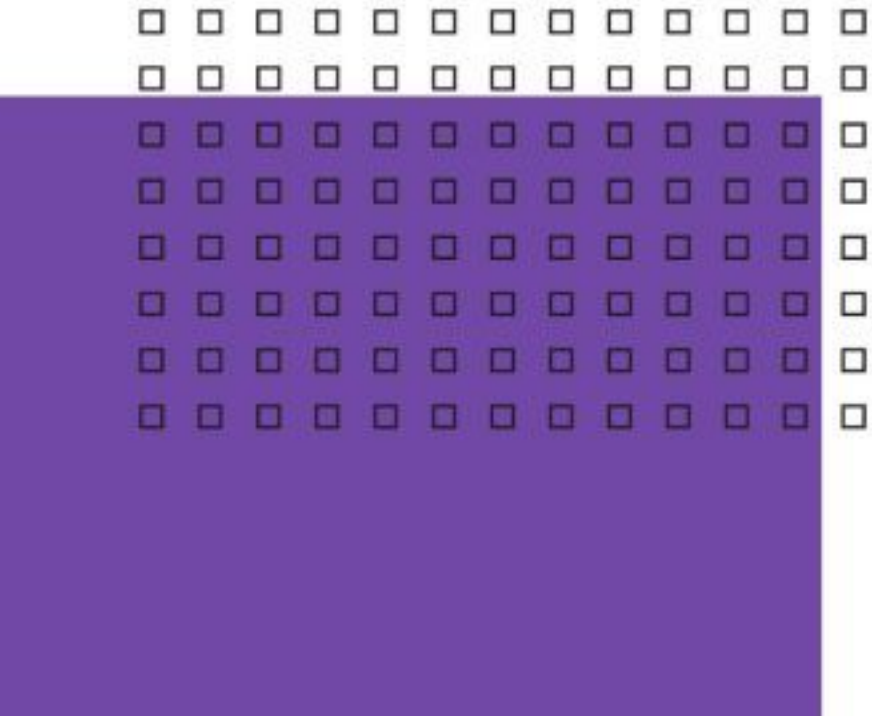
Е. А. Чернов

ПРОГРАММИРУЕМ PLC



«Инфра-Инженерия»

Е. А. Чернов

ПРОГРАММИРУЕМ PLC

Учебное пособие

Москва Вологда
«Инфра-Инженерия»
2023

УДК 621.9.022
ББК 34.63-5
Ч-49

Рецензенты:
доктор технических наук, профессор *А. С. Серебряков*;
доктор технических наук, профессор *В. Г. Титов*

Чернов, Е. А.

Ч-49 Программируем PLC : учебное пособие / Е. А. Чернов. – Москва ; Вологда : Инфра-Инженерия, 2023. – 516 с. : ил., табл.
ISBN 978-5-9729-1474-6

Рассмотрен основной комплекс задач, решаемых инженером-электриком при разработке и модернизации электрооборудования промышленного оборудования, выполненного на базе программируемых логических контроллеров (PLC). В качестве основной базы рассматриваются металлорежущие станки, как универсальные, так и с числовым программным управлением. Материал книги излагается в последовательности реального проектирования электрооборудования и алгоритмов PLC, выполняемого в конструкторских бюро: изучение объекта автоматизации, выбор устройств управления, подключение и общая блок-схема, синтаксис языка программирования, параметры, типовые решения, примеры реальных проектов. Дана инженерная методика проектирования алгоритмов работы дискретной электроавтоматики, универсальная для любой элементной базы и любых типов PLC.

Для студентов высших учебных заведений электроэнергетических направлений подготовки.

УДК 621.9.022
ББК 34.63-5

ISBN 978-5-9729-1474-6

ПРЕДИСЛОВИЕ

Немного об авторе.

Чернов Евгений Александрович, после окончания в 1958 году школы прошел творческий путь от электромонтажника, наладчика, рядового конструктора до заместителя главного конструктора Горьковского завода фрезерных станков (ГЗФС).

Работая электромонтажником, а после окончания института, наладчиком электрооборудования станков в производстве, затем в конструкторском бюро мне посчастливилось участвовать в постановке на серийное производство многих моделей универсальных консольно-фрезерных и продольно-фрезерных станков, в разработке большого числа специальных фрезерных станков, в становлении серийного производства в стране систем и станков с числовым программным управлением (ЧПУ), начиная практически с нуля осваивать различные системы управления, регулируемые электроприводы, различную элементную базу. Этот производственный опыт, а также преподавание в Горьковском (Нижегородском) политехническом институте, отраслевом институте повышения квалификации и предопределило написание серии инженерных книг, сначала по промышленным комплектным электроприводам [44–48], а затем по дискретной станочной электроавтоматике [61–64].

Настоящая книга, возможно завершающая, посвящена электроавтоматике программируемых логических контроллеров. Толчком к ее написанию явился просмотр книги о PLC Жиля Мишеля [14] и замечательная фраза в предисловии Б. Жирара: «*Речь идет о полной переработке книги, что следует приветствовать, ибо немногие авторы решаются пойти на это. Откровенно говоря, программируемые контроллеры заслужили появления новой книги…*». Как это актуально в наши дни.

Отечественная станкостроительная и инструментальная промышленность уничтожена. Страна живет за счет огромного задела металлорежущего оборудования, сделанного Советским Союзом [1, 2]. Часть оборудования покупается за рубежом, но в основном идет модернизация старых станков. Это не может продолжаться вечно, станочные структуры заканчиваются, поставки из-за рубежа могут быть сильно ограничены.

Итак, представляю новую инженерную книгу по проектированию электроавтоматики локальных промышленных объектов на базе программируемых логических контроллеров. Очень надеюсь, что она поможет восстановить подготовку специалистов по автоматизации, и в конечном итоге, своего полноценного производства металлообрабатывающего и другого машиностроительного производства.

Цель книги: научить, показать, из каких узлов состоит электрооборудование универсальных металлорежущих станков и как разрабатывается электроавтоматика их управления. Главное, чему должен научиться инженер, это систематизации и общему подходу к решению поставленной задачи. Изучить, понять и запомнить информацию по всем существующим электроприводам, программируемым контроллерам, языковым средствам невозможно. Нужно знать последовательность решения задачи, какую минимальную информацию следует выбрать из огромного объема сопроводительной, часто непонятной, технической документации и как ей воспользоваться. После чего разработать типовое решение и всегда им пользоваться.

Практический материал и типы PLC, приведенные в книге, определены последними разработками автора. Автор счел также необходимым привести начальные сведения по устройствам цифровой индикации, панелям оператора и электроприводам, практически всегда применяющимся совместно с контроллерами. Кроме того, глава 2 посвящена методологии проектирования алгоритмов управления дискретной электроавтоматики, являющихся основой при программировании контроллеров. Знания языков программирования бесполезны без умения проектировать алгоритмы управления.

Желаю читателям успехов в творческой работе при разработке и эксплуатации электроавтоматики на базе PLC.

ГЛАВА 1. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ, НАЧАЛЬНЫЕ СВЕДЕНИЯ

1.1. Общие сведения, классификация PLC

Программируемый логический контроллер (ПЛК, в зарубежной литературе PLC – Programmer Logical Controller) – это специализированное вычислительное устройство, выполненное на базе микропроцессорной техники и предназначенное для реализации функций электроавтоматики промышленных механизмов. Впервые программируемые контроллеры появились на рубеже 60–70-х годов прошлого века, что стало возможным благодаря появлению соответствующей элементной базы и разработок в области математического обеспечения. Уже в это время сложность и степень автоматизации промышленного оборудования достигла такого высокого уровня, при котором релейно-контактная элементная база не могла обеспечить необходимую надежность работы оборудования. Благодаря неоспоримым преимуществам по сравнению с жесткими релейно-контактными и бесконтактными схемами: высокой надежностью, гибкостью в перестройке на другие условия работы, доступностью программирования и др. ПЛК нашел широкое применение в практике автоматизированного производства и в станкостроении. Здесь же добавим, что все современные системы числового программного управления имеют встроенный программируемый контроллер.

При использовании ПЛК в качестве элементной базы электроавтоматики он может рассматриваться как «черный ящик» с клеммами для присоединения информационных каналов от входных и выходных сигналов (рис. 1.1).

Принципиальная электрическая схема с объектом управления в этом случае представляет собой схему стыковки с ПЛК командных, контролирующих и исполнительных элементов электрооборудования. Необходимая функциональная зависимость между ними обеспечивается за счет программы, вводимой в память ПЛК.

Следует сказать, что современный программируемый логический контроллер – это больше, чем просто средство для реализации функции дискретной электроавтоматики. Современные контроллеры характеризуются большим разнообразием архитектуры и, как следствие, языкового уровня программирования электроавтоматики. На них можно реализовать, например, следующие функции:

- прямое аналоговое или цифровое управление регулируемыми электроприводами;
- позиционное управление приводами, как с обратной связью по положению, так и без нее;

– решать задачи интерполяции;
– выполнять различные арифметические и тригонометрические действия, как с целыми числами, так и с числами с плавающей запятой;
– работать с матричными операндами;
– реализовывать ПИД-регулирование;
– создавать сетевые структуры и многое другое.

Конструктивно ПЛК могут иметь стандартное блочное исполнение, модульное исполнение или компактное исполнение. Отдельное направление, это контроллеры, *встроенные в системы ЧПУ* (см. гл. 9). Они дополнительно решают много специальных задач [62, 63].

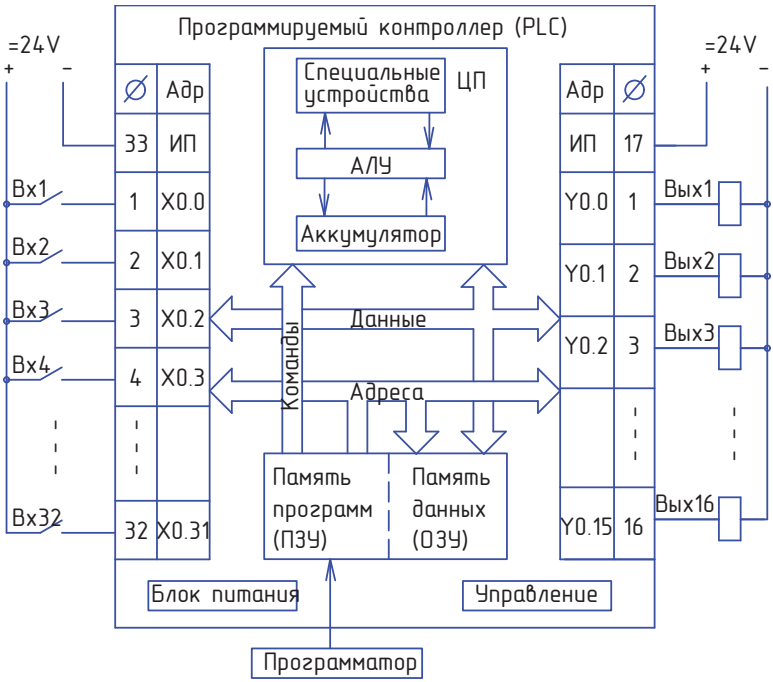


Рис. 1.1. Вариант структурной схемы программируемого логического контроллера

В состав ПЛК могут входить следующие блоки:
– блок питания;
– модуль центрального процессора;
– модули дискретных входов и выходов;
– модули аналоговых входов и выходов;
– блоки позиционирования;

– модули быстрых счетчиков;
– сетевые модули и др.

Источник питания формирует стабилизированные напряжения, необходимые для работы элементной базы, на которой реализован программируемый контроллер.

Центральный процессор (ЦП) является главным звеном контроллера, он управляет всеми блоками ПЛК в соответствии с командами программы, т. е. обеспечивает выполнение операций считывания и отработки команд в заданной последовательности, управляет процессом считывания и записи результатов выполнения операций в память, а также выдачей результатов вычислений в выходной модуль. ЦП реализуется на базе современной микропроцессорной техники и позволяет производить обработку, как битов информации, так и работы с байтами и словами.

Возможность работы с *битами* информации обеспечивает выполнение *базовых* логических операций И, ИЛИ, НЕ, эквивалентности и др. над дискретными логическими переменными, т. е. решать задачи электроавтоматики, аналогичные задачам, решаемым жесткой бесконтактной или релейно-контактной логикой.

Возможность обработки *байтов и слов* значительно расширяет возможности ПЛК, приближая его к вычислительным возможностям микро-ЭВМ.

Входные модули предназначены для приема сигналов от командных (кнопки управления, тумблеры, переключатели, и др.) и контролирующих (конечные выключатели, различного рода датчики и др.) элементов электрооборудования, их нормализации до уровня сигнала, определяемого элементной базой контроллера. Одновременно осуществляется гальваническая развязка, фильтрация от действия помех и индикация состояния входов.

Входные модули обычно выполняются на 8…32 входа с возможностью расширения общего числа входов ПЛК путем набора необходимого числа модулей.

Выходные модули преобразуют информацию, полученную от центрального процессора или модуля памяти в сигналы, управляют работой выходных силовых элементов электроавтоматики (промежуточных реле, пускателей, усилителей, электромагнитов, электромагнитных муфт и др.).

Предусматриваются модули на 8…32 выходов различной коммутационной способности, на различное напряжение и выполненные, в зависимости от назначения, на электромагнитных реле, транзисторах, симисторах и другой элементной базе. Индикация состояния выходов обычно выполняется малогабаритными сигнальными элементами расположенными непосредственно у выводов.

Расширение ПЛК до требуемого числа выходов также осуществляется путем набора необходимого числа модулей.

В целях дальнейшего повышения гибкости в компоновке ПЛК применяются также *смешанные модули*, включающие в себя как входные, так и выходные элементы.

Общее число входов и выходов при каскадном соединении каркасов с модулями может достигать 1024, 2048, 4096 и более.

Контроллер может включать также различные *специальные модули*, например, аналоговые входные (АЦП) и выходные (ЦАП) модули, модули позиционирования, модули быстрых счетчиков и др.

Блок памяти во многом определяет технические характеристики и возможности ПЛК. По функциональному назначению в архитектуре контроллера предусматривается два типа памяти: *служебная и рабочая*.

Служебная память предназначена для хранения программ управления работой контроллера, программ трансляторов и недоступна пользователю.

Рабочая память предназначена для хранения программ и информации пользователя и, следовательно, программно доступна. В рабочей памяти хранятся таблицы состояний входных и выходных сигналов, результаты промежуточных вычислений, уставки и текущие значения счетчиков, разрядов регистров, таймеров и др.

Память *программ* – это обычно память типа ПЗУ (постоянное запоминающее устройство). Международное обозначение ROM (Read Only Memory). Подобная память классифицируется следующим образом:

- собственно ПЗУ (ROM), т. е. память, запрограммированная изготовителем, последующее изменение информации невозможно;
- программируемое ПЗУ (ППЗУ). Международное обозначение PROM (Programmable Read Only Memory). Пользователь может сам программировать память подобного типа, например, выжиганием адресованных диодов в матрице микросхемы, после чего дальнейшее изменение содержимого памяти невозможно;
- репрограммируемое (или стираемое) ППЗУ – РППЗУ. Международное обозначение REPROM. Информация, записанная в память подобного типа может стираться и записываться вновь несколько раз. Применяется репрограммируемая память типа EPROM (erasably programmable ROM) с ультрафиолетовым стиранием, и типа EAROM (electrically alterable ROM) с электрическим стиранием.

Память *данных* – это память типа ОЗУ (оперативное запоминающее устройство). Международное обозначение RAM (Random Acsess Memory). В подобную память можно многократно записывать, стирать и считывать информацию, которая сохраняется при наличии питания. Обычно предусматривается подпитка от аккумуляторов или батареи.

По местоположению элементы памяти делятся на *внутренние и внешние*. Внутренняя память обеспечивает автономную работу ПЛК, внешняя, как правило, расположенная в программаторе – отладку и хранение программ.

Система шин команд, адресов и данных предназначена для обеспечения обмена и передачи информации между процессором, памятью, входными и выходными модулями. Центральный процессор имеет не показанную на рис. 1.1 свою собственную разветвленную шинную структуру.

Устройство управления (УУ) обеспечивает координацию работы ПЛК в соответствии с заданным циклом его функционирования.

Как правило, в состав ПЛК входят также различные *интерфейсные модули*, обеспечивающие связь с ЭВМ высшего ранга и различными периферийными устройствами, например, с телетайпом, принтером, перфоратором, панелями оператора и т. д.

Контроллеры могут быть как общего, так и специализированного назначения.

Программируемые контроллеры характеризуются большим разнообразием архитектуры и, как следствие, языкового уровня программирования электроавтоматики. Согласно международному стандарту IEC-1131 производители контроллеров применяют следующие типы языков программирования:

- язык релейно-контактных символов (LD – Ladder Diagram);
- аккумуляторный язык (IL – Instruction List);
- язык функциональных схем (FBD – Functional Blok Diagram);
- язык последовательного функционального управления (SFC – Sequential Function Chat).

При работе с программируемыми контроллерами следует помнить, что в разных типах контроллеров всегда отличаются схемы подключения, система адресации входов, выходов и промежуточной памяти, синтаксис языка и процедурные вопросы. Однако всегда сохраняется общий подход при решении любой задачи и, изучив какой-либо один тип контроллера, сделав несколько проектов, будет легко адаптироваться к другому типу.

Последовательность проектирования электроавтоматики следующая:

1. Изучить принцип работы автоматизируемого механизма.
2. Определить количество электроприводов и принять идеологию их управления.
3. Разработать систему органов управления.
4. Составить таблицу дискретных входов и выходов, определить их необходимое число.
5. Выбрать тип контроллера, отвечающий требованиям со стороны автоматизируемого объекта.

6. Изучить систему подключения контроллера, систему адресации входов и выходов и начертить принципиальную схему.

7. Изучить синтаксис языка электроавтоматики контроллера.

8. Разработать алгоритмы управления, поставить на компьютер необходимое программное обеспечение и набрать программу электроавтоматики на компьютере.

9. Установить связь между компьютером и контроллером, записать набранную программу в контроллер.

10. Отладить программу на стенде.

Примечание. В общей структуре электроавтоматики объекта автоматизации могут присутствовать также другие сложные аппаратные устройства, например:

- устройство цифровой индикации в комплекте с датчиками положения;
- программируемая панель оператора;
- регулируемые и сервоприводы;
- блоки электромагнитных муфт и другое.

В этом случае проделать аналогичную работу по изучению принципов их подключения, задания параметров, программирования и установки связи между ними.

Ниже излагаются общие начальные сведения о работе с PLC. Подробное изложение материала рассматривается в последующих главах.

1.2. Принципы функционирования контроллера

Пользователю исключительно важно знать основные принципы работы контроллера, так как в отличие от жестких релейно-контактных и бесконтактных схем электроавтоматики, осуществляющих *параллельную* обработку информации, программируемый контроллер работает по строго фиксированному вычислительному циклу и осуществляет *последовательную* обработку информации в соответствии с записанной в его памяти программой.

Основу работы ПЛК задает вычислительный цикл, в общем случае состоящий из трех этапов (рис. 1.2):

1. Считывание и запоминание во входном регистре логических значений входных сигналов (0 или 1), а также фиксация значений на данный момент всех сигналов промежуточной памяти.

2. Последовательная обработка данных в соответствии с программой, записанной в памяти контроллера и помещение результатов вычислений в адресованные промежуточные ячейки выходного регистра или промежуточной памяти.

3. Перепись результатов вычислений в регистровую память выходных модулей, и их передача на выходы с подключенными исполнительными элементами.

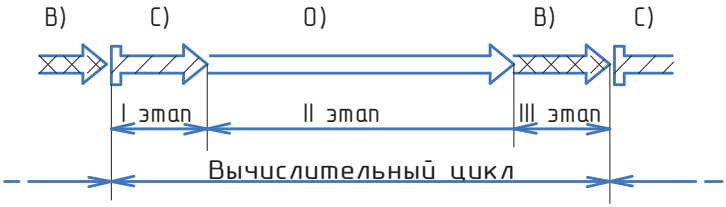


Рис. 1.2. Вычислительный цикл работы ПЛК

Если на втором этапе работы произойдет изменение состояния каких-либо входных сигналов, то эти изменения *не будут учитываться* до окончания текущего вычислительного цикла. Контроллер обработает эти сигналы *только в следующем* вычислительном цикле, т. е. произойдет задержка в один цикл (такт, скан).

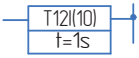
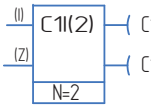
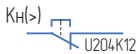
Сканирование (считывание) программы, как правило, осуществляется по *горизонтальными* строчкам программы (или РКС-алгоритма) слева направо, сверху вниз.

Однако можно встретить весьма экзотические решения. Вспоминается случай сорокалетней давности, контроллер типа TSX французской фирмы «Телемеканик». Сканирование релейного алгоритма осуществлялось *вертикальными* строчками сверху вниз, слева направо. Мы обнаружили это только тогда, когда некоторые неоднократно проверенные блоки программы перестали работать. В документации все было написано, даже был приведен рисунок последовательности сканирования, но мы не обратили на это внимания. Сюрпризы, очевидно, могут быть и в наше время.

Релейно-контактные алгоритмы, при их наборе, автоматически рисуются на экране компьютера в соответствии с синтаксисом языка конкретной фирмы. Битовые операнды (контакты) в большинстве случаев рисуются в виде «конденсаторов» без различия их назначения (кнопка, релейный вход, датчик давления, ограничительный конечный выключатель, промежуточный бит памяти и т. д.), выходные катушки в виде скобок и также обезличено (хотя при чтении алгоритма весьма полезно знать, что это – промежуточный сигнал или выходная катушка). Изображения функциональных команд (таймеры, счетчики, команды пересылок, сравнения, арифметики и т. д.) значительно отличаются от контроллера к контроллеру. Изменить здесь мы ничего не сможем, все подчиняется синтаксису конкретного языка. Небольшое спасение, это введение комментария (операндов, строк, блоков). Однако комментарии занимают много места и это очень трудоемкая работа.

Таблица 1.1

Графические символы, применяемые для изображения алгоритмов управления или их фрагментов

Назначение		Условное обозначение	Синтаксис языка PLC
Выходы	Аппаратный		U4A6=
	Ячейка ОЗУ		U50K0=
	Сообщение		U21K5=
Системная команда			U10K24=
Таймер			T12I(10)= T12U=
Счетчик			C1I(2)= C1Z= C1A= =[C1W]
Кнопка управления	на замыкание		I0A10
	на размыкание		/U204K12
Конечный выключатель	на замыкание		I1A8
	на размыкание		/I1A18
Операнд ОЗУ (пром. сигнал)	на замыкание		U65K14
	на размыкание		/U59K2

На каком бы языке мы не разрабатывали программу, автор настоятельно рекомендует при черновом проектировании алгоритмов использовать релейно-

контактную систему. И здесь мы вольны все делать так, как мы желаем, т. е. изображать операнды в виде понятных символов, сходных с символами, принятыми при начертании аппаратных принципиальных схем. Так мы и будем поступать (см. табл. 1.1).

Теперь вернемся к более подробному рассмотрению наиболее применяемых языковых средств. Обратимся к простейшей аппаратной релейно-контактной схеме управления реверсивным асинхронным двигателем (рис. 1.3) и реализуем разные варианты управления на ПЛК (см. рис. 1.7, 1.9 и 1.10).

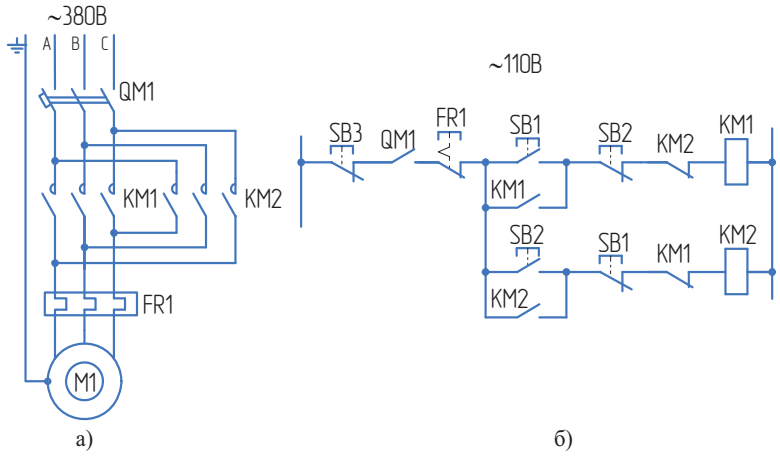


Рис. 1.3. Силовая схема (а) и аппаратная схема управления (б) асинхронным двигателем

Схема обеспечивает:

- пуск двигателя M1 по или против часовой стрелки из остановленного положения при нажатии, соответственно, кнопок SB1 или SB2;
- прямой реверс двигателя M1 с торможением противовключением при нажатии кнопки противоположного направления;
- останов двигателя на выбеге при нажатии кнопки SB2;
- максимальную защиту при помощи силового автомата QM1;
- тепловую защиту при помощи теплового реле FR1;
- нулевую защиту при помощи замыкающегося контакта автомата QM1, включенного в цепь управления силового пускателя KM1.

Рассмотрим, как реализовать эти же функции при управлении двигателем *от программируемого контроллера на различных языках*. Аппаратная принципиальная схема не зависит от языка программирования и должна включать следующие блоки:

– силовую схему подключения АД (одинакова);
– подключение питания контроллера, входных и выходных сигналов (определяется конкретным типом контроллера и зависит от полярности и величины питания дискретных входов и выходов, коммутационной способности выходов и выполняется по конкретной технической документации на ПЛК);
– аппаратную схему управления силовым пускателем КМ1.

При разработке аппаратной схемы управления пускателем необходимо учитывать специальные требования. В данном случае тепловое реле осуществляет косвенную защиту двигателя, поэтому его размыкающийся контакт должен быть включен в конечное звено, т. е. непосредственно в цепь силового пускателя. Обычно тепловое реле имеет дополнительный замыкающийся контакт, его следует подать на вход контроллера, что при срабатывании защиты позволит отключить память и реализовать диагностическое сообщение.

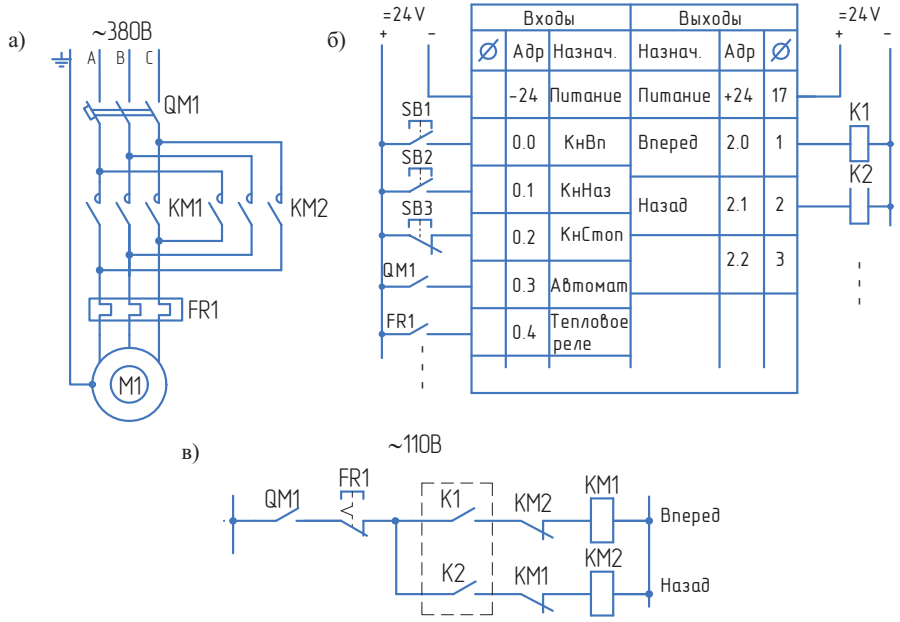


Рис. 1.4. Аппаратная схема управления АД от ПЛК

Вне зависимости от типа языка, перед написанием PLC-программы, следует разработать и формализовать в понятном виде алгоритм управления. Это внутренний документ разработчика и его можно сделать по-разному, однако автор настоятельно рекомендует использовать релейно-контактное изображение

алгоритма. При этом рекомендуется использовать стандартные обозначения принципиальных схем и максимально применять комментарии.

1.3. Описание языков программирования контроллеров

1.3.1. Программирование на языке мнемокода

Как уже было сказано и приведено в табл. 1.1, адресация и команды языка отличаются в зависимости от типа контроллера. Примем базовые команды и адреса контроллера японской фирмы «Фудзи», выпуска семидесятых годов, как наиболее простые:

R – считывание прямого значения адресованного операнда (Read);
RN – то же инверсного операнда (Read Not);
A – логическое умножение на прямой операнд (And);
AN – то же, на инверсный (And Not);
O – логическое сложение с прямым операндом (Or);
ON – то же, на инверсный (Or Not);
W – посылка результата вычисления на выход (Write).

В контроллере используется жесткая **безадресная** система адресации операндов типа «байт.бит». Для процессорного блока это:

– входы 0.0…0.7, 1.0…1.7;
– выходы 3.0…4.7;
– память 14.0…47.7.

Программа на языке мнемокода состоит из трех полей: *шага программы, команды (или инструкции) и операнда*. При черновом написании программы, во внутреннем документе разработчика, рекомендуется добавить четвертое поле – *комментарий*, это значительно облегчит работу.

Начнем с написания простейшей программы управления нереверсивным приводом охлаждения (рис. 1.5).

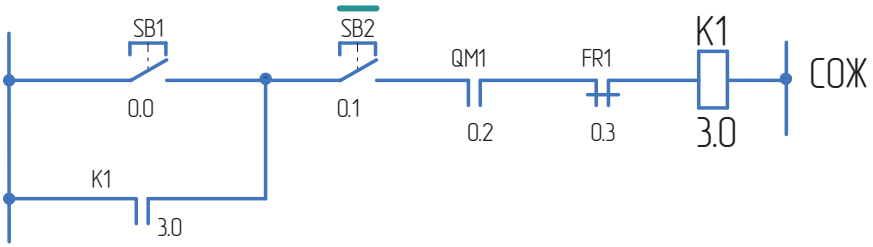


Рис. 1.5. Вариант начертания РКС-алгоритма управления АД при помощи ПЛК

Примечания:

1. Кнопка «Стоп» SB2 показана открытым контактом, так как на входе контроллера она подключена инверсно.

2. Контакт автомата QM1 показан открытым, так как в рабочем состоянии автомат включен и контакт замкнут.

Шаг	*Команда*	*Операнд*	*Комментарий*
1	R	0.0	Считывание SB1(0.0) и запись в аккумулятор, Акк = SB1
2	O	3.0	Логическое сложение содержимого аккумулятора с адресуемым операндом K1(3.0) и запись результата в аккумулятор, Акк = Акк + K1
3	A	0.1	Логическое умножение содержимого аккумулятора на адресуемый операнд SB2(0.1) и запись результата в аккумулятор, Акк = (SB1+K1) * SB2
4	A	0.2	Логическое умножение содержимого аккумулятора на адресуемый операнд QM1(0.2) и запись результата в аккумулятор, Акк = (SB1 + K1) * SB2 * QM1
5	AN	0.3	Логическое умножение содержимого аккумулятора на инверсный адресуемый операнд FR1(0.3) и запись результата в аккумулятор, Акк = (SB1 + K1) * SB2 * QM1 * / FR1
6	W	3.0	Посылка содержимого аккумулятора на адресуемый выход K1(3.0) K1 = (SB1 + K1) * SB2 * QM1 * / FR1

Видно, что язык мнемокода напоминает язык программирования микро-ЭВМ *Ассемблер*, являющийся мнемонической формой машинных языков. Программист должен постоянно отслеживать результаты обработки алгоритма, следить за содержимым аккумулятора, что достаточно затруднительно и часто приводит к ошибкам в наборе программы.

Тем не менее, такие языки в настоящее время применяют даже очень известные продвинутые фирмы, например, немецкая фирма «Хайденхайн».

Написание программы желательно выполнять в *одинаковой* последовательности с РКС-эскизом алгоритма, это облегчит его чтение и наладку. В этой связи обратимся к рис. 1.6, где кнопки «Пуск» и «Стоп» поменяны местами по сравнению с рис. 1.5.

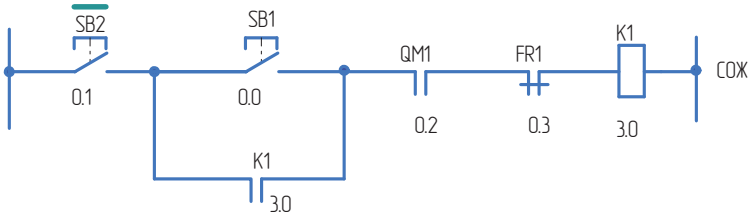


Рис. 1.6. Вариант начертания РКС-алгоритма управления АД при помощи ПЛК

Если сделать первый шаг R 0.1 и записать в аккумулятор инверсное значение кнопки «Стоп», то дальнейшее написание программы *простыми базовыми* инструкциями будет *невозможным*, так как содержимое аккумулятора, согласно начерченному алгоритму, нужно умножать на результат параллельного соединения кнопки «Пуск» и контакта реле К1, но такой информации нет. Следовательно, нужно изменить последовательность программирования, сначала обработав логическую сумму (SB1 + K1) и записав результат в промежуточную память. Далее все по порядку. Естественно, это неудобно, и здесь на помощь приходят *специальные функциональные* инструкции и использование скобок, позволяющие работать со *стеком*.

Стек – это специальный параллельный регистр (массив данных) со строго заданным порядком загрузки и выгрузки информации. Существует два способа организации стека:

FIFO (First In First Out) – первый вошел, первый вышел;

LIFO (Last In First Out) – последний вошел, первый вышел.

Процессор использует стек, организованный по системе LIFO, его можно сравнить с подпружиненным кластером для хранения металлических монет. Каждая новая монета кладется сверху и сжимает пружину, проталкивая предыдущую монету вниз. Выемка может происходить в обратной последовательности. Так же производится и обработка битовой информации.

В рассматриваемом контроллере имеется одноразрядный стек, управляемый следующими командами:

– A MRG – логическое умножение содержимого аккумулятора на содержимое регистра MRG, результат снова записывается в аккумулятор;
– MRG – логическое сложение содержимого аккумулятора с содержимым регистра MRG, результат снова записывается в аккумулятор.

Проталкивание в стек осуществляется командами R и RN, а выталкивание командами A MRG и O MRG.

Одноразрядность стека снова накладывает условности при программировании. В случае длинных составных цепочек необходимо следить за переполнением стека. Команды, осуществляющие проталкивание и выталкивание должны чередоваться друг с другом. Исключение составляет начало цепи, когда допустимо задание подряд двух команд записи R и RN до заполнения стека.

В настоящее время разрядность стека, как правило, достаточно большая и о переполнении стека можно не беспокоиться (но помнить, что такая ситуация может возникнуть, нужно).

Программу для рис. 1.6, с учетом сказанного, можно записать следующим образом:

Шаг	*Команда*	*Операнд*	*Комментарий*
1	R	0.1	Считывание SB2(0.1) и запись в аккумулятор, Акк = SB1
2	R	0.0	Проталкивание содержимого аккумулятора (SB2) в стек и считывание SB1(0.0) и запись в аккумулятор, Акк = SB1
3	O	3.0	Логическое сложение содержимого аккумулятора (SB1) с адресуемым операндом K1(3.0) и запись результата в аккумулятор, Акк = SB1 + K1
4	A MRG		Логическое умножение содержимого аккумулятора (SB1 + K1) с содержимым стека (SB2) и запись результата в аккумулятор, Акк = SB2* (SB1 + K1)
5	A	0.2	Логическое умножение содержимого аккумулятора на адресуемый операнд QM1(0.2) и запись результата в аккумулятор, Акк = SB2 * (SB1 + K1) * QM1
6	AN	0.3	Логическое умножение содержимого аккумулятора на инверсный адресуемый операнд FR1(0.3) и запись результата в аккумулятор, Акк = SB2 * (SB1 + K1) * QM1 * / FR1
7	W	3.0	Посылка содержимого аккумулятора на адресуемый выход K1(3.0) K1 = SB2 * (SB1 + K2) * QM1 * / FR2

Еще проще работать с *командами скобок*, если они есть в синтаксисе языка, так что обозначенные неудобства в современных контроллерах с языком мнемокода успешно преодолеваются. Однако нужно помнить, что каждый контроллер имеет какие-либо особенности и ограничения, которые необходимо выяснить по сопроводительной технической документации.

Сделаем еще одно важное замечание: когда говорится о языке мнемокода, то имеется ввиду программист электроавтоматики, что именно он программирует на данном языке. Фактически во всех контроллерах и на всех языках, после сканирования программы, базовая математика PLC работает с внутренним аккумулятором, но разработчика программы электроавтоматики это не касается.

1.3.2. Программирование на языке релейно-контактных символов

Программирование на данном языке выполняется *непосредственным набором* РКС-программы (Ladder-диаграммы) с клавиатуры программатора, персонального компьютера или с клавиатуры системы ЧПУ, используя предоставляемый разработчиком контроллера графический редактор. Это наиболее простой, понятный и поэтому наиболее распространенный способ программирования. Следует сказать, что любая аппаратная релейная принципиальная схема, адаптированная с учетом специфики синтаксиса используемого языка РКС, может служить готовым алгоритмом для набора программы (но, не наоборот, о чем пойдет речь в следующих главах книги).

Таким образом, принципиальная схема (рис. 1.3, *б*) может служить РКС-алгоритмом (шаблоном) программы. Следует только установить адресацию операндов в соответствии с принятой на конкретный контроллер системой адресации. Приведем пример реализации на контроллере фирмы «Дельта».

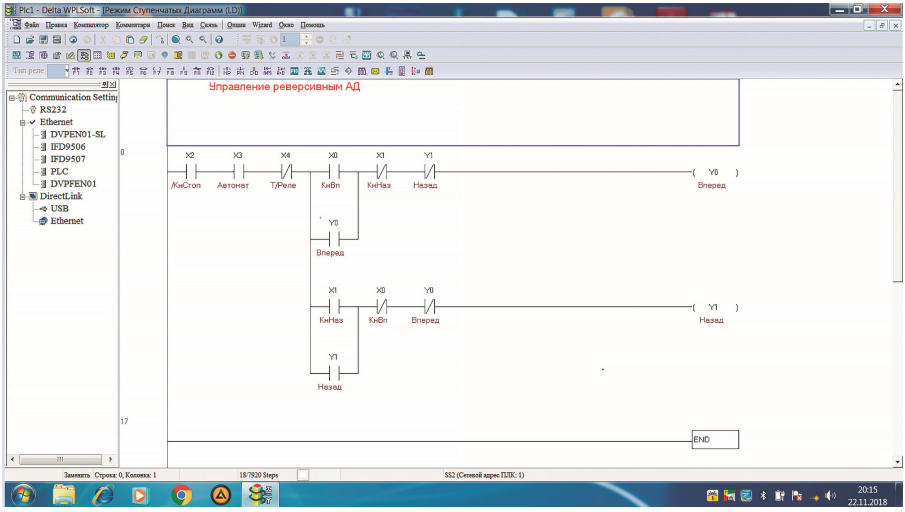


Рис. 1.7. Ladder-алгоритм управления реверсивным АД (контроллер фирмы «Дельта»)

Ниже приведена программа управления тем же объектом на контроллере фирмы «Дельта», но на языке *мнемокода* (рис. 1.8).

В ней используются:

1) базовые команды:

LD – считывание прямого операнда,
AND – логическое умножение на прямой операнд,
ANI – логическое умножение на инверсный операнд,
OR – логическое сложение с прямым операндом,
OUT – посылка результата на выход;

2) специальные команды:

MPS – запись в стек (общая часть),
ANB – логическое умножение на содержимое стека,
MPP – возврат к стеку.

Программы, написанные на языках РКС и мнемокода, можно легко преобразовывать друг в друга при помощи редактора, что и было сделано в данном случае. Набранная Ladder-диаграмма переведена на язык мнемокода при компиляции.

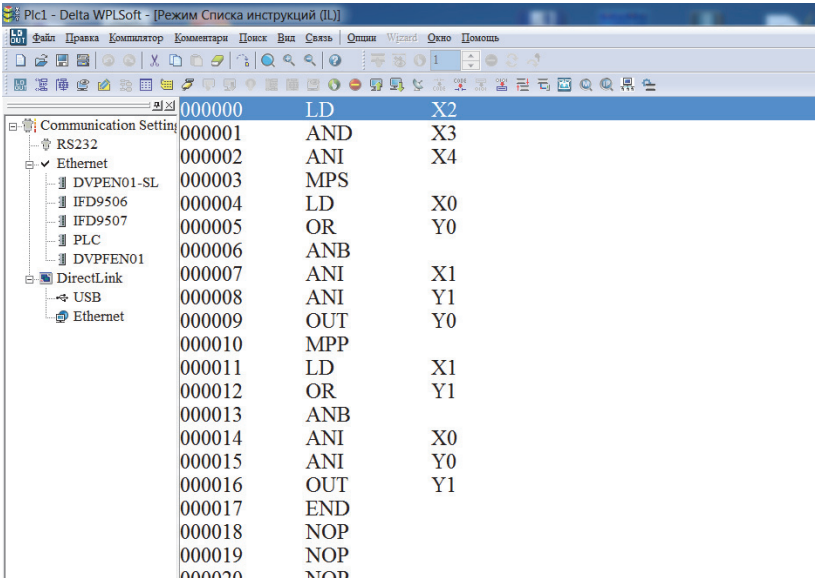


Рис. 1.8. Программа управления реверсивным АД на языке мнемокода фирмы «Дельта»

1.3.3. Программирование на языке функциональных инструкций

Программирование на языке функциональных инструкций при использовании базовых команд аналогично разработке бесконтактных логических схем в базисе И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ. При этом логическая схема, являющаяся PLC-программой, набирается непосредственно на экране дисплея. Для функциональных команд все аналогично, но по синтаксическим правилам и структуре языка конкретного контроллера.

На рис. 1.9 показана PLC-программа управления реверсивным асинхронным двигателем, написанная (нарисованная) на языке функциональных инструкций контроллера типа Zelio-Logic.

Назначение входов (I):

I1 – Кнопка «Вперед» (SB1);
I2 – Кнопка «Назад» (SB2);
I3 – Кнопка «Стоп» SB3 (инверсно, т. е. нормально замкнута);
I4 – Силовой автомат QM1 (инверсно, т. е. нормально включен);
I5 – Тепловое реле FR1 (второй контакт, нормально выключен).

Назначение выходов (Q):

Q1 – Вперед (K1 → KM1);
Q2 – Назад (K2 → KM2).

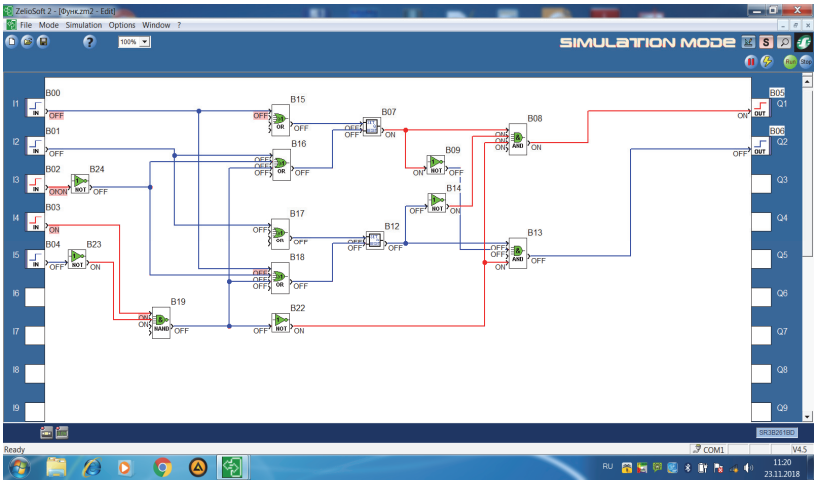


Рис. 1.9. PLC-программа управления реверсивным асинхронным двигателем на языке функциональных схем (контроллер Zelio-Logic)

В приведенной программе для организации памяти также использованы функциональные команды RS-триггеров.

1.3.4. Программирование на языке логических уравнений

Подобные языки позволяют писать PLC-программы непосредственно в логических уравнениях при использовании базовых команд и с некоторыми синтаксическими ограничениями при использовании функциональных команд. Язык логических уравнений используется, например, во встроенных в системы ЧПУ контроллерах фирм «Балт-Систем», «Модмаш-Софт», «Маяк» и др. Самым эффективным способом формализации постановки задачи здесь также являются релейные алгоритмы, по которым можно сразу писать программу. Покажем это на примере управления реверсивного асинхронного двигателя (рис. 1.10).

При написании программы используются синтактические правила языка и конкретные адреса входных, выходных и промежуточных сигналов, например, как показано в таблице 1.2. Обращаем внимание, что на рис. 1.10 показана принятая адресация для контроллера фирмы «Балт-Систем».

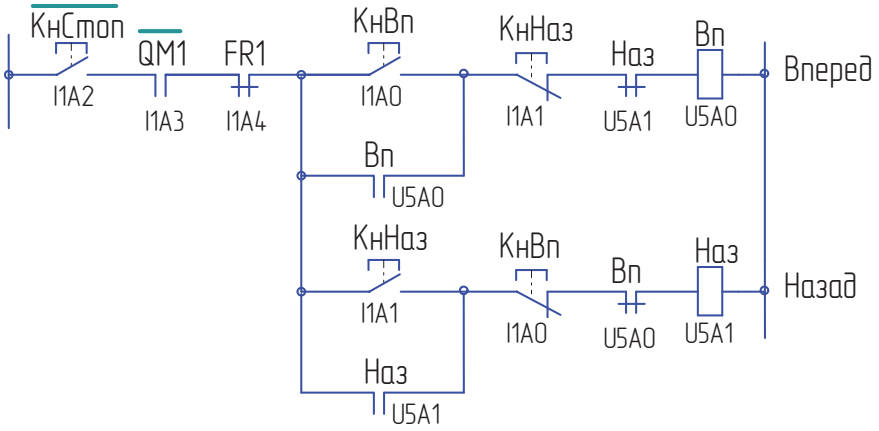


Рис. 1.10. РКС-алгоритм управления реверсивным асинхронным двигателем (формализация для написания программы)

Рабочие программы для разных систем ЧПУ запишутся так:

– для систем серии NC фирмы «Балт-Систем»:

U5A0=I1A2*I1A3*/I1A4*(I1A0+U5A0)*/I1A1*/U5A1

U5A1=I1A2*I1A3*/I1A4*(I1A1+U5A1)*/I1A0*/U5A0

– для систем фирмы «Маяк»:

A1.0=A17.2&&A17.3&&!A17.4&&(A17.0||A1.0)&&!A17.1&&!A1.1

A1.1=A17.2&&A17.3&&!A17.4&&(A17.1||A1.1)&&!A17.0&&!A1.0

– для систем фирмы «Модмаш»:

U1.1=I1.3*I1.4*/I1.5*(I1.1+U1.1)*/I1.2*/U1.2:

U1.2=I1.3*I1.4*/I1.5*(I1.2+U1.2)*/I1.1*/U1.1:

Таблица 1.2

Примеры синтаксических правил языка и конкретные адреса входных, выходных и промежуточных сигналов

Назначение	Фирма			Примечание
	«Балт-Систем»	«Модмаш»	«Маяк»	
Кнопка Вперед	I1A0	I1.1	A17.0	Входы
Кнопка Назад	I1A1	I1.2	A17.1	
Кнопка Стоп	I1A2	I1.3	A17.2	
Автомат	I1A3	I1.4	A17.3	
Тепловое реле	I1A4	I1.5	A17.4	
Пускатель ВПЕРЕД	U5A0	U1.1	A1.0	Выходы
Пускатель НАЗАД	U5A1	U1.2	A1.1	
Инверсия	/	/	!	Базовые команды
Лог. умножение	*	*	&&	
Лог. сложение	+	+	||	
Присвоение	=	=	=	
Конец кадра	Нет	:	нет	

1.3.5. Программирование на языках, основанных на формировании задач на обиходном языке (символьные языки)

Запись программы в этом случае осуществляется ключевыми словами и символами на обиходном, т. е. родном языке пользователя. Команды ПЛК включают в себя такие слова как: ЕСЛИ, ТОГДА (или ТО), ИНАЧЕ, вход, выход, включить, выключить и др.

Программа для логического уравнения $Y = X_1 \cdot \overline{X_2} + X_3$ (рис. 1.11) в этом случае запишется так:

ЕСЛИ	вход	X1
И	Не	X2
ИЛИ		X3
ТОГДА	включить	Y
ИНАЧЕ	выключить	Y

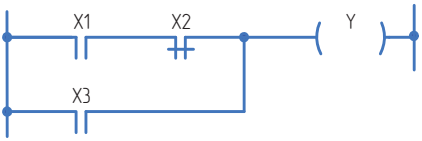


Рис. 1.11. РКС-алгоритм примера программирования на обиходном языке

Символьные языки позволяют осуществлять программирование не только по логическим уравнениям и релейно-контактным схемам, но и по классическим алгоритмам, описывающим работу механизма. Кажущаяся простота обманчива, программирование сложных разветвленных цепей и цепей с памятью весьма затруднительно и широкого распространения такие языки не получили.

1.3.6. Программирование на языках последовательного функционального управления (Графсет, Хайграф и др.)

Подобные языки позволяют составлять программы управления сложными промышленными механизмами путем последовательного графического представления этапов работы механизма, предваряемых условиями разрешения выполнения каждого этапа. Программа состоит из последовательности *шагов (этапов)*, описывающих активные действия исполнительных органов (включить шпиндель, включить охлаждение, повернуть манипулятор на 90 градусов и т. д.) и *переходов*, описывающих условия разрешения выполнения этапа (инструмент зажат, манипулятор в исходном положении, шпиндель выключен и ориентирован и т. д.), изображается в графическом виде, представляя собой *граф* последовательных шагов и переходов, синтезированных с учетом правил синтаксиса языка.

Используемые для построения структуры графа символы (для контроллера французской фирмы «Телемеканик») приведены на последующих рисунках.

Последовательные замкнутые графы (рис. 1.12) являются простейшими случаями применения данного языка и составляют его основу.

Как видно из рисунка, последовательный граф представляет собой аналог аппаратного решения с помощью *счетчика последовательности*, когда каждая последующая операция выполняется при условии наличия задания на выполнение предыдущей операции и контроля о ее выполнении.

В подобном графе всегда активен только один этап. Начальный (активный) этап, в принципе, может быть в любом месте графа.

Разветвленные графы имеют более сложную структуру, в них возможна организация:

– расходимости и сходимости графа по ИЛИ;

– расходимости и сходимости графа по И;
– условных и обратных переходов (см. рис. 1.14), т. е. они позволяют создать любые сложные структуры и делают практически неограниченными его возможности.

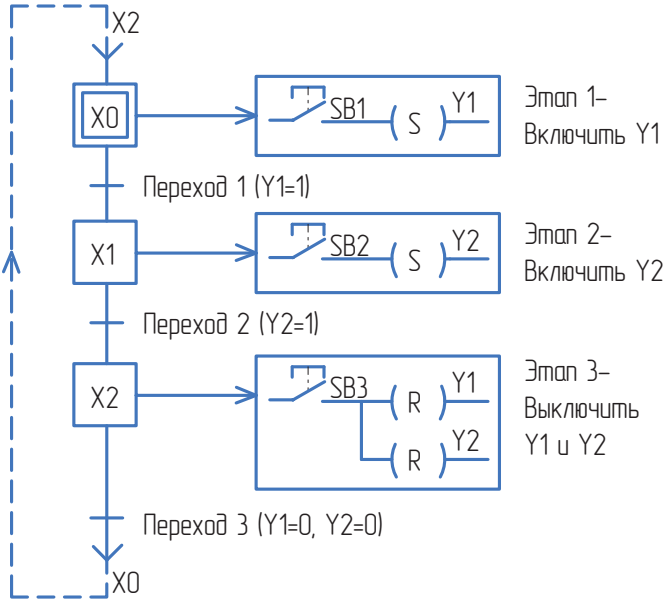


Рис. 1.12. Последовательный граф с одним активным элементом

В случае *расходимости по ИЛИ* (рис. 1.13, *а*) после активизации X0 осуществляется переход на одну или несколько ветвей, в зависимости от активности переходов #1 или #2 или #3 или #5. После завершения работы любой из активных ветвей по переходам #8/1 или #8/4 или #8/6 или #8/7 осуществляется переход к этапу X8, т. е. *сходимость графа по ИЛИ*. Обращаем внимание, что в алгоритме также показаны разрешенные синтаксисом языка внутренние расходимости и сходимости графа по ИЛИ.

В случае *расходимости по И* (рис. 1.13, *б*) после активизации X0 и активности перехода #1,3 осуществляется одновременный переход на одну или несколько ветвей, в зависимости от программы (здесь две ветви X1 и X3). Сходимость графа, т. е. переход на общий этап осуществляется только после завершения работы всех активных этапов. Обращаем внимание, что в алгоритме также показаны разрешенные синтаксисом языка внутренние расходимости и сходимости графа по И.

Приведенные ниже примеры переходов пояснений не требуют.

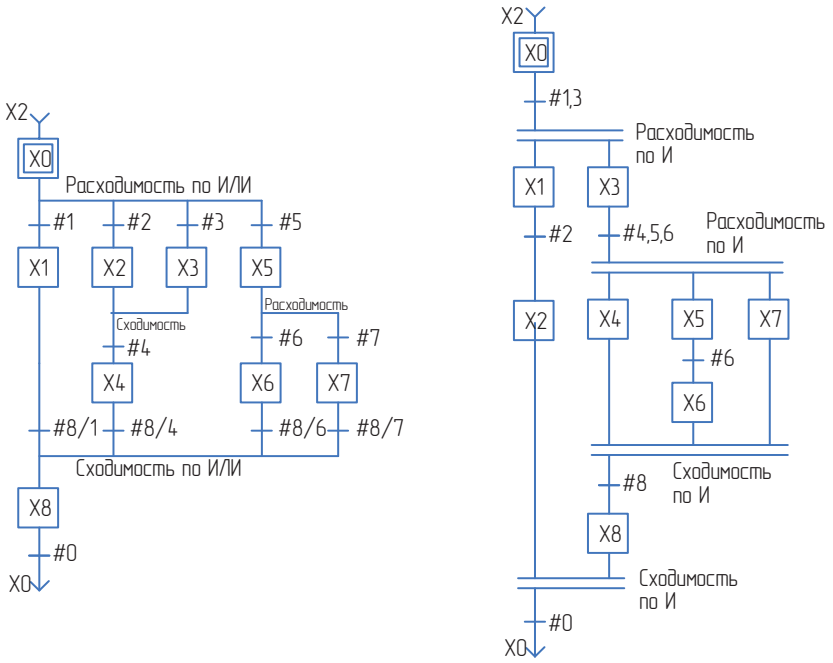


Рис. 1.13. Расходимость и сходимость графа по ИЛИ (а) и по И (б)

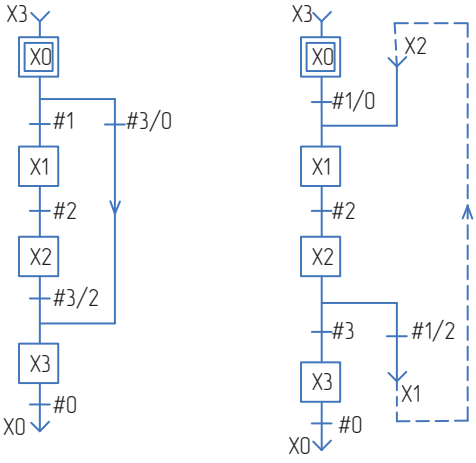


Рис. 1.14. Условный (а) и обратный (б) переходы графа

Тем не менее, в станкостроении такие языки широкого распространения не получили. Причина все та же, программирование на языке релейно-контактных символов проще и привычнее, хотя и не всегда оптимально.

1.3.7. Языки высокого уровня

К ним относятся, например, Си, Паскаль, Бейсик и др.

Примером могут служить устройства ЧПУ типа Микрос, где позиционируется программирование на специализированном языке Мик-Си, основанном на базе языка высокого уровня Си.

Программа электроавтоматики включает в себя следующие разделы:

- комментируемый заголовок, несущий информацию о типе станка и его основных характеристиках, фирме-изготовителе, дате разработки;
- препроцессор, т. е. поле определения переменных, подключения библиотеки др., при необходимости;
- главная циклическая программа **NODE_0**;
- подпрограммы **NODE_(1…99)**;
- подпрограммы быстрой логики **LOCK_(0…99)**;
- функции языка Си **funcN( )**;
- конец программы //.

Основу **структурной организации языка** составляют **операторы:**

- **циклической программы: NODE_0**
- **вызываемых блоков: NODE_n**
- **вызываемых блокировок: LOCK_n**
- **функций** (подпрограмм): **void SUB1(void)**

Структура циклической программы NODE_0

NODE_0
{
Начальные установки
for (;;)
{
PAUSE(0.2);
тело циклической программы
……
……
……
 	
}
}
END_NODE_0
	Начало программы
	
	Примечание. Из тела программы можно запускать ПП LOCK(N), NODE(N), вызывать Функции, осуществлять остановы (STOP) других ПП
	
	Конец программы

Начало программы отмечается оператором NODE_0, а конец оператором END_NODE_0. Содержимое всей программы заключается в фигурные скобки { }. Операторы начала и конца программы записываются **заглавными** латинскими буквами.

Программа состоит из блока **однократных** начальных установок и тела **циклической** программы, заключенной в новые фигурные скобки { } после команды **for(;;)** и **бесконечно** сканируемой в процессе работы.

Циклическая программа **NODE_0** сканируется постоянно, из нее осуществляется первоначальный вызов того или иного блока **NODE_n**, блокировки **LOCK_n** или подпрограммы **void SUB1(void)**. Она является программой **медленной** логики, максимальный цикл сканирования которой составляет 20 мс.

В теле подпрограммы могут быть любые команды языка электроавтоматики, разрешенные его синтаксисом.

Для вызова блоков и подпрограмм используются следующие команды:

RUN_NODE(n); – вызов блока;
RUN_LOCK(n); – вызов блокировки;
sub1(); – вызов функции (подпрограммы).

Для останова блоков и подпрограмм используются следующие команды:

STOP_NODE(n); – останов блока;
STOP_LOCK(n); – останов блокировки;
return; – выход из функции (подпрограммы);
STOP_ALL(); – останов всех блоков.

Структура подпрограмм NODE_1…NODE_99

Запускающая программа	NODE_N	Начало ПП
	{	
RUN_NODE(N);		*Примечание.*
(однократный запуск ПП)	……	1. Из ПП можно
	……	запускать другие ПП
	……	LOCK(N), NODE(N),
	тело программы	вызывать Функции,
	……	осуществлять остановы
	……	(STOP) других ПП.
	……	2. Можно
	……	осуществлять
		останов самой себя
	}	
STOP_NODE(N);	END_NODE_N	Конец ПП
(останов сканирования ПП)		

Начало подпрограммы отмечается оператором NODE_N, а конец оператором END_NODE_N. Содержимое всей подпрограммы заключается в фигурные скобки { }. Операторы начала и конца подпрограммы записываются **заглавными** латинскими буквами. Индекс N может принимать заначения в диапазоне 1…99.

Запуск подпрограммы NODE_N осуществляется командой **RUN_NODE(n);** из любой другой запущенной подпрограммы. Естественно, первая подпрограмма (по логике работы, а не по номеру) может быть запущена только из основной циклической программы NODE_0. Однажды запущенная подпрограмма сканируется циклически постоянно вслед за главной программой в порядке ее написания в листинге и является подпрограммой медленной логики.

Останов подпрограммы NODE_N осуществляется командой **STOP_ NODE(n);** из любой другой запущенной подпрограммы. Подпрограмма может также остановить сама себя.

В теле подпрограммы могут быть любые команды языка электроавтоматики, разрешенные его синтаксисом. Она может вызывать и останавливать любые другие подпрограммы с индексами 1…99.

Структура подпрограмм LOCK_0…LOCK_99

Запускающая программа	LOCK_N	Начало ПП
	{	
RUN_LOCK(N);		*Примечание.*
(однократный запуск	……	1. Из ПП можно запускать
ПП)	……	другие ПП LOCK(N),
	……	NODE(N), вызывать Функции,
	тело программы	осуществлять остановы (STOP)
	……	других ПП.
	……	2. В ПП LOCK(N) нельзя
	……	программировать команды
		остановов PAUSE и DURING
	}	
STOP_LOCK(N);	END_LOCK_N	Конец ПП
(останов сканирования		
ПП)		

Начало подпрограммы блокировок отмечается оператором LOCK_N, а конец оператором LOCK_NODE_N. Содержимое всей подпрограммы заключается в фигурные скобки { }. Операторы начала и конца подпрограммы записываются **заглавными** латинскими буквами. Индекс N может принимать значения в диапазоне 0…99.

Запуск подпрограммы LOCK_N осуществляется командой **RUN_LOCK(n);** из любой другой запущенной подпрограммы. Естественно, первая подпрограмма

(по логике работы, а не по номеру) может быть запущена только из основной циклической программы NODE_0.

Подпрограммы блокировок **LOCK_n** являются программами **быстрой** логики, сканируемой каждые 5 мс параллельно с главной программой. Если из нее вызывается какая-либо функция (подпрограмма), то она будет также программой быстрой логики.

Останов подпрограммы LOCK_N осуществляется командой: **STOP_ LOCK(n);** из любой другой запущенной подпрограммы. Подпрограмма может также остановить сама себя.

В теле подпрограммы могут быть любые команды языка электроавтоматики, разрешенные его синтаксисом, **кроме** операторов выдержек времени PAUSE и DURING.

Она может вызывать и останавливать любые другие подпрограммы.

Структура функций funcN( )

Вызывающая программа	void funcN(void)	Начало функции
	{	
……		
……	……	
……	……	
	……	
func( );	тело функции	
……	……	
……	……	
……	……	
	return;	
	}	Конец функции

Логическая часть электроавтоматики станка прописывается в теле программ, подпрограмм и функций в соответствии с синтаксисом языка Мик-Си, включающем специальные команды УЧПУ Микрос и стандартные команды языка С++.

Приведем пример практической реализации **дешифратора М-функций**.

Считывание М-функций осуществляется при помощи системной ячейки **BS0**, сопровождаемой стробом М0 (**S10.1**).

Содержимое байта **BS0** изменяется при задании в управляющей технологической программе новой М-функции. В кадре программы может быть задана только **одна** М-функция. Строб М0 сбрасывается ответом о выполнении операции.

Дешифратор М-функций может быть реализован различными способами, например, используя классическую дешифрацию. Однако предпочтительнее использовать функцию переключателя (**switch**).

Принцип работы переключателя рассмотрен в материалах по языку Си.

Обработка переключателя выносится в отдельную подпрограмму, вызываемую из главной программы, в нашем случае это NODE_10.

Фрагмент программы электроавтоматики приведен ниже.

```
//
NODE_10
{
……..
……..

/* Дешифратор М-функций */
if(S10.1==0) goto end;
switch(BS0)
{
case 2: M2( );
break;
case 3: M3( );
break;
case 4: M4( );
break;
case 5: M5( );
break;
case 8: M8( );
break;
case 9: M9( );
break;
……..
……..
default: Message("*Недопустимый код М-функции");
// break;
}
L8.1=1; // Ответ М-функции
end:
S10.6=S10.7=S11.0=0; // Ответы М0, М1, М2
}
END_NODE_10
```

Если новой М-функции нет, то нет и строба **S10.1**. По команде **goto end** осуществляется условный переход на метку **END**, переключатель не работает.

Если из технологической программы считывается новая М-функция, то по команде **case** вызывается соответствующая функция, например, **M8( )** включения охлаждения.

После выполнения команды осуществляется возврат на следующую за вызывающей командой M8( ) строку **break**, переход за переключатель (}) и формирование виртуального ответа L8.1 о выполнении команды M8 включения охлаждения.

Поскольку одновременно может быть только одна М-функция, то ячейка ответа L8.1 общая для всех команд. Далее, используя виртуальный ответ, нужно сформировать системный ответ **S25.4**.

Добавим, что достаточно часто в контроллерах, для расширения их функциональных возможностей, разработчики систем ЧПУ в дополнение к рассмотренным выше классическим языкам электроавтоматики добавляют фрагменты языков высокого уровня.

1.3.8. Специализированные языки

Специализированные языки – это, например, язык программирования Фокон, применяемый при программировании электроавтоматики для фонового контроллера устройства числового программного управления типа 2С42-65 или ЯФП (язык функциональных инструкций) системы ЧПУ Маяк-400.

В заключение обзора языковых средств электроавтоматики скажем, что встречаются чрезвычайно специфические, лучше сказать, экзотические языки, чего стоит только язык под названием ЯФП (язык функциональных инструкций), разработанный в свое время в Ленинграде и применявшийся в ранних версиях Ижевских систем ЧПУ «Маяк». Но не нужно бояться, дорогу осилит идущий. Ключ к успеху – систематизация и разработка типовых приемов и решений. Например, автор никогда не программировал на языке Си, ну и что, купил книжку «Си для чайников», пролистал, отбросил все лишнее, систематизировал минимальный набор команд, достаточных для решения задач станочной электроавтоматики и в течение месяца написал первую программу для токарного станка. Дальше стало проще. Даже написал книжку «Проектирование электроавтоматики на базе устройства ЧПУ типа МИКРОС-12Т(Ф)».

Из рассмотренного материала видно, что все языки программирования ПЛК являются проблемно-ориентированными, так как адаптивны к одной цели – решению задач промышленной электроавтоматики.

1.4. Варианты подключения дискретных входов и выходов

Одним из важнейших этапов проектирования электроавтоматики является правильное подключение дискретных входов и выходов контроллера и связанная с этим система их адресации. Следует строго следовать сопроводительной технической документации, и что важно, внимательно ее читать. Это связано с тем, что разные контроллеры имеют разную систему подключения и разную идеологию адресации дискретных входов и выходов, а адреса могут быть *не по порядку*.

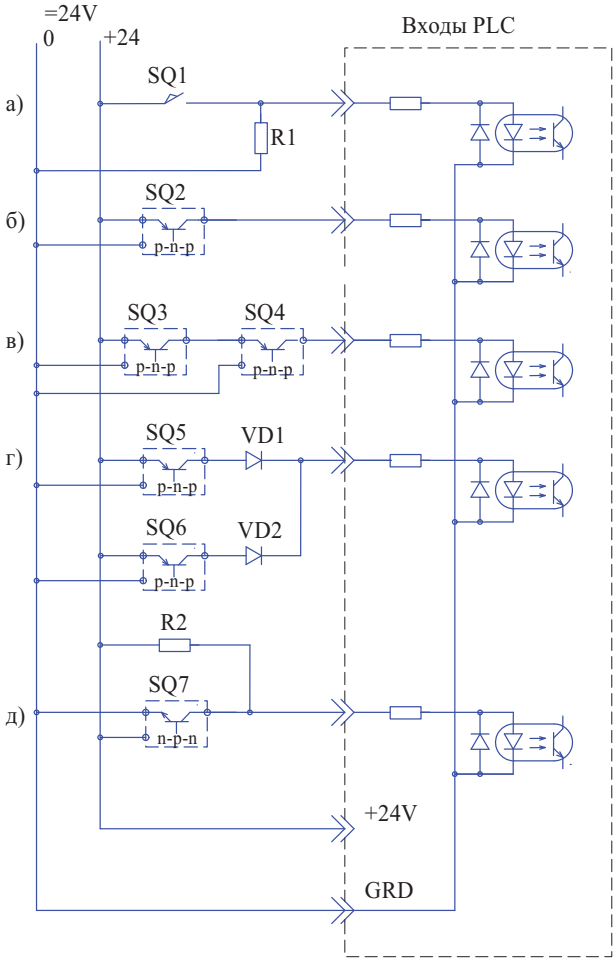


Рис. 1.15. Варианты подключения дискретных входов

Систему подключения можно классифицировать, например, следующим образом:

- непосредственное подключение или через специальные переходные блоки входов и выходов;
- подключение посредством разъема, втычное подключение или подключение посредством винтов;
- положительное, отрицательное или любое напряжение питания;
- питание от внешнего или внутреннего источника;
- подключение с одним общим проводом питания для всего разъема (адресного слова) или группами по нескольку сигналов;
- транзисторные, тиристорные или релейные выходные сигналы;
- нулевой или питающий общие провода выходных сигналов;
- наличие или отсутствие индикации состояния сигналов и др.

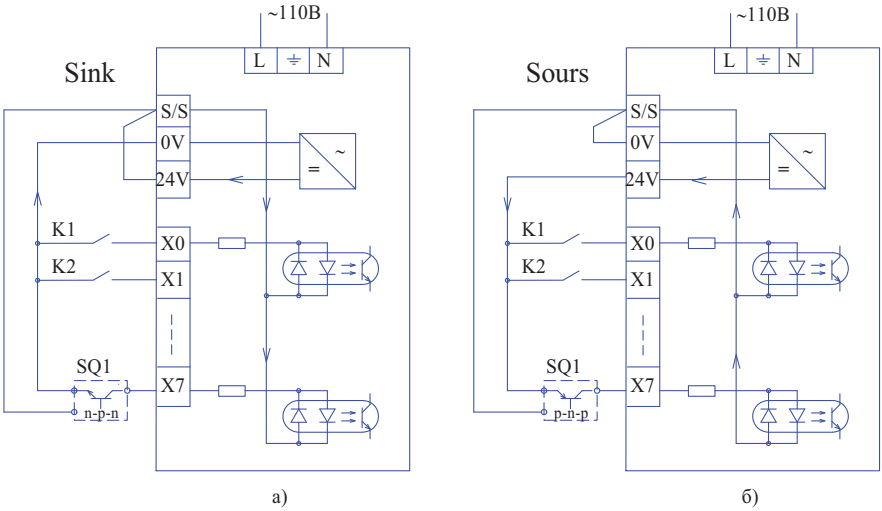


Рис. 1.16. Варианты подключения дискретных входов:
а – Sink; *б* – Sours

На рис. 1.15 приведен пример подключения к контроллеру с положительным питанием различных входных сигналов:

а) контактный входной сигнал. Резистор R1 устанавливается, при необходимости, если входной контакт не обеспечивает коммутацию штатного входного тока;

б) полупроводниковый p-n-p датчик, обеспечивающий прямое протекание входного тока;

в) каскадное соединение двух p-n-p датчиков;
г) параллельное соединение двух p-n-p датчиков;
д) полупроводниковый n-p-n датчик. Установка резистора R2 обеспечивает инверсную работу входа. Такое решение можно применять только в крайнем случае.

В ряде зарубежных контроллеров предусматриваются клеммы или перемычки для переключения полярности входных сигналов, режимы Sink или Sours (рис. 1.16).

Также предусматривается возможность, по желанию, использовать внешний или внутренний источник питания.

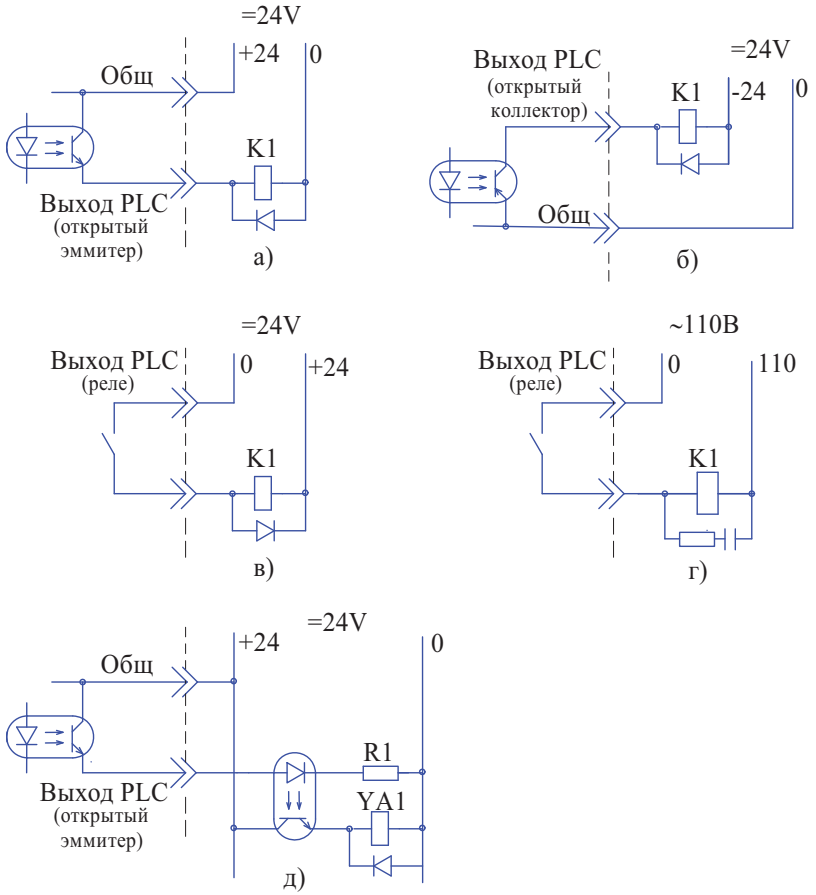


Рис. 1.17. Варианты подключения дискретных выходов

На рис. 1.17 приведены возможные варианты подключения дискретных выходов:

а) транзисторный выход с открытым эмиттером (общий плюс);
б) транзисторный выход с открытым коллектором (общий минус);
в) релейный выход с нагрузкой постоянного тока;
г) релейный выход с нагрузкой переменного тока;
д) транзисторный выход с подключением нагрузки через оптронную развязку.

При подключении выходов также может предусматриваться возможность коммутации внешнего и внутреннего источников питания.

1.5. Устройства цифровой индикации

Устройства цифровой индикации (УЦИ) изначально создавались для оснащения универсальных станков в целях, исключительно, контроля за величиной перемещения координат станка. В настоящее время, в связи с мощным развитием аппаратных и программных средств, УЦИ получили много дополнительных технологических возможностей.

Аппаратно они могут быть одно-, двух- или трехкоординатными; иметь ограниченное число органов управления или расширенную клавиатуру.

Программно это может быть как простейший вариант с отображением текущего положения, так и вариант с разным набором дополнительных функций, например, изменением системы отсчета, компенсации люфтов, компенсации погрешностей механических структур, элементов позиционирования, индикации радиуса или диаметра для токарных станков, обеспечение совместной работы с персональным компьютером и другие.

Устройства оснащаются линейными датчиками измерения перемещений. Наибольшее применение нашли фото-импульсные и шариковые датчики.

Дальнейшее развитие УЦИ привело к созданию на их базе полноценных позиционных систем управления со встроенным программируемым контроллером и различными модулями управления.

Широкую гамму устройств цифровой индикации выпускают многие ведущие производители систем управления для станко-инструментальной промышленности, в первую очередь, это Хайденхайн, Сименс, Фагор, Newall и др. В отечественной промышленности здесь первую роль играет С.-Петербургское Специальное конструкторское бюро измерительных систем (СКБ ИС).

СКБ ИС выпускает широкую гамму современных устройств цифровой индикации и датчиков положения. На рис. 1.18, 1.19 и 1.20, в качестве примера, приведены чертежи общих видов простейших УЦИ типа **ЛИР-510**, **ЛИР-520**

и **ЛИР-530** с минимальным набором функций. Это, соответственно, одно-, двух- и трехкоординатные УЦИ общего назначения.

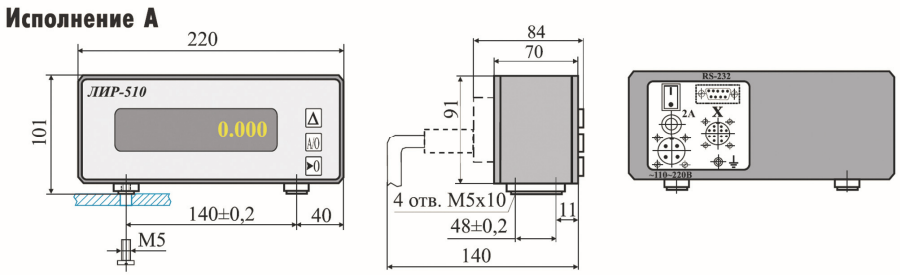


Рис. 1.18. Общий вид УЦИ ЛИР-510

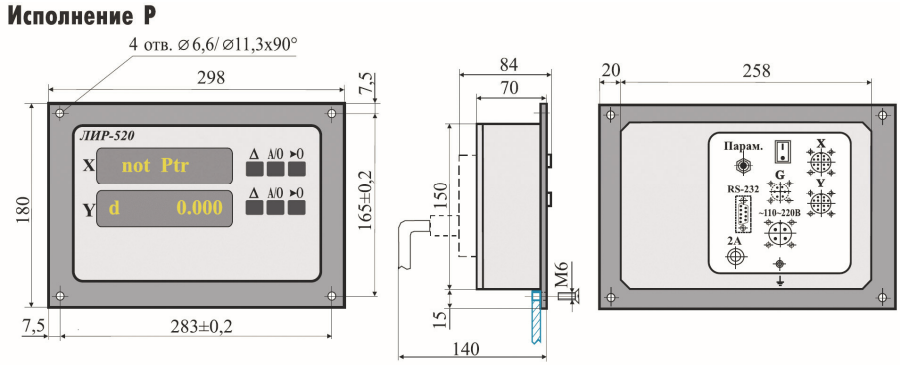


Рис. 1.19. Общий вид УЦИ ЛИР-520

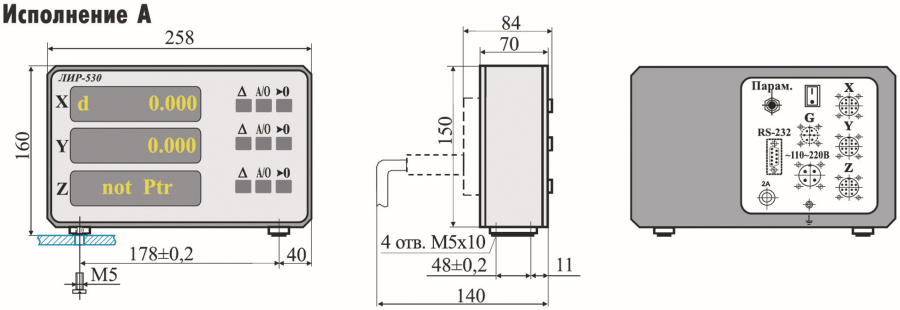


Рис. 1.20. Общий вид УЦИ ЛИР-530

Базовые функции таких УЦИ включают:

– обнуление текущей координаты в любой текущей точке;
– работа в приращениях;
– текущий контроль за перемещением координат.

Конструктивно устройства выпускаются в двух исполнениях:

– исполнение А – блочное на ножках;
– исполнение Р – встраиваемое панельное.

Устройства могут работать с преобразователями как линейных, так и угловых перемещений.

Выпускается также малогабаритное исполнение устройств ЛИР-510М, ЛИР-510МР и ЛИР-510МА.

Ниже приведена расшифровка кодирования заказа устройств.

На рис. 1.21 приведена схема подключения УЦИ английской фирмы NEWALL, использующая исключительно надежные и устойчивые к агрессивной окружающей среды шариковые датчики контроля положения.

Графические панели оператора предназначены для управления различным промышленным оборудованием и осуществления в реальном времени диагностики работы технологического процесса, причем отображение процесса может осуществляться как в статическом, так и в динамическом виде.

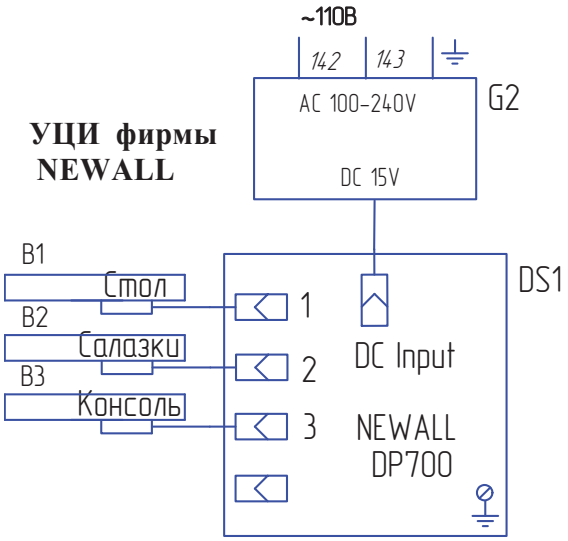


Рис. 1.21. Подключение УЦИ фирмы Newall

1.6. Программируемые панели оператора

Программируемые панели оператора широко применяются в настоящее время для управления различным автоматизированным промышленным оборудованием и универсальными металлорежущими станками. Ввод и отображение различных параметров на экране может выполняться в цифровом или графическом виде (рис. 1.22). Язык программирования панелей включает большой набор команд, позволяющих создавать удобный интерфейс между оператором и объектом управления. На нем можно отобразить любые кнопки управления, сигнальные лампы, программировать графики, осуществлять индикацию различных технических параметров, например, напряжение, ток, скорость, задавать и контролировать размерные перемещения координат, диагностировать состояние дискретных датчиков.

Различными фирмами выпускается большая номенклатура панелей оператора, различающаяся цветностью и размерами экрана, способами и протоколами связи с PLC и компьютером, объемом памяти и языком программирования. Последовательность освоения та же: разобраться с аппаратным подключением, поставить на компьютер требуемую программную среду, изучить синтаксис языка программирования, сформулировать задачу, написать программу, организовать связь панели с PLC и компьютером, загрузить в панель разработанную программу и ее отладить.

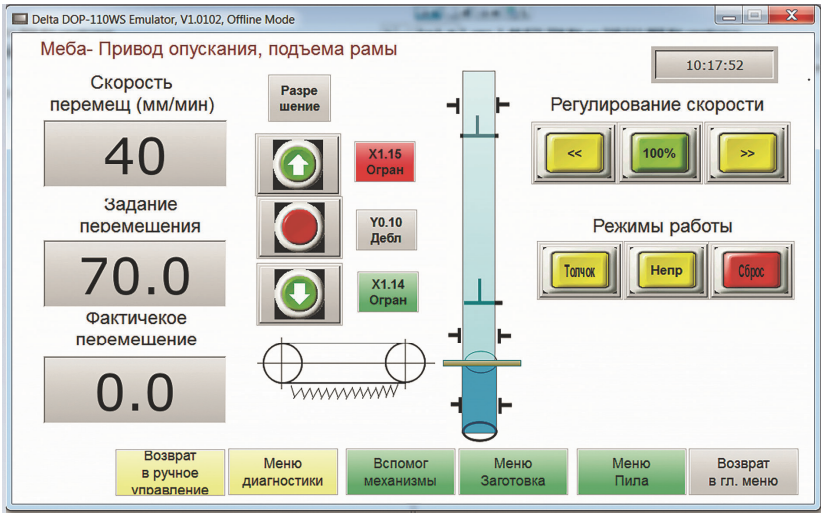


Рис. 1.22. Пример экрана панели оператора

1.7. Управление регулируемыми электроприводами от PLC

Любой современный регулируемый электропривод это сложное аппаратное и программное устройство, однако разработчику программы электроавтоматики управления им совершенно необязательно знать его внутреннюю структуру. Работа с любым типом электропривода совершенно одинакова, для этого необходимо знать следующее:

1. Система подключения электропривода всегда имеет четыре направления потока информации (см. рис. 1.23 и 1.24):
   - подача силового и вспомогательного напряжения питания;
   - подключение электродвигателя, датчика скорости, тормоза и др.;
   - подключение входных сигналов управления (деблокировка, задание скорости или положения и др.);
   - подключение выходных сигналов из привода (готовность, нулевая скорость, достижение заданной скорости и др.).
2. Любой современный электропривод, это цифровое устройство, требующее задания параметров, определяющих режимы его работы и назначение входов и выходов.
3. Любой привод требует формирования контроллером дискретного (битового) сигнала деблокировки.

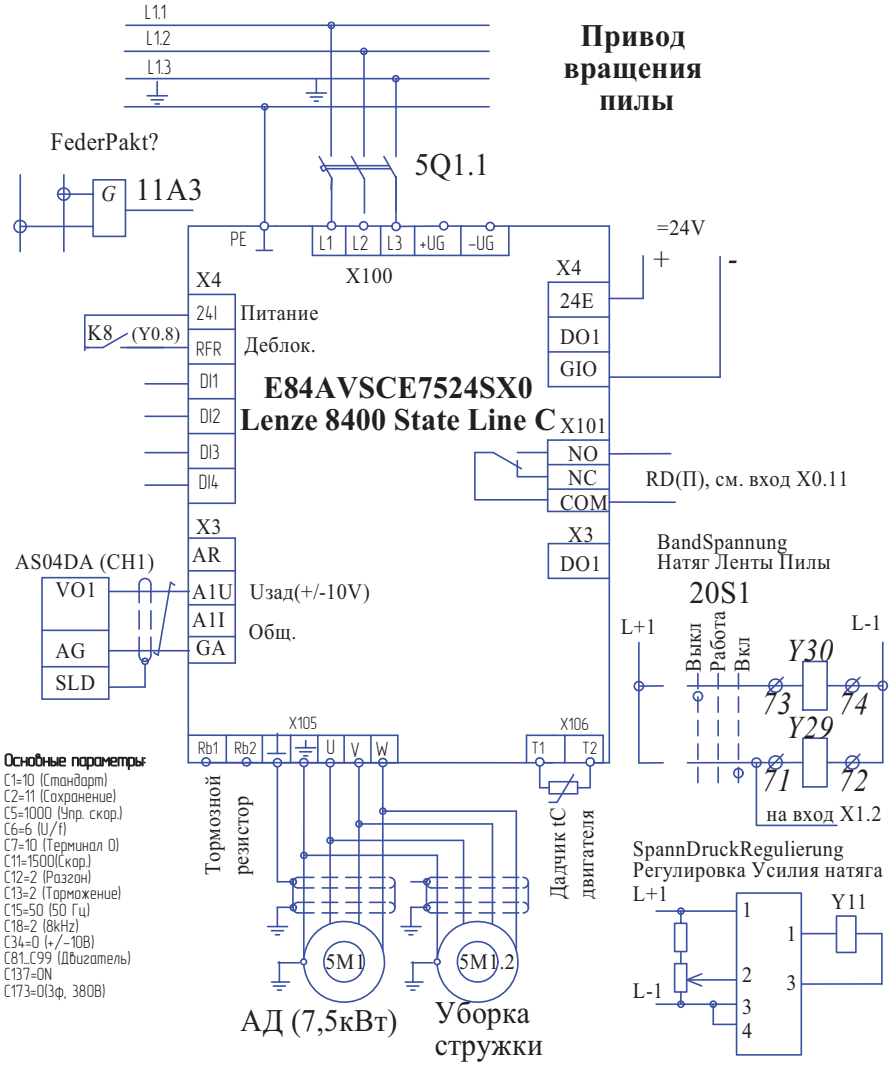


Рис. 1.23. Принципиальная схема привода Lenze 8400 (аналоговое управление от PLC)

4. Для управления только скоростью вращения двигателя достаточно сформировать аналоговое задающее напряжение $U_{цап}$. Реверс вращения, в зависимости от типа привода, осуществляется за счет изменения полярности напряжения ЦАП, или дискретными входными сигналами. Таким образом, в

структуре контроллера должен быть встроенный канал ЦАП или отдельный блок (см. рис. 1.23).

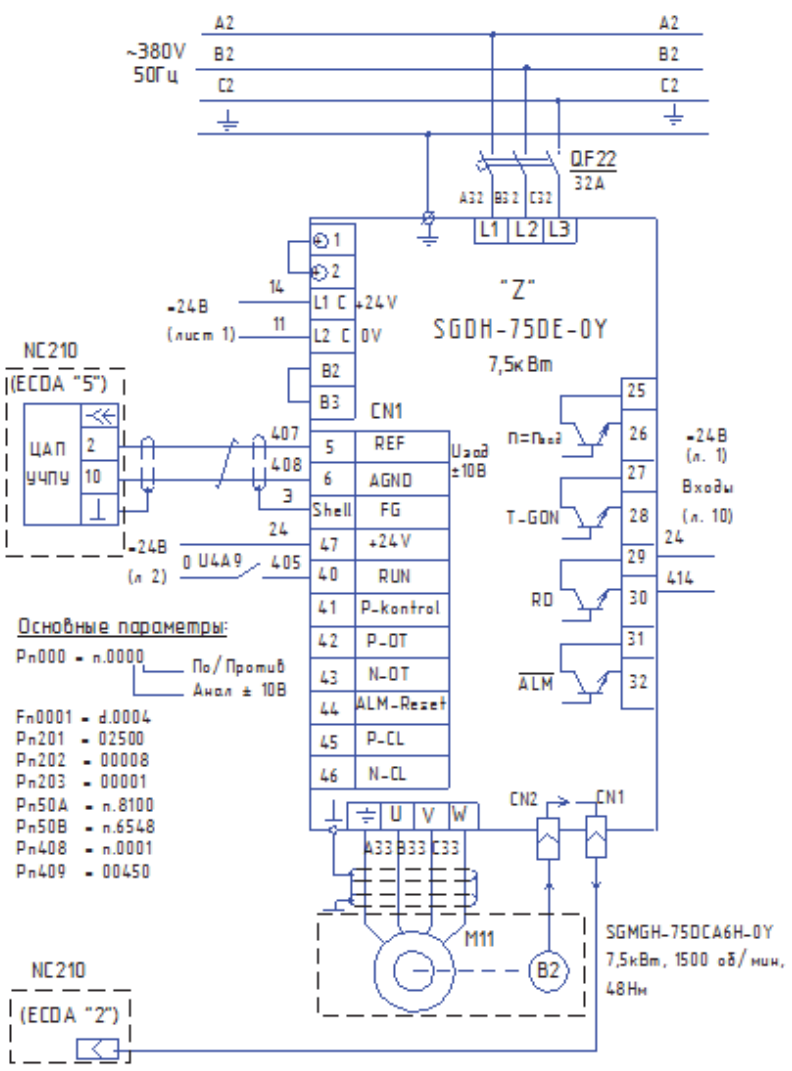


Рис. 1.24. Принципиальная схема привода SGDH фирмы Омрон

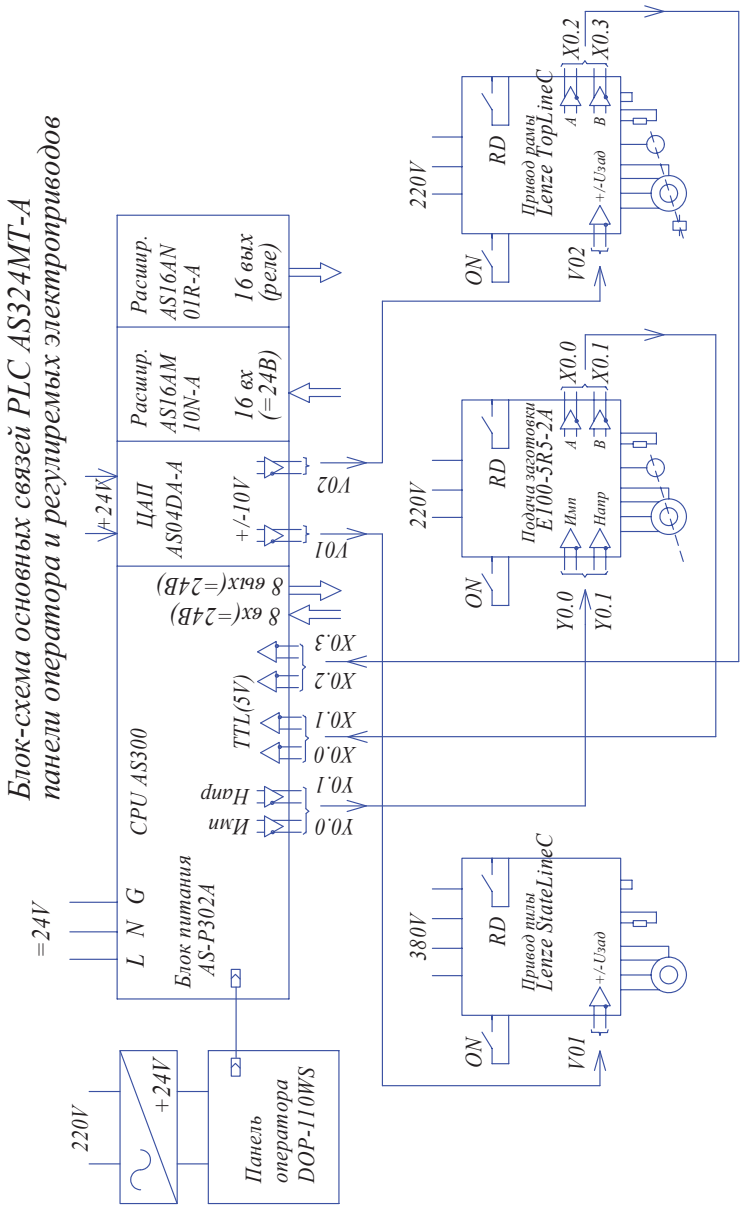


Рис. 1.25. Блок-схема конфигурации PLC и управления электроприводами

5. Для управления позиционным приводом, для управления положением, скоростью и направлением перемещения в структуре контроллера должны быть два встроенных выходных канала 5 вольтовых импульсных TTL-сигналов или отдельный блок позиционирования. Один канал, например, может служить для задания числа и частоты выдачи импульсов управления, а второй для задания направления (см. рис. 1.25).

Это все, остальное носит факультативный характер, разобравшись один раз и написав типовую программу, вы без особого труда, по аналогии, сможете управлять любым типом привода.

В заключение, в процессе проектирования, настоятельно рекомендую:

– составить блок-схему основных связей между контроллером, панелью оператора (УЦИ) и электроприводами (рис. 1.25);
– заполнять таблицу назначения адресов входных, выходных и промежуточных операндов по мере написания PLC-программы.

Это позволит избежать многих ошибок и облегчит наладку программы.

1.8. Системы счисления и способы кодирования сигналов

Кодирование сигналов служит для удобства обмена, обработки и хранения информации. Существуют следующие типы счисления и кодов:

1. *Десятичная* (Decimal). В системе используются цифры от 0 до 9, а основанием системы является число 10.

В общем случае число с различным основанием можно представить в виде:

$$Y = a_n \cdot N^m + a_{n-1} \cdot N^{m-1} + a_{n-2} \cdot N^{m-2} + \ldots + a_0 \cdot N^0,$$

где N – основание числа.

