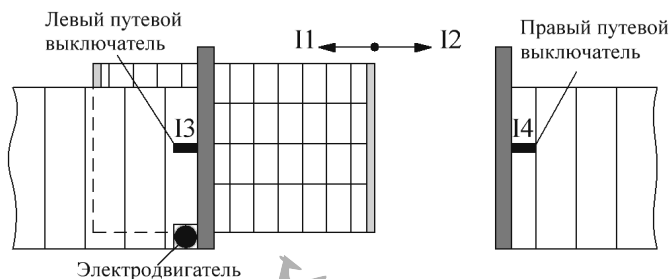


Рис. 9.69. Одностворчатые гаражные ворота

Элементы автоматики расположены на стационарном лабораторном стенде, приведенном на рис. рис.9.70.

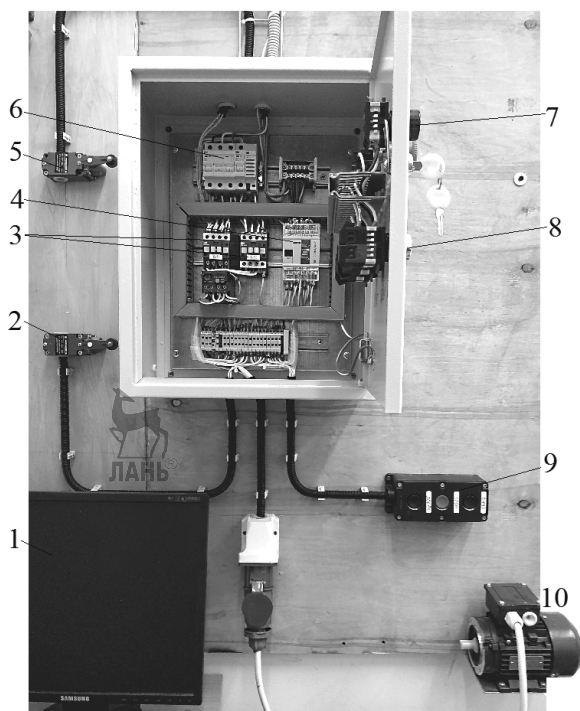


Рис. 9.70. Лабораторный стенд с программируемым реле OWEN

Состав стенда:

1. Компьютер.
2. Нижний путевой выключатель ВП-15К-21А-231-54У2.3.
3. Контактторы КМИ-10910.
4. Программируемое реле ОВЕН ПР110-220.8ДФ.4Р.
5. Верхний путевой выключатель ВП-15К-21А-231-54У2.3.
6. Блок автоматических выключателей.
7. Индикаторные лампы. ЛАНЬ®
8. Кнопки на дверце шкафа.
9. Выносной кнопочный пост.
10. Электродвигатель трехфазный асинхронный АИР 56А4, 0,12 кВт, 1350 об/мин.

Назначение элементов стенда.

Компьютер служит для разработки коммутационной программы и её отладки в режиме эмуляции.

Путевые (концевые) выключатели служат для автоматического отключения электродвигателя при полном закрывании или открывании ворот.

Контактторы являются промежуточным звеном между программируемым реле и электродвигателем. Момент запуска электродвигателя сопровождается протеканием больших токов по электрическим цепям. Если напрямую подключить электродвигатель к программируемому реле, то большие токи выведут реле из строя. Наряду с контактными для запуска электродвигателей применяются также электромагнитные пускатели. Пускатель обычно представляет собой модифицированный контактор, он может быть укомплектован дополнительными устройствами, такими как: тепловое реле для аварийного отключения двигателя; дополнительная слаботочная контактная группа, используемая в цепях управления; кнопка пуска. Иногда пускатели снабжаются устройством аварийного отключения при обрыве одной из фаз трёхфазной сети питания трёхфазных электродвигателей. ЛАНЬ®

Программируемое логическое реле. Функция реле в рассматриваемой системе автоматики – управлять направлением вращения двигателя, выдавать предупредительный звуковой сигнал, управлять режимом остановки двигателя либо от кнопки *Стоп*, либо от концевых выключателей, блокирование одновременного нажатия двух кнопок.

Электродвигатель. Вал электродвигателя может вращаться либо по часовой, либо против часовой стрелки, что соответствует либо открыванию, либо закрыванию гаражных ворот. Управление вращением вала двигателя осуществляется или выносным кнопочным пультом, или кнопками на щите управления. На пульте и на щите управления имеются по три кнопки: *Вниз*, *Стоп*, *Вверх*. Кнопки *Вверх* и *Вниз* запускают вращение двигателя или по часовой, или против часовой стрелки. Двигатель останавливается либо кнопкой *Стоп*, либо путевыми (концевыми) выключателями. Соответствие кнопок, концевых выключателей, контакторов и направления вращения, приведено в табл. 9.4.

Таблица 9.4

Кнопка	Вверх	Вниз
Направление вращения вала двигателя	По часовой стрелке	Против часовой стрелки
Контактор	КМ1	КМ2
Путевой выключатель	Верхний	Нижний

Разработка коммутационной программы.