Например, для десятичного числа 4502 можно записать:

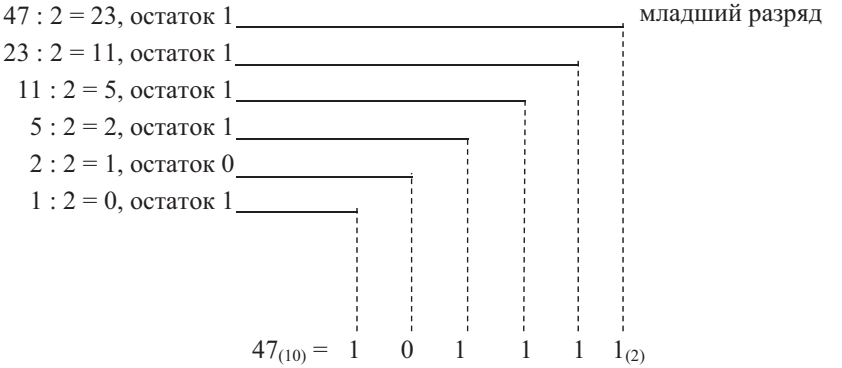
$$4502 = 4 \cdot 10^3 + 5 \cdot 10^2 + 0 \cdot 10^1 + 2 \cdot 10^0 = 4000 + 500 + 0 + 2.$$

Десятичная система проста и удобна для человека, ее мы применяем в повседневной жизни. Используя десятичную систему удобно вводить цифровую информацию в различные цифровые устройства, а также получать ответные результаты расчетов. Однако такая система неприемлема для организации самих цифровых устройств, поэтому в вычислительной технике и в электроавтоматике применяют другие системы счисления и коды.

2. *Двоичная* (Binary). Это система, в которой используются только две цифры 0 и 1. Эти числа называются *битами* (от binary digit). Физически бит 1 представляется высоким уровнем сигнала (high) или просто его наличием, а бит

0 – низким уровнем (low) или его отсутствием. Основанием системы является число 2. Все вычисления в цифровых системах осуществляются в двоичной системе.

Процедура преобразования десятичного числа в двоичное приведена ниже, на примере числа 47:



и

$$(1001111)_2 = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 32 + 0 + 8 + 4 + 2 + 1 = 47_{10}.$$

На основании двоичной системы счисления организуется *двоичный код* (B-код или BIN-код) с весами разрядов по битам 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 и т. д. 4 бита информации образуют тетраду, 8 бит – байт, 16 бит – 16-разрядное слово или просто слово, 32 бита – двойное или 32-разрядное слово.

Графически это представлено ниже.

3. *Восьмеричная* (Octal). В системе используется восемь цифр от 0 до 7, а основанием является число 8.

Процедура преобразования десятичного числа в восьмеричное приведена ниже, на примере того же числа 47:

47 : 8 = 5, остаток 7 ——— младший разряд
5 : 8 = 0, остаток 5 ———

$47_{(10)} = 5\ 7_{(8),}$

и

$$57_8 = 5 \cdot 8^1 + 7 \cdot 8^0 = 40 + 7 = 47_{10}.$$

В вычислительной технике восьмеричная запись используется для более удобного представления двоичных чисел, для чего двоичное число, начиная с младшего разряда, разбивается на группы по три бита (триады) с весами 1-2-4 и каждая триада заменяется эквивалентной восьмеричной цифрой, например:

0 1 1	1 1 1	0 0 0	1 0 0	Двоичное число 2588
3	7	0	4	Восьмеричное число

Процедура обратного преобразования очевидна, каждая десятичная цифра раздельно переводится в трехразрядный двоичный код.

4. *Шестнадцатеричная* (Hexadecimal). Система имеет основание 16 и использует 16 символов, цифры от 0 до 9 и латинские буквы A, B, C, D, E, F. При этом буква A соответствует цифре10, буква B цифре 11, буква C – 12, буква D – 13, буква E – 14 и буква F – 15.

Процедура преобразования десятичного числа 47 в шестнадцатеричное:

47 : 16 = 2, остаток 15 ——— младший разряд
2 : 16 = 0, остаток 2 ———

47(10) = 2 F(8)

и

$$2F_{16} = 2 \cdot 16^1 + F \cdot 16^0 = 2 \cdot 16^1 + 15 \cdot 16^0 = 32 + 15 = 47_{10}.$$

Шестнадцатеричная запись также используется для более удобного представления двоичных чисел, для чего двоичное число, начиная с младшего разряда, разбивается на группы по четыре бита (тетрады) с весами 1-2-4-8 и каждая тетрада заменяется эквивалентной 16-ричной цифрой, например:

0 0 00	0 1 1 1	1 1 0 0	0 1 0 0	Двоичное число 2588
0	7	C	4	16-ричное число

Процедура обратного преобразования очевидна, каждая десятичная цифра или буква раздельно переводятся в четырехразрядный двоичный код.

5. *Двоично-десятичная* (BCD – Binary Coded Decimal). BCD-система счисления предполагает замену каждой отдельно взятой десятичной цифры ее двоичным эквивалентом из *четырех* бит, например:

3	6	9	1	Десятичное число
0 0 1 1	0 1 1 0	1 0 0 1	0 0 0 1	BCD-число

Характерным отличием от шестнадцатеричной системы является то, что десятичное или двоичное число в тетраде не может быть больше 9, т. е. в каждой тетраде используются только десятичные числа от 0 до 9.

7	6	5	4	3	2	1	0	⟸ номер бита
80	40	20	10	8	4	2	1	⟸ вес битов BCD- кода

Эта система чрезвычайно удобна для построения в электроавтоматике различных кодирующих устройств и систем числовой индикации.

На ее основе в системах ЧПУ используется BCD-код с весами 1-2-4-8, 10-20-40-80, 100-200-400-800, 1000-2000 – и т. д., для выдачи различной информации, например, скорости вращения шпинделя, задания номера инструмента, кодированной выдаче технологических команд, задания кодов скорости подачи. Важно отметить, что кодирование младшей тетрады в пределах десятичных цифр 0…9 для BIN-кода и BCD-кода совпадают.

6. *Код Грея*. Это многоразрядный код, характерной особенностью которого является то обстоятельство, что при переходе от одной кодированной цифры кода к другой, в отличие от B-кода и BCD-кода, в нем изменяется *только один* разряд. Это очень важное обстоятельство, так как при изменении величины многоразрядного кода в момент его считывания, отдельные биты информации меняют свое значение не одновременно, и по этой причине может произойти ошибочное чтение кода. Этот недостаток и устраняет код Грея.

Ниже приведена сводная таблица кодирования в коде Грея для десятичных чисел 0–42.

В коде Грея обычно кодируются многопозиционные задатчики и корректоры скорости. Правила формирования кода Грея здесь не приводятся. Далее будут рассмотрены специальные инструкции преобразований кода Грея для конкретных программируемых контроллеров.

Таблица 1.3

Сводная таблица кодирования чисел в различных кодах

BIN	BIN-код								GRAY-код								Gray	H-код	
(дес)	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	(дес)	BIN	Gray
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1								1								1	1	1	1
2							1								1	1	3	2	3
3							1	1							1		4	3	2
4						1								1	1		6	4	6
5						1		1						1	1	1	7	5	7
6						1	1							1		1	5	6	5
7						1	1	1						1			4	7	4
8					1								1	1			12	8	C
9					1			1					1	1		1	13	9	D
10					1		1						1	1	1	1	15	A	F
11					1		1	1					1	1	1		14	B	E
12					1	1							1		1		10	C	A
13					1	1		1					1		1	1	11	D	B
14					1	1	1						1			1	9	E	9
15					1	1	1	1					1				8	F	8
16				1								1	1				24	10	18
17				1				1				1	1			1	25	11	19
18				1			1					1	1		1	1	27	12	1B
19				1			1	1				1	1		1		26	13	1A
20				1		1						1	1	1	1		30	14	1E
21				1		1		1				1	1	1	1	1	31	15	1F
22				1		1	1					1	1	1		1	29	16	1D
23				1		1	1	1				1	1	1			28	17	1C
24				1	1							1		1			20	18	14
25				1	1			1				1		1		1	21	19	15
26				1	1		1					1		1	1	1	23	1A	17
27				1	1		1	1				1		1	1		22	1B	16
28				1	1	1						1			1		18	1C	12
29				1	1	1		1				1			1	1	19	1D	13
30				1	1	1	1					1				1	17	1E	11
31				1	1	1	1	1				1					16	1F	10
32			1								1	1					48	20	30
33			1					1			1	1				1	49	21	31
34			1				1				1	1			1	1	51	22	33
35			1				1	1			1	1			1		50	23	32
36			1			1					1	1		1	1		54	24	36
37			1			1		1			1	1		1	1	1	55	25	37
38			1			1	1				1	1		1		1	53	26	35
39			1			1	1	1			1	1		1			52	27	34
40			1								1	1	1	1			60	28	3C
41			1					1			1	1	1	1		1	61	29	3D
42			1				1				1	1	1	1	1	1	63	2A	3F

7. *Обратный код*. Формируется путем изменения каждого бита прямого двоичного кода на противоположное значение, т. е. 1 заменяется на 0, и наоборот, 0 заменяется на 1. Например:

двоичный код числа 10 → 0000 1010;

обратный код числа 10 → 1111 0101.

8. *Дополнительный код*.

Дополнительный код *положительного числа* совпадает с его прямым кодом, например:

прямой код двоичного числа 10 → 0000 1010;

дополнительный код числа 10 → 0000 1010.

Дополнительный код *отрицательного числа* формируется по следующему правилу (в примере число 10):

1) формируется обратный код числа, т. е. 1111 0101;
2) к обратному коду прибавляется 1, т. е.

1111 0101 (обратный код) +

+ 0000 0001 (единица) =

= 1111 0110 (дополнительный код).

Дополнительные коды используются для выполнения арифметических операций с целыми числами. Для получения арифметической суммы двух чисел с учетом знака их нужно просто сложить в дополнительных кодах. Проделаем, например, разные операции с числами 8 и 3. Для этого сформируем сначала их дополнительные коды:

дополнительный код числа + 8: → 0000 1000;

дополнительный код числа – 8: → (1111 0111 + 1 = 1111 1000);

дополнительный код числа + 3: → 0000 0011;

дополнительный код числа – 3: → (1111 1100 + 1 = 1111 1101).

Внимание! Результат вычисления также получается в *дополнительном коде*.

сложение (3 + 8=11) : 0000 0011 (+3) +

+ 0000 1000 (+8) =

= 0000 1011 (+11).

сложение (−3 + 8=5) : 1111 1101 (−3) +

+ 0000 1000 (+8) =

= 0000 0101 (+5).

сложение (−8 + 3= −5) : 1111 1000 (−8) +

+ 0000 0011 (+3) =

= 1111 1011 (−5).

Действительно, обратный код числа −5 равен:

0000 0101 (прямой код 5)

1111 1010 (обратный код 5) +

+ 0000 0001 (единица) =

= 1111 1011 (дополнительный код −5).

Кроме рассмотренных выше классических для электроавтоматики кодов, при решении вычислительных задач используются также десятичные числа со знаком, с фиксированной и плавающей запятой, при необходимости, адресуем читателя к специальной литературе.

Существуют и другие специальные коды.

9. *Вещественные числа*

Выше были рассмотрены позиционные целочисленные коды и числа, наиболее часто применяемые в дискретной электроавтоматике.

Например:

– десятичный код (12345)10= $1\cdot10^4 +2\cdot10^3 +3\cdot10^2 +4\cdot10^1 +5\cdot10^0$;

– двоичный код (10110)2 = $1\cdot2^4 +0\cdot2^3 +1\cdot2^2 +1\cdot2^1 +0\cdot2^0$.

Однако при написании программ автоматизации может возникнуть необходимость, например, тригонометрических вычислений. Команды контроллеров при этом делают вычисления с вещественными числами с плавающей запятой. Для этого предусматривается специальный блок функциональных инструкций. По этой причине необходимо иметь хотя бы минимальные знания о том, как представляются и как работает процессор контроллера с такими числами.

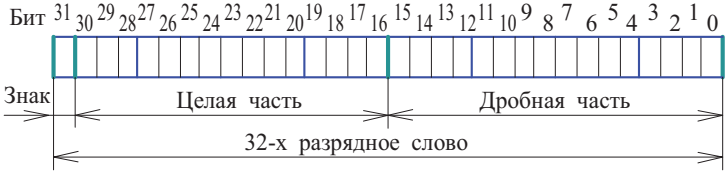
Теоретически существует несколько способов представления вещественных чисел, например:

– числа *с фиксированной запятой*, когда знак запятой «**,**» находится в фиксированной позиции. В этом случае биты рабочего слова вычислителя имеют фиксированное значение. Формат числа разбивается на три поля, например, для 32-х разрядного слова:

*) старший бит b31 – бит знака,

*) биты (b30 … b16) – хранение целой части числа,

*) биты (b15 … b0) – хранение дробной части числа



Сложение, вычитание, умножение и деление чисел с фиксированной запятой можно выполнить на компьютере так же, как с обычными целыми знаковыми числами, представив их в виде дробей с числителем и знаменателем и выполнять по классическим формулам.

Например: 1-е число А = 1,5 или в виде дроби а/с = 12/8

2-е число В = 2,5 или в виде дроби в/с = 20/8,

тогда алгоритм сложения $\left(\frac{a}{c}+\frac{в}{c}\right)=\frac{a+в}{c}=\frac{12+20}{8}=\frac{32}{8}=4$ и

алгоритм умножения $\left(\frac{a}{c}\cdot\frac{в}{c}\right)=\frac{(a\cdot в)/c}{c}=\frac{(12\cdot 20)/8}{8}=\frac{240/8}{8}=\frac{30}{8}=3,75.$

Экспоненциальная запись вещественного числа:

$$\pm M\cdot b^{E},$$

где M – мантисса;

b – база;

E – порядок.

Например, для числа 3,14 → $0,314\cdot 10^{1}$;

– *числа с плавающей запятой* (floating point)

В общем случае вещественные числа с плавающей запятой можно представлять следующем образом:

$$123,45=1234\cdot 10^{-2};$$

$$12345,6=1,23456\cdot 10^{4};$$

$$102,3456=1,023456\cdot 10^{2}=0,01023456\cdot 10^{4}$$

и выполнять вычисления, например, так

$$(12345,6+102,3456)=(1,23456\cdot 10^{4}+0,01023456\cdot 10^{4})=$$

$$=(1,23456+0,01023456)\cdot 10^{4}=$$

$$=1,24479456\cdot 10^{4}=12447,945.$$

В компьютерной технике формат представления чисел с плавающей точкой при вычислениях определяется международным стандартом **IEEE 754** (**IEC 60559**), который описывает числа:

- одинарной точности (single precision), 32-bit;
- двойной точности (double precision), 64-bit;
- двойной расширенной точности (double-extended precision), обычно 80-bit.

Все вычисления выполняются в двоичном коде. Стандарт предусматривает выполнение следующего алгоритма:

- преобразование десятичного числа в двоичное, например

$$138{,}625_{(10)} = 10001010{,}101_{(2)};$$

- нормализацию до двоичного числа с целой начальной единицей, т. е.

$$138{,}625_{(10)} = 1{,}38625 \cdot 10^{2} \text{ и}$$

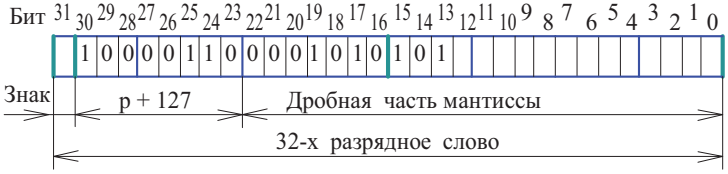
$$10001010{,}101_{(2)} = 1{,}0001010101 \cdot 2^{111};$$

- представление числа в виде

$$M \cdot 2^{p},$$

где M – мантисса (1,0001010101)
p – порядок (111)

- запись числа в слово (например, 32-разрядное) с выполнением особых условий:
  1) целая часть числа всегда равна единице (по умолчанию);
  2) знак числа указывается в старшем бите b31;
  3) порядок «p» записывается в биты (b30 … b23), причем к нему добавляется число $127_{(10)} = 0111\ 1111_{(2)}$, что позволяет хранить целые десятичные значения в диапазоне ±127;
  4) дробная часть мантиссы записывается в биты (b22 … b0), т. е. числа после запятой (,0001010101), причем начиная со старшего разряда.



Дробная часть мантиссы числа с плавающей точкой хранится в двоичной системе счисления. Поэтому числа, которые точно записываются в десятичной системе счисления, в двоичной системе можно представить только в виде бесконечной дроби, которая в памяти компьютера хранится с ограниченной точностью.

Резюме:

1. Внутренние вычисления, производимые компьютером с числами с плавающей запятой достаточно сложны и происходят автоматически при правильном программировании, т. е. соблюдении синтаксиса языка.

2. Иметь общее представления о процедуре вычисления необходимо по двум причинам:

а) необходимостью иметь базовые знания, владеть терминологией, понимать работу и назначение специальных инструкций PLC, работающих с плавающей запятой, т. е. чтобы быть инженером-профессионалом;

б) для того чтобы не удивляться совершенно непонятным цифровым значениям, появляющимся над операндами на экране дисплея при просмотре PLC-программы расчетов.

ГЛАВА 2. МЕТОДИКА ПРОЕКТИРОВАНИЯ ДИСКРЕТНОЙ ЭЛЕКТРОАВТОМАТИКИ

Электрооборудование современных промышленных механизмов, в том числе и станкостроительных отраслей, включает целый комплекс различных систем: электроприводы подачи, главного двигателя и вспомогательных механизмов; силовое электрооборудование; различного рода датчики, измерительные и регистрирующие устройства; устройства программного управления и цифровой индикации, как правило, выполненные на базе микро-ЭВМ; периферийное оборудование ввода и вывода информации; и наконец, устройства электроавтоматики, осуществляющие управление и координацию работы механизмов.

Устройства электроавтоматики на основе информации, заданной управляющим устройством более высокого уровня или оператором, и информации о фактическом состоянии механизма, получаемой от датчиков, формируют управляющие воздействия на исполнительные элементы – электроприводы, электромагнитные муфты, электромагниты управления гидравлическими золотниками и т. д.

Входные и выходные сигналы, и сами электроавтоматические устройства могут быть двух видов: *аналоговые (непрерывные) и дискретные*.

В настоящей главе рассматриваются дискретные станочные электроавтоматические устройства, наиболее широко применяемые в промышленности.

Для успешного синтеза дискретных устройств электроавтоматики необходимо решить следующие основные задачи:

1. Определить *способ графической формализации* условий работы промышленного механизма или какой-либо типовой схемы вычислительной техники. Критериями выбора являются простата и наглядность отображения реальных процессов, удобство математического анализа. Применяют диаграммы Венна (рис. 2.1) и Вейча, карты Карно, таблицы состояний, графы автоматов Милли и Мура, диаграммы и циклограммы работы, классические структурные алгоритмы и др.

2. Выбрать *математический аппарат*, позволяющий описать принятые способы графической формализации условий работы. Обычно это алгебра логики и теория дискретных автоматов.

3. Выбрать *элементную базу* и способ реализации. Применяют *жесткие и гибкие структуры* реализации электроавтоматики. Жесткие структуры выполняются в релейно-контактном или бесконтактном варианте, гибкие структуры – на программируемых логических матрицах (ПЛМ), программируемых логических контроллерах (ПЛК), фоновых контроллерах встроенных в устройство ЧПУ, на базе персональных компьютеров (ПК).

4. Выбрать *метод минимизации* и приведения полученных логических уравнений, описывающих работу синтезируемого устройства, к принятой элементной базе. Здесь применяют законы алгебры логики, методы матриц Карно, Квайна, Мак-Клакси, теории автоматов и др.

Выбор метода формализации и синтеза, как правило, субъективен, во многом зависит от первоначальных привязанностей конструктора, традиций и школы коллектива, в котором он работает, и других факторов.

Многолетний опыт автора и анализ технической литературы позволяет сделать следующие выводы. Диаграммы Венна носят чисто иллюстративный теоретический характер познаний первых шагов в мире логической алгебры; метод матриц (или карт) Карно практически неприемлем при числе логических переменных более пяти, дальнейшее его совершенствование – методы Квайна и Мак-Клакси – также приемлемы лишь при ограниченном числе логических переменных, что однозначно признается в публикациях по данной тематике [4–8]. Заметим, что реальные промышленные механизмы имеют десятки и сотни логических переменных (входных и выходных сигналов).

Во многих случаях (синтез шифраторов, дешифраторов, преобразователей кодов, сумматоров и т. д.) очень удобны таблицы состояний, ввиду простоты и наглядности.

Теория дискретных автоматов позволяет решить любую задачу, однако она очень сложна и трудоемка, требует высоких и постоянных профессиональных навыков и в практической инженерной деятельности конструкторских бюро промышленных предприятий также малопригодна.

Наиболее доступный математический аппарат – это теория *алгебры логики*.

Весьма наглядны диаграммы работы, представленные на рис. 2.1, однако они также малопригодны для непосредственного синтеза.

При использовании гибких методов реализации электроавтоматики с использованием языков программирования, допускающих структурирование, весьма эффективно применение стандартных классических алгоритмов.

Наиболее полно требованиям простоты и наглядности, возможности непосредственного использования математического аппарата алгебры логики отвечают *циклограммы*, и в наибольшей степени в начертании, используемом при описании работы средств микропроцессорной техники (см. рис. 2.1). Видно, что подобная циклограмма представляет собой развернутую во времени картину работы механизма, четко определяет время действия любого входного, промежуточного или выходного сигнала, при этом учитываются фронты сигналов, временные задержки, любые начальные условия и блокировки и т. д. Данный способ графической формализации в сочетании с использованием аппарата алгебры логики хорошо зарекомендовал себе на практике, позволил за короткое

время подготовить большое число высококвалифицированных специалистов в области станочной дискретной электроавтоматики.

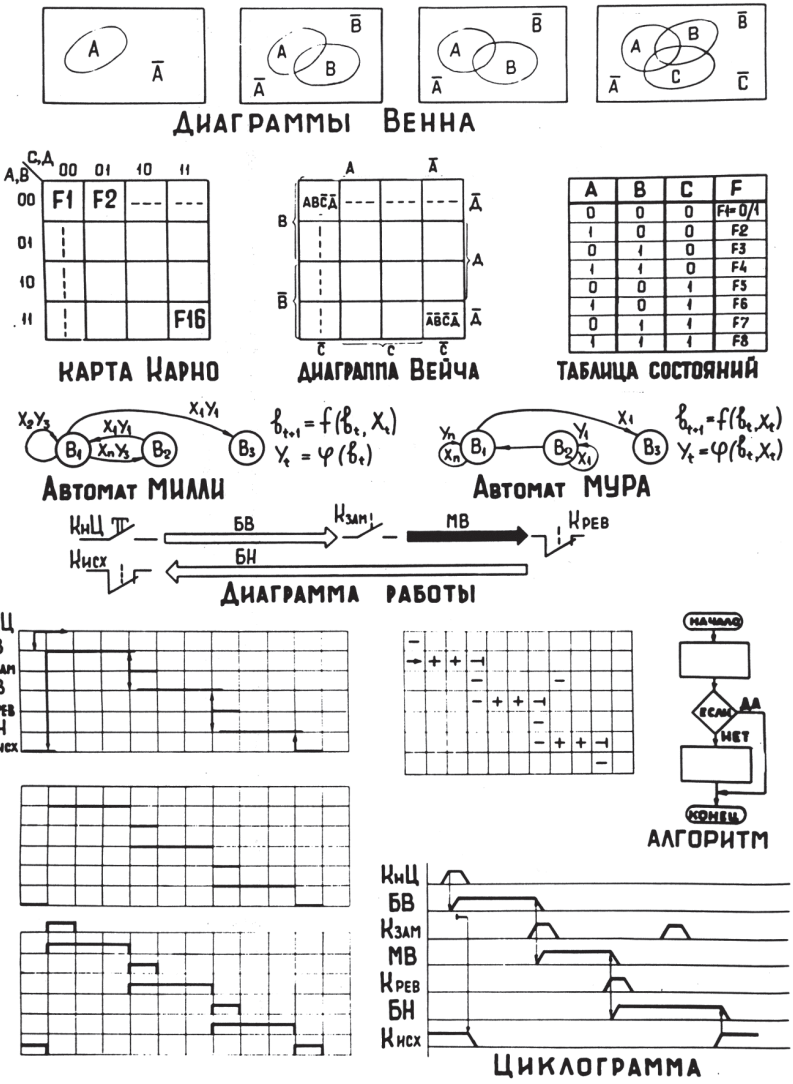


Рис. 2.1. Способы графической формализации работы электроавтоматики

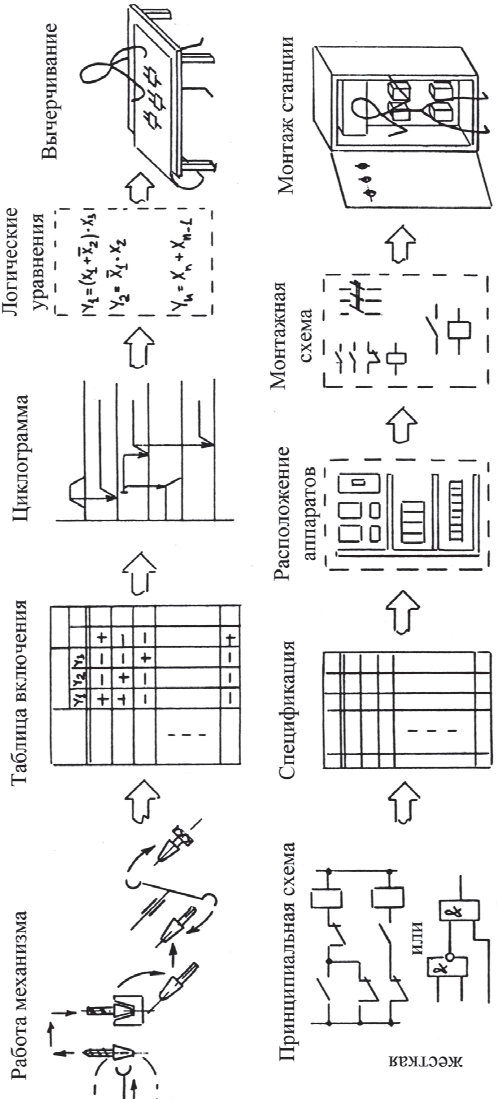


Рис. 2.2. Этапы проектирования электроавтоматики

Что касается способов реализации электроавтоматических устройств, то безусловное предпочтение следует отдать применению *гибких перепрограммируемых структур*, в большей части выполняемых на ПЛК. Гибкие структуры позволяют осуществлять оперативную настройку системы управления на разные условия работы.

На рис. 2.2 показаны основные этапы процесса проектирования электроавтоматики при использовании жесткой и гибкой структуры.

Видно, что при применении ПЛК, по сравнению с использованием жестких структур, исключается большой объем работ, связанных с выбором и спецификированием электрических аппаратов используемой элементной базы, разработкой конструкций блоков, размещением и крепежом аппаратуры в электрошкафе, составлением сложных монтажных схем, так как большая часть этой рутиной работы заменяется составлением алгоритмов и программированием. Работы, связанные с программированием, могут выполняться параллельно и независимо от процесса изготовления электрооборудования в производстве, что сокращает срок его внедрения и является одним из важнейших преимуществ ПЛК. Из этого не следует, что разработка конструкций, составление спецификаций и вычерчивание монтажных схем исключается полностью. Эти работы выполняют, однако их объем и сложность сводятся к минимуму. Кроме того, применение ПЛК обеспечивает большие функциональные и диагностические возможности, более высокие показатели надежности, сокращение сроков проектирования. Итак, ПЛК дают возможность создавать высокоэффективные устройства управления промышленными механизмами с малыми затратами на проектирование, отладку и эксплуатацию.

В заключение вводной части скажем, что метод синтеза электроавтоматических устройств на основе циклограмм и математического аппарата алгебры логики полностью пригоден при использовании в качестве элементной базы программируемых логических контроллеров.

2.1. Краткие сведения из теории алгебры логики

Логическое дискетное управляющее устройство промышленным механизмом можно представить в виде условного прямоугольника (рис. 2.3) с «n» входными и «m» выходными сигналами.

Входные сигналы X_1, $X_2 \ldots X_n$ – это сигналы, поступающие от кнопок управления, различного рода переключателей, контактных и бесконтактных путевых датчиков, датчиков температуры, давления и т. д.

Выходные сигналы Y_1, $Y_2 \ldots Y_m$ – это сигналы, управляющие исполнительными устройствами, т. е. электродвигателями, электромагнитами, электромагнитными муфтами.

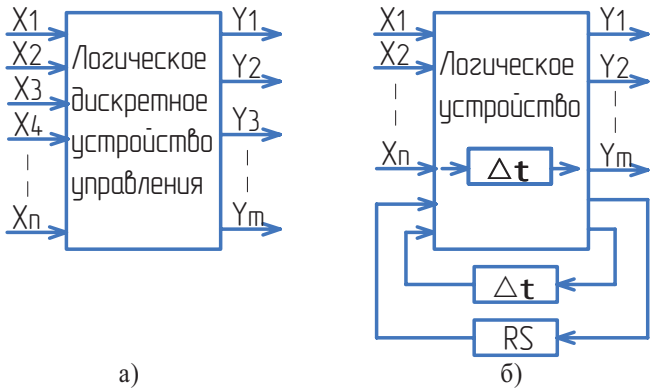


Рис. 2.3. Дискретное устройство комбинационного (а) и последовательностного (б) типа

В зависимости от типа функциональной зависимости между выходными и входными сигналами различают дискретные устройства управления двух типов:

- **комбинационные** устройства, в которых значение выходных сигналов Y_1, $Y_2 \ldots Y_m$ определяется конкретными значениями входных сигналов X_1, $X_2 \ldots X_n$ в рассматриваемый момент времени и не зависит от их состояния в предыдущие моменты времени и от последовательности появления в процессе работы. К данному классу устройств или логических схем относятся, например, шифраторы, дешифраторы, преобразователи кодов, схемы сравнения, сумматоры и т. д;
- **последовательностные** устройства, в которых значения выходных сигналов определяются не только состоянием входных сигналов на рассматриваемый момент времени, но и от их состояния на более ранние моменты времени. Для запоминания предыдущих состояний последовательностные устройства содержат элементы памяти.

Практически все схемы управления реальных промышленных механизмов являются последовательностными.

Для формального описания комбинационных логических схем применяют аппарат алгебры логики [3–8]. При определенных допущениях, процедуры синтеза комбинационных схем пригодны и для синтеза последовательностных схем, поэтому знания основ алгебры логики исключительно важны.

Аппарат алгебры логики оперирует **логическими переменными**, которые могут принимать только два значения: 0 (ноль) или 1 (единица). Применительно к дискретным схемам управления электроавтоматики значение 1 соответствует выдаче командного сигнала, наличию сигнала от датчиков контроля, замкнутому состоянию контакта реле, включенному состоянию усилителя,

высокому уровню потенциального сигнала промежуточного логического элемента вне зависимости от полярности питания и т. д. Значение 0 соответствует отсутствию перечисленных сигналов, разомкнутому состоянию контакта реле, низкому (нулевому) уровню потенциального сигнала логических элементов схемы.

Логической функцией называют зависимость выходных логических переменных от входных. Логическая функция также может принимать только два значения 0 или 1. При формальном рассмотрении законов алгебры логики логические переменные и функции обычно обозначают буквами латинского алфавита, например:

$X_1, X_2 \ldots X_n$ – входные логические переменные;

$Y_1, Y_2 \ldots Y_m$ – выходные логические переменные;

$f_1, f_2 \ldots f_n$ – логические функции.

Функционирование комбинационной логической схемы в этом случае однозначно описывается семейством логических функций:

$$Y_i = f_i(X_1, X_2 \ldots X_n), \text{ где } i = 1, 2 \ldots m.$$

Важнейшей задачей синтеза электроавтоматики являются получение этих функций в виде, удобном для анализа, минимизации и практической реализации принципиальных схем. По этой причине, при синтезе реальных схем, всем логическим переменным вместо обезличенных букв следует присваивать легко читаемые условные обозначения (аббревиатуры), например: КнП – кнопка «Пуск», ОТЖ – операция отжима, $К_{отж}$ – контроль отжатого состояния и т. д.

Логические функции в зависимости от числа входных переменных и логики работы устройства могут быть функциями одной, двух или многих переменных. Реальная схема электроавтоматики практически всегда является функциями многих переменных, однако, используя принцип суперпозиции, она может быть описана при помощи элементарных логических функций одной или двух переменных. Рассмотрим более подробно эти функции.

Логические функции одной переменной f(X) представлены в табл. 2.1. Так как входная логическая переменная X может принимать только два значения 0 или 1, то существует четыре элементарные логические функции одной переменной.

Из анализа табл. 2.1 можно сделать следующие выводы: логическое умножение любого значения функции на его инверсное состояние всегда равно нулю $\left(X \cdot \overline{X} = 0\right)$, а логическое сложение любого значения функции с его инверсным состоянием всегда равно единице $\left(X + \overline{X} = 1\right)$.

Таблица 2.1

Логические функции одной переменной f(X)

Название функции	Таблица состояний			Логическое уравнение	Описание выходного значения логической функции
	X	0	1		
Нулевая	f_1	0	0	$f_1 = 0 = X \cdot \overline{X}$	всегда равно *нулю*, каким бы не было значение входной логической переменной
Единичная	f_2	1	1	$f_2 = 1 = X + \overline{X}$	всегда равно *единице*
Повторение	f_3	0	1	$f_3 = X$	*повторяет* значения входной логической переменной
Инверсия (НЕ)	f_4	1	0	$f_4 = \overline{X}$	*инверсно значению* входной логической переменной

$\overline{}$ – знак логической инверсии

$\cdot$ – знак логического умножения

$+$ – знак логического сложения

Логические функции двух переменных f(X_1, X_2) представлены в таблице 2.2. Так как каждая входная логическая переменная X_1 и X_2 может принимать два значения, то возможны 16 различных логических функций двух переменных. Табл. 2.2 может служить первым шагом к освоению инженерного метода синтеза логических схем управления, основанного на формализации работы механизмов при помощи циклограмм.

Описание логических функций 1...6 (нулевая, единичная, повторения X_1 и X_2, инверсии X_1 и X_2) аналогично функциям одной переменной логической переменной.

Функцию И (7) называют функцией логического И, логического умножения, конъюнкцией и обозначают символами $\cdot$, $\wedge$, $\cap$, &. Запись $f_7 = X_1 \cdot X_2$ означает, что логическая функция f_7 принимает значение 1 только тогда, когда значение логической 1 одновременно принимают входные логические переменные X_1 и X_2. Это легко проследить по циклограмме 7 табл. 2.2.

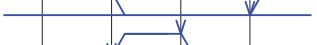
Функцию И-НЕ (8) называют инверсией логического произведения или штрихом Шеффера. Запись $f_8 = \overline{X_1 \cdot X_2}$ означает, что логическая функция f_8 принимает значение 1 во всех случаях, кроме состояния, когда обе логические переменные X_1 и X_2 равны единице (циклограмма 8 табл. 2.2).

Функция $f_9 = X_1 \cdot \overline{X_2}$ (запрет X_2) принимает значение 1, если значение X_1 равно 1 и значение X_2 равно 0 (знак инверсии над X_2) (циклограмма 9 табл. 2.2).

Функция $f_{10} = \overline{X_1} \cdot X_2$ (запрет X_1) принимает значение 1, если $X_1 = 0$ (знак инверсии над X_1) и $X_2 = 1$ (циклограмма 10 табл. 2.2).

Таблица 2.2

Логические функции двух переменных

№	Функция	Циклограмма	Логическое уравнение
	Входные сигналы		
1	Нулевая		$f1=0 = X1 \cdot \overline{X1}$
2	Единичная		$f2=1 = X1 + \overline{X1}$
3	Повторение X1		$f3=X1$
4	Инверсия X1		$f4=\overline{X1}$
5	Повторение X2		$f5=X2$
6	Инверсия X2		$f6=\overline{X2}$
7	И		$f7=X1 \cdot X2$
8	И –НЕ		$f8=\overline{X1 \cdot X2}$
9	Запрет X2		$f9=X1 \cdot \overline{X2}$
10	Запрет X1		$f10=\overline{X1} \cdot X2$
11	ИЛИ		$f11=X1+X2$
12	ИЛИ – НЕ		$f12=\overline{X1+X2}$
13	Импликация X1		$f13=X1+\overline{X2}$
14	Импликация X2		$f14=\overline{X1}+X2$
15	Равнозначность		$f15=X1 \cdot X2+\overline{X1} \cdot \overline{X2}$
16	Неравнозначность		$f16=X1 \cdot \overline{X2}+\overline{X1} \cdot X2$

Функцию ИЛИ (11) называют функцией логического ИЛИ, логического сложения, дизъюнкцией и обозначают символами +, $\vee$, $\cup$, 1.

Запись $f_{11} = X_1 + X_2$ означает, что функция $f_{11} = 1$, если $X_1 = 1$ или $X_2 = 1$.

Функцию ИЛИ-НЕ (12) называют инверсией логической суммы или стрелкой Пирса. Запись $f_{12} = \overline{X_1 + X_2}$ означает, что логическая функция f_{12} принимает значение 1, если обе логические переменные X_1 и X_2 имеют значение нуля (циклограмма 12 табл. 2.2).

Функция $f_{13} = X_1 + \overline{X}_2$ (импликация X_1) принимает значение 1, если $X_1 = 1$ или $X_2 = 0$.

Аналогично, функция $f_{14} = \overline{X}_1 + X_2$ (импликация X_2) принимает значение 1, если $X_1 = 0$ или $X_2 = 1$.

Функцию РАВНОЗНАЧНОСТЬ (15) называют также функцией эквивалентности и обозначают символом «≡» (тождество). Запись $f_{15} = (X_1X_2 + \overline{X}_1\overline{X}_2) = (X_1 \equiv X_2)$ означает, что функция f_{15} принимает значение 1 при равенстве значений входных логических переменных X_1 и X_2, т. е. когда обе они равны единице или нулю.

Функцию НЕРАВНОЗНАЧНОСТЬ (16) называют также функцией неэквивалентности, исключающим ИЛИ, сложением по модулю 2 и обозначают символом «$\oplus$» и «≠».

Запись $f_{16} = (X_1\overline{X}_2 + \overline{X}_1X_2) = (X_1 \oplus X_2) = (X_1 \neq X_2)$ означает, что функция f_{16} принимает значение 1 при неравенстве значений входных логических переменных X_1 и X_2, т. е. если $X_1 = 1$, $X_2 = 0$ или $X_1 = 0$, $X_2 = 1$.

Следует подчеркнуть, что функции равнозначность и неравнозначность в силу своего определения взаимоинверсны, т. е.

$$(X_1 \equiv X_2) = (\overline{X_1 \neq X_2}) \text{ и } (X_1 \neq X_2) = (\overline{X_1 \equiv X_2}).$$

2.2. Основные законы алгебры логики

Как было сказано выше, логическая переменная может принимать только одно из двух возможных значений: $X = 0$, если $X \neq 1$,

или $X = 1$, если $X \neq 0$.

Таким образом, значения переменных 1 и 0 взаимно инверсны:

$$1 = \overline{0} \text{ и } 0 = \overline{1}.$$

Принимается также, что:

$$0 \cdot 0 = 0;\ 0 \cdot 1 = 0;\ 1 \cdot 1 = 1;$$
$$0 + 0 = 0;\ 0 + 1 = 1;\ 1 + 1 = 1.$$

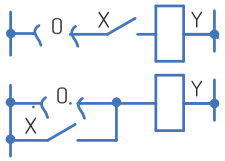
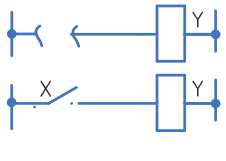
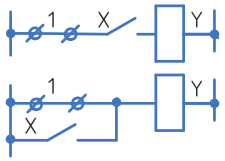
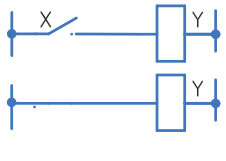
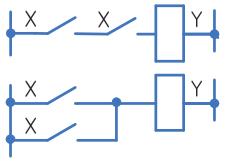
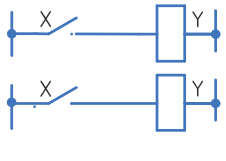
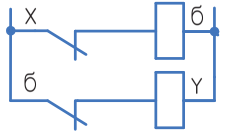
В табл. 2.3 приведены основные законы алгебры логики и их релейно-контактные эквиваленты. Для упрощения написания логических уравнений там, где это очевидно, знак логического умножения «·» можно опустить.

Знание законов алгебры логики совершенно необходимо для минимизации логических уравнений и их приведения к используемой элементной базе.

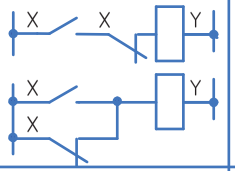
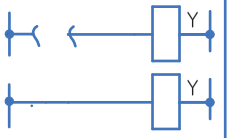
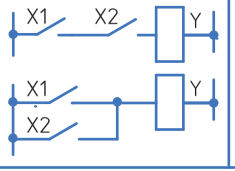
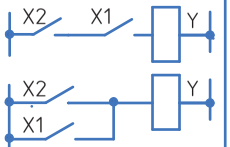
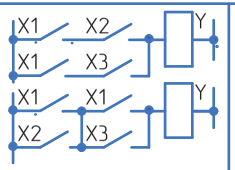
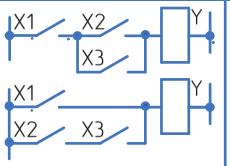
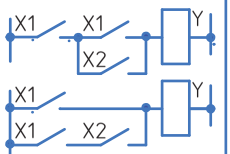
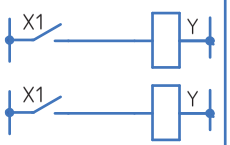
Ниже даются доказательства справедливости перечисленных в табл. 2.3 законов.

Таблица 2.3

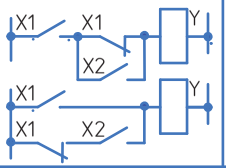
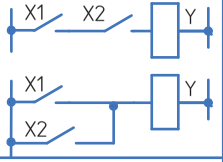
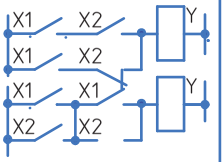
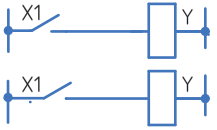
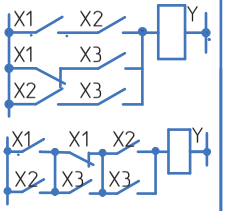
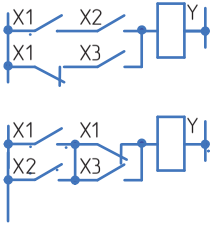
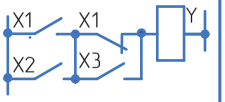
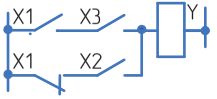
Законы алгебры логики

№		Логическое уравнение	Принципиальная схема		Формулировка закона
			Исходная	Эквивалентная	
1	2	3	4	5	6
1	Закон нулевого множества	0 • 0 = 0 0•X1•X2•. . . •Xn=0 0 + X = X			Произведение любого числа логических переменных равно нулю, если какая-либо одна переменная имеет значение ноль
2	Закон универсального множества	1 • X = X 1+X1+X2+. . . +Xn=1 1 + X = 1			Сумма любого числа логических переменных равна единице, если какая-либо одна переменная имеет значение единицы
3	Закон повторения	X • X = X X•X•X . . . •X = X X + X = X X+X+X . . . +X = X			Логические произведение или сумма двух или нескольких одинаковых переменных равны самой переменной
4	Закон двойной инверсии	$X = \overline{\overline{X}}$ (Б = $\overline{X}$; Y = $\overline{Б}$)			Двойную инверсию можно исключить

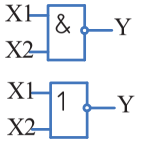
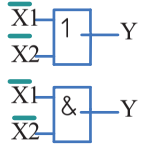
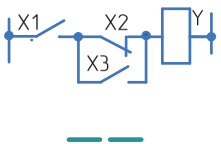
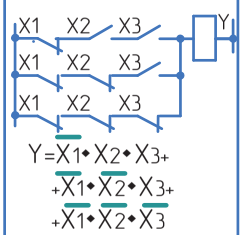
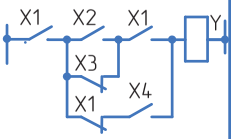
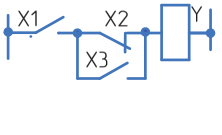
Продолжение таблицы 2.3

1	2	3	4	5	6
5	Закон дополнительности	$X \cdot \overline{X} = 0$ $X + \overline{X} = 1$			Логическое произведение или сумма любой логической переменной с ее инверсией равны нулю
6	Переместительные законы	$X1 \cdot X2 = X2 \cdot X1$ $X1 + X2 = X2 + X1$			Результаты выполнения логических операций умножения и суммы не зависят от последовательности переменных
7	Сочетательные законы	$X1(X2X3) = (X1X2)X3 = X1X2X3$ $X1+(X2+X3) = (X1+X2)+X3 = X1+X2+X3$			При логических умножении или сложении скобки можно опустить
8	Распределительные законы	$X1X2+X1X3=$ $=X1(X2+X3)$ $(X1+X2)\cdot(X1+X3)=$ $=X1+X2X3$			Разрешается вынос общих логических переменных за скобки с последующей минимизацией уравнений
9	Законы поглощения	$X1(X1+X2)=X1$ $X1(X1+X2)(X1+X3)=X1$ $X1+X1X2=X1$ $X1+X1X2+X1X3=X1$			

Продолжение таблицы 2.3

1	2	3	4	5	6
10		$X1 \cdot \overline{(X1+X2)} = X1 \cdot \overline{X2}$ $X1 + \overline{X1} \cdot X2 = X1 + X2$			При логическом сложении двух переменных любое слагаемое можно умножить на инверсное значение другой переменной
11	Законы склеивания	$X1 \cdot X2 + X1 \cdot \overline{X2} = X1$ $(X1+X2) \cdot \overline{(X1+X2)} = X1$			
12	Законы обобщенного склеивания	$X1 \cdot X2 + \overline{X1} \cdot X3 + X2 \cdot X3 = X1 \cdot X2 + \overline{X1} \cdot X3$ $(X1+X2)(\overline{X1}+X3)(X2+X3) = (X1+X2)(\overline{X1}+X3)$			
13		$(X1+X2) \cdot \overline{(X1+X3)} =$ $= X1 \cdot X3 + \overline{X1} \cdot X2$			

Окончание таблицы 2.3

1	2	3	4	5	6
14	Законы инверсии	$\overline{X1 \cdot X2 \cdot X3} = \overline{X1} + \overline{X2} + \overline{X3}$ $\overline{X1 + X2 + X3} = \overline{X1} \cdot \overline{X2} \cdot \overline{X3}$			Инверсия логического произведения есть сумма инверсий переменных Инверсия логической суммы есть произведение инверсий переменных
15	Теорема разложения	$f(X1, X2, \ldots, Xn) = X1 \cdot f(1, X2, \ldots, Xn) + \overline{X1} \cdot f(0, X2, \ldots, Xn)$ $f(X1, X2, \ldots, Xn) = \overline{X1} + f(0, X2, \ldots, Xn) \cdot (X1 + f(1, X2, \ldots, Xn))$			
16	Следствие теоремы разложения	$X1 \cdot f(X1, \overline{X1}, X2, \ldots, Xn) = X1 \cdot f(1, 0, X2, \ldots, Xn)$ $\overline{X1} \cdot f(X1, \overline{X1}, X2, \ldots, Xn) = \overline{X1} \cdot f(0, 1, X2, \ldots, Xn)$ $X1 + f(X1, \overline{X1}, X2, \ldots, Xn) = X1 + f(0, 1, X2, \ldots, Xn)$ $\overline{X1} + f(X1, \overline{X1}, X2, \ldots, Xn) = \overline{X1} + f(1, 0, X2, \ldots, Xn)$			

Законы 1…5 доказывают путем прямой подстановки значений логической переменной X, равных 1 или 0, что приводит к принятым аксиомам.

Законы 6 и 7 очевидны из сравнения исходных и эквивалентных уравнений и схем.

Последующие законы доказываются при помощи последовательных преобразований на основе аксиом и ранее доказанных законов, путем приведения обеих частей логического выражения к одному виду.

Закон 8

$$X_1(X_2 + X_3) = X_1X_2 + X_1X_3;$$
$$(X_1 + X_2)(X_1 + X_3) = X_1X_1 + X_1X_3 + X_2X_1 + X_2X_3 =$$
$$= X_1 + X_1X_3 + X_2X_1 + X_2X_3 =$$
$$= X_1(1 + X_3 + X_2) + X_2X_3 = X_1 + X_2X_3.$$

Закон 9

$$X_1(X_1 + X_2) = X_1X_1 + X_1X_2 =$$
$$= X_1 + X_1X_2 = X_1(1 + X_2) = X_1;$$
$$X_1 + X_1X_2 = X_1(1 + X_2) = X_1 =$$
$$= X_1X_3 + X_2\overline{X}_1 + X_2X_3 = X_1X_3 + \overline{X}_1X_2.$$

Закон 10

$$X_1(\overline{X}_1 + X_2) = X_1\overline{X}_1 + X_1X_2 = X_1X_2;$$
$$X_1 + \overline{X}_1X_2 = X_1(1 + X_2) + \overline{X}_1X_2 =$$
$$= X_1 + X_1X_2 + \overline{X}_1X_2 = X_1 + X_2(X_1 + \overline{X}_1) = X_1 + X_2.$$

Закон 11

$$X_1X_2 + X_1\overline{X}_2 = X_1(X_2 + \overline{X}_2) = X_1;$$
$$(X_1 + X_2)(X_1 + \overline{X}_2) = X_1X_1 + X_1\overline{X}_2 + X_2X_1 + X_2\overline{X}_2 =$$
$$= X_1 + X_1\overline{X}_2 + X_2X_1 = X_1(1 + \overline{X}_2 + X_2) = X_1$$

Закон 12

$$X_1X_2 + \overline{X}_1X_3 + X_2X_3 = X_2(X_1 + X_3) + \overline{X}_1X_3 =$$
$$= X_2(X_1 + \overline{X}_1X_3) + \overline{X}_1X_3 = X_2X_1 + X_2\overline{X}_1X_3 + \overline{X}_1X_3 =$$
$$= X_1X_2 + \overline{X}_1X_3(X_2 + 1) = X_1X_2 + \overline{X}_1X_3.$$

Закон 13

$$(X_1 + X_2)(\overline{X}_1 + X_3) =$$
$$= X_1\overline{X}_1 + X_1X_3 + X_2\overline{X}_1 + X_2X_3 =$$
$$= X_1X_3 + X_2\overline{X}_1 + X_2X_3 = X_1X_3 + \overline{X}_1X_2.$$

Закон 14 доказывают на основе перебора всех возможных значений логических переменных в соответствии с таблицей состояния.

Проинвертировав обе части логического уравнения табл. 2.3, получим:

$$X_1 \cdot X_2 \cdot \ldots \cdot X_n = \overline{\overline{X}_1 + \overline{X}_2 + \ldots + \overline{X}_n}$$

$$\text{и} \quad X_1 + X_2 + \ldots + X_n = \overline{\overline{X}_1 \cdot \overline{X}_2 \cdot \ldots \cdot \overline{X}_n}.$$

т. е. для перехода от логического произведения к логической сумме необходимо взять общую инверсию от суммы инверсий всех логических переменных. И, наоборот, для перехода от логической суммы переменных к логическому произведению необходимо взять общую инверсию от суммы инверсий всех переменных.

Законы 15 и 16 заслуживают особенного внимания. Они основаны на разложении логических функций на так называемые **конституенты** и их приведении к канонической форме. Определим понятие конституента.

Разложение единицы и нуля на конституенты

На основании известных положений, что $1 = X + \overline{X}$ и $1 \cdot 1 \cdot \ldots \cdot 1 = 1$, в общем случае можно записать:

$$\begin{aligned} 1 &= (X_1 + \overline{X}_1)(X_2 + \overline{X}_2)(X_3 + \overline{X}_3) \cdot \ldots \cdot (X_n + \overline{X}_n) = \\ &= X_1X_2X_3 \cdot \ldots \cdot X_n + \overline{X}_1X_2X_3 \cdot \ldots \cdot X_n + X_1\overline{X}_2X_3 \cdot \ldots \cdot X_n + \\ &+ \overline{X}_1\overline{X}_2X_3 \cdot \ldots \cdot X_n + \ldots + \overline{X}_1\overline{X}_2\overline{X}_3 \cdot \ldots \cdot \overline{X}_n. \end{aligned}$$

Данная запись означает, что принципиальная релейно-контактная схема, состоящая из параллельных ветвей, включавших все возможные комбинации логических произведений переменных, представляет собой постоянно закрытую цепь (рис. 2.4).

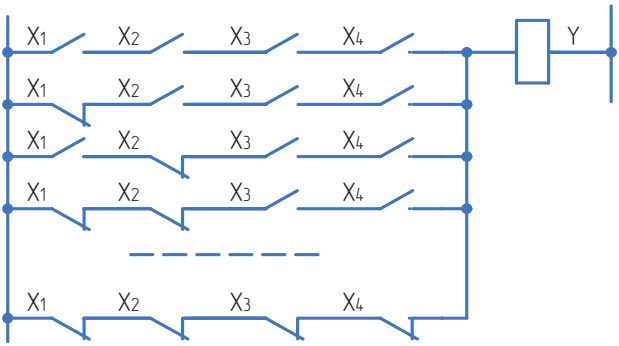


Рис. 2.4. Разложение на конституенты единицы

Члены приведенного выше выражения:

$$(X_1 \cdot X_2 \cdot \ldots \cdot X_n), \quad (\overline{X}_1 \cdot X_2 \cdot \ldots \cdot X_n), \quad \ldots, \quad (\overline{X}_1 \cdot \overline{X}_2 \cdot \ldots \cdot \overline{X}_n)$$

называют **конституентами разложения единицы**.

Аналогично, на основании, что $0 = X \cdot \overline{X}$ и 0 + 0 + 0 +…+ 0 = 0, можно записать:

$$0 = X_1\overline{X}_1 + X_2\overline{X}_2 + X_3\overline{X}_3 + \ldots + X_n\overline{X}_n =$$
$$= (X_1 + X_2 + X_3 + \ldots + X_n) \cdot (\overline{X}_1 + X_2 + X_3 + \ldots + X_n) \cdot$$
$$\cdot (X_1 + \overline{X}_2 + X_3 + \ldots + X_n) \cdot \ldots \cdot (\overline{X}_1 + \overline{X}_2 + \overline{X}_3 + \ldots + \overline{X}_n).$$

Запись означает, что принципиальная схема, состоящая из последовательно соединенных ветвей, содержащих все возможные комбинации логических сумм переменных, представляет собой постоянно разомкнутую цепь (рис. 2.5).

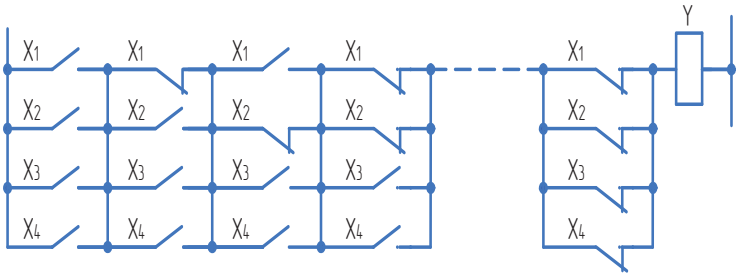


Рис. 2.5. Схема разложения на конституенты нуля

Члены приведенного выше выражения:

$$(X_1 + X_2 + \ldots + X_n),\ (\overline{X}_1 + X_2 + \ldots + X_n) \ldots (\overline{X}_1 + \overline{X}_2 + \ldots + X_n)$$

называют **конституентами разложения нуля**.

Разложение логической функции на конституенты единицы

Предположим, что из функции Y необходимо выделить некоторую входную логическую переменную X_1. В логическом уравнении может содержаться как прямое значении переменной X_1, так и инверсное $\overline{X}_1$. Вынося значения X_1 и $\overline{X}_1$ за скобки, получим:

$$Y = f(X_1, X_2, \ldots, X_n) = X_1 a + \overline{X}_1 b + c,$$

где a – переменные, находящиеся при X_1;

b – то же при $\overline{X}_1$;

c – переменные, независимые от X_1 и $\overline{X}_1$.

В дальнейших рассуждениях независимое слагаемое «c» в рассмотрение не принимаем, т. е. $f(X_1, X_2, \ldots, X_n) = X_1 a + \overline{X}_1 b$.

Это равенство справедливо при любых значениях переменной X_1:

– если $X_1 = 1$, то $\overline{X}_1 = 0$ и $f(1, X_2, X_3,\ldots,X_n) = a$;

– если $X_1 = 0$, то $\overline{X}_1 = 1$ и $f(0, X_2, X_3,\ldots,X_n) = b$, следовательно

$$f(X_1, X_2, X_3,\ldots,X_n) = X_1 \cdot f(1, X_2, X_3,\ldots,X_n) + \overline{X}_1 \cdot f(0, X_2, X_3,\ldots,X_n).$$

Если выполнить подобное преобразование по отношению ко всем входным логическим переменным, то получим:

$$\begin{aligned} f(X_1, X_2, X_3\ldots X_n) = {} & X_1X_2X_3 \cdot\ldots\cdot X_n \cdot f(1, 1, 1\ldots1) + \\ & + \overline{X}_1X_2X_3 \cdot\ldots\cdot X_n \cdot f(0, 1, 1\ldots1) + \\ & + X_1\overline{X}_2X_3 \cdot\ldots\cdot X_n \cdot f(1, 0, 1\ldots1) + \\ & + \overline{X}_1\overline{X}_2X_3 \cdot\ldots\cdot X_n \cdot f(0, 0, 1\ldots1) + \\ & + X_1X_2\overline{X}_3 \cdot\ldots\cdot X_n \cdot f(1, 1, 0\ldots1) + \\ & + \ldots + \\ & + \overline{X}_1\overline{X}_2\overline{X}_3 \cdot\ldots\cdot \overline{X}_n \cdot f(0, 0, 0\ldots0). \end{aligned}$$

Каждый из членов полученного выражения представляет собой конституент разложения единицы, логически умноженный на некоторый коэффициент F_X, принимающий значение выходной логической переменной Y, если в функции положить равным единице прямой член, соответствующий входной переменной X_i, и нулю – инверсный член.

Пример разложения логической функции $Y = X_1\overline{X}_2 + \overline{X}_1(X_2 + X_3)$ на конституенты единицы (рис. 2.6).

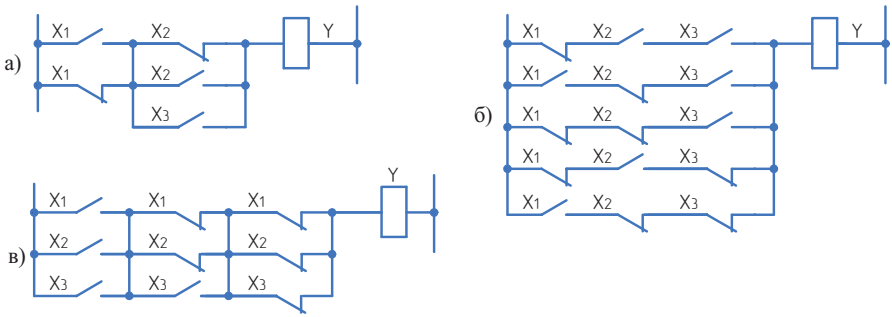


Рис. 2.6. Схемы разложения:
а – исходная; *б* – на конституенты единицы; *в* – нуля

Число конституентов для функции с тремя логическими переменными равно $2^3 = 8$. Процесс разложения на конституенты единицы показан в табл. 2.4.

Таблица 2.4

Разложение на конституенты единицы

Конституенты единицы	Двоичный эквивалент $X_1\ X_2\ X_3$	$Y = X_1\overline{X}_2 + \overline{X}_1(X_2 + X_3)$	F_X
$X_1X_2X_3$	1 1 1	$1 \cdot 0 + 0\ (1 + 1)$	0
$\overline{X}_1X_2X_3$	0 1 1	$0 \cdot 0 + 1\ (1 + 1)$	1
$X_1\overline{X}_2X_3$	1 0 1	$1 \cdot 1 + 0\ (0 + 1)$	1
$\overline{X}_1\overline{X}_2X_3$	0 0 1	$0 \cdot 1 + 1\ (0 + 1)$	1
$X_1X_2\overline{X}_3$	1 1 0	$1 \cdot 0 + 0\ (1 + 0)$	0
$\overline{X}_1X_2\overline{X}_3$	0 1 0	$0 \cdot 0 + 1\ (1 + 0)$	1
$X_1\overline{X}_2\overline{X}_3$	1 0 0	$1 \cdot 1 + 0\ (0 + 0)$	1
$\overline{X}_1\overline{X}_2\overline{X}_3$	0 0 0	$0 \cdot 1 + 1\ (0 + 0)$	0

На основании табл. 2.4 можно записать, что

$$\begin{aligned} Y = {} & X_1X_2X_3 \cdot 0 + \overline{X}_1X_2X_3 \cdot 1 + \\ & + X_1\overline{X}_2X_3 \cdot 1 + \overline{X}_1\overline{X}_2X_3 \cdot 1 + \\ & + X_1X_2\overline{X}_3 \cdot 0 + \overline{X}_1X_2\overline{X}_3 \cdot 1 + \\ & + X_1\overline{X}_2\overline{X}_3 \cdot 1 + \overline{X}_1\overline{X}_2\overline{X}_3 \cdot 0 = \\ = {} & \overline{X}_1X_2X_3 + X_1\overline{X}_2X_3 + \overline{X}_1\overline{X}_2X_3 + \overline{X}_1X_2\overline{X}_3 + X_1\overline{X}_2\overline{X}_3. \end{aligned}$$

Преобразованная принципиальная схема приведена на рис. 2.6, *б*. Такое представление функции называют **канонической суммой** или **совершенной дизъюнктивной нормальной формой** (СДНФ). Очевидно, что каждая конкретная логическая функция имеет лишь одно единственное представление такого вида.

Разложение логической функции на конституенты нуля

Аналогично выполним разложение по отношению к логической переменной X_1. Очевидно, что исходную функцию можно представить в виде

$$Y = f(X_1,\ X_2 \ldots X_n) = (X_1 + a) \cdot (\overline{X}_1 + b),$$

где a и b, соответственно, переменные находящиеся при X_1 и $\overline{X}_1$.

Равенство справедливо при любых значениях логической переменной X_1:

– если $X_1 = 1$, то $\overline{X}_1 = 0$ и $f(1,\ X_2,\ X_3, \ldots, X_n) = b$;

– если $X_1 = 0$, то $\overline{X}_1 = 1$ и $f(0, X_2, X_3,\ldots,X_n) = a$, следовательно

$$f(X_1, X_2, X_3,\ldots,X_n) = [X_1 + f(0, X_2, X_3,\ldots,X_n)]\cdot[\overline{X}_1 + f(1, X_2, X_3,\ldots,X_n)].$$

В общем виде, при разложении по отношению ко всем входным переменным, получим:

$$\begin{aligned} f(X_1, X_2, X_3,\ldots,X_n) = &\left[X_1 + X_2 + X_3 + \ldots + X_n + f(0,\ 0,\ 0\ldots0)\right]\cdot \\ &\cdot\left[\overline{X}_1 + X_2 + X_3 + \ldots + X_n + f(1,\ 0,\ 0\ldots0)\right]\cdot \\ &\cdot\left[X_1 + \overline{X}_2 + X_3 + \ldots + X_n + f(0,\ 1,\ 0\ldots0)\right]\cdot \\ &\cdot\left[\overline{X}_1 + \overline{X}_2 + X_3 + \ldots + X_n + f(1,\ 1,\ 0\ldots0)\right]\cdot \\ &\cdot\ldots\cdot \\ &\cdot\left[\overline{X}_1 + \overline{X}_1 + \overline{X}_1 + \ldots + \overline{X}_1 + f(1,\ 1,\ 1\ldots1)\right]. \end{aligned}$$

Каждый из членов полученного выражения представляет собой коэффициент разложения нуля, сложенный логически с некоторым коэффициентом F_X, принимающим значение выходной переменной Y, если в функции положить равным нулю прямой член, соответствующей входной логической переменной, и единице – инверсный член.

Пример разложения логической функции $Y = X_1\overline{X}_2 + \overline{X}_1(X_2 + X_3)$, на конституенты нуля (рис. 2.6).

Число конституентов – 8. Процесс разложения приведен в табл. 2.5, на основании которой можно записать, что

$$\begin{aligned} Y = &(X_1 + X_2 + X_3 + 0)\cdot(\overline{X}_1 + X_2 + X_3 + 1)\cdot \\ &\cdot\ (X_1 + \overline{X}_2 + X_3 + 1)\cdot(\overline{X}_1 + \overline{X}_2 + X_3 + 0)\cdot \\ &\cdot\ (X_1 + X_2 + \overline{X}_3 + 1)\cdot(\overline{X}_1 + X_2 + \overline{X}_3 + 1)\cdot \\ &\cdot\ (X_1 + \overline{X}_2 + \overline{X}_3 + 1)\cdot(\overline{X}_1 + \overline{X}_2 + \overline{X}_3 + 0) = \\ &= (X_1 + X_2 + X_3)\cdot(\overline{X}_1 + \overline{X}_2 + X_3)\cdot(\overline{X}_1 + \overline{X}_2 + \overline{X}_3). \end{aligned}$$

Таблица 2.5

Разложение на конституенты нуля

Конституенты нуля	Двоичный эквивалент $X_1\ X_2\ X_3$	$Y = X_1\overline{X}_2 + \overline{X}_1(X_2 + X_3)$	F_X
$X_1 + X_2 + X_3$	0 0 0	$0\cdot 1 + 1\ (0 + 0)$	0
$\overline{X}_1 + X_2 + X_3$	1 0 0	$1\cdot 1 + 0\ (0 + 0)$	1

Окончание таблицы 2.5

Конституенты нуля	Двоичный эквивалент $X_1\ X_2\ X_3$	$Y = X_1\overline{X}_2 + \overline{X}_1(X_2 + X_3)$	F_X
$X_1 + \overline{X}_2 + X_3$	0 1 0	$0 \cdot 0 + 1\,(1 + 0)$	1
$\overline{X}_1 + \overline{X}_2 + X_3$	1 1 0	$1 \cdot 0 + 0\,(1 + 0)$	0
$X_1 + X_2 + \overline{X}_3$	0 0 1	$0 \cdot 1 + 1\,(0 + 1)$	1
$\overline{X}_1 + X_2 + \overline{X}_3$	1 0 1	$1 \cdot 1 + 0\,(1 + 1)$	1
$X_1 + \overline{X}_2 + \overline{X}_3$	0 1 1	$0 \cdot 0 + 1\,(1 + 1)$	1
$\overline{X}_1 + \overline{X}_2 + \overline{X}_3$	1 1 1	$1 \cdot 0 + 0\,(1 + 1)$	0

Преобразованная принципиальная схема приведена на рис. 2.6, *в*. Такое представление функции называют **каноническим произведением** или **совершенной конъюнктивной нормальной формой** (СКНФ). Каждая конкретная логическая функция имеет единственное представление подобного вида.

Выводы:

1. Применение теоремы разложения для схем со многими входными логическими переменными достаточно сложно.

2. При разложении логического уравнения на конституенты единицы, т. е. приведении к СДНФ, преобразованная схема представляет собой параллельные цепочки, при замыкании любой из которых образуется замкнутая цепь, т. е. схема характеризует условия срабатывания выходной переменной.

3. При разложении на конституенты нуля, т. е. при приведении к СКНФ, преобразованная схема представляет собой последовательные комбинации логических переменных, образующих замкнутую цепь при условии одновременного равенства единице всех последовательных комбинаций, т. е. схема характеризует условия несрабатывания выходной переменной.

Из закона 16 следует:

- если прямая логическая переменная X какой-либо функции вынесена за скобки (в релейной схеме в общую цепь), то оставшиеся в скобках значения X могут быть приняты равными единице (в случае релейной схемы закорочены), а инверсные значения $\overline{X}$ приняты равными нулю (в случаи релейной схемы последовательная с $\overline{X}$ цепь удаляется);
- если инверсная логическая переменная $\overline{X}$ какой-либо функции вынесена за скобки (в общую цепь), то оставшиеся в скобках прямые значения X могут быть приравнены к нулю (удалены), а инверсные значения $\overline{X}$ приравнены единице (закорочены);

– если прямая логическая переменная X логически сложена с какой-либо сложной функцией (в случае релейной схемы включена параллельно), то все остальные прямые значения X, имеющиеся в этой функции, могут быть приравнены нулю (в случае релейной схемы последовательная с X цепь удаляется), а инверсные значения $\overline{X}$ приравнены единице (закорочены);
– если инверсная логическая переменная $\overline{X}$ логически сложена с какой-либо сложной функцией (включена параллельно), то все остальные прямые значения X могут быть приравнены единице (закорочены), а инверсные значения $\overline{X}$ приравнены нулю (удалены).

2.3. Правила формального построения принципиальных схем по уравнениям алгебры логики

В связи с тем, что синтез схем электроавтоматики с использованием аппарата алгебры логики является универсальным средством, пригодным для любой элементной базы, основные правила рассмотрим как для бесконтактных, так и для релейных схем. Последние имеют большое практическое значение, так как при использовании в качестве элементной базы программируемых логических контроллеров (ПЛК), алгоритмы программ электроавтоматики чаще всего выполняют в релейном варианте.

Правила для построения ***релейно-контактных схем****:*

1. Исходное логическое выражение преобразуется к виду, содержащему только элементарные логические функции И, ИЛИ, НЕ и минимизируется. Под знаком инверсии не должно быть более одной логической переменной, так как в противном случае ее реализация возможна только при введении дополнительных промежуточных реле.

2. Прямому значению логической переменной X логического уравнения в принципиальной схеме соответствует замыкающий контакт реле (табл. 2.6).

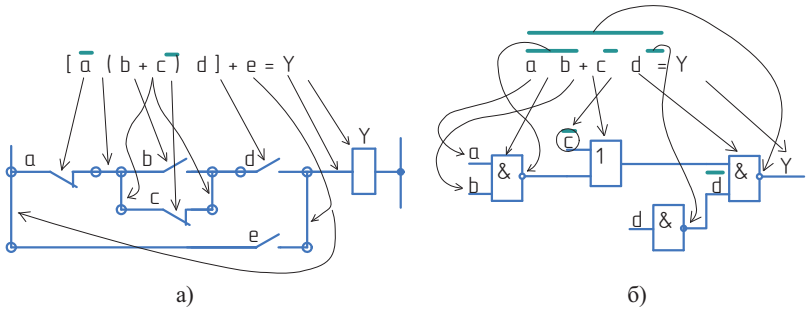
3. Инверсному значению логической переменной $\overline{X}$ логического уравнения в принципиальной схеме соответствует размыкающийся контакт реле.

4. Логическому произведению переменных в принципиальной схеме соответствует последовательное соединение контактов реле.

5. Логической сумме переменных в принципиальной схеме соответствует параллельное соединение контактов реле.

6. Знаку равенства «=» соответствует катушка выходного реле Y.

Принцип формального построения релейно-контактных схем по логическим уравнениям поясняет рис. 2.7, *а*.



а) б)

Рис. 2.7. Формальное вычерчивание релейно-контактных (а) и бесконтактных (б) схем

Рассмотрим несколько примеров:

а) $Y_1 = X_1\overline{X_2X_3} + X_4$.

Под знаком инверсии находится две логические переменные. Формально схема может быть построена с применением дополнительного промежуточного реле. Сделаем подстановку $X_2X_3 = \alpha$, тогда $Y_1 = X_1\overline{\alpha} + X_4$, что реализуется согласно изложенным выше правилам на двух выходных реле α и Y_1. Очевидно, что это решение не является оптимальным.

Выполним преобразование исходного уравнения, используя законы инверсии $Y_1 = X_1\overline{X_2X_3} + X_4 = X_1(\overline{X}_2 + \overline{X}_3) + X_4$, что реализуется на одном выходном реле.

б) $Y_2 = \overline{X_1\overline{X_2X_3}} + X_4$.

Логическое выражение также не удовлетворяет первому пункту правил, так как содержит элемент И-НЕ с двойной входимостью.

Каких-либо четких правил приведения исходного уравнения к виду И, ИЛИ, НЕ сформулировать практически невозможно. Следует действовать методом проб и ошибок. При хорошем навыке решение находится очень быстро.

Заменим знак логического умножения между X_1 и $\overline{X_2X_3}$ на знак логической суммы. Тогда

$$Y_2 = \overline{X_1\overline{X_2X_3}} + X_4 = \overline{X}_1 + X_2X_3 + X_4,$$

т. е. сразу найдено решение.

Предположим мы пошли другим путем, решив заменить знак логического умножения между переменными X_2 и X_3.

Тогда $Y_2 = \overline{X_1\overline{X_2X_3}} = \overline{X_1(\overline{X}_2 + \overline{X}_3)} + X_4$. Решения нет.

Заменим знак логического произведения между X_1 и $(\overline{X}_2 + \overline{X}_3)$.

Тогда $Y_2 = \overline{X}_1 + \overline{\overline{X}_2 + \overline{X}_3} + X_4$. Решения нет.

Заменим знак логической суммы между $\overline{X}_2$ и $\overline{X}_3$.

Тогда $Y_2 = \overline{X}_1 + X_2X_3 + X_4$, т. е. пришли к первоначальному решению.

в) $Y_3 = \overline{X_1\overline{\overline{X_2}\overline{X_3}} + X_4}$.

Заменим знак логического умножения между X_1 и $\overline{X_2X_3}$ на знак логической суммы.

Тогда $Y_3 = \overline{\overline{\overline{X}_{1}} + X_2X_3} + X_4$. Решения нет.

Заменим знак логической суммы между $\overline{\overline{X}_1 + X_2X_3}$ и X_4 на знак логического произведения.

Тогда $Y_3 = (\overline{X}_1 + X_2X_3) \cdot \overline{X}_4$. Решение найдено.

г) $Y_4 = X_1(\overline{\overline{X_2}\overline{X_3} + X_4})$.

Заменим знак логической суммы между $\overline{X_2X_3}$ и X_4 на знак логического произведения. Тогда $Y_4 = X_1 \cdot X_2X_3 \cdot \overline{X}_4$. Решение найдено.

Принципиальная схема приведена на рис. 2.8.

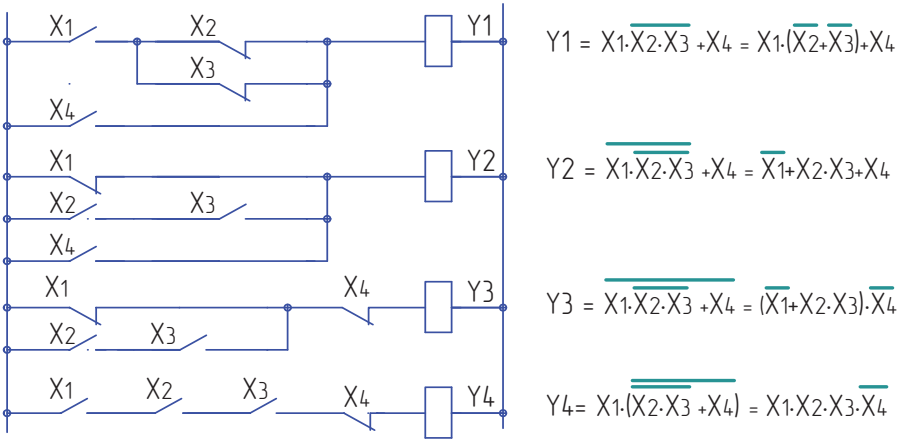
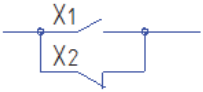
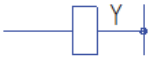
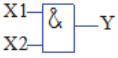
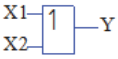
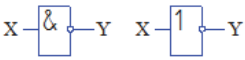
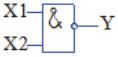
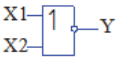
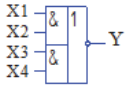
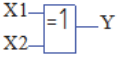


Рис. 2.8. Релейно-контактная реализация принципиальной схемы

Таблица 2.6

Элементы контактных и бесконтактных схем

Реализуемая функция	Элемент логического уравнения	РКС-эквивалент в принципиальной схеме
Прямая переменная	X	
Инверсная переменная	$\overline{X}$	
Логическое произведение	$X1 \cdot \overline{X2}$	
Логическое сложение	$X1 + \overline{X2}$	
Выходная переменная	=Y или (Y=)	

Логическая функция	Элемент логического уравнения	Эквивалент в бесконтактной схеме
И	$Y = X1 \cdot X2$	
ИЛИ	$Y = X1 + X2$	
НЕ	$Y = \overline{X}$	
И-НЕ	$Y = \overline{X1 \cdot X2}$	
ИЛИ-НЕ	$Y = \overline{X1 + X2}$	
И-ИЛИ-НЕ	$Y = \overline{X1 \cdot X2 + X3 \cdot X4}$	
Исключающее ИЛИ	$Y = X1 \neq X2$ $Y = X1 \cdot \overline{X2} + X1 \cdot \overline{X2}$	

Правила построения **бесконтактных схем**:

1) исходное логическое уравнение приводится к виду, состоящему только из элементарных логических функций, реализуемых элементами выбранной проектировщиком элементной базы, и минимизируется;
2) каждой элементарной логической функции логического уравнения, в принципиальной схеме соответствует типовой элемент, реализующий эту функцию (табл. 2.6);
3) при многократной входимости в уравнении одной элементарной логической функции в другую по логическим сумме, произведению или инверсии начертание принципиальной схемы следует начинать от последней внутренней входимости и заканчивать внешней.

Принцип формального построения бесконтактных схем по логическим уравнениям поясняет рис. 2.7, *б*.

Рассмотрим несколько примеров реализации бесконтактных схем на базе логического уравнения:

$$Y = \overline{X_1X_2 + \overline{X}_3} + \overline{X_4 + \overline{X}_5}.$$

1. На элементах типа И-НЕ.

Выполним преобразование исходного уравнения на основании закона инверсии, перейдя от логического суммирования к логическому умножению:

$$Y = \overline{X_1X_2 + \overline{X}_3} + \overline{X_4 + \overline{X}_5} = \overline{(X_1X_2 + \overline{X}_3)(X_4 + \overline{X}_5)} = \overline{\overline{X_1X_2}X_3\overline{X}_4X_5}.$$

Решение найдено, функция Y описывается только элементарными функциями И-НЕ многократно входящими друг в друга.

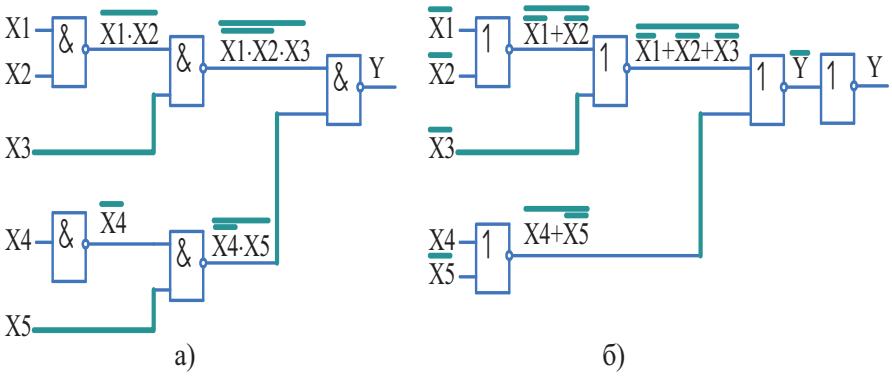


Рис. 2.9. Бесконтактная реализация принципиальной схемы на элементах И-НЕ (а) и элементах ИЛИ-НЕ (б)

2. На элементах типа ИЛИ-НЕ.

Преобразуем логическое произведение между X_1 и X_2 в логическую сумму

$$Y = \overline{\overline{\overline{X}_1 + \overline{X}_2} + \overline{X}_3} + \overline{\overline{X}_4 + \overline{X}_5}.$$

Функция Y описывается только элементарными функциями ИЛИ-НЕ многократно входящими друг в друга.

Принципиальные схемы, построенные по полученным логическим уравнениям, приведены на рис. 2.9.

В заключение раздела приведем наиболее часто применяемые практические приемы по преобразованию логических уравнений:

– переход от логической суммы к логическому произведению:

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

– переход от логического произведения к логической сумме:

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

– логическое умножение любого слагаемого логической суммы двух переменных на инверсное значение другого слагаемого:

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

– логическое умножение на единицу:

$$f(X) = f(X) \cdot 1 = f(X) \cdot (X + \overline{X});$$

– логическое суммирование с нулем:

$$f(X) = f(X) + 0 = f(X) + X \cdot \overline{X};$$

– взятие двойной инверсии:

$$f(X) = \overline{\overline{f(X)}};$$

– раскрытие скобок, например:

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

2.4. Применение законов алгебры логики для схем с вентильными элементами

При разработке и преобразовании релейно-контактных схем электроавтоматики, выполненных на аппаратуре постоянного тока, опытные проектанты часто применяют вентильные элементы – диоды. Применение диодов в релейно-контактных цепях в ряде случаев позволяет значительно упростить эти цепи, а также устранить «ложные цепи» в сложных разветвленных схемах. По этой

причине знание основных законов преобразования подобных цепей представляется весьма полезным.

Обозначим вентильный элемент, включенный в проводящем направлении, буквой В, а включенный в запирающем направлении инверсным значением $\overline{B}$.

Как показано в [3], для цепей с вентильными элементами справедливы основные соотношения и законы алгебры логики. Очевидно, что с точки зрения состояния релейно-контактной цепи вентильный элемент аналогичен контакту, включенному в эту цепь. Разница состоит в том, что состояние вентильного элемента зависит от направления его включения и всегда неизменно. В соответствии с принятым обозначением В будет обозначать всегда замкнутую цепь, а $\overline{B}$ – всегда разомкнутую цепь.

Опустив доказательства, в силу их очевидности, сведем законы алгебры логики для цепей с вентильными элементами в табл. 2.7, удобную для практического применения.

При анализе и преобразовании следует также иметь в виду, что вентильные элементы, включенные в проводящем направлении в неразветвленной цепи, можно исключить.

На рис. 2.10 показан пример устранения ложных цепей, возникающих при преобразованиях контактных цепей, при помощи введения в схему вентильных элементов.

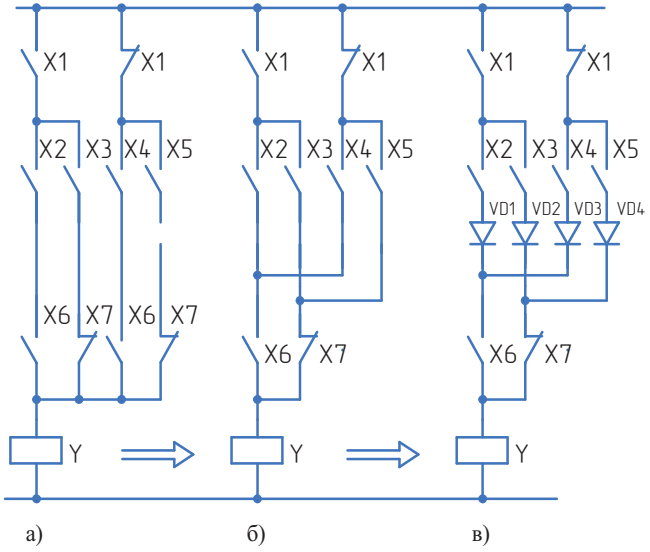
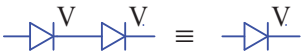
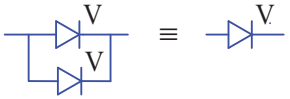
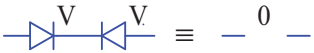
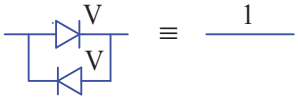
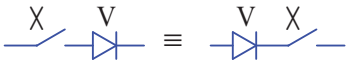
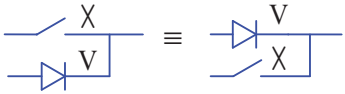
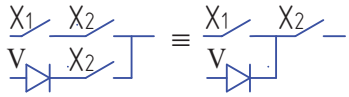
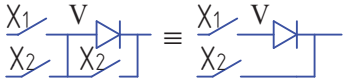
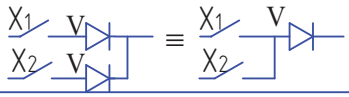
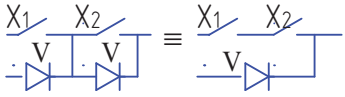


Рис. 2.10. Устранение ложных цепей при помощи вентильных элементов

Таблица 2.7

Законы алгебры логики для цепей с вентильными элементами

№	Наименование закона	Логическое уравнение	Схемные эквиваленты
1	Законы повторения	$V \cdot V = V$ $\overline{V} \cdot \overline{V} = \overline{V}$	
		$V + \overline{V} = V$ $\overline{V} + \overline{V} = \overline{V}$	
2	Законы дополнительности	$V \cdot \overline{V} = 0$	
		$V + \overline{V} = 1$	
3	Переместительные законы	$X \cdot V = V \cdot X$	
		$X + V = V + X$	
4	Сочетательные законы	$(X_1 \cdot V) \cdot X_2 = X_1 \cdot (V \cdot X_2)$ $(X_1 + V) + X_2 = X_1 + (V + X_2)$	
5	Распределительные законы	$X_1 \cdot X_2 + V \cdot X_2 =$ $= (X_1 + V) \cdot X_2$	
		$(X_1 + X_2) \cdot (V + X_2) =$ $= X_1 \cdot V + X_2$	
		$X_1 \cdot V + X_2 \cdot V =$ $= (X_1 + X_2) \cdot V$	
		$(X_1 + V) \cdot (X_2 + V) =$ $= X_1 \cdot X_2 + V$	
6	Законы инверсии	$\overline{\overline{X} + \overline{V}} = X \cdot V$ $\overline{\overline{X} \cdot \overline{V}} = X + V$	

Исходная схема (рис. 2.10, *а*) обеспечивает нормальное функционирование. Выходная логическая переменная описывается следующим уравнением:

$$Y = X_1(X_2X_6 + X_3\overline{X}_7) + \overline{X}_1(X_4X_6 + X_5\overline{X}_7) =$$
$$= X_1X_2X_6 + X_1X_3\overline{X}_7 + \overline{X}_1X_4X_6 + \overline{X}_1X_5\overline{X}_7 .$$

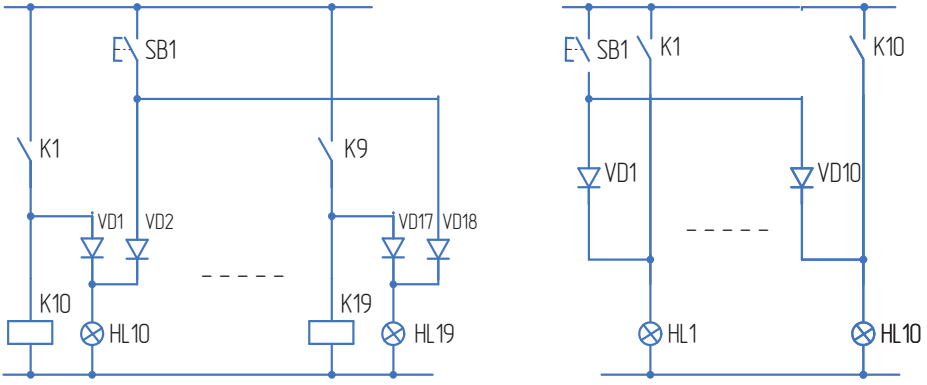
В *минимизированной схеме* (рис. 2.10, *б*) произведено объединение логических переменных X_6 и $\overline{X}_7$. Однако данная схема не будет эквивалентна исходной, так как появились новые цепи включения выходного реле:

$$Y = X_1X_2X_6 + X_1X_3\overline{X}_7 + \overline{X}_1X_4X_6 + \overline{X}_1X_5\overline{X}_7 + X_1X_2X_4X_5\overline{X}_7 +$$
$$+ X_1X_3X_5X_4X_6 + \overline{X}_1X_4X_2X_3\overline{X}_7 + \overline{X}_1X_5X_3X_2X_6 .$$

Итак, появилось четыре дополнительных ложных цепи включения реле Y.

Для *устранения ложных цепей за счет включения вентильных элементов* необходимо, чтобы произведение проводимостей элементов поперечных и продольных цепей, прилегающих к точке соединения, было равно нулю. Это достигается включением вентильных элементов в цепях, подходящих к точке соединения в одном и том же направлении. Тогда в дополнительных цепях оказываются встречно-включенные вентильные элементы, например, $X_1{\cdot}X_2{\cdot}B_1{\cdot}\overline{B}_3{\cdot}X_4{\cdot}X_5{\cdot}B_4{\cdot}\overline{X}_7$ или $X_1{\cdot}X_3{\cdot}B_2{\cdot}\overline{B}_4{\cdot}X_5{\cdot}X_4{\cdot}B_3{\cdot}X_6$ (рис. 2.10, *в*). Проводимость общей цепи становится равной нулю.

На рис. 2.11 показан пример применения вентильных элементов для исключения ложных цепей при построении цепей диагностирования исправности сигнальных ламп.



а) б)

Рис. 2.11. Использование вентильных элементов в сигнальных цепях

До настоящего времени мы имели дело с дискретными элементами двоичной логики, имеющими только **два** возможных состояния: логического нуля и логической единицы. Наиболее типичная схема выходного каскада стандартного логического элемента приведена на рис. 2.12, *а*. Управляющие напряжения U_1 и U_2 выходных транзисторов VT1 и VT2 всегда находятся в противофазе.

Если на выходе уровень логического нуля, то управляющее напряжение U_1 имеет низкий уровень, и транзистор VT1 закрыт, а управляющее напряжение U_2 имеет высокий уровень, и транзистор VT2 открыт, выходной ток через переход коллектор-эмиттер замыкается через нагрузку.

2.5. Логические элементы с тремя состояниями выхода

В состоянии логической единицы, наоборот напряжение U_1 имеет высокий уровень, и транзистор VT1 открыт, а напряжение U_2 имеет низкий уровень, и транзистор VT2 закрыт. Такой выходной каскад обеспечивает большой выходной ток при открытом нижнем транзисторе VT2 в состоянии логического нуля и малый при открытом верхнем транзисторе VT1 в состоянии логической единицы. Этому способствует наличие диода VD1 и ограничивающего резистора R1. Кроме того, при замыкании выхода на корпус элемент не выходит из строя.

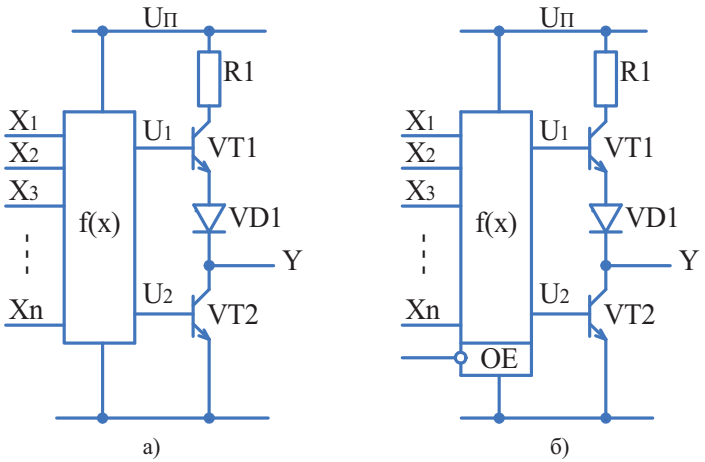


Рис. 2.12. Логический элемент с двумя (а) и тремя (б) устойчивыми состояниями

На базе рассмотренного логического элемента путем изменения его внутренней логики управления строятся логические элементы ***с тремя состояниями***

выхода (рис. 2.12, *б*). Такой элемент имеет специальный управляющий вход OE (Output Enable – разрешение выхода). При $\overline{OE}=0$ элемент работает по классической схеме, обеспечивая двухуровневый выходной сигнал 0 или 1 в зависимости от комбинации сигналов на его входах. При $\overline{OE}=1$ оба внутренних управляющих напряжения U_1 и U_2 получают низкий уровень, что приводит к закрытию обоих выходных транзисторов VT1 и VT2, и, следовательно, делает невозможным протекание выходных токов. Элемент находится в третьем, ***высокоимпедансном или Z-состоянии***, выход как бы отключен от нагрузки.

Логические элементы с тремя состояниями выхода находят большое применение в вычислительной и управляющей технике, так как значительно расширяют схемотехнические возможности элементной базы. На их основе строятся, например, различные шинные формирователи и приемопередатчики. Они очень эффективны при построении схем монтажной логики. На электрических схемах они обозначаются специальным значком в виде буквы *Z* или ромба.

2.6. Схемы монтажной логики

Логические элементы с выходным каскадом, выполненным в виде ***открытого коллектора***, позволяют реализовывать различные схемы их подключения к нагрузке в зависимости от решаемой задачи. Одним из важных их применений является реализация так называемых схем ***монтажной логики***. В этом случае открытые коллекторы выходных каскадов различных логических элементов соединяются параллельно на одну общую нагрузку (рис. 2.13).

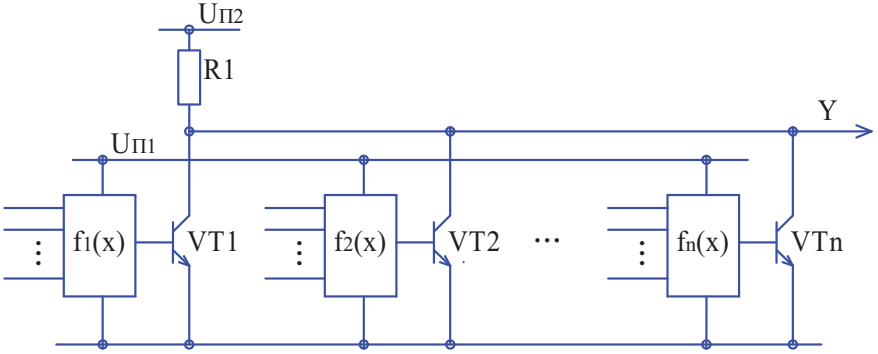


Рис. 2.13. Принцип монтажной логики

Очевидно, что выходной сигнал Y принимает значение единицы только в том случае, когда закрыты все выходные транзисторы, т. е. на всех выходах также имеются сигналы логических единиц $Y_1(X) = 1$, $Y_2(X) = 1 \ldots Y_n(X) = 1$. Если хотя бы на одном из выходов имеется сигнал логического нуля, то и выходной сигнал принимает нулевой уровень $Y = 0$.