Разработаем коммутационную программу для управления гаражными воротами. Составим таблицу истинности в соответствии с логикой работы гаражных ворот.

Введем следующие обозначения:

I1 (A) – кнопка I1. При нажатии этой кнопки ворота начинают двигаться влево (открываться).

I2 (B) – кнопка I2. При нажатии этой кнопки ворота начинают двигаться вправо (закрываются).

I3 (C) – левый путевой выключатель. При срабатывании выключателя ворота останавливаются (полностью открыты).

I4 (D) – правый путевой выключатель. При срабатывании выключателя ворота останавливаются (полностью закрыты).

I5 (E) – кнопка Стоп. При срабатывании кнопки ворота немедленно останавливаются.

Q1 – вращение двигателя против часовой стрелки (ворота открываются).

Q2 – вращение двигателя по часовой стрелке (ворота закрываются).

Составим таблицу истинности (табл. 9.5)

Таблица 9.5

I1 (A)	I2 (B)	I3 (C)	I4 (D)	I5 (E)	Q1	Q2
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	0	1	0	0
0	0	1	1	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	0	1	0	0
0	1	1	1	0	0	0

0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	0	1	0	0
1	0	0	1	0	1	0
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	0	1	0	0
1	0	1	1	0	0	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	0	0	1	0	0
1	1	0	1	0	0	0
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0

Для получения логических выражений применим программу Multisim. На рис. 9.71 показана таблица истинности для выхода Q1. Аналогичным образом в программе Multisim составляется таблица истинности для выхода Q2. Логические выражения можно видеть в нижней части таблицы логического преобразователя. Получим для Q1 и Q2 следующие логические выражения

$$Q1 = A \cdot B' \cdot C' \cdot E'$$

$$Q2 = A' \cdot B \cdot D' \cdot E'$$

Верхний штрих означает отрицание.

Добавим к полученным выражениям третий выход Q3, на который выдается предупредительный мигающий сигнал, и окончательно получим коммутационную программу, показанную на рис. 9.72.



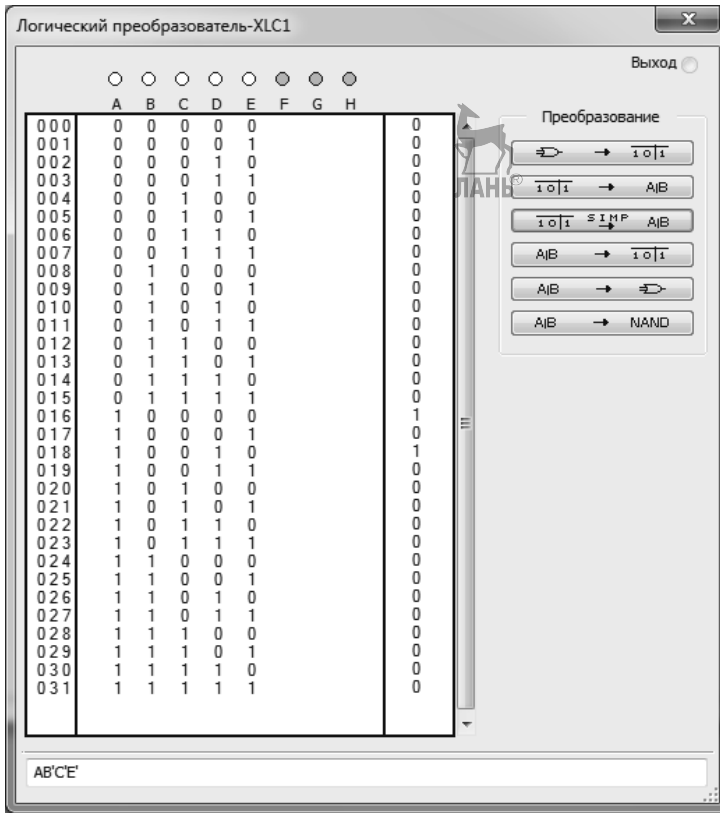


Рис. 9.71. Логический преобразователь программы Multisim

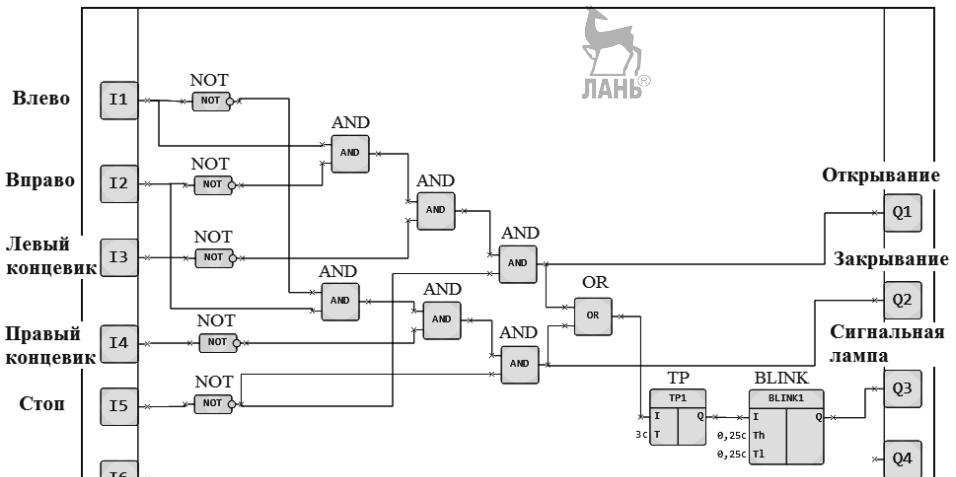


Рис. 9.72. Коммутационная программа

Порядок выполнения работы.