Таким образом, такое соединение реализует функцию монтажного (или проводного) И по единицам, или функцию ИЛИ по нулям.

$$Y = Y_1(X) \cdot Y_2(\times) \cdot \ldots \cdot Y_n(X) \text{ или } \overline{Y} = \overline{Y_1(X)} + \overline{Y_2(X)} + \ldots + \overline{Y_n(X)}\ .$$

Схемы монтажной логики используются для расширения по входам, а также сокращения числа используемых логических элементов. Рассмотрим некоторые варианты применения (табл. 2.8):

– каскадирование функции ИЛИ – НЕ (схема 1 табл. 2.8):

$$Y = Y_1 \cdot Y_2 = (X_1 + X_2) \cdot (X_3 + X_4) = \overline{\overline{X_1 + X_2} + \overline{X_3 + X_4}};$$

– расширение входов функции ИЛИ – НЕ (схема 2):

$$Y = Y_1 \cdot Y_2 = \overline{X_1 + X_2} \cdot \overline{X_3 + X_4} = \overline{X_1 + X_2 + X_3 + X_4};$$

– расширение входов функции И (схема 3):

$$Y = Y_1 \cdot Y_2 = X_1 \cdot X_2 \cdot X_3 \cdot X_4;$$

– каскадирование функции И – НЕ (схема 4)**:**

$$Y = Y_1 \cdot Y_2 = \overline{X_1 \cdot X_2} \cdot \overline{X_3 \cdot X_4} = \overline{\overline{X_1 \cdot X_2} \cdot \overline{X_3 \cdot X_4}};$$

– расширение входов функции И – ИЛИ – НЕ (схема 5):

$$Y = Y_1 \cdot Y_2 = \overline{X_1X_2 + X_3X_4} \cdot \overline{X_5X_6 + X_7X_8} = \overline{X_1X_2 + X_3X_4 + X_5X_6 + X_7X_8};$$

– логическое умножение схем неравнозначности (схема 6):

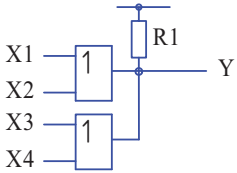
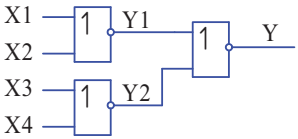
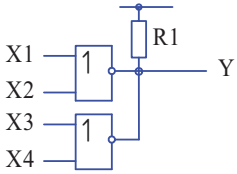
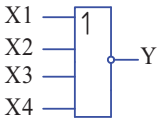
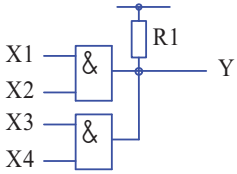
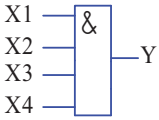
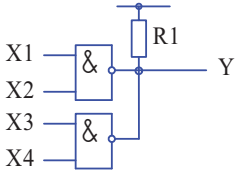
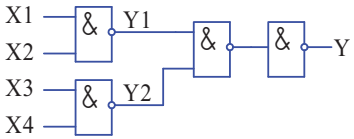
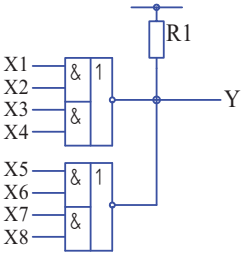
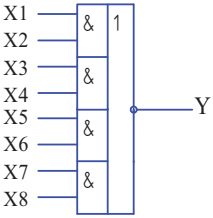
$$Y = Y_1 \cdot Y_2 = (X_1 \neq X_2) \cdot (X_3 \neq X_4) = \overline{(X_1 \equiv X_2) + (X_3 \equiv X_4)}\ .$$

В заключение отметим, что в схемах монтажной логики можно объединять логические элементы с открытым коллектором различного функционального назначения, например, как показано на рис. 7 таблицы 2.8, схему неравнозначности и логический элемент И-НЕ.

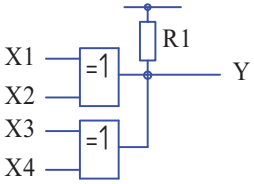
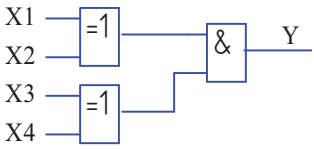
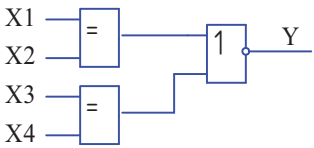
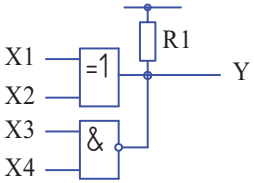
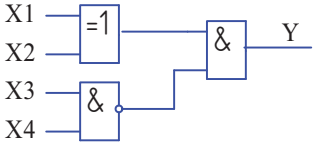
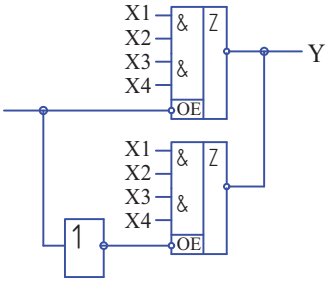
$$Y = Y_1 \cdot Y_2 = (X_1 \neq X_2) \cdot \overline{(X_1 \cdot X_2)}.$$

Таблица 2.8

Примеры монтажной логики

№	Схема монтажного ИЛИ	Логический аналог
1	2	3
1		
2		
3		
4		
5		

Окончание таблицы 2.8

№	Схема монтажного ИЛИ	Логический аналог
1	2	3
6		 
7		
8		

Подобные схемы довольно часто применяются в различных устройствах числового управления в условиях дефицита места на печатной плате и отсутствия элементов с нужным числом входов. Их существенным недостатком является сложность отыскания неисправного элемента, если на его выходе оказывается сигнал логического нуля. Поскольку выходные транзисторы микросхем соединены параллельно, отыскать неисправный элемент без демонтажа общей точки невозможно.

Эффективно применение монтажной логики для логических элементов с тремя состояниями выхода (схема 8, табл. 2.8), так как они имеют повышенную нагрузочную способность.

При $OE = 0$, $Y = Y_1 \cdot Y_2 = Y_1 \cdot Z = \overline{X_1 X_2 X_3 X_4}$;

При $OE = 1$, $Y = Y_1 \cdot Y_2 = Z \cdot Y_2 = \overline{X_5 X_6 X_7 X_8}$.

2.7. Инженерная методика синтеза схем электроавтоматики на основе циклограмм работы

Одним из важнейших факторов, определяющих выбор метода синтеза электроавтоматики, является простота и наглядность способа формализации работы промышленного механизма или типовой схемы вычислительной техники, позволяющего без особых усилий описать их работу семейством логических функций $Y_i = f_i(X_1, X_2, \ldots, X_n)$, приводимых к любой элементной базе.

Наиболее эффективным и полно отвечающим этим требованиям способом формализации являются **циклограммы**, представляющие собой развернутую в безразмерном масштабе времени последовательность командных, исполнительных и контролирующих сигналов в порядке их появления в процессе работы механизма. Преимущества циклограмм:

- простота построения и наглядность;
- возможность учета любых начальных условий и взаимных блокировок с другими схемами;
- возможность учета фронтов и временных факторов;
- простота нахождения неисправностей на любой временной момент работы механизма и реализации систем диагностики;
- простота описания циклограммы средствами математического аппарата алгебры логики;
- возможность формирования исходных данных для автоматизированного синтеза.

Очевидный недостаток – громоздкость при очень большом числе логических переменных, легко преодолевается разбитием общей циклограммы на законченные логические узлы.

Что касается математического аппарата, то речь может идти о классической **теории дискретных автоматов**, однозначно устанавливающей связь между множествами входных и выходных переменных, и внутренних состояний автомата и о математическом **аппарате алгебры логики**, однозначно описывающем только класс комбинационных логических схем.

Изначально принятая ниша применения излагаемого в книге метода – это конструкторские бюро промышленных предприятий, представляющих самую многочисленную среду проектных организации, среду, где на передний план выдвигаются требования простоты и наглядности, обеспечивающие короткие сроки проведения проектных работ и внедрения в производство.

Теория дискретных автоматов сложна и ее применение в инженерной практике станкостроения и других отраслях промышленности в большинстве случаев неоправданно. Классическая теория алгебры логики однозначно решает лишь задачи **комбинационной** логики, поэтому при разработке метода ее требуется адаптировать к синтезу **последовательностных** схем, какими являются практически все реальные электроавтоматические системы. При определенных **инженерных допущениях** эта задача легко решается.

На основании изложенного, при разработке инженерного метода синтеза станочной электроавтоматики, принимается:

- формализация при помощи циклограмм;
- применение аппарата алгебры логики, решающие задачу любой реальной сложности и на любой реальной элементной базе.

2.7.1. Формализация работы механизмов при помощи циклограмм

Основой для построения циклограммы является последовательность работы во времени командных (кнопок, тумблеров и т. д.), исполнительных (управляющих электродвигателями, электромагнитами, электромагнитными муфтами и т. д.) и контролирующих (путевых переключателей, различного рода датчиков и т. д.) элементов электроавтоматики. В процессе синтеза в исходную циклограмму, при необходимости, могут вводиться промежуточные и временные сигналы.

На **оси ординат** циклограммы показывают сигналы, поступающие от указанных элементов, в порядке их появления в процессе работы механизма с начала цикла, а на **оси абсцисс** – время в безразмерном масштабе. Факт **наличия сигнала** (единичный уровень) изображается высоким уровнем с производной амплитудой, а его **отсутствия** (нулевой уровень) – совпадает с осью времени.

Воздействие сигналов друг на друга изображают стрелкой, направленной от переднего или заднего фронта управляющего сигнала к управляемому. **Время выполнения** какой-либо операции, а также временные задержки, показывают горизонтальной линией произвольной длины, начинающейся от командного сигнала, с переходом в вертикальную стрелку в конце операций или выдержки времени.

Наклон фронтов сигналов может быть произвольным, однако для наглядности чтения циклограммы рекомендуется наклон 60°.

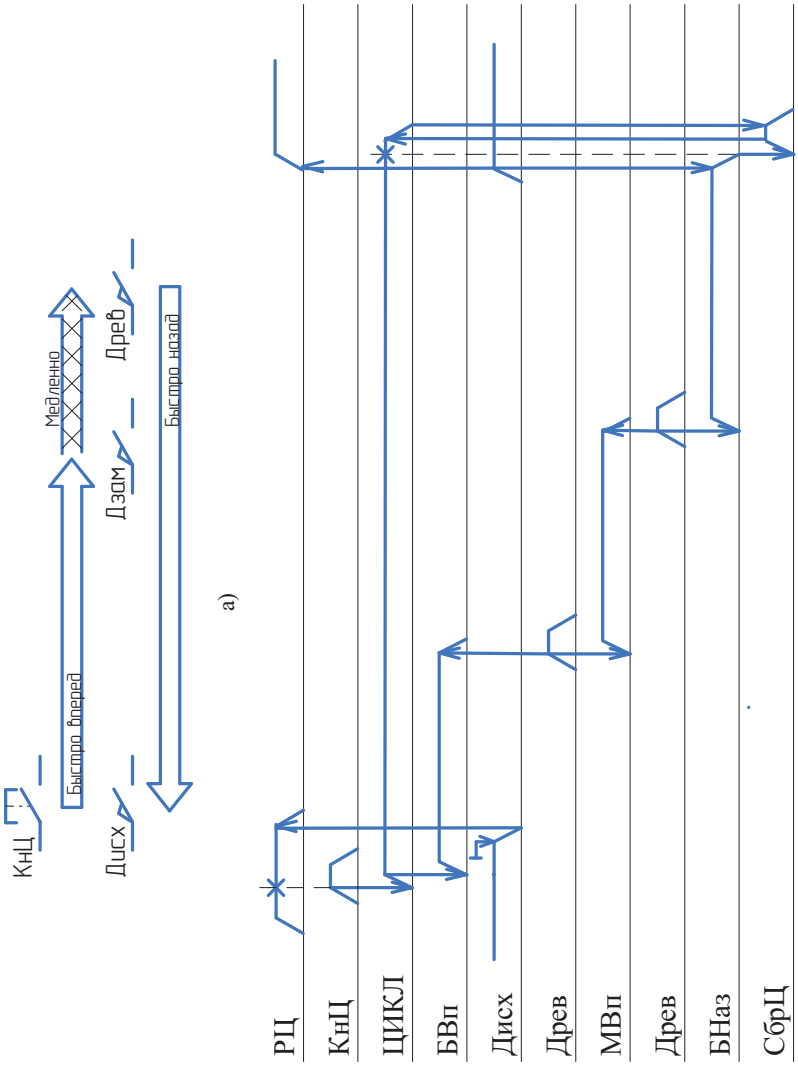


Рис. 2.14. Пример циклограммы

Пример построения циклограммы приведен на рис. 2.14. Поскольку она является основой для описания условий работы механизма (или типовой схемы вычислительной техники), то для упрощения процесса синтеза входным, промежуточным и выходным сигналам присваивают легко запоминающиеся условные обозначения – **аббревиатуры**.

Любая реальная циклограмма должна начинаться с сигнала, учитывающего различные начальные условия и блокировки, назовем его РЦ – **разрешение цикла**, и заканчиваться сигналом, определяющим окончание работы данного цикла и служащим **разрешением** для работы другого узла. Сигнал тела цикла (ЦИКЛ) может использоваться для **блокирования** других несовместимых циклов.

Принятые в циклограмме аббревиатуры:

а) командные сигналы:
КнЦ – кнопка пуска цикла (начало движения);

б) исполнительные команды:
БВп – движение быстро вперед;
МВп – движение медленно вперед;
БНаз – движение быстро назад;

в) контролирующие сигналы:
РЦ – разрешение цикла;
Дисх – контроль исходного положения механизма;
Дзам – датчик замедления;
Древ – датчик реверса;

г) промежуточные сигналы:
ЦИКЛ – сигнал тела цикла;
СбрЦ – сигнал сброса цикла.

В зависимости от конкретного механизма, его движения вперед и назад, на быстром или медленном ходу, могут осуществляться как от гидропривода путем включения соответствующих электрогидравлических золотников, так и от регулируемого электропривода путем формирования соответствующих сигналов управления. В данном случае это не имеет значения, так как показывается лишь общий подход к построению циклограммы.

Циклограмма отражает следующую последовательность работы механизма:

– нажатие кнопки КнЦ, проверка наличия разрешающего сигнала РЦ;
– формирование сигнала тела цикла ЦИКЛ. Все последующие рабочие операции осуществляются внутри тела цикла;
– формирование сигнала БВп, обеспечивающего быстрое перемещение механизма вперед. В начале движения пропадает сигнал контроля исходного положения Дисх;

– замедление движения по сигналу датчика Кзам;
– считывание сигнала датчика реверса Древ и формирование сигнала БНаз, обеспечивающего реверс на быстром ходу;
– останов движения при возврате в исходное положение;
– формирование сигнала сброса цикла СбрЦ;
– отключение сигнала тела цикла ЦИКЛ;
– автоматический сброс сигнала СбрЦ.

Специально приведем рис. 2.15, показывающий правильные и **неправильные связи** в циклограмме. Это позволит избежать ошибок при построении циклограмм.

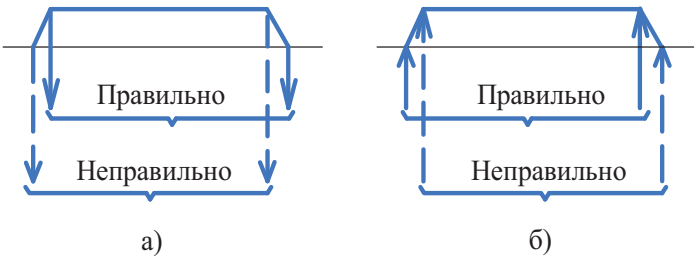


Рис. 2.15. Пример циклограммы (1)

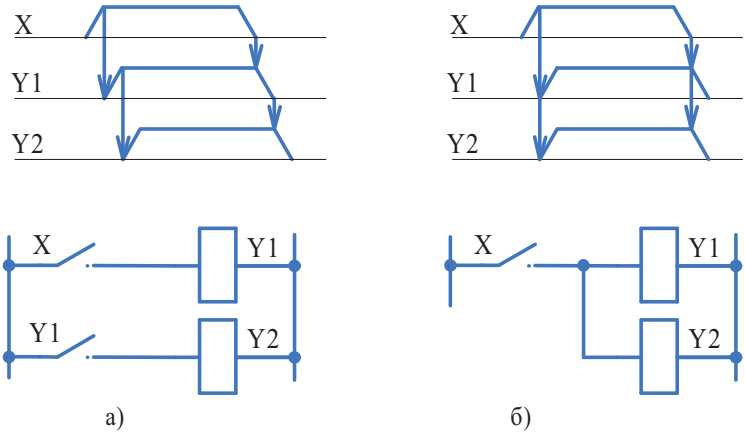


Рис. 2.16. Пример циклограммы (2)

Следует также различать функции логического повторения (рис. 2.16, *а*) и функции копирования, т. е. параллельного соединения (рис. 2.16, *б*).

Как было сказано ранее, одним из преимуществ формализации при помощи циклограмм является возможность учета влияния фронтов на работоспособность схеме уже на первом этапе синтеза. Рассмотрим этот вопрос более подробно.

2.7.2. Учет влияния фронтов при явлениях состязания в логических схемах

При рассмотрении основных законов алгебры логики было условлено, что значения всех логических переменных изменяются мгновенно и одновременно, что соответствует статическому установившемуся режиму. Только в этом случае справедливы соотношения $X \cdot \overline{X} = 0$ и $X + \overline{X} = 1$ для сигналов X, измеряемых в различных точках реальной принципиальной схемы. Очевидно, что в реальной релейно-контактной или бесконтактной схеме изменения сигналов по мере их прохождения по схеме будут проходить с задержкой, определяемой быстродействием используемой элементной базы.

В релейно-контактных схемах будет сказываться и неодновременное переключение контактов одного и того же реле. Таким образом, если в схеме управления имеется несколько параллельных путей активизации какого-либо выхода, то при их различном быстродействии на выходе возможно появление сигнала, не соответствующего статическому состоянию логических переменных, т. е. **возможно нарушение** работы схемы в соответствии с заданным алгоритмом. Обратимся к анализу характерных ситуаций (табл. 2.9).

1. *Логическое уравнение:* $Y = X \cdot \overline{X} = 0$.

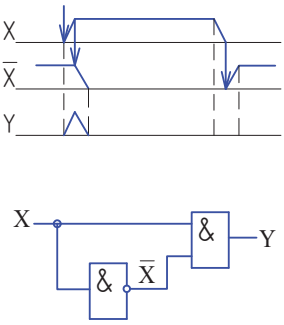
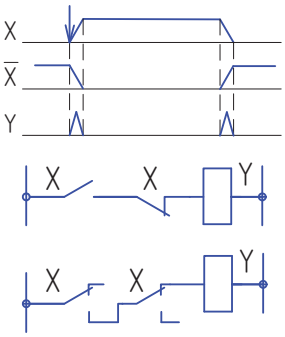
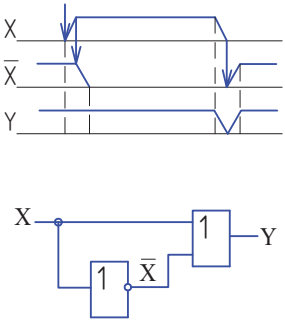
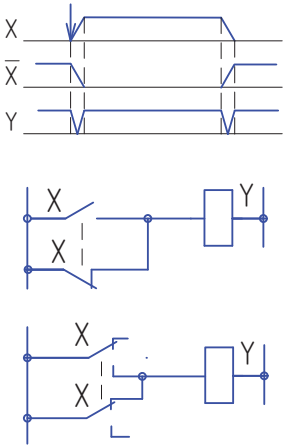
Для **бесконтактной схемы** изменения фронтов сигналов X и $\overline{X}$ происходят последовательно и переходный процесс занимает два фронта. Из циклограммы следует, что в момент изменения значения сигнала X с 0 на 1, на выходе Y из-за задержки перехода сигнала $\overline{X}$ с 1 на 0 возможно кратковременное появление сигнала Y = 1, т. е. в переходном режиме может оказаться, что $X \cdot \overline{X} \neq 0$.

Для **релейно-контактной схемы** изменения фронтов сигналов X и $\overline{X}$ происходят одновременно, так как используются контакты одного и того же реле. Процесс в этом случае занимает один фронт. Для того чтобы определить поведение схемы (а) необходимо знать конкретную ее реализацию. При использовании разных групп контактов с общим выводом (б) поведение схемы в переходных процессах непредсказуемо, так как зависит от разброса времени переключения групп контактов. Ложный импульс может возникнуть при любой коммутации сигнала X, либо не возникнуть вообще. В случае использования одной группы контактов с раздельными выводами (в) возникновение ложного

импульса принципиально невозможно. Наоборот, в случае применения специальных реле с перекрытием, формирование единичного импульса при переключении гарантировано.

Таблица 2.9

Характерные примеры явления состязания

Логическое уравнение	Бесконтактная реализация	Релейно-контактная реализация
$Y=X\cdot\overline{X}=0$		
$Y=X+\overline{X}=1$		

2. *Логическое уравнение*: $Y = X \cdot \overline{X} = 1$.

Для **бесконтактной схемы** возможно кратковременное появление нулевого импульса $Y = 0$ при переключении входного сигнала X с 1 на 0.

Для **релейно-контактной схемы,** в общем случае (а), ее поведение не предсказуемо. Для случая (б) формирование ложного импульса на выходе контактной группы гарантировано при любой коммутации реле X. Для случая (в) возможны разные ситуации, зависящие от разброса времени переключения групп контактов. В случае применения реле с перекрытием возникновенияе ложного импульса принципиально невозможно.

Таким образом, в переходном режиме может оказаться, что $X + \overline{X} \neq 1$.

Явление, когда в переходных режимах возникают ситуации типа $X \cdot \overline{X} \neq 0$ и $X + \overline{X} \neq 1$, получило название ***состязания (или риска) сигналов*** в комбинационных схемах и учет его влияния очень важен при проектировании.

Рассмотренные нами в табл. 2.9 явления в действительности гораздо сложнее и глубже, так как на них влияют число элементов в схеме; число параллельных путей формирования сигналов; различное быстродействие разных типов логических элементов; разное время срабатывания разных типов реле; количество промежуточных логических элементов или реле, используемых при формировании сигналов; разброс времени изменения сигналов на разных входах и т. д. В случае использования программируемого логического контроллера имеет значение последовательность написания программы, особенности процесса сканирования конкретного ПЛК.

Итак, в результате состязаний на выходе или в промежуточных точках принципиальной схемы, в общем случае, возможно **несанкционированное** появление:

а) одного или нескольких единичных импульсов при нулевом уровне статического состояния (единичный всплеск);
б) одного или нескольких нулевых импульсов при единичном уровне статического состояния (нулевой всплеск);
в) одного или нескольких разнополярных импульсов перед окончательным изменением уровня коммутационного сигнала с 1 на 0 или с 0 на 1 (дребезг).

Появление всплесков иногда называют **статическим** состязанием, а дребезга – **динамическим** состязанием.

Если в результате состязания нарушения алгоритма работы схемы управления не происходит, то такие состязания называют ***допустимыми или некритическими***.

Примером могут служить повторные импульсы установки S(Set) для RS-триггера во время его единичного состояния или повторные импульсы сброса R(Reset) во время его нулевого состояния.

Состязания, приводящие к нарушению алгоритма работы, называют ***недопустимыми или критическими***.

Следует сказать, что схемы, построенные с возможностью возникновения состязаний, как правило, неработоспособны, поэтому **допускать состязания нельзя**. Это нужно учитывать с первого шага синтеза.

Теория выявления условий возникновения состязаний и методов их устранения достаточно сложна и малопригодна в инженерной практике.

Учет влияния фронтов сигналов на переходный процесс, а, следовательно, и возможность избежать явления состязаний, в большинстве случаев просто и наглядно обеспечивает рассмотренный нами метод циклограмм. Рассмотрим циклограмму рис. 2.17.

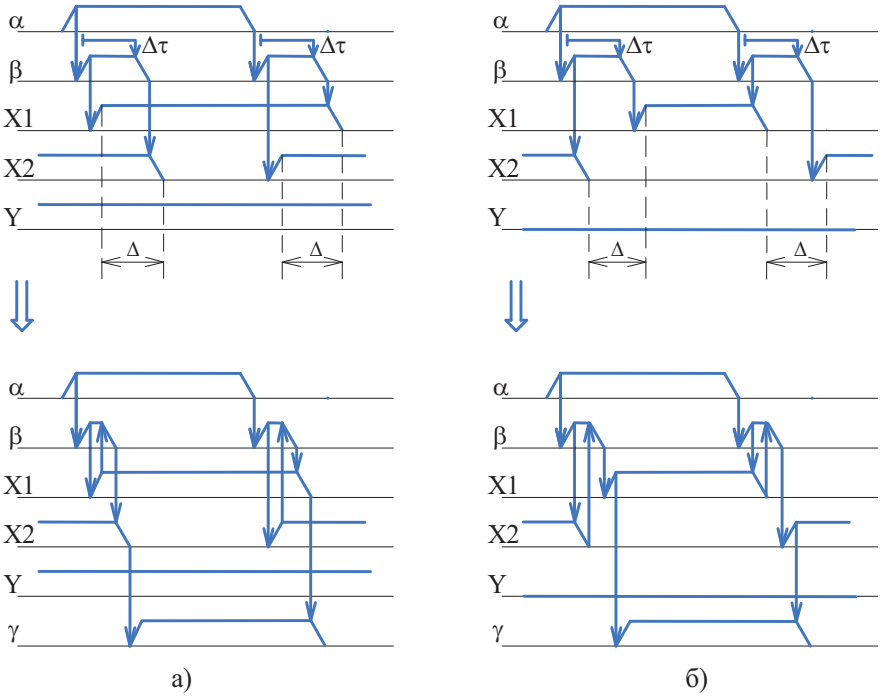


Рис. 2.17. Пример анализа циклограммы на явление состязания

Предположим, что при ее построении поставлена задача гарантированного обеспечения действия логических уравнений: $Y_1 = X_1 + X_2 = 1$ (а) и $Y_2 = X_1X_2 = 0$ (б). Управление рабочими сигналами X_1 и X_2 осуществляется некоторыми вспомогательными сигналами α и β. Это могут быть как путевые так и временные сигналы.

Поскольку в обеих циклограммах имеются зоны перекрытия Δ сигналов X_1 и X_2 либо по единицам (а), либо по нулям (б), гарантированно превышающих

по времени длительность фронтов сигналов X_1 и X_2, то поставленная задача явно обеспечивается. Ни о каких состязаниях речи быть не может.

Будем уменьшать длительность промежуточного сигнала β (за счет уменьшения выдержки времени или уменьшения длины кулачка, формирующего этот сигнал), уменьшая тем самым зону перекрытия Δ. Очевидно, что в какой-то момент мы достигнем критической точки, приводящей к состязаниям.

Рассмотрим крайний случай, когда сигнал β вообще равен нулю, а управление рабочими сигналами X_1 и X_2 осуществляется непосредственно фронтами вспомогательного сигнала α. Возможны следующие синтаксические варианты (рис. 2.18):

а) последовательное управление при включении и при выключении сигнала α, т. е. $\alpha \rightarrow X_1 \rightarrow X_2$.

Логические уравнения формирования рабочих сигналов:

$$X_1 = \alpha;\ X_2 = \overline{X_1}.$$

Из циклограмм видна вероятность нулевого всплеска в сигнале $Y_1 = X_1 + X_2$ и единичного всплеска в сигнале $Y_2 = X_1X_2$;

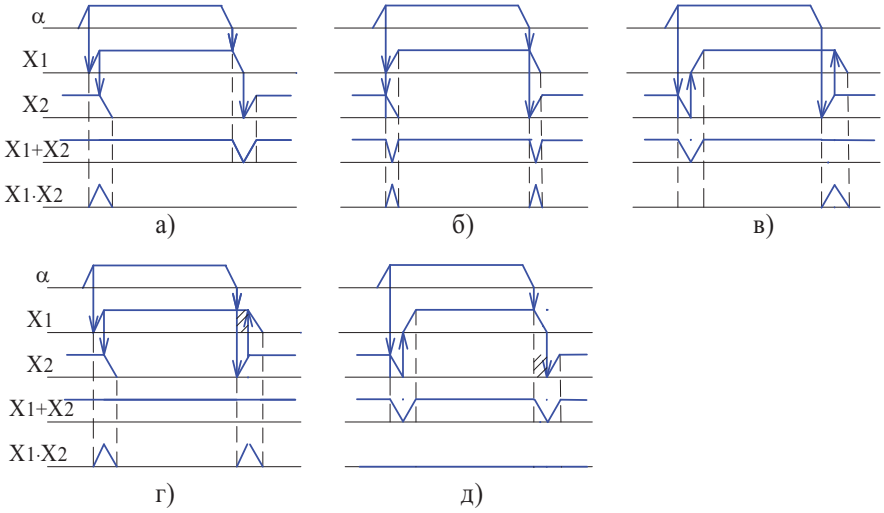


Рис. 2.18. Варианты управления фронтами

б) одновременное управление сигналами X_1 и X_2, т. е. $\alpha \rightarrow X_1, X_2$.

Логические уравнения:

$$X_1 = \alpha;\ X_2 = \overline{\alpha}.$$

Наблюдается вероятность сдвоенных всплесков в обоих выходных сигналах;

в) обратная последовательность управления, т. е. $\alpha \rightarrow X_2 \rightarrow X_1$.

Логические уравнения:

$$X_1 = \overline{X}_2; \; X_2 = \overline{\alpha}.$$

Наблюдается вероятность нулевого всплеска в сигнале $Y_1 = X_1 + X_2$ и единичного всплеска в сигнале $Y_2 = X_1X_2$;

г) прямая последовательность управления при включении, т. е. $\alpha \rightarrow X_1 \rightarrow X_2$ и обратная при включении, т. е. $\alpha \rightarrow X_2 \rightarrow X_1$.

Логические уравнения:

Для сигнала X_1 – решения нет, требуется введение промежуточного сигнала. Кажущееся решение $X_1 = \alpha\overline{X}_2$ не учитывает заштрихованной зоны. Сигнал $X_2 = \overline{X_1\alpha} = \overline{X}_1 + \overline{\alpha}$. Наблюдается вероятность двух единичных всплесков для выходного сигнала $Y_2 = X_1X_2$;

д) обратная последовательность при включении управляющего сигнала α и прямая при включении.

Логические уравнения:

$$X_1 = \overline{X}_2\alpha.$$

Для сигнала X_2 – требуется введение промежуточного сигнала. Кажущееся решение $X_2 = \overline{\alpha} + \overline{X}_1$ не учитывает заштрихованную зону. Имеется вероятность двух отрицательных всплесков для выходного сигнала $Y_1 = X_1 + X_2$.

Итак, выявление по циклограмме вероятных зон возникновения состязаний затруднений не вызывает. Для обеспечения полной гарантии, к зонам риска отнесем и случаи, когда установившийся статический режим по циклограмме сводится в точку, что показано на рис. 2.19.

Чтобы исключить зоны риска, исходную циклограмму следует преобразовать таким образом, чтобы статический режим в зонах риска циклограммы был бы не менее одного фронта рабочего сигнала (рис. 2.19, *б*).

При одинаковом быстродействии компонентов элементной базы это, как правило, гарантирует устранение явления состязания.

Следуя данному правилу, на рис. 2.18, *в* и *г* показано преобразование исходных циклограмм *а* и *б*, выполненное путем замены временного промежуточного сигнала, сигналом β, управляемым фронтами: (α – включение, X_1, X_2 – выключение).

К синтезу преобразованных циклограмм вернемся после изучения основ инженерной методики.

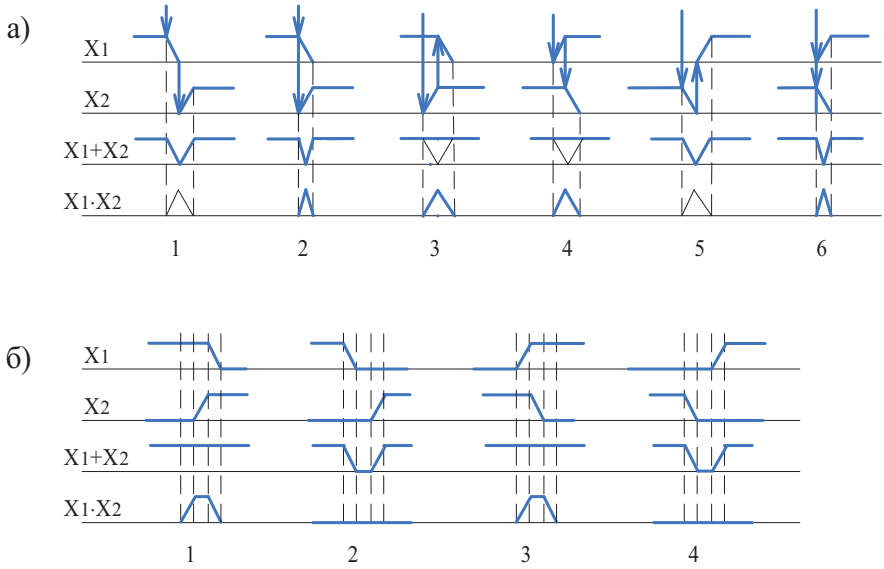


Рис. 2.19. Точечные зоны риска (а) и их устранение (б)

Во многих случаях циклограмма позволяет избежать вероятностей возникновения состязаний на этапе синтеза, ввиду очевидности, что затруднительно при использовании только законов алгебры логики.

Рассмотрим *следующий пример*. Пусть логическая функция при словесном описании работы механизма определяется следующим выражением $Y = aX + b\overline{X}$. Причем допускается одновременное равенство единице переменных а и б. В этом случае в статическом режиме $Y = 1X + 1\overline{X} = X + \overline{X} = 1$. Однако в динамическом режиме при переключении контактов реле X возможен нулевой всплеск (рис. 2.20).

Его можно ликвидировать, преобразовав уравнение в соответствии с законом обобщенного склеивания $Y = aX + b\overline{X} + ab$ (рис. 2.20, *б*), однако прийти к этому решению не так-то просто.

Если же синтез выполнялся по циклограмме (рис. 2.20, *в*), то запись уравнения $Y = aX + b\overline{X} + ab$ однозначно вытекает из ее анализа.

Перейдем к изложению основных материалов методики.

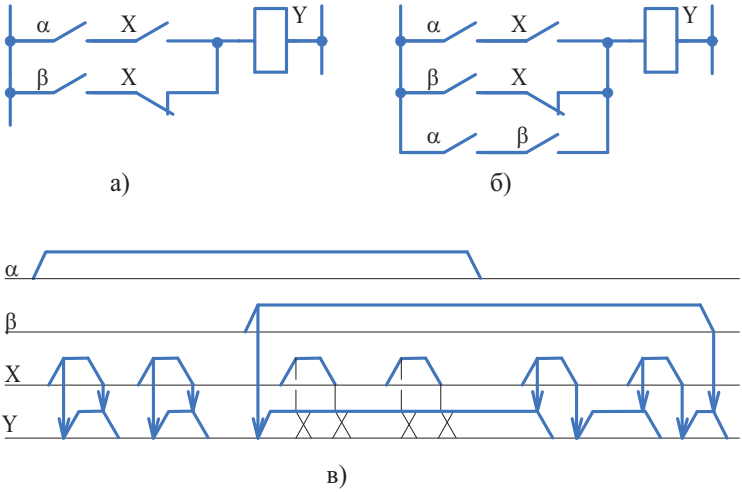


Рис. 2.20. Пример учета влияния фронтов при анализе циклограммы

2.8. Рекомендуемая последовательность синтеза

1. Изучить принцип работы механизма, для которого необходимо разработать принципиальную схему или алгоритм работы ПЛК. Вычертить характерные упрощенные состояния механизма, показывающие его основные этапы работы и состояния датчиков контроля.

2. Составить таблицы исполнительных и контролирующих элементов с указанием их состояний по этапам работы механизма. Логическим переменным следует присвоить легко читаемые условные обозначения.

3. Составить исходную циклограмму работы механизма. Входные и выходные логические переменные по оси ординат рекомендуется вычерчивать в порядке их появления в процессе выполнения цикла работы. Циклограмма должна начинаться с блокировочных сигналов от других схем и содержать ответные сигналы о процессе действия данного цикла или его окончании для передачи в другие связанные с ним схемы.

4. Проанализировать циклограмму и описать выходные переменные уравнениями алгебры логики. Алгоритм анализа будет рассмотрен далее.

5. Привести полученные логические уравнения к виду используемой элементной базы и минимизировать.

6. Вычертить принципиальную схему и составить спецификацию используемых элементов.

Прежде чем рассмотреть проблему синтеза принципиальных схем в комплексе – от изучения принципа работы механизма до формального вычерчивания схемы, рассмотрим методы поэтапного анализа циклограмм.

2.9. Обобщенный алгоритм анализа циклограмм

Анализ циклограмм следует выполнять в соответствии со следующим обобщенным алгоритмом (рис. 2.21):

1. Поиск комбинационного решения, когда синтезируемая выходная логическая переменная однозначно по циклограмме описывается уравнением алгебры логики. Если решения нет, то перейти к пункту 2.
2. Применение типового элемента памяти с синтезом по циклограмме только сигналов установки и сброса этой памяти. Если решения нет, то перейти к пункту 3.
3. Введение в исходную циклограмму одного или нескольких промежуточных сигналов, исключающих неоднозначность решения задачи, и переход к пункту 1 алгоритма.
4. Учет начальных условий и блокировок.

Рассмотрим подробно каждый из этапов синтеза.

2.9.1. Методы поиска комбинационного решения

Алгоритм поиска по циклограмме рис. 2.21 комбинационного решения показан на рис. 2.22.

Результатом поиска должно быть определение по циклограмме логической комбинации входных, промежуточных и, при необходимости, выходных сигналов, однозначно определяющих конкретный синтезируемый выходной или вновь вводимый промежуточный сигнал, т. е. логической комбинации, имеющей место ***только*** во время действия синтезируемого сигнала. Обязательно проведение проверок на отсутствие слева и справа от синтезируемого сигнала комбинации других сигналов, описываемых тем же логическим уравнением. Такое решение обеспечивает заданный циклограммой алгоритм работы при условии правильного функционирования всех аппаратов управления и контроля.

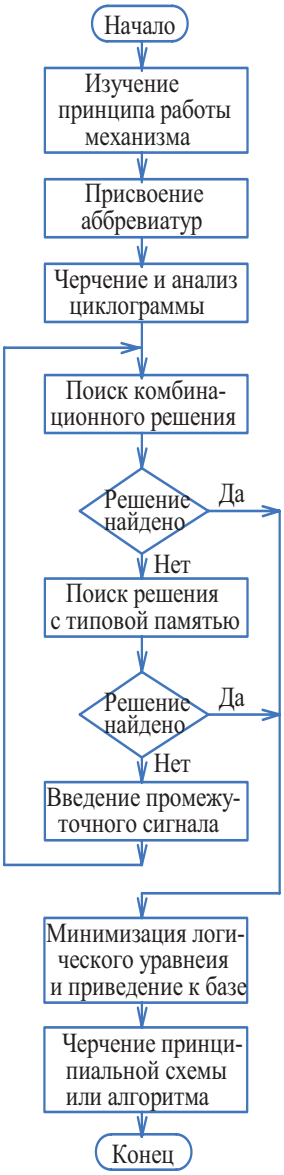


Рис. 2.21. Обобщенный алгоритм синтеза комбинационного решения

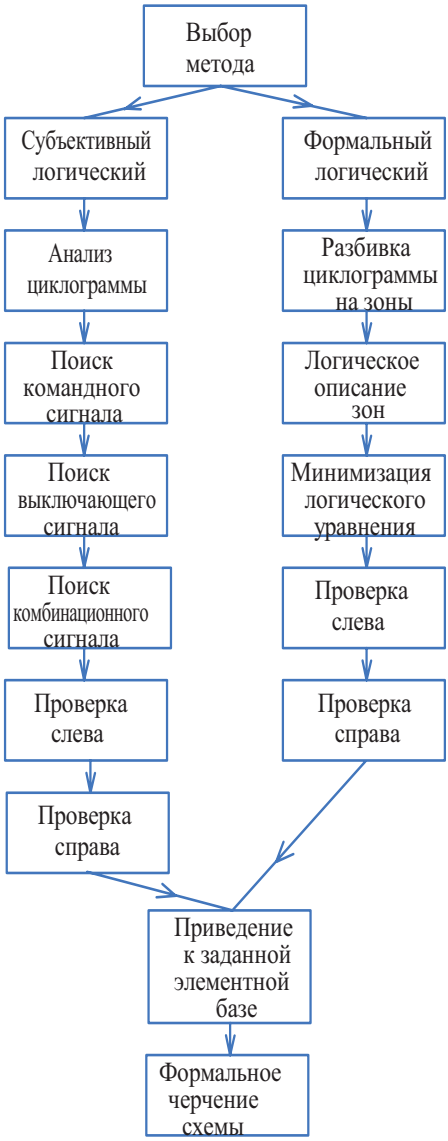


Рис. 2.22. Алгоритм поиска

Можно использовать два метода: ***творческий*** (субъективный) логический и ***формальный*** логический.

1. Используя **творческий логический метод,** проектант находит решение, мысленно анализируя всю циклограмму, и сразу записывает конечное (в общем случае не минимизированное) логическое уравнение. Из циклограммы рис. 2.23 видно, что:

- выходной сигнал Y_1 равен единице только во II зоне, когда присутствует (равен единице) входной сигнал X_1 и отсутствует входной сигнал X_2, т. е. $Y_1 = X_1 \cdot \overline{X}_2$. Ни слева (зона I) от синтезируемого сигнала Y_1, ни справа от него (зоны III–V) такой логической комбинации переменных нет, следовательно, найдено единственно правильное решение;
- выходной сигнал Y_2 равен единице только в зоне III при условии одновременного наличия входных сигналов X_1 и X_2, т. е. $Y_2 = X_1 \cdot X_2$;
- выходной сигнал Y_3 равен единице в зонах II–IV, когда присутствует входной сигнал X_1 или когда присутствует входной сигнал X_2, т. е. $Y_3 = X_1 + X_2$;
- выходной сигнал Y_4 равен единице в зонах I–II, когда отсутствует входной сигнал X_2, а также в зонах IV–V, когда отсутствует сигнал X_1, т. е. $Y_4 = \overline{X}_2 + \overline{X}_1$.

Можно найти и другое решение: сигнал Y_4 равен единице во всех зонах, кроме зоны III, когда одновременно присутствуют сигналы X_1 и X_2, т. е. $Y_4 = \overline{X_1 \cdot X_2}$. Очевидно, что эти решения равнозначны, так как по закону инверсии $\overline{X_1 \cdot X_2} = \overline{X}_1 + \overline{X}_2$.

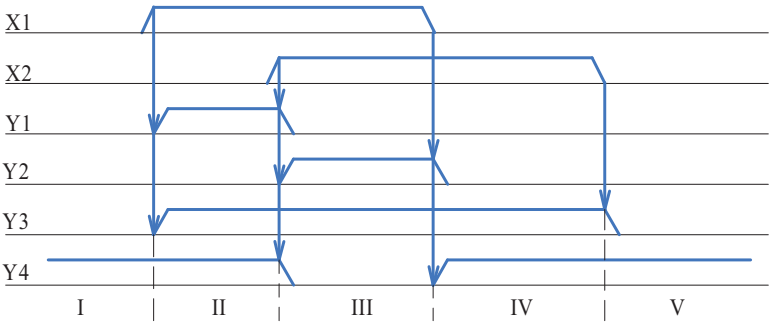


Рис. 2.23. Пример циклограммы

Заметим, что при поиске решения всегда необходимо посмотреть, какой сигнал инициирует синтезируемый выходной сигнал и какой его выключает.

Если найти таким образом решение не удается, то следует применить формальный метод.

При использовании **формального** метода логическое уравнение для синтезируемого сигнала записывается по зонам (или тактам работы) в виде канонической суммы (СДНФ), после чего выполняется минимизация и осуществляется проверка на отсутствие полученных условий слева и справа от синтезируемого сигнала.

Для выходного сигнала Y_3 (такты II, III, IV):

$$Y_3 = X_1\overline{X}_2 + X_1X_2 + \overline{X}_1X_2 = X_1(\overline{X}_2 + X_2) + \overline{X}_1X_2 = X_1 + \overline{X}_1X_2 = X_1 + X_2.$$

Для выходного сигнала Y_4 (такты I, II, IV и V):

$$Y_4 = \overline{X}_1\overline{X}_2 + X_1\overline{X}_2 + \overline{X}_1X_2 + \overline{X}_1\overline{X}_2 = \overline{X}_1\overline{X}_2 + X_1\overline{X}_2 + \overline{X}_1X_2 =$$
$$= \overline{X}_2(\overline{X}_1 + X_1) + \overline{X}_1X_2 = \overline{X}_2 + \overline{X}_1X_2 = \overline{X}_2 + \overline{X}_1 = \overline{X_1X_2}.$$

Видно, что получены аналогичные результаты.

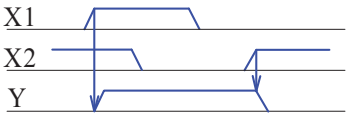
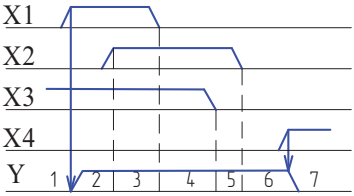
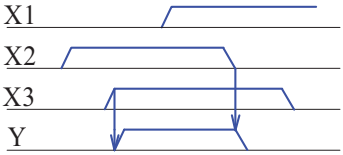
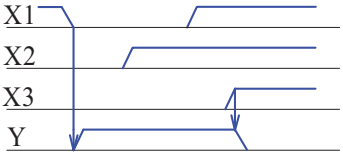
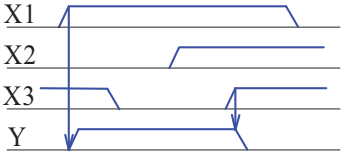
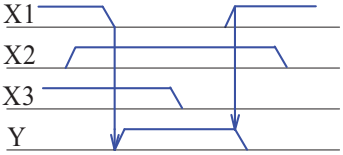
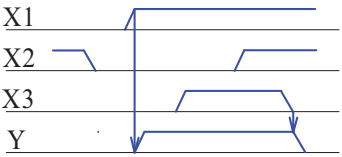
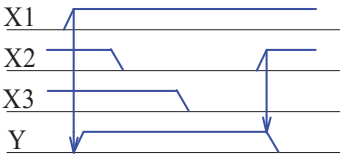
Формальный метод является основой для машинной обработки циклограммы, т. е. автоматизированного синтеза.

Полученные выше логические уравнения, описывающие состояние выходных сигналов Y_1–Y_4, являются ***минимально-достаточными***. В практике проектирования реальных схем управления промышленными механизмами в минимально-достаточные уравнения вводят дополнительные сигналы, повышающие невосприимчивость схемы к действию внутренних и внешних, случайных воздействий, т. е. блокировочные и «помехозащитные» сигналы. Такие логические уравнения будем называть ***рабочими***. Дать четкое правило введения «помехозащитных» сигналов не представляется возможным. Ясно, что чем больше дополнительных сигналов введено в уравнение, тем выше устойчивость схемы. Однако при этом возрастает ее сложность. Для схем промышленной электроавтоматики обычно достаточно двух-трех сигналов, так как их одновременное ложное появление маловероятно. Уже при небольшом навыке анализа циклограмм сразу легко записывается рабочее уравнение.

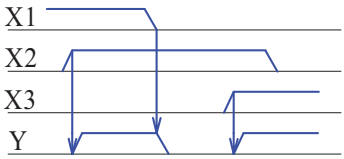
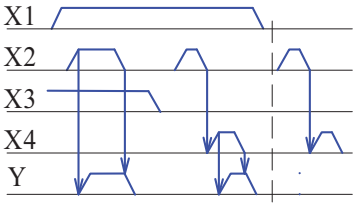
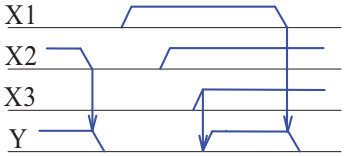
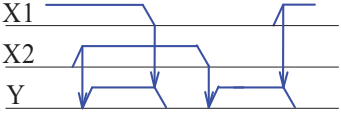
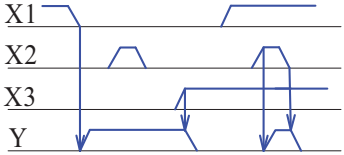
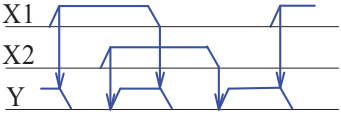
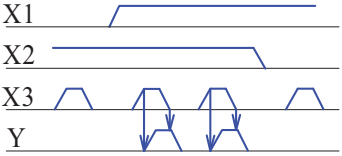
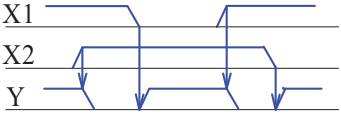
В табл. 2.10 приведены примеры фрагментов циклограмм, для которых существует комбинационное решение, т. е. выходной сигнал Y однозначно описывается уравнением алгебры логики. Под циклограммами **приведены минимально достаточные логические уравнения**. Читателю предлагается самому проанализировать правильность уравнений, что поможет получить первые самостоятельные навыки в анализе циклограмм. Ниже приводятся избранные комментарии:

Таблица 2.10

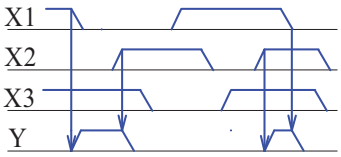
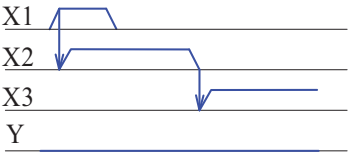
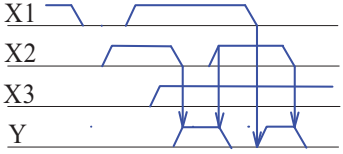
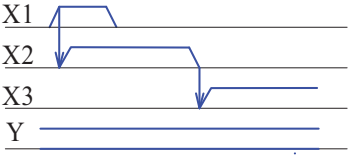
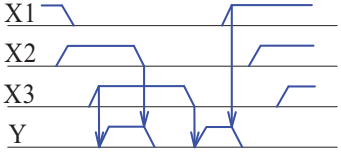
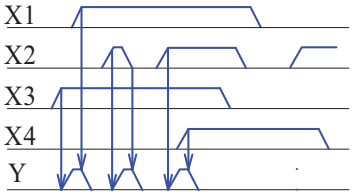
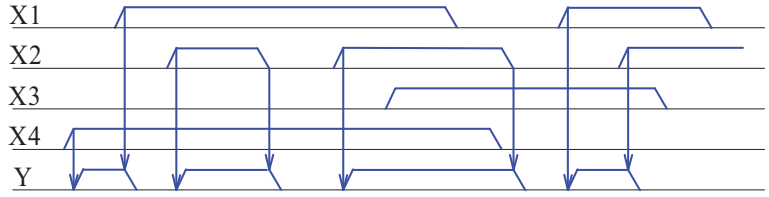
Примеры комбинационного решения

№	Циклограмма	№	Циклограмма
1	2	3	4
1	 $Y = X1 + \overline{X2}$	2	 $Y = (X1 + X2 + \overline{X3}) \cdot \overline{X4}$
3	 $Y = X3 \cdot X2$	4	 $Y = (\overline{X1} + X2) \cdot \overline{X3}$
5	 $Y = X1 \cdot (\overline{X2} + \overline{X3})$	6	 $Y = \overline{X1}$ или $Y = \overline{X1} \cdot X2$
7	 $Y = X1 \cdot (\overline{X2} + X3)$	8	 $Y = X1 \cdot (X3 + \overline{X2})$

Продолжение таблицы 2.10

№	Циклограмма	№	Циклограмма
1	2	3	4
9	 $Y = X2 \cdot X1 + X3$	10	 $Y = X1 \cdot (X2 \cdot X3 + X4)$
11	 $Y = X2 \cdot \overline{X1} \cdot \overline{X3} + X3 \cdot X1$	12	 $Y = (X1 \equiv X2)$
13	 $Y = \overline{X1} \cdot \overline{X3} + X2)$	14	 $Y = (X1 \equiv X2)$
15	 $Y = X1 \cdot X2 \cdot X3$	16	 $Y = (X1 \neq X2)$

Окончание таблицы 2.10

№	Циклограмма	№	Циклограмма
1	2	3	4
17	 $Y = (X1 \equiv X2) \cdot X3$	18	 $Y = 0;\ \ (Y = X \cdot \overline{X})$
19	 $Y = (X1 \neq X2) \cdot X3$	20	 $Y = 1;\ \ (Y = X + \overline{X})$
21	 $Y = (X1 \equiv X2) \cdot \overline{X1}$	22	 $Y = (X1 \equiv X2) \cdot X3 \cdot \overline{X4}$
23	 $Y = (X1 \equiv X2) \cdot X4 + (X1 \neq X2) \cdot X3$		

Циклограмма 2. Возможный ход рассуждений следующий.

Выходной сигнал Y включается входным сигналом X_1. Если принять, что $Y = X_1$, то он будет включен во 2 и 3 тактах.

В третьем такте сигнал X_1 перекрывается с сигналом X_2, поэтому, если написать, что $Y = X_1 + X_2$, то он будет включен со 2-го по 5-й такт.

В пятом такте сигнал X_2 перекрывается с инверсным сигналом X_3. Если записать, что $Y = X_1 + X_2 + \overline{X}_3$, то он будет включен со 2-го такта по 7-й. Последний такт необходимо ограничить. Видно, что в седьмом такте выходной сигнал Y выключается появлением выходного сигнала X_4, который равен нулю во время действия синтезируемого сигнала Y, следовательно $Y = (X_1 + X_2 + \overline{X}_3) \cdot \overline{X}_4$.

Это уравнение минимально достаточно, т. е. самое простое, обеспечивающее заданное включение сигнала Y при заведомо правильной коммутации сигналов X_1–X_4 по циклограмме.

Итак, нахождение комбинационного решения очень простое.

При использовании формального метода решения задачи уравнение записывается в виде канонической суммы и минимизируется, т. е.

$$\begin{aligned}
Y &= X_1\overline{X}_2X_3\overline{X}_4 + X_1X_2X_3\overline{X}_4 + \overline{X}_1X_2X_3\overline{X}_4 + \overline{X}_1X_2\overline{X}_3\overline{X}_4 + \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 = \\
&= (X_1\overline{X}_2X_3 + X_1X_2X_3 + \overline{X}_1X_2X_3 + \overline{X}_1X_2\overline{X}_3 + \overline{X}_1\overline{X}_2\overline{X}_3)\overline{X}_4 = \\
&= \left[X_1X_3(\overline{X}_2 + X_2) + \overline{X}_1X_2(X_3 + \overline{X}_3) + \overline{X}_1\overline{X}_2\overline{X}_3 \right]\overline{X}_4 = \\
&= (X_1X_3 + \overline{X}_1X_2 + \overline{X}_1\overline{X}_2\overline{X}_3)X_4 = \\
&= \left[X_1X_3 + \overline{X}_1(X_2 + \overline{X}_2\overline{X}_3) \right]\overline{X}_4 = \left[X_1X_3 + \overline{X}_1(X_2 + \overline{X}_3) \right]\overline{X}_4 = \\
&= \left[X_1X_3 + \overline{X}_1X_2 + \overline{X}_1\overline{X}_3 \right]\cdot\overline{X}_4.
\end{aligned}$$

Дальнейшее преобразование в сторону упрощения по правилам алгебры логики практически невозможно. Полученное уравнение сложнее минимально достаточного, однако в него автоматически вошли дополнительные «защитные» сигналы. Действительно, выходной сигнал Y не включится с появлением входного сигнала X_1, если в такте 2 будет отсутствовать сигнал X_3. Осуществляется как бы проверка начальных условий.

Циклограмма 6. Очевидно, что выходной сигнал Y является инверсией входного сигнала X_1 и формально не зависит от состояния входных сигналов X_2 и X_3, т. е. $Y = \overline{X}_1$. Это минимально достаточная запись. В рабочее уравнение следует добавить сигнал X_2 , т. е. $Y = \overline{X}_1 \cdot X_2$. Такая запись исключит включение сигнала Y при аварийном пропадании сигнала X_1 , если еще не было сигнала X_2, как того требует цикл работы.

Состояние входа X_3 безразлично. Покажем это:

$$Y = \overline{X}_1X_2X_3 + \overline{X}_1X_2\overline{X}_3 = \overline{X}_1X_2(X_3 + \overline{X}_3) = \overline{X}_1X_2.$$

Циклограмма 9. Минимально достаточное уравнение очевидно $Y = X_2 \cdot X_1 + X_3$, так как выходной сигнал Y включен, когда присутствуют оба

сигнала X_1 и X_2 или, когда включается сигнал X_3. В рабочее уравнение можно добавить инверсные сигналы X_3 и X_1, так как логическая комбинация $X_1 \cdot X_2$ по условиям циклограммы не может быть при включенном состоянии X_3, а включенное состояние X_3 допустимо только при выключенном сигнале X_1, т. е. $Y = X_2 \cdot X_1 \cdot \overline{X}_3 + X_3 \cdot \overline{X}_1$. Необходимость блокировки решается субъективно в каждом конкретном случае.

2.9.2. Поиск решения с типовой памятью

Выполним анализ циклограммы рис. 2.24, применив формальный метод. Очевидно, что

$$Y = X_1X_2 + \overline{X}_1X_2 + \overline{X}_1\overline{X}_2 = X_2(X_1 + \overline{X}_1) + \overline{X}_1\overline{X}_2 = X_2 + \overline{X}_1\overline{X}_2 = X_2 + \overline{X}_1 \ .$$

Данное логическое уравнение правильно описывает синтезируемый выходной сигнал Y в тактах III÷IV, где он должен быть равен единице по условиям циклограммы, однако проверка на отсутствие логической комбинации $(X_2 + X_1)$ слева (такты I, II) и справа (такты VI, VII) от синтезируемого сигнала дает отрицательный результат. В тактах I и VII входной сигнал X_1 инверсен, т. е. при практической реализации уравнения $Y = X_2 + \overline{X}_1$ здесь также произойдет включение сигнала Y. Исходные условия нарушены, **решение неправильное**.

Подобные случаи, практически, всегда имеют место, так как реальные схемы промышленной электроавтоматики в большинстве случаев являются ***последовательностными***, а не комбинационными. Синтез последовательностных схем может быть выполнен при помощи теории дискретных автоматов, однако она исключительно трудоемка и сложна, и ее применение в практической инженерной деятельности конструкторских бюро, занимающихся промышленной электроавтоматикой, в большинстве случаев, не оправдано.

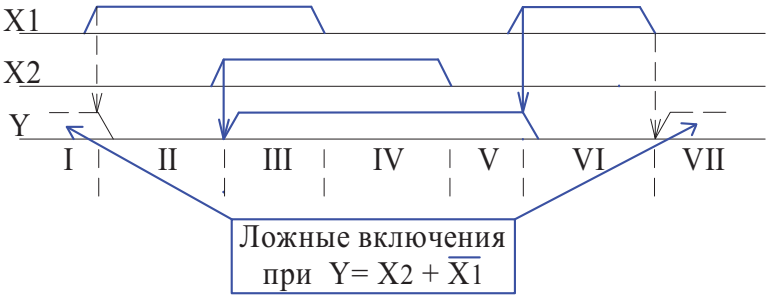


Рис. 2.24. Пример циклограммы

Предлагаемое здесь **инженерное допущение** позволяет просто и эффективно выполнить синтез последовательностных схем методами комбинационной логики, т. е. осуществить приведение последовательностных схем к комбинационным.

Известно, что характерной особенностью последовательностных схем является наличие элементов памяти, так как состояние выходных логических переменных в подобных схемах зависит от состояния входных логических переменных как на данный, так и на предыдущий момент времени.

Принцип приведения последовательностных схем к комбинационным основывается на положении о том, что в инженерной практике применяются типовые элементы памяти (RS, D, T, V, JK-триггеры), серийно выпускаемые промышленностью в виде законченных микросхем или логических элементов, либо схемы памяти организуются по типовым схемам из отдельных элементов (И-НЕ, ИЛИ-НЕ, реле). Следовательно, нет необходимости заниматься их синтезом, а следует применить готовую типовую схему, выбрав ее из каталога.

Рис. 2.25. Алгоритм решения с памятью

Алгоритм поиска решения с типовой памятью можно представить в следующей последовательности (рис. 2.25):

- анализ циклограммы, подтверждение невозможности решения задачи методами комбинационной логики;
- принятие решения о применении типовой памяти;
- выбор типа и элементной базы схемы памяти;
- синтез по циклограмме методами комбинационной логики условия включения $S = f(X_i , Y_j)$ синтезируемого сигнала;
- проверка на отсутствие сигналов, определяемых функцией $S = f (X_i , Y_j)$ до и после (слева и справа) от синтезируемого выходного сигнала Y. Повторное появление включающего сигнала в течение времени действия сигнала Y для RS-триггера, D-триггера и памяти на реле **допустимо**;
- синтез по циклограмме методами комбинационной логики условия выключения $R = f(X_i, Y_j)$ синтезируемого сигнала;
- проверка на отсутствие сигналов, определяемых функцией $R = f(X_i, Y_j)$ во время действия синтезируемого сигнала Y. Появление выключающего сигнала R до (слева) начала действия сигнала Y и его повторное появление после (справа) окончания действия сигнала Y для RS-триггеров, D-триггеров и памяти на реле допустимо;
- минимизация и приведение синтезированных уравнений $S = f(X_i, Y_j)$ и $R = f(X_i, Y_j)$ к применяемой элементной базе;
- формальное вычерчивание принципиальной схемы.

В промышленной электроавтоматике в качестве элементов памяти наибольшее применение находят простейшие асинхронные RS-триггеры.

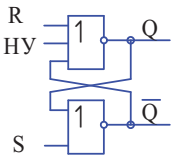
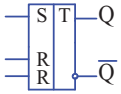
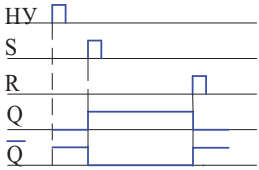
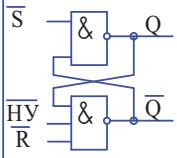
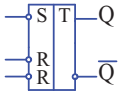
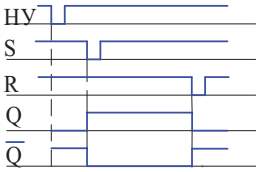
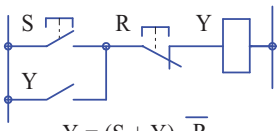
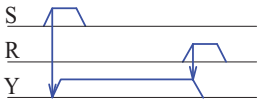
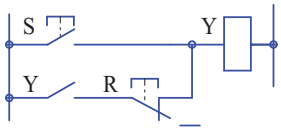
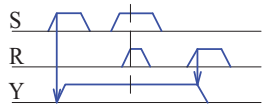
Асинхронный RS-триггер (табл. 2.11) имеет два выхода: Q – прямой и $\overline{Q}$ – инверсный, разрешенные сигналы которых в триггерном режиме всегда имеют противоположные значения, и два входа: S – установка (Set) и R – сброс (Reset). Сигнал S записывает в триггер единицу, а сигнал R – ноль.

Для сброса триггера в исходное нулевое положение необходимо подать кратковременный единичный импульс по входу R. Процесс переключения аналогичен.

В общем случае, RS-триггеры имеют по несколько R- и S-входов. Поскольку при включении питания триггер может установиться в произвольное состояние, один из входов всегда должен использоваться для ***начальной установки*** триггера в необходимое положение (вход НУ). Это может быть как R-, так и S-вход.

Таблица 2.11

Типовые схемы памяти

Элементная база	Принципиальная схема и условное обозначение		Диаграмма работы
1	2		3
ИЛИ-НЕ			
И-НЕ			
РКС	 $Y = (S + Y) \cdot \overline{R}$ Приоритет сигнала выключения R		
	 $Y = S + Y \cdot \overline{R}$ Приоритет сигнала включения S		

RS-триггер на элементах ИЛИ-НЕ (а), в силу логической сущности ячеек $\left(Y = \overline{X_1 + X_2}\right)$, на которых он реализован, управляется ***единичными*** сигналами R = 1 и S = 1. При нормальной работе в режиме асинхронного триггера сигналы по входам R и S должны подаваться поочередно.

Процесс переключения триггера из нулевого исходного состояния ($Q = 0$, $\overline{Q} = 1$), происходит в следующей последовательности:

- подача управляющего импульса S = 1 на вход элемента D2 (при этом предполагается, что R = 0 и НУ = 0);
- установка элемента D2 в состояние ноль ($\overline{Q} = 0$), так как

$$\overline{Q} = \overline{Q + S} = \overline{0 + 1} = \overline{1} = 0;$$

- передача состояния выхода $\overline{Q} = 0$ элемента D2 по цепи обратной связи на вход элемента D1;
- установка элемента D1 в состояние единица (Q = 1), так как

$$Q = \overline{R + НУ + \overline{Q}} = \overline{0 + 0 + 0} = \overline{0} = 1;$$

- передача состояния выхода Q = 1 элемента D1 по цепи обратной связи на вход элемента D2;
- запоминание (подтверждение) нового состояния $\overline{Q} = 0$ элемента D2, так как $\overline{Q} = \overline{Q + S} = \overline{1 + 1} = 0$;
- снятие управляющего сигнала S = 1. Новое единичное ($Q = 1$, $\overline{Q} = 0$) состояние триггера зафиксировано:

$$\overline{Q} = \overline{Q + S} = \overline{1 + 0} = 0 \quad \text{и} \quad Q = \overline{R + НУ + \overline{Q}} = \overline{0 + 0 + 0} = 1.$$

Рассмотрим, что произойдет, если условие поочередной подачи сигналов по входам R и S соблюдаться не будет:

1. *На входы R и S одновременно поданы единичные сигналы* (R = 1, S = 1).

Очевидно, что оба элемента ИЛИ-НЕ, составляющие триггер D1 и D2, установятся в устойчивое состояние нуля ($Q = 0$ и $\overline{Q} = 0$), что можно классифицировать как разрыв триггерной связи. RS-триггер превратился в **два независимых инвертора**. Допустима такая ситуация или нет, определяется в каждом конкретном случае индивидуально.

2. *На входы R и S одновременно поданы нулевые сигналы* (R = 0, S = 0). Такая ситуация **допустима** после сброса или установки триггера, так как в этих случаях он сохраняет (запоминает) заданное состояние, и **недопустима** в момент подачи на схему напряжения питания, так как триггер может занять произвольное неконтролируемое состояние, что приведет к неправильной работе или аварии. По этой причине один из входов триггеров всегда используется для его начальной установки в требуемое положение. Варианты формирования импульсов начальной установки (НУ) при включении питания приведены на рис. 2.26.

Дополнительная кнопка КнНУ обеспечивает вывод схемы с использованием RS-триггеров в наладочном режиме РН из нерабочего состояния при сбое.

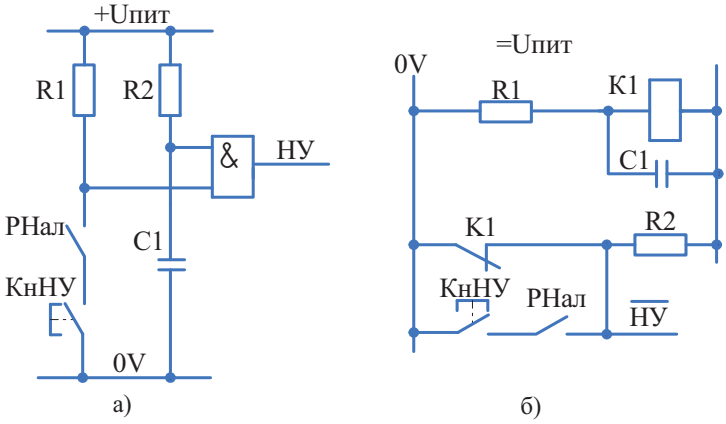


Рис. 2.26. Принципы формирования сигнала начальной установки

RS-триггер на элементах И-НЕ (табл. 2.11), в силу логической сущности ячеек $\left(Y = \overline{X_1 \cdot X_2}\right)$, на которых он реализован, управляется ***нулевыми*** сигналами (R = 0 и S = 0).

При нормальной работе в режиме асинхронного триггера подача сигналов по входам R и S осуществляется поочередно. Процесс переключения триггера аналогичен.

При одновременной подаче на входы R и S нулевых сигналов происходит разрыв триггерных связей, превращающих его в два инвертора ($Q = 1$ и $\overline{Q} = 1$).

Одновременная подача на входы R и S единичных сигналов допустима после штатного переключения триггера в любое состояние и недопустима при включении питания. Для ориентированной установки RS-триггера на элементах И-НЕ формируется инверсный сигнал начальной установки $\overline{НУ} = 0$.

При вычерчивании принципиальных схем управления промышленной электроавтоматикой удобно изображение RS-триггеров в виде, показанном на рис. 2.27.

Такое начертание понятно для чтения и позволяет более компактно изобразить всю схему.

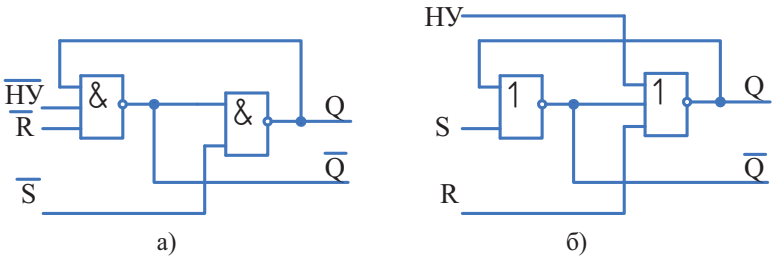


Рис. 2.27. Варианты начертания RS-триггера

Релейно-контактные ячейки памяти (аналоги RS-триггеров) могут быть реализованы по двум вариантам: $Y = (S + Y) \cdot \overline{R}$ или $Y = S + Y \cdot \overline{R}$ (см. табл. 2.11). Оба варианта функционально равнозначны. При кратковременной подаче сигнала S включается выходное реле Y и встает на самопитание. Выключение памяти осуществляется кратковременной подачей сигнала сброса R. Видно, что включение памяти осуществляется прямым сигналом S (замыкание), а отключение инверсным R (размыкание). Сигнала начальной установки в релейном варианте не требуется.

Реализация памяти по уравнению $Y = (S + Y) \cdot \overline{R}$ **предпочтительней**, так как в этом случае исключается срабатывание выходного реле Y при одновременной подаче сигналов S и R.

При разработке схем электроавтоматики с использованием рассмотренных типовых ячеек памяти синтезу подвергаются только сигналы установки S и сброса R, после чего в соответствие с табл. 2.12 определяют необходимость инвертирования сигналов управления. Одновременно осуществляется приведение логических уравнений к выбранной элементной базе.

Таблица 2.12

Приведение сигналов управления к элементной базе

Элементная база	**Входной сигнал**	
	Установка (S)	**Сброс (R)**
ИЛИ-НЕ	прямой (S)	прямой (R)
И-НЕ	инверсный ($\overline{S}$)	инверсный ($\overline{R}$)
Релейно-контактная	прямой (S)	инверсный ($\overline{R}$)

В заключение сделаем **важный вывод**: асинхронные RS-триггеры не реагируют на многократное повторное воздействие по входам R или S после завершения процесса установки или сброса, т. е. **не реагируют на «дребезг» входных сигналов**.

Это обстоятельство всегда учитывается при синтезе или специально используется для повышения помехозащищенности различных счетных схем.

Другие типы триггеров будут рассмотрены в главе IV.

Вновь вернемся к циклограмме рис. 2.24, при анализе которой было выяснено, что решение не может быть найдено при помощи комбинационной логики. В соответствии с обобщенным алгоритмом синтеза принимаем решение использовать типовую память. Кажущееся решение $S = X_2$ и $R = X_1$ неверно, так как в такте III будет одновременная подача сигналов установки и сброса, что в триггерном режиме недопустимо.

Неопределенность исключается путем логического умножения сигнала X_1 на инверсный сигнал $\overline{X}_2$, т. е. $R = X_1 \cdot \overline{X}_2$.

Дальнейший анализ показывает, что состояние $X_1 \cdot \overline{X}_2$ имеет место и в такте II. Однако формирование сигнала сброса R в такте II, предшествующему включенному состоянию триггера не изменяет логики работы циклограммы и допустимо. Найденное решение является минимально достаточным. В логическую формулу управляющего сигнала включения $S = X_2$ можно ввести логическое умножение на прямой сигнал X_1, как бы учитывая начальные условия, т. е. $S = X_2 \cdot X_1$. В этом случае включение выхода Y при подаче сигнала X_2 произойдет только при условии, что перед этим, как требует циклограмма, был включен сигнал X_1.

Заметим, что при $S = X_2 \cdot X_1$ время действия включающего сигнала ограничивается только тактом III. При $S = X_2$ его действие занимает два такта III и IV. Принятие окончательного решения – субъективно. При синтезе реальных схем следует учитывать и начальные условия и формировать максимально возможные по длительности управляющие сигналы, так как при этом возрастает защищенность от помех и случайных воздействий.

Для построения принципиальной схемы (рис. 2.28) полученные логические уравнения необходимо привести к используемой элементной базе.

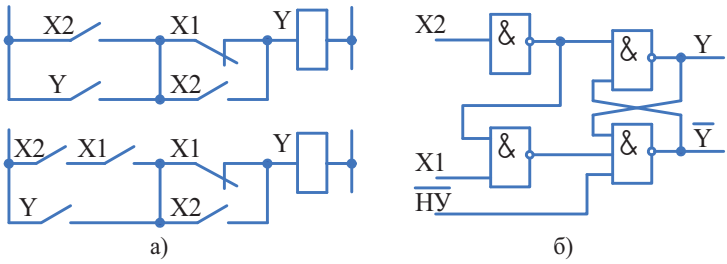


Рис. 2.28. Принципиальные схемы на реле (а) и элементах И-НЕ (б), синтезированные по циклограмме рис. 2.24

Сигнал включения:

а) релейная база $S = X_2$ или $S = X_2 \cdot X_1$;

б) элементы И-НЕ $S = X_2$ и $\overline{S} = \overline{X}_2$.

Сигнал включения:

а) релейная база $R = X_1 \cdot \overline{X}_2$ и $\overline{R} = \overline{X_1 \overline{X_2}} = \overline{X}_1 + X_2$;

б) элементы И-НЕ $R = X_1 \cdot \overline{X}_2$ и $\overline{R} = \overline{X_1 \overline{X_2}}$.

В табл. 2.13 приведены фрагменты элементарных циклограмм, реализуемых при помощи типовых ячеек памяти. Логические уравнения для сигналов установки и сброса памяти, приведенные под циклограммами являются минимально-достаточными и не привязаны к какой-либо конкретной элементной базе. Могут быть найдены и другие варианты решений. Подчеркнем, что синтез этих сигналов следует проводить для **прямых** значений S и R, и только потом проводить инвертирование в соответствии с требованиями элементной базы (табл. 2.12).

Приведем некоторые комментарии.

Циклограмма 3. Первое включение выхода Y осуществляет входной сигнал X_1, второе – X_2. Первое выключение осуществляет сигнал X_2, второе – X_1. Неопределенность легко устранится при помощи третьего входного сигнала X_3. Если такого сигнала в исходной циклограмме нет, то его следует сформировать искусственно (о чем будет сказано ниже).

Приведение к элементарной базе **И-НЕ**:

$$\overline{S} = \overline{X_1 X_3 + X_2 \overline{X}_3} = \overline{X_1 X_3} \cdot \overline{X_2 \overline{X}_3} = \overline{\overline{X_1 X_3} \cdot \overline{X_2 \overline{X}_3}},$$

$$\overline{R} = \overline{X_2 X_3 + X_1 \overline{X}_3} = \overline{X_2 X_3} \cdot \overline{X_2 \overline{X}_3} = \overline{\overline{X_2 X_3} \cdot \overline{X_2 \overline{X}_3}}.$$

Практическая реализация триггеров возможна с использованием как общего, так и раздельных сигналов управления (рис. 2.29, *а*, *б*).

Приведение к элементной базе **ИЛИ-НЕ**:

$$S = X_1 X_3 + X_2 \overline{X}_3 = \overline{\overline{X}_1 + \overline{X}_3} + \overline{\overline{X}_2 + X_3} = \overline{\overline{X}_1 + \overline{X}_3 + \overline{X}_2 + X_3},$$

$$R = X_2 X_3 + X_1 \overline{X}_3 = \overline{\overline{X}_2 + \overline{X}_3 + \overline{X}_1 + X_3}.$$

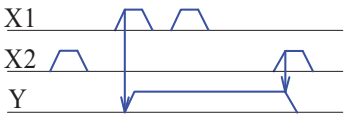
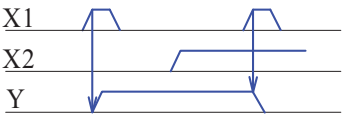
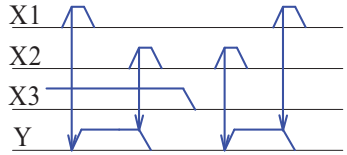
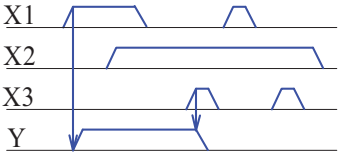
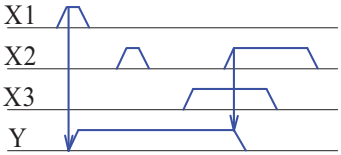
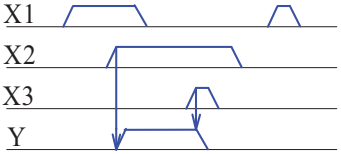
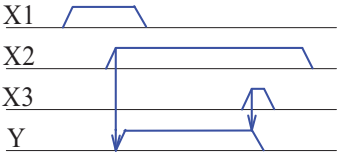
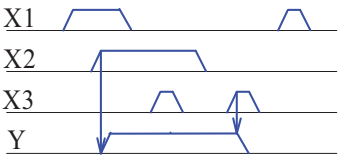
Синтезированная принципиальная схема приведена на рис. 2.29, *в*. Приведение к **релейно-контактной** элементной базе (рис. 2.29, *г*):

$$S = X_1 X_3 + X_2 \overline{X}_3,$$

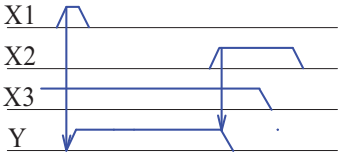
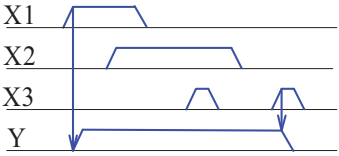
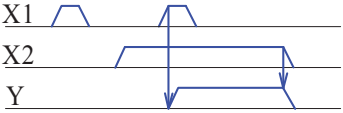
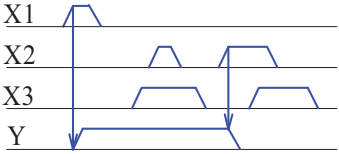
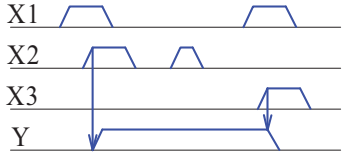
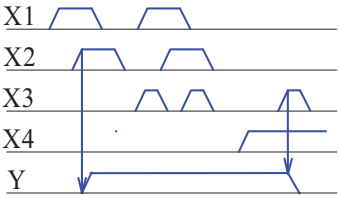
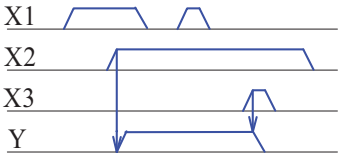
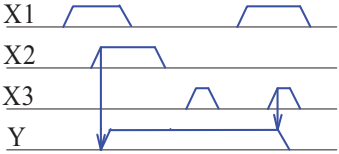
$$\overline{R} = \overline{X_2 X_3 + X_1 \overline{X}_3} = \overline{X_2 X_3} \cdot \overline{X_1 \overline{X}_3} = (\overline{X}_2 + \overline{X}_3)(\overline{X}_1 + X_3).$$

Таблица 2.13

Примеры решения с типовой памятью

№	Циклограмма	№	Циклограмма
1	2	3	4
1	 $S = X1$ $R = X2$	2	 $S = X1 \cdot \overline{X2}$ $R = X1 \cdot X2$
3	 $S = X1 \cdot X3 + X1 \cdot \overline{X3}$ $R = X2 \cdot X3 + X2 \cdot \overline{X3}$	4	 $S = X1 \cdot \overline{X2}$ $R = X3$
5	 $S = X1$ $R = X2 \cdot X3$	6	 $S = X2 \cdot X1$ $R = X3$
7	 $S = X2 \cdot X1$ $R = X3$	8	 $S = X2 \cdot X1$ $R = X3 \cdot \overline{X2}$

Продолжение таблицы 2.13

№	Циклограмма	№	Циклограмма
1	2	3	4
9	 S = X1 R = X2	10	 S = X1 R = X3 · $\overline{X2}$
11	 S = X1 · $\overline{X2}$ R = X2	12	 S = X1 R = X2 · $\overline{X3}$
13	 S = X2 или S = X2 · X1 R = X3 или R = X3 · X1	14	 S = X2 R = X3 · X4
15	 S = X2 · X1 R = X3	16	 S = X2 · X1 или S = X2 R = X3 · X1

Окончание таблицы 2.13

№	Циклограмма	№	Циклограмма
1	2	3	4
17	X1 X2 X3 Y S = X2 или S = X2 · X1 R = X3	18	X1 X2 X3 Y S = X2 · X1 R = X3 · X2

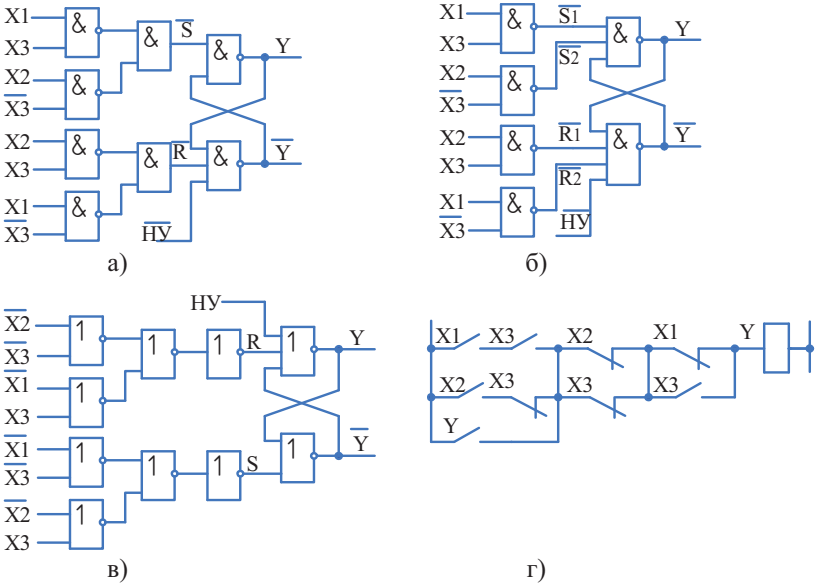


Рис. 2.29. Принципиальные схемы на элементах И-НЕ (а, б), ИЛИ-НЕ (в) и реле (г), синтезированные по циклограмме 3 табл. 2.13

Циклограмма 6. Выходной сигнал Y однозначно включается входным сигналом X_2 и выключается сигналом X_1, однако длительность сигнала X_2 превышает длительность сигнала Y, что приведет к его повторному включению после пропадания сигнала X_3. Введение логического умножения на сигнал X_1 решает задачу:

$$S = X_2 \cdot X_1.$$

Циклограмма 10. Включение выходного сигнала Y осуществляет входной сигнал X_2, а выключение второй сигнал X_3. Выделение второго импульса X_2 легко осуществляется логическим умножением на инверсный сигнал X_2, т. е. $R = X_3 \cdot \overline{X}_2$. В логическое уравнение для сигнала включения может быть введен учет начальных условий, т. е. $S = X_2 \cdot X_1$.

Полученных теперь знаний достаточно, чтобы вернуться к анализу циклограмм рис. 2.17, *г*, *в*, объясняющих приемы исключения на этапе синтеза явления состязания. Временной принцип управления фронтами здесь заменен на четкий последовательный процесс управления.

Повторим ее (рис. 2.30). Выходные сигналы легко синтезируются по рассмотренной методике.

Циклограмма а, случай $Y = X_1 + X_2 = 1$.

Сигнал β синтезируется комбинационной логикой $\beta = \alpha\overline{X}_1 + \overline{\alpha}X_1$.

Сигналы X_1 и X_2 синтезируются при помощи типовой ячейки памяти.

X_1: Сигнал установки S однозначно описывается уравнением $S = \beta \cdot \alpha$.

Сигнал сброса R на первый взгляд можно описать уравнениями

$$R = \overline{\beta}\overline{\alpha} \text{ и } R = \overline{\beta}X_2,$$

но для них существуют зоны риска в точках циклограммы 1) и 2), соответственно.

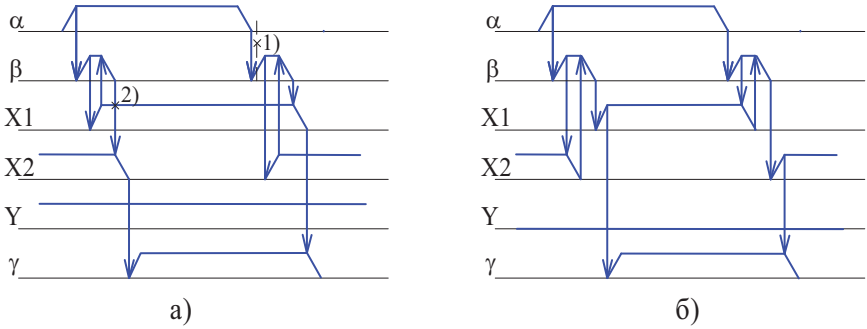


Рис. 2.30. Пример циклограммы

По этой причине вводим промежуточный сигнал γ, легко реализуемый типовой памятью: $S = \overline{X}_2$ и $R = \overline{X}_1$.

Тогда уравнение для сигнала сброса $R = \overline{\beta}X_2\gamma$.

X_2: Сигнал установки $S = \beta\overline{\alpha}$

Сигнал сброса $R = \overline{\beta}X_1\overline{\gamma}$.

Циклограмма б, случай $Y_2 = X_1 X_2 = 0$. Решение находится аналогично:

$$\beta = \alpha X_2 + \bar{\alpha} X_1;$$

$$\gamma: \; S = X_1 \text{ и } R = X_2;$$

$$X_1: \; S = \bar{\beta} X_2 \bar{\gamma} \text{ и } R = \beta\bar{\alpha};$$

$$X_2: \; S = \bar{\beta} X_1 \gamma \text{ и } R = \beta\alpha.$$

В реальной практике проектирования довольно часто встречаются случаи, когда синтезируемый сигнал активируется *несколько раз за цикл работы,* причем в одном случае он синтезируется при помощи комбинационного логического уравнения, а в другом только при помощи типового элемента памяти. Такой пример показан на рис. 2.31.

Первая активация сигнала Y1 описывается комбинационным логическим уравнением $Y_1 = X_1 \cdot X_2$, а вторая Y_2 – при помощи использования типовой памяти, где $S = X_3$ и $R = X_3$.

Если логически сложить эти два решения, то

$$Y = Y_1 + Y_2 = X_1 \cdot X_2 + (X_3 \cdot Y) \cdot \overline{X}_4.$$

Как следует из схемы рис. 2.31, *а*, в момент единичного состояния сигнала X_1 выходное реле Y включится и встанет на самопитание, работа схемы будет нарушена.

Правильное решение показано на рис. 2.31, *б*. Оно заключается в раздельной реализации сигналов Y_1 и Y_2 с последующим их логическим суммированием, т. е. $Y = Y_1 + Y_2$.

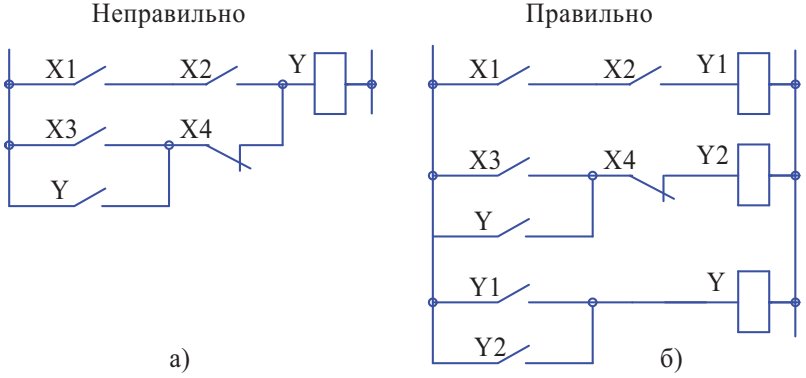


Рис. 2.31. Пример неправильного (а) и правильного (б) решения задачи согласно циклограмме (рис. 2.30, а)

2.9.3. Поиск решения с промежуточными сигналами

Обратимся к анализу циклограмм рис. 2.32, *а*.

Поиск комбинационного решения результата не дает, так как запись $y = x + \overline{x} = 1$ приводит к постоянному включению сигнала Y.

Поиск решения с типовой памятью типа RS-триггера дает неопределенность.

Задача легко решается, если применить счетный Т-триггер, но такая элементная база применяется редко, в том числе и в программируемых логических контроллерах.

Решение легко находится, если бы в зоне 3, между входными сигналами X, имелся промежуточный сигнал α, но такого сигнала в исходной циклограмме нет. Следовательно, его нужно ввести, при этом он должен отвечать следующим условиям:

- обеспечить раскрытие неопределенности, т. е. начинаться в любой точке между сигналами X;
- не зависеть от длительностей сигналов X и Y, т. е. быть четко привязанным к их переднему или заднему фронтам.

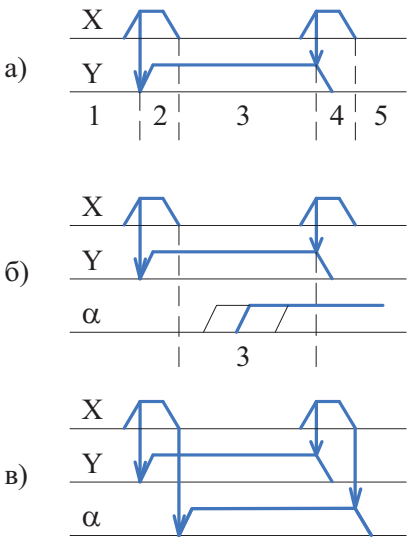


Рис. 2.32. Исходная (а), теоретическая (б) и рабочая (в) циклограммы

Этим условиям удовлетворяет промежуточный сигнал «α», определяемый задними фронтами сигналов X (рис. 2.32, *в*).

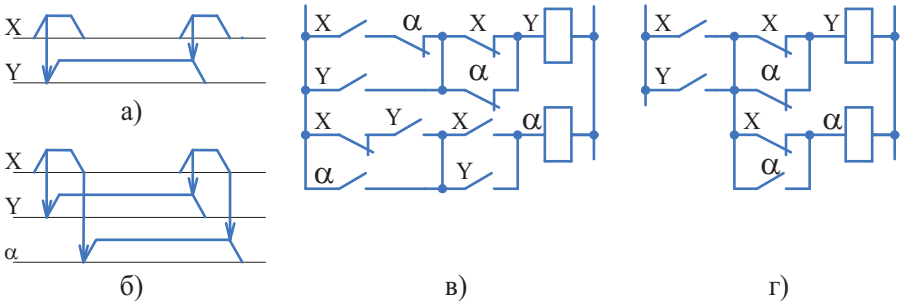


Рис. 2.33. Исходная (а) и рабочая (б) циклограммы, схемные решения (в, г)

Введя в циклограмму промежуточный сигнал, следует вновь попытаться найти сначала комбинационное решение (в нашем случае его нет), либо решение с типовой памятью:

Память сигнала Y: установка $S = X \cdot \overline{\alpha}$; сброс $R = X \cdot \alpha$.

Память сигнала α: установка $S = \overline{X} \cdot Y$; сброс $R = \overline{X} \cdot \overline{Y}$.

Подобное решение может служить типовой ячейкой счетного триггера (Т-триггера) на реле (рис. 2.33, *в*). Заменив условное обозначение входного сигнала X на T получим:

Реле Y: установка $S = T \cdot \overline{\alpha}$; сброс $\overline{R} = \overline{T \cdot \alpha} = \overline{T} + \overline{\alpha}$.

Реле α: установка $S = \overline{T} \cdot Y$; сброс $\overline{R} = \overline{\overline{T} \cdot \overline{Y}} = T + Y$.

Если для реле Y использовать уравнение типа $Y = (S + Y) \cdot \overline{R}$, а для реле α – типа $\alpha = S + \alpha \cdot \overline{R}$, то можно уменьшить общее число контактов, используемых в схеме.

Действительно:

$$Y = (S + Y)R = (T \cdot \overline{\alpha} + Y)(\overline{T} + \overline{\alpha}) = T \cdot \overline{\alpha} + Y \cdot \overline{T} + Y \cdot \overline{\alpha} =$$
$$= \overline{\alpha}(T + Y) + Y \cdot \overline{T} + T \cdot \overline{T} = \overline{\alpha}(T + Y) + \overline{T}(Y + T) =$$
$$= (T + Y)(\overline{T} + \overline{\alpha});$$
$$\alpha = S + \alpha \cdot \overline{R} = \overline{T} \cdot Y + \alpha \cdot \overline{\overline{T} \cdot \overline{Y}} = \overline{T} \cdot Y + \alpha(T + Y) =$$
$$= T \cdot \overline{T} + \overline{T} \cdot Y + \alpha(T + Y) = \overline{T}(T + Y) + \alpha(T + Y) =$$
$$= (T + Y)(\overline{T} + \alpha).$$

Выделив общую цель (T+Y), вычерчиваем схему, показанную на рисунке 2.33, *г*.