- Подготовить в OWEN Logic коммутационную программу.
- Загрузить коммутационную программу в логическое реле лабораторного стенда.
- Проверить работу стенда на режимах, заданных в таблице 9.7.

Таблица 9.6

Режим 1	Запуск двигателя в прямом и обратном направлениях с помощью выносного кнопочного пульта.
Режим 2	Запуск двигателя в прямом и обратном направлениях с помощью кнопок на щите управления.
Режим 3	Запуск двигателя в прямом и обратном направлениях с помощью выносного кнопочного пульта. Останов с помощью концевых выключателей.



Контрольные вопросы

1. Какое назначение имеет контактор?
2. Чем отличается электромагнитный пускатель от контактора?
3. Привести формулу для расчета количества строк таблицы истинности.
4. С помощью какого комплекта программируется логическое реле OWEN ПР110?



9.3.3. Лабораторная работа №12 «Охранная сигнализация с ИК датчиком движения»

Описание лабораторной работы.

В лабораторной работе воспроизводится система охранной сигнализации с инфракрасным датчиком движения. Коммутационная программа представлена на рис. 9.73.

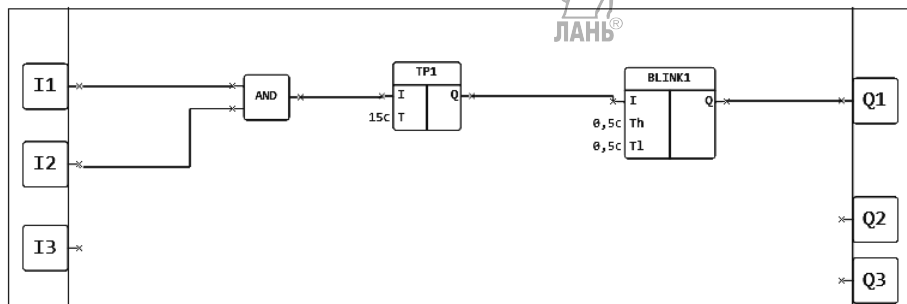


Рис. 9.73. Коммутационная программа

В коммутационной программе используются следующие блоки:

AND – Логическая функция «И».

TP – Импульс включения заданной длительности.

BLINK – Генератор импульсов.

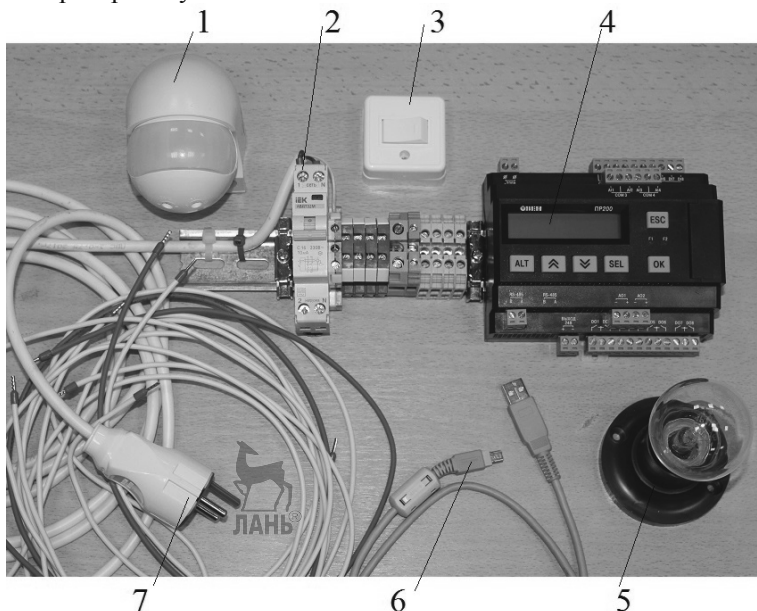


Рис. 9.74. Компоненты лабораторного макета

В процессе выполнения лабораторной работы надо собрать из отдельных компонентов макетный лабораторный стенд. Компоненты для лабораторного стенда показаны на рис. 9.74.

Цифрами на рис 9.74 обозначены:

1. Инфракрасный датчик движения ST12.
2. Дифференциальный автомат iЭК АВДТ32М.
3. Выключатель (ключ).
4. Программируемое логическое реле OWEN ПР200-220.2.1.0.
5. Лампа 220 В, 40 Вт.
6. Соединительный USB-кабель компьютер – логическое реле.
7. Соединительные провода.