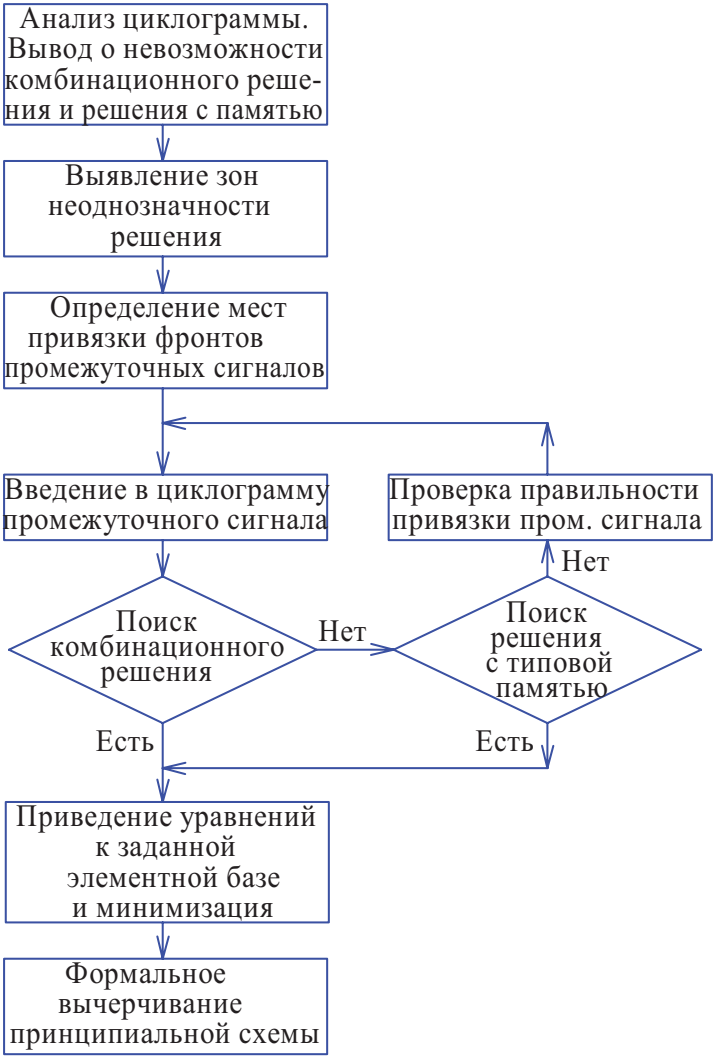


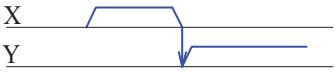
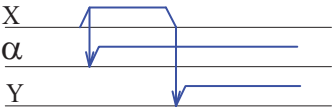
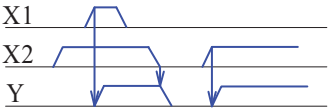
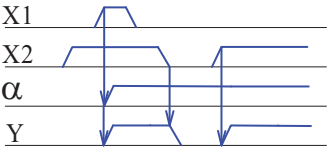
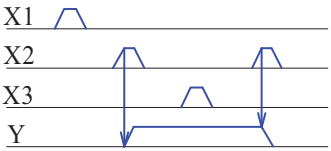
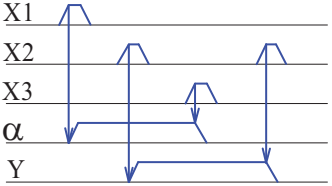
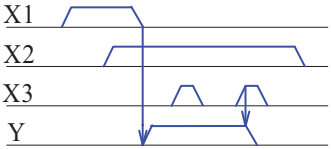
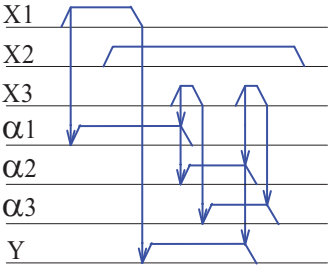
Рис. 2.34. Обобщенный алгоритм синтеза с введением промежуточного сигнала

Обобщенный алгоритм синтеза с введением промежуточного элемента приведен на рис. 2.34.

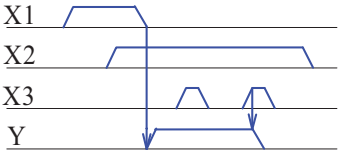
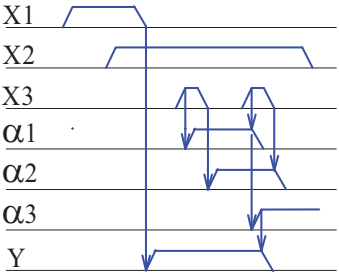
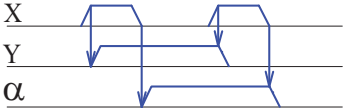
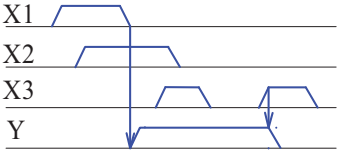
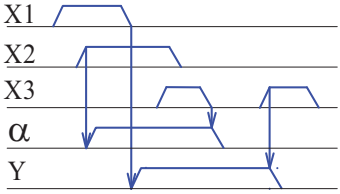
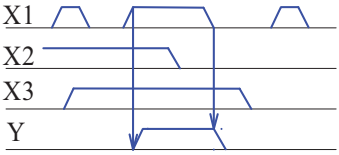
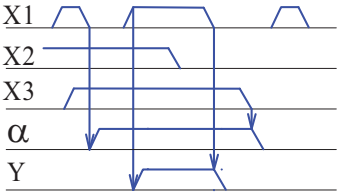
В табл. 2.14 приведены фрагменты элементарных циклограмм, решение которых осуществляется путем введения в исходную циклограмму промежуточных сигналов.

Таблица 2.14

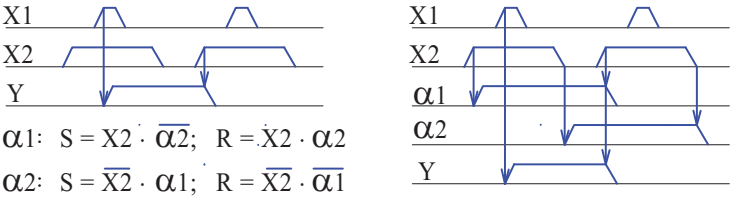
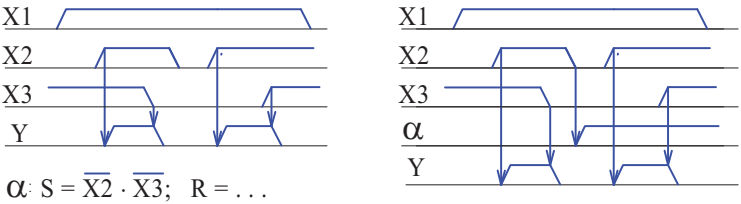
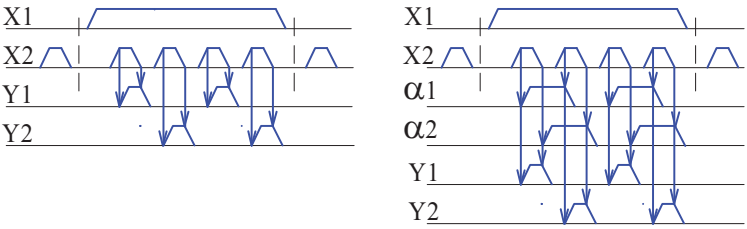
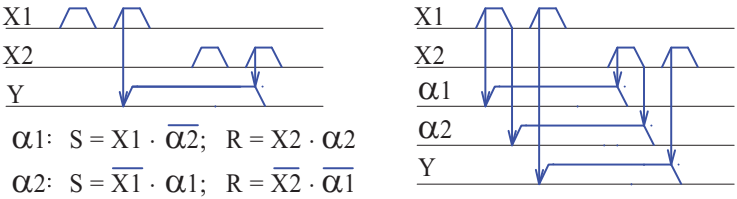
Примеры решений с введением промежуточных сигналов

№	Циклограммы: Исходная ⇒ Преобразованная	
1	2	3
1	 α: S = X; R = ... $Y = \overline{X} \cdot \alpha$	
2	 α: S = X1; R = ... $Y = X1 + X2 \cdot \alpha$	
3	 α: S = X1; R = X3 Y: S = X2 · α; R = X2 · $\overline{\alpha}$	
4	 $\alpha 1$: S = X1; R = X3 $\alpha 2$: S = X3 · $\overline{\alpha 3}$; R = X3 · $\alpha 3$ $\alpha 3$: S = $\overline{X3}$ · $\alpha 2$; R = $\overline{X3}$ · $\overline{\alpha 2}$ Y: S = $\overline{X1}$ · $\alpha 3$; R = X3 · $\alpha 3$	

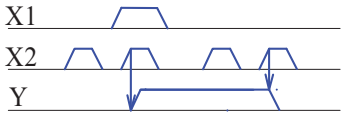
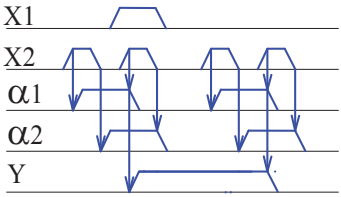
Продолжение таблицы 2.14

№	Циклограммы: Исходная ⇒ Преобразованная	
1	2	3
5	 $\alpha 1$: $S = X3 \cdot \overline{\alpha 2}$; $R = X3 \cdot \alpha 2$ $\alpha 2$: $S = \overline{X3} \cdot \alpha 1$; $R = \overline{X3} \cdot \overline{\alpha 1}$ $\alpha 3$: $S = X3 \cdot \alpha 2$; $R = \ldots$ $Y = \overline{X1} \cdot \overline{\alpha 3} \cdot X2$	
6	 α: $S = X \cdot \overline{Y}$; $R = \overline{X} \cdot \overline{Y}$ Y: $S = X \cdot \overline{\alpha}$; $R = X \cdot \alpha$	
7	 α: $S = X2$; $R = \overline{X3} \cdot \overline{X2}$ Y: $S = \overline{X1} \cdot X2$; $R = X3 \cdot \overline{\alpha}$	
8	 α: $S = \overline{X1} \cdot X2 \cdot X3$; $R = \overline{X3}$ $Y = X1 \cdot \alpha$	

Продолжение таблицы 2.14

№	Циклограммы: Исходная ⇒ Преобразованная
1	2 | 3
9	 $\alpha 1$: $S = X2 \cdot \overline{\alpha 2}$; $R = \overline{X2} \cdot \alpha 2$ $\alpha 2$: $S = \overline{X2} \cdot \alpha 1$; $R = \overline{X2} \cdot \overline{\alpha 1}$ Y: $S = X1 \cdot \alpha 1$; $R = X3 \cdot \alpha 2$
10	 α: $S = \overline{X2} \cdot \overline{X3}$; $R = \ldots$ Y: $S = X2 \cdot X3 \cdot \overline{\alpha} + X2 \cdot \overline{X3} \cdot \alpha = X2 \cdot (X3 \neq \alpha)$
11	 $\alpha 1$: $S = X2 \cdot \overline{\alpha 2} \cdot X1$; $R = \overline{X2} \cdot \alpha 2 \cdot X1$ $\alpha 2$: $S = \overline{X2} \cdot \alpha 1 \cdot X1$; $R = \overline{X2} \cdot \overline{\alpha 1} \cdot X1$ $Y1 = X2 \cdot \overline{\alpha 2} \cdot X1$; $Y2 = X2 \cdot \alpha 2 \cdot X1$
12	 $\alpha 1$: $S = X1 \cdot \overline{\alpha 2}$; $R = X2 \cdot \alpha 2$ $\alpha 2$: $S = \overline{X1} \cdot \alpha 1$; $R = \overline{X2} \cdot \overline{\alpha 1}$ Y: $S = X1 \cdot \alpha 2$; $R = X2 \cdot \overline{\alpha 2}$

Окончание таблицы 2.14

№	Циклограммы	
	Исходная ⇒	Преобразованная
1	2	3
13	  $\alpha 1$: $S = X2 \cdot \overline{\alpha 2}$; $R = X2 \cdot \alpha 2$ $\alpha 2$: $S = \overline{X2} \cdot \alpha 1$; $R = \overline{X2} \cdot \overline{\alpha 1}$ Y: $S = X2 \cdot \alpha 2 \cdot X1$; $R = X2 \cdot \alpha 2 \cdot \overline{X1}$	

Приведенные в таблице решения во многих случаях не являются единственно возможными. Читателю предлагается самостоятельно выполнить их анализ и попытаться найти другие варианты. Критерий к решению – последовательное раскрытие обнаруженных при анализе неопределенностей решения.

Приведем некоторые комментарии.

Циклограмма 3. Очевидно, что задача может быть решена по аналогии предыдущего примера с управлением промежуточным сигналом задними фронтами сигнала X_2. Однако наличие в исходной циклограмме сигналов X_1 и X_3 дает более простое решение.

Циклограммы 4 и 5. На них приведены два решения одной и той же задачи. Рабочая (с промежуточными сигналами) циклограмма 5 отличается от циклограммы 4 тем, что формирует выходной сигнал Y комбинационными способами, однако его длительность на величину одного фронта больше заданного. Допустимо это или нет решается в каждом конкретном случае индивидуально (как правило, допускается). Кроме того, рабочая циклограмма 5 не допускает повторного цикла, так как не предусмотрено выключение промежуточного сигнала α_3. При необходимости это легко реализовать, сформировав сигнал сброса $R\ (\alpha_3) = \overline{X}_2$.

Циклограмма 12. Возможно решение с двумя счетными триггерами, управляемыми, соответственно, сигналами X_1 и X_2. Очевидно, что это решение сложнее, так как потребует не два промежуточных сигнала, а четыре.

2.10. Определение минимального числа логических переменных, необходимых для перевода нереализуемых условий работы в реализуемые

Как уже ясно из анализа циклограмм табл. 2.14, нереализуемые условия работы схемы переводятся в реализуемые путем введения в циклограмму определенного числа дополнительных сигналов. Естественно, это необходимо сделать путем введения их минимально возможного числа. Точное определение минимального числа дополнительных сигналов практически невозможно. Рассмотрим методику, предложенную в [63] и основанную на определении верхней ($n_в$) и нижней ($n_н$) оценок числа элементов, необходимых для построения схемы. Верхняя оценка $n_в$ означает, что схема может быть заведомо реализуема с числом элементов, не превосходящих $n_в$. Нижняя оценка $n_н$ означает, что для данных условий, может быть, можно построить схему с числом элементов равным $n_н$. Вычисление оценок $n_в$ и $n_н$ позволяет существенно облегчить синтез схемы, так как проектант заранее знает необходимое число элементов $n_н \le n \le n_в$.

Анализ выполняется в следующей последовательности:

1. Определяется число **совпадающих тактов** m_i для каждой i-й комбинаций исходных входных сигналов.

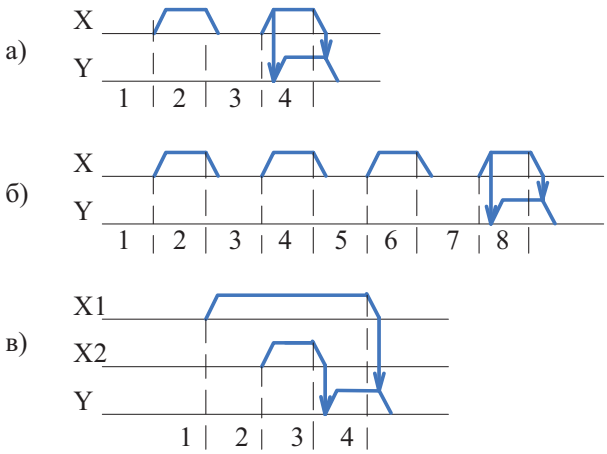


Рис. 2.35. Примеры совпадающих тактов

Совпадающими тактами называются такие такты, в которых состояния выходных сигналов различны, а состояние входных сигналов одинаково. Например, в циклограмме рис. 2.35, *а* совпадающими тактами являются такты 2 и 4, рис. 2.35, *б* – 2, 4, 6 и 8, рис. 2.35, *в* – 2 и 4.

Наличие совпадающих тактов говорит о нереализуемости схемы.

К числу совпадающих тактов относятся также и такты, образующие так называемую **существенную последовательность** (рис. 2.36), т. е. входящие в последовательность входных сигналов счетной схемы.

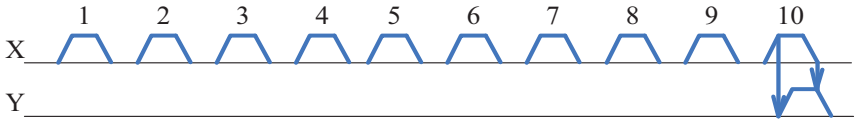


Рис. 2.36. Существенная последовательность

При определении m_i подсчитывается число всех комбинаций входных сигналов, например, для рис. 2.35, *в* имеется три комбинации входных логических переменных X_1 и X_2. Это 00, 10 и 11. Каждая из комбинаций 00 и 11 встречается лишь в одном такте, следовательно, $m_{00} = m_{11} = 1$. Комбинация 10 встречается в двух тактах, причем эти такты являются совпадающими, поэтому $m_{10} = 2$. Комбинация 01 вообще отсутствует, поэтому $m_{01} = 0$.

Итак: $m_{00} = 1$; $m_{11} = 1$; $m_{10} = 2$; $m_{01} = 0$.

2. Из всех определенных значений m выбирается наибольшее число, которое обозначается m_{max}. В нашем случае $m_{max} = m_{10} = 2$.

3. Вычисляется **нижняя** оценка $n_н$ по формуле $2^{n}_{н} \geq m_{max}$.

4. Вычисляется **верхняя** оценка $n_в$ по формуле $2^{nв} \geq 2_{m\ max} - 1$, т. е. $2^{nв} \geq 3$, откуда $n_в = 2$.

Из приведенных формул видно, что верхняя и нижняя оценки могут отличаться друг от друга только на один элемент.

Приведем еще один характерный пример. Пусть необходимо выполнить схему счета импульсов с выдачей выходного сигнала на десятом входном импульсе (рис. 2.36).

Входная комбинация X = 0 встречается 10 раз во всех нечетных тактах, но так как выходной сигнал здесь одинаков и всегда равен Y = 0, то $m_0 = 1$.

Комбинация X = 1 также встречается 10 раз во всех четных тактах, но здесь мы имеем существенную последовательность с разными значениями выходных сигналов, относящуюся к совпадающим тактам, так как $m_1 = 10$, откуда $m_0 = 1$; $m_1 = 10$; $m_{max} = m_1 = 10$.

Нижняя оценка $2^{nн} \geq m_{max}$, $2^{nн} \geq 10$ и $n_н = 4$.

Верхняя оценка $2^{nв} \geq 2_{m\ max} - 1$, $2^{nв} \geq 19$, $n_в = 5$, т. е. схема реализуется на 4–5 элементах.

Следует сказать, что при синтезе реальных схем возникает очень много вариантов включения дополнительных элементов, и однозначно сказать, какой

из них ведет к их минимальному числу, заранее невозможно. Поэтому вычисленные оценки позволяют лишь оценить, на сколько выбранный вариант близок к оптимальному. Например, если счетчик реализован на 6 элементах, то этот результат можно считать приемлемым, а если на 8–9 элементах, то следует попытаться найти другое решение. В этом мы убедимся при синтезе счетчиков и регистров.

2.11. Синтез временных логических схем на основе метода циклограмм

Временные логические функции осуществляют преобразование рабочих сигналов по длительности, а также генерирование одиночных или последовательных импульсов, и широко применяются в промышленной электроавтоматике, в том числе и станкостроении.

Базовые временные логические функции (табл. 2.15) осуществляют:

- задержку переднего, заднего или обоих фронтов входного сигнала X(t);
- формирование импульса заданной длительности при включении, выключении или по обоим фронтам входного сигнала;
- генерирование одиночного импульса заданной длительности или последовательности импульсов заданной частоты и скважности.

Из табл. 2.15 видно, что в формировании выходных логических функций $Y_1 \ldots Y_5$ участвует сигнал $X(t-\Delta\tau)$, задержанный на время $\Delta\tau$ по отношению к выходному сигналу X(t), имеющий чисто теоретический характер. В применяемой на практике элементной базе такой сигнал отсутствует. Базовыми элементами для временных логических функций, как правило, являются:

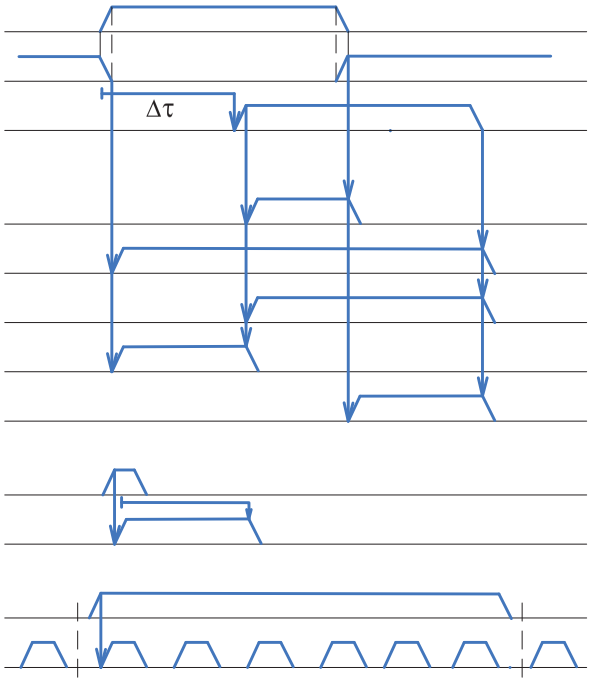
1. Для жесткой бесконтактной логики – типовые схемы задержки переднего или заднего фронта, выполненные на базе конденсаторных задержек.
2. Для релейных систем – реле времени с замыкающимися и размыкающимися с выдержкой контактами.
3. Для программируемых логических контроллеров (ПЛК) – выдержка времени при включении.

Синтез и анализ циклограмм для каждой логической переменной следует проводить по уже известному нам алгоритму:

1. Вычерчивание **исходной** циклограммы в соответствии с поставленной задачей и ее анализ (исходная циклограмма включает только физически существующие сигналы).
2. Поиск комбинационного решения, однозначно описывающего синтезируемый сигнал по исходной циклограмме уравнениями алгебры логики. Если решения нет, то переходим к 3.

Таблица 2.15

Базовые временные логические функции

Функция	Циклограмма	Уравнение
Входные сигналы: $X(t)$ $\overline{X(t)}$ $X(t - \Delta\tau)$ Выходные сигналы: Задержка переднего фронта Задержка заднего фронта Задержка обоих фронтов Импульс по переднему фронту Импульс по заднему фронту Одновибратор Управляемый или неуправляемый генератор		$Y1 = X(t) \cdot X(t - \Delta\tau)$ $Y2 = X(t) + X(t - \Delta\tau)$ $Y3 = X(t - \Delta\tau)$ $Y4 = X(t) \cdot \overline{X(t - \Delta\tau)}$ $Y5 = \overline{X(t)} \cdot X(t - \Delta\tau)$

3. Поиск решения с применением типовой памяти типа RS-триггера. При этом синтезу по циклограмме подвергаются только сигналы установки памяти S и ее сброса R. Если решения нет, то: переходим к 4.

4. Введение в исходную циклограмму промежуточного сигнала α в целях раскрытия неопределенностей, не позволяющих найти комбинационное решение или решение с типовой памятью и переход к пункту 2 настоящего алгоритма; программа с веденными в нее промежуточными сигналами называется **рабочей.**

5. Минимизация полученных логических уравнений и их приведение к виду элементной базы, на которой реализуется практическая схема.

6. Вычерчивание принципиальной схемы или алгоритма для программирования ПЛК.

При этом следует также применять понятные для чтения условные обозначения, например:

$\Delta\tau$ – общее обозначение выдержки времени;

$\Delta\tau$ ($\uparrow$) – задержка переднего фронта;

$\Delta\tau$ ($\downarrow$) – задержка заднего фронта;

ТМ (Timer) – общее обозначение реле времени (таймера) или ВЭ (временной элемент) или РВ (реле времени);

$\Delta\tau$ ($\uparrow\downarrow$) – задержка переднего и заднего фронтов;

ТМ ($\uparrow$) – контакт реле с задержкой переднего фронта (момента замыкания);

ТМ ($\downarrow$) – контакт реле с задержкой заднего фронта (момента размыкания) и т. д.

При использовании программируемых контроллеров следует применять условные обозначения, определенные синтаксисом языка.

Варианты начертания циклограмм для временных схем приведены на рисунке 2.37.

В связи с тем, что время включенного состояния катушки реле и его контактов не совпадают, то возможны три варианта изображения циклограммы:

а) раздельное изображение катушки ТМ и замыкающего контакта ТМ ($\uparrow$). Сигнал ТМ ($\uparrow$) синтезу не подлежит (рис. 2.37, *а*);

б) отображение на циклограмме только сигнала, относящегося к катушке ТМ. Момент замыкания катушки реле изображается вертикальной стрелкой с индексом $\Delta\tau$ (рис. 2.37, *б*);

в) отображение на циклограмме только сигнала, относящегося к контакту реле ТМ ($\uparrow$). Время включенного состояния катушки реле ТМ определяется горизонтальной стрелкой с индексом $\Delta\tau$ (рис. 2.37, *в*).

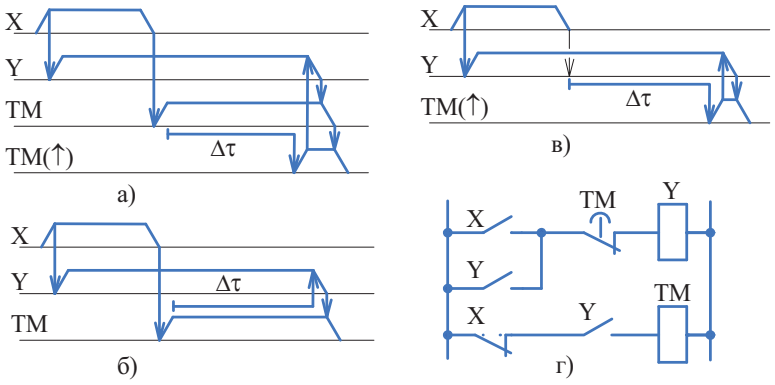


Рис. 2.37. Варианты начертания циклограммы (а, б, в) и схема (г) временного элемента

Анализ всех трех циклограмм дает одинаковый результат:

– реле Y: S = X; R = TM (↑) и $\overline{R} = \overline{TM(\uparrow)}$;

– реле $TM = \overline{X} \cdot Y$.

Многолетний опыт автора при проектировании различных схем на основе метода циклограмм показал, что при синтезе временных схем на циклограмме целесообразно показывать **раздельные** рисунки для синтезируемого сигнала ТМ, отражающего катушку реле или сигнал запуска таймера ПЛК и выходного сигнала реле времени или таймера ПЛК ТМ (Δτ), используемого при синтезе других логических переменных, т. е. вариант **а)**. Еще раз подчеркнем, что это связано с тем, что в отличие от обычной логической функции входные и выходные сигналы временной функции не совпадают по времени и их раздельное изображение значительно упрощает синтез и анализ циклограмм.

Там, где результат очевиден, можно применять циклограммы вариантов *б* и *в*.

В практической инженерной деятельности следует иметь набор проверенных типовых схем на различной элементной базе и на различный диапазон выдержек времени. Подобные схемы приводятся в справочниках и руководящих материалах на элементную базу, там же даются данные по максимально допустимой выдержке времени в той или иной схеме включения.

Ниже, в релейном варианте, приводится синтез типовых временных схем (алгоритмов для PLC) по изложенной выше методике.

Задержка переднего фронта входного сигнала X

В качестве базовой примем циклограмму задержки при включении, отражающую реальную работу реле времени (таймера) электронного типа, например серии ВЛ, и функции таймера любого программируемого контроллера.

Анализ исходной циклограммы (рис. 2.38, *а*), состоящей из входного (X) и выходного (Y) сигналов, показывает, что задача не решается ни комбинационным путем, так как нельзя написать однозначное уравнение Y= f(X); ни при помощи типового элемента памяти, так как нельзя написать однозначные уравнения S = f(X, Y) и R = f(X, Y) для сигналов включения и выключения памяти. Для раскрытия неопределенности принимается решение о введении промежуточного сигнала, в качестве которого целесообразно использовать временной сигнал с задержкой ТМ.

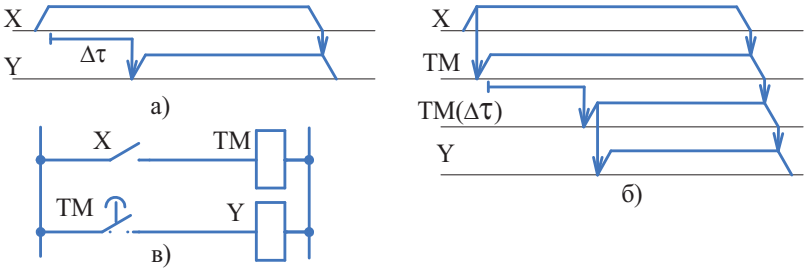


Рис. 2.38. Исходная (а), рабочая (б) циклограммы и схема (в) задержки переднего фронта

Анализ рабочей циклограммы (рис. 2.38, *б*) показывает, что сигнал включения катушки таймера ТМ повторяет входной сигнал X, а выходной сигнал схемы Y повторяет выходной сигнал таймера ТМ (Δτ), т. е. ТМ = X и Y = TM(Δτ).

Принципиальная релейно-контактная схема, являющаяся одновременно алгоритмом для написания программы ПЛК, приведена на рис. 2.38, *в*.

Задержка заднего фронта входного сигнала X

Здесь и дальше будем проводить **только** рабочие циклограммы с уже введенными промежуточными сигналами, необходимыми для получения однозначного решения задачи, сформированной исходной циклограммой (сигналы X и Y).

На рис. 2.39, *а* приведено решение с реле времени, имеющим задержку при *включении* питания.

Входной сигнал Y не описывается комбинационным уравнением алгебры логики, так как нет однозначного решения Y = f(X). Принимаем решение, применить типовой элемент памяти.

Сигнал включения памяти S = X.

Сигнал включения памяти R = TM(Δτ). Поскольку типовая память на реле сбрасывается инверсным сигналом, инвертируем сигнал сброса $\overline{R} = \overline{TM(\Delta\tau)}$.

Сигнал включения катушки таймера описывается комбинационным уравнением $TM = \overline{X} \cdot Y$.

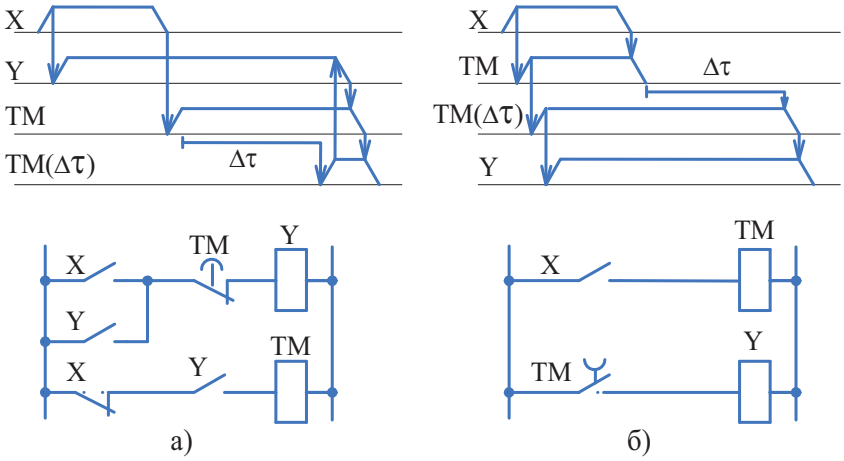


Рис. 2.39. Циклограммы и схемы задержки заднего фронта при использовании реле с выдержкой при включении (а) и при отключении (б)

На рис. 2.39, *б* приведено решение с реле времени, имеющим задержку при *выключении* питания.

Логические уравнения очевидны: TM1 = X и Y = TM1(↓).

Формирование импульса заданной длительности по переднему фронту входного сигнала X (рис. 2.40)

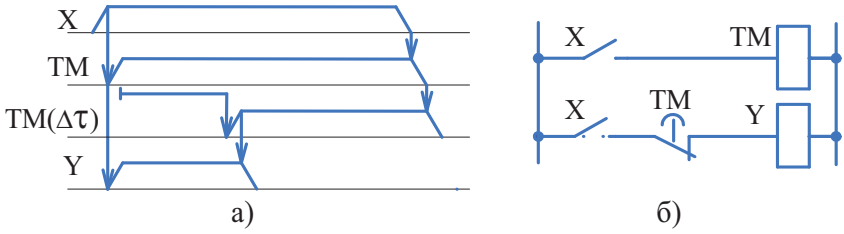


Рис. 2.40. Циклограмма (а) и схема (б) формирования импульса по переднему фронту

Сигнал включения таймера повторяет входной сигнал, т. е. TM = X.

Выходной сигнал $Y = X \cdot \overline{TM(\Delta\tau)}$.

Формирование импульса заданной длительности по заднему фронту входного сигнала X

1. Формирование импульса с использованием реле с выдержкой времени *при включении* (рис. 2.41).

Для раскрытия неопределенности в исходную циклограмму введено два промежуточных сигнала α и ТМ.

Логические уравнения решения задачи:

– реле α – память. S = X, R = ТМ(Δτ) и $\overline{R} = \overline{ТМ(\Delta\tau)}$;

– реле ТМ = $\overline{X} \cdot \alpha$;

– реле $Y = \overline{X} \cdot \alpha$

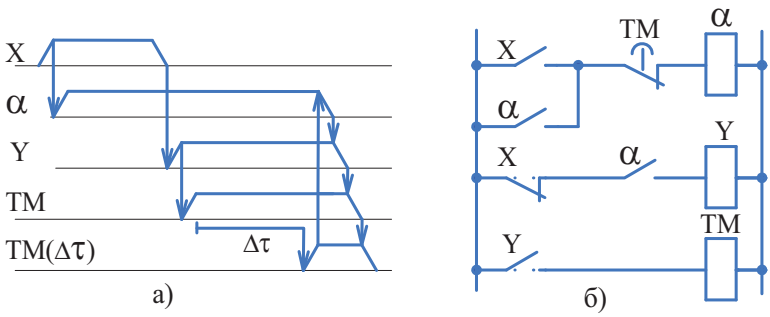


Рис. 2.41. Циклограмма (а) и схема (б) формирования импульса по заднему фронту на реле с выдержкой при включении

2. Формирование импульса с использованием реле с выдержкой времени *при отключении* (рис. 2.42).

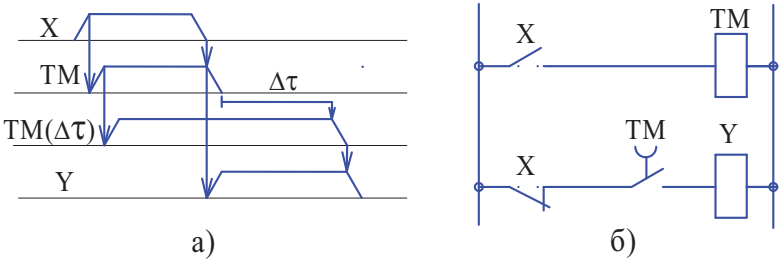


Рис. 2.42. Циклограмма (а) и схема (б) формирования импульса по заднему фронту на реле с выдержкой при отключении

Логические уравнения решения задачи:

– реле TM1 = X;

– реле α = TM1($\Delta\tau\downarrow$);

– реле $Y = \overline{X} \cdot \alpha$.

Формирование импульсов равной длительности по обоим фронтам входного сигнала

Задача легко решается при использовании реле времени с выдержкой при включении (рис. 2.43).

Выходной сигнал Y описывается функцией неравнозначности между входным сигналом X и промежуточным сигналом α, т. е.

$$Y = (X \neq \alpha) = X \cdot \overline{\alpha} + \overline{X} \cdot \alpha.$$

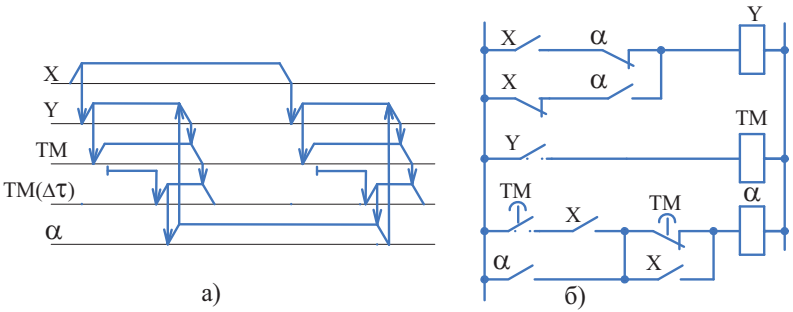


Рис. 2.43. Формирование импульсов равной длительности по обоим фронтам входного сигнала

Сигнал включения катушки таймера есть функция повторения выходного сигнала, т. е. TM = Y.

Промежуточный сигнал α реализуется при помощи типовой памяти

$$S = TM(\Delta\tau) \cdot X, \; R = TM(\Delta\tau) \cdot \overline{X} \text{ или } \overline{R} = \overline{TM(\Delta\tau)} + X.$$

Формирование импульсов разной длительности по обоим фронтам входного сигнала

Задача может быть решена как с использованием реле разного типа, т. е. при формировании первого импульса на реле с выдержкой времени при включении, а второго – на реле с выдержкой при отключении; так и на реле одного типа – с выдержкой только при включении. Приведем второй вариант (рис. 2.44).

Логические уравнения решения задачи:

– выходное реле: $Y = (X \neq \alpha) = X \cdot \overline{\alpha} + \overline{X} \cdot \alpha$;

– первый таймер TM1 $= Y \cdot X$;

– промежуточное α – память: $S = TM1(\Delta\tau)$ и $\overline{R} = \overline{TM2(\Delta\tau)}$;

– второй таймер $TM2 = Y \cdot \overline{X}$.

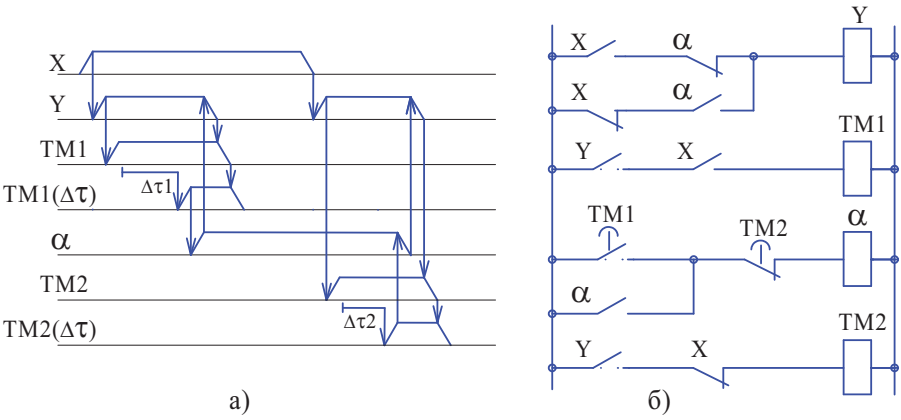


Рис. 2.44. Формирование импульсов по обоим фронтам входного сигнала с использованием реле одного типа

Задержка обоих фронтов входного сигнала X на одинаковое время, т. е. сдвиг входного сигнала вправо (рис. 2.45)

Сигнал включения катушки таймера описывается функцией неравнозначности входного и выходного сигналов, т. е.

$$TM = (X \neq Y) = X\overline{Y} + \overline{X}Y.$$

Выходной сигнал Y реализуется при помощи типовой ячейки памяти:

– включение $S = TM(\Delta\tau) \cdot X$;

– выключение $R = TM(\Delta\tau) \cdot \overline{X}$ или $\overline{R} = \overline{TM(\Delta\tau)} + X$.

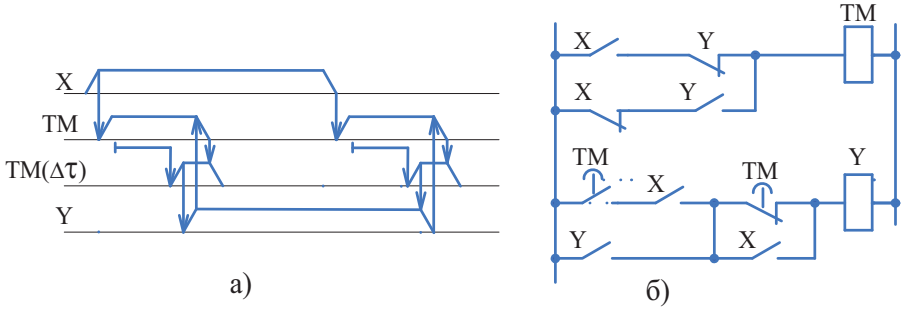


Рис. 2.45. Сдвиг входного сигнала вправо

Задержка обоих фронтов входного сигнала X на разное время (рис. 2.46)

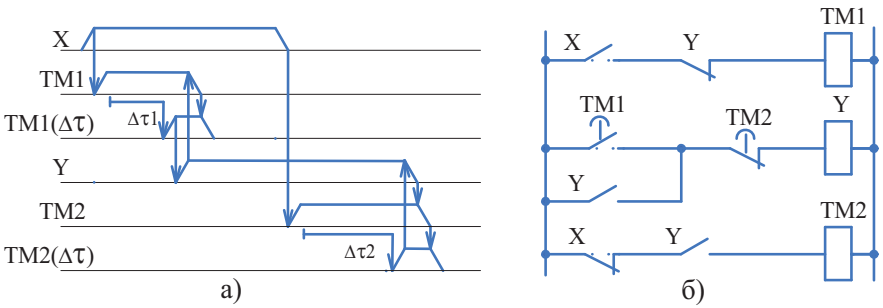


Рис. 2.46. Задержка обоих фронтов входного сигнала на разное время

Задача решается при помощи двух таймеров с различными уставками выдержки времени. Логические уравнения решения задачи:

– первый таймер $TM1 = X \cdot \overline{Y}$;

– выходное реле Y – память: $S = TM1(\Delta\tau)$ и $\overline{R} = \overline{TM2(\Delta\tau)}$;

– второй таймер $TM2 = \overline{X} \cdot Y$.

Одновибратор, т. е. схема формирования выходного импульса заданной длительности по короткому входному импульсу X (рис. 2.47).

Задача решается путем введения в исходную циклограмму промежуточного временного сигнала TM1. Логические уравнения:

– выходной сигнал Y-память: $S = X$ и $\overline{R} = \overline{TMl(\Delta\tau)}$;

– реле времени TM1 = Y.

В случае, если используемая элементная база не позволяет реализовать заданное время включения выходного реле, то следует применить каскадное включение таймеров.

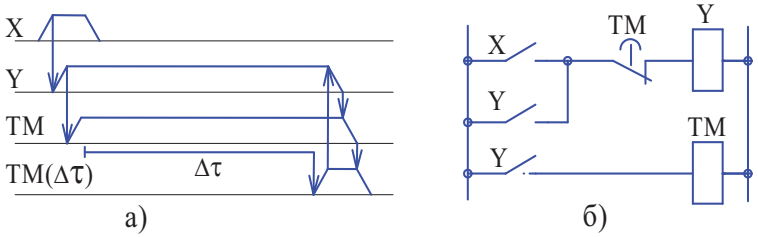


Рис. 2.47. Одновибратор

Каскадное включение таймеров (рис. 2.48)

Синтез схемы очевиден:

– Y: S = X; R = TM2 (↑) и $\overline{R} = \overline{TM2(\uparrow)}$;

– TM1 = Y;

– TM2 = TM1(↑).

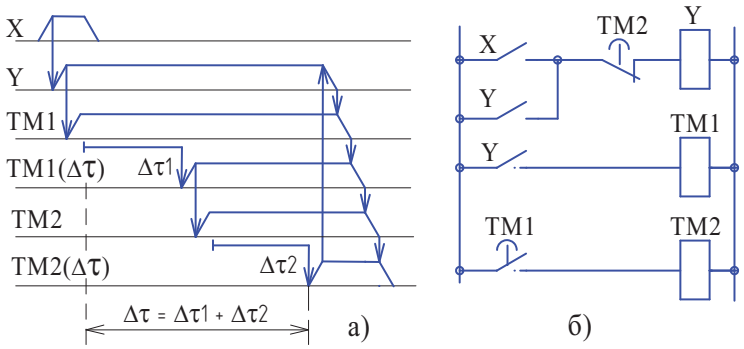


Рис. 2.48. Циклограмма (а) и схема (б) каскадного включения таймеров

Управляемый генератор импульсов (рис. 2.49)

Включение генератора осуществляется кнопкой «Пуск» (сигнал КнП), а выключение кнопкой «Стоп» (сигнал КнС). Кнопки управляют схемой памяти (на схеме не показано) промежуточного сигнала X, определяющего общее время работы генератора. Основу автономного генератора составляют два таймера TM1 и TM2, соответственно, задающих время длительности выходного сигнала Y и время паузы.

Логические уравнения:

– промежуточный сигнал X – память: $S = КнП$ и $\overline{R} = \overline{КнС}$;

– первый таймер $TM1 = X \cdot \overline{TM2(\Delta\tau)}$;

– второй таймер $TM2 = X \cdot TM1(\Delta\tau)$;

– выходной сигнал $Y = X \cdot \overline{TM1(\Delta\tau)} \cdot TM2(\Delta\tau)$.

Очевидно, что, изменив рабочую циклограмму соответственно поставленной задаче, легко синтезировать генератор с фиксированным периодом и регулируемой скважностью за счет изменения длительности импульса или времени паузы.

Дополнив генератор счетчиком сигналов Y, можно синтезировать генератор заданного числа импульсов и многие другие схемы.

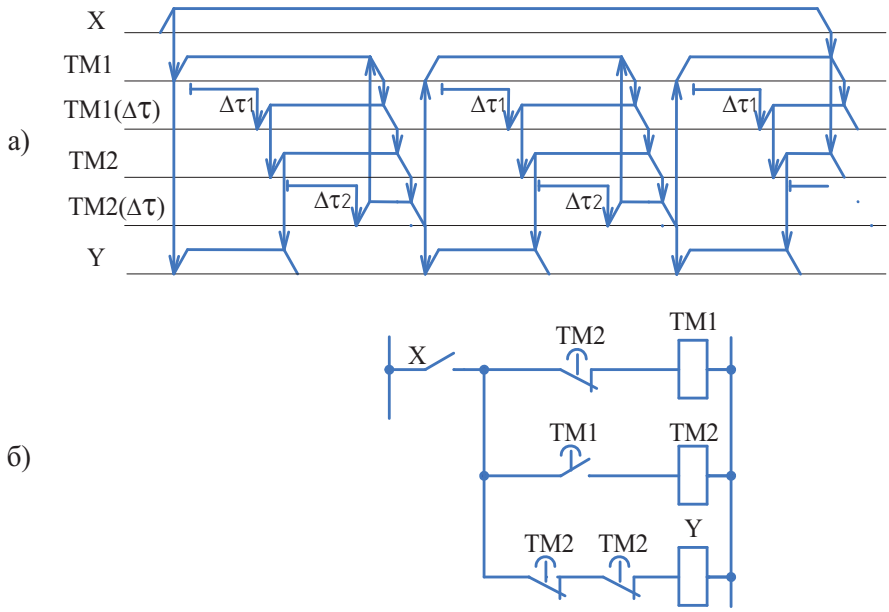


Рис. 2.49. Управляемый генератор импульсов

Синтез приведенных в разделе типовых временных схем, конечно, не исчерпывает все задачи, которые могут возникнуть при проектировании промышленной электроавтоматики, однако автор в очередной раз осмеливается утверждать, что предложенный инженерный метод на основе циклограмм и аппарата алгебры логики позволяет легко решать любые задачи дискретной автоматики при любом числе логических переменных и на любой элементной базе, что он гораздо эффективнее методов типа карт Карно и диаграмм Вейча и гораздо практичнее теории дискретных автоматов.

Все выше приведенные решения пригодны для любой элементной базы, для любых языков программируемой логики.

2.12. Комплексное применение методики синтеза

Ниже излагается методология и последовательность проектирования реальных алгоритмов станочной электроавтоматики. В качестве базовой системы ЧПУ и встроенного в нее программируемого контроллера взята серия NC производства Санкт-Петербургской фирмы «Балт-Систем».

2.12.1. Стадии проектирования электроавтоматики станка с устройством ЧПУ серии NC

I. Выбрать основные комплектующие изделия:

1. Устройство ЧПУ.
2. Электроприводы подачи и главного движения.
3. Датчики обратной связи по положению.
4. Пульты управления и штурвалы.
5. Низковольтную аппаратуру.
6. Прочее.

II. Разработать принципиальную схему и перечень элементов.

III. Разработать проект привязки системы ЧПУ к станку.

1. Файловая структура (Система параметров)
   - системный файл FCRSYS;
   - характеризации осей FXCFIL;
   - характеризации процесса PGCFIL;
   - характеризации логики IOCFIL;
   - файл назначения горизонтальных клавиш УЧПУ LGCMNU;
   - файл пиктограмм вертикальных клавиш CNC.USR;
   - файл перемещения осей из электроавтоматики FILMOVE;
   - файл управления трехбуквенными кодами FILCMD;
   - файл корректоров FILCOR;
   - файл начальных точек FILOR.
2. Файл управления виртуальными клавишами.
3. Файл электроавтоматики и диагностики.

2.12.2. Типовой состав электроавтоматики

В общем случае, в состав электроавтоматики станка входят следующие разделы:

1. Включение станка и гидравлики.
2. Формирование режимов работы.
3. Управление электроприводами подачи:

- включение приводов подачи и постановка их на слежение. Сюда же входит управление тормозами осей координат и тормозов, встроенных в двигатели;
- ручное управление перемещением координат (безразмерное и фиксированное);
- управление штурвалом;

- управление корректором подачи F%;
- формирование сигнала «Стоп подачи»;
- обработка сигналов ограничения перемещений (основное, аварийное, программное);
- формирование цикла выхода в ноль;
- сход с конечников.

4. Управление вспомогательными приводами и механизмами, например:
   а) общее:
      - смазкой;
      - охлаждением;
   б) для фрезерных станков:
      - консолью;
      - поперечиной;
      - зажимом детали;
      - зажимом инструмента и др.;
   в) для токарных станков:
      - патроном;
      - пинолью;
      - задней бабкой;
      - люнетом;
      - резцедержкой и др.
5. Дешифрирование М-функций.
6. Управление приводом главного движения:
   - дешифрирование кода скорости;
   - управление корректором скорости S%;
   - формирование команд включения;
   - формирование цикла переключения коробки скоростей;
   - включение и останов привода.
7. Управление инструментальными магазинами.
8. Управление манипуляторной сменой инструмента. При необходимости, формирование пошагового и обратного циклов, а также сменой корректоров.
9. Управление сменой столов-спутников.
10. Формирование ответных сигналов или запрета ввода кадра.
11. Формирование сигналов остановов.
12. Пуск и стоп программы.
13. Сброс электроавтоматики.
14. Прямое управление электромагнитами.
15. Блокировки и диагностика.

16. Начальные установки и др.

Примечание. Общую программу электроавтоматики условно можно разбить на три части:

– подпрограмма оболочки управления (виртуальных клавиш);
– основная программа;
– подпрограмма диагностики.

Естественно, что может быть и другой подход к структуре программы. Например, можно интегрировать диагностику в основное тело программы. Это очень красиво, если программа уже отлажена. Но ведь в процессе работы всегда возникает изрядное количество ошибок или описок. Иногда из-за отсутствия информации или непонимания принципов работы механизма программа вообще пишется неправильно. В этом случае диагностика будет мешать и на период отладки ее придется подключать по частям или отключать вообще. Так работать можно, если никто не торопит и нет ограничения по времени. По мнению автора, значительно удобнее вынести диагностику в отдельную подпрограмму, пронумеровав сообщения.

Подробная процедура разработки перечисленных выше разделов электроавтоматики будет приведена во втором и третьем томах настоящей книги.

2.12.3. Перечень решаемых вопросов по системе ЧПУ

1. Конфигурация устройства, состав блоков и их соединение между собой, подключение приводов и датчиков, подключение дискретных входов и выходов, подключение периферийных устройств.
2. Состав необходимых периферийных устройств и программных средств.
3. Адресация программно-доступных аппаратных средств.
4. Синтаксис языка электроавтоматики.
5. Перечень и назначение обменных сигналов.
6. Процедура формирования сообщений.
7. Типовые программы и алгоритмы.
8. Файловая структура системы.
9. Система параметров.
10. Привязка корректоров.
11. Элементы технологического программирования.
12. Процедурные вопросы работы с устройством.
13. Специальные требования.

2.12.4. Загрузка устройства ЧПУ (ООО «Балт-Систем»)

1. В зону памяти MP0 загрузить вновь созданный системный файл FCRSYS или в имеющийся файл ввести имена следующих рабочих файлов:
   – AXCF/MPxx;
   – PGCF/MPxx;
   – IOCF/MPxx;
   – MES/MPxx;
   – ……
2. В зону памяти MP0 загрузить файл LGCMNU.
3. В зону памяти MP0 загрузить файл CNC.USR (только через NORTON).
4. В выбранную рабочую зону загрузить файл электроавтоматики ЭЛА/MPxx.
5. В выбранную рабочую зону загрузить файл виртуальных клавиш ВКл/MPxx.
6. Установить СРЕДУ:
   – файл электроавтоматики (1-я страница);
   – файл виртуальных клавиш (1-я страница);
   – объем памяти электроавтоматики (2-я страница).
7. Установить точку с запятой в инструкции ALM файла IOCFIL.
8. Выключить УЧПУ.
9. Включить УЧПУ и откомпилировать программу электроавтоматики.
10. Исправить синтаксические ошибки, отладить и вновь откомпилировать программу.
11. Убрать точку с запятой в инструкции ALM файла IOCFIL, выключить УЧПУ.
12. Включить УЧПУ, произойдет автоматическая загрузка на рабочий режим.

2.12.5. Организация циклов управления

При разработке алгоритмов управления сложными механизмами, например, инструментальными магазинами, устройствами смены инструментов и спутников, резцедержками, автоматизированной задней бабки, патроном, пинолью и т. д., полный цикл работы которых включает несколько операций, целесообразно формировать специальный сигнал, определяющий «тело цикла». Действие этого сигнала, назовем его «Цикл», начинается с команды включения цикла управления механизмом и заканчивается контролем выполнения последней операции.

На рис. 2.50 приведена теоретическая **учебная** циклограмма, являющаяся **частью** общего цикла смены инструментов многооперационного станка с ЧПУ. Предполагается, что манипулятор одной клешней уже захватил следующий по технологии обработки инструмент в магазине, а второй клешней захватил инструмент, находящийся в шпинделе. Это стартовые условия нашего цикла. Цикл выполняется по команде от кнопки в наладочном (ручном) режиме.

В общем случае алгоритм управления циклом и его циклограмма включают четыре части:

1. Проверка начальных условий, определяющих можно ли начинать Цикл.
2. Активизация командного сигнала начала Цикла.
3. Рабочее тело Цикла.
4. Формирование сигнала «Сброс Цикла», определяющего его завершение и приводящего схему (алгоритм) в начальное исходное положение.

Проверку *начальных условий* (1) рекомендуется выполнять в несколько шагов:

- формирование сигнала «Запрет», объединяющий все несовместимые с данным циклом условия. В нашем это режим Автомат, Включенное состояние шпинделя, Режим прямого управления и др.;
- проверка исходного положения механизмов, связанных с работой цикла. В нашем случае манипулятор должен быть вдвинут и находиться в вертикальном положении, клешня манипулятора должна захватывать шейку инструмента в шпинделе, т. е. нужно контролировать ее координаты, инструмент в шпинделе должен быть зажат и др. Назовем этот сигнал Кисх (контроль исходного). Если контролируемых сигналов много, то их следует разбить на группы, например, Кисх (руки), Кисх (шпинделя), Кисх (координаты) и т. д. В этом случае легче выполнять наладочные работы;
- формирование сигнала разрешения работы цикла РЦ, активизируемого при отсутствии сигнала «Запрет», наличии всех сигналов контроля исходного положения и установки рабочего режима, т. е.

$$РЦ = РРучн \cdot /Запрет \cdot Кисх1 \cdot Кисх2.$$

Активизация начала работы цикла (2) выполняется при условии наличия сигнала разрешения цикла РЦ и поступлении командного сигнала, в данном случае нажатии кнопки «Цикл».

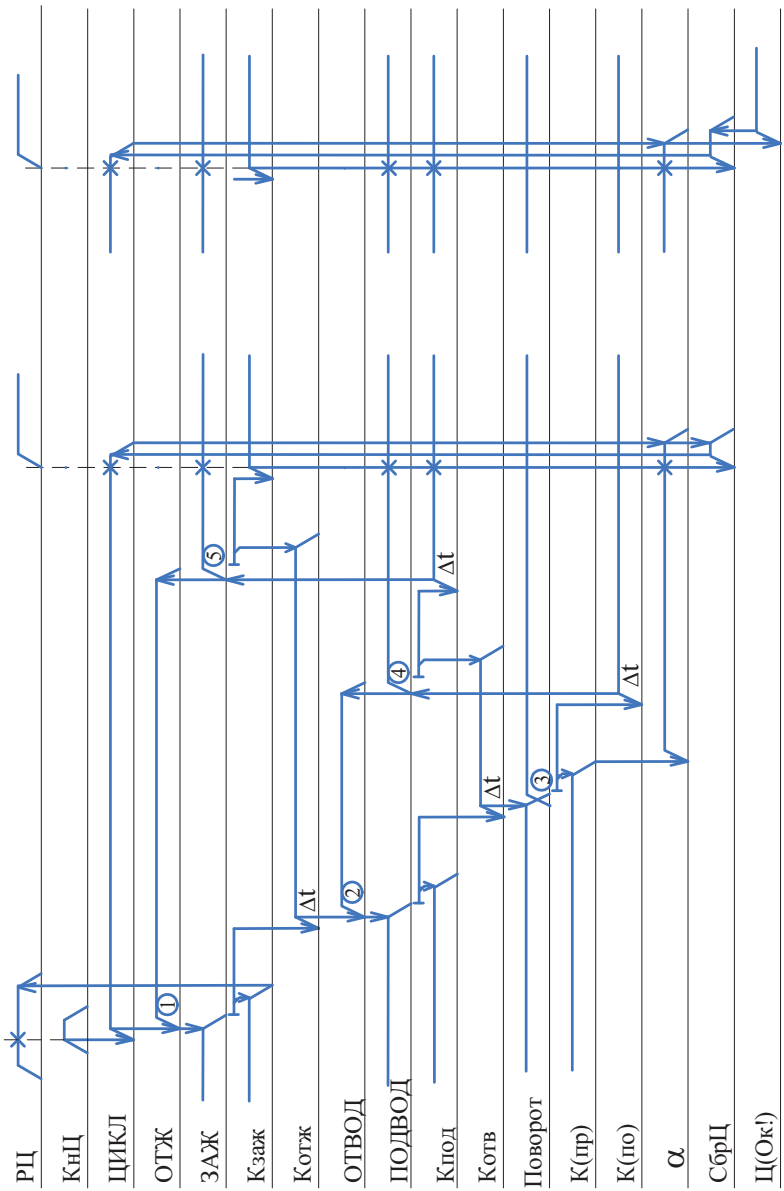


Рис. 2.50. Учебная циклограмма формирования сигнала «Цикл»

Рабочее тело цикла (3) определяется последовательностью работы конкретного механизма, в нашем примере оно включает следующие этапы (рис. 2.51):

1. Отжим инструмента в шпинделе.
2. Отвод манипулятора от станка, т. е. вытаскивание инструмента из шпинделя.
3. Поворот манипулятора на 180 градусов по или против часовой стрелки в зависимости от начального положения.
4. Подвод манипулятора к станку, т. е. вставление нового инструмента в шпиндель.
5. Зажим инструмента в шпинделе. По окончании зажима **считаем**, что наш учебный цикл закончен. Реально, еще нужно отвести в безопасное место от шпинделя манипулятор.

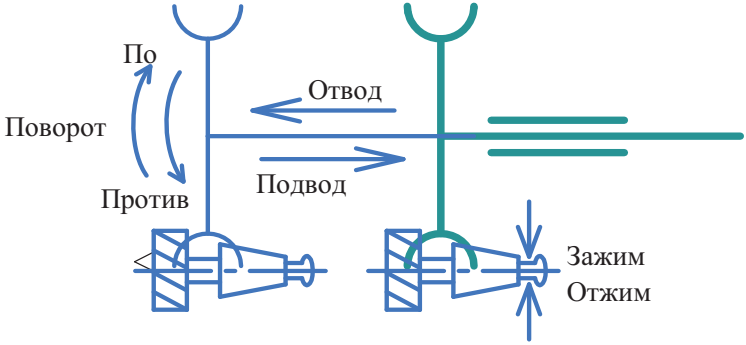


Рис. 2.51. Этапы работы манипулятора

Сигнал «Сброс цикла» (4) формируется при обязательном условии наличия сигнала ЦИКЛ и завершении его работы, что определяется исходным состоянием управляющих движениями манипулятора исполнительных элементов (электромагнитов) и сигналов датчиков контроля. Для того чтобы цикл мог быть повторен, их начальное состояние и состояние в конце цикла должны быть одинаковыми.

Теперь обсудим некоторые ключевые моменты:

1. Управление движениями манипулятора, в зависимости от решения конструктора-механика, может осуществляться как *двухходовыми* золотниками, так и *одноходовыми*.

На рис. 2.52, *а* приведен фрагмент циклограммы операции «Зажим/Отжим» с двумя взаимно инверсными золотниками. Каждая операция осуществляется включением отдельного электромагнита, причем теоретически после ее завершения электромагнит может быть выключен, однако делать этого

не следует. Это связано с возможными утечками в гидравлике, и как следствие, потери установленной позиции. Электромагниты в цикле нужно оставлять включенными.

На рис. 2.52, *б* приведен фрагмент циклограммы той же операции «Отжим» для одноходового золотника. В этом случае зажим инструмента осуществляется механически, а для его отжима нужно включить и обязательно удерживать электромагнит на все время, когда должна выполняться эта операция.

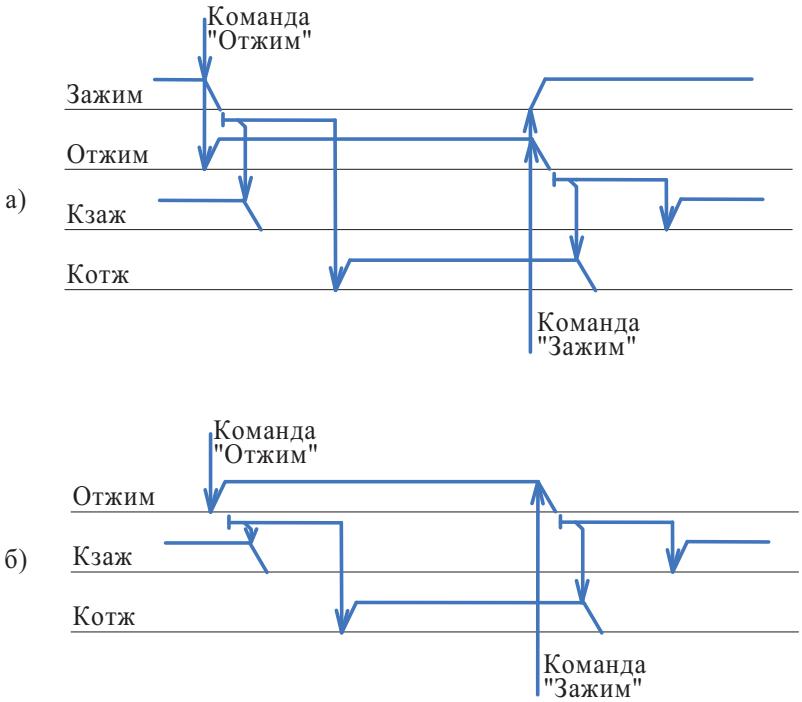


Рис. 2.52. Управление операцией «Отжим» двухходовым (а) и одноходовым (б) золотниками

2. Движения манипулятора, как правило, выполняются до механического упора. В таких случаях исключительно важным является вопрос настройки датчиков контроля. Очевидно, что датчик любого типа имеет дифференциал срабатывания, и если его поставить точно на уровне конечного положения механизма, то может возникнуть ситуация, когда датчик никогда не сработает (рис. 2.53, *в*). По этой причине установку датчика и момент его срабатывания следует настраивать упреждение (рис. 2.53, *а* и 2.53, *б*). Но в этом случае возникают другие проблемы.

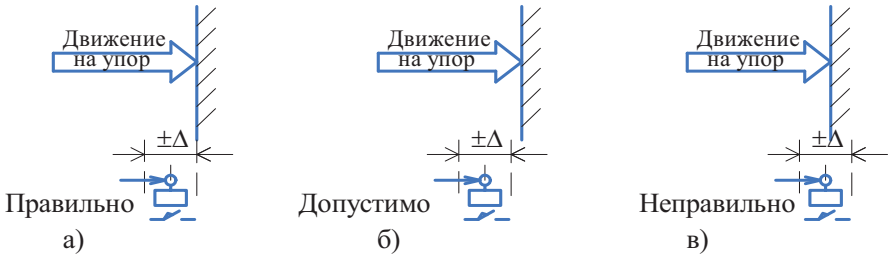


Рис. 2.53. Расстановка датчиков контроля

Например, манипулятор еще не выдвинул инструмент из шпинделя, а уже начинается его поворот, что может привести к поломке. Те же проблемы и при отжиме инструмента, еще нет полного отжима, а уже начинается выдвижение, последствия очевидны.

В таких критических случаях следует задерживать сигналы датчиков при помощи выдержек времени, например, на одну секунду.

Чтобы не усложнять циклограмму, показывать на ней сигналы таймеров необязательно, достаточно сделать пометки типа Δt в нужных местах.

3. В зависимости от конкретной ситуации возможны разные способы управления двухходовыми золотниками.

Например, если производится операция расфиксации/фиксации поворотного стола или инструментального магазина, когда по умолчанию эти механизмы должны быть зажаты и их автоматический зажим (фиксация) при включении станка не приведет к травме персонала, можно применить схему рис. 2.54, *а*.

Если строится схема управления подводом/отводом манипулятора и станок включается, когда манипулятор стоит в произвольном отведенном положении, то предыдущий вариант управления абсолютно не допустим.

Следует применить вариант рис. 2.54, *б*, вывести на экран сообщение о его нештатном положении, и исправить ситуацию путем прямого контролируемого оператором воздействия на соответствующий электромагнит. Принципы организации прямого управления будут рассмотрены далее.

Если строится схема управления поочередным поворотом манипулятора или поворотного механизма смены столов-спутников, можно рекомендовать алгоритм управления рис. 2.54, *в* на двух инверсных памятях. Блокировку невозможности самопроизвольного включения следует предусмотреть при формировании сигналов управления S.

В подобных схемах следует предусматривать также блокировку невозможности повторного поворота в одном цикле.

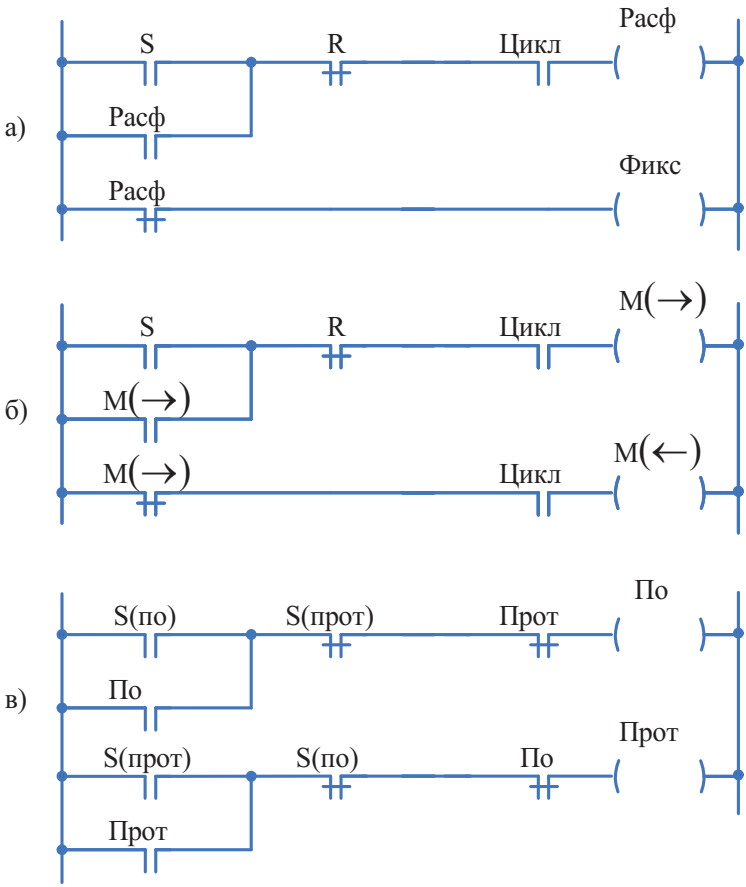


Рис. 2.54. Варианты управления двухходовыми золотниками

4. В связи с тем, что комбинации сигналов начала цикла и его окончания всегда одинаковы, что приводит к неоднозначности решения задачи, следует не задумываясь предусматривать промежуточный сигнал (α), разделяющий циклограмму на две половины. Это облегчит анализ циклограммы и синтез алгоритма правления.

5. Возможны два варианта окончания циклограммы:

– вариант рис. 2.50, *а* с полным завершением цикла по сигналу «Сброс»;

– вариант рис. 2.50, *б* с формированием дополнительного сигнала «Цикл (Ок!)», который можно использовать для контролируемого запуска следующих цикловых операций.

Теперь рассмотрим полноценный пример синтеза электроавтоматики манипуляторной смены инструмента.

2.12.6. Пример синтеза схемы управления манипулятором многооперационного станка с ЧПУ с горизонтальным расположением шпинделя

Графический цикл, поясняющий этапы работы манипулятора, приведен ниже на рис. 2.55. Перечень исполнительных органов, датчиков контроля, а также принятые условные сокращения сведены в табл. 2.16.

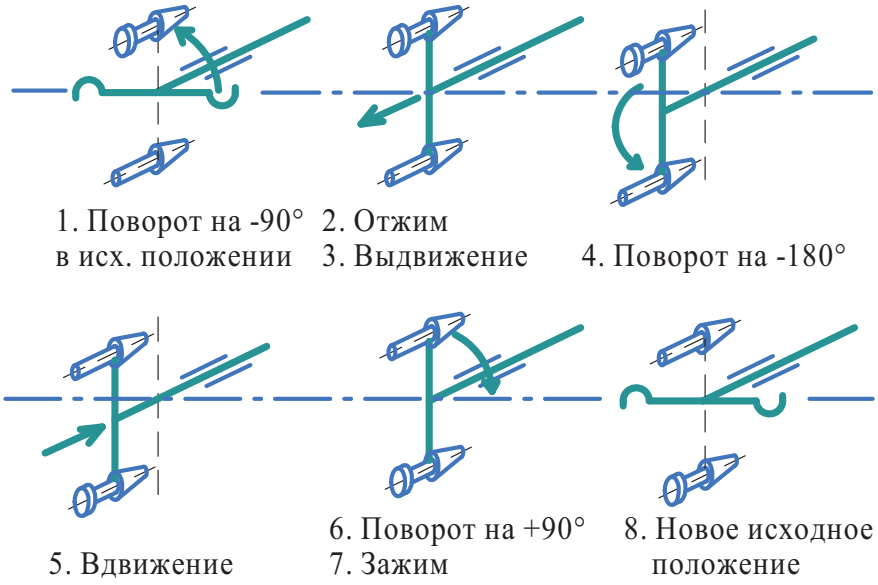


Рис. 2.55. Цикл работы манипулятора

Движение манипулятора в процессе выполнения цикла, а также операции отжима и зажима инструмента в шпинделе производятся за счет включения соответствующих **электромагнитов** $У_1 \ldots У_4$, управляющих **гидравлическими золотниками.**

Работа над проектом включает *следующие этапы*:

1. Изучение принципа работы механизма (рис. 2.55) и разработка аппаратной принципиальной схемы (рис. 2.56). Она включает в себя:

– асинхронный привод насоса гидравлики;

– схему подключения дискретных входов и выходов;
– схемы управления электромагнитами на постоянном токе и включения насоса гидравлики на переменном токе (не показано).

Таблица 2.16

Перечень исполнительных органов, датчиков контроля и условные сокращения

Операция (рис. 2.55)	Обозначение	Электромагнит				Элемент контроля
		$У_1$	$У_2$	$У_3$	$У_4$	
1. Поворот манипулятора против часовой стрелки на 90°	*П90*	+	–	–	–	$К_{90}$
2. Отжим инструментов	*ОТЖ*	+	–	–	+	$К_{ОТЖ}$
3. Выдвижение манипулятора	*ВЫДВ*	+	+	–	+	$К_{ВЫДВ}$
4. Поворот манипулятора против часовой стрелки на 180°	*П180*	+	+	+	+	$К_{180}$
5. Вдвижение манипулятора	–	+	–	+	+	$К_{ВДВ}$
6. Зажим инструментов	–	+	–	+	–	$К_{ЗАЖ}$
7. Поворот манипулятора по часовой стрелке в исходное положение	–	–	–	–	–	*Кгор*

Здесь же приводятся принятые адреса входных и выходных сигналов.

2. Формализация работы механизма при помощи циклограммы (рис. 2.57). Еще раз напомним, что рабочие сигналы показываются в порядке их появления в процессе выполнения цикла и всегда в **прямом виде**, т. е. включенному состоянию элемента соответствует высокий (единичный) уровень сигнала, а отключенному – низкий (нулевой).

Присвоение логическим переменным легко запоминающихся условных обозначений упрощает процесс синтеза и его чтение другими лицами.

Контроль выполнения операций осуществляется бесконтактными путевыми выключателями.

3. Синтез по циклограмме логических уравнений, описывающих поэтапную работу механизма.

4. Формальное вычерчивание релейно-контактного алгоритма, адаптированного к конкретному языку программируемого контроллера.

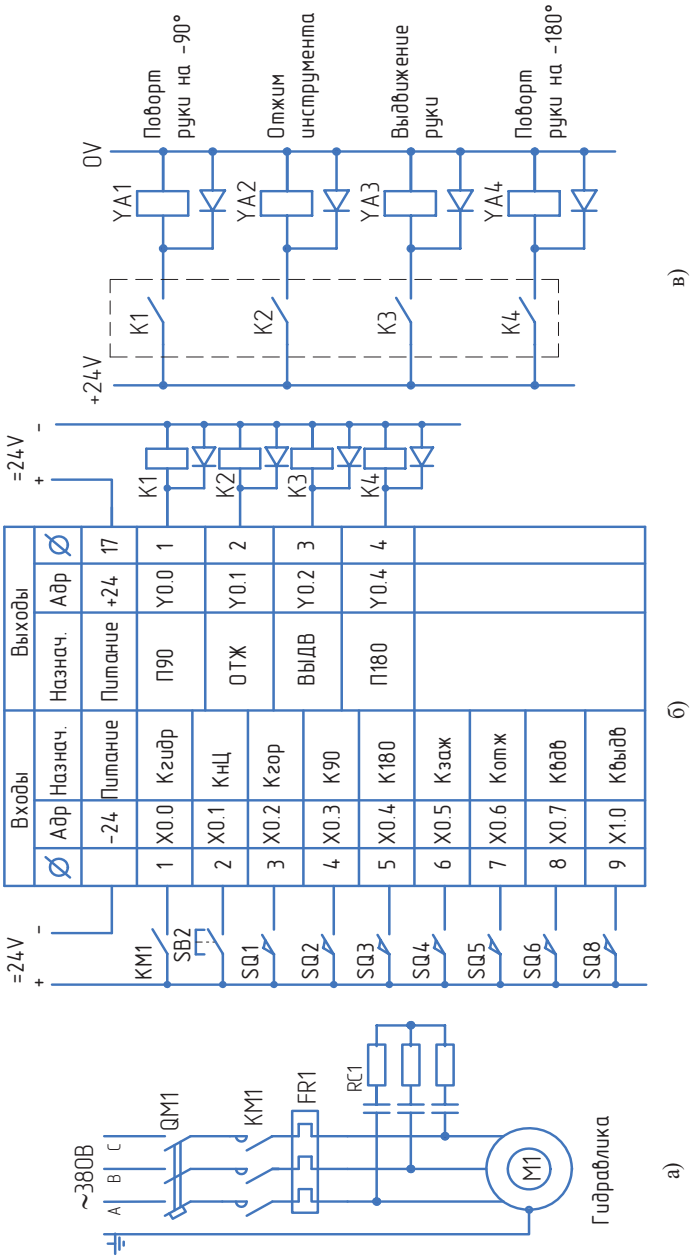


Рис. 2.56. Принципиальная схема проекта

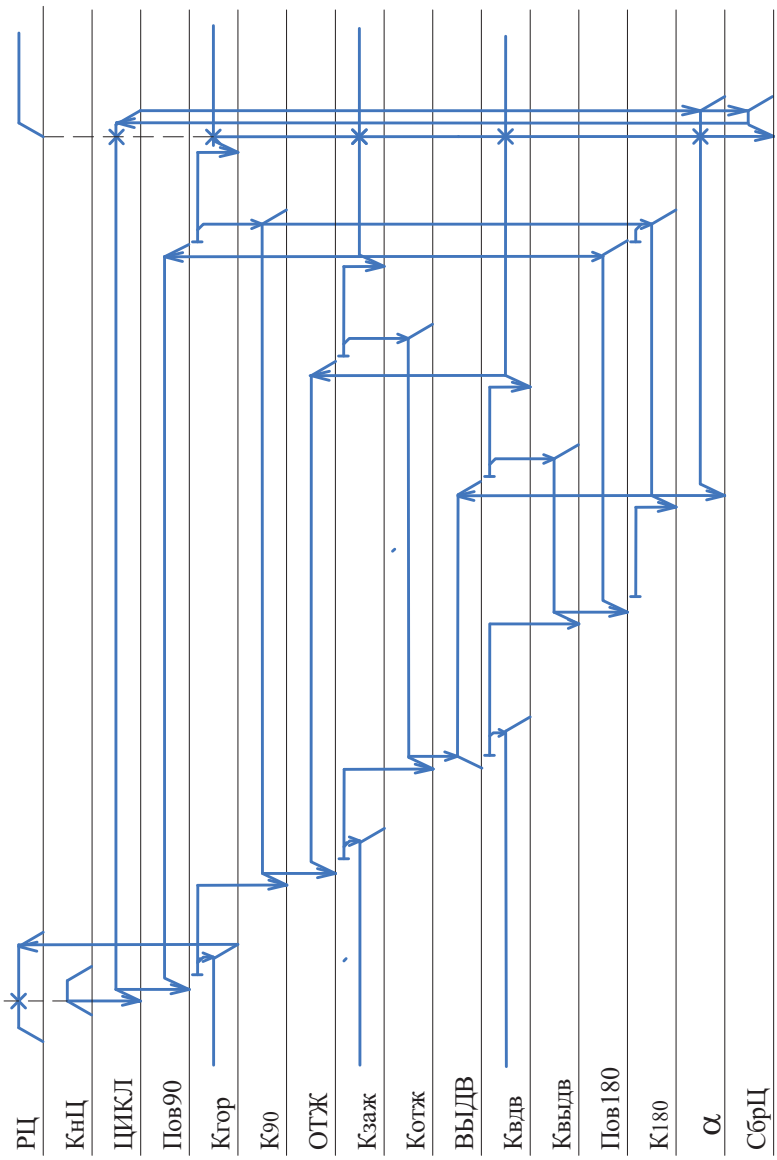


Рис. 2.57. Циклограмма работы системы управления манипулятором

Приступим к анализу циклограммы.

1. Сигнал «**Запрет»** формируется, если не включен станок, активизирован автоматический режим, включен шпиндель, получен сигнал «Аварии» из диагностической части программы, т. е.

$$\text{Запр} = / \text{Станок} + \text{Ц(Авт)} + \text{Шп} + \text{Авария}.$$

Контроль исходного положения манипулятора.

2. **Кисх** формируется при условии зажатого состояния инструмента в шпинделе, горизонтального и вдвинутого положения манипулятора, включенной гидравлике, т. е.

$$\text{Кисх} = \text{Кзаж} \cdot \text{Кгор} \cdot \text{Квдв} \cdot \text{Кгидр}.$$

3. Сигнал разрешения цикла **РЦ** формируется при условии отсутствия сигнала Запрет, включения ручного режима, наличия сигнала контроля исходного положения и правильной координаты шпинделя по оси Z, т. е

$$\text{РЦ} = / \text{Запр} \cdot \text{РРучн} \cdot \text{Кисх} \cdot \text{К(Z)}.$$

4. Сигнал **«Цикл»** синтезируется при помощи типовой памяти:

– включение S = РЦ · КнЦ
– выключение R = Запр + СбрЦ + Сбр или, приведя к релейной элементной базе

$$\overline{\text{R}} = \overline{\text{Запр}} \cdot \overline{\text{СбрЦ}} \cdot \overline{\text{Сбр}},$$

где КнЦ – кнопка активизации цикла;
Сбр – внешний сигнал сброса электроавтоматики.

Все последующие операции осуществляются в «теле цикла», поэтому данный сигнал может использоваться для блокировок других несовместимых циклов.

5. Сигнал **П90** не описывается однозначно уравнением алгебры логики с использованием только рабочих сигналов, так как время действия кнопки «Цикл» неопределенно. Длительность ее нажатия субъективно зависит от действий оператора, управляющего станком. Для снятия неопределенности, следуя приведенным выше рекомендациям в циклограмму заранее введен промежуточный сигнал α, легко описываемый уравнением

$$\alpha = (\text{К180} \cdot \text{П180} \cdot \text{ОТЖ} \cdot \text{Котж} + \alpha) \cdot \text{Цикл}.$$

В этом случае синтез сигнала П90 решается комбинационно

$$\text{П90} = (\overline{\alpha} \cdot \overline{\text{Квдв}}) \cdot \text{Цикл}.$$

Заметим, что можно найти решение и без применения промежуточного сигнала с использованием типовой памяти, но оно будет сложнее.

6. Синтез сигнала **ОТЖ** аналогичен:

$$\text{ОТЖ} = (\bar{\alpha} \cdot \overline{\text{Квдв}}) \cdot \text{К}_{90} \cdot \text{П90} \cdot \text{Цикл}.$$

Здесь сделаем очень важное пояснение. Формально, с точки зрения алгебры логики минимально-достаточным является уравнение:

$$\text{ОТЖ} = (\bar{\alpha} \cdot \overline{\text{Квдв}}) \cdot \text{К}_{90}.$$

Оно обеспечивает выполнение операции отжима при правильном функционировании манипулятора, в строгой последовательности выполнения операций, определенных циклограммой. Однако такая комбинация сигналов может образоваться и при случайном срабатывании элементов контроля или выполнении наладчиком пробных перемещений манипулятора путем прямого включения магнитов, например, нажатием отверткой. При этом может произойти отжим, и инструмент выпадет из шпинделя, что в свою очередь может привести к травме. Для исключения этого в управление вводится «помехозащитный» сигнал П90, который появляется только после нажатия кнопки «Пуск» и активизации Цикла. В этом случае случайное несанкционированное действие сигнала ОТЖ исключается. Такие сигналы будут введены и дальше.

7. Формирование сигнала Котж (Δt). Сигнал с датчика контроля отжима Котж следует задержать на время $\Delta t = 1$с, чтобы выдвижение манипулятора из шпинделя с инструментов начиналось при достоверном завершении операции отжима. Задача решается установкой таймера на включение.

8. Синтез сигнала **ВЫДВ** решается комбинационно:

$$\text{ВЫДВ} = \text{Котж}(\Delta t) \cdot / \text{К180} \cdot \text{ОТЖ} \cdot \text{Цикл}.$$

9. Синтез сигнала **П180** решается с применением типовой памяти:

– включение S = Квыдв,

– выключение R = Кзаж + / Цикл или, приведя к релейной элементной базе $\overline{\text{R}} = \overline{\text{Кзаж}} \cdot \text{Цикл}$.

10. Сигнал сброса цикла **СбрЦ** формально синтезируется комбинационно

$$\text{СбрЦ} = \text{Квдв} \cdot \text{Кзаж} \cdot \text{Кгор} \cdot \alpha \cdot \text{Цикл}.$$

Однако рекомендуется поставить его на память, чтобы он был активен до окончания сброса сигнала Цикл, т. е.

$$\text{СбрЦ} = (\text{Квдв} \cdot \text{Кзаж} \cdot \text{Кгор} \cdot \alpha + \text{СбрЦ}) \cdot \text{Цикл}.$$

Этапы работы ВДВИЖЕНИЕ, ЗАЖИМ и ПОВОРОТ В ИСХОДНОЕ ПОЛОЖЕНИЕ синтезу не подлежат, так как, в данном случае, они осуществляются автоматически при **выключении** одноходовых электрогидравлических золотников, выполняющих операции П90, ОТЖ, ВЫДВ и П180.

По синтезированным уравнениям легко строится логическая часть принципиальной схемы (рис. 2.58).

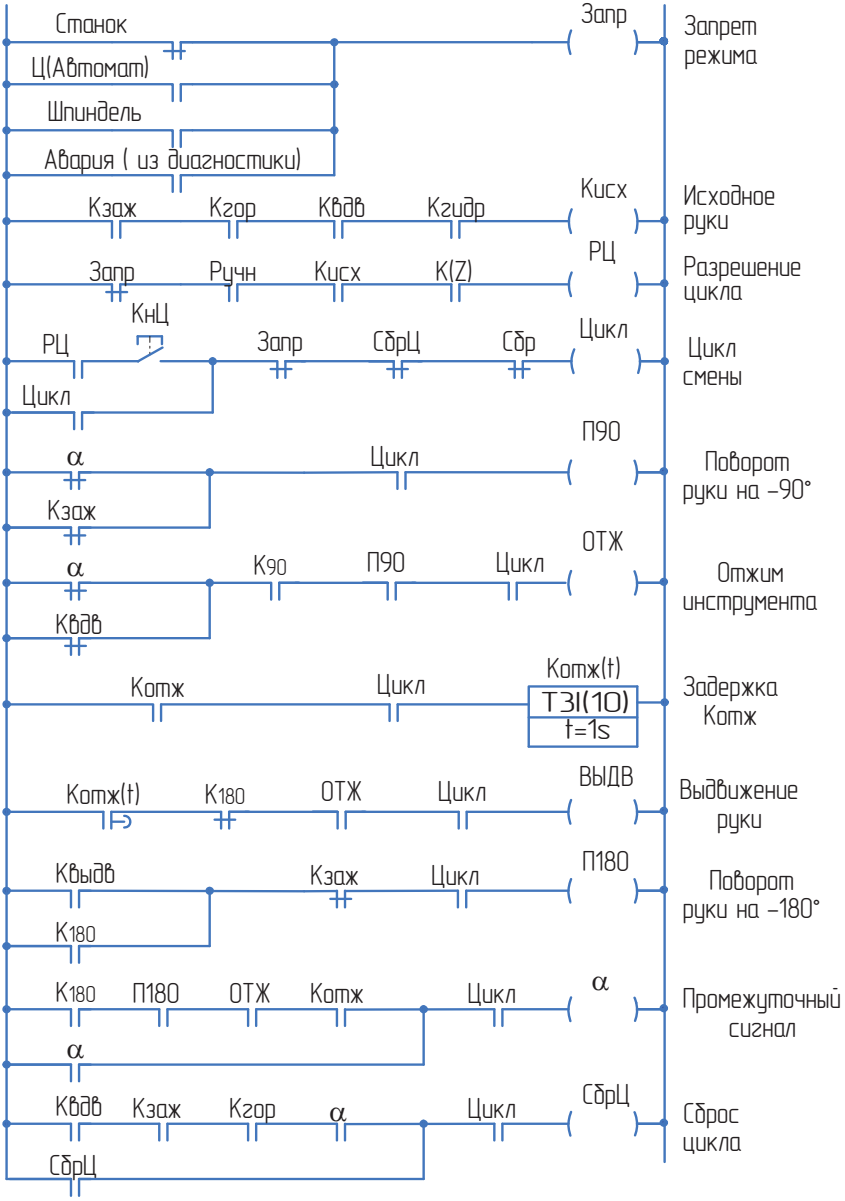


Рис. 2.58. Релейный алгоритм работы манипулятора

2.13. Метод счетчика последовательности

Суть метода счетчика последовательности заключается в том, что команды на выполнение каждой последующей операции (включения или выключения исполнительных органов) выдаются последовательно разрядами счетчика, причем запись сигналов в последующий разряд счетчика разрешается только при выполнении условий задания предыдущей операции и наличия сигнала контроля о ее выполнении. Таким образом, условия включения памятей разрядов счетчика запишутся так.

Первая операция (I оп).

$$S_1 = \text{Цикл} \cdot \text{РЦ},$$

где Цикл – сигнал, объединяющий условия начала цикла;
РЦ – блокировочный сигнал разрешения цикла.

Вторая операция (II оп)

$$S_2 = I_{\text{ОП}} \cdot K_1,$$

где $I_{\text{ОП}}$ – выходной сигнал первой операции;
K_1 – сигнал контроля выполнения первой операции.

Третья операция (III оп):

$$S_3 = II_{\text{ОП}} \cdot K_2 \text{ и т. д.}$$

Очевидно, что при необходимости в сигналы установки могут быть введены и какие-либо критические блокировочные сигналы.

В зависимости от способа выключения разрядов счётчика последовательности можно реализовать следующие практические структуры.

2.13.1. Структура с общим сбросом (рис. 2.59)

В этом случае число разрядов счётчика определяется общим числом включений и выключений исполнительных элементов схемы, определяемых последовательностью работы механизма. Например, для цикла управления манипулятором станка с ЧПУ (рис. 2.55):

Выполним синтез алгоритма в соответствии с циклограммой рис. 2.60.

I оп – Поворот манипулятора на 90°.

Условие включения $S1 = \text{РЦ} \cdot \text{КнЦ}$.

Виртуальный сигнал триггера П90 (+) обеспечивает включение электромагнита Y1, управляющего операцией поворота. Знак «+» в скобках означает начало операции или момент включения.

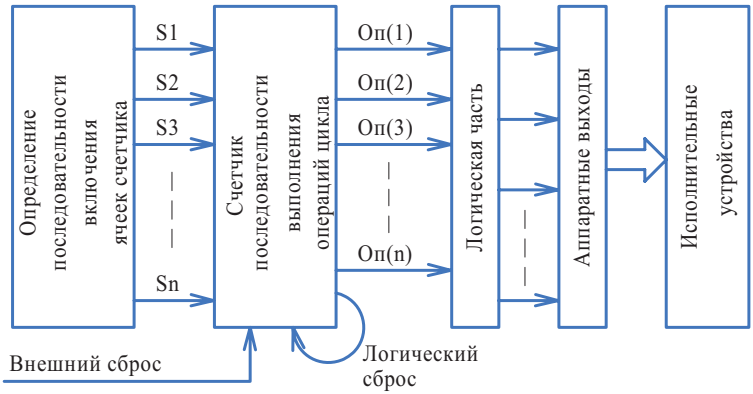


Рис. 2.59. Структурная схема счетчика последовательности с общим сбросом

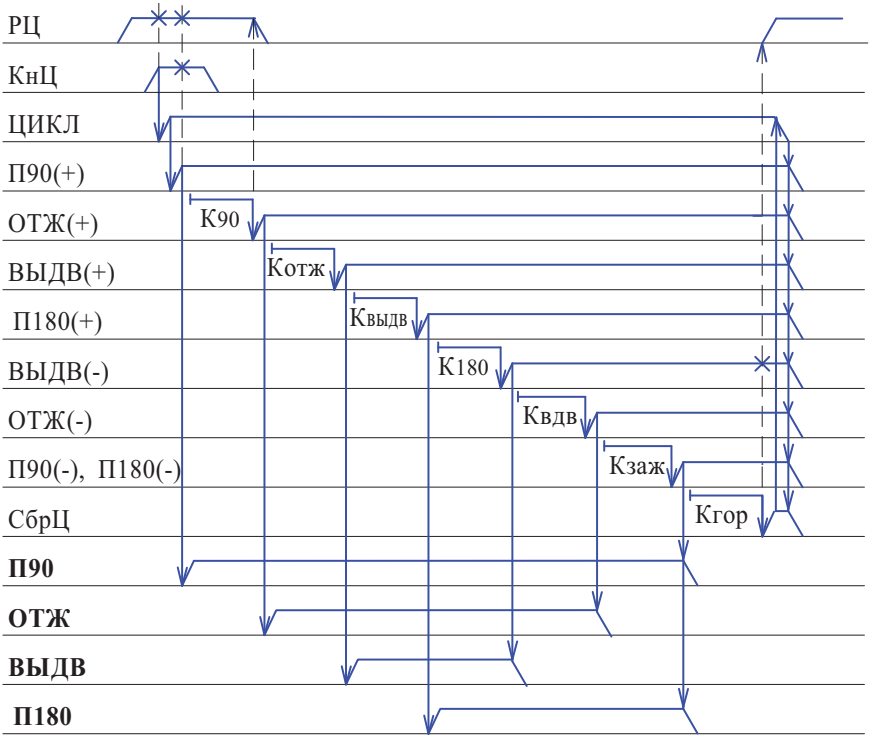


Рис. 2.60. Циклограмма счетчика последовательности с общим сбросом

II оп – Отжим инструментов.

Условие включения $S2 = П90(+) \cdot К90$, т. е. определяется наличием сигнала задания первой операции П90(+) и наличием сигнала контроля о её завершении К90.

Виртуальный сигнал триггера ОТЖ(+) определяет момент включения электромагнита Y4, управляющего операцией отжима.

III оп – Выдвижение манипулятора.

Условие включения $S3 = ОТЖ(+) \cdot Котж$.

Виртуальный сигнал триггера ВЫДВ(+) определяет момент включения электромагнита Y2 , управляющего операцией выдвижения.

IV оп – Поворот манипулятора на 180°.

Условие включения $S4 = ВЫДВ(+) \cdot Квыдв$.

Виртуальный сигнал триггера П180(+) определяет момент включения электромагнита Y3, управляющего операцией поворота на 180°.

V оп – Вдвижение манипулятора.

Условие включения $S5 = П180(+) \cdot К180$.

Виртуальный сигнал триггера ВЫДВ(−) определяет конец операции выдвижения (знак «−»), осуществляемый отключением электромагнита Y2.

VI оп – Зажим инструментов.

Условие включения $S6 = ВЫДВ(-) \cdot Квдв$.

Виртуальный сигнал триггера ОТЖ(−), определяет конец операции зажима, осуществляемой отключением электромагнита Y4.