Технические характеристики ИК датчика движения приведены в таблице 9.7.

Таблица 9.7

Модель	Technolight ST12
Питающее напряжение	~220...240 В / 50 Гц
Дальность обнаружения	до 12 м
Угол обзора	140...180° (регулируется)
Освещенность	3...2000 люкс (регулируется)
Время задержки срабатывания детектора	10 с / 12 мин (регулируется)
Рабочая нагрузка	до 1200 Вт
Высота установки	1,8...2,5 м
Скорость передвижения объекта обнаружения	0,6...1,5 м/с
Температурный диапазон	-20...+40°C
Степень защиты	IP44

Электрическая схема соединений лабораторного стенда представлена на рис. 9.75.

Порядок выполнения работы.

- Запустить на компьютере OWEN Logic и подготовить коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Собрать лабораторный стенд в соответствии со схемой электрических соединений.
- Подать питание на стенд.
- Загрузить в контроллер коммутационную программу. Предварительная настройка прибора OWEN описана в лабораторной работе «Светофор».
- Запустить программу на исполнение. Убедиться в правильной работе лабораторного стенда. При срабатывании датчика движения должна мигать сигнальная лампа.
- **По завершении эксперимента отключить питание стенда.**

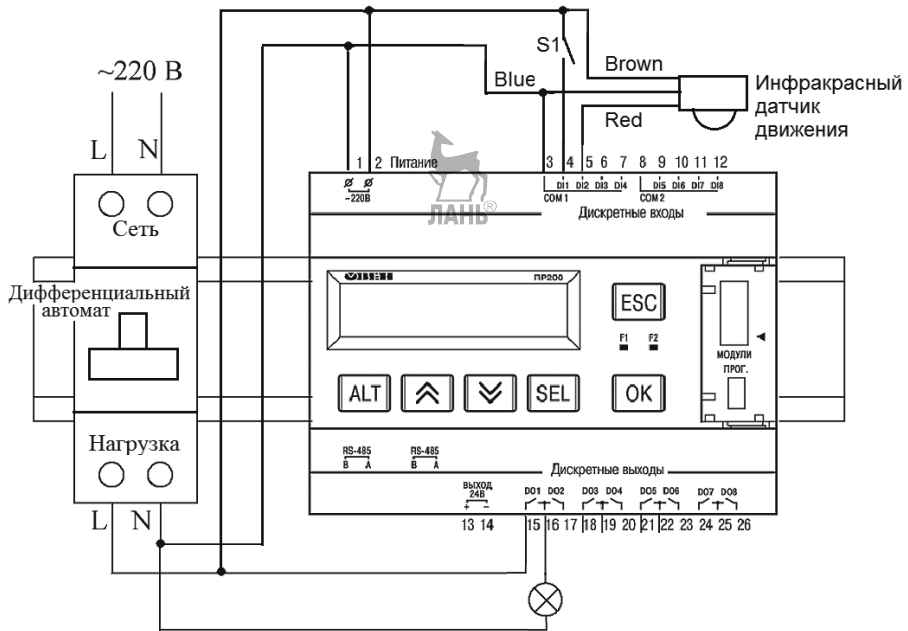


Рис. 9.75. Схема электрических соединений

Работа собранного макета с инфракрасным датчиком движения показана на рис. 9.76.

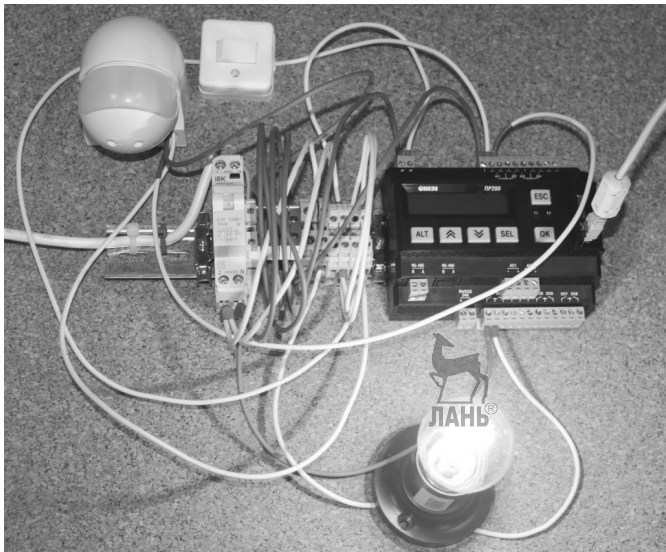


Рис. 9.76. Лабораторный стенд в процессе работы

Контрольные вопросы и задания

1. Как определить фазный и нулевой провод в электропроводке?
2. Куда подключаются Brown, Blue, Red провода инфракрасного датчика?
3. Инфракрасный датчик движения является пассивным или активным?
4. Какие объекты являются источниками инфракрасного излучения?
5. Разработать коммутационную программу охранной сигнализации с инфракрасным датчиком движения и установленным дополнительно контактным (герконовым) датчиком.



9.3.4. Лабораторная работа №13 «Охранная сигнализация с инфракрасным и микроволновым датчиками движения»



Описание лабораторной работы.

В лабораторной работе воспроизводится система охранной сигнализации с инфракрасным и микроволновым датчиками движения.

Отличительная особенность микроволнового датчика движения – это способность обнаруживать движущийся объект за неметаллическими преградами небольшой толщины (десятки миллиметров). Коммутационная программа представлена на рис. 9.77.

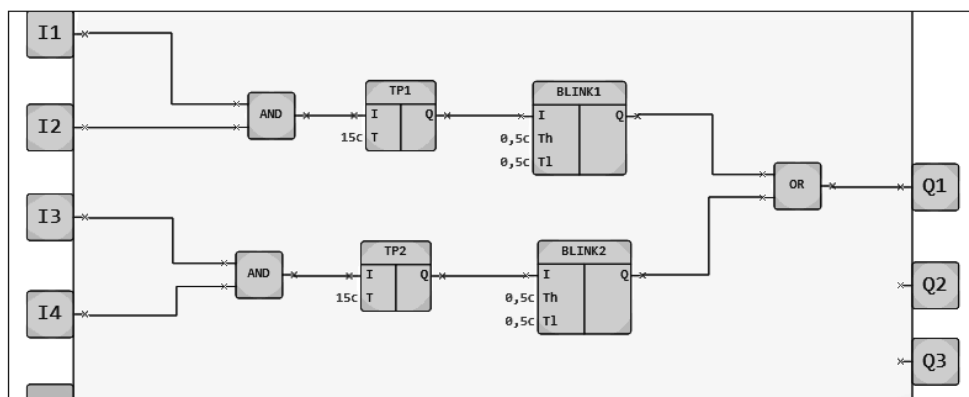


Рис. 9.77. Коммутационная программа

В коммутационной программе используются следующие блоки:

TON – Задержка включения.

AND – Логическая функция «И».

OR – логическая функция «ИЛИ».

TP – Импульс включения заданной длительности.

BLINK – Генератор импульсов.

В процессе выполнения лабораторной работы необходимо собрать действующий лабораторный стенд с инфракрасным и микроволновым датчиками движения, представленный на рис. 9.78.

Технические характеристики микроволнового датчика движения приведены в таблице 9.8.



Таблица 9.8

Модель	Feron SEN40
Рабочее напряжение	~230 В / 50 Гц
Угол обнаружения	360°
Расстояние обнаружения	1...8 м (регулируется)
Временная задержка min/max	10 с / 12 мин
Рабочая нагрузка	до 1200 Вт
Частота электромагнитных волн	5,8 ГГц
Мощность передатчика	0,2 мВт
Высота установки	1,5...3,5 м
Скорость обнаружения	0,6...1,5 м/с
Температура окружающей среды	-20...+40°C
Размеры ДхШхВ	42x81x44 мм
Степень защиты	IP20

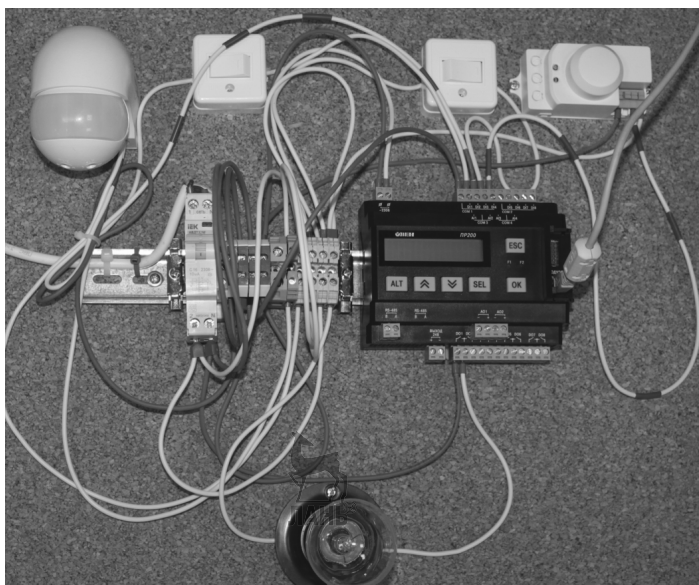


Рис. 9.78. Лабораторный стенд с инфракрасным и микроволновым датчиками

Порядок выполнения работы.

- Запустить на компьютере OWEN Logic и подготовить коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Собрать лабораторный стенд в соответствии со схемой электрических соединений.
- Подать питание на стенд.

- Загрузить в контроллер коммутационную программу. Предварительная настройка прибора описана в лабораторной работе «Светофор».
- Запустить программу на исполнение. Убедиться в правильной работе лабораторного стенда. При срабатывании датчика движения должна загораться сигнальная лампа.
- По завершении эксперимента отключить питание стенда.

Схема электрических соединений лабораторного стенда представлена на рис. 9.79.