VII оп – Поворот манипулятора в исходное положение.

Условие включения $S7 = ОТЖ(-) \cdot Кзаж$.

Виртуальный сигнал триггера ПОВ(−) определяет окончание действия сигналов П90(+) и П180(+) и поворот манипулятора в исходное положение, осуществляемой путем отключения электромагнитов Y1 и Y3.

VIII – Общий сброс цикла осуществляется при следующих условиях

$$СбрЦ = ПОВ(-) \cdot Квдв \cdot Кзаж \cdot Кгор \cdot Цикл.$$

Также следует предусмотреть внешний сброс.

Выходные сигналы управления манипулятором, непосредственно активизирующие электромагниты, описываются следующими уравнениями:

$$П90 = П90(+) \cdot \overline{ПОВ(-)},$$

$$ОТЖ = ОТЖ(+) \cdot \overline{ОТЖ(-)},$$

$$\text{ВЫДВ} = \text{ВЫДВ}(+) \cdot \overline{\text{ВЫДВ}(-)},$$

$$\text{П180} = \text{П180}(+) \cdot \overline{\text{ПОВ}(-)}.$$

Релейно-контактный алгоритм приведен на рис. 2.61.

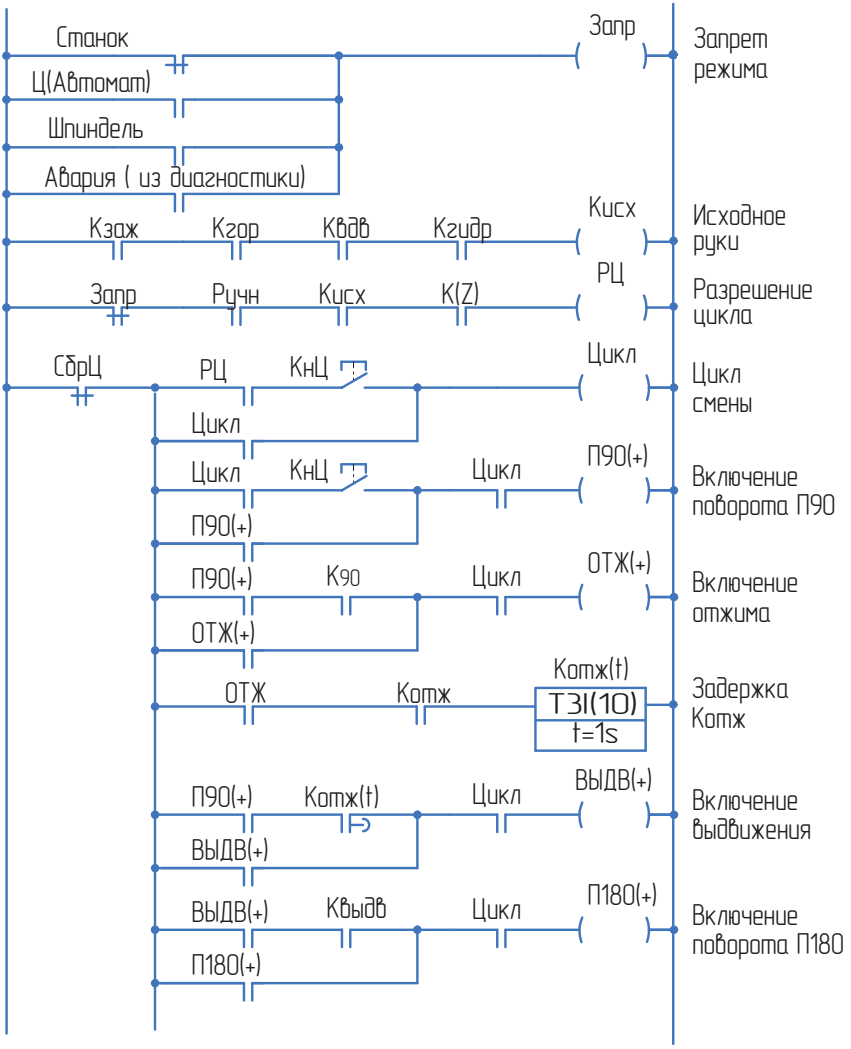


Рис. 2.61. Релейный алгоритм работы манипулятора (счетчик последовательности с общим сбросом) (начало)

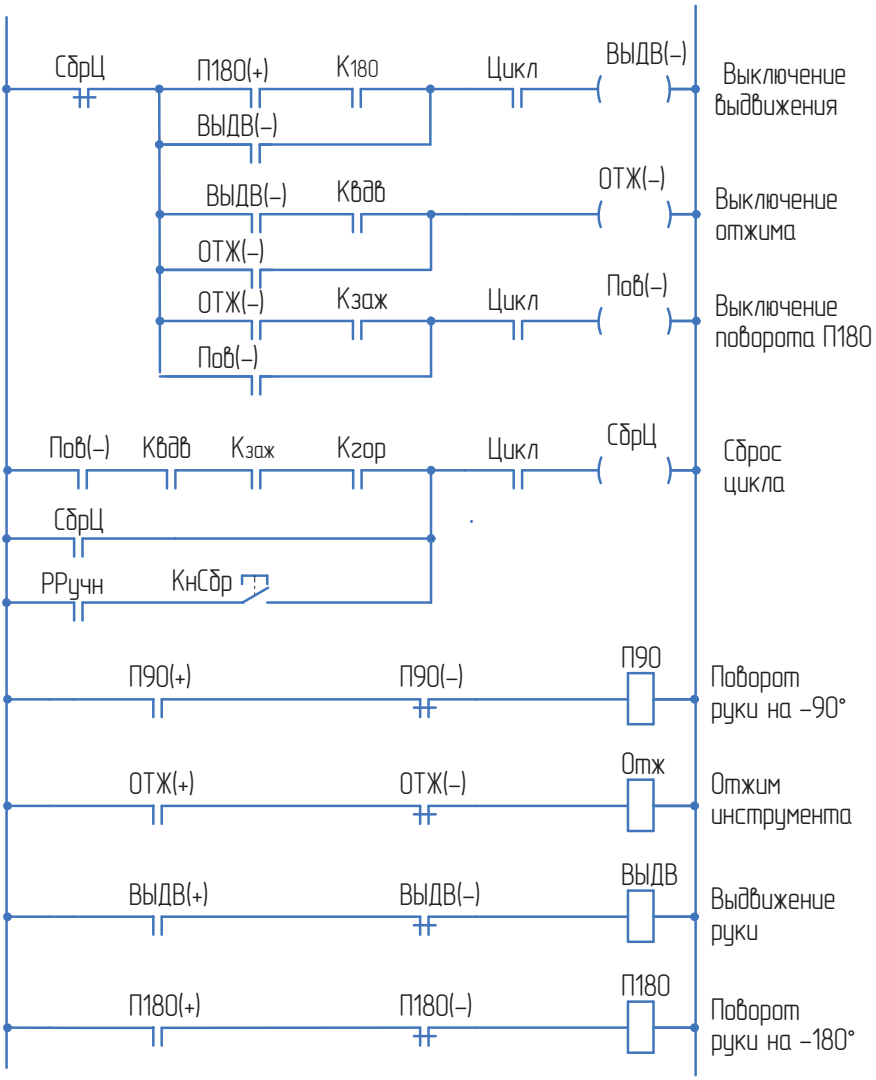


Рис. 2.61. Релейный алгоритм работы манипулятора (счетчик последовательности с общим сбросом) (окончание)

Преимуществом данного метода является простота проектирования, и его можно рекомендовать для синтеза схем средней сложности, когда определяющим являются сроки выполнения работы, например, при отладке работы механизма в цеховых условиях.

2.13.2. Структура с логическим сбросом (рис. 2.62)

Счетчик последовательности с логическим сбросом позволяет значительно сократить число ячеек структуры алгоритма по сравнению со счетчиком с общим сбросом, но более сложен в разработке. Конечный результат мало чем отличается от классического решения. Тем не менее, метод представляет интерес, поэтому приведем решение той же задачи.

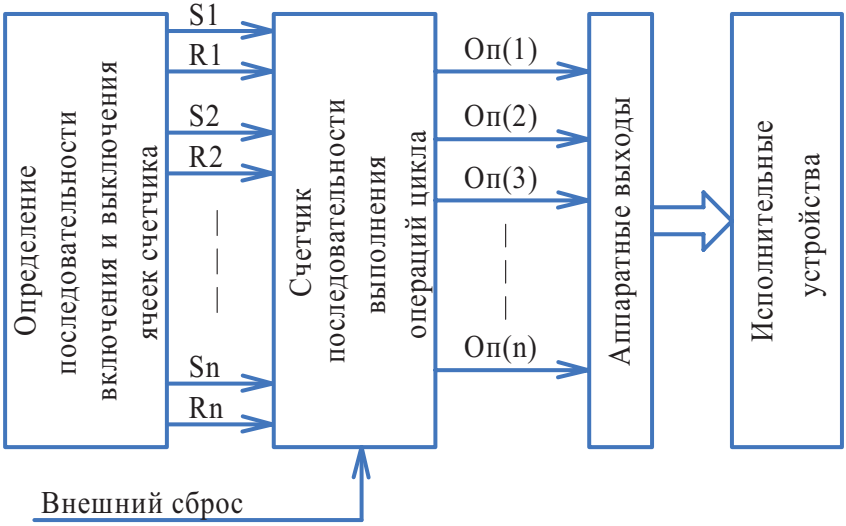


Рис. 2.62. Структурная схема счетчика последовательности с логическим сбросом

Циклограмма и релейно-контактный алгоритм управления манипулятором с применением счетчика последовательности с логическим сбросом приведены, соответственно, на рис. 2.63 и 2.64. Такой счетчик не имеет виртуальных ячеек, в нем сразу синтезируются выходные сигналы, т. е. число разрядов счетчика определяется числом исполнительных органов.

Включение памятей осуществляется аналогично счетчику с общим сбросом, при наличии задания и контроля выполнения предыдущей операции, а выключение синтезируется по циклограмме индивидуально, логически, для каждого выхода. Такой подход требует введения промежуточного сигнала α, разделяющего циклограмму на две зоны. Исследуя метод, автор пришел к выводу, что на циклограмме, после сигнала α удобно изображать логические сигналы сброса, как показано на рис. 2.63.

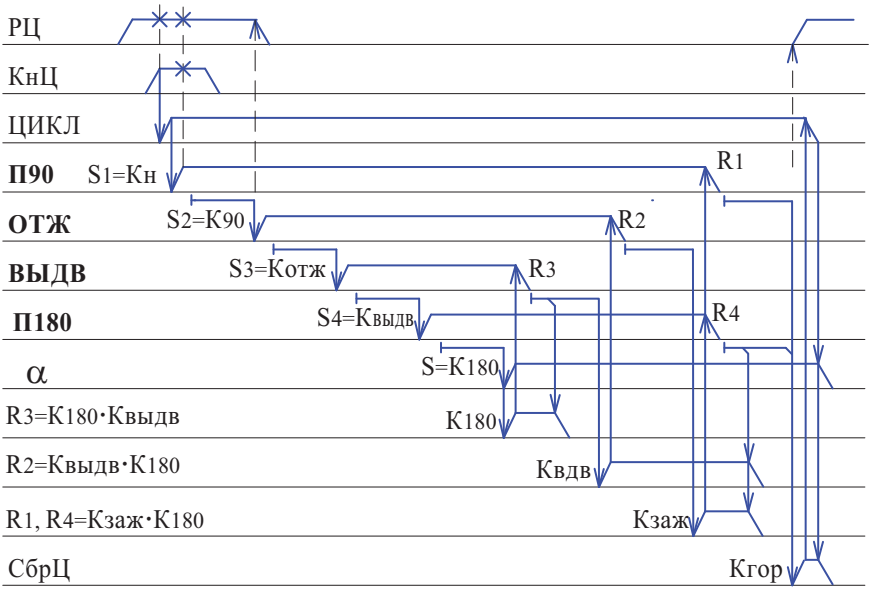


Рис. 2.63. Циклограмма управления манипулятором с использованием счетчика последовательности с логическим сбросом

Итак, условия включения:

- выход Цикл S = РЦ · КнЦ;
- выход П90 S = Цикл · КнЦ;
- выход ОТЖ S = П90 · К90 · / α;
- выход ВЫДВ S = ОТЖ · Котж · / α;
- выход П180 S = ВЫДВ · Квыдв;
- сигнал α: S = П180 · К180.

Сигналы выключения:

- выход Цикл R = СбрЦ;
- выход П90 R = Кзаж + К180;
- выход ОТЖ R = Кзаж + К180;
- выход ВЫДВ R = К180 + Квыдв;
- выход П180 R = Кзаж + К180;
- сигнал α: R = / Цикл.

Естественно, что все выходы находятся в теле цикла.

Сигнал сброса цикла R = α · Квдв · Кзаж · Кгор · Цикл.

Его следует поставить на самопитание, а также предусмотреть внешний сброс.

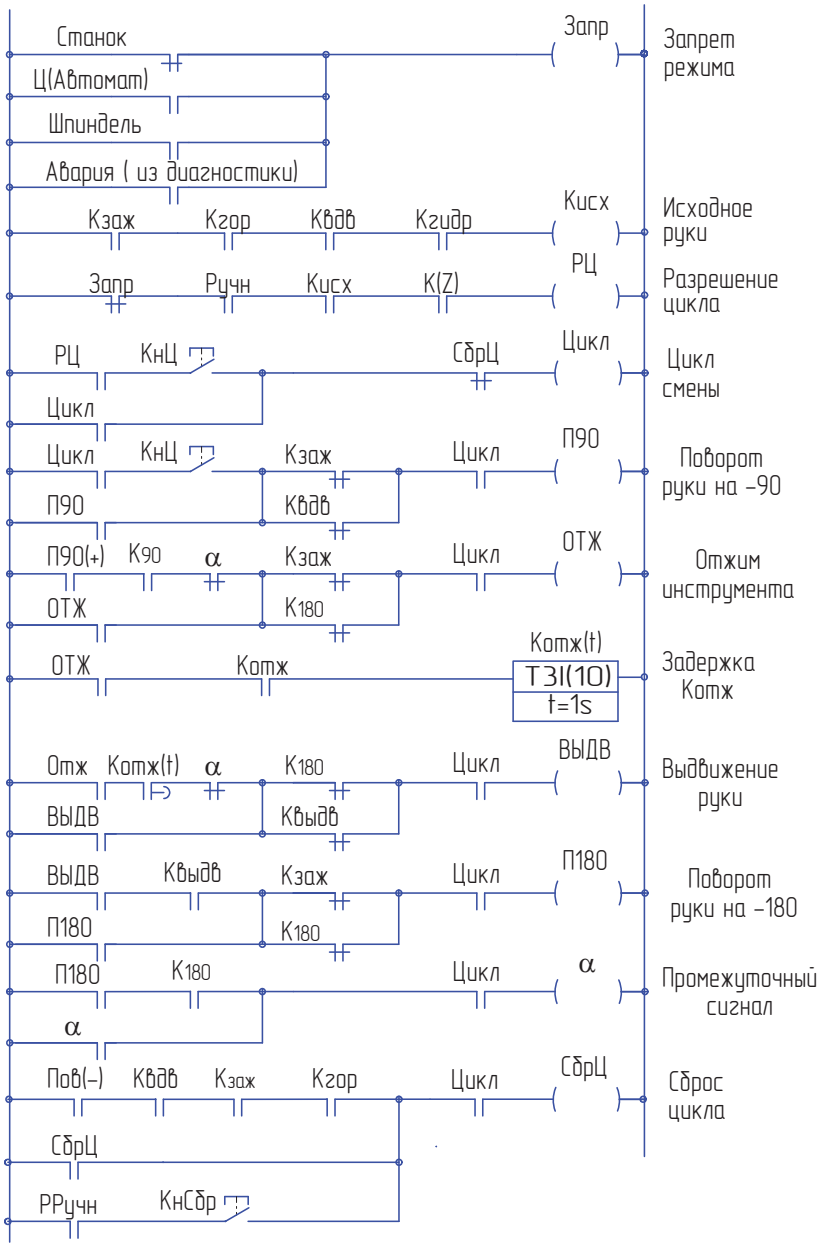


Рис. 2.64. Релейно-контактный алгоритм управления манипулятором с использованием счетчика последовательности с логическим сбросом

2.14. Организация пошагового выполнения цикла

Принцип пошагового управления каким-либо автоматическим циклом, например, сменой инструментов или столов-спутников, перемещения поперечины со сложным управлением, очень полезен при первоначальной наладке станка или при проведении регламентных работ.

Принцип пошагового (поэтапного) режима управления заключается в том, что при каждом нажатии кнопки «Цикл» выполняется только одна операция (один шаг). Для продолжения работы следует повторно нажать кнопку по завершении предыдущей операции.

Пошаговый режим, как правило, организуется в ручном режиме или режиме преднабора. При работе в автоматическом режиме пошаговый режим блокируется.

Последовательность синтеза пошагового режима следующая:

1. *Активизация режима.*

Режим активизируется специальной кнопкой «Шаг», расположенной на пульте управления станка.

Снятие режима может быть организовано, в зависимости от идеологии управления и располагаемым ресурсом свободных органов управления различными способами, например:

- вторичным нажатием кнопки «Шаг»;
- отдельной кнопкой «Отключение режима»;
- кнопкой «Сброс»;
- переходом на другой режим работы.

Предпочтительней и экономичней первый способ с индикацией включения режима светодиодом или сообщением на экране дисплея (рис. 2.65).

При необходимости формируется сигнал запрета включения режима.

2. *Организация сигнала прерывания* (разрешения) цикла по завершении этапа работы (рис. 2.65).

Удобнее формировать инверсный сигнал **Авт/Шаг** «Разрешение», включаемый в условия начала работы каждого этапа.

Этот сигнал принимает единичное значение при следующих условиях:

- всегда в автоматическом режиме, обеспечивая непрерывное выполнение цикла;
- в ручном режиме при условии выключенного режима «Шаг», также обеспечивая непрерывное выполнение цикла;
- при включенных ручном и пошаговом режимах при нажатой кнопке «Цикл», что обеспечивает пошаговое выполнение автоматического цикла.

3. *Врезка сигналов прерывания в основной алгоритм*.

При снятии режима «Шаг» во время выполнения текущей операции продолжается автоматический цикл.

Примечание. По соображениям техники безопасности следует запретить продолжение автоматического цикла при его прерывании и последующем переключением из ручного режима в автомат.

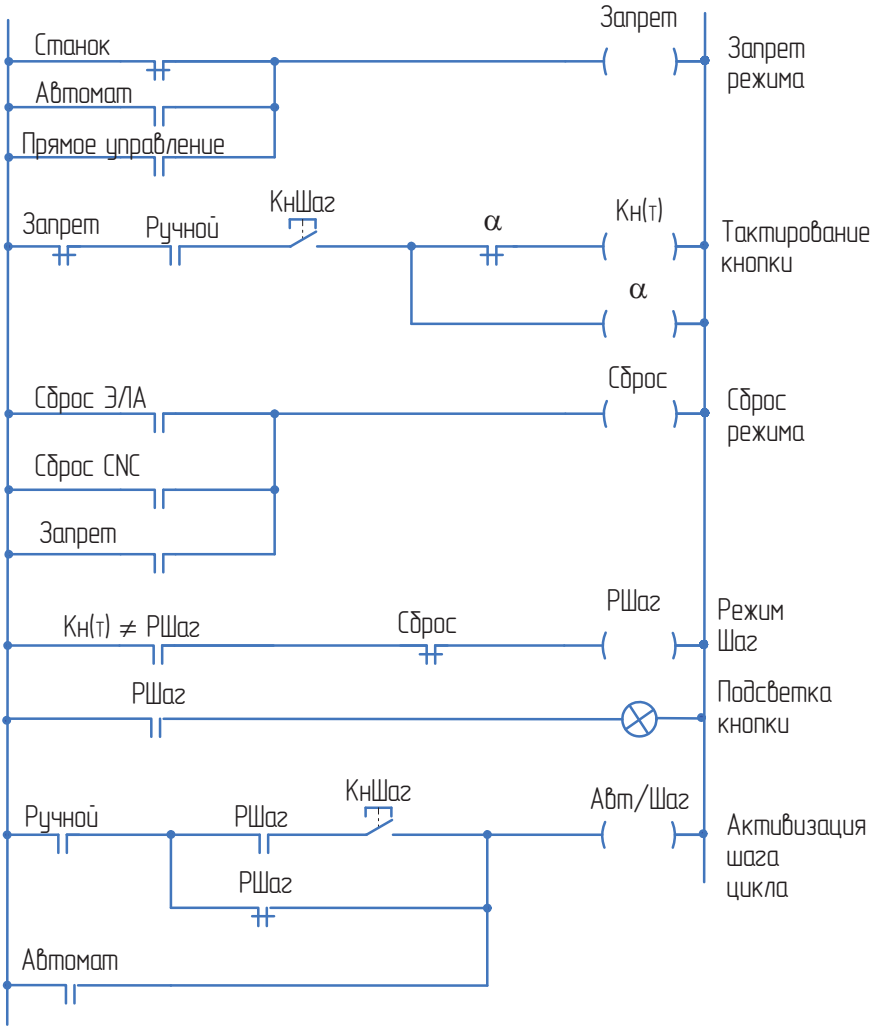


Рис. 2.65. РКС-алгоритм активизации режима «Шаг»

Принцип организации прерывания показан на следующих рисунках.

Здесь возможны такие варианты:

Вариант 1 (рис. 2.66, *а*).

Выходной сигнал операции Y1 реализуется при помощи типовой релейной памяти, т. е. Y1 = (S1 + Y1) * / R1, сигналы установки S1 и сброса R1 памяти достаточно сложны и поэтому реализованы в виде самостоятельных комбинационных сигналов. Такой подход увеличивает число логических уравнений, но значительно облегчает чтение и отладку алгоритма.

В этом варианте сигнал пошагового режима **Авт/Шаг** включается раздельно в цепь включения S1 и выключения R1 выходного сигнала Y1.

В автоматическом и в наладочном режимах при выключенном пошаговом режиме сигнал **Авт/Шаг** = 1 и не влияет на процесс выполнения цикловой автоматики.

При включенном пошаговом режиме действие сигналов включения S1 и выключения R1 **прерывается** и осуществляется **только** при условии нового нажатия кнопки **Шаг**.

Вариант 2 (рис. 2.66, *б*)

Применяется в случае, если условия включения S2 и выключения R2 памяти выходного сигнала Y2 просты, т. е. содержат не более одного-трех операндов.

В этом варианте сигнал пошагового режима **Авт/Шаг** включается **последовательно** в виде замыкающегося контакта в цепь включения S2 и **параллельно** в виде размыкающегося контакта в цепь выключения R2 выходного сигнала Y2.

Вариант 3 (рис. 2.66, *в*)

Применяется когда выходные сигналы операции Y1 и Y2 являются взаимосвязанными командами (например, Отжим – Зажим, поворот По – Против часовой стрелки др.).

Сигнал управления S1 включает 1-ю операцию и выключает 2-ю операцию.

Сигнал управления S2, наоборот, включает 2-ю операцию и выключает 1-ю операцию.

В этом варианте сигнал пошагового режима **Авт/Шаг** включается в обе цепи управления S1 и S2, разрешая или прерывая процесс в зависимости от активизированного режима.

При необходимости, может организовываться начальная установка памятей.

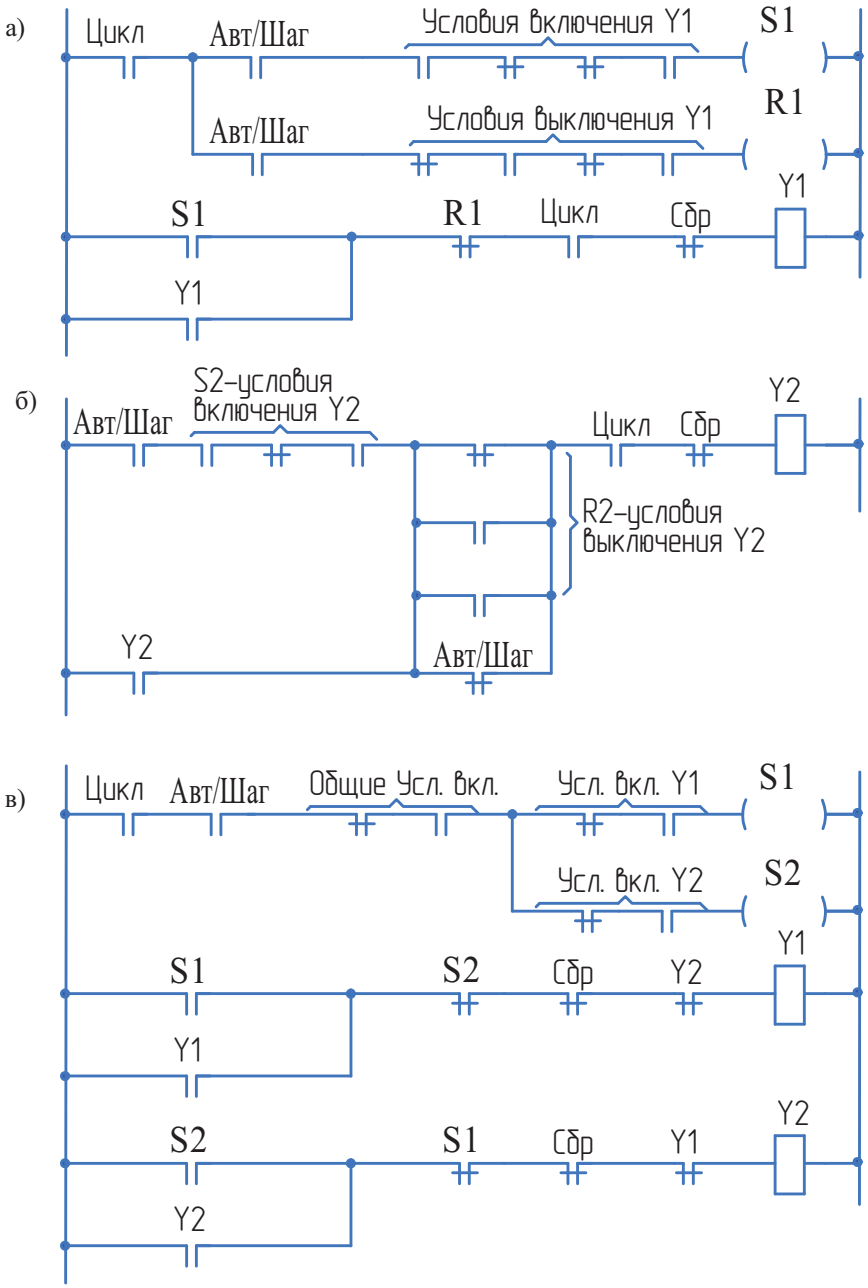


Рис. 2.66. Организация прерывания в случае решения с типовой памятью

Вариант 4 (рис. 2.67, *а*)

Выходной сигнал Y3 синтезируется комбинационным логическим уравнением, т. е. Y3 = X1·X2·X3·…·Xn.

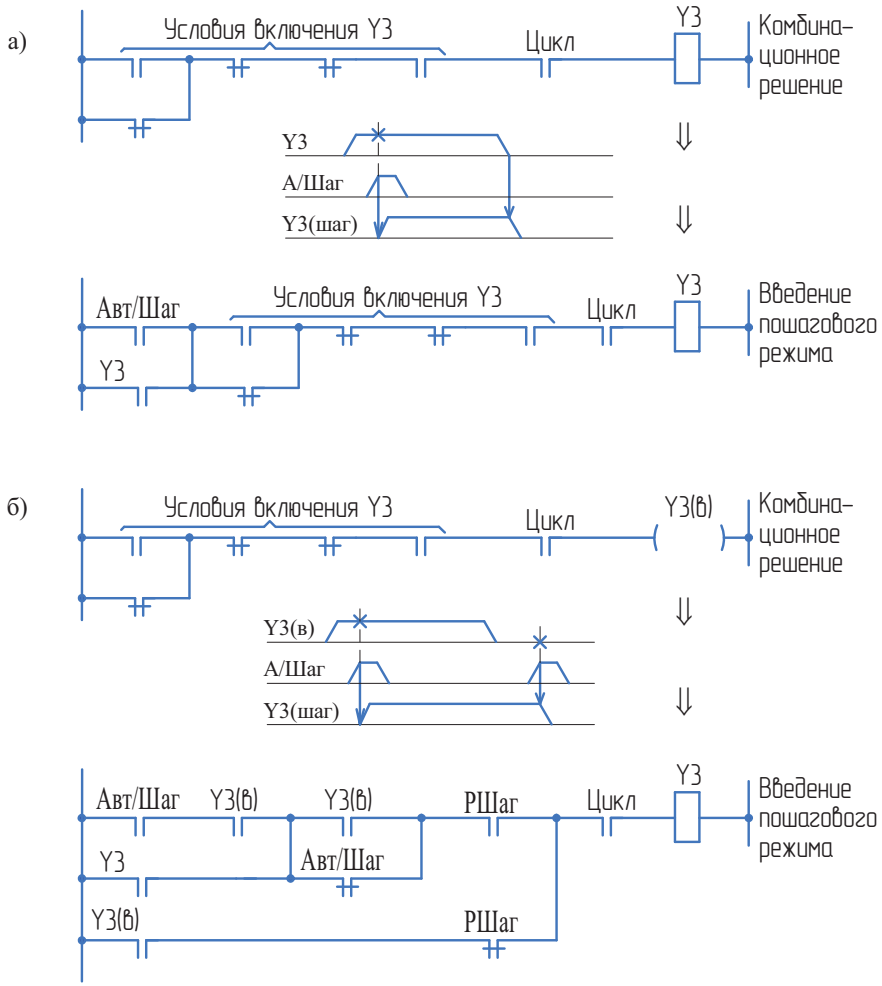


Рис. 2.67. Организация прерывания в случае комбинационного решения

В этом варианте сигнал пошагового режима **Авт/Шаг добавляется** последовательно в комбинационную цепь и шунтируется выходным сигналом операции Y3, ставя его на самопитание, т. е.

$$Y3 = (Авт / Шаг + Y3) \cdot X1 \cdot X2 \cdot X3 \cdot \ldots \cdot Xn.$$

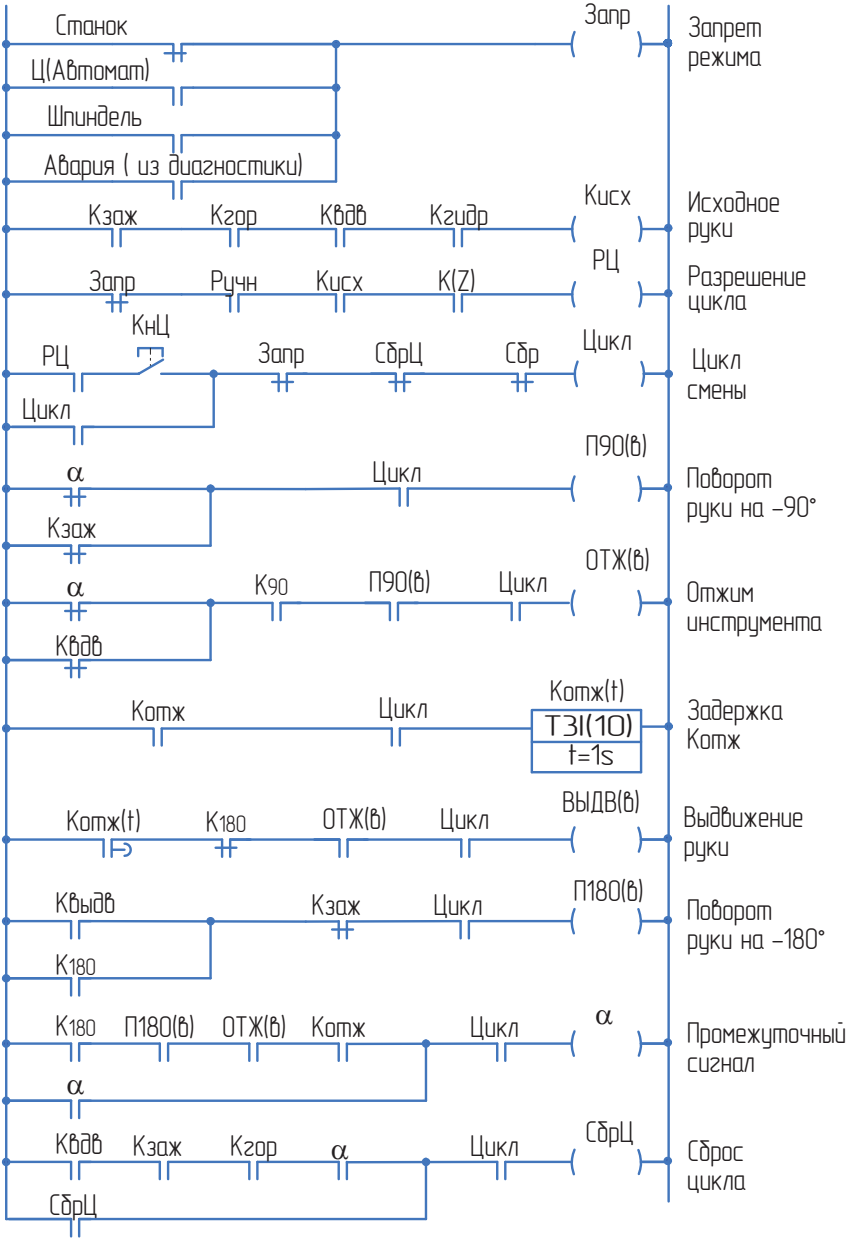


Рис. 2.68. Релейный алгоритм управления манипулятором с выходными виртуальными ячейками

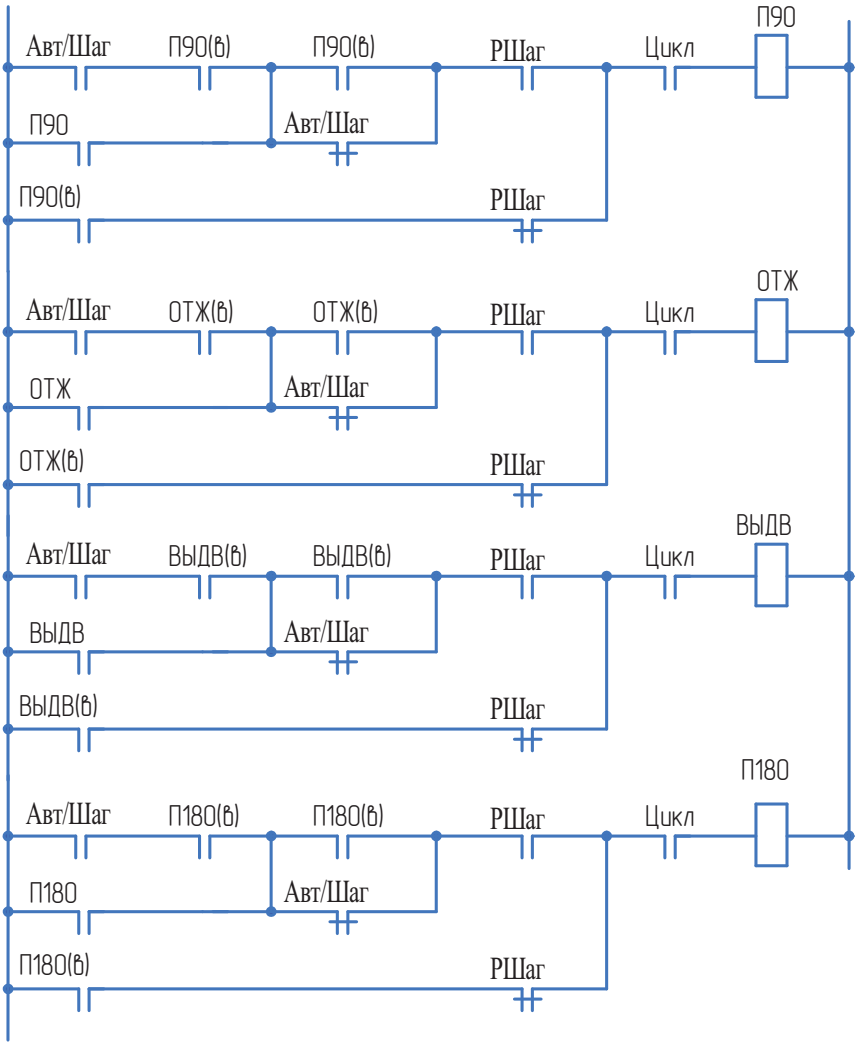


Рис. 2.69. Организация прерывания в выходной части алгоритма (вариант рис. 2.67, б)

Вариант 5 (рис. 2.67, *б*)

Выходной сигнал Y3 аналогично синтезируется комбинационным логическим уравнением, т. е. Y3 = X1·X2·X3· … ·Xn, однако ставится другая задача.

Необходимо, чтобы после активизации шага и его выполнения, выходной сигнал запоминался и отключался только при повторном нажатии кнопки при

переходе к следующему шагу. Такое решение необходимо при работе с одноходовыми золотниками, отключение которых автоматически приводит к противоположному действию.

В этом варианте синтезированные сигналы следует перевести в виртуальные, а пошаговый режим организовать на выходных сигналах. Выходной сигнал Y3 управляется двумя параллельными цепями, с памятью в шаговом режиме (Авт / Шаг = 1) и в комбинационном виде в автоматическом режиме (Авт / Шаг = 0).

Ниже приведен пример управления манипулятором, движения которого осуществляются при помощи одноходовых золотников (рис. 2.58 и его повтор рис. 2.68), с введением возможности пошагового управления по варианту 5 (рис. 2.69).

2.15. Прямое управление электромагнитами

Прямое управление электромагнитами механизмов, работающих по сложному автоматическому циклу, применяется для выхода из различных нештатных ситуаций, могущих возникнуть как при первоначальном включении станка, так и при сбоях или зависании электроавтоматики в процессе работы. Режим очень полезен при первоначальной наладке механизмов станка.

Организация режима прямого управления предполагает следующее:

- защиту от случайной активизации режима. Вызов режима должен быть либо скрыт, либо усложнен по специальной процедуре (например, удержание кнопки в течение трех секунд в алгоритме рис. 2.70);
- наличие органа (органов) вызова режима и индикацию его активного состояния;
- наличие органов управления электромагнитами. В зависимости от аппаратных возможностей могут быть раздельные органы управления для каждого электромагнита, так и общий с предварительным выбором электромагнита методом перебора;
- защиту и диагностические сообщения о запрещенных комбинациях включения электромагнитов.

  Внимание! Необходимо учитывать только те начальные условия, отсутствие которых может привести к поломке механизма. В целом, это достаточно сложная для принятия решения ситуация. Например, если в манипуляторе нет инструментов, то его можно вращать в любом положении, если же в нем находятся инструменты, то только в выдвинутом положении и т. д. Как поступить? Можно предусмотреть какие-то ограничения и установить определенную последовательность выпол-

нения операций, например, прежде чем вращать, выполнить операцию отвода. Можно все разрешить под ответственность оператора или наладчика. Чаще всего так и делается, прямое управление организуется с минимумом учета начальных условий положения механизма, последние учитываются только в критических случаях, когда их отсутствие может действительно привести к поломке механизма;

– диагностические сообщения о завершении выполнения операций и автоматическом отключении электромагнита;
– процедуру отключения режима.

Ниже приводятся два варианта организации алгоритма прямого управления.

Вариант 1. Ограниченные аппаратные возможности, т. е. на пультах управления станком отсутствует достаточное для организации режима число кнопок управления (системы ЧПУ Синумерик 802Д, NC110, 4СК, «Микрос», «Фанук» и др.).

Такой вариант приведен на рис. 2.70 и рис 2.71. В этом случае, кроме кнопки активизации режима нужно предусмотреть еще две:

– кнопку кольцевого перебора управляемых операций с выводом сообщений о выбранной операции на экран. Перебор можно организовать, например, на кольцевом сдвиговом регистре или счетчике. На рис. 2.71 приведены только результирующие сигналы алгоритма перебора: «Отвод» (прямое) и «Поворот/По» (прямое);
– кнопку активизации операции. В зависимости от ситуации, здесь можно использовать ту же кнопку активизации режима прямого управления, сделав ей двойное назначение. На рис. 2.71 это кнопка выполнения операции КнВып.

Выход из режима осуществляется переходом в автоматический режим, сбросом или при активизации другого несовместимого с прямым управлением режима.

Вариант 2. Неограниченные аппаратные возможности, например, при использовании виртуальных клавиш устройства ЧПУ. В этом случае для прямого управления конкретными электромагнитами используются раздельные кнопки управления для каждого магнита. Можно использовать как горизонтальные, так и вертикальные клавиши.

При активизации режима в подпрограмме формирования виртуальных клавиш следует обеспечить автоматический вызов кнопок прямого управления, например:

КнОтж – отжим инструмента;
КнЗаж – зажим инструмента;

КнОтв – отвод манипулятора;
КнПод – подвод манипулятора;
КнПо – поворот манипулятора по часовой стрелке;
КнПрот – поворот манипулятора против часовой стрелке.

Пример такой организации прямого управления приведен на рис. 2.72.

В обоих вариантах обеспечиваются минимальные блокировки и диагностическое сопровождение, смысл которых ясен из комментариев, приведенных на алгоритмах.

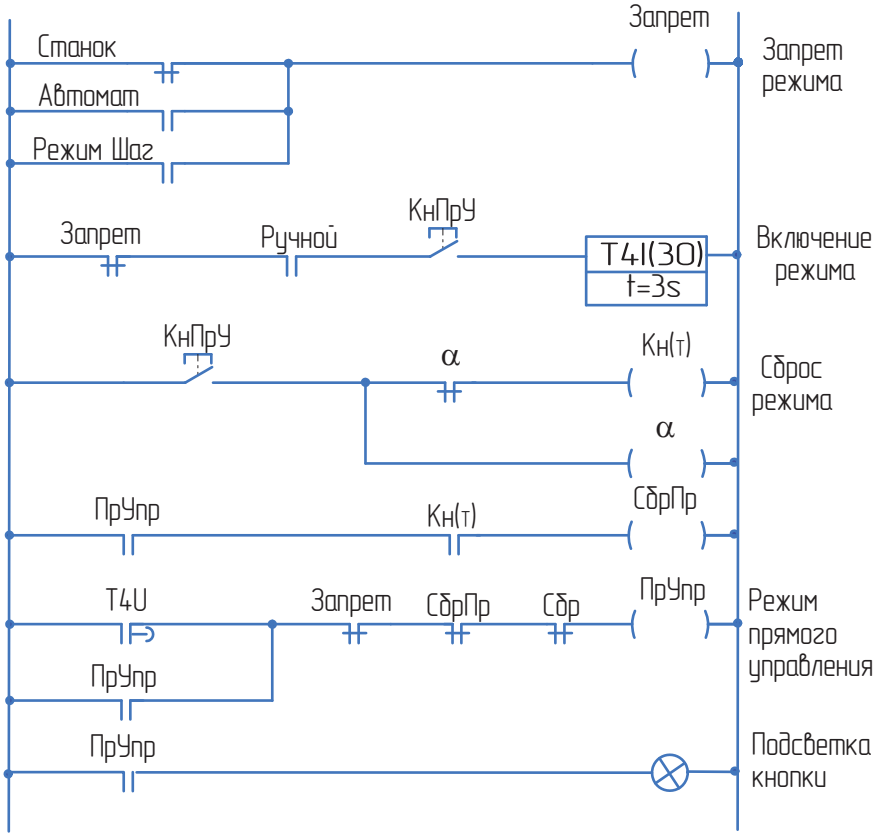


Рис. 2.70. Активизация режима прямого управления

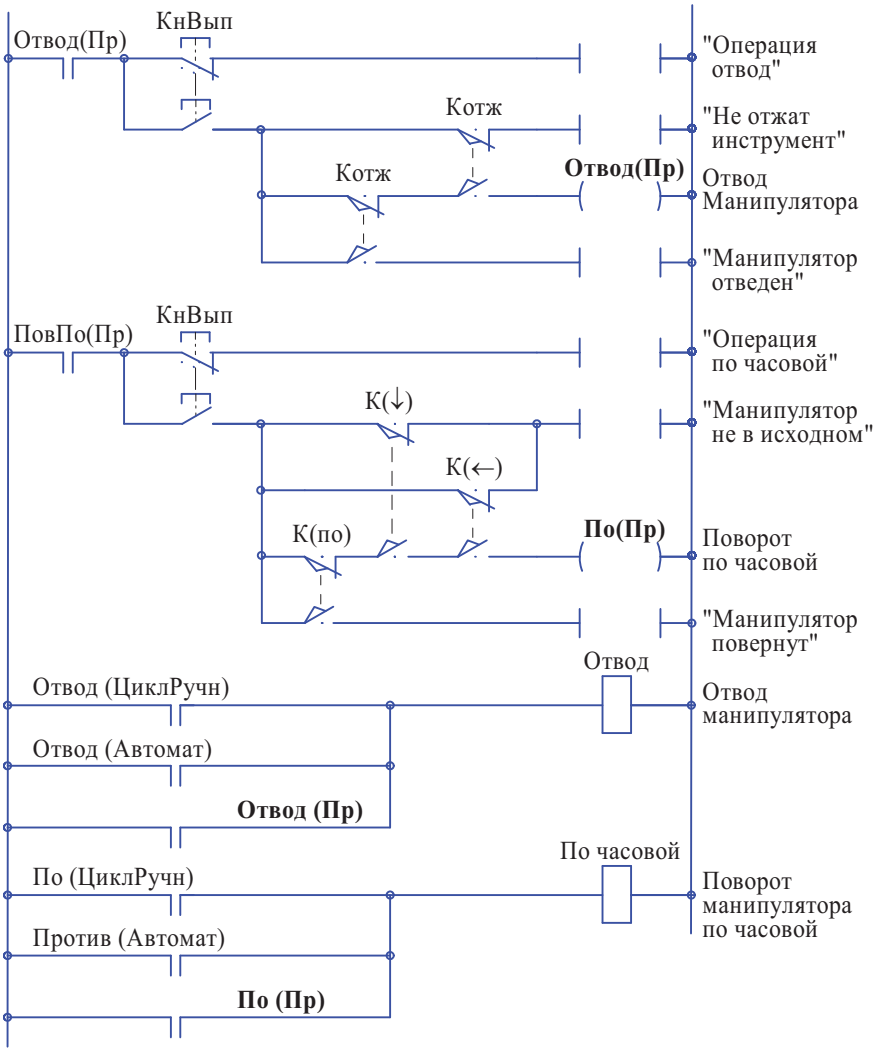


Рис. 2.71. Прямое управление методом перебора и активизацией от одной кнопки

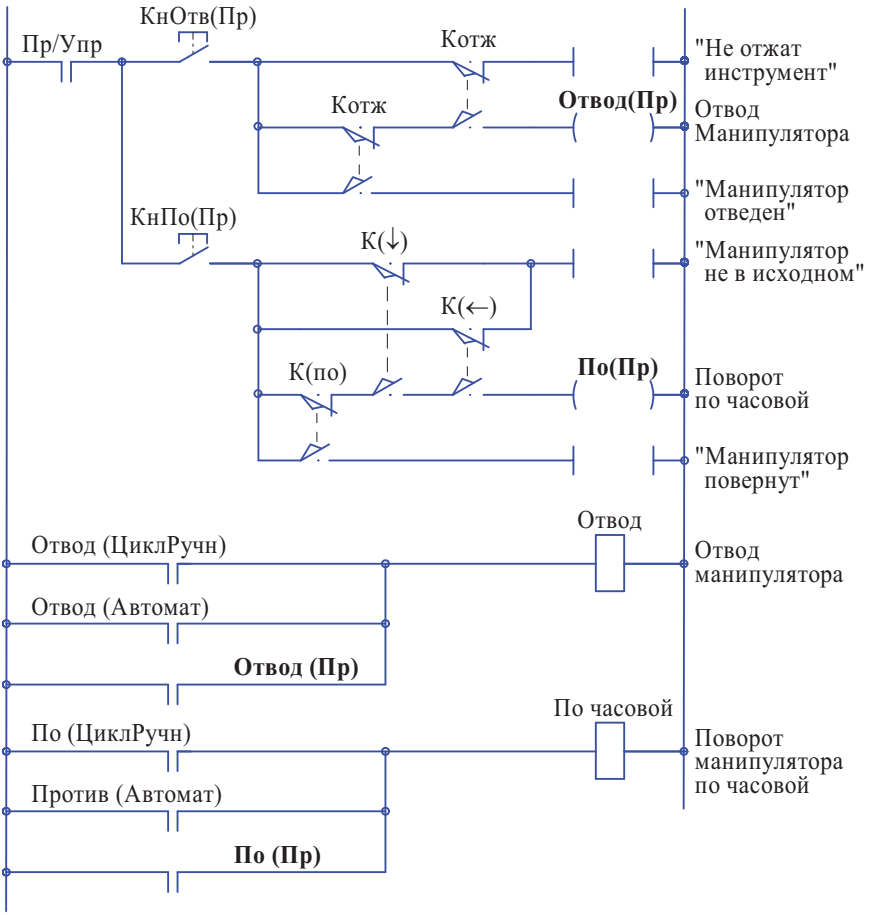


Рис. 2.72. Прямое управление с раздельными кнопками управления

ГЛАВА 3. ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР ТИПА DVP20SX2

3.1. Общие сведения о фирме-производителе

Тайваньская фирма Delta Electronics, основанная в середине девяностых годов прошлого века, является в настоящее время одним из ведущих мировых производителей различных систем автоматизации промышленного электронного оборудования. Это: электроприводы, двигатели, датчики, системы числового программного управления, программируемые контроллеры, панели оператора, источники питания, контрольно-измерительные системы, сетевые решения и др. Разработки делались постепенно, шаг за шагом в течение 25 лет, и развитие продолжается. В головной фирме работает около 3000 человек, из них 400 занимаются научно-исследовательской работой. Офисы продаж и сервиса – по всему миру. Вот пример для подражания, разве мы не в состоянии это сделать? Нужно только начать.

Но вернемся к программируемым контроллерам. Производимая гамма контроллеров поражает (рис. 3.1):

- AH500 и AS300 – высокопроизводительные PLC для масштабных объектов автоматизации;
- DVP-ES2 / EX2 – модульные PLC стандартной функциональности;
- DVP-SS2 / SA2 / SX2 – второе поколение, соответственно, стандартной и повышенной функциональности, со встроенными аналоговыми каналами;
- DVP-SE – компактные PLC с сетевыми возможностями;
- DVP-PM – контроллеры управления движением.

Рис. 3.1. Гамма контроллеров фирмы Дельта

В настоящей главе подробно рассматривается контроллер типа **SX2-11R**, общий вид, компоновка и размеры которого приведены, соответственно, на рис. 3.2 и 3.3.

3.2. Основные технические характеристики

Основные технические характеристики:

- общее питание = 24 В;
- дискретные входы – 8 (= 24 В, 5 мА);
- дискретные выходы – 6 (реле);
- аналоговые входы – 4 (= 5 В, 5 мА);
- аналоговые выходы – 2 (12 бит);
- индикация входов и выходов;
- втычное подключение расширительных блоков Вх / Вых;
- память программ 16 К шагов;
- скорость обработки инструкций: 0,35 мкс (LD), 3,4 мкс (MOV);
- поддержка модулей позиционирования;
- высокоскоростные выходы: 2×100 кГц, 2×10 кГц;
- высокоскоростные счетчики 2×100 кГц, 6×10 кГц;
- поддержка линейной и круговой интерполяции;
- встроенные порты: 1×RS232, 2×RS485, 1×USB;
- два встроенных потенциометра;
- поддержка протоколов MODBUS и PLC Link;
- программное обеспечение WplSoftV3.11.

Ниже приведена система расшифровки заказного кода процессорного блока контроллера:

```
DVP  -xx    -xx    -xx   -x    x    Модуль процессора
  !    !      !      !     !    !__  Версия
  !    !      !      !     !_______  Тип выхода; R – реле
  !    !      !      !               T – транзистор NPN
  !    !      !      !               S – транзистор PNP
  !    !      !      !               M – дифф. сигнал
  !    !      !      !               N – нет выходов
  !    !      !      !_____________  Питание:  00 – пер. 220 В
  !    !      !                      11 – пост. 24 В
  !    !      !____________________  Тип ES2/EX2/SS2/SA2/SX2/SV2/PM
  !    !_____________________________  Общее число входов/выходов
  !__________________________________  Серия контроллеров
```

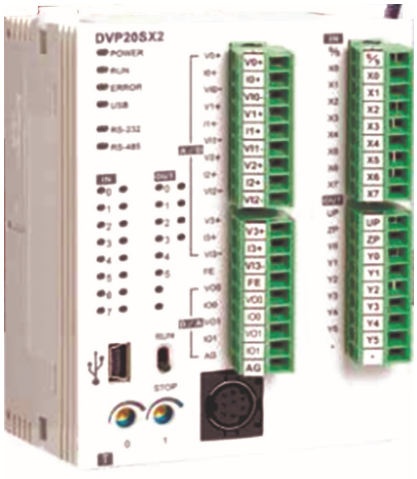


Рис. 3.2. Общий вид контроллера

Компоновка и размеры изделия

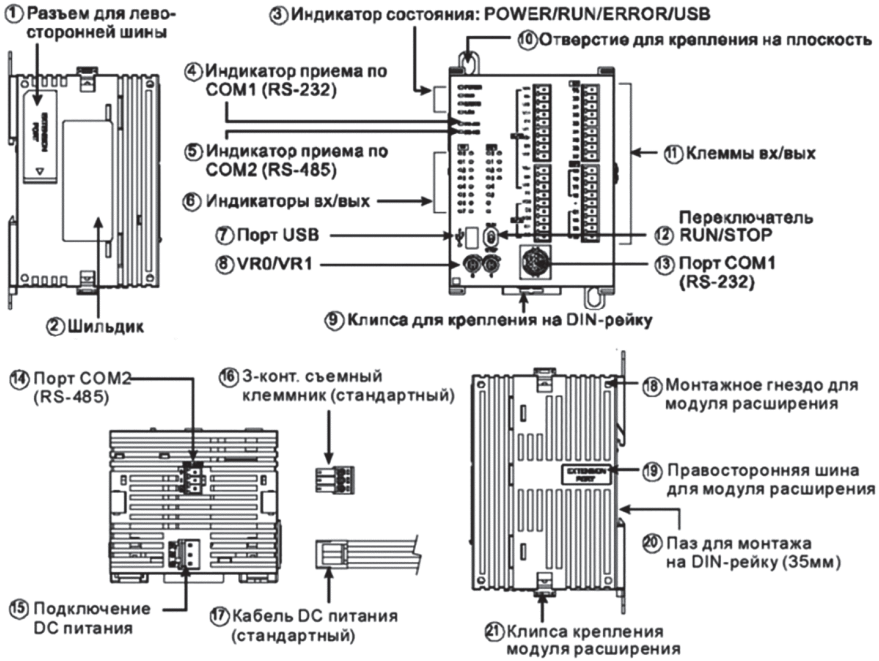


Рис. 3.3. Компоновка и габаритные размеры контроллера

Обращаем внимание на буквы кодирования выходов: R – релейные, T – транзисторные NPN, S – транзисторные PNP. Это очень важно при чтении документации, во избежание ошибок при подключении.

3.3. Аппаратное подключение процессорного модуля

Подключение общего питания контроллера **= 24 В** осуществляется со стороны дна его корпуса (см. рис. 3.3).

Подключение 8 **дискретных** входов одинаково для всех модификаций контроллера. Напряжение питания = 24 В любой полярности. Входы X0…X7 имеют общий провод S/S. Назначение клемм приведено в табл. 3.1, а принципиальная схема подключение на рис. 3.4.

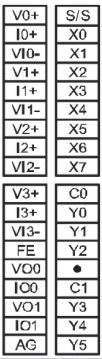
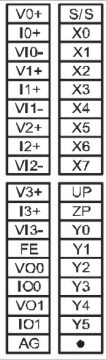
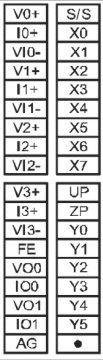
Подключение 6 **дискретных** выходов различно в зависимости от модификации контроллера (см. табл. 3.1, рис. 3.4 и 3.5).

Технические характеристики дискретных входов:

- напряжение питания = 24 В любой полярности;
- входной ток 5 мА;
- уровень логической единицы > 15 В, логического нуля < 5 В;
- время отклика на включение (2,5–20) мкс, на выключение (5–50) мкс в зависимости от номера входа;
- предусмотрен фильтр входного сигнала, параметр D1020 = (0…20) мс;
- все входы объединены на один общий провод S/S.

Таблица 3.1

Подключение входов и выходов

Модель	Входы		Выходы		Конфигурация вх/вых		
	Точки	Тип	Точки	Тип	Реле	NPN	PNP
20SX211R	8	DC (PNP или NPN)	6	Реле			
20SX211T				NPN транзистор			
20SX211S				NPN транзистор			
SX2-R/T/S	4	Аналоговый	2	Аналоговый			

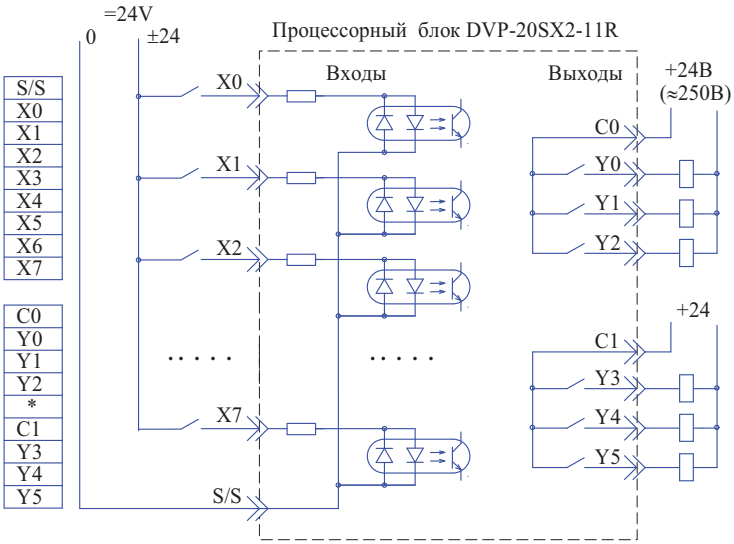


Рис. 3.4. Схема подключения дискретных входов и выходов исп. 11R

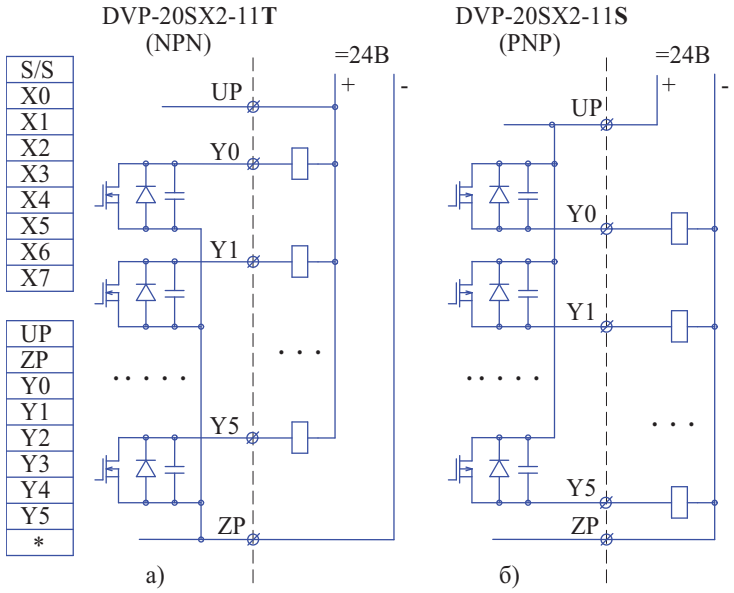


Рис. 3.5. Схема подключения дискретных выходов исп. 11T (а) и 11S (б)

Технические характеристики дискретных выходов исполнений R и T сведены в табл. 3.2.

Таблица 3.2

Технические характеристики выходов

Параметр \ Тип		Выходы			
		Реле - R	Транзистор -T		
Номер выхода		Y0 … Y5	Y0, Y2	Y1, Y3	Y4, Y5
Макс. частота		1 Гц	100 kГц	10 kГц	1 kГц
Рабочее напряжение		250 VAC, < 30 VDC	5 ... 30 VDC #1		
Макс. нагрузка	Резистивная	1.5 A/1 точка (5 A/COM)	SX211T: 0.5A/1 point (3A/ZP) SX211S: 0.3A/1 point (1.8A/UP)		
	Индуктивная	#2	15 W (30 VDC)		
	Лампы	20 Вт DC/100 Вт AC	2.5 W (30 VDC)		
Время отклика	Выкл. → Вкл.	прим. 10 мс	2 мкс	20 мкс	100 мкс
	Вкл. → Выкл.		3 мкс	30 мкс	100 мкс

Релейные выходы имеют предельную частоту коммутации равную 1 Гц.

При работе с высокоскоростными инструкциями необходимо заказывать модули с транзисторными выходами на 100 кгц.

На рис. 3.6 и 3.7 показано, соответственно, подключение **аналоговых** входов и выходов, встроенных в процессорный блок. Разрядность рабочих сигналов равна 12 бит.

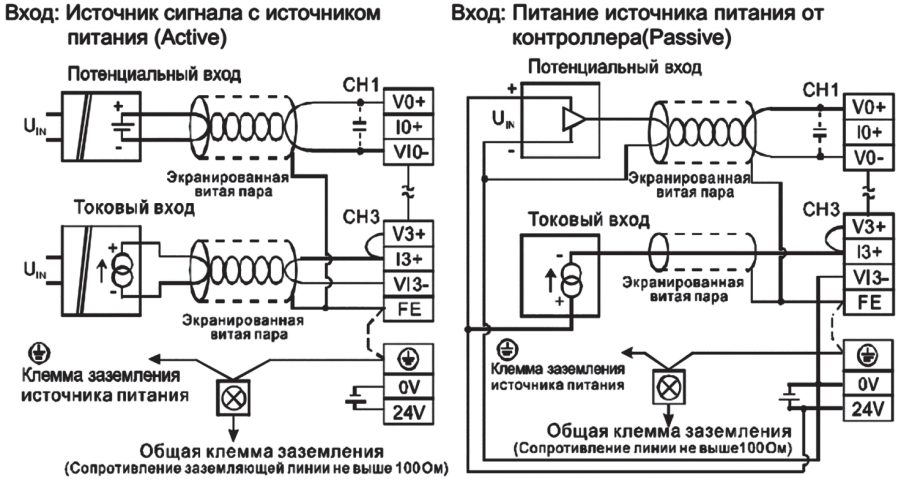


Рис. 3.6. Схемы подключения аналоговых входов

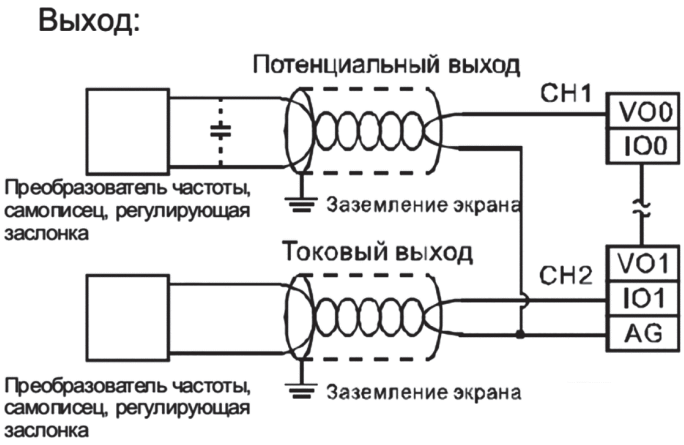


Рис. 3.7. Схема подключения аналоговых выходов

Предусмотрено 4 аналоговых **входных** канала:
- 1-й канал CH1, клеммы V0+, VI0–;
- 2-й канал CH2, клеммы V1+, VI1–;
- 3-й канал CH3, клеммы V3+, VI3–;
- 4-й канал CH4, клеммы V4+, VI4–;
- общая точка, клемма PE.

Характеристики каналов определяются следующими параметрами:

D1114/b0…D1114/b3 сигналы активизации входа

D1115/b0…D1115/b3 определяют тип сигнала входа

(b0…b3) = 0 входной сигнал напряжение ±(0–10) В;

(b0…b3) = 1 токовый входной сигнал ±(0–20) мА;

D1110…D1113 среднее значение сигнала;

D1118 время выборки, мс.

Предусмотрено 2 аналоговых **выходных** канала:
- 1-й канал CH1, клеммы VO0 (напряжение), VI0 (ток);
- 2-й канал CH2, клеммы VO1, VI1;
- общая точка, клемма AG.

Характеристики каналов определяются следующими параметрами:

D1115/b4, D1115/b5 определяют тип выходного сигнала

(b4, b5) = 0 выходной сигнал напряжение ±(0–10) В;

(b4, b5) = 1 токовый выходной сигнал ±(0–20) мА;

D1116 выход 1-го канала (код ±2000 → ±10 В или ±20 мА;

D1117 выход 2-го канала (код ±2000 → ±10 В или ±20 мА.

3.4. Аппаратное подключение расширительных блоков дискретных входов и выходов

Ниже в качестве примера приведены схемы подключения двух расширительных блоков: модуль на 16 дискретных входов DVP-16SM-1N (рис. 3.8) и комбинированный модуль на 8 дискретных входов и 8 релейных выходов DVP-16SP-11R (рис. 3.9).

Характеристики модулей:

- потребляемая мощность обоих модулей 2 Вт;
- питание входов = 24 В, 5 мА, Sink / Sours;
- входы разбиты на две группы: 1-я X0–X7, 2-я X10–X17, каждая со своим общим проводом S/S. Обращаем внимание, что аббревиатура X0–X17 это не логические адреса, а обозначение клемм на блоке. Адресация устанавливается при конфигурации проекта и будет объяснена позднее;
- релейные выходы (рис. 3.9) допускают коммутацию напряжения до 250 В переменного тока и до 30 В постоянного тока;
- выходы Y1–Y7 объединены в одну группу с общим проводом C0. Допустимый ток одного канала 1,5 А, всего блока 5 А.

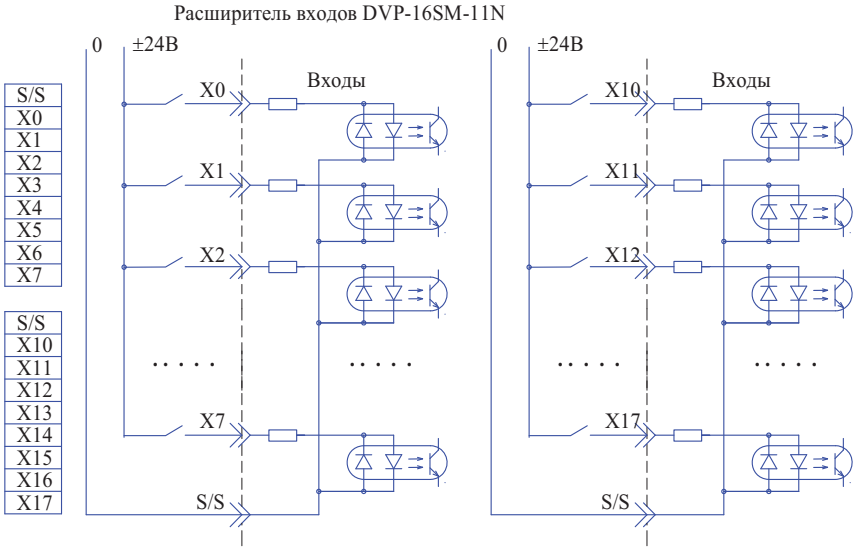


Рис. 3.8. Схема подключения расширительного модуля DVP-16SM-1N

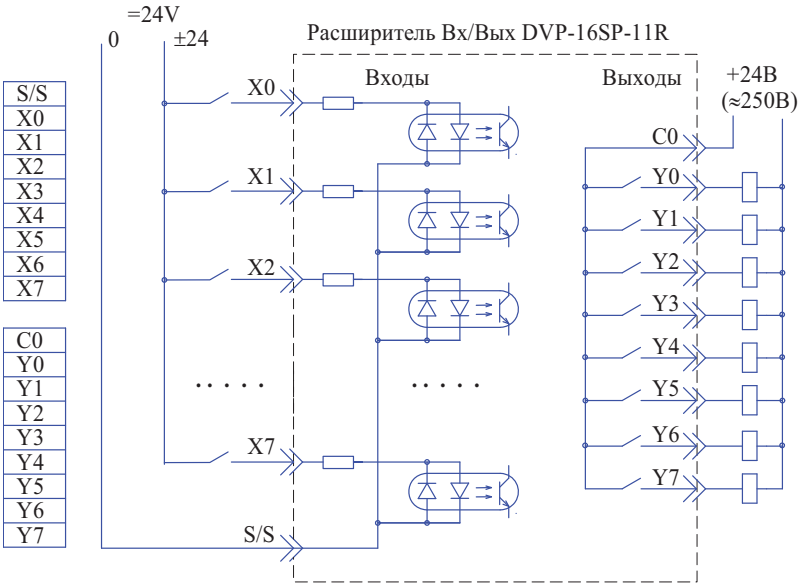


Рис. 3.9. Схема подключения расширительного модуля DVP-16SP-11R

3.5. Аппаратное подключение модуля позиционирования

Модуль позиционирования типа DVP-01PU-S предназначен для управления скоростью или положением одной координатной осью. В качестве привода может использоваться регулируемый, серво или шаговый привод. Подробно работа с модулем будет рассмотрена несколько позднее, здесь отметим основные моменты:

- модуль требует **отдельного** источника питания +24 В;
- подключение модулей позиционирования к другим модулям контроллера – втычное. Программные адреса 0…7 модулей автоматически устанавливаются по порядку их подключения;
- управление подключенным к модулю приводом может осуществляться как полностью из программы электроавтоматики, так и от 5 дискретных входных сигналов напряжением = 24 В, непосредственно подключаемых к модулю. Полярность подключения входов любая;
- задание режимов работы модулей осуществляется при помощи специальных CR – регистров при программировании, при этом используются инструкции FROM (считывание) и TO (Запись).

Схема подключения модуля приведена на рис. 3.10, а назначение и характеристики его входов и выходов сведены в табл. 3.3.

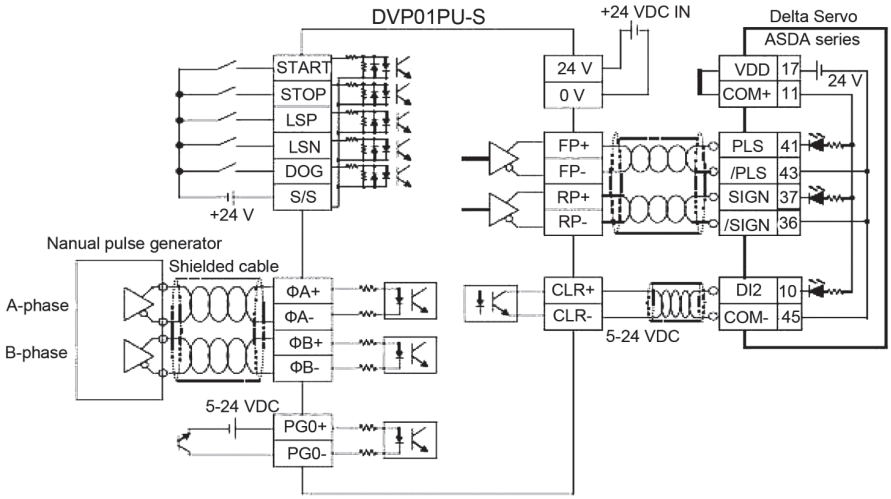


Рис. 3.10. Схема подключения модуля позиционирования DVP-01PU-S

Таблица 3.3

Технические характеристики выходов

Описание	Имя входа	Назначение	Время реакции
Вход питания	+24 В, 0 В	Вход питания 24 VDC (−15÷+20 %) Ток потребления 70 ±10 мА. Пиковый ток при включении 1,3 А	–
Входы	START	Вход сигнала ПУСК	4 мсек/ 12 мсек
	STOP	Вход сигнала СТОП	4 мсек
	LSP/LSN	Входы ограничителей справа и слева	1 мсек
	A+/A–	Импульсный вход A, линейный драйвер	200 кГц
	B+/B–	Импульсный вход B, линейный драйвер	200 кГц
	PG0+/PG0–	Импульсный вход Z, линейный драйвер	4 мсек
	DOG	Вход имеет две функции по режимам работы 1) сигнал достижения нулевого положения 2) сигнал пуска при прерывании в режиме скорости	1 мсек
	S/S	Общий для сигналов START, STOP, DOC, LSP, LSN	–
Выходы	CLR+/SLR–	Сигнал сброса (сброс счетчика для сервопривода)	4 мсек
	FP+/FP–	FP/RP режим: выход сигнала CW I/O режим: выходной импульсный сигнал AB режим: выходной импульсный сигнал фазы –A	200 кГц
	RP+/RP–	FP/RP режим: выход сигнала CCW I/O режим: выходной сигнал направления AB режим: выходной импульсный сигнал фазы –B	200 кГц

Все примеры программ электроавтоматики данной главы базируются на проекте автоматизации ленточной пилы (рис. 3.11 и 3.12).

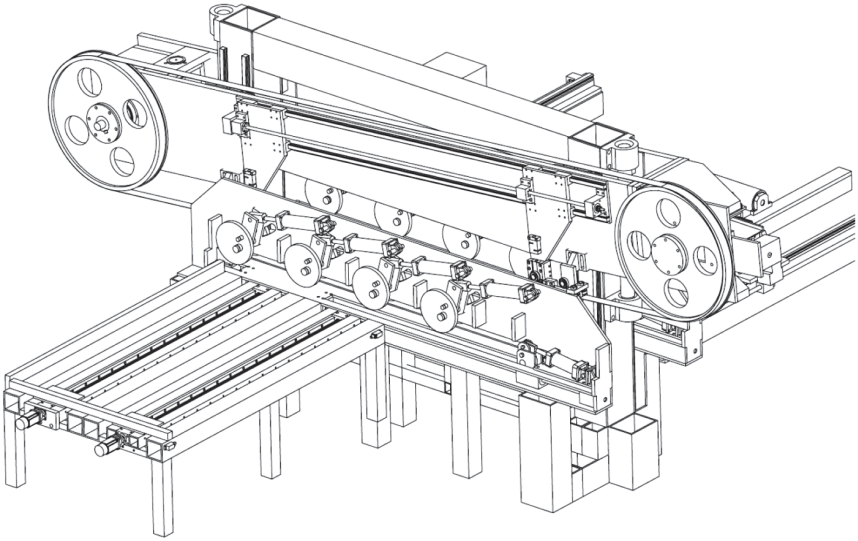


Рис. 3.11. Ленточная пила со стороны начальной подачи листа для отрезания заготовки

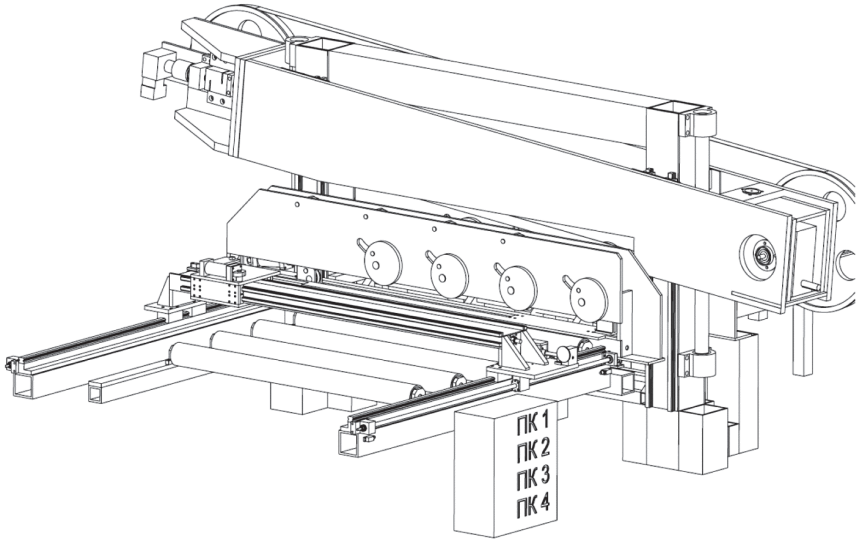


Рис. 3.12. Ленточная пила со стороны подачи заготовок

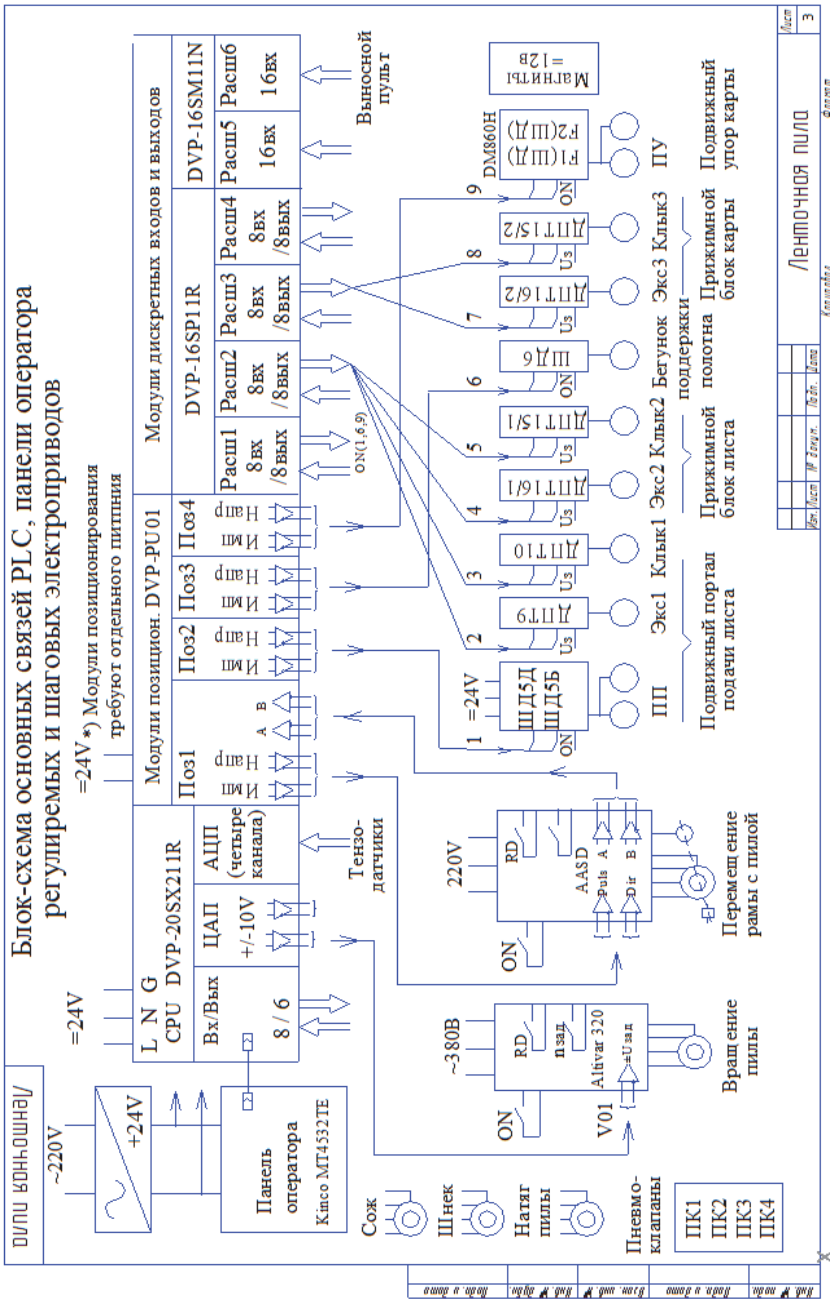


Рис. 3.13. Блок-схема основных связей проекта электрооборудования управления ленточной пилой

3.6. Объект автоматизации

В проекте электрооборудования ленточной пилы используются следующие модули программируемого контроллера:

- блок питания;
- модуль центрального процессора DVP-SX2-11R;
- четыре модуля позиционирования DVP-PU01;
- четыре расширительных модуля дискретных входов и выходов DVP-16SP-11R;
- два расширительных модуля DVP-16SM-11N.

Другое оборудование:

- программируемая панель оператора фирмы Kinco типа MT4532TE;
- частотный преобразователь привода вращения пилы Altivar 320;
- сервопривод перемещения рамы с пилой AASD;
- шесть электроприводов постоянного тока механизмов зажима листа;
- два шаговых электропривода подачи листа;
- четыре пневмо-агрегата управления эксцентриками зажимов и поворотом откидного стола.

Изучив систему подключения модулей, используемых в проекте, и другого электрооборудования, рекомендуется нарисовать общую блок-схему основных связей, например, как показано на рис. 3.13.

Прежде чем рисовать рабочие чертежи на подключение каждого модуля, необходимо разобраться с системой их адресации. Для этого следует изучить доступные для данного контроллера операнды и выполнить процедуру конфигурации проекта, при которой определятся рабочие адреса для каждого модуля.

3.7. Доступные операнды, система адресации, конфигурация проекта

Синтаксис языка программирования контроллера типа DVP-SX2 предусматривает использование следующих операндов:

(X0–X377) – дискретные входы;
(Y0–Y377) – дискретные выходы;
(M0–M4095) – маркеры (промежуточная память);
(T0–T255) – таймеры;
(C0–C231) – счетчики;
(S0–S1023) – шаговые реле;
(D0–D1999) – регистры данных;

(N0–N7) – Master Control Point;
(P0–P255) – указатели переходов;
I – операнды прерывания;
K – десятичные константы;
H – шестнадцатеричные константы.

Подробное назначение операндов приведено в табл. 3.4.

Для формирования адресного пространства дискретных модулей входов и выходов, для установки адресов модулей позиционирования и последующего набора программы необходимо сделать следующее:

– установить на персональный компьютер программное обеспечение, например, WPLSoft версии V2.39 или выше;
– аппаратно собрать контроллер, согласно принятой разработчиком конфигурации;
– соединить компьютер и контроллер покупным кабелем со стандартным USB-разъемом со стороны PC и Mini-USB со стороны PLC;
– подключить питание и установить связь между компьютером и контроллером согласно процедуре, приведенной в сопроводительной документации на контроллер. Адреса дискретных входов и выходов модулей установятся автоматически (рис. 3.14).

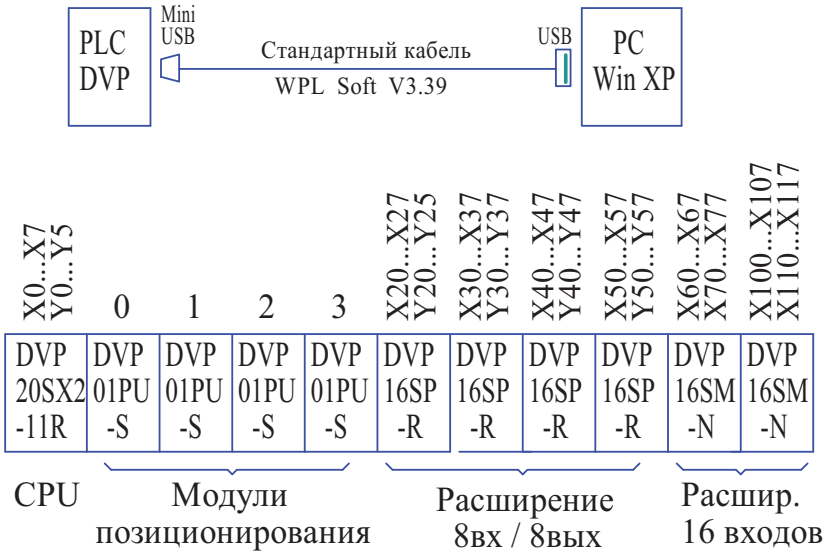


Рис. 3.14. Конфигурация проекта, установка адресов

Таблица 3.4

Доступные операнды языка программирования

Specifications					
Control Method				Stored program, cyclic scan system	
I/O Processing Method				Batch processing method (when END instruction is executed)	
Execution Speed				LD instructions – 0.54μs, MOV instructions – 3.4μs	
Program language				Instruction List + Ladder + SFC	
Program Capacity				15872 steps	
Bit Contacts	X	External inputs		X0~X377, octal number system, 256 points max, (*4)	Total 256+16 I/O
	Y	External outputs		Y0~Y377, octal number system, 256 points max, (*4)	
	M	Auxiliary relay	General	M0~M511, 512 points, (*1) M768~M999, 232 points, (*1) M2000~M2047, 48 points, (*1)	Total 4096 points
			Latched	M512~M767, 256 points, (*2) M2048~M4095, 2048 points, (*2)	
			Special	M1000~M1999, 1000 points, some are latched	
	T	Timer	100ms (M1028=ON, T64~T126: 10ms)	T0~T126, 127 points, (*1) T128~T183, 56 points, (*1)	Total 256 points
				T184~T199 for Subroutines, 16 points, (*1)	
				T250~T255(accumulative), 6 points (*1)	
			10ms (M1038=ON, T200~T245: 1ms)	T200~T239, 40 points, (*1)	
				T240~T245(accumulative), 6 points, (*1)	
			1ms	T127, 1 points, (*1) T246~T249(accumulative), 4 points, (*1)	
	C	Counter	16-bit count up	C0~C111, 112 points, (*1) C128~C199,72 points, (*1)	Total 232 points
				C112~C127,16 points, (*2)	
			32-bit count up/down	C200~C223, 24 points, (*1)	
				C224~C231, 8 points, (*2)	

Продолжение таблицы 3.4

			32bit high-speed count up/down	Soft-ware	C235~C242, 1 phase 1 input, 8 points, (*2)	Total 23 points
					C232~C234, 2 phase 2 input, 3 points, (*2)	
				Hard-ware	C243~C244, 1 phase 1 input, 2 points, (*2)	
					C245~C250, 1 phase 2 input, 6 points, (*2)	
					C251~C254 2 phase 2 input, 4 points, (*2)	
	S	Step point	Initial step point		S0~S9, 10 points, (*2)	Total 1024 points
			Zero point return		S10~S19, 10 points (use with IST instruction), (*2)	
			Latched		S20~S127, 108 points, (*2)	
			General		S128~S911, 784 points, (*1)	
			Alarm		S912~S1023, 112 points, (*2)	
Word Register	T	Current value			T0~T255, 256 words	
	C	Current value			C0~C199, 16-bit counter, 200 words	
					C200~C254, 32-bit counter, 55 words	
	D	Data register	General		D0~D407, 408 words, (*1) D600~D999, 400 words, (*1) D3920~D9999, 6080 words, (*1)	Total 10000 points
			Latched		D408~D599, 192 words, (*2) D2000~D3919, 1920 words, (*2)	
			Special		D1000~D1999, 1000 words, some are latched	
			For Special mudules		D9900~D9999，100 words，(*1), (*5)	
			Index		E0~E7, F0~F7, 16 words, (*1)	
Pointer	N	Master control loop			N0~N7, 8 points	
	P	Pointer			P0~P255, 256 points	
	I	Interrupt Service	External interrupt		I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01: rising-edge trigger ⌐, 00: falling-edge trigger ¬)	

Окончание таблицы 3.4

			Timer interrupt	I602~I699, I702~I799, 2 points (Timer resolution = 1ms)
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080,8 points
			Communication interrupt	I140(COM1), I150(COM2), I160(COM3), 3 points, (*3)
Constant	K	Decimal		K-32,768 ~ K32,767 (16-bit operation), K-2,147,483,648 ~ K2,147,483,647 (32-bit operation)
	H	Hexadecimal		H0000 ~ HFFFF (16-bit operation), H00000000 ~HFFFFFFFF (32-bit operation)
Serial ports				COM1: built-in RS-232 ((Master/Slave) COM2: built-in RS-485 (Master/Slave) COM3: built-in RS-485 (Master/Slave) COM1 is typically the programming port.
Real Time Clock				Year, Month, Day, Week, Hours, Minutes, Seconds
Special I/O Modules				Up to 8 special I/O modules can be connected

Notes:

1. Non-latched area cannot be modified
2. Latched area cannot be modified
3. COM1: built-in RS232 port. COM2: built-in RS485 port. COM3: built-in RS485 port.
4. When input points(X) are expanded to 256 points, only 16 output points(Y) are applicable. Also, when ouput points(Y) are expanded to 256 points, only 16 input points(X) are applicable.
5. This area is applicable only when the ES2/EX2 MPU is connected with special I/O modules. Every special I/O module occupies 10 points.

Обращаем внимание на следующее:

- адресация входов X и выходов Y выполняется в **восьмеричном** коде, т. е. в адресах используются только цифры от 0 до 7;
- в отличие от входов и выходов при адресации маркеров M и регистров D используется сквозная **десятичная** система;
- адреса входов и выходов процессорного блока, соответственно, X0–X7 и Y0–Y5;
- адреса входов X10–X17 и выходов Y10–Y17 не используются;

– адресация расширительных блоков начинается с X20 (входы), Y20 (выходы) и увеличивается в соответствии с порядком подключения модулей относительно процессорного блока;
– адреса блоков позиционирования начинаются с нуля и увеличиваются в соответствии с порядком их подключения относительно процессорного блока;
– адреса встроенных в процессорный блок аналоговых входов и выходов фиксированы и определяются D-параметрами.

3.8. Практические советы по программированию

Контроллеры серии DVP имеют мощный язык программирования электроавтоматики, позволяющий решать практически любые реальные задачи автоматизации. Предусмотрено три языка программирования:
– язык релейно-контактных символов (LD – Ladder Diagram);
– язык мнемонических инструкций (IL – Instruction List);
– язык последовательных функциональных инструкций (SFC – Sequential Function Chat.

Сотни инструкций. Полностью разобраться в этом огромном массиве информации довольно сложно, да и не нужно.

Вне зависимости от типа контроллера, на каком бы языке в итоге не писалась программа, автор настоятельно рекомендует предварительно разрабатывать алгоритмы управления, используя релейно-контактную формализацию. Это просто, понятно, легко читается. В этой связи, безусловное предпочтение следует отдать языку релейно-контактных символов, позволяющему легко отлаживать программы в режиме ON-Line на экране компьютера. Кроме того, если уже имеются релейные принципиальные схемы управления, то их легко перевести в РКС-программу контроллера.

Программирование на языке мнемокода значительно сложнее, чтение и отладка программ трудоемки и такой язык следует применять только в контроллерах, где отсутствует язык РКС. Сложность программ мнемокода будет показана ниже, поскольку в данном контроллере можно легко транслировать программу РКС в программу мнемокода, и наоборот.

Программирование на языке последовательных инструкций подробно рассмотрено ниже в главе 4. Оно носит весьма специфический характер и трудоемко, автор имеет небольшой опыт работы с данным языком, однако абсолютно уверен, что при локальной автоматизации его применять не следует, предпочтительнее программировать на РКС. Приведенные в книге сравнительные примеры помогут читателю принять собственное решение.

Итак, мы имеем несколько языков и огромное число инструкций. Пользователь должен уметь отличить «зерна от плевел», отобрать ограниченное количество наиболее важных инструкций, научиться ими пользоваться и не вдаваться в дальнейшие подробности, иначе их можно изучать всю жизнь. Приведу некоторые практические соображения по методологии проектирования электроавтоматики. Можно сформулировать два подхода к решению поставленной задачи.

Подход *первый* – отталкиваемся от типа контроллера. Если пользователь специализируется на одном типе контроллера, или, в крайнем случае, ограниченном их числе, то естественно, следует их досконально изучить и максимально использовать все возможности языка, создавать «красивые» и оптимальные программы. Но алгоритмы, заложенные в этих программах, из-за применения специфических только для данного типа контроллера инструкций, чаще всего будут непригодными для использования в других типах контроллеров, т. е. не будут универсальными. Придется каждый раз искать разные решения.

Подход *второй* – отталкиваемся от типов объектов автоматизации. Если разработчик работает с разными типами, например, станков и разными типами контроллеров, причем один и тот же тип станка по просьбам заказчиков может оснащаться разными типами контроллеров (или систем ЧПУ), то гораздо эффективнее использовать типовые решения. Для этого необходимо проанализировать решаемые задачи, разделить их на типовые блоки и разработать шаблоны алгоритмов программ с использованием ограниченного числа инструкций. При этом следует применять универсальные инструкции, присутствующие в наборе, практически, всех типов контроллеров. При таком подходе общие алгоритмы управления можно компоновать из готовых «кубиков», а при написании программ только адаптировать их к синтаксису конкретного языка.

Сроки разработки и отладки программ значительно сокращаются.

Следующее соображение касается изучения синтаксиса нового для пользователя языка, набора и отладки программ. Признаком правильной организации работы будет изучение, набор и предварительная отладка программы на стенде. Это позволит сократить сроки и избежать большинства ошибок при отладке реальных механизмов. Изготовить простой и удобный универсальный стенд не так и сложно. На рис. 3.15 приведена фотография стенда, на который можно легко установить, практически, любой тип контроллера и панели оператора. Автономный имитатор станка со встроенным источником питания и двухполярными светодиодами позволяет подключить 128 дискретных входов и 96 выходов, что в большинстве случаев достаточно. Если контроллер или система ЧПУ имеют разъемное подключение, то меняются только соединительные кабели. В заводских условиях не представляет особой трудности дооснастить стенд и электроприводами, питающимися от трехфазной промышленной сети.

Любую программу начинайте с формирования специальных ячеек, значение которых всегда равно единице и всегда равно нулю, присвоив им легко запоминающиеся адреса, например, M0 (= 0) и M1 (= 1), также нескольких генераторов, например, с периодами M2 (0,5 с) и M3 (1 с). Такие ячейки всегда должны быть под рукой для резервирования места и формирования мигающих сообщений. Как правило, во всех контроллерах уже имеются такие системные ячейки, однако их адреса трудно запомнить.

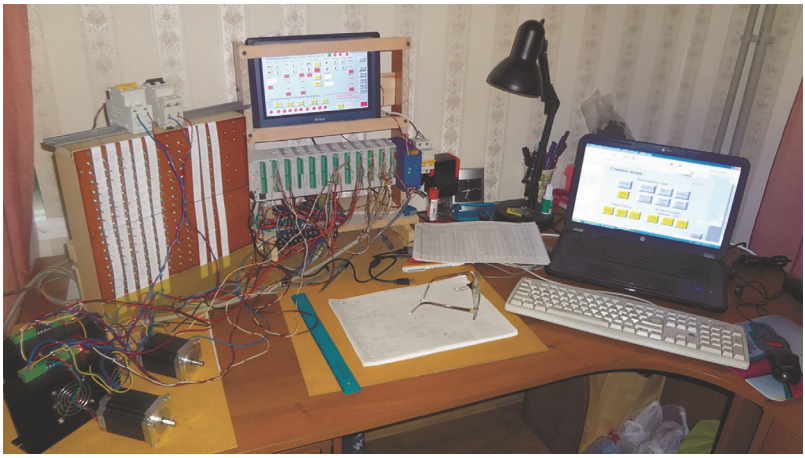


Рис. 3.15. Стенд отладки программ электроавтоматики

Итак, постоянно систематизируйте свои разработки, изучайте и анализируйте разработки других специалистов, сравнивайте их со своими, применяйте лучшие решения и никогда не стесняйтесь задавать вопросы, если что-то непонятно. Если зациклились при решении какого-либо вопроса, оставьте его на какое-то время, займитесь другим, идея должна созреть, и все получится.