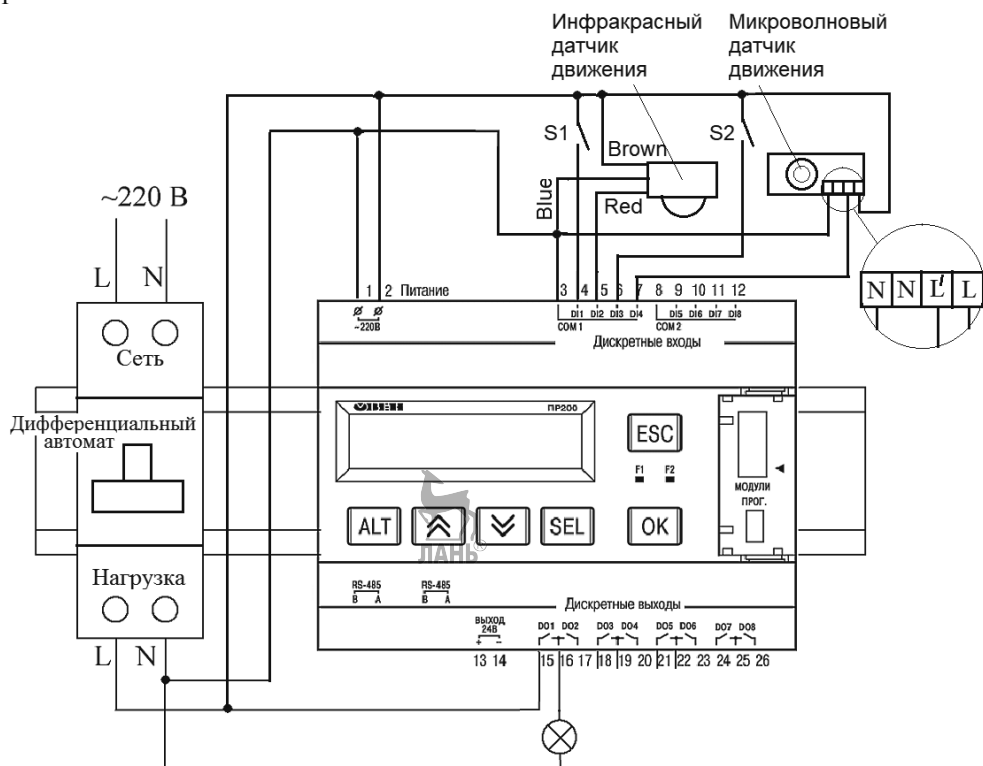


Рис. 9.79. Схема электрических соединений

Контрольные вопросы

1. Какие преимущества имеет микроволновый датчик по сравнению с инфракрасным датчиком?
2. Микроволновый датчик является пассивным или активным?

3. Укажите предпочтительные ситуации применения для инфракрасных и микроволновых датчиков движения.
4. На какой частоте работает микроволновый датчик?
5. Какой датчик, инфракрасный или микроволновый, легче обнаружить с помощью приемников излучения?
6. Какой датчик, инфракрасный или микроволновый, лучше защищён от влаги и пыли (для датчиков, использованных в лабораторных работах)? (Пояснение. Степень защиты датчиков определяется значением показателя IP).



9.3.5. Лабораторная работа №14 «Подключение к контроллеру силовой нагрузки»



Описание лабораторной работы.

В лабораторной работе моделируется подключение силовой нагрузки к контроллеру. Работа силовой нагрузки сопровождается большими токами в электрической цепи, поэтому силовая нагрузка подключается через контактор. Коммутационная программа представлена на рис. 9.80.

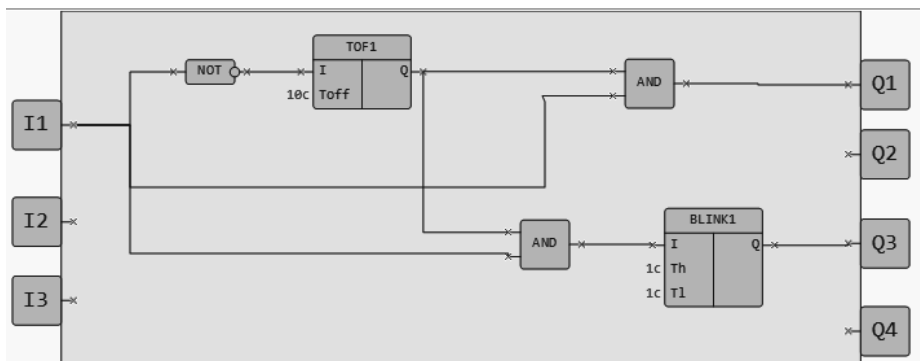


Рис. 9.80. Коммутационная программа

В коммутационной программе используются следующие блоки:

TOF – Задержка выключения.

BLINK – Генератор импульсов.

AND – «И».

NOT – «НЕ».

Для практической реализации программы необходимо собрать из отдельных компонентов действующий лабораторный стенд. Компоненты, необходимые для лабораторного стенда, показаны на рис. 9.81.