3.9. Обзор синтаксиса языка релейно-контактных схем (РКС-алгоритмов, Ladder-диаграмм)

Релейно-контактные языки программирования электроавтоматики принято разбивать на две группы, **базовые**, осуществляющие простейшие логические операции с битовыми операндами и **функциональные**, предназначенные для расширения возможностей языка и могущие работать со словами.

К *базовым* относятся следующие команды:

– считать первый прямой или инверсный операнд (контакт) логической релейной цепи;

– логически умножить на прямой или инверсный операнд;
– логически сложить с прямым или инверсны операндом;
– послать результат вычисления на выход.

Такие команды есть в синтаксисе любого контроллера, отличие состоит только в аббревиатурах их написания. К базовым командам следует отнести еще инверсию выхода, исключающее ИЛИ (неравнозначность) и скобки, если такие команды присутствуют в языке, а также таймеры и счетчики. Формально, при помощи базовых команд можно написать любую реальную цикловую программу электроавтоматики управления локальными объектами, если владеть методологией проектирования алгоритмов, изложенной в главе 2.

Однако, используя функциональные инструкции, можно серьезно упростить процедуру синтеза.

Битовые операнды это дискретные входы X, дискретные выходы Y, ячейки оперативной памяти M, шаговые реле S.

Внимание! Если при проектировании не используется язык последовательных инструкций, в качестве оперативной памяти дополнительно можно использовать шаговые реле **S**. Это в два раза расширяет адресное пространство и особенно удобно при внесении добавлений при отладке программ.

Наиболее применяемые *функциональные* инструкции, например, это:
– пересылка и сортировка данных;
– команды сравнения на равенство, большее или меньшее значение;
– шифраторы, дешифраторы и преобразование кодов;
– формирование тактовых сигналов;
– арифметические вычисления;
– формирование регистров;
– условные переходы;
– сбросы и многие другие (о которых в большинстве применений можно забыть!).

Если речь идет о решении задач позиционирования, работы с быстрыми счетчиками, выполнения арифметических и тригонометрических вычислений, то здесь без функциональных инструкций не обойтись.

Еще раз подчеркнем, любую из перечисленных выше функциональных инструкций можно реализовать при помощи базовых команд, но это будет неэффективно и сложно. Иногда это приходится делать. Вспоминается, для примера, первый проект с контроллером движения американской фирмы Дельта-Тау. У автора не было достаточной документации на систему, никак не удавалось понять программирование выдержки времени, консультаций не было, а сроки сдачи станка поджимали. Было принято решение реализовать временные задержки на основе счета числа вычислительных циклов контроллера. Задача

была решена, станок сдан заказчику. Когда об этом позднее узнали производители системы, то сказали «Ну, ты даешь!» и стали уважать. Так что не нужно бояться нестандартных решений.

Графический набор базовых команд осуществляется перетаскиванием при помощи мышки соответствующих иконок в нужное место наборного поля компьютера с последующим набором нужного адреса.

Основные из них следующие:

– нормально открытый контакт;
– нормально закрытый контакт;
– выходная катушка реле;
– тактированный контакт по переднему фронту;
– тактированный контакт по заднему фронту;
– стрелка тактирования по переднему фронту;
– стрелка тактирования по заднему фронту;
– знак инверсии результата;
– вставка и стирание горизонтальной линии;
– вставка и стирание вертикальной линии.

Набор программы в мнемокоде осуществляется вызовом режима и последующим заполнением таблицы. Если уже набрана РКС-программа, то ее легко конвертировать в мнемокод.

Графический набор функциональных команд осуществляется заполнением специального окна, вызываемого иконкой «Функциональная команда».

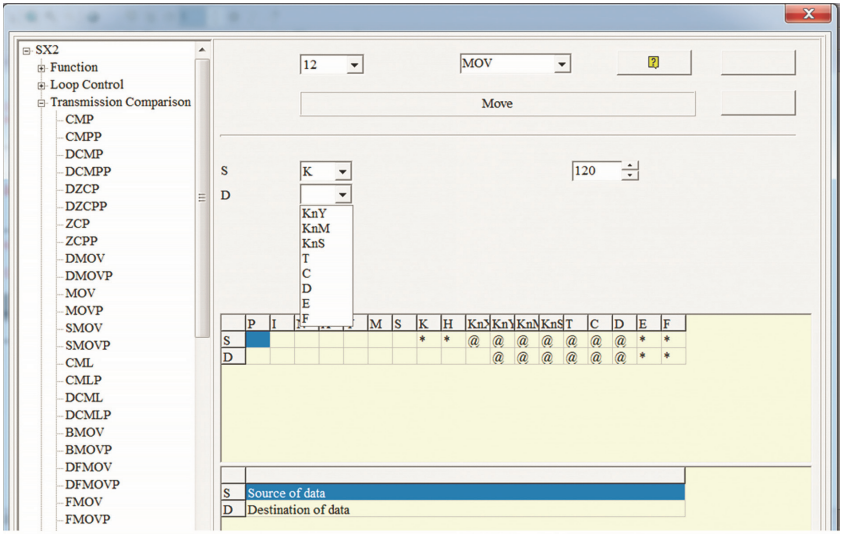


Рис. 3.16. Окно набора функциональной инструкции MOV

Рис. 3.16 показывает, что вызвать инструкцию MOV можно из трех мест окна набора:

– непосредственно по номеру инструкции API = 12;
– отметив инструкцию в левом падающем меню;
– буквенным набором MOV в горизонтальном окне команды.

В окне набора приводятся подсказки:

– назначение символьных операндов S (Source of Data) и D (Destination of Data);
– справочная горизонтальная таблица доступных операндов;
– падающее меню доступных операндов для ввода (KnY, KnM, KnS,T, C, D, E, F).

Таким образом, ошибиться в синтаксисе практически невозможно.
Приступим к изучению базовых команд.

3.9.1. Базовые команды

На рис. 3.17 приведены РКС-алгоритмы и мнемокоды основных команд:

а) LD X0 – считывание прямого операнда;
   LDI X0 – считывание инверсного операнда;
б) AND X1 – логическое умножение X0 на прямой операнд X1;
   ANI X1 – логическое умножение X0 на инверсный операнд X1;
в) OR X2 – логическое сложение X1 с прямым операндом X2;
   ORI X2 – логическое сложение X1 с инверсным операндом X2;
г) OUT Y0 – посылка результата логических вычислений на выход Y0;
д) \ – инверсная посылка результата логических вычислений на выход Y0.

В языке используются аббревиатуры от английских слов:
LD (Load – загрузить);
LDI (Load Implement – загрузить инверсно);
AND (AND – И);
ANI (AND Implement – И инверсно);
OR (OR – ИЛИ);
ORI (OR Implement – ИЛИ инверсно);
OUT (Output – Выход).

Вспомним, что в примере 1-ой главы с контроллером фирмы Фудзи для аналогичных команд использовались совершенно другие аббревиатуры:

R (Read – читать), A (And), O (Or), N (Negativ), W (Write – записать) и, соответственно: RN, AN, ON.

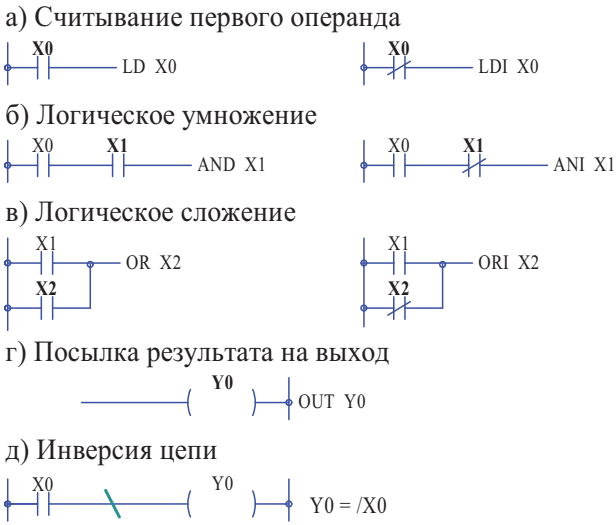


Рис. 3.17. Основные базовые инструкции

Языки мнемокода называют также аккумуляторными языками, поскольку все промежуточные результаты логических вычислений хранятся в специальной памяти – «Аккумуляторе».

В табл. 3.5 показан пример последовательности программирования двух простейших цепочек Y0 = (X0·/X1 + X2)·X3 и Y1 = (X0·/X1)·X2 + X3 на двух языках: РКС и мнемокода.

Подробно последовательность обработки процессором логического уравнения Y0 = (X0·/X1 + X2)·X3 приведена в табл. 3.6. Казалось бы все просто и понятно, но это только для простейших логических последовательных уравнений (алгоритмов).

На рис. 3.18 приведено три сложных релейных алгоритма и их аналоги на языке мнемокода. Обратимся к рис. 3.18, *а*, где показан РКС-алгоритм, соответствующий логическому уравнению Y1 = (X0 + X1·/ X2)·(X3 + X4)·X5. Для него нельзя непосредственно написать программу в мнемокоде, используя рассмотренные выше команды. Это связано с тем, что во втором шаге программы операнд X0 необходимо сложить с результатом перемножения X1·/X2, которого еще нет.

Таблица 3.5

Пример пошагового программирования простейших цепочек

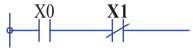
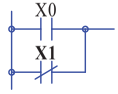
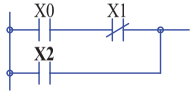
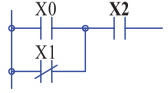
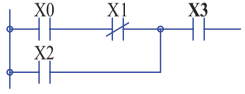
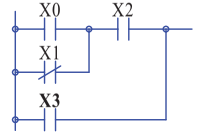
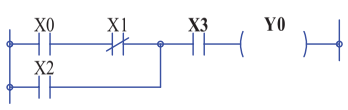
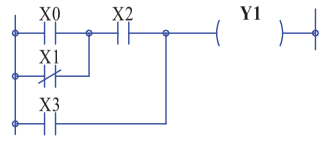
Y0= (X0*/X1+X2)*X3			Y1= (X0+/X1)*X2+X3		
Процесс набора РКС- Алгоритма	Программа мнемокода	Процесс вычисления в аккумулятре	Процесс набора РКС- Алгоритма	Программа мнемокода	Процесс вычисления в аккумулятре
	LD X0	Acc=X0		LD X0	Acc=X0
	ANI X1	Acc=X0*/X1		ORI X1	Acc=X0+/X1
	OR X2	Acc=X0*/X1 +X2		AND X2	Acc=(X0+*/X1) *X2
	AND X3	Acc=(X0*/X1+X2) *X3		OR X3	Acc=(X0+/X1)*X2 +X3
	OUT Y0	Y0=Acc =(X0*/X1+X2) *X3		OUT Y1	Y1=Acc =(X0+/X1)*X2 +X3

Таблица 3.6

Последовательность обработки программы Y0 = (X0·/X1 + X2)·X3

Шаг программы	Программа	Комментарий
1	LD X0	Считывание первого операнда X0 и запись его значения в аккумулятор (Акк = X0)
2	ANI X1	Считывание инверсного операнда X1 и умножение на содержимое аккумулятора. Результат действия снова записывается в аккумулятор (Акк = X·/X1)
3	OR X2	Считывание прямого операнда X2 и сложение с содержимым аккумулятора. Результат действия снова записывается в аккумулятор Акк = X0·/X1 + X2
4	AND X3	Считывание прямого операнда X3 и умножение на содержимое аккумулятора. Результат действия снова записывается в аккумулятор Акк = (X0·/X1 + X2) ·X3
5	OUT Y0	Посылка результата вычисления из аккумулятора на выход Y0

Как выйти из этой ситуации? Вариантов может быть два.

Первый: заранее вычислить действие X1·/X2 и его результат записать в промежуточную ячейку, например M10 = X1·/X2, а затем сложить (X0 + M10).

Второй: переставить слагаемые местами X1·/X2 + X0 и сделать вычисления рассмотренным ранее методом.

Однако дальше необходимо вновь произвести логическое умножение на скобки (X3 + X4) и здесь, кроме предварительного промежуточного сложения, ничего сделать нельзя. Очевидно, что такое программирование совершенно недопустимо. Для этого в аппаратную структуру контроллера вводится стековая память (см. гл. 1), а в системы команд специальные безадресные инструкции. Процесс программирования приведен в табл. 3.7

Из приведенных примеров видно, что программирование на языке мнемокода достаточно сложно и трудоемко. Программируя на различных типах контроллеров нужно также знать общую разрядность стека и, если она мала, постоянно следить, чтобы не было его переполнения.

В целом, можно сделать вывод, что если в перечне доступных языков контроллера присутствует язык РКС, то следует применять только его. При этом для каждого типа контроллера необходимо разобраться, какие структуры запрещены (рис. 3.19), а какие нет (рис. 3.20, рис. 3.21).

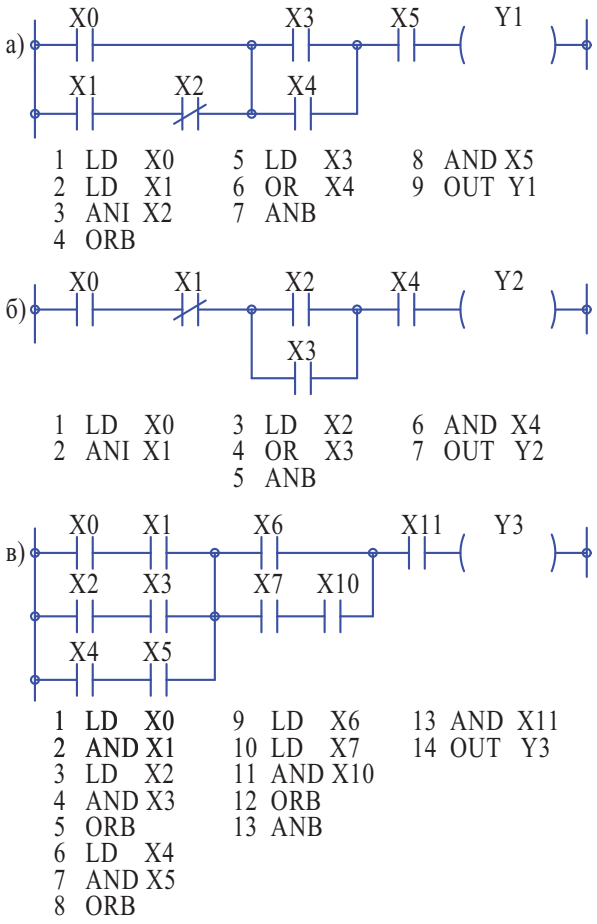


Рис. 3.18. Примеры РКС-алгоритмов и их программы на языке мнемокода

Таблица 3.7

Последовательность обработки программы Y1 = (X0 + X1·/X2)·(X3 + X4)·X5

Шаг программы	Программа	Комментарий
1	LD X0	Считывание первого операнда X0 и запись его значения в аккумулятор (Акк = X0)
2	LD X1	Считывание и запись прямого операнда X1 в аккумулятор с предварительным проталкиванием его (аккумулятора) содержимого в стек. Результат: Стек = X0, Акк = X1

Окончание таблицы 3.7

Шаг программы	Программа	Комментарий
3	ANI X2	Считывание инверсного операнда X2 и умножение на содержимое аккумулятора. Результат действия снова записывается в аккумулятор Акк = X1·/X2. Содержимое стека без изменений (X0)
4	**ORB**	Безадресная команда сложения содержимого стека X0 и аккумулятора X1·/X2 с последующим выталкиванием результата в аккумулятор Акк = X0 + X1·/X2
5	LD X3	Считывание и запись прямого операнда X3 в аккумулятор с предварительным проталкиванием его (аккумулятора) содержимого в стек. Результат: Стек = X0 + X1·/X2, Акк = X3
6	OR X4	Считывание прямого значения X4, сложение с содержимым аккумулятора X3. Результат сложения снова записывается в аккумулятор, содержимое стека не изменяется. Итак: Акк = X3 + X4, Стек = X0 + X1·/X2
7	**ANB**	Безадресная команда умножения содержимого стека X0+X1·/X2 и аккумулятора X1+X4 с последующим выталкиванием результата в аккумулятор Акк = (X0 + X1·/X2)·(X3 + X4)
8	AND X5	Считывание прямого значения X5 и умножение на содержимое аккумулятора (X0 + X1·/X2)·(X3 + X4). Результат умножения снова записывается в аккумулятор Итак: Акк = (X0 + X1·/X2)·(X3 + X4)·X5
9	OUT Y0	Посылка результата вычисления из аккумулятора на выход Y1

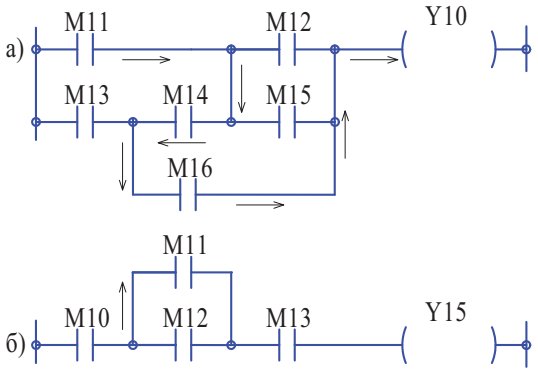


Рис. 3.19. Запрещенные варианты набора РКС-алгоритмов

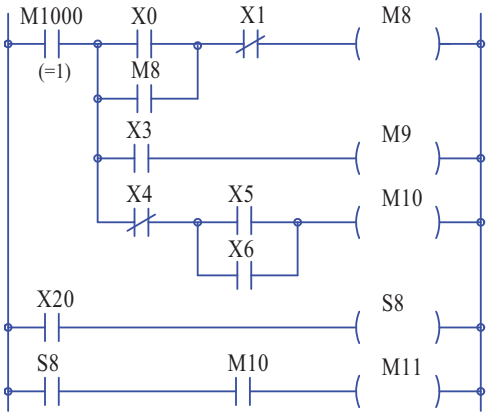


Рис. 3.20. Разрешенный разветвленный РКС-алгоритм

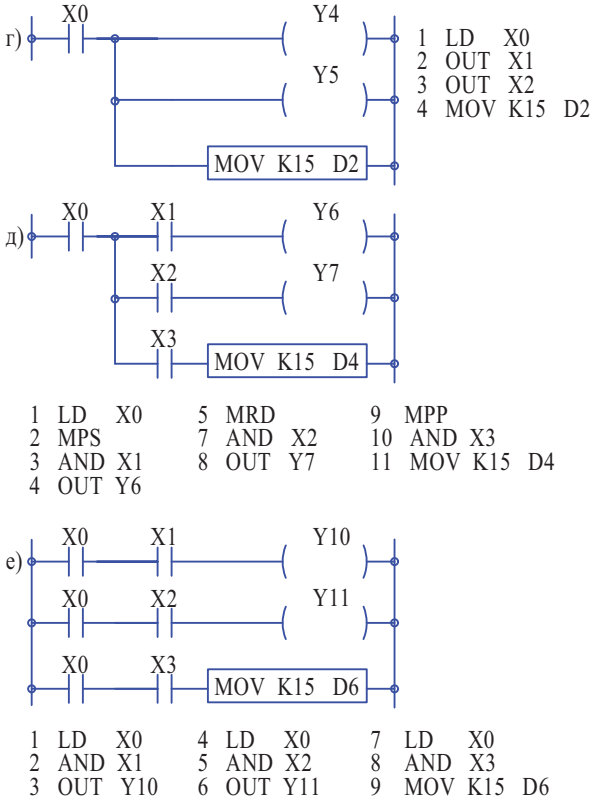


Рис. 3.21. Разрешенные варианты набора РКС-алгоритмов

При компиляции РКС-алгоритмов происходит их автоматическая конвертация в мнемокод, который при загрузке программы записывается в память контроллера. Если программист работает с РКС-языком, то он может не разбираться в тонкостях мнемокода, достаточно выполнить набор программы на экране дисплея и выполнить стандартные процедуры компиляции, запоминания и загрузки программы в контроллер. Подробно разбираться в программировании на мнемокоде нужно, если контроллер не предусматривает набор РКС-программ, а их ввод и редактирование осуществляются не на ПК, а с встроенного кнопочного программатора.

Как видно из разрешенных структур рис. 3.20 и 3.21, синтаксис языка позволяет набирать сложные разветвленные РКС-алгоритмы, т. е. осуществлять как бы «контроль общего провода». Для этой цели предусмотрены специальные команды **MPS**, **MPD** и **MPP** (рис. 3.22).

MPS – начало разветвления;

MPD – промежуточные разветвления;

MPP – конец разветвления.

Однако эти команды относятся уже к функциональным инструкциям.

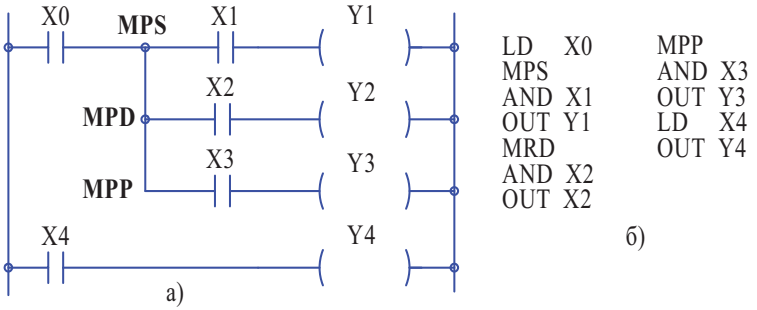


Рис. 3.22. Команды мнемокода программирования разветвленных цепей

Если набор программы и ее редактирование осуществляется только в графике, то о таких командах можно не вспоминать.

Допускается *каскадное* разветвление РКС-цепей.

Перейдем к изучению функциональных инструкций.

3.9.2. Функциональные команды

В технической документации на контроллер каждая функциональная инструкция сопровождается таблицей, отображающей ее синтаксис, применимость и свойства. В качестве примера ниже приведена таблица для команды пересылки MOV (табл. 3.8).

Таблица 3.8

Сжатое описание синтаксиса функциональной команды

API	Mnemonic	Operands	Function	Controllers
12	D MOV P	(S) (D)	Move	ES2/EX2 SS2 EX2 SX2

OP \ Type	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S					*	*	*	*	*	*	*	*	*	*	*	MOV, MOVP: 5 steps
D								*	*	*	*	*	*	*	*	DMOV, DMOVP: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

В ней содержится следующая информация:

Описание синтаксиса инструкции:

API – номер инструкции в списке, позволяющий быстро находить инструкцию при наборе PLC-программы (здесь 12);

D – факультативный признак 32-разрядного двойного слова (Double);

MOV – инструкция пересылки (Move). При отсутствии признака D осуществляется работа с 16-разрядной информацией;

P – факультативный признак одноразового выполнения команды (Pulse). При программировании признака команда выполняется только один раз, что сокращает время выполнения программы электроавтоматики;

(S) – Источник пересылаемой информации (Sours). Допустимые операнды указаны в таблице (здесь: K, H, KnX, KnY, KnM, KnS, T, C, D, E, F);

(D) – Приемник пересылаемой информации (Destination). Допустимые операнды указываются в таблице (здесь: KnY, KnM, KnS, T, C, D, E, F).

Аббревиатура инструкции (Function).

Применимость инструкции к типу контроллера (Controllers).

Число внутренних шагов, требуемых для выполнения инструкции (Programm Steps).

Возможность одноразового выполнения инструкции, доступность работы с 16 и 32-разрядными словами (Pulse, 16 – bit, 32 – bit).

Как правило, приводятся простейшие примеры применения инструкции, ограничения и особенности применения.

Данная информации используется при начальном изучении синтаксиса языка. При рабочем наборе инструкции в среде WPLSoft в окне установки операндов выдаются подсказки и контроль за правильностью программирования, исключающий синтаксические ошибки (но не алгоритмические!).

Набор и количество функциональных команд существенно отличаются в разных типах контроллеров. Изучить их все практически невозможно. Ниже достаточно большой обзор для рассматриваемого в главе контроллера типа DVP-SX2.

Инструкция MC / MCR (Master Control)

Данная инструкция в какой-то мере может быть отнесена к разновидности инструкций контроля общего провода (MPS, MPD и MPP) в разветвленных релейных цепях (см. рис. 3.22).

Инструкции MC/MCR программируются блоками с указанием его номера N0–N7 (рис. 3.23, *а*).

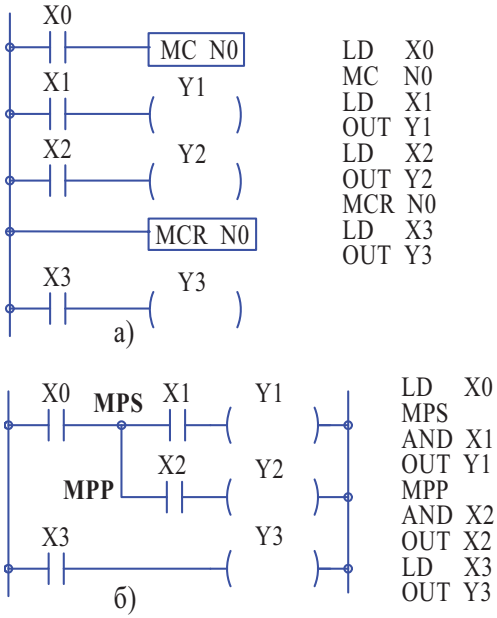


Рис. 3.23. Инструкция MC/MCR (а) и ее релейный аналог (б)

MC N0 – начало блока, в котором указывается условие его выполнения.

MCR N0 конец блока.

Если начальное условие выполняется, т. е. (MC N0) = 1, то выполняется и программа, написанная между инструкциями MC и MCR, если (MC N0) = 0, то программа между ними игнорируется.

Это обстоятельство можно трактовать разными способами:

– инструкция как бы контролирует общий провод последующей релейной цепи (см. рис. 3.23, *б*). Однако, если такой релейный аналог конвертировать в мнемокод, то система использует команды MPS, MPD, MPP;
– инструкция запускает некую локальную подпрограмму;
– инструкция выполняет функцию условного перехода по инверсному управляющему сигналу;
– рассматривать инструкцию как инструмент для нестандартных решений.

Даже самые простые приведенные здесь примеры показывают, что использование данных инструкций, как в релейном виде, так и в мнемокоде, сложнее и непонятнее для чтения, чем классический вариант (см. рис. 3.23, *б* и 3.24, *а*). Автор не сторонник их применения.

Допускается каскадное программирование с несколькими входимостями (рис. 3.24).

Инструкцию MC/MCR можно использовать каскадно, т. е. реализовать функции входимости (рис. 3.24).

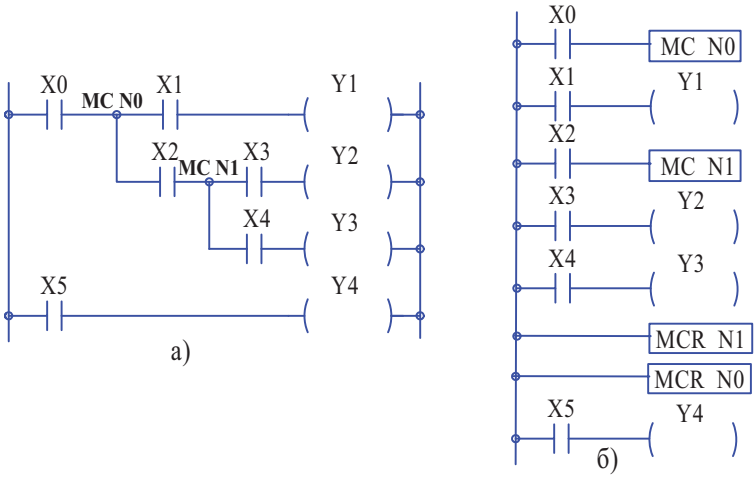


Рис. 3.24. Каскадное программирование инструкций MC/MCR:
а – релейный аналог; *б* – рабочий РКС-алгоритм

Формирователи тактов

Формирователи тактов используются для формирования импульсов длительностью в один вычислительный цикл (скан, такт) по переднему, заднему

или обоим фронтам входного сигнала. Тактирование сигналов позволяет просто программировать на контроллере различные специальные алгоритмы, например, Т-триггеры, сдвиговые регистры и др.

Как правило, в современных контроллерах предусматривается несколько вариантов таких формирователей, что часто бывает очень удобно.

Инструкции PLS / PLF (рис. 3.25)

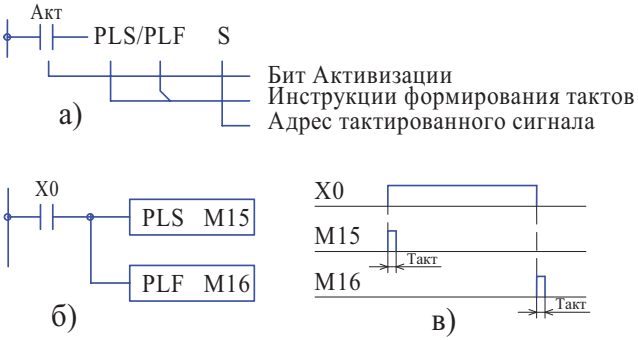


Рис. 3.25. Синтаксис (а), РКС-алгоритм (б) и диаграмма работы (в) инструкций формирования тактов PLS / PLF

Инструкция PLS формирует такт по переднему фронту входного сигнала, а инструкция PLF по заднему.

Формирование тактированных сигналов при помощи символа стрелки

На рис. 3.26 показаны следующие варианты формирования тактов специальными иконками:

а – такт по переднему фронту операнда X1;

б – такт по заднему фронту операнда X1;

в – логическое умножение с тактированным по переднему фронту операндом X1 и на тактированный операнд X2;

г – логическое умножение с тактированным по заднему фронту операндом X1 и на тактированный операнд X2;

д – логическое сложение с тактированным по переднему фронту операндом X2;

е – логическое сложение с тактированным по заднему фронту операндом X2.

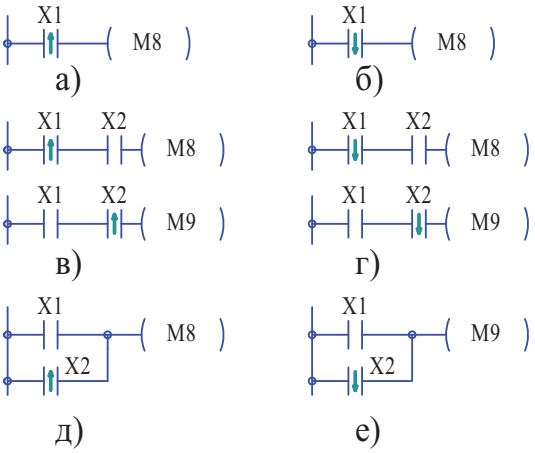


Рис. 3.26. Формирование тактированных сигналов с использованием символа «стрелка»

Формирование групповых тактированных сигналов

На рис. 3.27 показан пример формирование тактированных сигналов по результатам вычисления *групповых логических цепей*.

На ячейке M15 организована типовая RS – память M15 = (M12 + M15) ·/M13.

При выполнении условия (X1 + X2)·X3·/X4 = 1 команда «Стрелка-↑» формирует тактовый сигнал M12, включающий память.

При выполнении условия (/X1·/X2·X5 = 1 команда «Стрелка-↑» формирует тактовый сигнал M13 выключения памяти.

В программе мнемокода команда «Стрелка-↑» обозначается аббревиатурой NP, «Стрелка-↓» аббревиатурой PN.

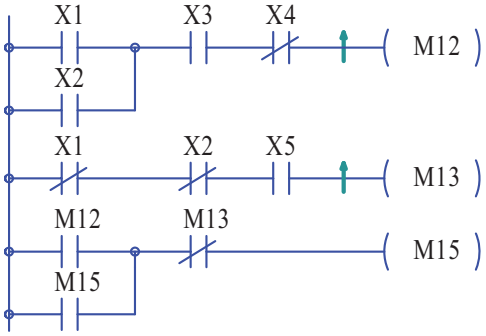


Рис. 3.27. Формирование групповых тактов

Аналогично можно сформировать групповые тактированные сигналы по отключению, т. е. перехода из нуля в единицу.

На рис. 3.28 приведен пример использования тактированных сигналов для формирования Т-триггера. Если тем или иным способом протактировать входной сигнал управления триггером, то триггер это уравнение неравнозначности (Исключающее ИЛИ) между тактированным сигналом управления и выходным сигналом, т. е. Y = (Такт ≠ Y).

Если в языке контроллера команда «Исключающее ИЛИ» отсутствует, то следует раскрыть это уравнение, т. е. Y = (Такт·/Y + / Такт·Y), что и сделано на рис. 3.28.

При необходимости процедура тактирования легко выполняется при помощи базовых битовых команд (рис. 3.29).

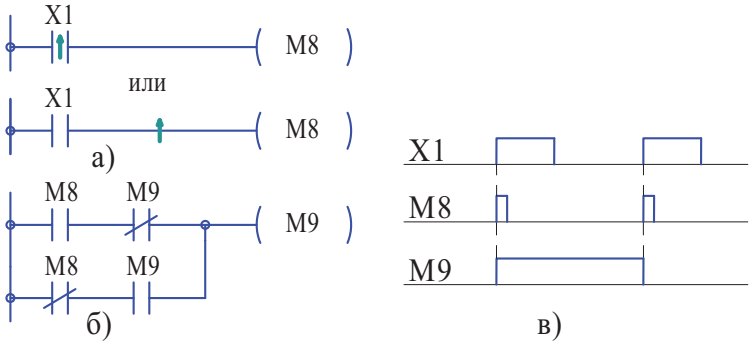


Рис. 3.28. Т-триггер на тактированном входном сигнале

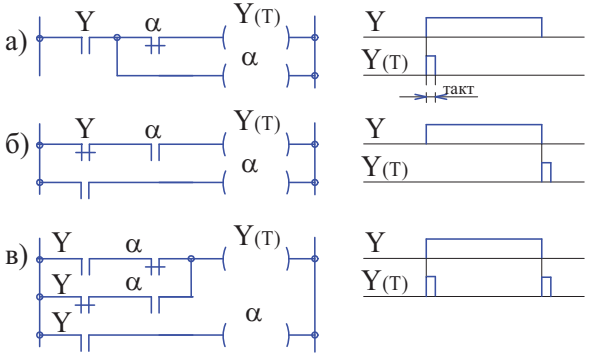


Рис. 3.29. Формирование тактированных сигналов при помощи базовых команд

Инструкции NOP (Нет операции)

Инструкция применяется для резервирования места при программировании на языке мнемокода (рис. 3.30), что полезно при отладке программы, например:

```
LD X0
NOP
OUT Y20
```

Рис. 3.30. Релейное представление инструкции NOP

В РКС-языке инструкция не отображается. В сложных релейных цепях, когда нет полной уверенности в правильности набора цепи, для резерва автор рекомендует: в последовательной цепи вставлять резервные ячейки «всегда = 1», а в параллельной «всегда = 0». Они не будут влиять на написанную программу, но в процессе отладки их оперативно можно будет заменить на нужный дополнительный рабочий операнд. Это важно, так как процедура вставки дополнительных операндов в разветвленных цепях достаточно неудобная. Например, в процедуре набора РКС-алгоритмов данного контроллера нет команды «Вставить столбец».

Адресация блоков

Выше в табл. 3.8 описания функциональной инструкции MOV, в графе допустимых операндов, были приведены следующие сокращения: KnX, KnY, KnM и KnS. Давайте разберемся, что это такое? Расшифровка аббревиатур показана на рис. 3.31.

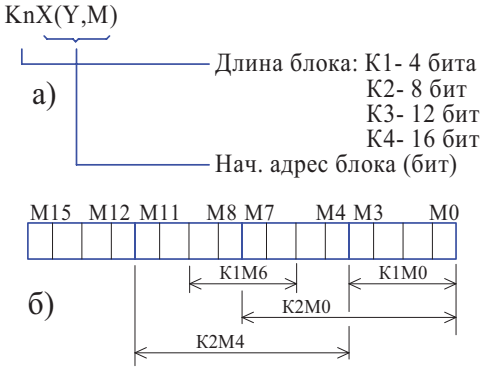


Рис. 3.31. Принцип адресации блоков

Видно, что аббревиатуры KnX, KnY, KnM и KnS позволяют адресовать тетрадами, байтами, полуторными байтами и словами: дискретные входы (X) и выходы (Y), ячейки памяти (M) и шаговые реле (S). Это во многих случаях облегчает программирование, работая с функциональными инструкциями.

Инструкции пересылки и обмена данными

Данный класс инструкций позволяет производить самые различные действия по трансформации блоков данных и включает в себя следующие инструкции:

MOV – простая пересылка данных от источника к адресуемому приемнику;
BMOV – пересылка блока данных;
FMOV – пересылка данных в несколько адресов;
SMOV – пересылка данных со смещением;
XCH – обмен данными;
CML – передача данных с инвертированием;
SWAP – внутренний обмен данными.

Инструкция MOV (рис. 3.32)

Данная инструкция осуществляет простейшую пересылку (перепись) данных от источника к адресованному приемнику.

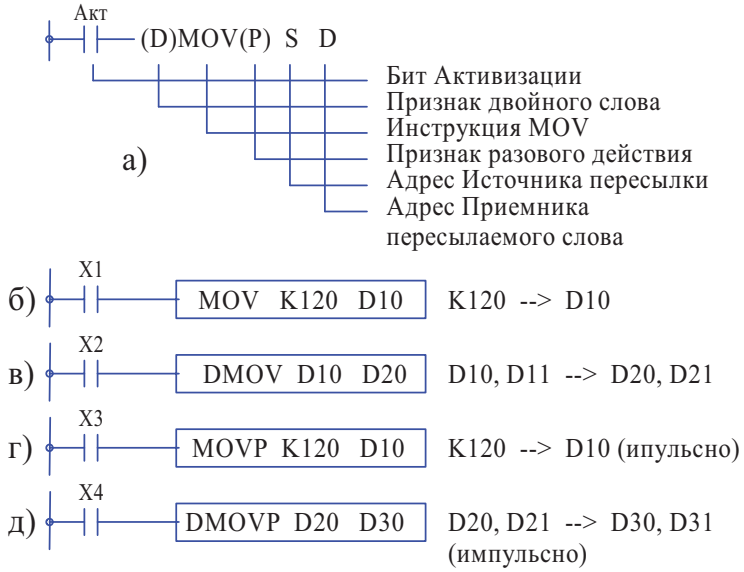


Рис. 3.32. Синтаксис (а) и примеры программ (б, в, г, д) инструкции MOV

Основные особенности инструкции:
– разрешенные источники K, H, KnX, KnY, KnM, KnS, T, C, D, E, F;
– разрешенные приемники KnY, KnM, KnS, T, C, D, E, F;
– интерпретация любых данных в словах в двоичном коде (BIN);
– данные источника при пересылке сохраняются;
– доступны варианты MOV, MOVP, DMOV и DMOVP.

В алгоритме *б* при условии X1 = 1 по команде MOV десятичное число K120 источника перепишется в 16-разрядное слово приемника D10. Операция переписи будет повторяться непрерывно до тех пор, пока не выключится бит активизации, т. е. при X1 = 0. Значение D10 = K120 после выключения X1 сохраняется.

В алгоритме *в* по команде DMOV будет непрерывно выполняться пересылка двойного слова источника D10, D11 в двойное слово приемника D20, D21.

В алгоритме *г* по команде MOVP выполнится однократная перепись десятичного числа K120 в 16-разрядное слово D10.

В алгоритме *д* по команде DMOVP выполнится однократная перепись содержимого двойного слова D20, D21 в двойное слово D30, D31.

При однократной переписи экономятся ресурсы контроллера. Отметим, что одноразовую перепись можно организовать и с командой MOV, если протактировать бит активизации.

Для упрощения начертания алгоритмов при черновой разработке алгоритмов в тетради можно не рисовать рамку, что сократит время, упростит правки и сэкономит место, например:

!--- – X1 -------- – MOV K120 D10.

При рабочем наборе программы на дисплее компьютера в среде WPLSoft рамка нарисуется автоматически.

Инструкция BMOV (рис. 3.33)

Инструкция дает возможность переслать блок из нескольких 16-разрядных слов в другое место адресного пространства по назначенному адресу. В инструкции указываются начальные адреса источника и приемника, а также число переносимых слов.

Особенности инструкции:
– разрешенные источники KnX, KnY, KnM, KnS, T, C, D;
– разрешенные приемники KnY, KnM, KnS, T, C, D;
– данные источника при пересылке сохраняются;
– доступна импульсная перепись BMOVP.

Необходимо следить за достаточностью места в памяти приемника и, если используется однотипная память, за отсутствием наложения адресного пространства источника и приемника, так как это не считается ошибкой (см. рис. 3.33, *в*, *г*).

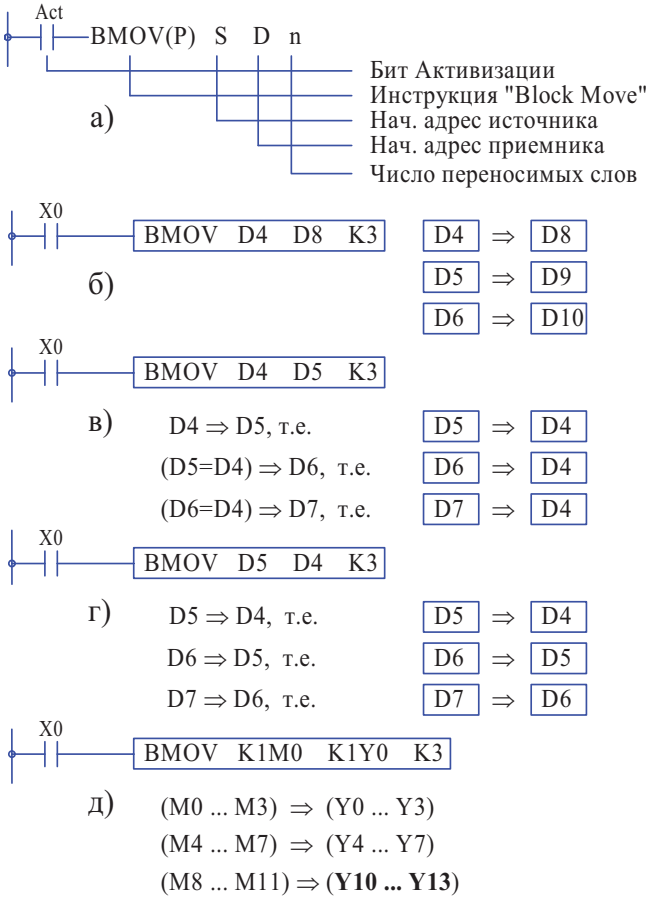


Рис. 3.33. Синтаксис и примеры программирования инструкции BMOV

Инструкция FMOV (рис. 3.34)

Инструкция предназначена для передачи данных **одного** источника в несколько приемников. При программировании задаются адрес источника данных, начальный адрес приемника и число слов приемника.

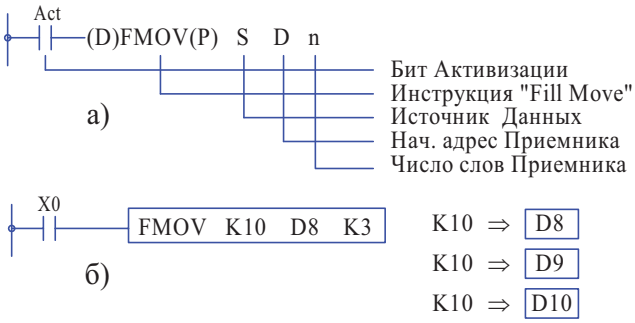


Рис. 3.34. Синтаксис и пример программирования инструкции FMOV

Основные особенности инструкции:
- разрешенные источники K, H, KnX, KnY, KnM, KnS, T, C, D, E, F;
- разрешенные приемники KnY, KnM, KnS, T, C, D;
- интерпретация любых данных в словах в двоичном коде (BIN);
- данные источника при пересылке сохраняются;
- доступны варианты с двойными словами и импульсной переписью.

Инструкция SMOV (рис. 3.35)

Инструкция предназначена для замены **десятичных** чисел приемника на адресуемые числа источника, путем из пересылки из одного слова в другое с указанием местоположения.

В программе указываются адрес источника S, номер начальной цифры источника m1, число сдвигаемых цифр m2, адрес приемника D и номер начальной цифры приемника n.

Предусмотрено два режима работы, переключаемые системным битом (параметром) M1168, определяющим условия переноса: прямой перенос двоичных чисел (M1168 = 1) или с промежуточным преобразованием исходных чисел в BCD-код и обратно (M1168 = 0).

Используется 16-разрядное слово, развиваемое на четыре тетрады, каждая из которых предназначена для хранения числа от 0 до 9, т. е. общий рабочий диапазон десятичных чисел составляет К0…К9999. Номера цифр как источника так и приемника шифруются **слева направо** начиная с младшей цифры «ц1…ц4».

Хранение чисел и их отображение на экране дисплея осуществляются в двоичном коде.

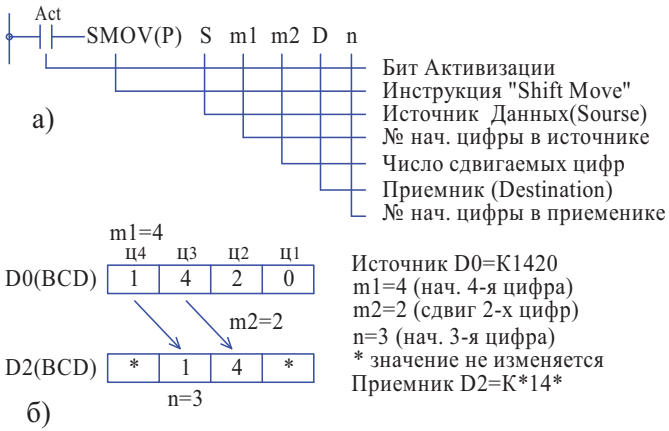


Рис. 3.35. Синтаксис и идеология работы инструкции SMOV

На рис. 3.36 приведена программа и идеология непосредственной (с точки зрения пользователя) замены чисел (M1168 = 0):

– число источника D0=K**14**20;
– число приемника D2=K1**56**2.

Например, необходимо заменить вторую и третью цифры приемника 56 на первую и вторую цифры источника 14, т. е. получить число D2 = K1142. Для этой цели в инструкции заданы следующие параметры:

– номер первой первая цифра источника m1 = 4, т. е. ц4 = **1**;
– число переносимых цифр m2 = 2, т. е. цифры **14**;
– номер начальной цифры приемника (первой заменяемой цифры) n = 3, т. е. цифра **5**;
– служебный бит M1168 = **0**. В этом случае при выполнении инструкции автоматически осуществляется промежуточное преобразование исходных двоичных чисел в BCD-код, перепись указанных цифр в нужное место приемника и обратное преобразование в BIN-код.

Таким образом, инструкция выполнит следующее преобразование:

(D2 = K 1 **5 6** 2) → (D2= K 1 **1 4** 2).

Поняв принцип работы, путем изменения параметров, можно производить подобные замены, особо не вдаваясь в суть происходящего. Например, если n = 2, то в приемнике будет число K15**14**.

Важно! При задании чисел необходимо использовать букву **K**1420. Если задать **H**1420, то процессор конвертирует это число в двоичное K5152 и дальше будет работать с другими цифрами 5152.

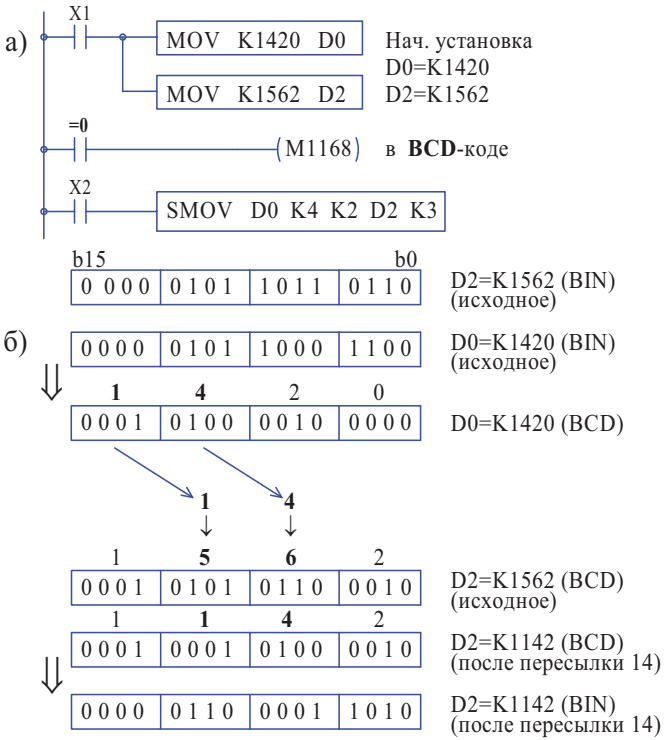


Рис. 3.36. РКС-алгоритм (а) и идеология (б) замены цифр при M1168 = 0

На рис. 3.37 приведена программа и идеология прямой пересылки и замены двоичных чисел (M1168 = 1). При таком способе программирования осуществляется пересылка и замена битовых значений адресуемых тетрад двоичных чисел.

Инструкция выполнит следующее преобразование:

$$(D2 = K\ 1\ 5\ 6\ 2\ ) \rightarrow (D2 = K\ 90).$$

Инструкция CML (рис. 3.38)

Инструкция предназначена для передачи данных из источника в приемник с предварительным побитным инвертированием числа источника, т. е. его «дополнения». Диапазон передаваемых данных определяется указанием адресного пространства приемника.

На рис. 3.38 четыре младших бита источника (M8 – M11) переписываются инверсно в адресное пространство приемника (M12 – M15). Содержимое приемника при этом не изменяется.

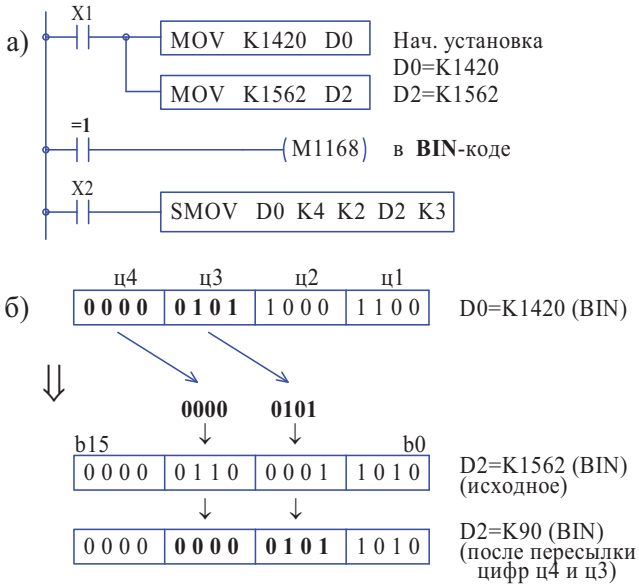


Рис. 3.37. РКС-алгоритм (а) и идеология (б) замены двоичных цифр при M1168 = 1

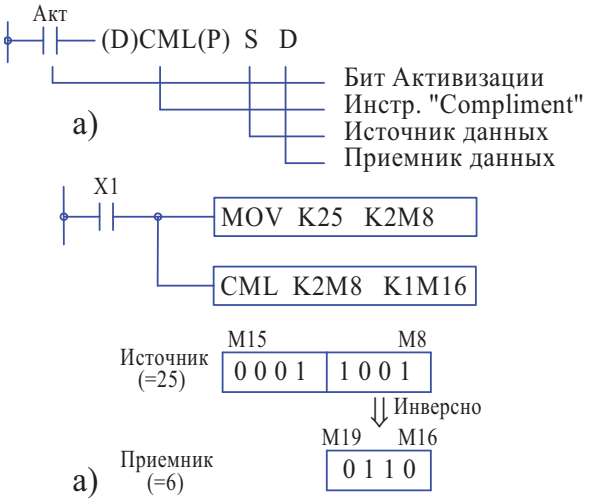


Рис. 3.38. Синтаксис, пример программирования и идеология инструкции CML

Инструкции обмена данными

К данному классу инструкций относятся:

Инструкция XCH, предназначенная для обмена данными между разными словами (рис. 3.39).

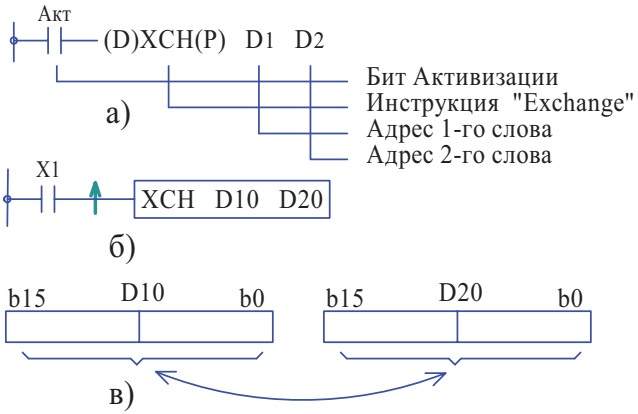


Рис. 3.39. Синтаксис, пример программирования и идеология инструкции XCH

Инструкция SWAP, предназначенная для внутреннего обмена байтами в 16 и 32-разрядных словах (рис. 3.40).

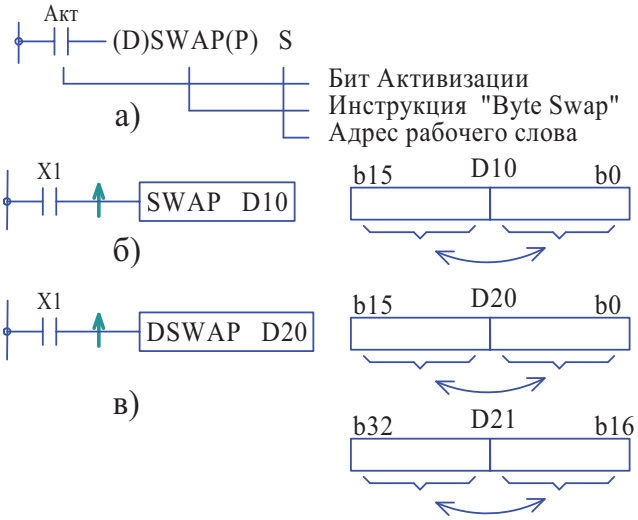


Рис. 3.40. Синтаксис и примеры программирования инструкции SWAP

Таймеры

В перечне временных инструкций контроллера предусмотрено несколько типов таймеров. Рассмотрим основные.

Таймер с выдержкой времени при включении (рис. 3.41)

Это основной базовый тип таймера, позволяющий на его основе спроектировать любой временной алгоритм.

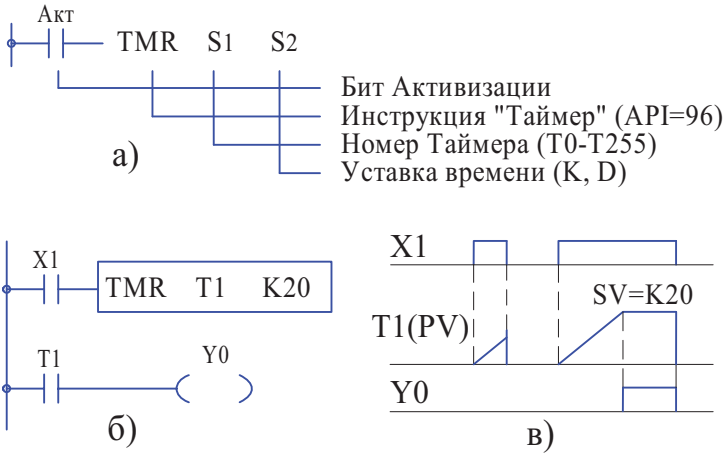


Рис. 3.41. Синтаксис, программа и диаграмма работы стандартного таймера

При программировании указывается номер таймера T0–T255 и величина уставки выдержки времени (SV – Set Value).

Выдержка времени может задаваться как напрямую в дискретах (Kxx), так и косвенно через D-слова. Соответствие величин дискрет и номеров таймеров приведено в табл. 3.9.

Выходом таймера является замыкающийся или размыкающийся контакт реле, указываемый в отдельной строке Ladder-диаграммы и имеющий адрес номера таймера.

Если за время активизации таймера его текущее состояние (PV – Previous Value) не достигнет величины уставки, то его значение сбрасывается в ноль и при новой активизации отсчет времени начнется с нуля.

Если за время активизации текущее значение PV достигнет величины уставки SV, то активизируется выходной бит и остается включенным до момента снятия сигнала активизации.

Таблица 3.9

Задание величин дискретности таймеров

Таймер	Параметр	Дискрета (мс)		
		100	10	1
T0-T64		*		
T64-T126	M1028 = 0	*		
	M1028 = 1		*	
T127				*
T128-T183		*		
T200-T239	M1038 = 0		*	
	M1038 = 1			*

Накаливающий Таймер (рис. 3.42)

Отличительной особенностью накапливающего (аккумулирующего) таймера является то, что при прерывании сигнала активизации его текущее состояние не сбрасывается, а запоминается. При повторной активизации продолжается отсчет величины выдержки времени от запомненной величины. Далее аналогично, при PV = SV активизируется выходной бит.

Аккумулирующие таймеры имеют фиксированные адреса:

(T250–T255) – дискрета 100 мс;

(T240–T245) – дискрета 10 мс;

(T246–T249) – дискрета 1 мс.

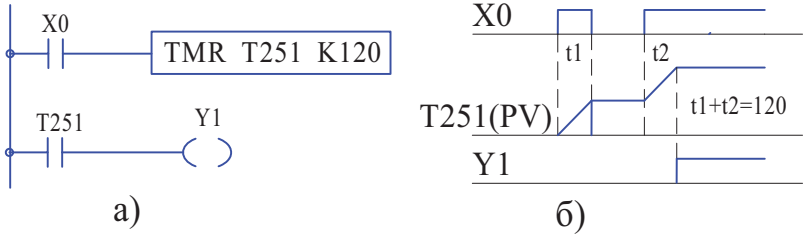


Рис. 3.42. Синтаксис, программа и диаграмма работы накапливающего таймера

***Обучаемый Таймер* TTMR** (рис. 3.43)

Обучаемый таймер является инструментом для предварительного задания и регулирования величины выдержки времени стандартного таймера TMR и программируется в паре с ним.

В данном примере, при активизации входа X1 обучаемого таймера с темпом, определяемым параметром K(0–2) в слово D10 по линейному закону записывается обезличенное числовое значение, наблюдаемое на экране дисплея.

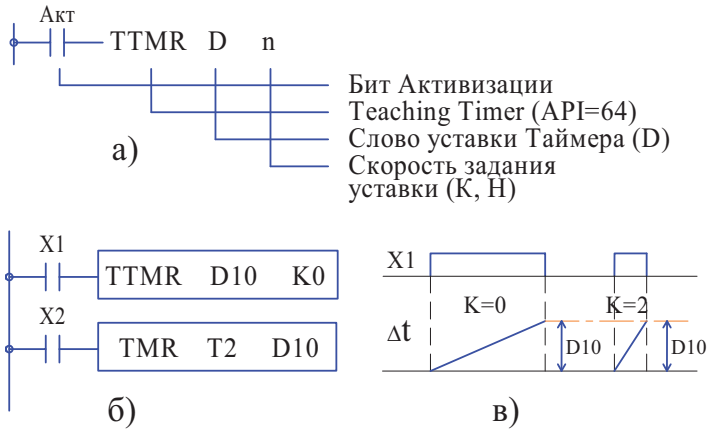


Рис. 3.43. Синтаксис, программа и диаграмма работы обучаемого таймера

При достижении нужного значения оператор отключает активизацию. В следующей строке программы заданная уставка переписывается в инструкцию основного таймера.

Специальный Таймер **STMR** (рис. 3.44)

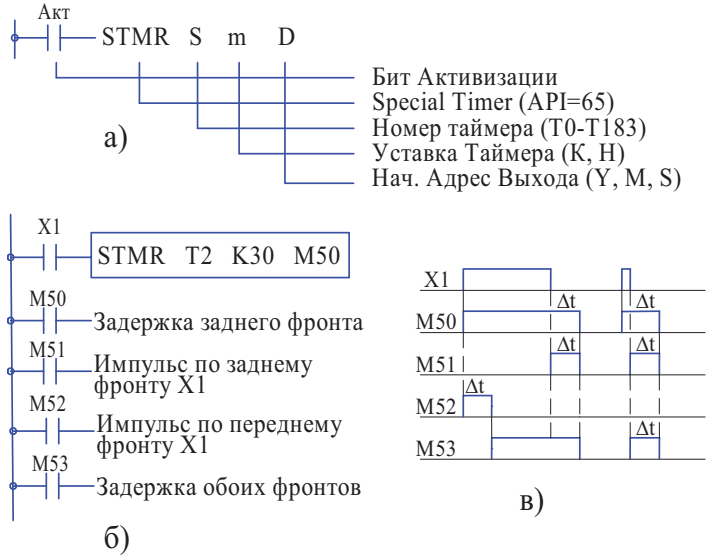


Рис. 3.44. Синтаксис, программа и диаграмма работы специального таймера

Специальный таймер позволяет одной строчкой программы реализовать следующие функции относительно входного бита активизации:

– сформировать сигнал с задержкой заднего фронта бита активизации;
– сформировать импульсы по переднему и заднему фронтам входного сигнала;
– сформировать сигнал с задержкой обоих фронтов бита активизации.

Принцип работы таймера ясен из поясняющего рисунка.

Доступные номера таймеров T0–T183, дискрета фиксирована и равна 100 мс.

В качестве выходов могут использоваться операнды Y, M и S.

***Счетчик* (D)CNT**

В контроллере предусмотрены следующие виды счетчиков:

(C0–C199) – стандартные нереверсивные 16-разрядные счетчики;
(C200–C231) – стандартные реверсивные 32-разрядные счетчики;
(C232–C254) – высокоскоростные счетчики.

Синтаксис и программа работы стандартного нереверсивного счетчика показаны на рис. 3.45. Счетчик 16-разрядный и кодируется аббревиатурой CNT.

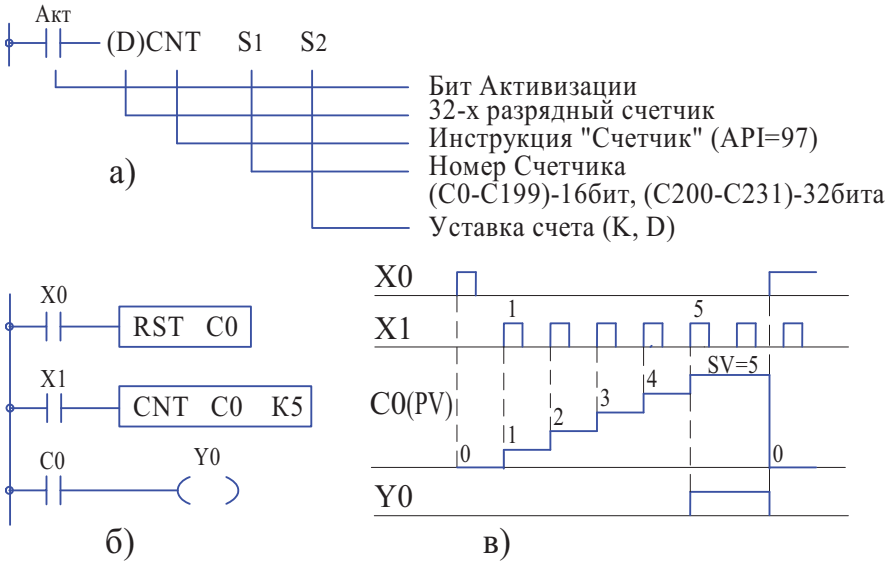


Рис. 3.45. Синтаксис, программа и диаграмма работы стандартного нереверсивного счетчика

Уставка счета может задаваться в прямом десятичном коде (К0–К32767) или косвенно через D-слова.

Счетчик начинает считать с нулевого значения.

При достижении текущего состояния (PV) значения уставки счета (SV) активизируется выходной бит счетчика, программируемый в отдельной строке РКС-алгоритма. Выход имеет адрес номера счетчика, предпосылаемый буквой С.

После достижения уставки на дальнейшие входные импульсы счетчик не реагирует.

Сброс текущего состояния осуществляется функциональной командой сброса RST.

Для реализации кольцевого счетчика необходимо синтезировать алгоритм с автоматическим самосбросом при достижении уставки.

Обращаем внимание, номер счетчика в инструкции, выходная катушка для сброса и выходной контакт обозначаются одинаково: Сх, где х – номер счетчика.

На рис. 3.46 приведена программа и диаграмма работы реверсивного стандартного счетчика. Счетчик 32-разрядный и кодируется аббревиатурой DCNT.

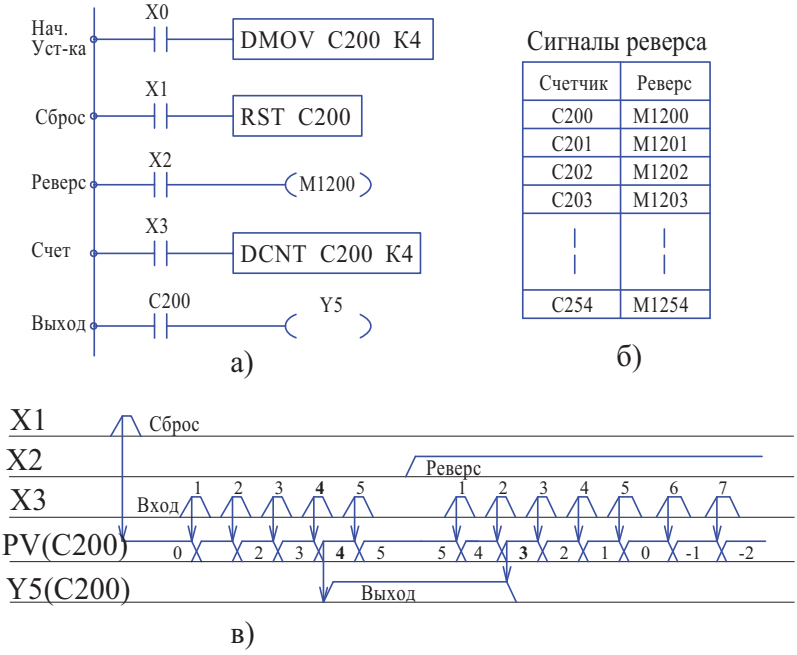


Рис. 3.46. Программа и диаграмма работы реверсивного счетчика

Направление счета изменяется активизацией служебных маркеров (параметров), жестко привязанных к номеру счетчика (см. таблицу рис. 3.46, *б*).

Командой DMOV, при необходимости, можно осуществить предустановку начального значения счетчика, а также переслать его значение в какое-либо слово.

Высокоскоростной счетчик (см. гл. 5) здесь не рассматривается. Скажем только, что выход высокоскоростного счетчика активизируется сразу после достижения уставки, не дожидаясь окончания вычислительного цикла. Для его реализации необходимо заказывать соответствующее исполнение контроллера.

***Инструкции* SET / RST** (рис. 3.47)

Инструкция **SET** (Установка) предназначена для установки с запоминанием в единичное состояние катушек однобитных операндов Y, M и S.

Инструкция **RST** (Сброс) предназначена для обнуления с запоминанием однобитных операндов Y, M, S, текущего состояния таймеров и счетчиков T, C, многоразрядных слов D, E, F.

Принцип работы ясен из приведенного ниже рисунка.

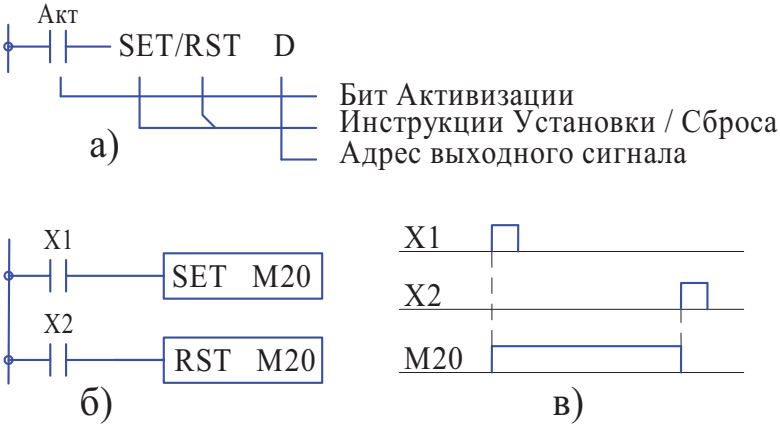


Рис. 3.47. Синтаксис, программа и диаграмма работы инструкций установки и сброса

***Инструкция* ZRST** (рис. 3.48)

Инструкция **ZRST** предназначена для группового сброса катушек однобитных операндов Y, M, S, текущего состояния таймеров и счетчиков T, C и многоразрядных регистров D.

В инструкции должно соблюдаться условие D1 < D2.

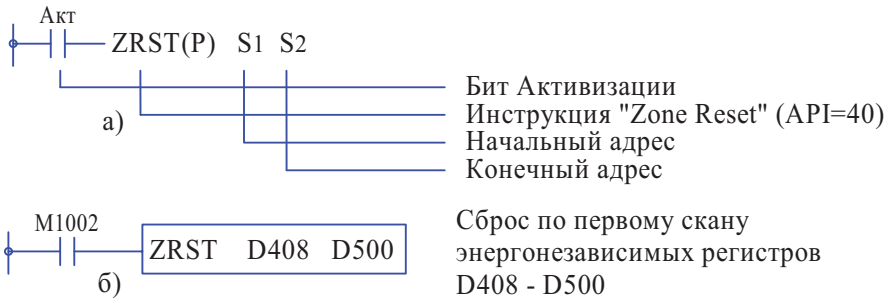


Рис. 3.48. Синтаксис и программа инструкции группового сброса

Инструкции сравнения

В контроллере предусмотрены следующие виды инструкций сравнения:

CMP – универсальная инструкция сравнения битовых операндов и слов на равенство, большее и меньшее значение;

ZCP – сравнение на принадлежность эталонного значения к указанной допустимой зоне;

LD (признак) – сравнение битовых операндов на равенство, неравенство, большее или меньшее значение.

***Инструкция* CMP** (рис. 3.49)

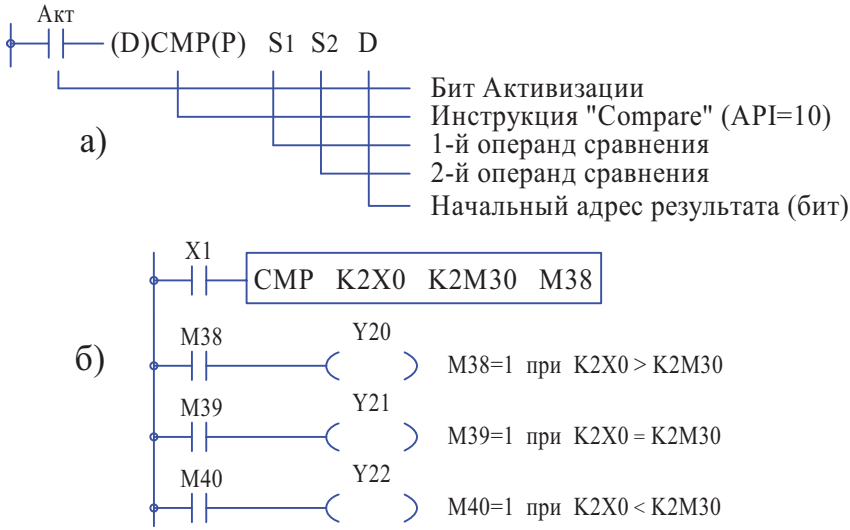


Рис. 3.49. Синтаксис и программирование инструкции сравнения CMP

При программировании указываются адреса сравниваемых операндов и начальный адрес бита ячеек хранения результата сравнения.

Доступно сравнение констант, битовых операндов, одинарных и двойных слов, содержимого таймеров и счетчиков.

Основные особенности использования инструкции:

– при сравнении используется двоичное представление операндов;
– результат сравнения запоминается в трех битовых ячейках:
  D (Начальный адрес в инструкции) = 1, если S1 > S2;
  (D + 1) = 1, если S1 = S2;
  (D = 2) = 1, если S1 < S2;
– предусмотрено одноразовое сравнение;
– сброс результата сравнения осуществляется инструкциями RST / ZRST.

***Инструкция* ZCP** (рис. 3.50)

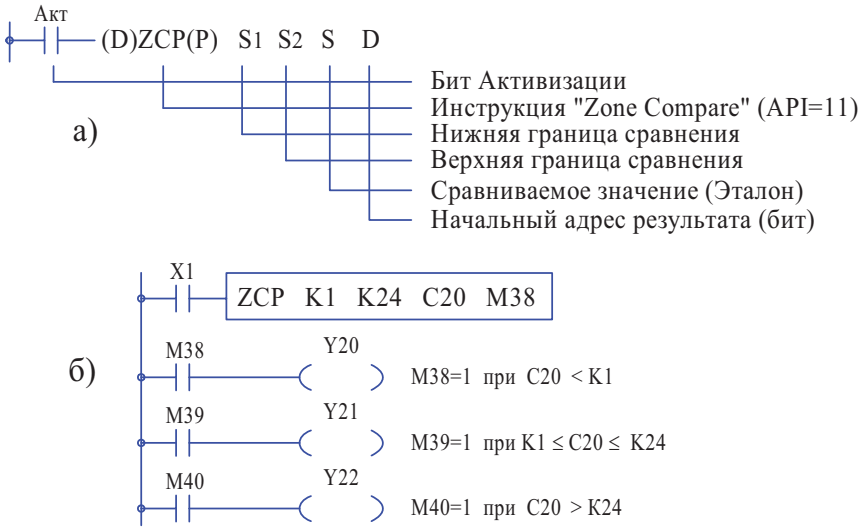


Рис. 3.50. Синтаксис и программирование инструкции ZCP

Инструкция осуществляет проверку принадлежности какой-либо эталонной величины заданной рабочей зоне.

При программировании указываются нижняя и верхняя допустимые границы зоны отклонения, эталонный адрес и начальный адрес бита ячеек хранения результата сравнения.

Доступно сравнение констант, битовых операндов, одинарных и двойных слов, содержимого таймеров и счетчиков.

Основные особенности использования инструкции:

– при сравнении используется двоичное представление операндов;
– результат сравнения запоминается в трех битовых ячейках (Y, M, S):
  D (Начальный адрес в инструкции) = 1, если S < S1, т. е. эталонное значение меньше нижней допустимой границы;
  (D + 1) = 1, если S1 ≤ S ≤ S2, т. е. эталонное значение находится в допустимой зоне;
  (D = 2) = 1, если S > S2, т. е. эталонное значение больше верхней допустимой границы;
– предусмотрено одноразовое сравнение;
– сброс результата сравнения осуществляется инструкциями RST / ZRST.

***Инструкция* LD=** (рис. 3.51)

Инструкция предназначена для сравнения битовых операндов на равенство, большее и меньшее значение.

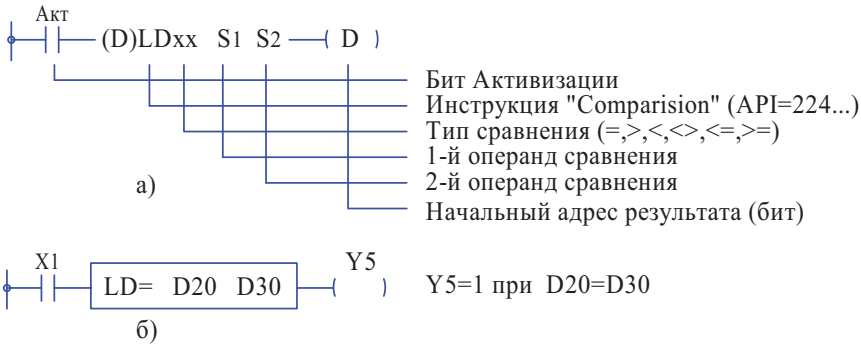


Рис. 3.51. Синтаксис и программирование инструкции сравнения LD=

Основные особенности программирования:

– допускается сравнение операндов с адресами K, H, KnY, KnM, KnS, T, C, D, E, F;
– сравнение выполняется в двоичном представлении чисел;
– разрешены следующие виды сравнения:
  инструкция LD=, то D = 1, если (S1 = S2);
  инструкция LD>, то D = 1, если (S1 > S2);
  инструкция LD<, то D = 1, если (S1 < S2);
  инструкция LD< >, то D = 1, если (S1 ≠ S2);
  инструкция LD<=, то D = 1, если (S1 ≤ S2);
  инструкция LD>=, то D = 1, если (S1 ≥ S2).

Если условие сравнения не выполняется, то выходной операнд равен нулю:
– допускается сравнение двойных слов;
– текущий результат сравнения записывается в отдельный битовый операнд (катушку).

Инструкции преобразования кодов

В контроллере предусмотрены следующие виды инструкций сравнения:

BCD – преобразование двоичного кода (BIN) в двоично-десятичный (BCD);

BIN – преобразование двоично-десятичного кода (BCD) в двоичный (BIN);

GRY – преобразование двоичного кода (BIN) в код Грея;

GBIN – преобразование кода Грея в двоичный (BIN).

***Инструкции* BIN / BCD (рис. 3.52)**

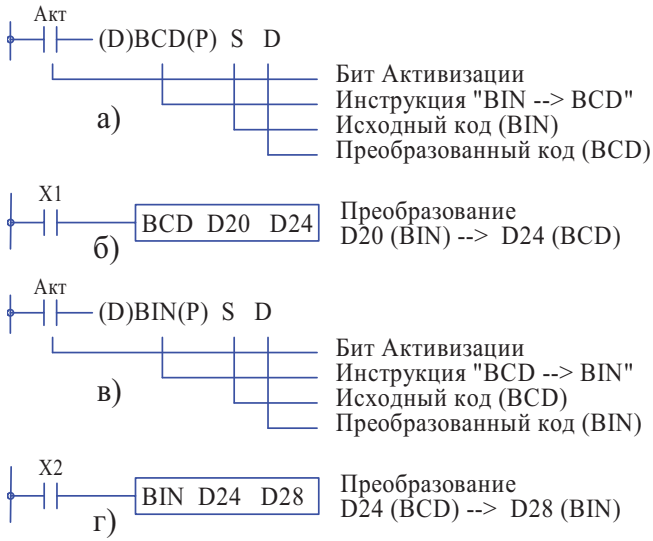


Рис. 3.52. Синтаксис и программы преобразования кодов BIN → BCD и BCD → BIN

Особенности программирования:
– внутренние преобразования в процессоре осуществляются в двоичном коде;
– максимальное число в BCD для шестнадцатеричного слова равно 9999, что соответствует двоичному числу 39321, по этой причине следует

следить за переполнением кода, иначе выдаются ошибки (M1067, M1068 и D1067);
- разрешено преобразование двойных слов (признак D);
- разрешен режим однократного преобразования (признак P);
- отображение цифр на экране дисплея в двоичном коде. Если делается преобразование 64(BIN) → 64(BCD), то на экране отобразится число K100, так как 64(BCD) – это 0110 0100, а его двоичное отображение 64 + 32 + 4 = 100.

Для облегчения работы следует иметь под руками таблицу преобразования кодов (см. гл. 1) или начертить «Шпаргалку» (рис. 3.53).

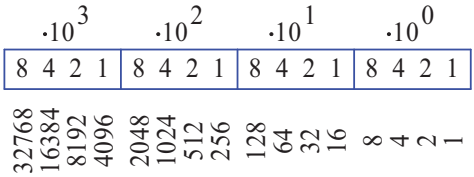


Рис. 3.53. Весовое соответствие BCD и BIN кодов

Инструкции **GREY** / GBIN (рис. 3.54)

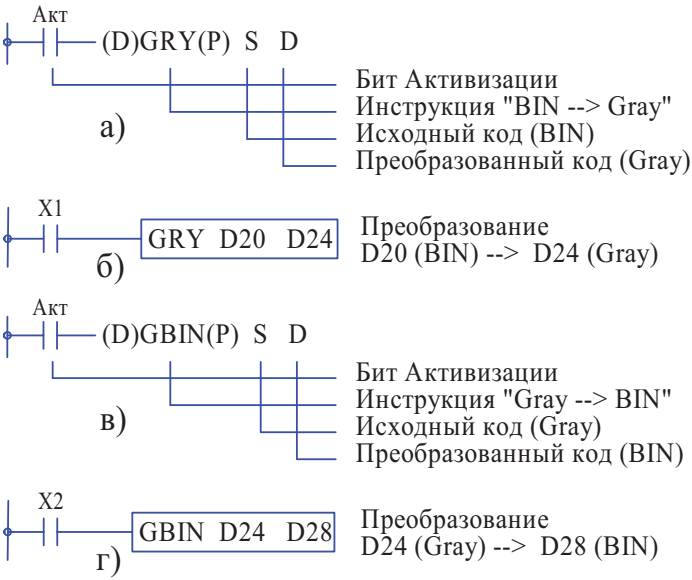


Рис. 3.54. Синтаксис и программы преобразования кодов BIN → Grey и Grey → BIN

Код Грея – это специализированный многоразрядный код, особенностью которого является то обстоятельство, что при переходе от любого числа к следующему, в его двоичном отображении изменяется только один разряд.

Принципы формирования такого кода изложены в главе 1.

В традиционных многоразрядных кодах при аналогичном переходе одновременно изменяется несколько бит числа, быстродействие переключения которых может быть различным, и что без принятия специальных мер приводит к ошибкам. В коде Грея это исключено. Основное его применение, это обработка, например, многопозиционных десятичных переключателей.

***Инструкции* INC / DEC** (рис. 3.55)

Инструкция INC – инкремент при подаче на ее вход **тактированного** сигнала активизации выполняет операцию увеличения на единицу содержимого адресуемого слова, т. е. D = D + 1.

Инструкция DEC – декремент при подаче на ее вход **тактированного** сигнала активизации выполняет операцию уменьшения на единицу содержимого адресуемого слова, т. е. D = D – 1.

Доступные операнды KnY, KnM, KnS, D, T, C, E, F.

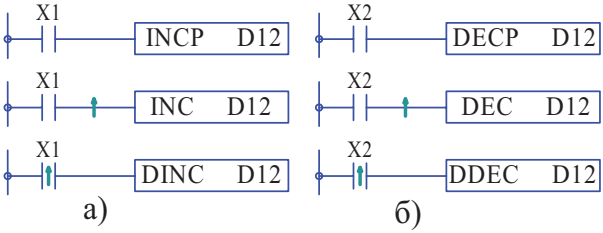


Рис. 3.55. Разрешенный синтаксис команд инкремента (а) и декремента (б)

Команды инкремента / декремента удобно использовать для визуального контроля на панелях оператора изменения различных параметров управления объектом автоматизации.

Если не использовать тактирование, то будет происходить непрерывное увеличение (уменьшение) содержимого адресованного операнда с частотой вычислительного цикла контроллера до полного заполнения памяти. Такой режим теоретически можно использовать для быстрой установки какой-либо уставки, с прерыванием при помощи команды сравнения. Но, ведь это можно сделать простой пересылкой.

Логические операции с битами слов (рис. 3.56)

Синтаксис языка предусматривает возможность выполнения побитовых операций логического умножения (AND), логического сложения (OR) и исключающего ИЛИ (XOR) в 16-разрядных и двойных словах.

Признаком шестнадцатеричного слова является начальная буква W, а двойного слова буква D.

Доступны любые регистровые операнды, разрешено импульсное выполнение операции (P).

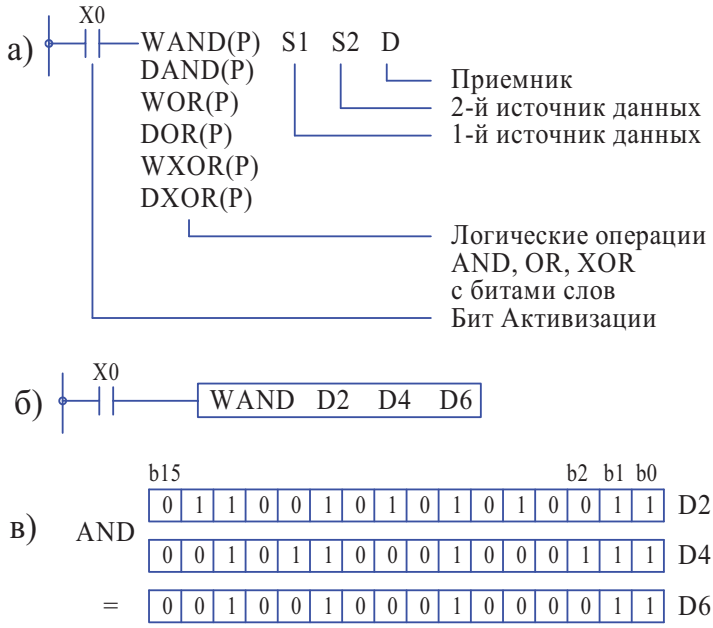


Рис. 3.56. Синтаксис (а), пример программы (б) и процедура выполнения (в) побитовых операций

Инструкция ALT (рис. 3.57)

Инструкция ALT – это инструкция, реализующая счетный Т-триггер при условии подачи на ее вход тактируемого входного сигнала. В качестве выхода используются операнды Y, M и S. Весьма полезная инструкция.

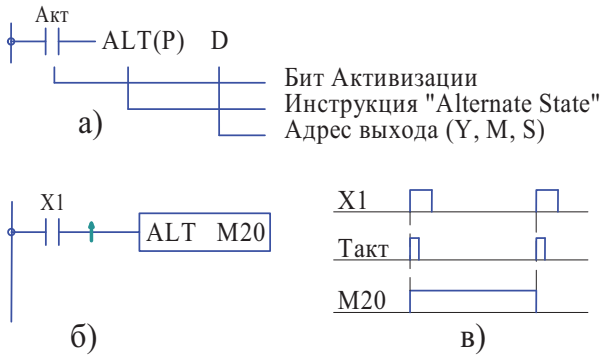


Рис. 3.57. Счетный триггер

Инструкция GPWM (рис. 3.58)

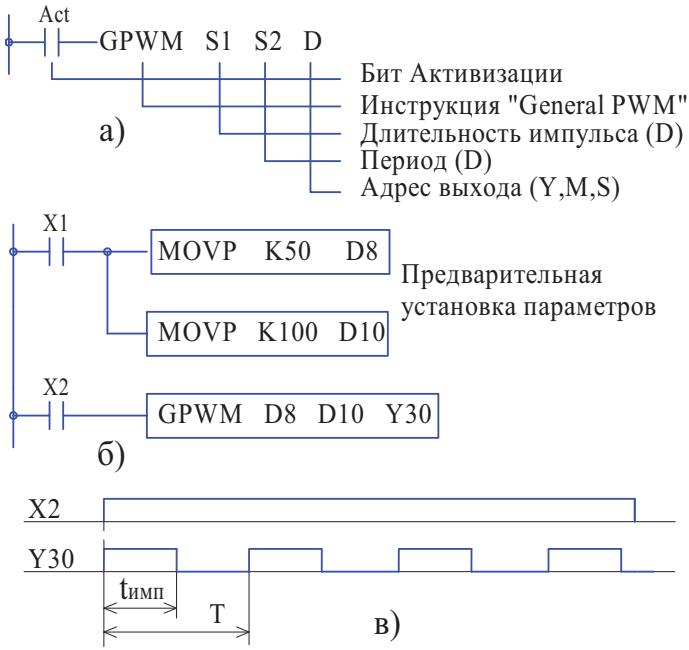


Рис. 3.58. Программируемый генератор импульсов

Инструкция GPWM – это инструкция, реализующая программируемый генератор импульсов (*широтно-импульсный модулятор* в документации).

Длительность и период формирования выходного импульса (Y, M и S) задаются предварительной записью в адресуемые D-слова.

Особенности применения:

– в параметрах S1 и S2 используются только D-слова;
– S1 (до 32767) < S2 (1–32767);
– (S1 + 1) и (S2 + 1) являются служебными словами, их нельзя занимать пользователю;
– параметры S1 и S2 можно изменять на ходу;
– дискрета задания времени равна 1 мс;
– в качестве выходов используются операнды Y, M и S.

Инструкция FOR / NEXT (рис. 3.59)

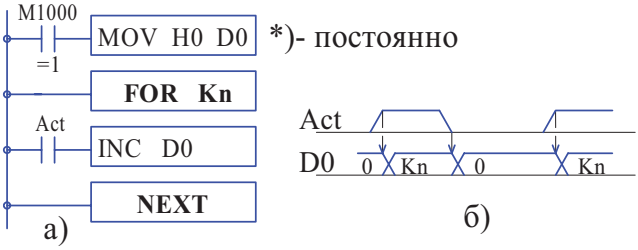


Рис. 3.59. Инструкция FOR / Next

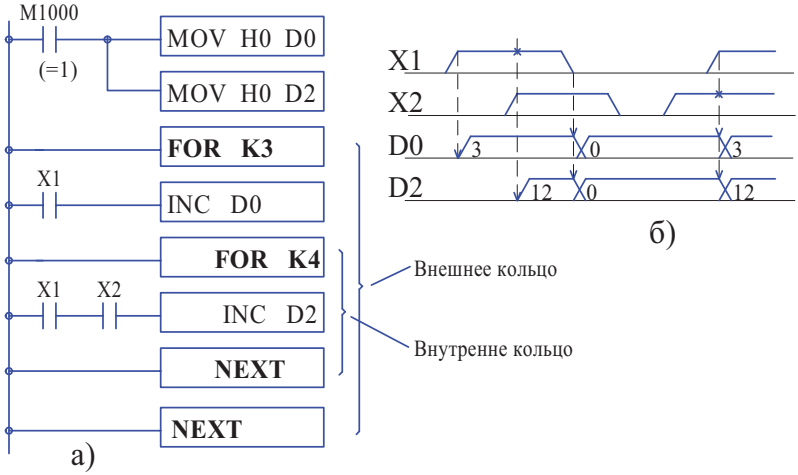


Рис. 3.60. Каскадное включение инструкций FOR / Next

Это парная цикловая инструкция, где FOR – начало цикла, а NEXT – его окончание. Используется для программирования повторения блоков программы. Часть программы, расположенная между командами FOR и NEXT повторяется «n» раз, после чего осуществляется переход к шагу программы, расположенному после команды NEXT.

Допускается входимость инструкций друг в друга (рис. 3.60).

В каких случаях эффективно ее применение решать пользователю. В приведенных примерах специально исключено тактирование перед инструкциями инкремента INC, чтобы быстро посмотреть число повторений, т. е. понять работу FOR – NEXT.

***Инструкция* DECO** (рис. 3.61)

Инструкция декодирования DECO выполняет следующую функцию:

- декодирует n – младших бит источника (в примере X20–X21);
- записывает единицу в бит (в примере b5) с номером, равным декодируемому десятичному числу (в примере 5), слова приемника длиной равной 2^n;
- остальные биты приемника обнуляются. В результате в приемнике двоичное число 32.

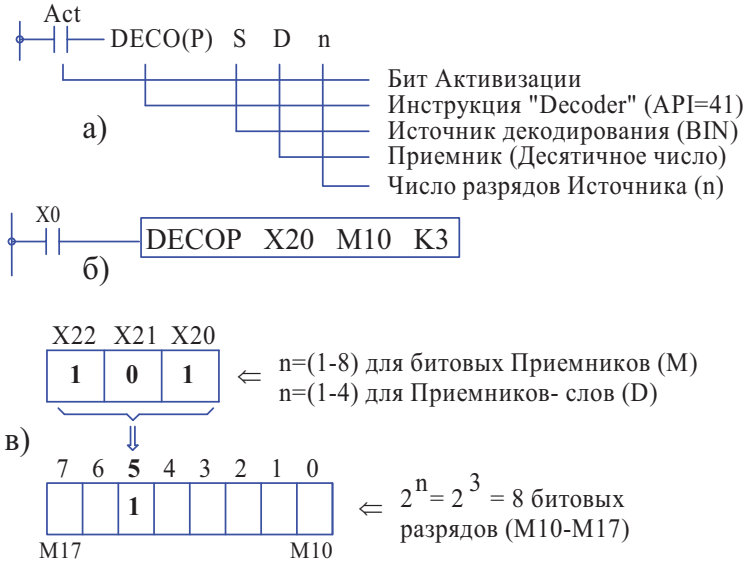


Рис. 3.61. Синтаксис, программа и процесс декодирования инструкции DECO

Инструкция ENCO (рис. 3.62)

Инструкция кодирования ENCO выполняет следующее действие:

– 2^n разрядов источника (в примере те же X20–X21) кодируются;
– результат (число 5) в двоичном коде переписывается n – младших бит приемника (в примере b0–b2);
– остальные биты приемника обнуляются.

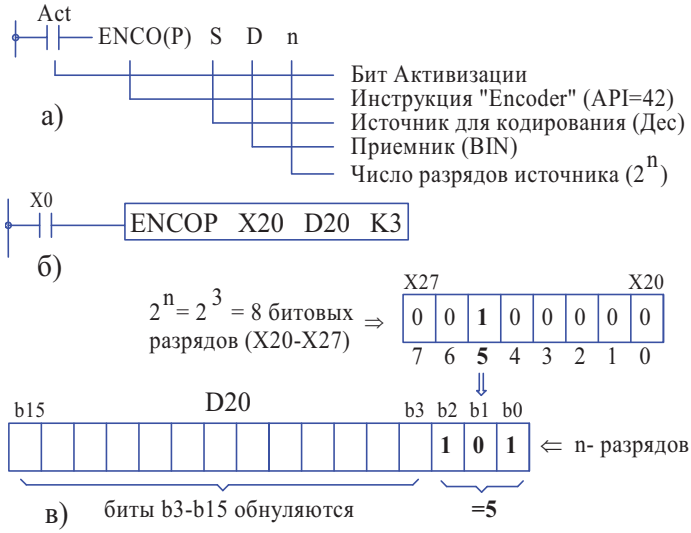


Рис. 3.62. Синтаксис, программа и процесс кодирования инструкции ENCO

Сдвиговые регистры

В номенклатуре контроллера имеются как кольцевые, так и выталкивающих сдвиговые регистры. Ниже рассматриваются наиболее применяемые типы.

Инструкции ROR / ROL

Инструкция ROR реализует классический кольцевой сдвиг адресуемого числа бит регистра вправо.

Инструкция ROL это аналогичный кольцевой регистр, но со сдвигом адресуемого числа бит влево.

В качестве регистра можно использовать следующие операнды: KnY, KnM, KnS, D, T, C, E, F. Перед сдвигом в регистр заносится исходное число.

Бит активизации должен быть тактированным.

Разрядность рабочего регистра может быть 16 или 32 бита.

На рис. 3.63 приведен 16-разрядный регистр сдвига вправо.

Оба регистра сопровождаются флагом M1022.

Для тех, кто сам умеет строить регистры:

В технической документации он позиционируется как флаг переноса (Carry Flag), однако в блок-схеме (рис. 3.62, в), взятой из сопроводительной документации, он обозначен как независимая ячейка. Из классики построения таких регистров известно, что для обеспечения сдвига информации в регистре без потери информации необходимы одна общая или раздельные по разрядам буферные ячейки. Действительно, если дать команду на перепись содержимого любой ячейки в следующую, то последнюю следует сначала сохранить в буфере, иначе она будет стерта. Для этой цели во внутреннем алгоритме переноса, нам не известном, и служит ячейка переноса. В блок-схеме фирмы Carry Flag это нечто другое. Ячейка M1022 лишь говорит, что выполняется перепись последнего разряда регистра.

Если забыть о ячейке *Carry Flag*, то это классический кольцевой 16-разрядный сдвиговый регистр. Любая записанная в него информация будет перемещаться по кольцу. Ячейка M1022 – это вспомогательная ячейка и в состав самого регистра не входит.

Важно! Бит активизации должен быть тактированным.

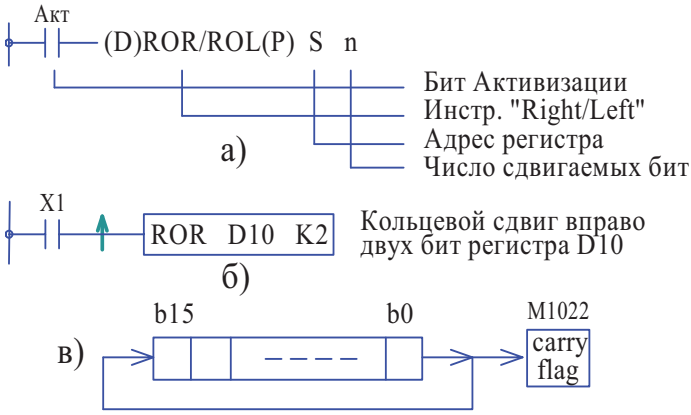


Рис. 3.63. Классический кольцевой регистр со сдвигом вправо

Инструкции RCR / RCL

Инструкции **RCR** и **RCL** это, соответственно, кольцевые регистры сдвига вправо и влево, но отличающиеся от регистров ROR и ROL тем, что в них флаг переноса M1022 используется как *дополнительный,* 17-й бит регистра.

Остальное аналогично.

В качестве регистра можно использовать следующие операнды: KnY, KnM, KnS, D, T, C, E, F. Перед сдвигом в регистр заносится исходное число.

Бит активизации должен быть тактированным.

Разрядность стандартного регистра **17** бит, двойного **33** бита.

На рис. 3.64 приведен пример регистра RCL сдвига влево.

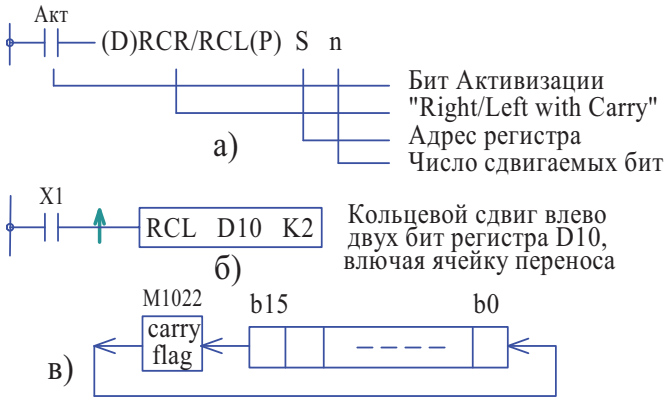


Рис. 3.64. Инструкция кольцевого регистра RCL сдвига влево

Инструкции SFTR / SFTL

Инструкции SFTR и SFTL это, соответственно, групповые битовые выталкивающие сдвиговые регистры вправо и влево. Они выполняют следующую операцию:

– информация регистра D (он же приемник) сдвигается на адресуемое число бит «n2», в зависимости от команды вправо или влево;
– первый по направлению сдвига блок из «n2-бит» выталкивается из регистра в служебные ячейки флага, т. е. теряется;
– в освободившееся при сдвиге место регистра переписывается блок из «n2-бит» источника S.

Разрешенные источники: X, Y, M и S;

Разрешенные приемники (т. е. регистры): Y, M и S. Длина приемника указывается параметром «n1». Обращаем внимание, D-слова нельзя использовать в качестве приемника.

Бит активизации должен быть тактированным.

На рис. 3.65 приведен пример организации регистра типа SFTR сдвига вправо. Процесс выполняется в следующей последовательности:

1 – битовый блок M0–M3 выталкивается в ячейки флага (Carry);

2 – битовый блок M4–M7 переписывается в освободившееся адресное пространство M0–M3;

3 – битовый блок M8–M11 переписывается в освободившееся адресное пространство M4–M7;

4 – битовый блок M12–M15 переписывается в освободившееся адресное пространство M8–M11;

5 – битовый блок источника X0–X3 переписывается в освободившееся адресное пространство приемника M12–M15.

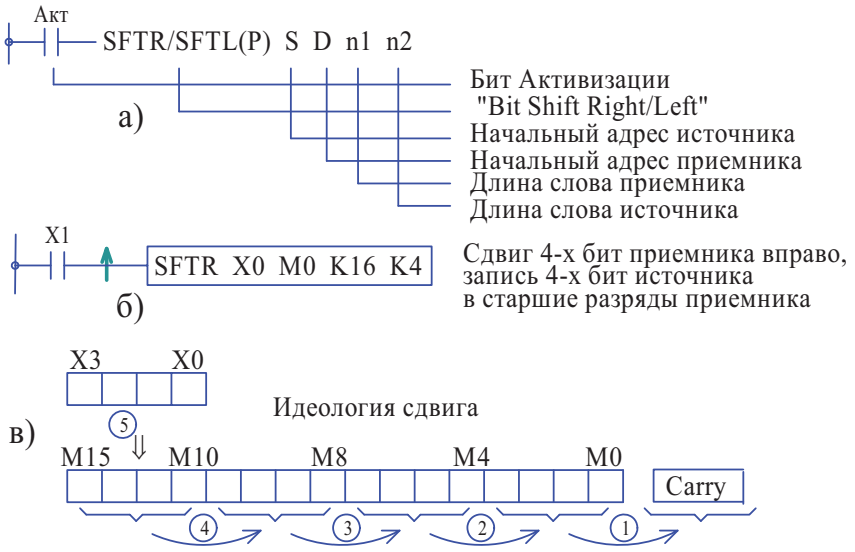


Рис. 3.65. Инструкция группового сдвига вправо SFTR

Работа инструкции SFTL сдвига влево аналогична.

Внимание! В случае необходимости для выполнения аналогичных операций с D-словами следует использовать инструкции WSFR и WSFL.

Инструкции организации стека

Синтаксис языка контроллера позволяет средствами электроавтоматики формировать стековую память, работающую по принципу FIFO (первый вошел – первый вышел), для чего используются инструкции **SFWR** и **SFRD**.

На рис. 3.66 и 3.67, соответственно, приведен синтаксис и принцип работы инструкций записи информации в стек (SFWR) и считывания из стека (SFRD). Инструкции работают в паре. Максимальная разрядность стека n = 512.

На рисунках приведен 4-разрядный стек, где:

D24 – источник данных;

(D1–D4) – рабочие регистры стека;

D0 – Указатель состояния стека;

m – числовое значение указателя;

M1022 – служебный параметр (маркер).

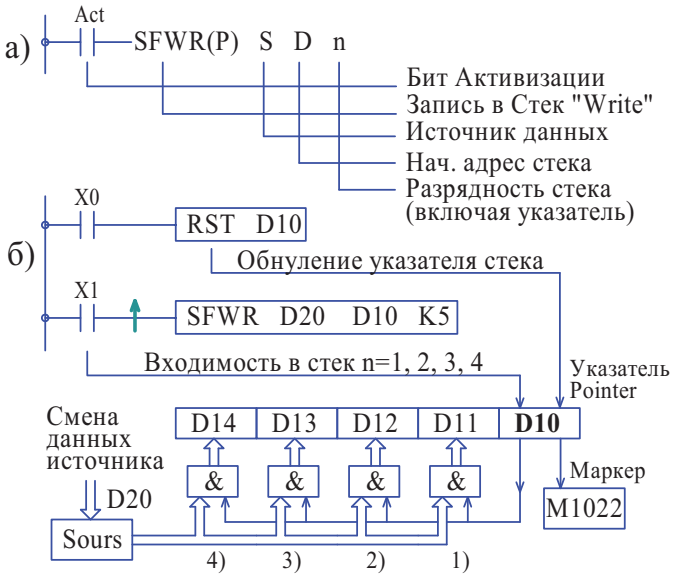


Рис. 3.66. Синтаксис и принцип работы инструкции SFWR

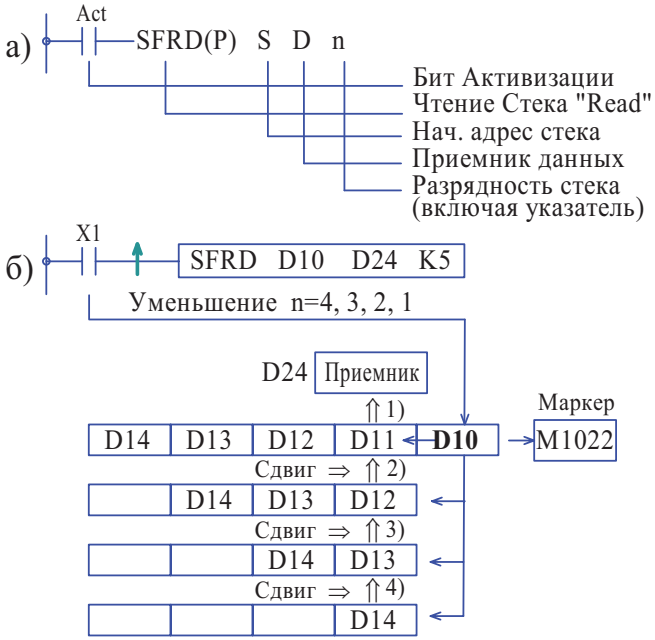


Рис. 3.67. Синтаксис и принцип работы инструкции SFRD

Последовательность работы стека при записи следующая:
- запись первого рабочего слова в источник данных, т. е. D24 = W1;
- обнуление указателя стека D0 (m = 0);
- Имп Акт → Увеличение Указателя на единицу (m = 1) → Запись 1-го слова источника в 1-й регистр стека, т. е. D1 = W1;
- запись второго рабочего слова в источник данных, т. е. D24 = W2;
- Имп Акт → Увеличение Указателя на единицу (m = 2) → Запись 2-го слова источника во 2-й регистр стека, т. е. D2 = W2;
- …
- запись четвертого рабочего слова в источник данных, т. е. D24 = W4;
- Имп Акт → Увеличение Указателя на единицу (m = 4) → Запись 4-го слова источника в 4-й регистр стека, т. е. D4 = W4.

При заполнении стека m = (n – 1) = 4 осуществляется прерывание работы инструкции и установка информационного маркера M1022 = 1.

Последовательность работы при чтении следующая:
- стек заполнен словами W1–W4 инструкцией SFWR;
- указатель установлен последней записью в стек;
- Имп Акт → Перепись 1-го рабочего слова из стека в источник, т. е. D24 = W1;
- уменьшение указателя на единицу m = (4 – 1) = 3;
- готовность к следующему чтению стека и т. д.

При m = 0 останов работы стека и установка маркера M1022 = 1.

Арифметические инструкции

В системе команд контроллера предусмотрены следующие арифметические инструкции:

ADD – арифметическое сложение;
SUB – арифметическое вычитание;
MUL – арифметическое умножение;
DIV – арифметическое деление;
ABS – взятие абсолютного значение числа;
SQR – вычисление квадратного корня;
SUM – подсчет числа единиц в слове;
MEAN – вычисление среднего значения;
RAND – генерация случайных чисел.

Предусмотрена также **специальная** арифметика для работы с числами с плавающей запятой.

Все арифметические операции выполняются в двоичном коде и алгебраически обрабатываются с учетом знака (последний бит слова операнда).

Разрешены действия как с шестнадцатеричными, так и с двойными словами.

Возможно задание однократности выполнения операции прямым указанием признака «P» в аббревиатуре инструкций.

При выполнении операций работают флаги:

M1020 – флаг нуля;

M1021 – флаг заимствования;

M1022 – флаг переноса.

Согласно техническому описанию доступна широкая номенклатура операндов, однако при выполнении арифметических действий во избежание ошибок (которые довольно трудно обнаружить) при сложных вычислениях автор рекомендует придерживаться следующих двух простых правил:

1. Во всех вычислениях использовать только D-слова;
2. Вне зависимости от типа инструкции и длины слов всегда выделять для одного операнда четыре одноразрядных слова, т. е. D0, D4, D8, D12 и т. д.

В этом случае не нужно будет думать, например, сколько слов занимает то или иное действие, куда записался остаток от деления и не пересекся ли его адрес с адресами других арифметических действий. Все будет нормально работать.

Инструкция ADD

Инструкция ADD (рис. 3.68) осуществляет арифметическое сложение двух вещественных чисел, адресуемых параметрами S1 и S2. Результат выполнения операции записывается в приемник D.

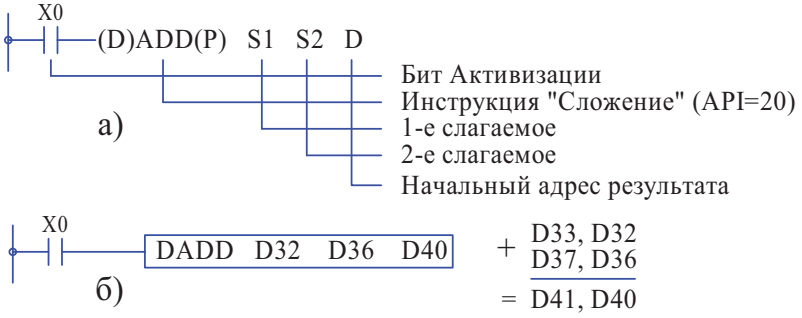


Рис. 3.68. Инструкция арифметического сложения

Инструкция SUB

Инструкция SUB (рис. 3.69) осуществляет операцию арифметического вычитания с двумя вещественными числами, адресуемыми параметрами

S1(уменьшаемое) и S2 (вычитаемое). Результат выполнения операции (остаток) записывается в приемник D.

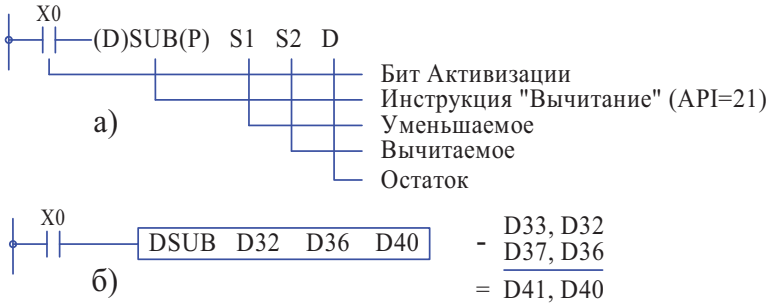


Рис. 3.69. Инструкция арифметического вычитания

Инструкция MUL

Инструкция MUL (рис. 3.70) осуществляет арифметическое умножение двух вещественных чисел, адресуемых параметрами S1 (умножаемое) и S2 (множитель). Результат выполнения операции записывается в приемник D.

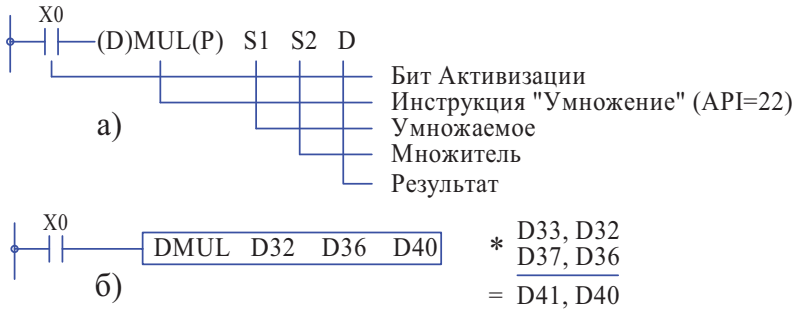


Рис. 3.70. Инструкция арифметического умножения

Инструкция DIV

Инструкция DIV (рис. 3.71) осуществляет арифметическое деление двух вещественных чисел, адресуемых параметрами S1 (делимое) и S2 (делитель). Результат выполнения операции записывается в приемник D (целое), D + 1 (остаток).

Инструкция ABS

Инструкция ABS (рис. 3.72) предназначена для выделения абсолютного значения адресуемого операнда D. Работает с одинарными и двойными словами, разрешено одноразовое преобразование.

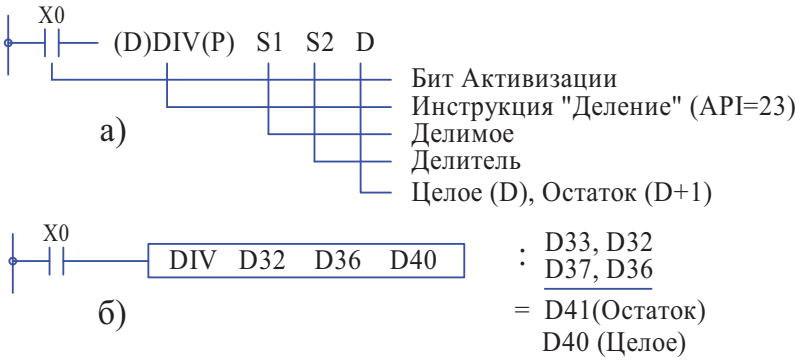


Рис. 3.71. Инструкция арифметического деления

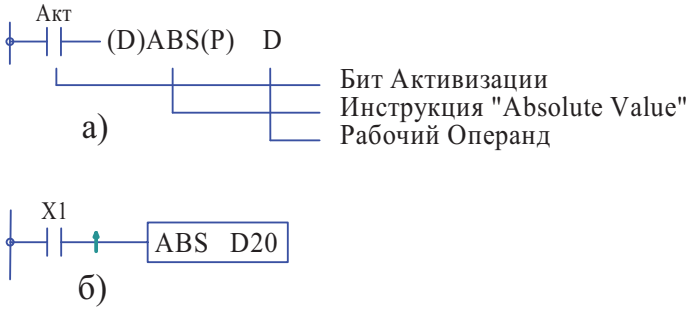


Рис. 3.72. Инструкция выделения абсолютного значения

Инструкция SQR

Инструкция SQR (рис. 3.73) предназначена для вычисления квадратного корня из адресованного подкоренного операнда S (прямые числа K, H или косвенное задание через слово D). Целочисленный результат записывается в слово приемника D. Работает с одиночными и двойными словами, разрешено прямое одноразовое вычисление

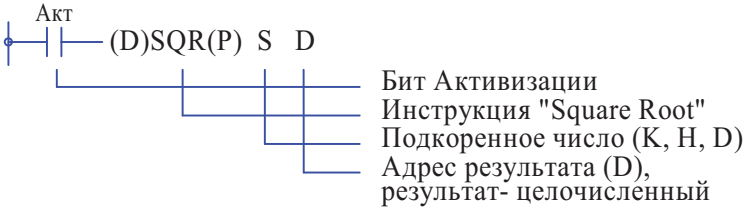


Рис. 3.73. Инструкция вычисления квадратного корня

Инструкция SUM

Инструкция SUM (рис. 3.74) предназначена для вычисления числа активных бит (единиц) в адресуемом слове S и записи результата по адресу выхода D.

Если нет активных битов, то устанавливается флаг нуля M1020.

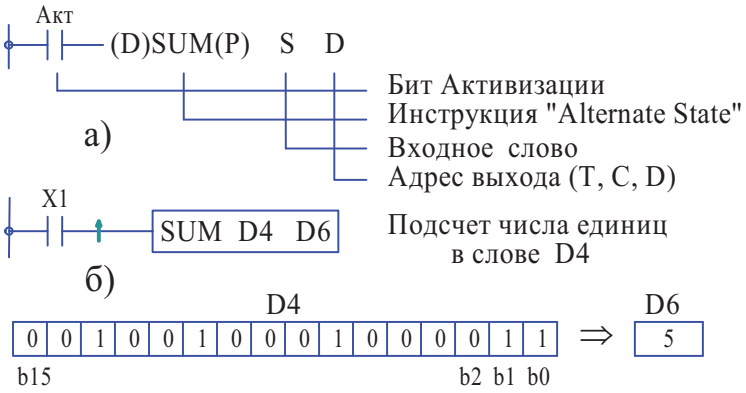


Рис. 3.74. Инструкция подсчета активных бит

Инструкция MEAN

Инструкция MEAN (рис. 3.75) рассчитывает среднее целое число из нескольких (n) адресуемых операндов S и записывает результат вычисления и слово приемника D.

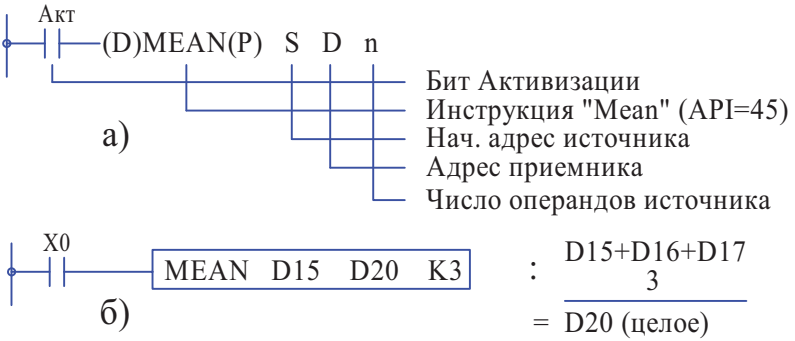


Рис. 3.75. Инструкция вычисления среднего значения

Инструкция RAND

При активизации инструкция RAND (рис. 3.76), в пределах заданных границ (S1–S2), генерирует случайное число и посылает его в приемник D.

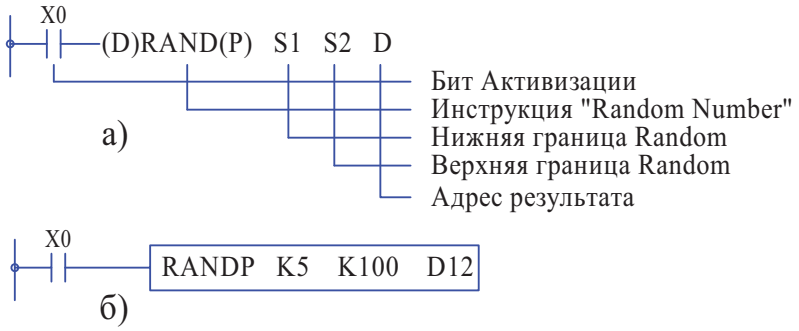


Рис. 3.76. Инструкция генерирования случайных чисел

Инструкции VRRD / VRSC

Инструкции VRRD и VRSC предназначены для съема и обработки информации встроенных в процессорный блок двух потенциометров.

Их можно использовать для разных целей, например, для внешнего задания уставок таймеров или для различных проверок, имитируя аналоговые датчики.

Инструкция VRRD (рис. 3.77) позволяет считать положение движка адресуемого потенциометра S (K0, K1), преобразовать его в двоичный код (0–255) и отправить по адресу приемника D.

Доступны следующие служебные операнды:

M1178 (M1179) – включенное состояние потенциометров VR0 (VR1);

D1178 (D1179) – численное значение двоичного кода VR0 (VR1).

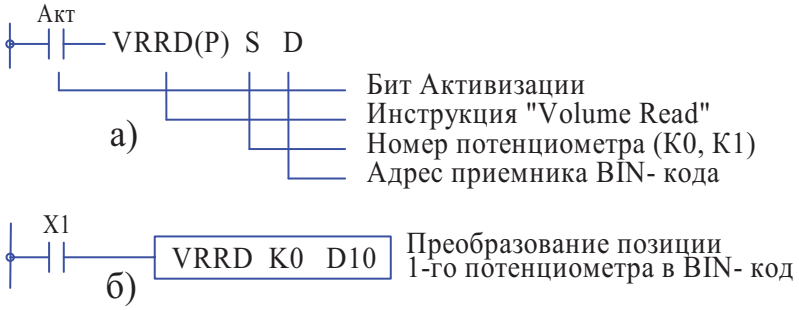


Рис. 3.77. Инструкция VRRD считывания значения потенциометра

Инструкция VRSC (рис. 3.78) позволяет в диапазоне 0–10 считать положение движка адресуемого потенциометра S (K0, K1) и отправить его по адресу приемника D с округлением до целого числа.

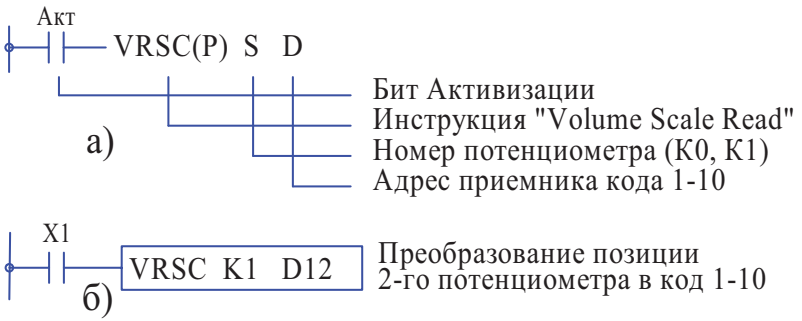


Рис. 3.78. Инструкция VRSC преобразования позиции движка потенциометра

Инструкция условного перехода CJ

При условии активного состояния (Акт = 1) бита активизации инструкции **CJ Px** (рис. 3.79) выполняется условный переход к строке программы, обозначенной меткой перехода S (P0–P255). Запрещается дублировать номера меток с инструкцией вызова подпрограмм Call.

При Акт = 0 инструкция CJ игнорируется и выполняется непрерывное выполнение программы по ее строчкам.

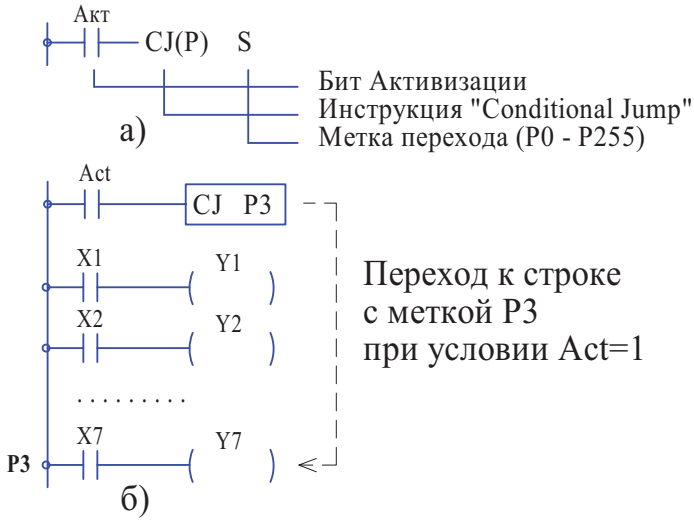


Рис. 3.79. Синтаксис (а) и выполнение (б) инструкции условного перехода

Инструкция вызова подпрограмм CALL

Инструкции CALL Px это инструкция вызова подпрограмм (рис. 3.80). Основные правила программирования:

- подпрограммы располагаются после окончания основной программы (FEND);
- подпрограмма начинается с метки P0–P255 и заканчивается инструкцией конца подпрограмм SRET;
- номера меток не должны совпадать с номерами меток инструкций условного перехода CJ;
- допускается входимость одной подпрограммы в другую;
- одну и ту же подпрограмму инструкцией CALL можно вызывать несколько раз.

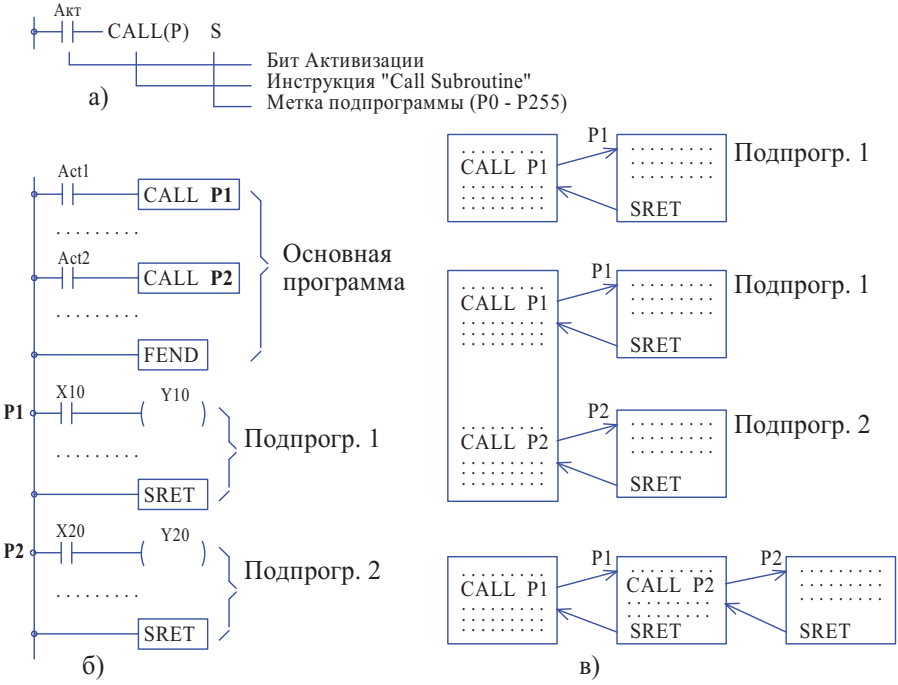


Рис. 3.80. Синтаксис (а), пример программы (б) и возможные структуры инструкции CALL (в)

Инструкции DSW

Инструкции DSW (рис. 3.81) предназначены для считывания числовой информации с 4-х разрядного DIP-переключателя, закодированного в двоично-десятичном коде. Возможна обработка 2-х групп переключателей. Принципиальная схема одной группы приведена на рис. 3.79, *б*.

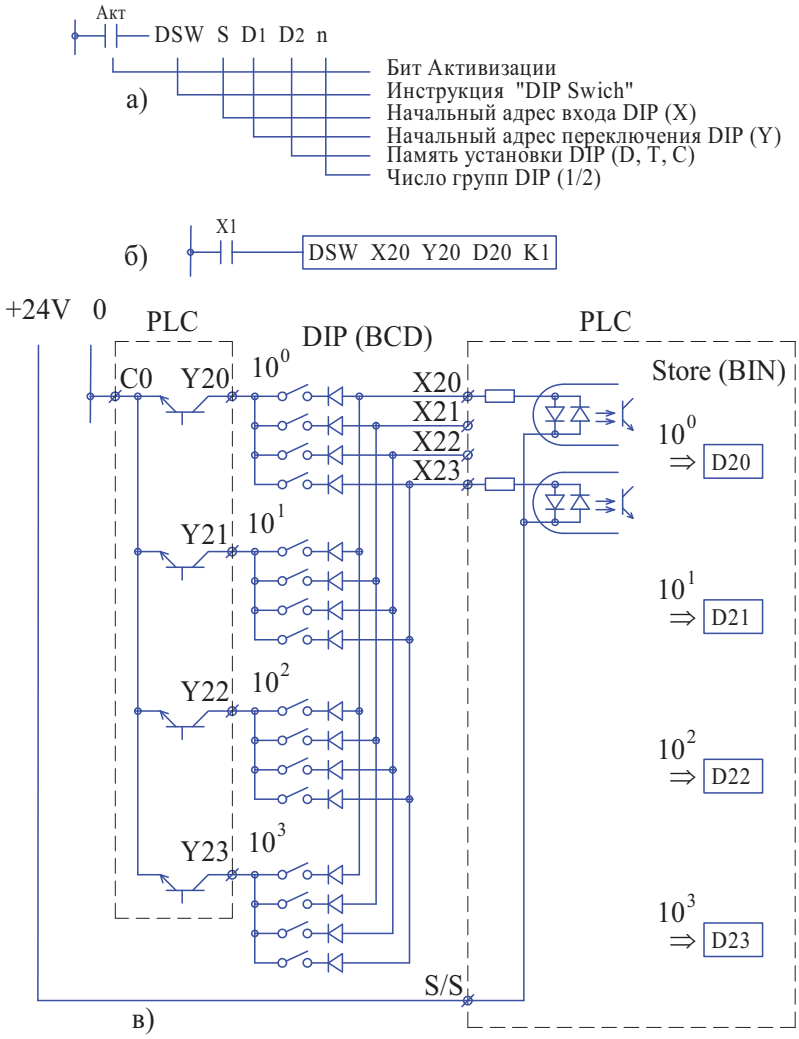


Рис. 3.81. Синтаксис (а), инструкция (б) и схема подключения DIP-переключателя (в)

Максимальное число группы 9999. Передача цифр осуществляется последовательно.

В инструкции DSW задается следующая информация:

– начальный адрес входа контроллера S (в примере X20), к которому подключаются выходы переключателя, соединенные через диоды по логической схеме ИЛИ;
– начальный адрес коммутатора разрядов цифр D1 (в примере Y20);
– начальный адрес памяти записи считываемой информации D2 (в примере D20);
– число групп переключателей n (в примере K1, одна группа).

Диаграмма работы инструкции приведена на рис. 3.82. Последовательность считывания следующая:

– включение бита активизации;
– последовательная коммутация выходов контроллера Y20–Y23.

Время включения каждого выхода 0,1 сек:

при Y20 = 1 в слово D20 считывается значения единиц 10^0;
при Y21 = 1 в слово D21 считывается значения десятков 10^1;
при Y22 = 1 в слово D22 считывается значения сотен 10^2;
при Y23 = 1 в слово D23 считывается значения тысяч 10^3.

По окончанию операции устанавливается маркер M1029 = 1.

Обращаем внимание, что числа передаются через одни и те же входы X20–X23.

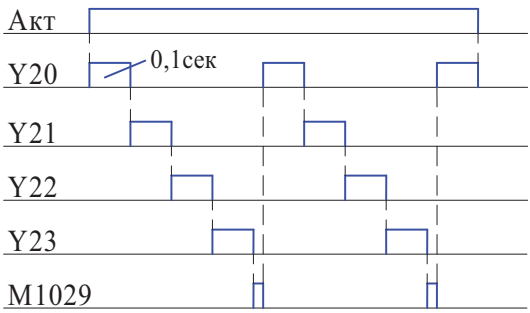


Рис. 3.82. Диаграмма работы инструкции DSW

Для второй группы, если она указана, используются входы X24-X27.

Инструкции SEGD

Инструкция SEGD (рис. 3.83) предназначена для управления цифровым 7-ми сегментным индикатором.

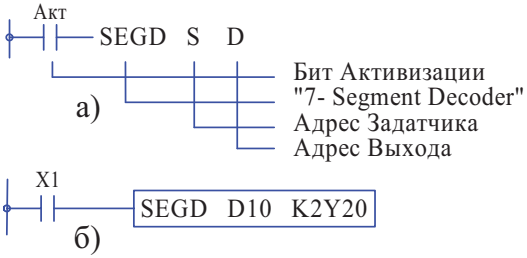


Рис. 3.83. Синтаксис (а) и программа (б) инструкции SEGD

В соответствии с табл. 3.10 инструкция дешифрирует 4 младших разряда источника S и записывает их в 8 младших разрядов приемника D.

Таблица 3.10

Принцип кодирования сегментов индикатора

Hex	Bit combi-nation	Composition of the 7-segment display	Status of each segment							Data displayed
			B0(a)	B1(b)	B2(c)	B3(d)	B4(e)	B5(f)	B6(g)	
0	0000		ON	ON	ON	ON	ON	ON	OFF	0
1	0001		OFF	ON	ON	OFF	OFF	OFF	OFF	1
2	0010		ON	ON	OFF	ON	ON	OFF	ON	2
3	0011		ON	ON	ON	ON	OFF	OFF	ON	3
4	0100		OFF	ON	ON	OFF	OFF	ON	ON	4
5	0101		ON	OFF	ON	ON	OFF	ON	ON	5
6	0110		ON	OFF	ON	ON	ON	ON	ON	6
7	0111		ON	ON	ON	OFF	OFF	ON	OFF	7
8	1000		ON	ON	ON	ON	ON	ON	ON	8
9	1001		ON	ON	ON	ON	OFF	ON	ON	9
A	1010		ON	ON	ON	OFF	ON	ON	ON	A
B	1011		OFF	OFF	ON	ON	ON	ON	ON	b
C	1100		ON	OFF	OFF	ON	ON	ON	OFF	C
D	1101		OFF	ON	ON	ON	ON	OFF	ON	d
E	1110		ON	OFF	OFF	ON	ON	ON	ON	E
F	1111		ON	OFF	OFF	OFF	ON	ON	ON	F

Выходы приемника осуществляют аппаратную коммутацию сегментов измерительного прибора, формируя необходимую цифру или букву информационного табло.

Инструкции работы с матрицами

Синтаксис языка включает достаточно большой набор инструкций для работы с матрицами. Структура матрицы приведена на рис. 3.84:

- для формирования структуры матрицы могут использоваться как D-слова, так и операнды KnX, KnY, KnM и KnS;
- ширина матрицы фиксирована и составляет 16 бит. Нумерация битов в матрице – сквозная;
- число строк матрицы для D-слов составляет n = (1–255), а для битовых слов n = (1–4);
- в инструкциях указывается начальный адрес исходных матриц и матрицы результата. Общее число слов в матрице определяется числом заданных строк (n);
- при программировании, в соответствии с синтаксисом команд, используются служебные маркеры (M) и указатель (Pr);
- адрес указателя, где это необходимо, задается пользователем в соответствующей инструкции;
- для всех инструкций допускается импульсная активизация (P).

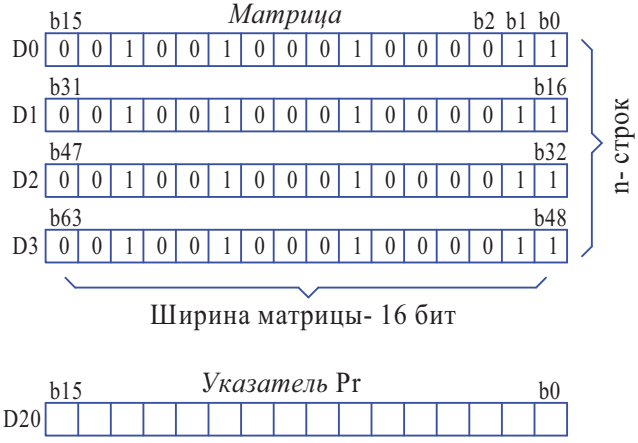


Рис. 3.84. Структура матрицы

На рис. 3.85 приведен синтаксис доступных инструкций работы с матрицами.

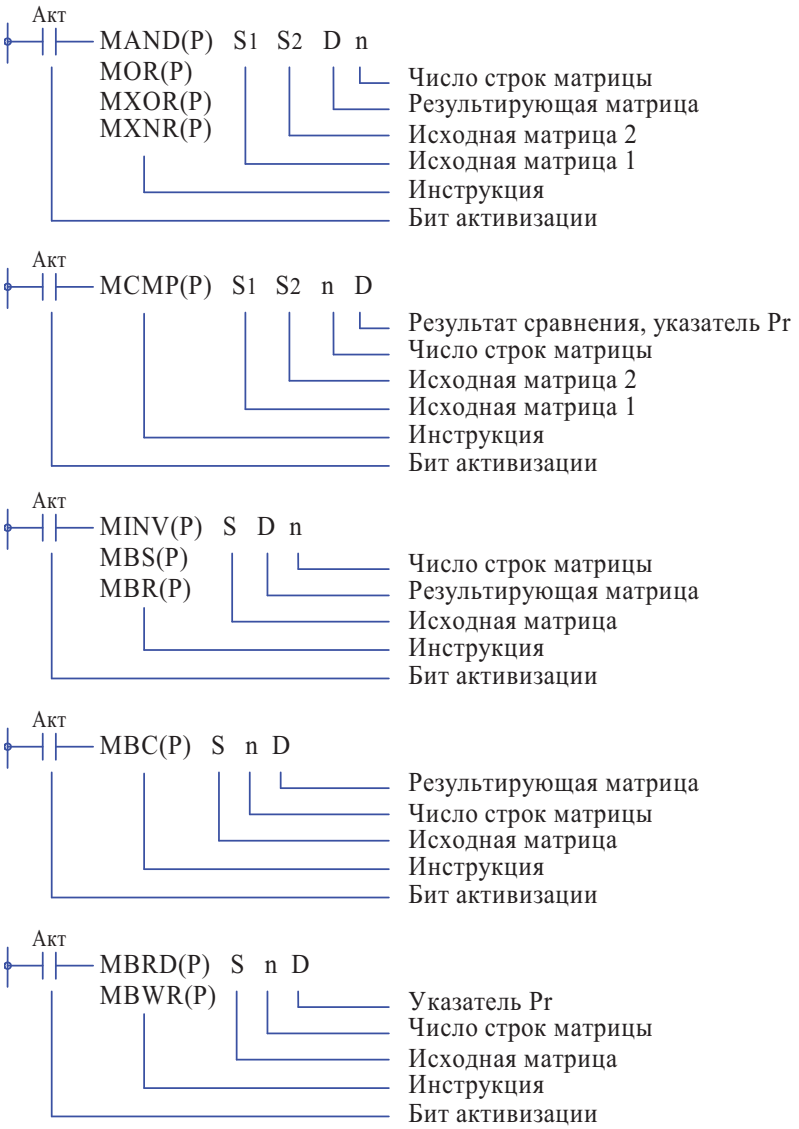


Рис. 3.85. Синтаксис инструкций работы с матрицами

Назначение инструкций:

MAND – побитовое логическое умножение содержимого 2-х исходных матриц S1 и S2 с записью результата в матрицу D;

MOR – побитовое логическое сложение содержимого 2-х исходных матриц S1 и S2 с записью результата в матрицу D;

MXOR – побитовое выполнение логической функции исключающего ИЛИ (неравнозначности) с содержимым 2-х исходных матриц S1 и S2 с записью результата в матрицу D;

MXNR – побитовое выполнение логической функции равнозначности (инверсной функции исключающего ИЛИ) с содержимым 2-х исходных матриц S1 и S2 с записью результата в матрицу D;

MCMP – инструкция сравнения содержимого 2-х исходных матриц S1 и S2. При подаче тактированного входного сигнала (или MCMPP) выполняется сравнение содержимого матриц и номер первого отличающегося бита записывается в слово указателя Pr (D). Следующее сравнение матриц начинается с бита на единицу большего, чем записан в указателе при предыдущем сравнении. Используются служебные маркеры:

M1088 = 1 сравнение на равенство;

= 0 сравнение на неравенство;

M1089 – сравнение закончено;

M1090 – индикация начала сравнения;

M1091 – индикация получения результата;

M1092 – ошибка указателя;

MINV – инверсия содержимого исходной матрицы S с записью результата в матрицу D. Содержимое исходной матрицы не изменяется;

MBS – проходной побитный сдвиг содержимого исходной матрицы S влево (M1097 = 0) или вправо (M1097 = 1) с записью результата в матрицу D. Содержимое исходной матрицы при этом не изменяется. Выталкиваемый из регистра последний бит матрицы записывается во флаг переноса M1095, а в освобождающийся первый бит матрицы записывается информация из флага заимствования M 1096.

MBR кольцевой побитный сдвиг содержимого исходной матрицы S влево (M1097 = 0) или вправо (M1097 = 1) с записью результата в матрицу D. Содержимое исходной матрицы при этом не изменяется. Выталкиваемый из регистра последний бит матрицы записывается в первый бит матрицы и во флаг переноса M1095;

MBC – подсчет числа единиц (M1098 = 1) или числа нулей (M1098 = 0) в исходной матрице S и запись результата в адресуемое слово D. Содержимое исходной матрицы не изменяется. Если результат подсчета равен нулю активизируется маркер M1099;

MBRD – при наличии тактированного входного сигнала выполняется чтение отдельного бита матрицы S, номер которого записан в указателе Pr(D) и запись его значения в маркер переноса M1095. При следующей активизации содержимое указателя увеличивается на единицу и считывается аналогично следующий бит матрицы. При работе с инструкцией используются следующие маркеры:

M1089 – завершение чтения;
M1092 – ошибка указателя;
M1093 – флаг увеличения указателя;
M1094 – флаг сброса указателя;
M1095 – флаг переноса;

MBWR – при наличии тактированного входного сигнала выполняется запись отдельного бита матрицы S, номер которого записан в указателе Pr(D) из флага заимствования M 1096, т. е. источника информации для записи. При следующей активизации содержимое указателя увеличивается на единицу и записывается аналогично следующий бит матрицы. При работе с инструкцией используются следующие маркеры:

M1089 – завершение чтения;
M1092 – ошибка указателя;
M1093 – флаг увеличения указателя;
M1094 – флаг сброса указателя;
M1096 – флаг заимствования.

Из рассмотренного материала ясно, что при программировании матриц следует предварительно продумать и разработать алгоритм последовательности выполнения операций в соответствии с поставленной общей задачей. Это будет не такая простая задача.

Инструкции работы с функциями реального времени RTS

Функции RTS – Real Time Clock предназначены для установки в контроллер реального времени и даты, с возможностью их последующей обработки в PLC-программе. К ним относятся следующие инструкции:

TWR – начальная запись времени и даты в память контроллера;
TRD – чтение времени и даты из памяти;
TCMP – сравнение текущего времени с заданным эталоном;
TZCP – сравнение на принадлежность текущего времени заданной зоне;
TADD – сложение временных уставок;
TSUB – вычитание временных уставок.

Инструкция TWR

Инструкция TWR (рис. 3.86) осуществляет предварительную запись в 7 слов приемника S (в примере D10–D16) следующей информации:

D10 – Год (0–99);
D11 – День недели (1–7);
D12 – Месяц (1–12);
D13 – День месяца (1–31);
D14 – Часы;
D15 – Минуты;
D16 – Секунды.

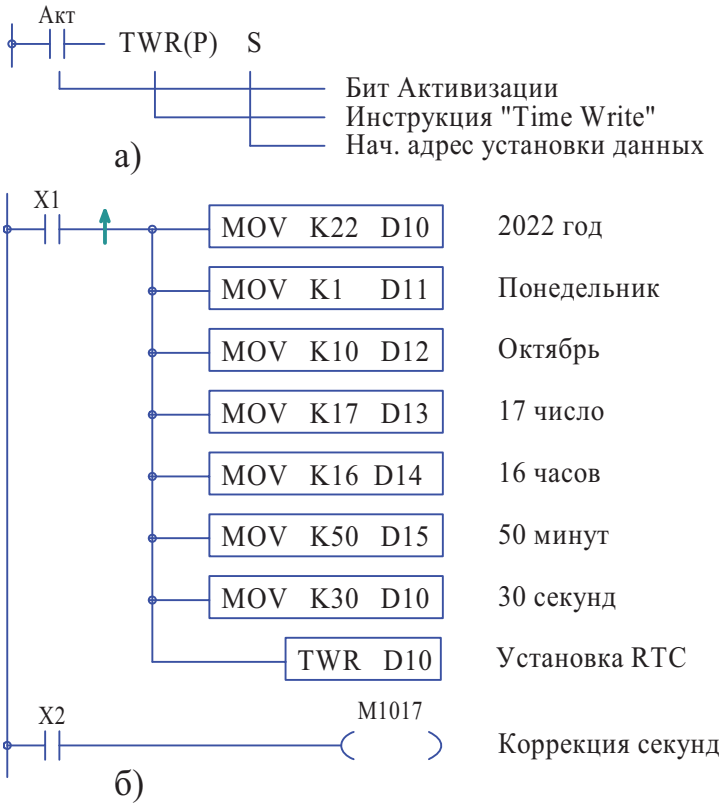


Рис. 3.86. Синтаксис (а) и программа (б) записи даты и времени

После выполнения инструкции произойдет автоматическая перепись введенных данных (D10–D16) в служебную энергонезависимую память D1319 (год)–D1313 (секунды).

Для оперативной коррекции секунд предусмотрен служебный маркер М 1017.

При его активизации в зоне (0–29) секунд происходит обнуление текущих секунд, а в зоне (30–59) секунд добавление минуты.

Рекомендуемая программа приведена на рис. 3.86, *б*. По ней можно читать время и дату в режиме реального времени. Формально, после записи ее можно стереть, так как текущие данные будут отслеживаться в служебных маркерах, но лучше этого не делать.

Служебный маркер М1016 позволяет изменять принцип установки года:

М1016 = 0 (две цифры года);

М1016 = 1 (Две цифры + 2000).

Ошибки программы индикатируют маркеры М1067 и М1068.

Инструкция TRD

Инструкция TRD (рис. 3.87) осуществляет чтение текущей информации о дате и времени в служебных маркерах и в режиме реального времени передает их в область памяти указанной в инструкции записи TWR, т. е. в D10–D16.

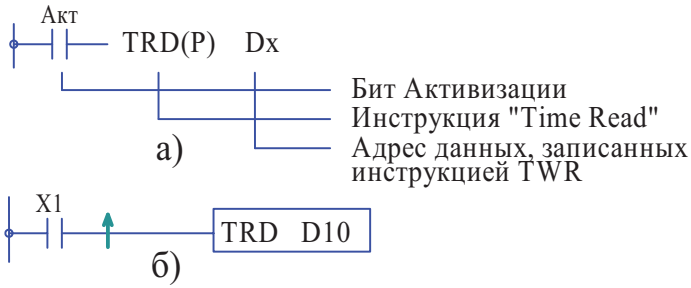


Рис. 3.87. Синтаксис (а) и программа (б) инструкции TRD

Инструкция TCMP

Инструкция TCMP (рис. 3.88) осуществляет сравнение на равенство, большее или меньшее значение текущего времени S (в примере D14) с заданным эталоном времени (S1–S3). Результат сравнения записывается в 3 бита приемника D (в примере М15–М17 и М18–М20).

Комментарий к программе рис. 3.88, *б*.

D10 – начальный адрес всех параметров D10–D16;

D14 – начальный адрес параметров «Часы, Минуты, Секунды»;

К16 / К54 / К30 – эталонное время 16 : 54 : 30;

М15 – начальный адрес первого сравнения;

K17 / K7 / K30 – эталонное время 17 : 07 : 30;
M18 – начальный адрес второго сравнения.

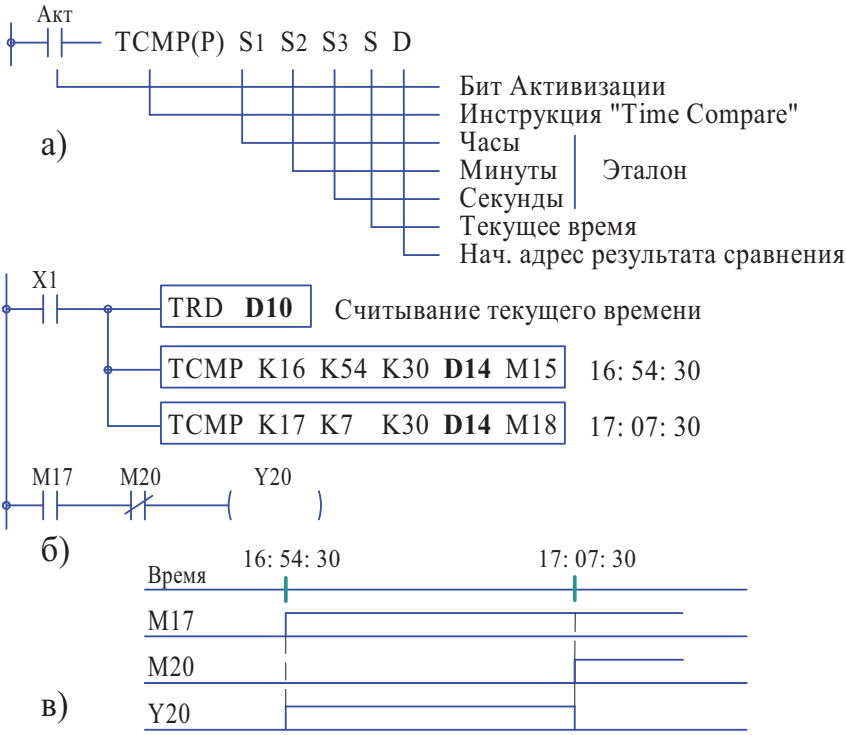


Рис. 3.88. Синтаксис (а), программа (б) и диаграмма работы инструкции сравнения времени TCMP

Результат сравнения следующий:

M15 = 1, если текущее время (D14, D15, D16) < 16 : 54 : 30;
M16 = 1, если текущее время (D14, D15, D16) = 16 : 54 : 30;
M17 = 1, если текущее время (D14, D15, D16) > 16 : 54 : 30

и

M18 = 1, если текущее время (D14, D15, D16) < 17 : 07 : 30;
M19 = 1, если текущее время (D14, D15, D16) = 17 : 07 : 30;
M20 = 1, если текущее время (D14, D15, D16) > 17 : 07 : 30.

Используя результаты сравнения можно синтезировать любое необходимое действие в требуемый момент или диапазон времени. В приведенной программе в течение заданного отрезка времени включается выход Y20.

Ниже приведен синтаксис еще трех инструкций работы реальным временем:

TZCP – инструкция сравнения текущего времени с эталонным интервалом (рис. 3.89). Выполняет сравнение на большее значение (>) и на меньше или равно (≤). Область применения инструкции аналогична задачам, решенным на рис. 3.88 при помощи инструкции TCMP;

TADD – инструкция сложения двух эталонов времени (рис. 3.90);

TSUB – инструкция вычитания двух эталонов времени (рис. 3.91).

Инструкция TZCP

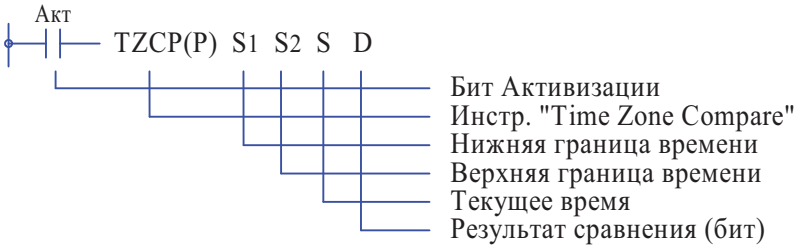


Рис. 3.89. Синтаксис инструкции TZCP

Инструкция TADD

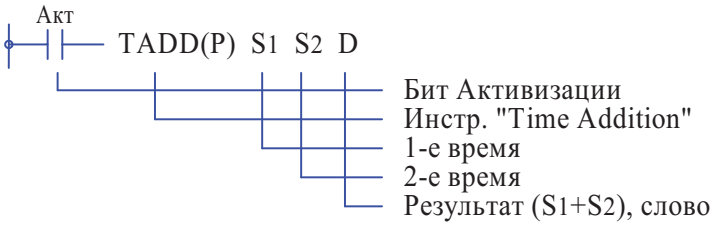


Рис. 3.90. Синтаксис инструкции TADD

Инструкция TSUB

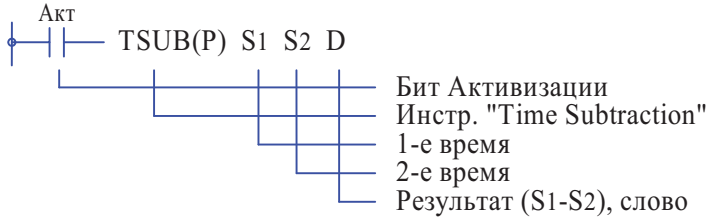


Рис. 3.91. Синтаксис инструкции TSUB

PID-регуляторы

Прежде чем рассмотреть инструкцию программного PID-регулятора давайте разберемся, что это такое. Полагаю, что это будет полезно многим, так как это достаточно сложное устройство автоматического регулирования, и не только по структуре, но по наладке. На рис. 3.92 приведена классическая структурная схема системы автоматического регулирования управления промышленным объектом, где:

SV (Set Value) – Управляющее (задающее) воздействие;
PV (Previous Value) – сигнал обратной связи от объекта управления;
E (Error) – сигнал ошибки регулирования (рассогласование);
U – сигнал управления объектом;
Исп. Оран – исполнительный орган управления объектом, например, электропривод).

Задача регулятора заключается в том, чтобы на основании сравнения информации задающего сигнала и сигнала обратной связи от исполнительного органа или объекта управления (ошибки) сформировать управляющий сигнал u(t), обеспечивающий требуемое качество работы в стационарных и переходных режимах.

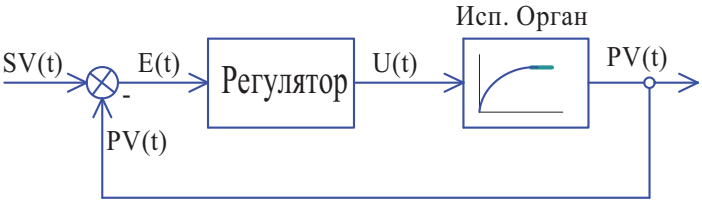


Рис. 3.92. Структурная схема регулирования

Основные виды регуляторов:
пропорциональный П-регулятор;
интегральный И-регулятор;
пропорционально-интегральный ПИ-регулятор;
дифференциальный Д-регулятор;
пропорционально-интегрально-дифференциальный ПИД-регулятор;
релейный Р-регулятор;
нечеткие регуляторы (Fuzzy Logic).

На рис. 3.93 приведены простейшие регуляторы **без** обратной связи, построенные на базе операционных усилителей.

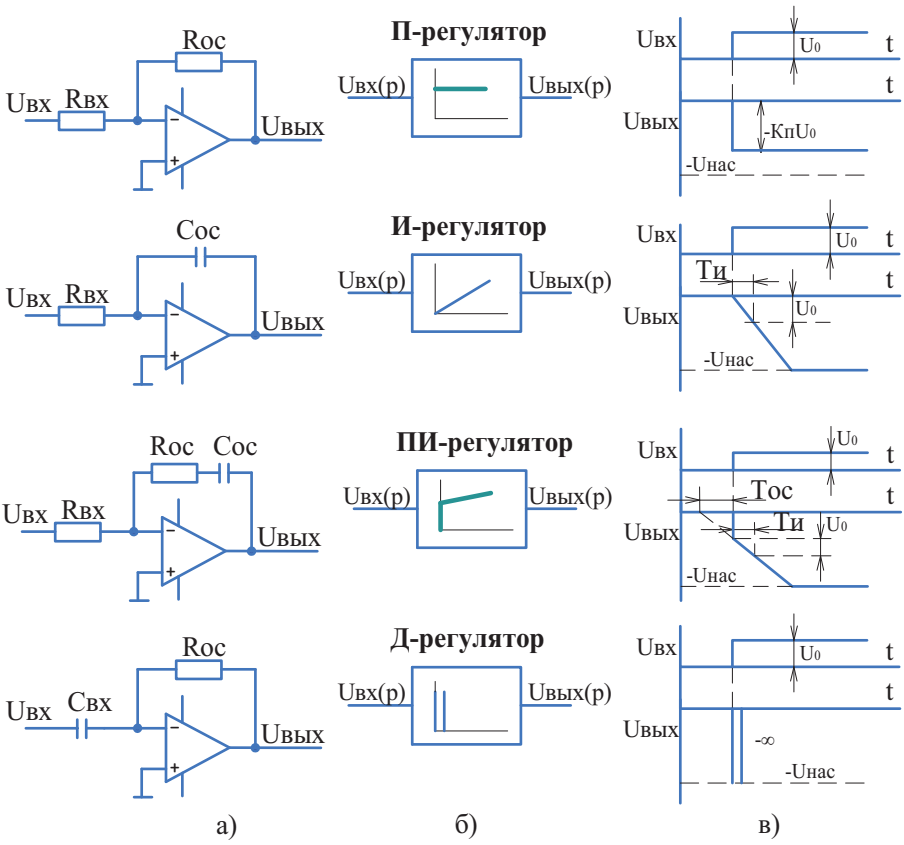


Рис. 3.93. П, И, ПИ, Д-регуляторы (принципиальная схема, условное обозначение, диаграммы работы)

Пропорциональный П-регулятор (а) формирует управляющее воздействие путем простого умножения задающего воздействия или сигнала ошибки в регуляторе с обратной связью на коэффициент пропорционального усиления, т. е. $u(t) = \text{Кп} \cdot E(t)$.

Преимуществом такого регулятора является простота и высокое быстродействие, так как в нем отсутствуют атрибуты задержек.

Принципиальным недостатком является невозможность полного устранения ошибки. Чем выше коэффициент усиления, тем быстрее реагирует регулятор и меньше ошибка. Однако большинство реальных объектов управления, с точки зрения теории управления, не являются пропорциональными, а являются,

например, апериодическими звеньями, то при увеличении усиления в системе возникают колебания.

При малых Кп такая система имеет небольшое перерегулирование, но большую статическую погрешность. С ростом Кп погрешность уменьшается, но перерегулирование возрастает.

Интегральный И-регулятор (б) формирует управляющее воздействие пропорционально интегралу отклонения регулируемой величины $u(t) = \frac{1}{\text{Ти}} \cdot \int_0^t E(t)dt$ до тех пор, пока не восстановится ее заданное значение.

Главным атрибутом регулятора является постоянная времени интегрирования Ти.

При больших значениях постоянной времени Ти переходный процесс в системе имеет апериодический характер. По мере уменьшения Ти переходный процесс становится колебательным.

И главное, система с интегральным регулятором не имеет ошибки в установившемся режиме.

Пропорционально-интегральный ПИ-регулятор (в) объединяет возможности пропорционального и интегрального регулятора, позволяя раздельно регулировать как коэффициент пропорционального усиления Кп, так и постоянную времени интегрирования Ти. Это дает возможность достичь качественных переходных процессов в системе регулирования.

Обратим внимание на диаграмму реакции регулятора на скачек задающего сигнала. В начальный момент управляющий сигнал равен $u(t)$ = Кп·$E(t)$ и только после этого начинается его увеличение по интегральному закону.

Регулятор как бы предугадывает последующее изменение задания и начинает реагировать заранее, т. е. учитывает предыдущее состояние.

Идеальный ПИ-регулятор также не имеет ошибки в установившемся режиме.

Перечисленные преимущества обеспечили ему широкое применение.

Дифференциальный Д-регулятор (г) формирует управляющее воздействие путем дифференцирования задающего сигнала ил сигнала ошибки, т. е. $u(t) = \text{Тд} \cdot \frac{dE(t)}{dt}$.

Главным атрибутом регулятора является постоянная времени дифференцирования Тд.

Знание о скорости изменения ошибки позволяет регулятору как бы реагировать на последующие изменения управляющего воздействия, и тем самым изменить в лучшую сторону характер переходного процесса.

Если ошибка постоянна, то такой регулятор, в силу своего принципа работы, на нее не реагирует. Отсюда следует, что он не может применяться самостоятельно, только в паре с П-регулятором.

Пропорционально-интегрально-дифференциальный ПИД-регулятор является наиболее гибким инструментом управления, объединяющим три способа регулирования со всеми их преимуществами и недостатками (рис. 3.94).

Управляющее воздействие такого регулятора формируется согласно уравнению $U(t) = \text{Кп} \cdot E(t) + \frac{1}{\text{Ти}} \cdot \int_0^t E(t)\,dt + \text{Тд} \cdot \frac{dE(t)}{dt}$.

В настоящее время это наиболее распространенный тип автономных регуляторов, выпускаемых аппаратно и реализуемых программным путем в составе программируемых логических контроллеров.

Однако широкие возможности обуславливают и сложность их настройки. Ведущие производители, как правило, приводят инженерные методики, позволяющие облегчить эту задачу, в структуру PID-регуляторов вводятся специальные алгоритмы автоматической настройки. Главным определяющим фактором здесь является сам объект управления, его сложность и функциональные характеристики, не следует также забывать, что при автоматизации сложных объектов возникают проблемы дополнительных внешних воздействий и «шумов».

Рассмотрев эту предварительную информацию обратимся к программному PID-регулятору контроллера.

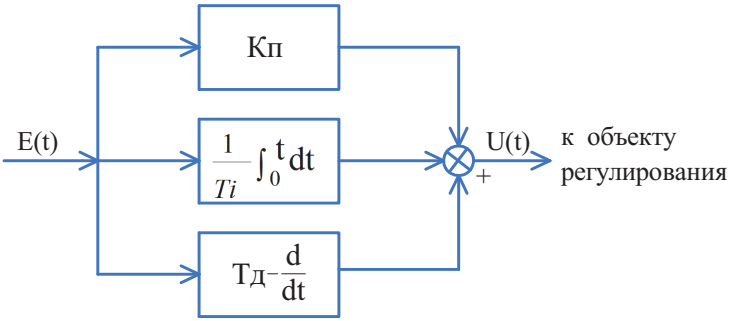


Рис. 3.94. Структура ПИД-регулятора

Инструкция PID

Структура инструкции (рис. 3.95) включает задание следующих операндов:

SV (Set Value) – уставка задания;

PV (Present Value) – текущая величина;

PS (Parameter Setting) – нач. адрес системы параметров регулятора;

OV (Output Value) – выходная величина.

При программировании входных величин используются только шестнадцатеричные или двойные D-слова, информация в которые (S1, S2, S3) заносится заранее. Выходные расчетные данные также заносятся в D-слово.

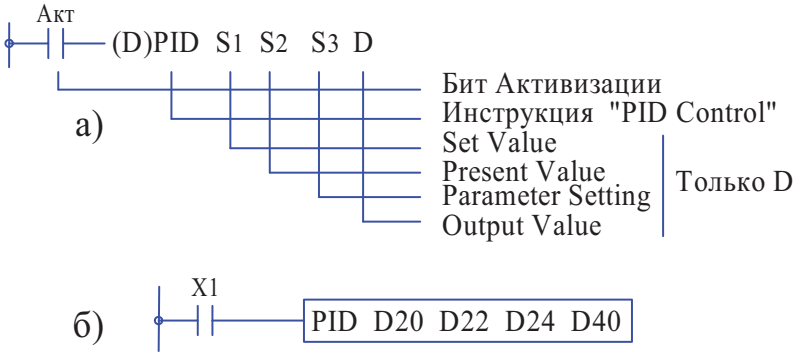


Рис. 3.95. Синтаксис (а) и программа (б) PID-регулятора

Система параметров 16-разрядного регулятора занимает 20 адресов:

S3 – время Ts на вычисления перед выдачей очередного расчетного выходного значения (1–2000, дискрета 10 мс);

(S3 + 1) – коэффициент пропорционального усиления (0–30 000 %);

(S3 + 2) – интегральный коэффициент Ки для режимов К0–К5 (0–30 000 %)

– постоянная времени интегрирования Ти для режима К10 (0–30 000 мс);

(S3 + 3) – дифференциальный коэффициент Кд для режимов К0–К5 (±30 000 %)

– постоянная времени дифференцирования Тд для режима К10 (±30 000 мс);

(S3 + 4) – режим работы регулятора:

= 0 – автоматическое регулирование;

= 1 – прямое регулирование E = (SV–PV);

= 2 – обратное регулирование E = (PV–SV);

= 3 – режим автонастройки при регулировании температуры, по окончании которого, в слово (S + 4) автоматически записывается число К4 (переход в режим 4);

= 4 – специальный режим для регулирования температуры;

= 5 – автоматический режим с управлением по верхней и нижней границам выхода OV;

= 10 – автоматический режим Ти/Тд с контролем верхней и нижней границ выхода OV;

(S3 + 5) – допустимая зона ошибки (0–32767);

(S3 + 6) – ограничение верхнего значения выходного сигнала (32 767);

(S3 + 7) – ограничение нижнего значения выходного сигнала (32 767);

(S3 + 8) – ограничение верхнего уровня постоянной времени интегрирования (32767);

(S3 +9) – ограничение нижнего уровня постоянной времени интегрирования (32767);

(S3 + 10, + 11) – накопленное значение времени интегрирования (справка);

(S3 + 12) – данные с датчика обратной связи PV (справка);

(S3 + 13 … +19) – служебная информация.

Для 32-разрядного регулятора числовые значения ограничений составляют ±2 147 483 647, поэтому параметры, начиная с (S + 5), занимают по два слова и их адреса, соответственно, смещены (S3 + 5, 6), (S3 + 7, 8), (S3 + 9, 10), (S3 + 11, 12), (S3 + 13, 14), (S3 + 15, 16), (S3 + 17, 19), (S3 + 20).

Как видно из перечня параметров и их числовых значений, достаточно трудно решить, с чего же начать, какие начальные цифры установить? Совершенно очевидно, что осваивать эту технику нужно на полноценном стенде или осторожно на реальном объекте.

В сопроводительной документации на контроллер приведены начальные сведения по его применению и рекомендации по установке параметров.

Первый шаг можно сделать в режиме использования регулятора для поддержания температуры, для чего в паре с PID-регулятором применять инструкцию широтно-импульсной модуляции GPWM, а в качестве датчика температуры использовать встроенный потенциометр и инструкции обработки его сигнала VRRD и VRSC (рис. 3.96).

Краткое описание программы рис. 3.96:

– поддержание температуры на заданном уровне осуществляется за счет регулирования времени включения нагревателя (выход Y20, инструкция GPWM, период ШИМ фиксирован);
– время включенного состояния Y20 определяет PID-регулятор;
– датчик температуры имитируется встроенным в контроллер потенциометром VR0 и считывается инструкцией VRRD;
– в начале программы импульсом начальной установки задаются основные параметры PID-регулятора, а также период ШИМ инструкции GPWM;
– предусмотрено задание двух режимов работы PID-регулятора, с автонастройкой (К3) и спецрежима регулирования температуры (К4);

– в случае задания режима К3 после нескольких поворотов потенциометра (имитация изменения температуры) выполняется автонастройка регулятора и автоматическое переключение на режим К4.

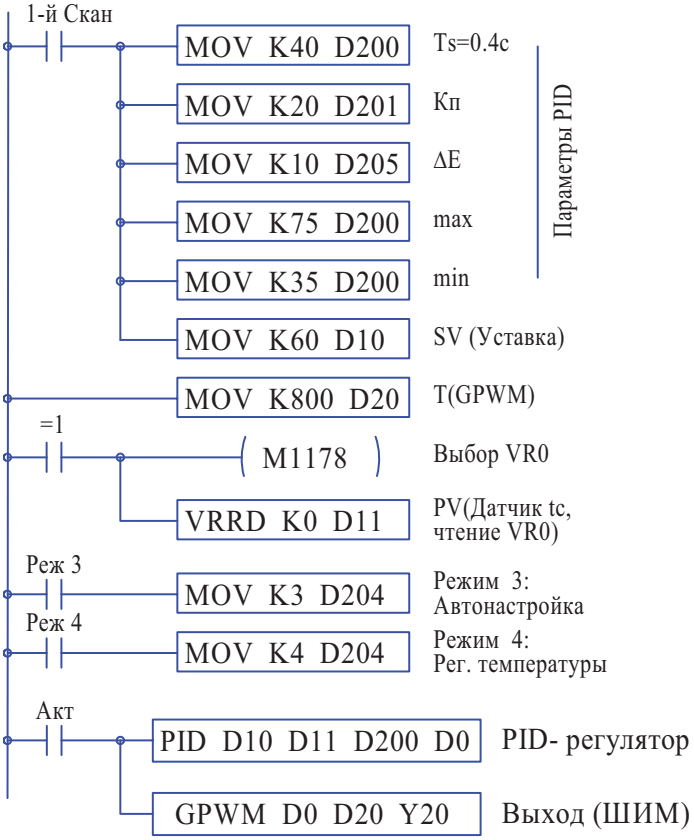


Рис. 3.96. PID-регулятор температуры (для изучения)

Программа позволяет понять лишь общие принципы работы регулятора. Далее нужно продолжить освоение на реальном стенде, приобретать опыт. Это долго, так как процесс нагрева и остывания инерционный, но другого пути нет. И не нужно бояться спрашивать у коллег, имеющих такой опыт.

Другого совета автор дать не может, так как сам так же этого опыта не имеет.

Инструкции чтения и записи специальных модулей

Данная группа инструкций предназначена для работы со специальными модулями, например, аналоговыми или позиционирования, параметры которых хранятся в специальных CR – регистрах. К ним относятся:

FROM – чтение параметра (рис. 3.97, *а*);

TO – запись параметра (рис. 3.97, *б*);

BON – дешифратор бита из слова (рис. 3.98).

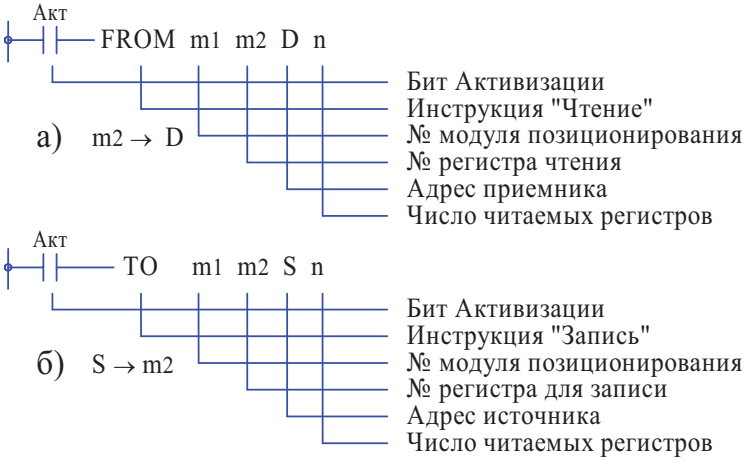


Рис. 3.97. Синтаксис инструкций FROM (а) и TO (б)

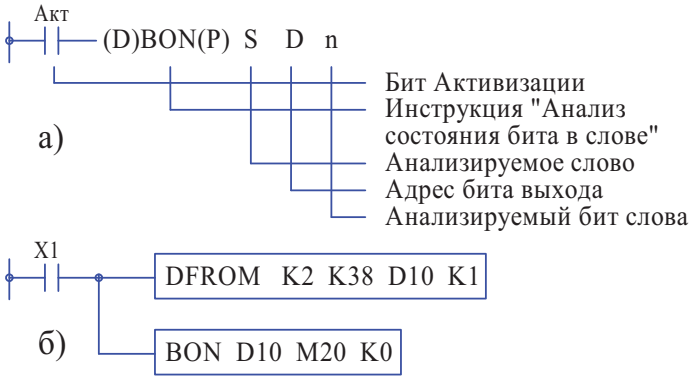


Рис. 3.98. Синтаксис (а) и программа дешифрации бита (б)

При программировании указывается номер модуля, номер регистра, адрес приемника и число читаемых (записываемых) регистров.

Адресация модулей устанавливается при конфигурации контроллера и начинается с нуля.

Инструкция BON позволяет прочитать любой бит параметра и записать его в адресуемую ячейку для дальнейшей работы.

В примере рис. 3.98:

– инструкция DFROM читает слово параметра CR38 (K38) блока позиционирования №3 (K2) и записывает его в слово приемника D10. Число читаемых регистров – один (K1);
– инструкция BON анализирует слово D10, находит в нем нужный бит (K0) и его значение (ноль или единица) записывает в адресуемый приемник, т. е. ячейку M20.

Функциональные инструкции для работы с числами с плавающей запятой

Необходимость изучения инструкций, предназначенных для работы с числами с плавающей запятой, связана с тем, что все тригонометрические вычисления в контроллере выполняются в данном формате представления чисел. Необходимые (и, надеюсь, достаточные) сведения о числах с плавающей запятой изложены ранее в главе 1.

Признаком числа с плавающей запятой является буква F.

Важно! Если при решении задач автоматизации выполняются какие-либо тригонометрические действия, связанные, например, с вычислением синуса или косинуса, а затем требуется выполнить арифметические действия, то нельзя использовать стандартные арифметические инструкции, так как не совпадут форматы представления чисел. Необходимо использовать специальную арифметику работы с числами с плавающей запятой.

Приведем список таких инструкций (их признак – буква E в начале инструкции):

(D)EADD(P) – арифметическое сложение;
(D)ESUB(P) – арифметическое вычитание;
(D)EMUL(P) – арифметическое умножение;
(D)EDIV(P) – арифметическое деление;
(D)ESQR(P) – вычисление квадратного корня;
(D)EBCD(P) – преобразование двоичных чисел в десятичные;
(D)EBIN(P) – преобразование десятичных в двоичные;
(D)ECMP(P) – сравнение чисел с плавающей запятой;
(D)EZCMP(P) – зонное сравнение чисел с плавающей запятой.

Тригонометрические инструкции:

(D)SIN(P) – вычисление синуса;

(D)COS(P) – вычисление косинуса;
(D)TAN(P) – вычисление тангенса;
(D)ASIN(P) – вычисление арксинуса;
(D)ACOS(P) – вычисление арккосинуса;
(D)ATAN(P) – вычисление арктангенса;
(D)RAD(P) – перевод градусов в радианы;
(D)DEG(P) – перевод радиан в градусы.

Так же работают с плавающей запятой:
(D)MOVR(P) – пересылка чисел с плавающей запятой;
(D)INT(P) – преобразование в целое двоичное число;
(D)FLT(P) – преобразование целого числа в число с плавающей запятой и наоборот;
(D)EXP(P) – вычисление экспоненты;
(D)LOG(P) – вычисление логарифма;
(D)LN(P) – вычисление натурального алгоритма;
(D)POW(P) – возведение в степень.

Инструкции RAD / DEG

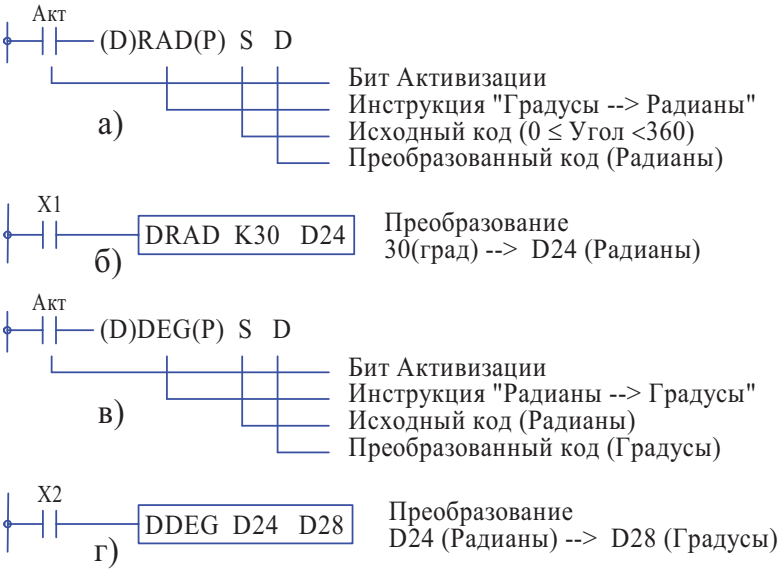


Рис. 3.99. Синтаксис (а, в) и Ladder-диаграммы (б, г) инструкций RAD и DEG

Инструкция RAD (рис. 3.99, *а*, *б*) предназначена для преобразования градусов в радианы, $\text{Рад} = \text{Град} \cdot \frac{\pi}{180}$, а инструкция DEG (рис. 3.99, *в*, *г*) для обратного преобразования радиан в градусы, $\text{Град} = \text{Рад} \cdot \frac{180}{\pi}$.

Доступные операнды источника: K, H, D.

В качестве приемника используется только слова D.

Для контроллеров данной серии разрешаются *только* двойные слава.

Доступны следующие маркеры:

M1020 – флаг нулевого значения (результат равен нулю);

M1021 – флаг заимствования (абсолютное значение результата меньше минимально допустимого числа с плавающей запятой);

M1022 – флаг переноса (абсолютное значение результата превышает максимально допустимое число с плавающей запятой).

Инструкция INT

Инструкция INT (рис. 3.100) преобразует двоичное число с плавающей запятой в двоичное целое число.

Доступны только двойные слова.

Работают маркеры M1020, M1021 и M1022.

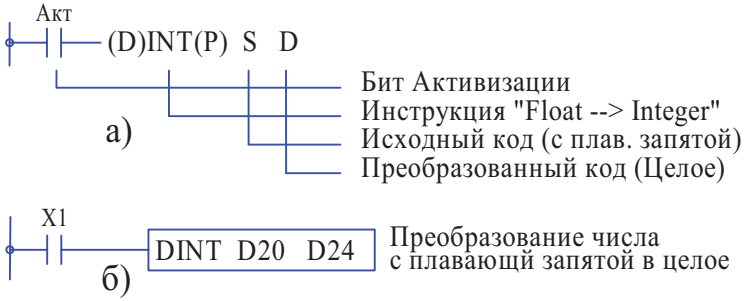


Рис. 3.100. Синтаксис (а) и Ladder-диаграмма (б) инструкции INT

Инструкция FLT

В отличие от инструкции INT, инструкция FLT *универсальна*, выполняемое инструкцией преобразование зависит от состояния маркера M1081:

при M1081 = 0 выполняется преобразование целого двоичного числа в двоичное число с плавающей запятой;

при M1081 = 1 выполняется обратное преобразование двоичного числа с плавающей запятой в целое двоичное число.

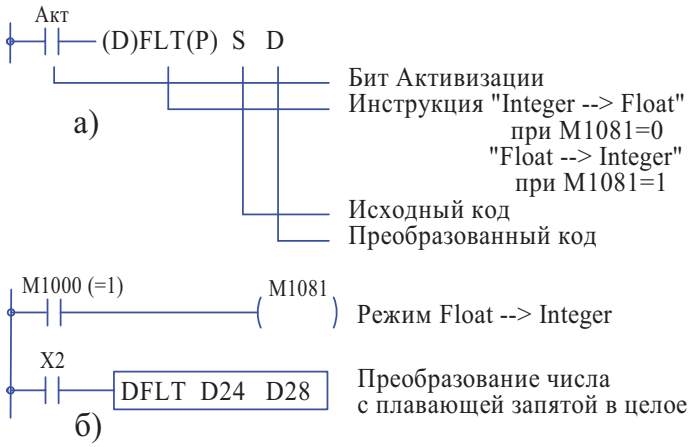


Рис. 3.101. Синтаксис (а) и Ladder-диаграмма (б) инструкции FLT

Доступны как шестнадцатеричные, так и двойные слова.
Работают маркеры M1020, M1021 и M1022.

Инструкция MOVR

Инструкция предназначена для выполнения операции пересылки чисел, представленных в формате с плавающей запятой.

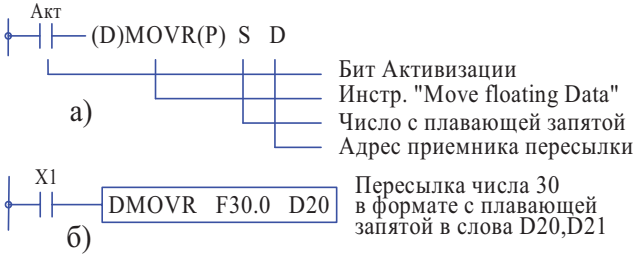


Рис. 3.102. Синтаксис (а) и Ladder-диаграмма (б) инструкции MOVR

Тригонометрические инструкции

В языке контроллера предусмотрены инструкции вычисления синуса, косинуса, тангенса, а также обратных функций арксинуса, арккосинуса и арктангенса.

Инструкция вычисления синуса показана на рис. 3.103

Вычисление может выполняться как в градусах, так и в радианах, в зависимости от уставки маркера M1018.

Для данной серии контроллеров разрешены вычисления только в двойных словах.

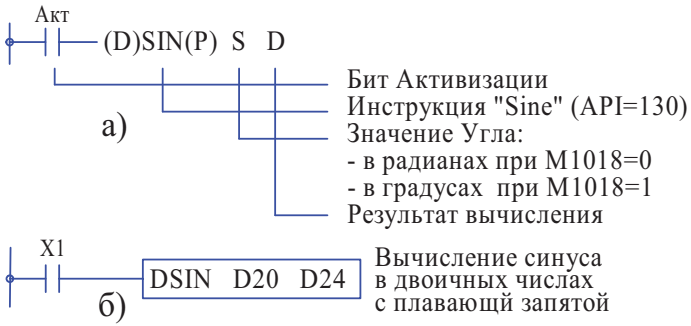


Рис. 3.103. Синтаксис (а) и Ladder-диаграмма (б) инструкции SIN

Принцип вычисления косинуса и тангенса аналогичен.

Инструкция вычисления арксинуса показана на рис. 3.104

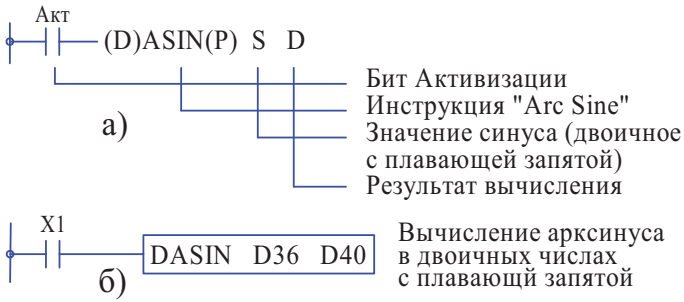


Рис. 3.104. Синтаксис (а) и Ladder-диаграмма (б) инструкции ASIN

Структура инструкций также аналогична.

Ниже, в качестве примера, приведены рабочие программы тригонометрических вычислений с функцией синуса в градусах (рис. 3.105) и радианах (рис. 3.106).

Особый класс инструкций составляют высокоскоростные инструкции и инструкции позиционирования. Первые предназначены для работы совместно с высокоскоростными счетчиками, вторые с блоками позиционирования. Здесь приведем лишь их неполный перечень. Рабочие примеры использования данных инструкций будут приведены в других главах.

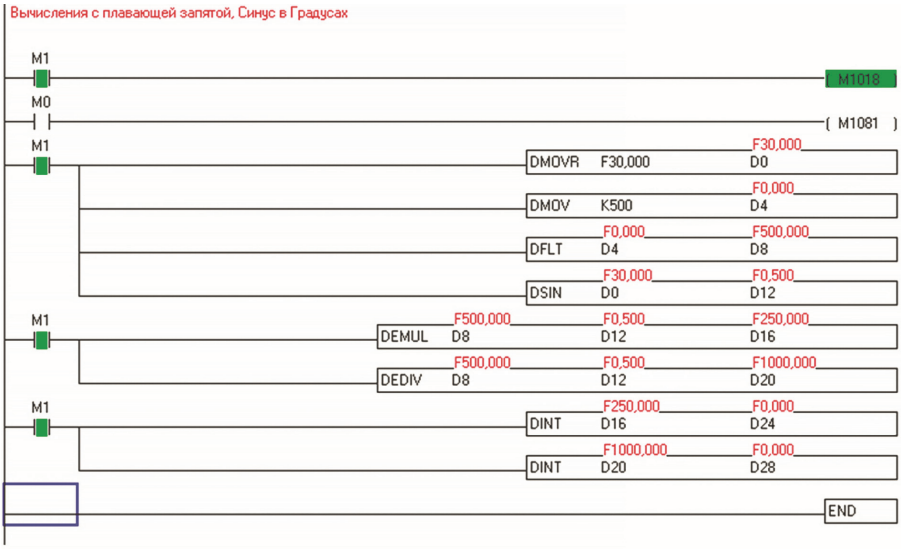


Рис. 3.105. Пример программы вычислений с числами с плавающей запятой (задание угла в градусах)

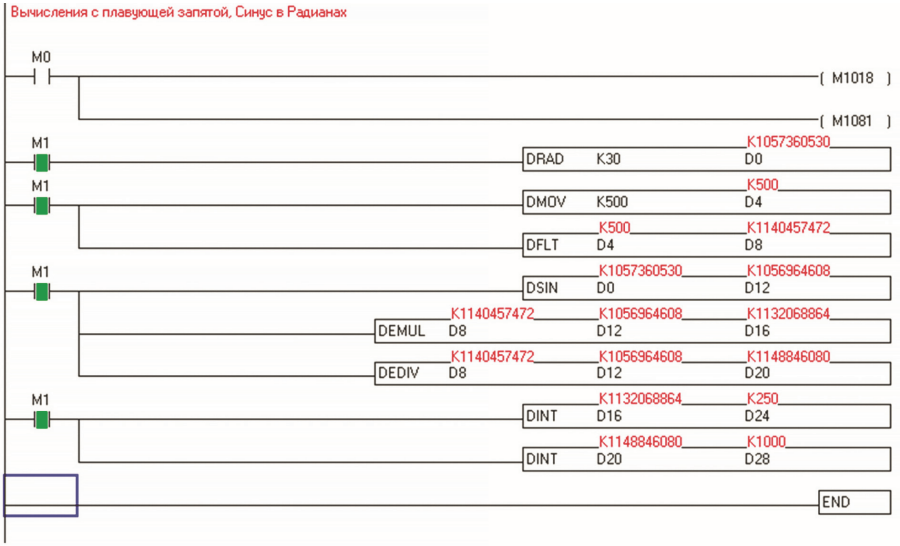


Рис. 3.106. Пример программы вычислений с числами с плавающей запятой (задание угла в радианах)

Высокоскоростные инструкции:

DPLSY – формирование заданного числа и частоты последовательности импульсов (до 100 000 Гц);

DPLSR – формирование заданного числа и частоты последовательности импульсов с программируемым ускорением и замедлением;

SPD – расчет скорости выдачи импульсов;

PWM – импульсный выход с ШИМ;

DHSCS – включение выхода высокоскоростного счетчика;

DHSCR – выключение выхода высокоскоростного счетчика;

DHSZ – зонное сравнение при высокоскоростном счете.

Инструкции позиционирования:

DABSR – чтение абсолютного текущего положения;

DZRN – инструкция выход в ноль (исходное положение);

DPLSV – формирование заданного направление вращения и частоты управляющих импульсов для сервоприводов;

DDRVI – управление позиционированием в относительных координатах;

DDRVA – управление позиционированием в абсолютных координатах.

Приведенный выше далеко не полный перечень функциональных инструкций показывает огромные возможности современных программируемых контроллеров. Изучить все и профессионально применять это трудно даже для специалистов, специализирующихся на конкретном типе контроллера. Это и понятно, у них так же нет возможности проверить все на практике. В этой связи, вновь выдвигаю тезис, что нужно как можно больше применять проверенные простые типовые решения, пригодные для разных типов контроллеров. Пусть они будут не оптимальны с точки зрения профессионального программиста, но могут быть быстро и без ошибок интегрированы в рабочие программы. Из шестидесятилетнего опыта знаю, что профессиональные программисты, как правило, плохо или совсем не разбираются в объектах, для которых они пишут программы, не учитывают несовместимые и аварийные ситуации, пропускают очевидные блокировки, делают неудобные интерфейсы и т. д. Часто, при консультациях, с ними трудно найти общий язык, и приходится все решать самостоятельно. Это трудно, но решение всегда находится. Так что дерзайте, и все получится.

3.10. Примеры программирования

Ниже приводятся примеры рабочих программ управления различными механизмами автоматизированной пилы по распилке металлических листов (см. рис. 3.11 и 3.12).

3.10.1. Аналоговое управление частотным преобразователем

В качестве электропривода вращения пилы используется частотный преобразователь фирмы «Альтивар». Для задания скорости вращения используется однополярный аналоговый канал +10 В, встроенный в процессорный блок (см. рис. 3.7). Реверс осуществляется дискретными входными сигналами CW и CCW с совмещением функции деблокировки привода.

Управление приводом осуществляется с программируемых панелей оператора (рис. 3.107 и 3.111), на которых расположены следующие органы управления:

– цифровой задатчик скорости в режиме преднабора параметров (рисунок 3.105);
– кнопки управления вращением (на панели Наладка):
  КнВ – включение вращения по часовой стрелке;
  КнН – включение вращения против часовой стрелки;
  КнС – выключение привода;
– кнопки оперативной коррекции скорости (рис. 3.107):
  Кн>> – увеличение скорости вращения;
  Кн<< – уменьшение скорости вращения;
  Кн100 – установка рабочей скорости.

Рис. 3.107. Окно задания параметров управления пилой

Ниже приведены фрагменты проекта:

рис. 3.108 – блок-схема управления приводом вращения;
рис. 3.109 – рабочая программа управления вращением привода;
рис. 3.110 – рабочая программа оперативного изменения скорости вращения.

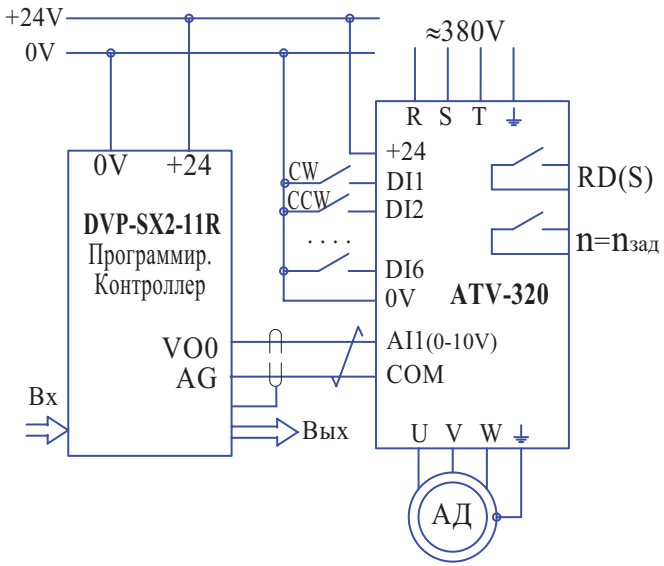


Рис. 3.108. Блок-схема управления приводом вращения пилы

Программа рис. 3.109 построена по следующему принципу:

– включение силового пускателя привода;
– выбор способа управления +10 В (D1115 = 0);
– формирование вспомогательных сигналов Запрета и Разрешения включения;
– формирование виртуальных сигналов вращения Вперед (По часовой) и Назад (Против часовой);
– формирование сигналов деблокировки вращения По (CW) и Против (CCW) часовой стрелки;
– выдача кода 1-го канала ЦАП (D1116). Код 2000 соответствует напряжению 10В. Здесь установлена фиксированная скорость К1000;
– обнуление ЦАП при остановке.