Цифрами на рис 9.81 обозначены:

1. Электродвигатель постоянного тока P2XR520, 12 В, 300 Вт, 2500 об/мин.
2. Выключатель (ключ).
3. Контактор модульный iEK KM25-40.
4. Сигнальная лампа 220 В, 40 Вт.
5. Соединительный кабель компьютер – логическое реле USB-cable.
6. Программируемое логическое реле OWEN PP200-220.2.1.0.
7. Источник питания 12 В (аккумуляторная батарея).
8. Дифференциальный автомат iEK АДТ32М.
9. Соединительные провода.

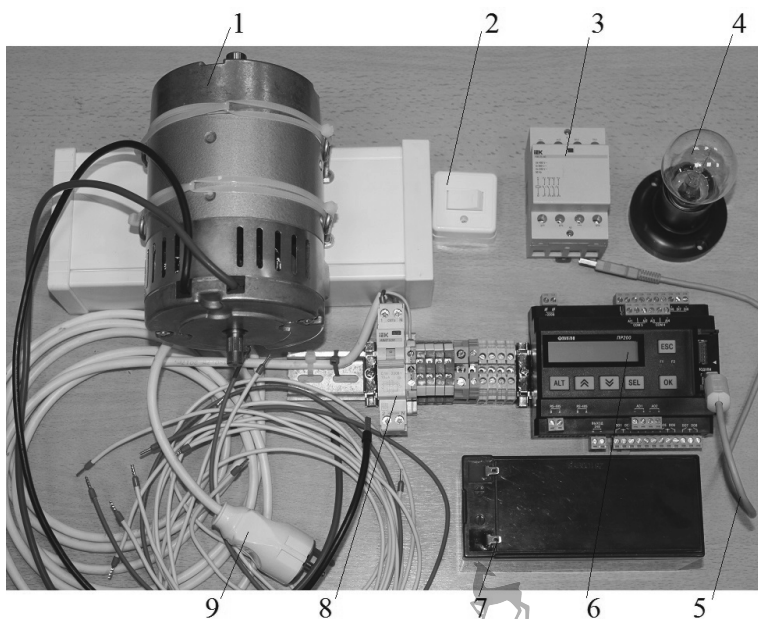


Рис. 9.81. Компоненты лабораторного стенда

Электрическая схема лабораторного стенда представлена на рис. 9.82.



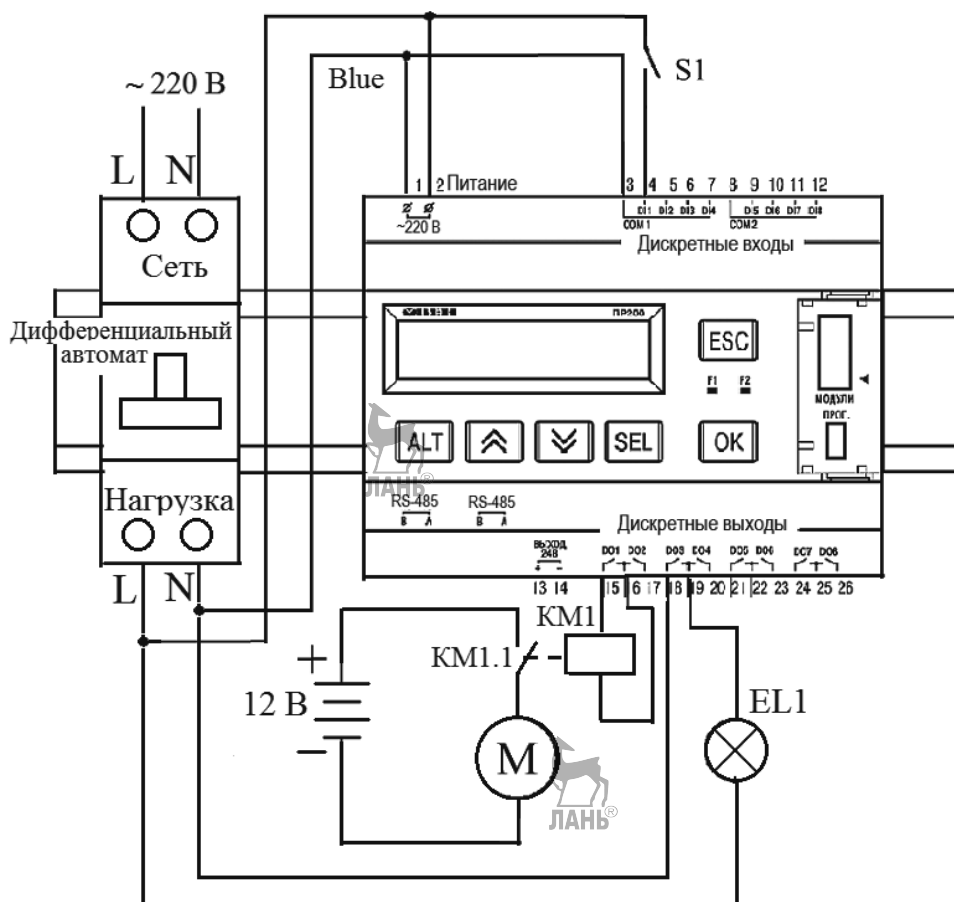


Рис. 9.82. Электрическая схема

Порядок выполнения работы.

- Запустить на компьютере OWEN Logic и собрать коммутационную программу.
- Запустить режим «Эмуляция» и проверить правильность работы коммутационной программы.
- Собрать лабораторный стенд в соответствии со схемой электрических соединений.
- Подать питание на стенд.
- Загрузить в контроллер коммутационную программу.
- Запустить программу на исполнение. Убедиться в правильной работе лабораторного стенда.
- По завершении эксперимента отключить питание стенда.

Контрольные вопросы и задания

1. Почему силовую нагрузку нельзя подключать непосредственно к контроллеру?
2. Что означает термин «гальваническая развязка» электрических цепей?
3. Каким образом электрическая цепь силовой нагрузки изолируется от электрической цепи контроллера?
4. Какое назначение имеет контактор в выходной цепи логического контроллера?
5. Функции каких двух защитных устройств совмещает в себе дифференциальный автомат?
6. Что произойдет, если сигнальную лампу EL1 подключить к выходу Q5 контроллера?
7. Разработать коммутационную схему с двумя ключами S1 и S2. При включении ключа S1 вал электродвигателя вращается по часовой стрелке, при включении ключа S2 вал электродвигателя вращается против часовой стрелки, при одновременном включении ключей S1 и S2 питание на электродвигатель не подается.



Литература



1. Иванов В.Н. Применение компьютерных технологий при проектировании электрических схем. – М.: СОЛОН-Пресс, 2017.
2. Карташов Б.А., Шабаетв Е.А., Козлов О.С., Щекатуров А.М. Среда динамического моделирования технических систем SimInTech: Практикум по моделированию систем автоматического регулирования. – М.: ДМК Пресс, 2017.
3. Красногорцев И.Л., Сенигов П.Н. Автоматика на основе программируемого контроллера. Руководство по выполнению базовых экспериментов. – Челябинск: ИПЦ «Учебная техника», 2007.
4. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. – М.: СОЛОН-Пресс, 2010.
5. Шишмарев В.Ю. Автоматика. – М.: Академия, 2019.

Интернет – источники.

6. Гайнутдинов К.Р. Простое и понятное программирование в CoDeSys (3 части), 2012.
7. LOGO! Руководство 06/2014, A5E33039675 (руководство применимо к устройствам серий LOGO! 0BA8).
8. www.siemens.ru/logo/ – Модули LOGO!
9. https://cache.industry.siemens.com/dl/dl-media/659/109750659/att_931307/v1/logo/basic/menu.html?mode=standalone/ – Базовые знания LOGO!
10. https://w3.siemens.ru/about_us/businesses/dfpd/products/ – Продукты и системы компании Siemens.
11. <https://w3.siemens.com/mcms/programmable-logic-controller/en/logic-module-logo/demo-software/pages/default.aspx> – Программное обеспечение LOGO!Soft Comfort.
12. www.oni-system.com/ – Официальный сайт компании IEK Group.
13. www.oni-system.com/ – Оборудование ТМ ONI: программное обеспечение ONI PLR Studio.
14. www.iek.ru/ – Программируемые логические реле ONI PLR-S. Системное руководство.
15. www.owen.ru/ – Оборудование для автоматизации. Официальный сайт компании ОВЕН.
16. https://www.youtube.com/playlist?list=PL2EcVEe6E9SCZ_pruKV980b7CfmZTe hYe – Видеокурс по программированию логических реле ОВЕН в Owen Logic.
17. <http://elektrik.info/ebooks/1065-videokurs-po-rabote-s-oven-plk110-v-srede-codesys.html> – Видеокурс по программированию контроллеров ОВЕН в Codesys.



Виктор Никитович Иванов

Программирование логических контроллеров

Ответственный за выпуск: **В. Митин**
Верстка и обложка: **СОЛОН-Пресс**

По вопросам приобретения обращаться:
ООО «СОЛОН-Пресс»
123001, г. Москва, а/я 82
Телефоны: (495) 617-39-64, (495) 617-39-65
E-mail: kniga@solon-press.ru, www.solon-press.ru

Оптовые закупки
ООО КТК «Галактика»
115487, г. Москва, проспект Андропова, д. 38
Телефоны: (499) 782-38-89
E-mail: books@alians-kniga.ru, <http://www.alians-kniga.ru>
Завод 1 — 100

По вопросам подписки на журнал «Ремонт & Сервис» обращаться:

ООО «СОЛОН-Пресс»
тел.: (495) 617-39-64,
www.remserv.ru

ООО «СОЛОН-Пресс»
115487, г. Москва,
пр-кт Андропова, дом 38, помещение № 8, комната № 2.
Формат 70×100/16. Объем 22,25 п. л. Тираж 1000 экз.