Рис. 3.110 – это типовой шаблон для оперативного изменения того или иного параметра. Принцип его построения следующий:

D50 – виртуальное слово для хранения изменяемого параметра.

При включении контроллера в слово D50 автоматически заносится начальное значение параметра, в данном случае 100. Ту же установку можно сделать в любой момент нажатием кнопки 100 %.

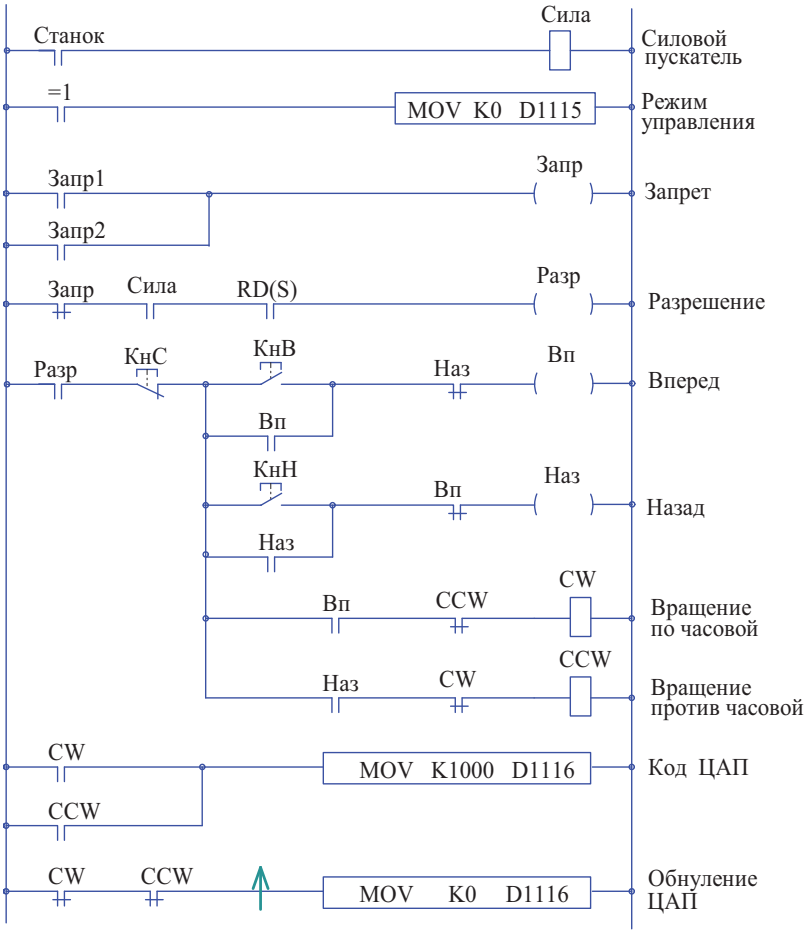


Рис. 3.109. Рабочая программа управления вращением пилы

При нажатии кнопок Кн>> или Кн<< запускается генератор импульсов и при помощи инструкции INC или DEC осуществляется увеличение или уменьшение кода, записанного в слово D50.

Предусмотрено ограничение максимального кода на уровне K120 и минимального на уровне K80, при достижении которых генератор отключается, прекращая дальнейшее изменение кода.

Для точной подгонки скорости вращения объекта управления (в данном случае пилы) заданной в параметрах предусмотрено масштабирование кода D50 путем последовательных операций деления и умножения и переписи

результата в слово D54. Код, сформированный в D54, является результирующим для записи в ЦАП (инструкция MOV D54 D1116).

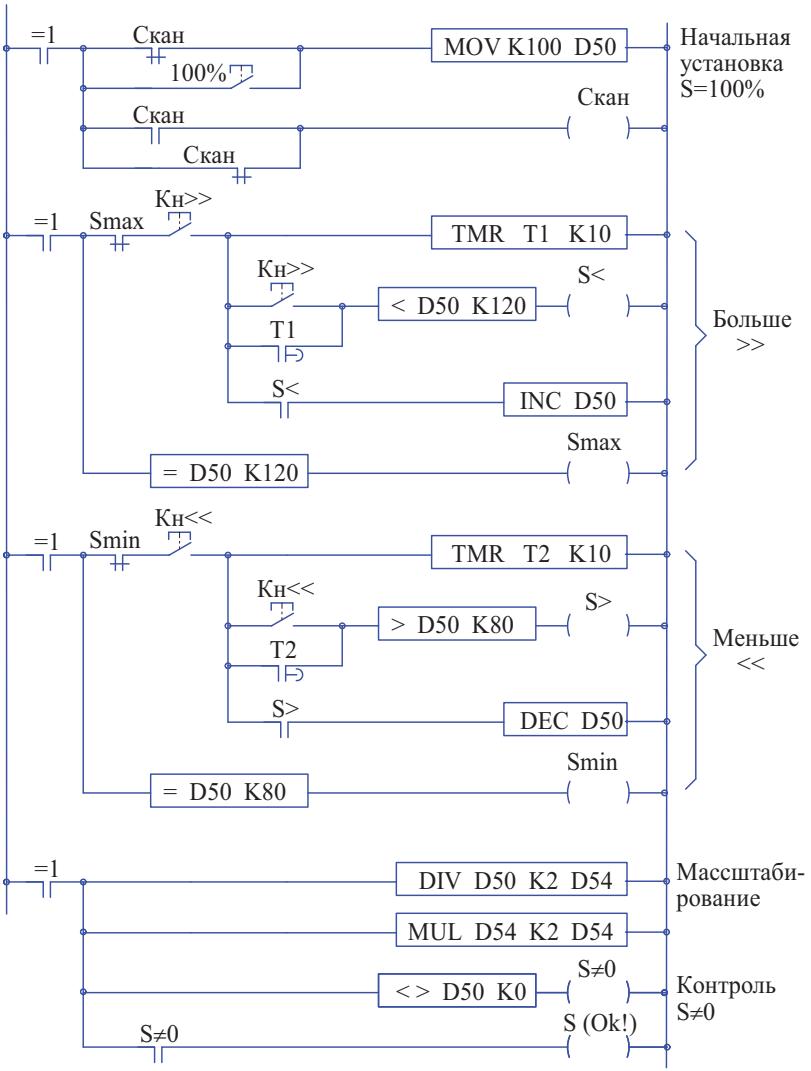


Рис. 3.110. Типовая программа оперативного регулирования скорости с панели оператора

В конце программы выполняется проверка на выполнение операции записи кода S ≠ 0 и S(Ок!).

Еще раз напоминаем, что во избежание «сюрпризов» при выполнении арифметических операций, не задумываясь, отводите на них четыре слова, здесь (D50–D53), (D54–D57). Запаса D-слов вполне достаточно, а любые наслоения адресов будут исключены.

3.10.2. Управление от модуля позиционированием

Реализовать позиционирование координат на контроллерах серии DVP можно двумя способами:

– используя высокоскоростные инструкции и быстрые счетчики, для чего следует заказать соответствующую конфигурацию контроллера с высокоскоростными транзисторными выходами. В нашей конфигурации таких выходов нет. Данный способ будет рассмотрен в гл. 5;

– используя специализированные модули позиционирования (раздел 3.5). В конфигурации рассматриваемого контроллера имеется четыре модуля позиционирования (см. рис. 3.14). Три управляют шаговыми электроприводами подачи листа и карты (заготовки) в зону резания, а также бегунком поддержки лезвия пилы. Четвертый блок управляет сервоприводом перемещения пилы вверх и вниз при резании. Общий вид пилы приведен на рис. 3.11 (зона подачи листа) и 3.12 (зона подачи карты).

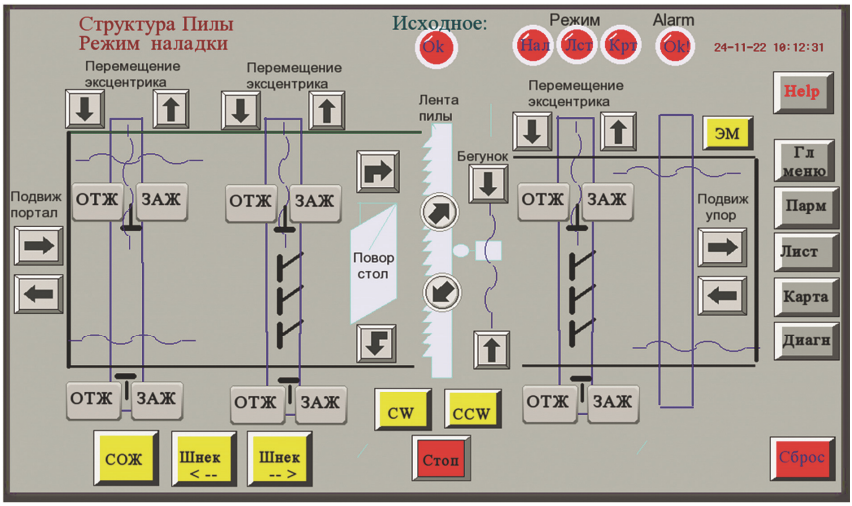


Рис. 3.111. Окно управления пилой в наладочном режиме

На рис. 3.111 показаны кнопочные органы управления перечисленными приводами в наладочном режиме. Задание параметров движения (размер, скорость) осуществляется из окна установки параметров (см. рис. 3.107).

В проекте предусмотрено также параллельное управление от ручного выносного пульта с фиксированными скоростями.

Блок-схема управления позиционированием шагового привода DM860H от модуля DVP-01PU-S приведена ниже (рис. 3.112). Модуль позволяет управлять приводом как от электроавтоматики PLC, так и от органов, непосредственно подключаемых к модулю (внешнее управление).

Модуль позволяет управлять позиционированием в различных режимах, определяемых внутренними параметрами (CR-регистрами). Прежде чем писать программу электроавтоматики следует разобраться в следующих вопросах:

– как подключить и какие установить параметры в приводе объекта управления. В данном случае это шаговый электропривод. Любой современный электропривод имеет свою внутреннюю систему параметров. В шаговых приводах серии DM, переключателями на их корпусе устанавливается рабочий ток двигателя и, в зависимости от типа привода, скорость вращения двигателя или число импульсов на оборот вала;
– как подключить и какие установить параметры для управления в том или ином режиме работы.

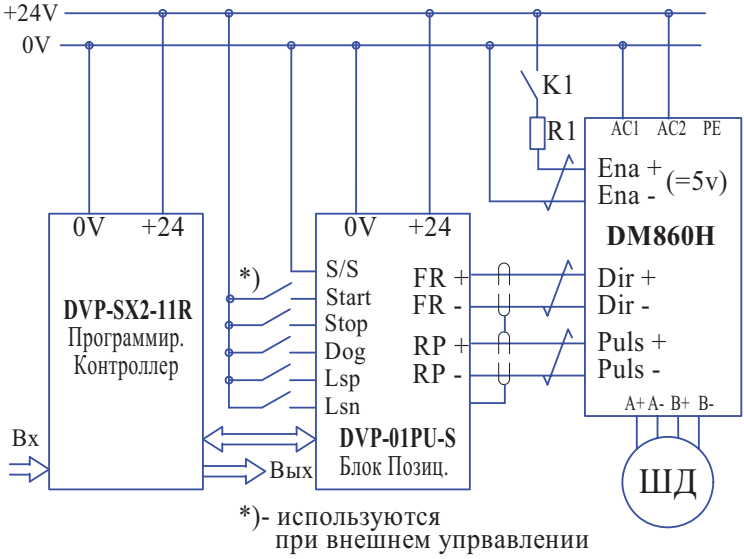


Рис. 3.112. Блок-схема управления позиционированием

Подключение модуля и привода показано на рис. 3.112.

Ниже приводится система параметров (CR-регистров) модуля позиционирования. В программе электроавтоматики номер параметра указывается с буквой К.

CR0 – номер модели;

CR1, CR2 – число импульсов на оборот (по умолчанию 2000);

CR3, CR4 – диапазон в единицах (по умолчанию 1000);

CR5 – битовые параметры, назначение:

b0, b1 – единицы измерения: 00 – относительно двигателя,
01 – относительно механизма

b2, b3 – диапазон установка положения:

$$00 – 10^0, 01 – 10^1, 10 – 10^2, 11 – 10^3;$$

b4, b5 – формат управления: 00 – FR + RP,
01 – импульсы + направление,
10 – двухфазные сигналы A / B.

b6 – полярность сигнала ограничения LSP (0/1);

b7 – полярность сигнала ограничения LSN (0/1);

b8 – направление возврата к нулю;

b9 – направление перемещения (реверс);

b10: = 0 – передний рабочий фронт импульсов;
= 1 – задний рабочий фронт импульсов;

b11: = 0 – конечник DOG на замыкание;
= 1 – конечник DOG на размыкание;

b12 – тип ускорения: 0 – трапецеидальное, 1 – линейное;

b13 – время реакции сигнала Start: 0 – 4 мс, 1 – 8 мс;

b14 – рабочая полярность сигнала Start;

b15 – рабочая полярность сигнала Stop;

CR6, CR7 – максимальная скорость Vmax;

CR8, CR9 – смещение скорости Vbias;

CR10, CR11 – скорость Vjog в наладочном режиме JOG;

CR12, CR13 – скорость Vrt выхода в ноль;

CR14, CR15 – скорость Vcr замедления при выходе в ноль;

CR16 – число импульсов PG0 в режиме N выхода в ноль; CR17 – число импульсов PG0 в режиме P выхода в ноль;;

CR18 – b0 = 0 – режим выхода в ноль, b0 = 1 – режим приоритета;
b1 – режим работы DOG;

CR19, CR20 – параметр выхода в ноль;

CR21 – время разгона Tacc;

CR22 – время торможения Tdec;

CR23, CR24 – задание положения 1;

CR25, CR26 – скорость при задании положения 1;
CR27, CR28 – задание положения 2;
CR29, CR30 – скорость при задании положения 2;
CR31 – битовые параметры, назначение:
b0 – сброс ошибки (флага CR38/b5);
b1 – стоп позиционирования через ПО (аналог КнСтоп);
b2 – запрет CW;
b3 – запрет CCW:
b4 – запуск +JOG;
b5 – запуск –JOG;
b6 – запуск выхода в ноль;
b7 = 0 – абсолютный отсчет;
= 1 – относительный отсчет;
b8 – управление из регистра CR32;
b9 – не используется;
b10 – обнуление текущей позиции;
b11 – не используется;
b12 = 0 – выдача сигнала CLR=130мс после выхода в ноль;
= 1 – выдача сигнала CLR определяется параметром b13;
b13: = 0 – выключен, = 1 – включен;
b14 – не используется;
b15 – не используется;
CR32 – битовые параметры, назначение:
b0 – пуск задания 1 (по умолчанию = 1);
b1 – прерывание задания 1;
b2 – пуск задания 2;
b3 – прерывание задания 2:
b4 – режим управления с переменной скоростью;
b5 – активизация ручного генератора (CR40–CR46);
b6 – режим остановки:
= 0 – игнорирование незавершенного пути;
= 1 – доработка незавершенного пути;
b7 = 0 – число импульсов ручного генератора неограниченно;
=1 – ограниченно (P1) и P(2) ;
b8 = 0 – остановка по LSP / LSN с замедлением;
=1 – остановка без замедления;
b9…b11 – маскирование скорости 1, 2 и прерывания 1, 2;
b12 – сброс параметров на заводские установки;
b13: = 0 – индикация скорости и текущего положения CR35 в импульсах;
= 1 – тоже в единицах пользователя;

b14 – не используется;
b15 – не используется;
CR33, CR34 – текущее положение (CP);
CR35, CR36 – текущая скорость (CS);
CR37 – протокол связи:
b0 – 4800;
b1 – 9600;
b2 – 19200;
b3 – 38400:
b4 – 57600;
b5 – 115200;
b6 – резерв:
b7 – = 0 – протокол RTU;
= 1 – протокол ASCII;
b8…b15 – коммуникационный адрес для RS485 K1;
CR38 – статусный регистр:
b0: =0 состояние готовности;
=1 отработка положения;
b1 – выдача импульсов CW;
b2 – выдача импульсов CCW;
b3 – есть выход в ноль:
b4 – переполнение текущего положения (CR33, CR34);
b5 – обнаружение ошибки (код в CR39);
b6 – положение достигнуто (Jog, Ноль, Позиция):
b7 – пауза движения (состояние Стоп);
b8 – не используется:
b9 – индикация импульсов генератора CW:
b10 – индикация импульсов генератора CCW:
CR39 – коды ошибок:
H0000 – Ошибок нет;
H0001 – ошибка уставки задания 1;
H0002 – ошибка уставки задания 2;
H0010 – ошибка уставки скорости 1;
H0011 – ошибка уставки скорости 2;
H0012 – ошибка уставки замедления при выходе в ноль;
H0013 – ошибка уставки скорости выхода в ноль;
H0014 – ошибка уставки скорости режима JOG;
H0020 – выдача импульсов CW запрещена;
H0021 – выдача импульсов CCW запрещена;
H0080 – аппаратная неисправность;
H0081 – ошибка записи в память;

CR40 – числитель ручного генератора MRG;
CR41 – знаменатель ручного генератора MRG;
CR42, CR43 – выходная частота ручного генератора;
CR44, CR45 – аккумулятор ручного генератора;
CR46 – скорость реакции выхода MRG;
CR47 – индикация состояния входов:
b0 – Start;
b1 – Stop;
b2 – Dog;
b3 – PG0;
b4 – LSP;
b5 – LSN;
b6 – фаза A;
b7 – фаза B;
b8 – CLR;
CR48 – версия программного обеспечения.

Внимание! Чтение и запись параметров управления позиционированием осуществляется при помощи инструкций FROM (чтение) и TO (запись).

Управление позиционированием в режиме Jog

На рис. 3.113 приведен простейший алгоритм управления шаговым двигателем от модуля позиционирования в наладочном режиме от кнопок. Алгоритм соответствует схеме подключения рис. 3.110.

В программе задействован 1-й модуль позиционирования с адресом K0.

Последовательность написания алгоритма. При нажатии и удержании кнопки Вперед или Назад и наличии сигнала разрешения выполняются следующие операции:

1. Устанавливаются следующие параметры
   CR5 = H1010, т. е режим «Импульсы + Направление» (b4) и линейное ускорение (b12).
   CR10 = K2000 – скорость перемещения;
   CR21 = CR22 = H200 – время разгона и торможения.
2. Формируется виртуальный сигнал деблокировки привода. Здесь – инверсный. Так сделано в данном типе привода!
3. Снимается обнуление позиции (CR33).
4. Формируется задержанный сигнал Старт(Δt);
5. В зависимости от нажатой кнопки, формируются сигналы перемещения вперед +JOG (CR31 = K144 = b4 + b7) или перемещения назад – JOG (CR31 = K160 = b5 + b7) в относительной системе координат (b7).

6. Формируется выходной сигнал деблокировки привода и пока удерживается кнопка, осуществляется безразмерное перемещение. Параметр CR33 позволяет считывать текущее положение оси.

7. При отпускании кнопки происходит останов движения (снятие деблокировки, установка ±JOG = 0) и обнуление текущей позиции (CR33 = 0).

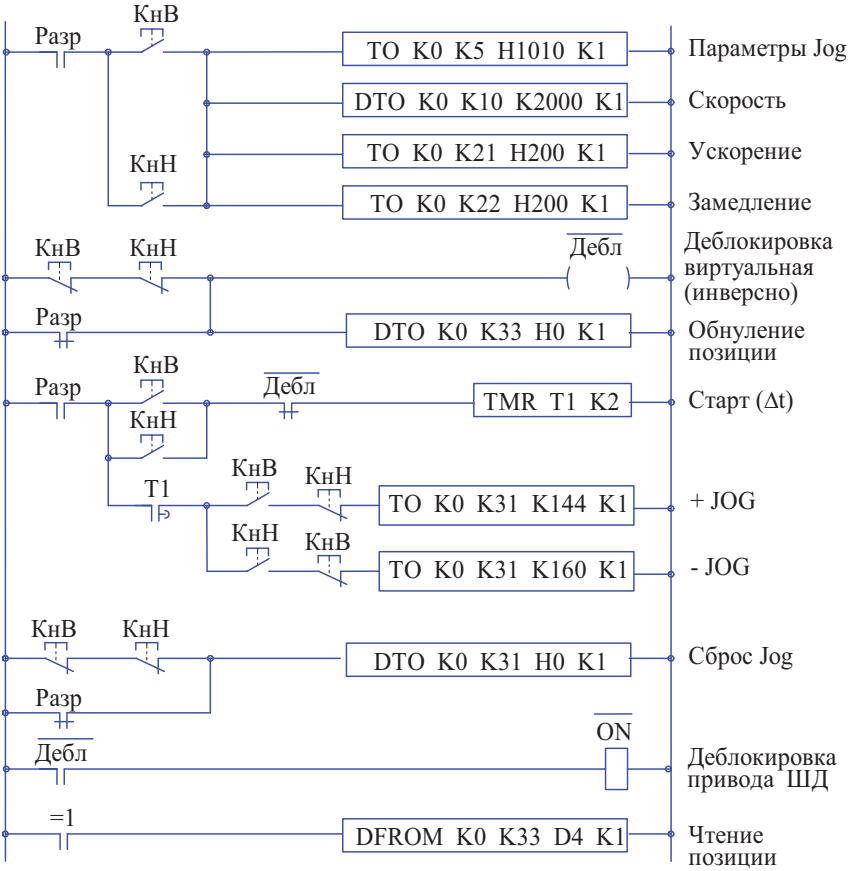


Рис. 3.113. Алгоритм управления шаговым приводом в режиме позиционирования от кнопок

Реальное отображение программы на экране компьютера в среде WPL Soft приведено на рис. 3.114. Проверка алгоритма выполнялась без панели оператора, поэтому для управления использованы адреса дискретных входов (X). При управлении от панели оператора это будут адреса ячеек памяти (M).

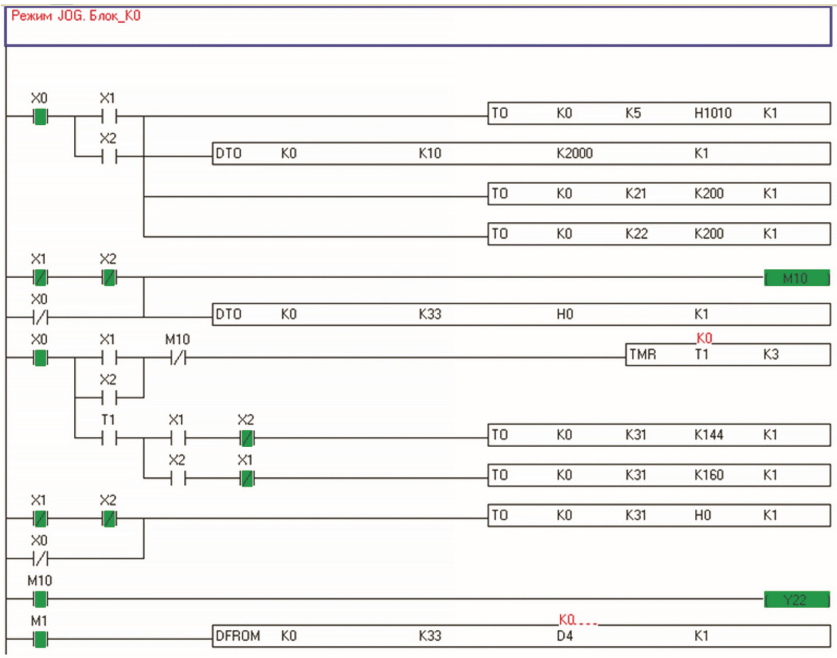


Рис. 3.114. Ladder-диаграмма управления в режиме JOG на экране компьютера

Управление Позиционированием

На рис. 3.115 приведен простейший алгоритм автоматического позиционирования шагового привода. Алгоритм также соответствует схеме подключения рис. 3.110. В программе задействован 1-й модуль позиционирования с адресом K0.

Назначение основных операндов:

M8, M9 – запрет, разрешение;

X1, X2 – кнопки старта позиционирования вперед / назад;

M10, M11, M12 – циклы позиционирования, соответственно, вперед, назад и общий. Все остальные операции выполняются в теле цикла;

M13 – выртуальная деблокировка привода (инверсная);

M17 – ось в движении;

M18 – позиция достигнута;

M19 – память движения;

M20 – сброс цикла;

Y22 – аппаратная деблокировка привода (инверсная).

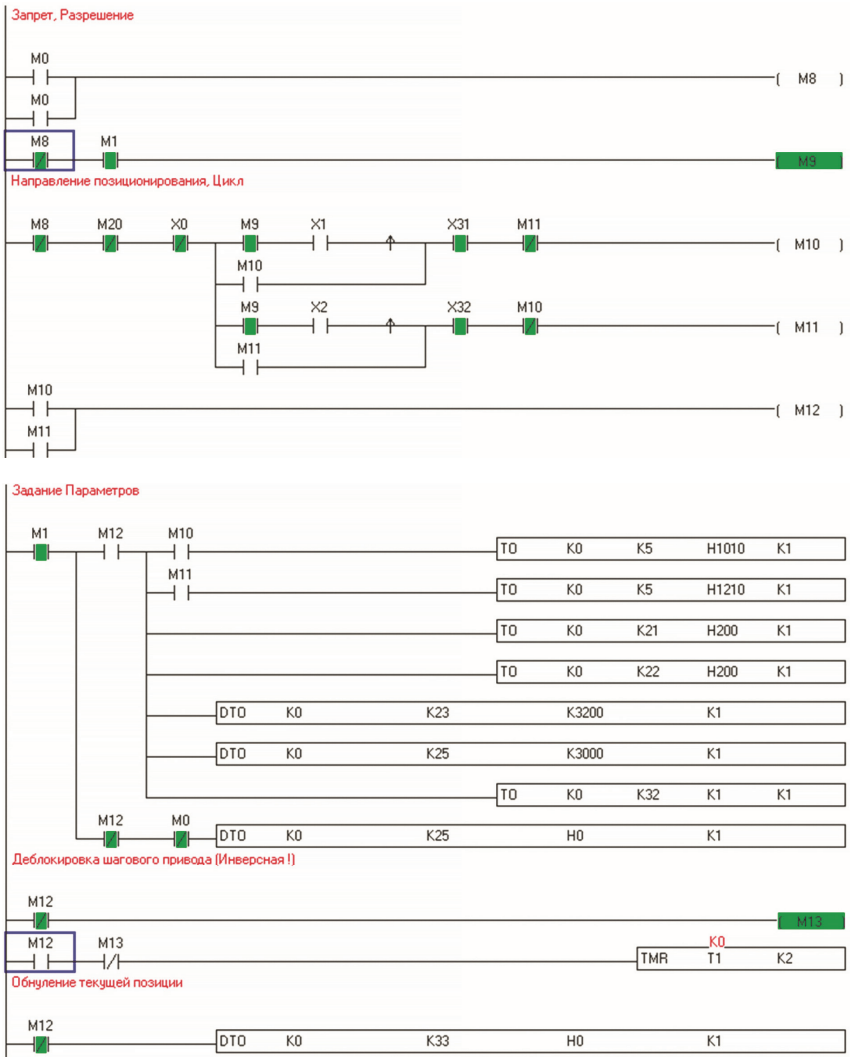


Рис. 3.115. Ladder-диаграмма управления позиционированием в относительных координатах (начало)

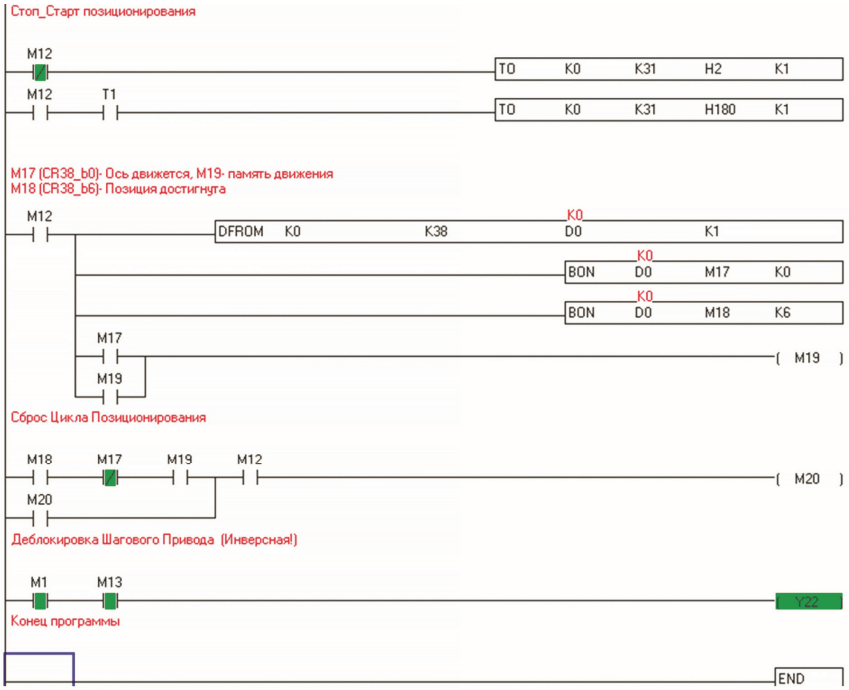


Рис. 3.115. Ladder-диаграмма управления позиционированием в относительных координатах (окончание)

Последовательность написания алгоритма. При кратковременном нажатии кнопки Вперед или Назад и наличии сигнала разрешения выполняются следующие операции:

1. Включается и встает на самопитание соответствующий сигнал цикла (M10 или M11).
2. Общий сигнал Цикл (M12) устанавливает параметры позиционирования:

   CR5 = H1010 – «Имп + Напр», позиционирование вперед или
   CR5 = H1210 – «Имп + Напр», позиционирование назад;
   CR21, CR22 = K200 – время разгона и замедления;
   CR23 = K1600 – позиция позиционирования;
   CR25 = K3000 – скорость движения при позиционировании.
3. Инверсный сигнал цикла снимает обнуление скорости (CR25).
4. Формируется виртуальный M13 и аппаратный Y22 сигналы деблокировки привода. Здесь – инверсные, так сделано в данном типе привода!
5. Снимается сигнал останова (CR31/b1).

6. Через выдержку времени после деблокировки привода формируются параметры пуска позиционирования CR31 = H180, где: b1 = 0 стоп, b7 = 1 относительные координаты, b8 = 1 управление через регистр CR32. В регистре CR32, по умолчанию, установлен бит пуска позиционирования b0 = 1. Если этот бит был сброшен, то его следует установить.

7. В процессе движения отслеживается и дешифрируется текущее состояние следующих сигналов:

CR38/b0 – ось движется и
CR38/b6 – позиция достигнута.

При выполнении условий:
движение было задано (M19 = 1);
движение прекращено (M17 = 0);
позиция достигнута (M18 = 1);
формируется сигнал M20 сброса цикла позиционирования.

8. Алгоритм приходит в исходное состояние, можно повторять цикл.

Внешнее управление блоком позиционирования

Для внешнего управления электроавтоматикой от модуля позиционирования предусмотрены следующие сигналы:

– пять дискретных входов (=24 В любой полярности):
Start – Пуск движения;
Stop – Стоп движения;
Dog – конечный выключатель нуля;
LSP – ограничение движения вперед;
LSN – ограничение движения назад;
– дифференциальные импульсные TTL-сигналы:
±A – фаза A;
±B – фаза B;
– PG0 – нуль-метка.

Для активизации режима необходимо установить следующие параметры:
(CR31/b8 = 0) – запрет управления от регистра CR32;
(CR32/b0 = 1) – сохранить значение бита по умолчанию, а также характеризовать работу конечных выключателей:
(CR5/b6) – тип конечного выключателя ограничения перемещения LSP;
(CR5/ b7) – тип конечного выключателя ограничения перемещения LSN;
(CR5/b9, b10) – принцип работы конечного выключателя нуля DOG.

Остальные параметры устанавливаются в зависимости от решаемой задачи.

Внешнее управление позволяет работать в наладочном режиме, режиме выхода в ноль и в режиме позиционирования.

Технически можно организовать три способа выхода в ноль:

1. Классический, с грубым датчиком DOG, TTL – датчиком положения и сигналом нуль-метки (рис. 3.116);
2. Только по грубому датчику DOG с последующей корректировкой положения при помощи параметра смещения;
3. Принудительным обнулением в любой точке абсолютных координат.

Выход в ноль только по кулачку DOG

В этом случае датчик положения (A, B, Z) к модулю контроллера не подключается.

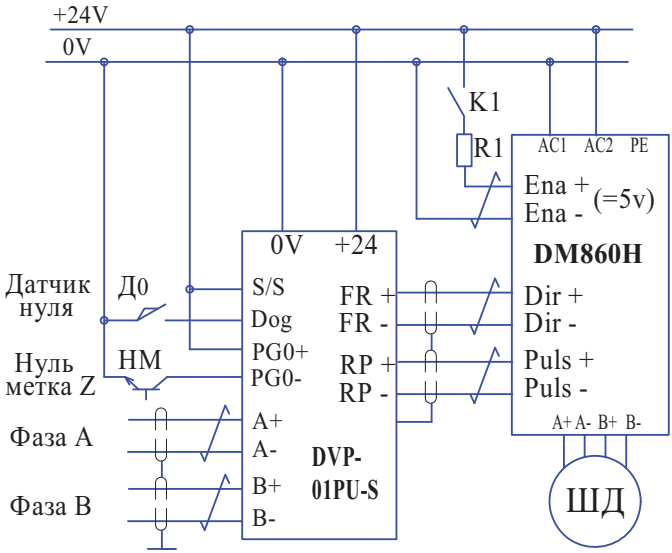


Рис. 3.116. Схема подключения сигналов при выходе в ноль

Диаграмма последовательности выхода в ноль по одному грубому датчику приведена на рис. 3.117, а алгоритм работы на рис. 3.118.

Обеспечивается следующая последовательность работы:

– установка режима выхода в ноль и его параметров;
– пуск цикла, движение на быстром ходу;
– замедление по переднему фронту конечного выключателя нуля;
– останов по заднему фронту выключателя нуля с учетом заданного в параметрах смещения;

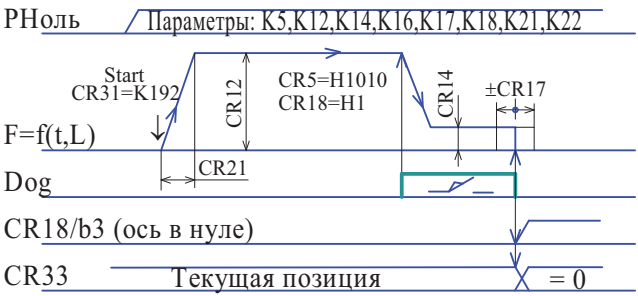


Рис. 3.117. Диаграмма выхода в ноль только по датчику DOG

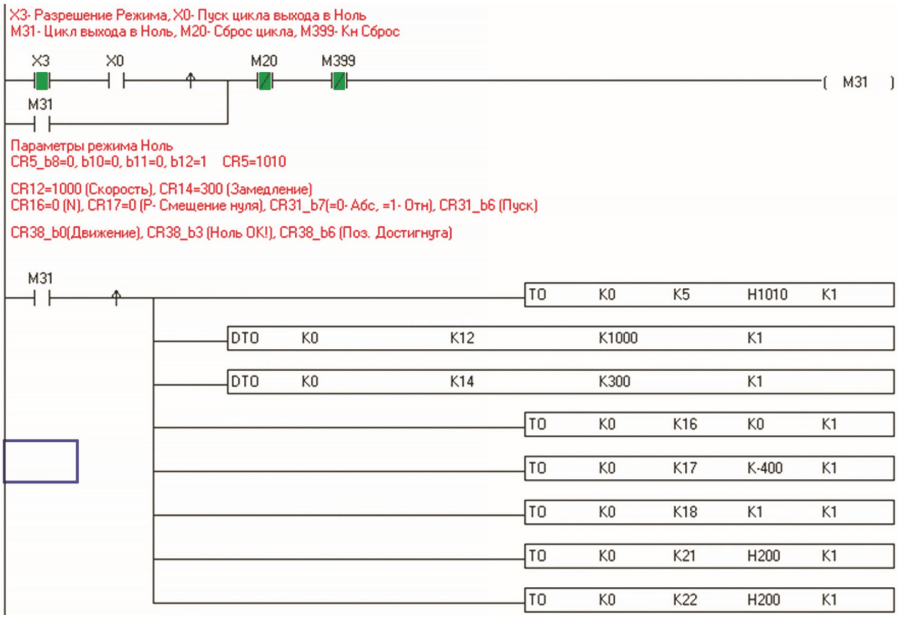


Рис. 3.118, а. Ladder-диаграмма выхода в ноль только по датчику DOG (запуск цикла и установка параметров)

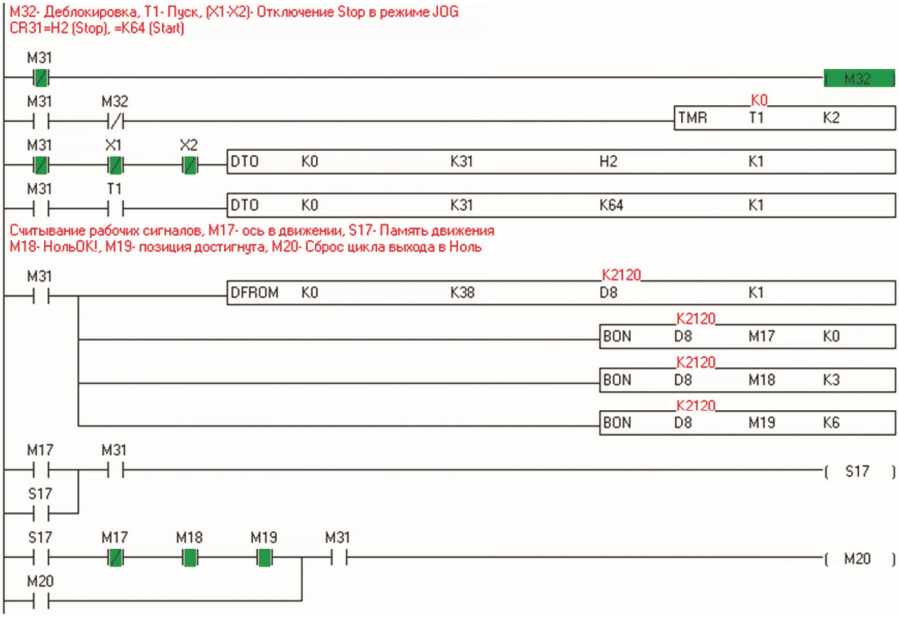


Рис. 3.118, б. Ladder-диаграмма выхода в ноль только по датчику DOG (старт движения, контроль процесса, сброс цикла)

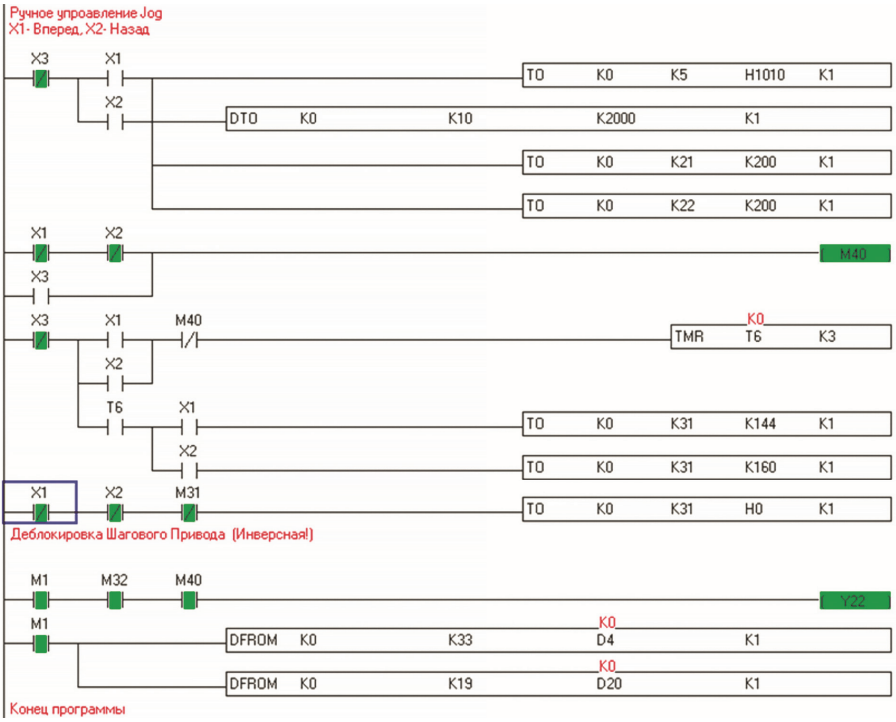


Рис. 3.118, в. Ladder-диаграмма выхода в ноль только по датчику DOG (управление в Jog для проверки)

Основные параметры:

(CR5 = 1010) – «Имп. + Напр», Направление Вперед, Dog на замыкание, линейное ускорение;
CR18 = 1 – Режим приоритета;
CR12 – Скорость выхода в ноль;
CR21, 22 – Время разгона и торможения;
CR14 – Скорость замедления при подходе к нулю;
CR17 – Смещение нулевой точки;
CR31/b6 – Старт позиционирования.

Основные операнды:

X0 – кнопка пуска цикла выхода в ноль;
M31 – цикл выхода в ноль;
M32 – виртуальная деблокировка (инверсная);
M17 – ось в движении;
S17 – память задания движения;

M18 – есть выход в ноль;
M19 – позиция достигнута;
M20 – сброс цикла.

На рис. 3.118, *в* приведен алгоритм ручного управления осью после выхода в ноль. Здесь он преследует следующие цели:

– посмотреть, как изменяется позиция оси после выхода в ноль;
– сделать повторный выход с разных позиций;
– показать, какие взаимные блокировки необходимо сделать для корректной работы в обоих режимах;
– показать принцип логического объединения общих сигналов.

Установка Нуля принудительным сбросом в нужной позиции в JOG

Это самый простой и во многих применениях эффективный способ определения исходной точки отсчета. Например, при автоматизации различного рода пил, всегда сначала подрезается торец исходной заготовки, будь то круглые стержни или плоский лист. Позиция заготовки после подрезки и принимается за точку отсчета при дальнейшем отрезании нужного числа и размера заготовок. Если заготовка уже подрезана, выполняется операция ее предварительной установки и зажима в исходной позиции, используя измерительные приборы. При этом есть и очевидный плюс, не нужно делать лишнюю подрезку.

Рабочая программа (рис. 3.119) состоит из следующих условных блоков:

– задание трех режимов работы: Jog (M10), обнуление (M11) и позиционирование (M12);
– перемещение заготовки в наладочном режиме от кнопок в позицию нуля:
  M13 – цикл Jog;
  M14 – виртуальная деблокировка (Y22);
  M15, M15 – команды движение вперед и назад;
  M17 – останов при отпускании кнопки и наезде на ограничительные конечники;
  M1 – сброс цикла Jog. Процедура зажима заготовки здесь не рассматривается;
– принудительное обнуление позиции (CR33 = 0) при установке режима;
– запись в преднаборе требуемой позиции относительно нуля в буферную память (здесь D24). В программе предусмотрено 5 позиций;
– цикл позиционирования:
  M20 – цикл позиционирования;
  M21 – виртуальная деблокировка (Y22);
  M22, S22, M23 – контроль позиционирования;
  M22, S22, M23 – контроль позиционирования;
  M24 – сброс цикла позиционирования.

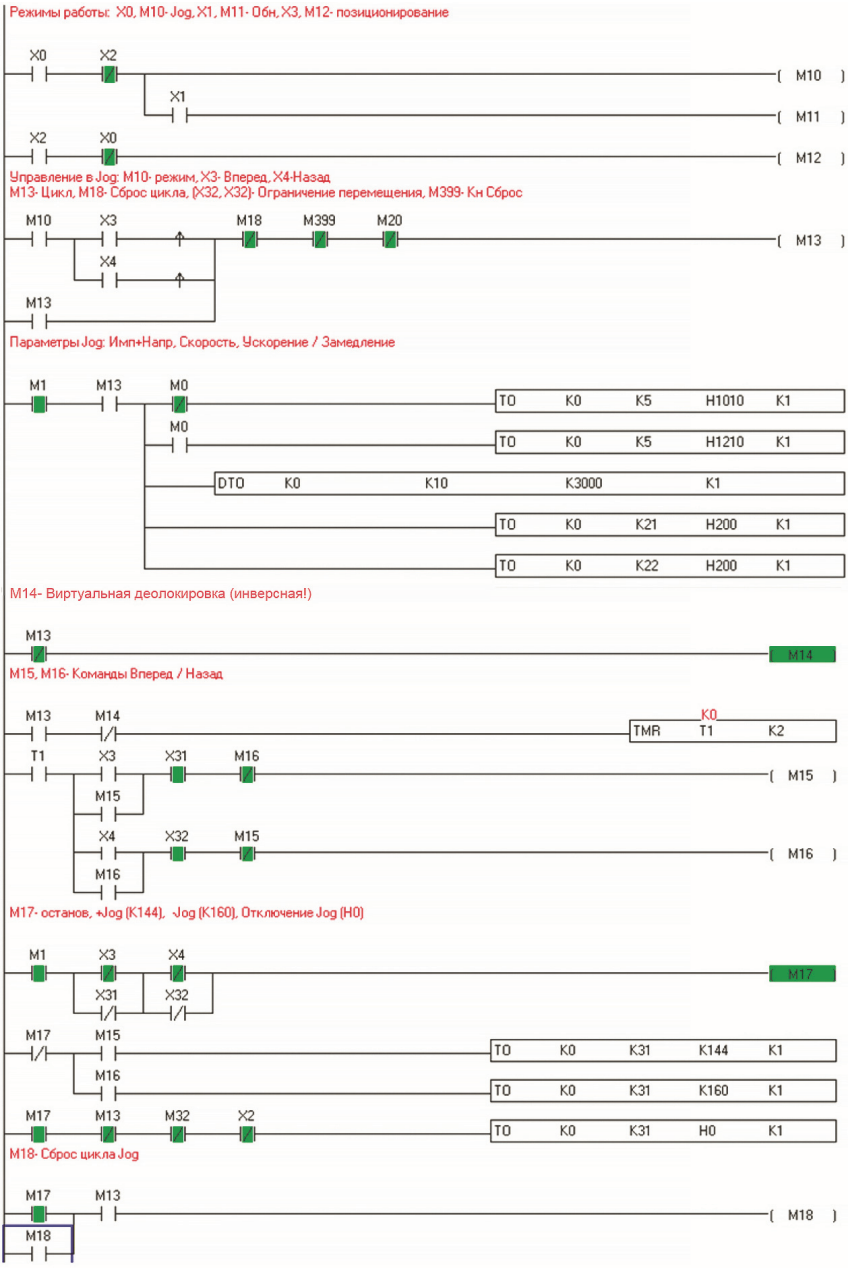


Рис. 3.119. Алгоритм установки ноля путем обнуления величины позиции (начало)

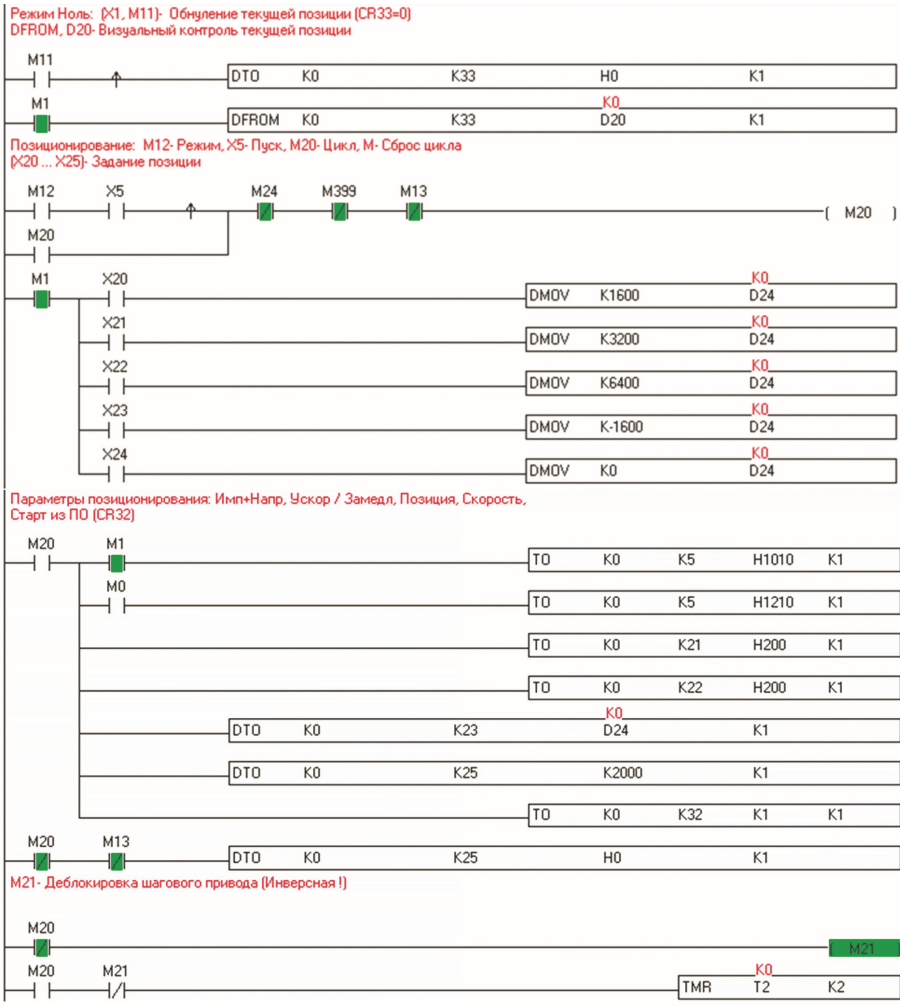


Рис. 3.119. Алгоритм установки ноля путем обнуления величины позиции (продолжение)

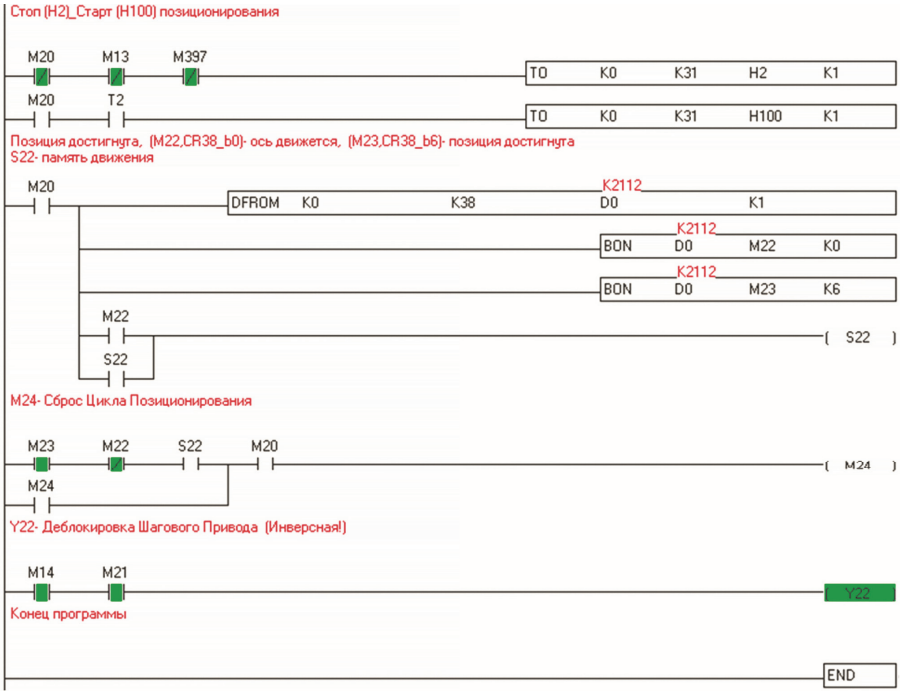


Рис. 3.119. Алгоритм установки ноля путем обнуления величины позиции (окончание)

3.10.3. Боковой зажим листа «Клыком»

Боковой зажим листа «клыком» осуществляется его подводом к торцу листа «на упор» двигателем постоянного тока, управляемого специальной платой производства Китая (рис. 3.120). При отжиме клык также доходит до упора. Объективные датчики контроля зажима и отжима на механизме отсутствуют, поэтому было принято решение контролировать выполнение этих операций по току двигателя.

На рис. 3.121 приведены теоретические циклограммы при нормальной работе механизма и при аварийной ситуации. Операции зажима и отжима в данном примере осуществляются от кнопок в ручном режиме. Оператор нажимает соответствующую кнопку и удерживает ее до окончания операции, наблюдая за процессом визуально и по экрану панели оператора.

При *нормальной* работе таймер Т1 отсекает контроль и срабатывание защиты на время пуска двигателя. При срабатывании реле тока КА(I) на упоре:

– формируется и запоминается сигнал контроля зажима (отжима) Кз/о, отключающий работу двигателя;

– формируются и запоминаются сигналы контроля Заж(Ок!) или Отж(Ок!) завершения операции. Кнопка может быть отпущена.

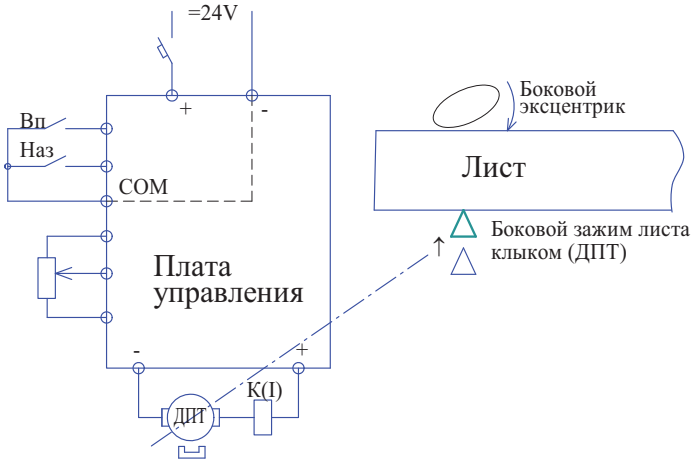


Рис. 3.120. Принципиальная схема управления боковым зажимом листа «Клыком»

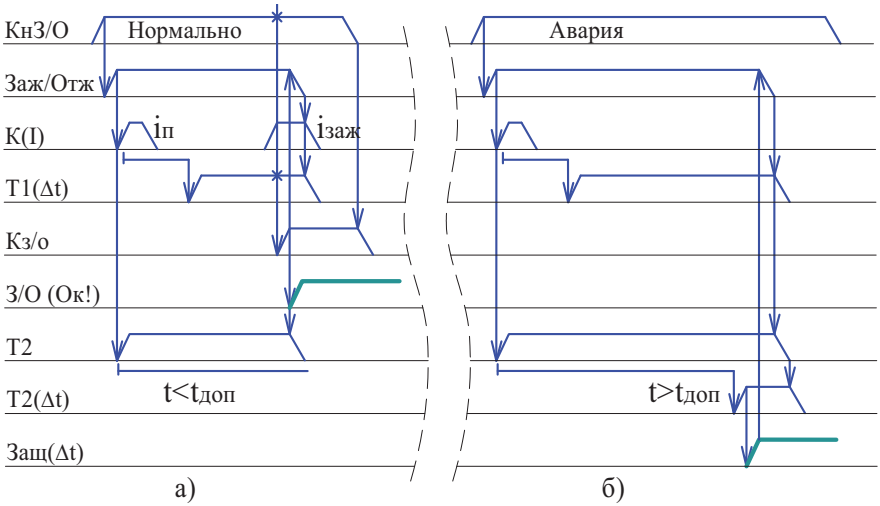


Рис. 3.121. Циклограмма работы бокового зажима листа:
а – штатно; *б* – авария

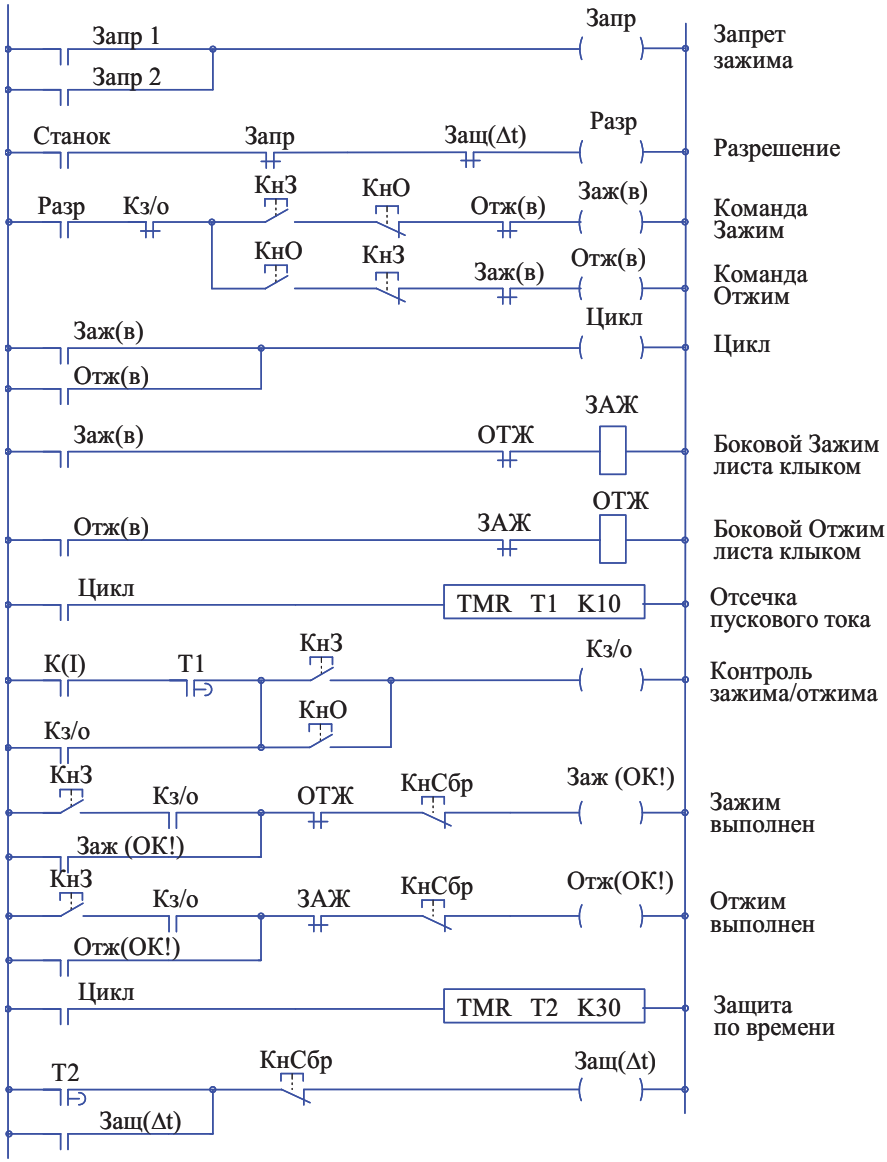


Рис. 3.122. РКС-алгоритм бокового зажима листа клыком в наладочном режиме

При *аварийной* ситуации, если по каким-либо причинам двигатель не отключится в течение контрольного времени (таймер Т2), срабатывает и запоминается защита по времени работы Защ(Δ*t*). Двигатель отключается, снимается

разрешение на дальнейшую работу, на экран выводится сообщение об аварийной ситуации.

РКС-алгоритм управления боковым зажимом клыком в наладочном режиме от кнопок приведен на рис. 3.122.

В сложных циклах автоматической подачи и отрезания листа на нужный размер данный алгоритм следует взять за основу, заменив кнопочные сигналы управления на цикловые команды и ввести необходимые запреты и блокировки. Например, нельзя выполнять операцию отжима (зажима), если включен автоматический режим, выполняется цикл подачи листа, идет операция резания и т. д.

3.10.4. Подвод бокового эксцентрика зажима листа

Боковой эксцентрик зажимает лист с противоположной стороны от клыка. Ширина листа разная, поэтому перед зажимом эксцентрик следует подвести к заднему торцу листа (рис. 3.123). Эта операция производится оператором вручную при подготовке пилы к работе. Для подвода используется аналогичный провод постоянного тока. Предусмотрены ограничительные конечные выключатели крайних положений механизма зажима. Контроль подвода – визуальный. По этой причине дополнительно осуществляется контроль от длительной перегрузки по току двигателя.

3.10.5. Боковой зажим листа эксцентриком

Зажим листа боковым эксцентриком осуществляется его поворотом при помощи пневматик (рис. 3.124).

Перед зажимом осуществляется контролируемый оператором подвод эксцентрика к торцу листа от привода постоянного тока (см. раздел 3.10.4).

Алгоритм рис. 3.124 осуществляет ручной зажим или отжим от кнопок, расположенных на панели оператора (см. рис. 3.111). В автоматических циклах его следует модернизировать, заменив кнопки на цикловые сигналы, и введя необходимые блокировки.

Контроль зажима (отжима) осуществляется по реле давления, при этом формируются контролирующие сигналы Заж(Ок!) и Отж(Ок!).

Предусмотрена защита от превышения контрольного времени зажима.

Три рассмотренных механизма зажима, а также зажимные эксцентрики плоскости листа, составляют единый зажимной узел. Всего таких узла три: на подвижном портале перемещения листа, неподвижном портале в зоне зажима листа при резании и на неподвижном портале в зоне зажима карты (заготовки). Таким образом, это достаточно сложная система зажимов, работающая по разным алгоритмам в зависимости от выполняемой операции, например, при подаче листа в зону торцевания, при резании листа, при подаче заготовки на требуемый

размер, при отрезании заготовки. Везде разная последовательность работы, однако общий алгоритм складывается из рассмотренных нами шаблонов.

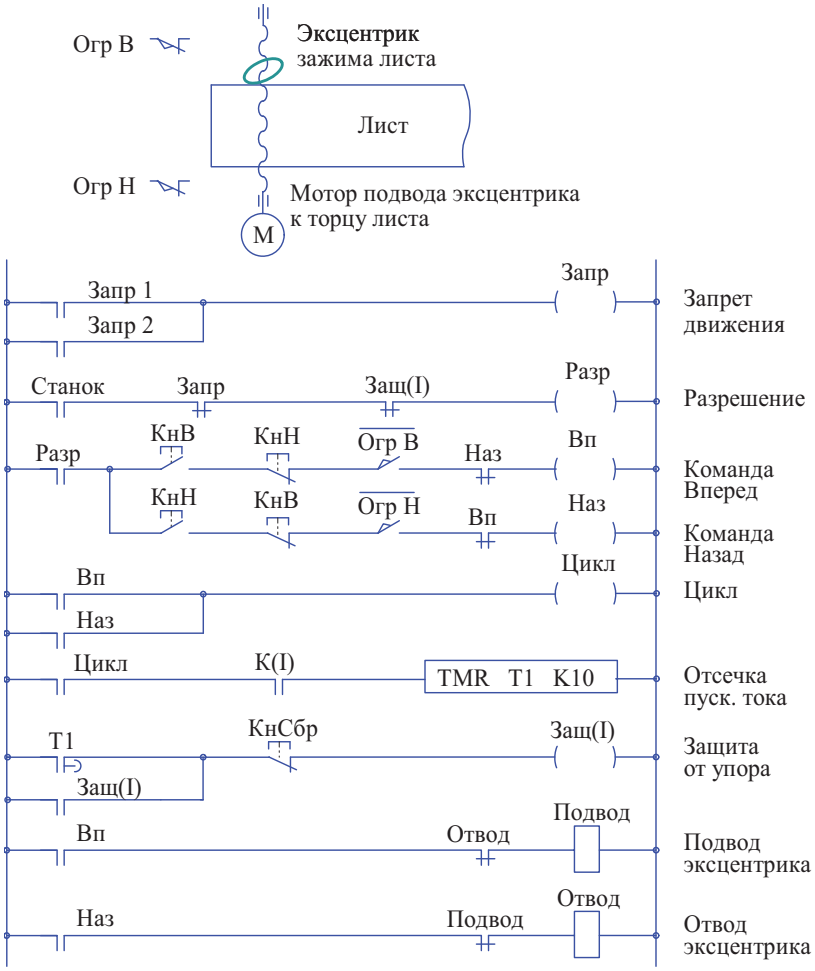


Рис. 3.123. Алгоритм подвода бокового эксцентрика к торцу листа

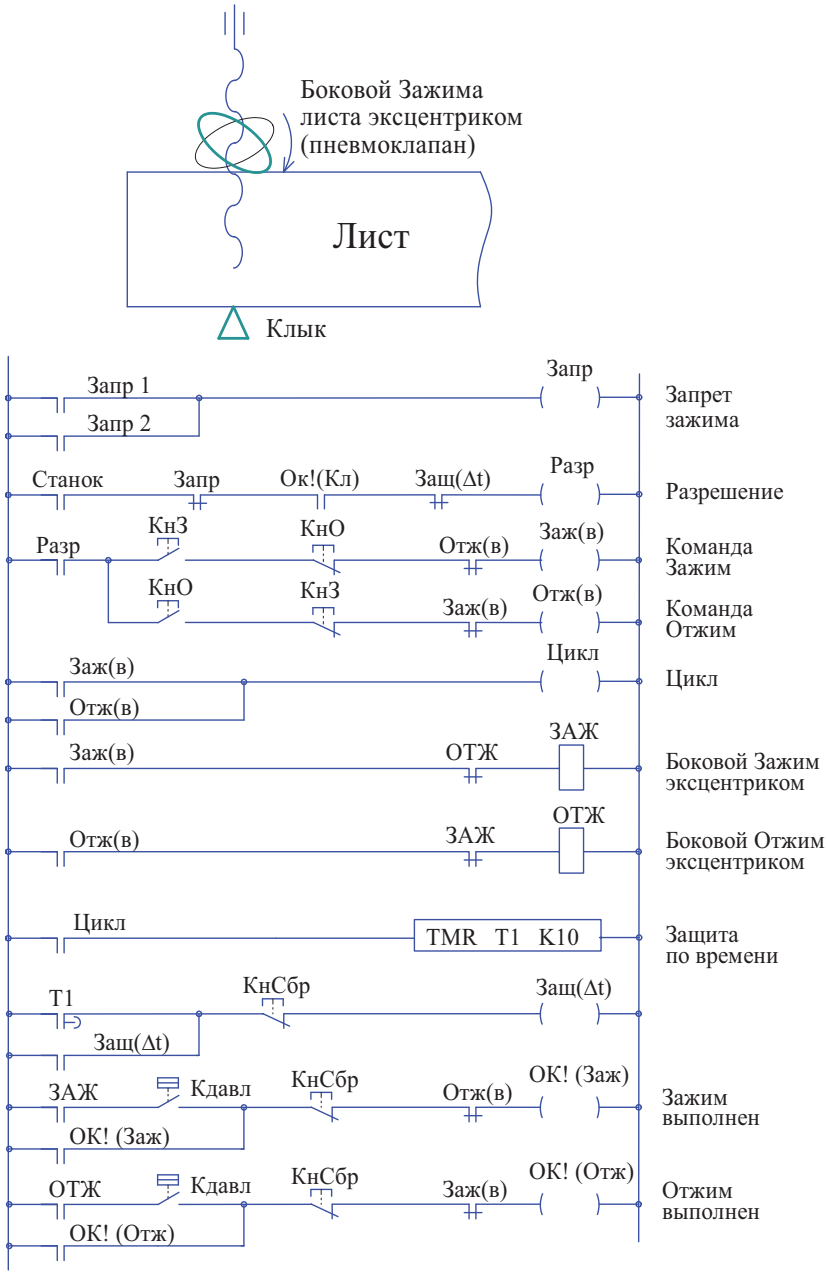


Рис. 3.124. Алгоритм зажима листа боковым эксцентриком

3.10.6. Расчеты для движения рамы при резании

При разрезании листа выполняется следующий цикл движения рамы с вращающейся пилой (рис. 3.125):

– перемещение пилы вниз на быстром ходу из верхнего исходного положения на расстояние Zбх до безопасной точки (запас Δ1) от поверхности распиливаемого листа;
– перемещение на рабочей подаче на расстояние Zрп при выполнении резания до полного выхода полотна пилы ниже листа (запас Δ2);
– подъем вверх на быстром ходу на расстояние (Zрп + Zбх), т. е. до крайней верхней точки при однократном резе или на расстояние Zрп при многократных повторяющихся резах.

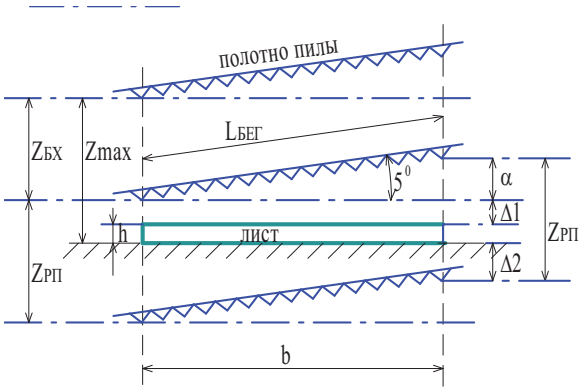


Рис. 3.125. Эскиз перемещения пилы при резании

Вращение полотна пилы осуществляется от частотного преобразователя, управляемого по аналоговому выходному каналу центрального процессора (см. раздел 3.10.1), а перемещение рамы с пилой от сервопривода, управляемого от модуля позиционирования в режиме «Импульсы + направление», аналогично рассмотренному выше управлению шаговым двигателем (см. раздел 3.10.2).

Чтобы организовать такой автоматический цикл, необходимо задать исходные данные (параметры) и на их основе произвести расчеты пути перемещения на быстром ходу и на рабочей подаче в зависимости от ширины листа.

Кроме того, необходимо обеспечить синхронное перемещение бегунка, поддерживающего полотно пилы при резании.

Исходные данные для расчетов.

Константы:

Zmax – расстояние от нижней кромки листа до верхнего исходного положения рамы. Измеряется оператором при подготовке пилы к работе;

Δ1 = 5 мм – безопасное расстояние до листа;
Δ2 = 3 мм – запас при выходе пилы из листа;
угол наклона полотна пилы – 5 градусов.

Параметры:
h(мм) – толщина листа;
b(мм) – ширина листа;
V(мм/мин) – скорость вращения пилы;
Fрп(мм/мин) – скорость резания;
Fбх(мм/мин) – скорость быстрого хода.

Формулы для расчетов:
Перемещение на быстром ходу

$$Z\text{бх} = Z\text{max} - h - \Delta 1.$$

Перемещение на рабочей подаче

$$Z\text{рп} = \alpha + \Delta 1 + h + \Delta 2;$$

$$\alpha = b\text{л} \cdot \operatorname{tg} 5^\circ$$

где $\operatorname{tg} 5^\circ = 0{,}087488663 = 7/80.$

Расстояние перемещения бегунка

$$L\text{бег} = \frac{b\text{л}}{\cos 5^\circ},$$

где $\cos 5^\circ = 0{,}9962 = 255/256.$

Время резания

$$t\text{рез} = \frac{Z\text{рп}}{F\text{рп}}.$$

Скорость перемещения бегунка

$$V\text{бег} = \frac{L\text{бег}}{t\text{реза}}$$

Пример алгоритма вычислений приведен ниже на рис. 3.126.

Примечание. *Для того чтобы упростить расчеты и избежать вычисления инструкциями с плавающей запятой, значения тангенса и косинуса угла наклона пилы в 5 градусов были взяты из таблиц Брадиса и округлены до целых дробей.*

Перед выполнением расчетов константы и параметры записываются в соответствующие D-слова через окно параметров панели оператора.

Операнды M446–M449 это ячейки активизации расчетов в разных автоматических циклах общего проекта.

Аналогичные расчеты, но с другими размерами, производятся также при разрезании карты на готовые заготовки.

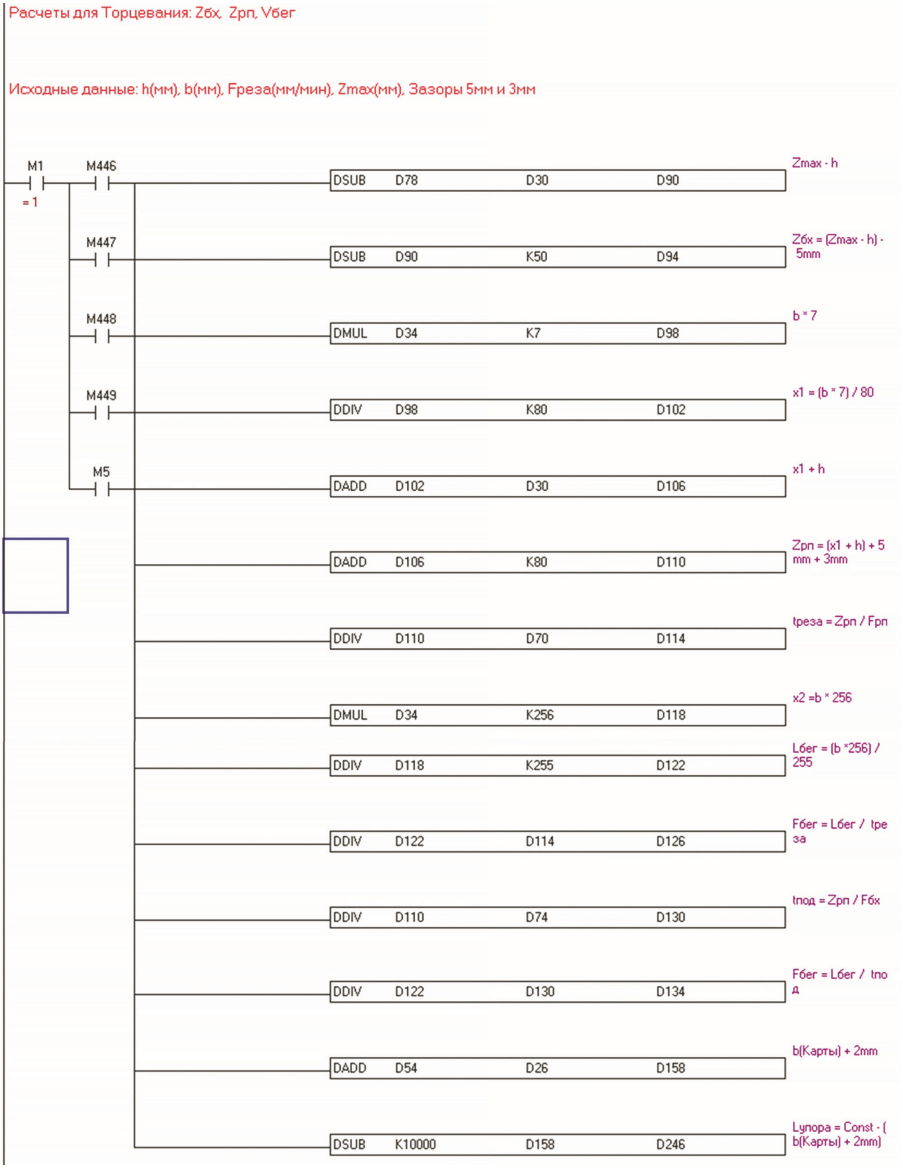


Рис. 3.126. Алгоритм арифметических вычислений (начало)

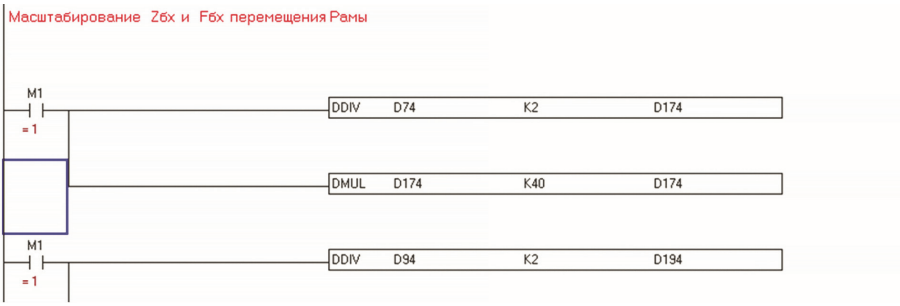


Рис. 3.126. Алгоритм арифметических вычислений (окончание)

3.10.7. Расчеты и алгоритм определения числа резов заготовки

После отрезания от листа заготовки нужного размера (L * b) в левой стороне пилы (см. рис. 3.11), заготовка снимается краном, переворачивается на 90 градусов и устанавливается на правой стороне пилы (см. рис. 3.12). После подрезки торца путем деления длины заготовки (L) на ширину детали заготовки (b) определяется число деталей (N), которые можно из нее сделать, а следовательно и число резов (рис. 3.127) в автоматическом цикле.

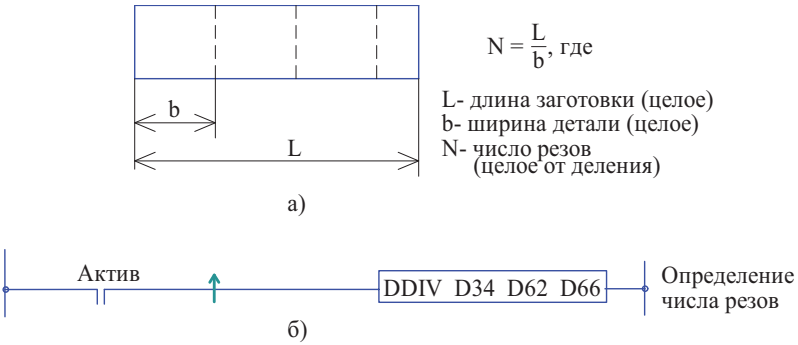


Рис. 3.127. Эскиз карты (а) и инструкция (б) определения числа резов

Здесь: D62 – длина заготовки L;
D34 – ширина детали b;
D66 – число резов N. Дробная часть от деления игнорируется.

На рис. 3.128 и 3.129, соответственно, приведены циклограмма и алгоритм счета числа отрезаемых деталей. Общий алгоритм отрезания детали, называемый «Шинкованием», включает операции отжима заготовки, подачи на размер детали с учетом ширины пилы, зажима, вложенного цикла отрезания.

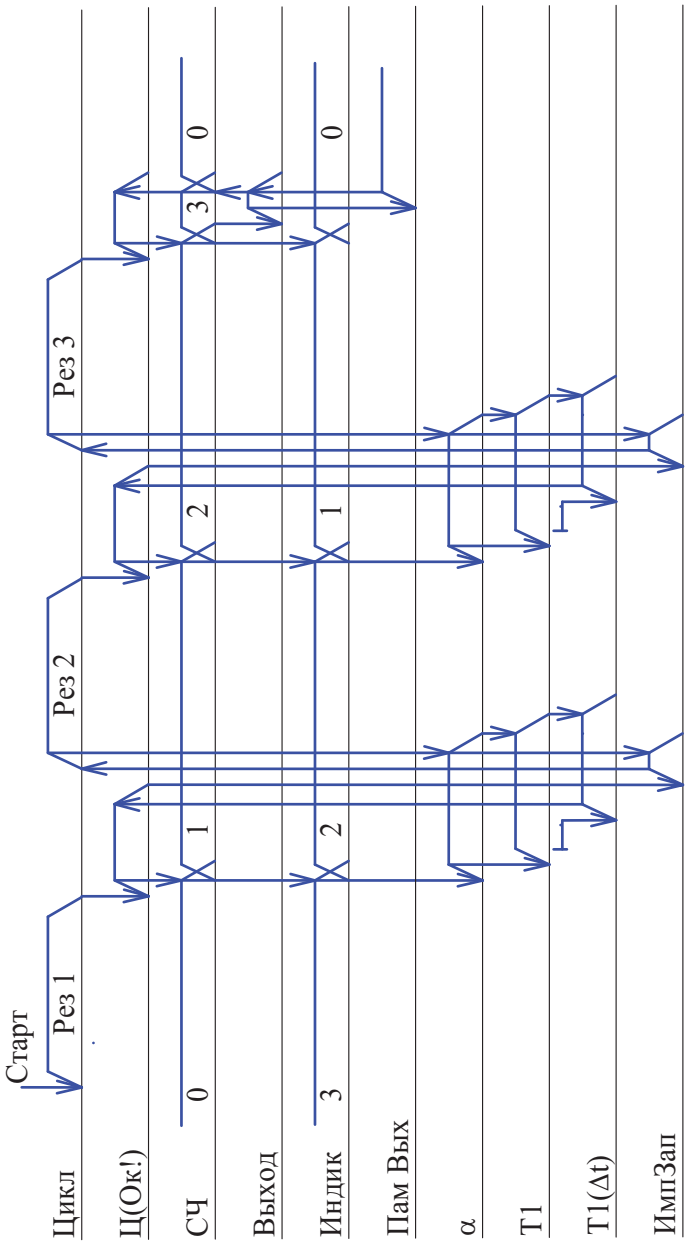


Рис. 3.128. Циклограмма счета отрезаемых деталей

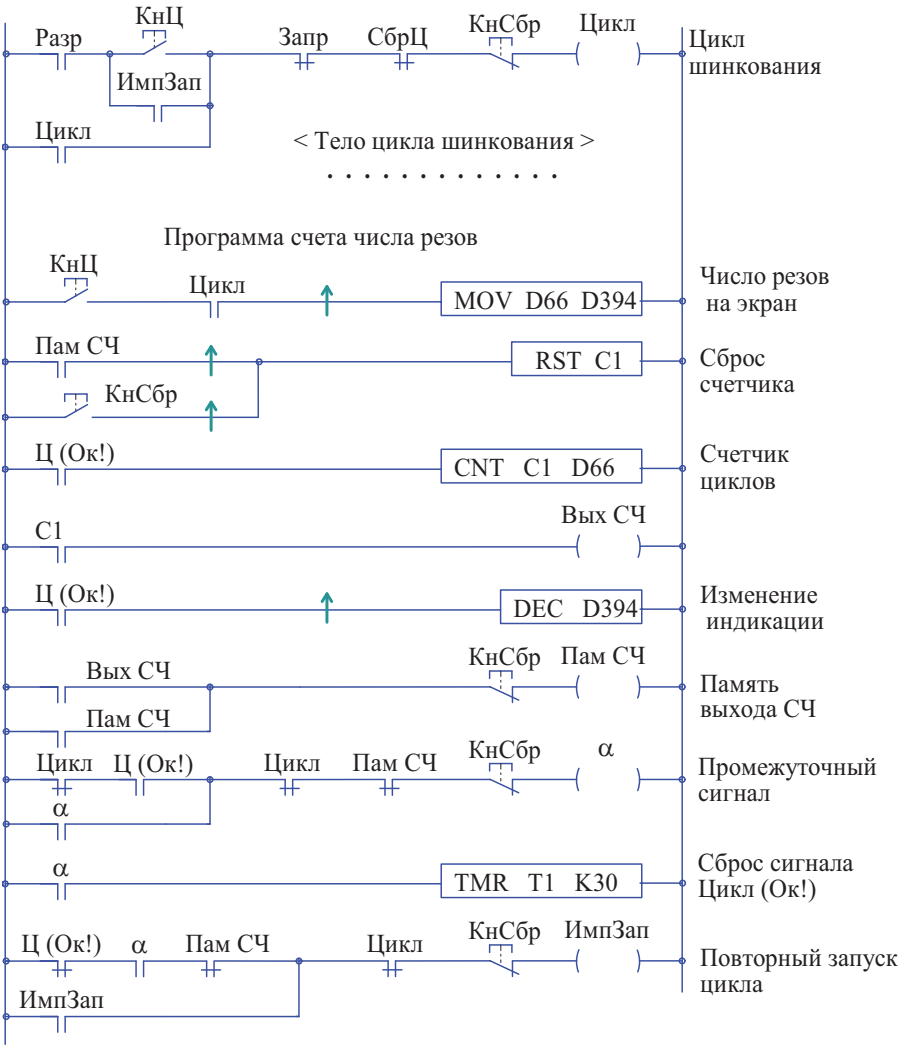


Рис. 3.129. Алгоритм счета числа резов

По окончании каждого вложенного цикла осуществляется счет отрезанных деталей с указанием на панели оператора числа оставшихся резов, и если их число меньше расчетного значения N, то осуществляется повторный запуск. После окончания последнего реза цикл шинкования сбрасывается, рама отводится в верхнее положение, вращение ленты пилы выключается. Используется

стандартный CNT-счетчик. После отрезания последней детали производится их съем и подготовка пилы к отрезанию и шинкованию новой карта.

Синтез алгоритма ясен из анализа циклограммы.

3.11. Установка параметров связи через электроавтоматику

Система параметров контроллера и синтаксис языка позволяют через PLC-программу автоматики устанавливать параметры связи между компьютером и контроллером. Система параметров приведена в табл. 3.11. Жирно выделены параметры, необходимые для установки скорости передачи данных от компьютера к контроллеру.

Таблица 3.11

Параметры установки связи

Группа параметров	Номер порта			Назначение параметра
	COM1	**COM2**	COM3	
Protocol Setting	M1138	**M1120**	M1136	Retain communication setting
	M1139	**M1143**	M1320	ASCij/RTU mode selection
	D1036	**D1120**	D1109	Communication protocol
	D1121	**D1121**	D1255	PLC communication address
Sending Request	–	M1161	–	8/16 bit Mode Selection
	–	M1121	–	Indicate Transmission Status
	M1312	M1122	M13216	Sending Request
	–	M1126	–	Set STX/ETX
	–	M1130	–	Set STX/ETX
	–	D1124	–	Definition of STX (RS)
	–	D1125	–	Definition of ETX1
	–	D1126	–	Definition of ETX2
	D1249	**D1129**	D1252	Communication time out Setting
	–	D1122	–	Residual № of words transmission
	–	D1256	–	Store the sent data of MODRW instruction
	–	……..	–	
	–	D1295	–	
	–	D1089	–	Store the sent data of MODRD, FWD, REV, Stop, RDST,RSTEF instruction
	–	……..	–	
	–	D1099	–	

Окончание таблицы 3.11

Группа параметров	Номер порта			Назначение параметра
	COM1	**COM2**	COM3	
Data Setting	M1313	M1124	M1317	Data receiving Ready
	–	M1125	–	Communication ready status reset
	–	M1128	–	Transmitting/Receiving status
	–	D1123	–	Residual № of words receiving
	–	D1070	–	Store of feedback data of Modbus communication
	–	……..	–	
	–	D1085	–	
	D1167	D1168	D1169	Store of specific end word
Receiving Completed	M1314	M1123	M1318	Data Receiving Completed
	–	M1127	–	COM2 (RS–485) data Sending/Receiving/Converting completed
	–	M1131	–	ON When MODRD/MODRW data converting ASCij to Hex
	–	D1296	–	Store converting HEX data of MODRW instruction
	–	……..	–	
	–	D1311	–	
	–	D1050	–	Store converting HEX data of MODRD instruction
	–	……..	–	
	–	D1055	–	
Errors	M1315	–	M1319	Data receiving Error
	D1250	–	D1253	Communication Error code
	–	M1129	–	COM2 (RS485) receiving time out
	–	M1140	–	COM2 (RS485) /MODRD /MODWR /MODWR data receiving Error
	–	M1141	–	MODRD (WR,RW) parameter Error
	–	M1142	–	Data receiving Error of VFD–A handy instruction
	–	D1130	–	COM2(RS–485) Error code returning from MODBUS communication

В табл. 3.12 приведена расшифровка слова параметров **D1120** протокола связи (Communication Protocol).

Таблица 3.12

Расшифровка слова протокола связи

Код	Бит	Параметр	Уставки		
1	b0	Data length	= **0** (7 bits)	= 1 (8 bits)	
2	b1	Parity	= 0 Non	= 1 Odd	= **1 E**ven
4	b2	Bits	= 1	= 0	= **1**
8	b3	Stop bit	= 0 (1 bit)	= 1 (2 bits)	
1	b4	Bod Rate	H1 – 110bps	H5 – 1200bps	H9 – 19200bps
2	b5		H2 – 150bps	H6 – 2400bps	HA – 38400bps
4	b6		H3 – 300bps	H7 – 4800bps	HB – 57600bps
8	b7		H4 – 600bps	H8 – 9600bps	**HC** – 115200bps
1	b8	STX	= 0 (None)	= 1 (D1124)	
2	b9	ETX1	= 0 (None)	= 1 (D1125)	
4	b10	ETX2	= 0 (None)	= 1 (D1126)	
8	b11-b15	Not available			

Устанавливаем следующие параметры:

HC6 → 115200, 7, E, 1

M1143 = 0 (ASCij Mode)

M1129= 100ms

D1121=1

и активизируем их в начальной части PLC-программы рис. 3.130.

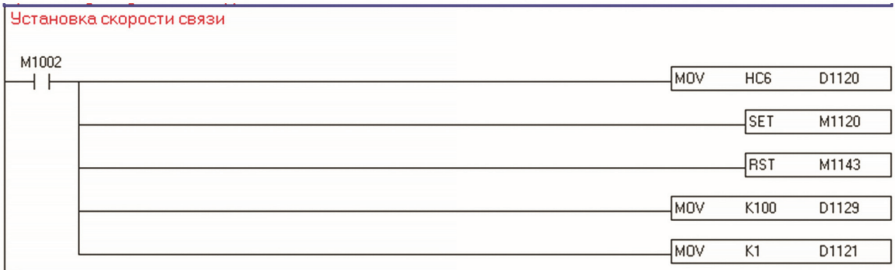


Рис. 3.130. Установка параметров связи между ПК и PLC

Для облегчения поиска рассмотренных в книге инструкций, ниже, в разделе 3.12 приводится их полный перечень в порядке рассмотрения.

3.12. Последовательность изложения Функциональных Инструкций

MPS, MPD, MPP – разветвленные релейные цепи;
MC, MCR – контроль общей цепи;
PLS, PLF – формирователи тактов по переднему и заднему фронтам;
NOP – резервирование места (нет операции);
KnX(Y, M) – адресация блоков;
MOV – стандартная пересылка;
BMOV – пересылка блока данных;
FMOV – пересылка данных в несколько адресов;
SMOV – пересылка данных со смещением;
CML – передача данных с инвертированием;
XCH – обмен данными;
SWAP – внутренний обмен данными;
TMR – стандартный таймер;
TTMR – обучаемый таймер;
STMR – специальный таймер;
CNT – стандартный счетчик;
DCNT – реверсивный счетчик;
SET, RST – установка битов / сброс;
ZRST – групповой сброс;
CMP – стандартное сравнение;
ZCP – зонное сравнение
LD=(=, <, >) – сравнение битовых операндов;
BCD, BIN – преобразование (BIN → BCD), (BCD → BIN);
GREY, GBIN – преобразование (BIN → GRY), (GRY → BIN);
INC, DEC – инкремент / декремент;
WAND, WOR, WXOR – логика с битами слов (D – двойные слова);
ALT – счетный Т-триггер;
GPWM – программируемый генератор импульсов;
FOR, NEXT – цикловая инструкция;
DECO – дешифратор;
ENCO – шифратор;
ROR, ROL – стандартные кольцевые сдвиговые регистры;
RCR, RCL – специальные кольцевые сдвиговые регистры;
SFTR, SFTL – групповые выталкивающие сдвиговые регистры;
SFWR, SFRD – запись / считывание стека;
ADD – арифметическое сложение;
SUB – арифметическое вычитание;
MUL – арифметическое умножение;

DIV – арифметическое деление;
ABS – взятие абсолютного значения;
SQR – вычисление квадратного корня;
SUM – подсчет числа единиц в слове;
MEAN – расчет среднего значения нескольких операндов;
RAND – генерирование случайных чисел;
VRRD – чтение положения движка встроенного потенциометра;
VRSC – чтение зоны положения движка потенциометра;
CJ – условный переход;
CALL – вызов подпрограммы;
DSW – считывание информации с DIP-переключателей;
SEGM – управление сегментным индикатором
MAND, MOR, MXOR, MXNR, MCMP, MINV, MBS, MBR, MBC, MBRD – работа с матрицами;
TWR – запись времени и даты в память;
TRD – чтение времени и даты из памяти;
TCMP – сравнение текущего времени с эталоном;
TZCP – зонное сравнение текущего времени;
TADD – сложение временных уставок;
TSUB – вычитание временных уставок;
PID – ПИД-регулятор;
FROM, TO – чтение / запись параметров;
BON – чтение бита параметра;
EADD, ESUB, EMUL, EDIV и др. – инструкции работы с числами с плавающей запятой;
RAD – перевод градусов в радианы;
DEG – перевод радиан в градусы;
INT – преобразование числа с плавающей запятой в целое двоичное число;
FLT – преобразование целого числа в число с плавающей запятой и наоборот;
MOVR – пересылка чисел в формате с плавающей запятой;
SIN – вычисление синуса;
ASIN – вычисление арксинуса;
DPLSY, DPLSR и др. – высокоскоростные инструкции;
DABSR, DDRVI, DDRVA и др. – инструкции позиционирования.

ГЛАВА 4. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ SFC (STL)

4.1. Общие сведения

Как было сказано в главе 1, подобные языки относятся к языкам последовательного графического представления этапов работы механизма, предваряемых условиями разрешения выполнения каждого этапа. Синтаксис графических языков разных фирм, при общей идеологии построения, довольно сильно отличается и всегда следует придерживаться сопроводительной технической документации. Отличается и расшифровка аббревиатур.

В данной главе рассматривается синтаксис языка SFC контроллера типа SX2 фирмы Дельта. SFC – это Step Function Control или Sequential Function Chart.

Программа (рис. 4.1) состоит из *начального* шага **S0** последовательности семи *шагов (этапов)* **S1–S7**, описывающих активные действия исполнительных органов (включить шпиндель, включить охлаждение, повернуть манипулятор на 90 градусов и т. д.) и семи *переходов* **1–7**, описывающих условия разрешения выполнения этапа (инструмент зажат, манипулятор в исходном положении, шпиндель выключен и т. д.). Таким образом, она представляет собой *граф* последовательных шагов и переходов, синтезированных с учетом правил синтаксиса языка.

Перед программой SFC и по ее завершению могут находиться классические блоки Ladder-диаграмм (Lad-0 и Lad-1).

Окончание программы кодируется командой **RET**.

Шаги программы обозначаются буквой **S** (Step) и кодируются в диапазоне S0–S1023. Синтаксис языка позволяет строить следующие структуры программ:

- простая линейная последовательность шагов (операций), как показано на рис. 4.1;
- простая линейная последовательность шагов с наличием разного рода условных переходов из одного места программы в другое (рис. 4.16, 4.19);
- линейные разветвленные структуры с разветвлениями и сходимостями по логическим функциям И, ИЛИ (см. рис. 4.12, 4.13);
- параллельные независимые структуры;
- параллельные структуры с возможностью перехода из одной структуры в другую (см. рис. 4.22).

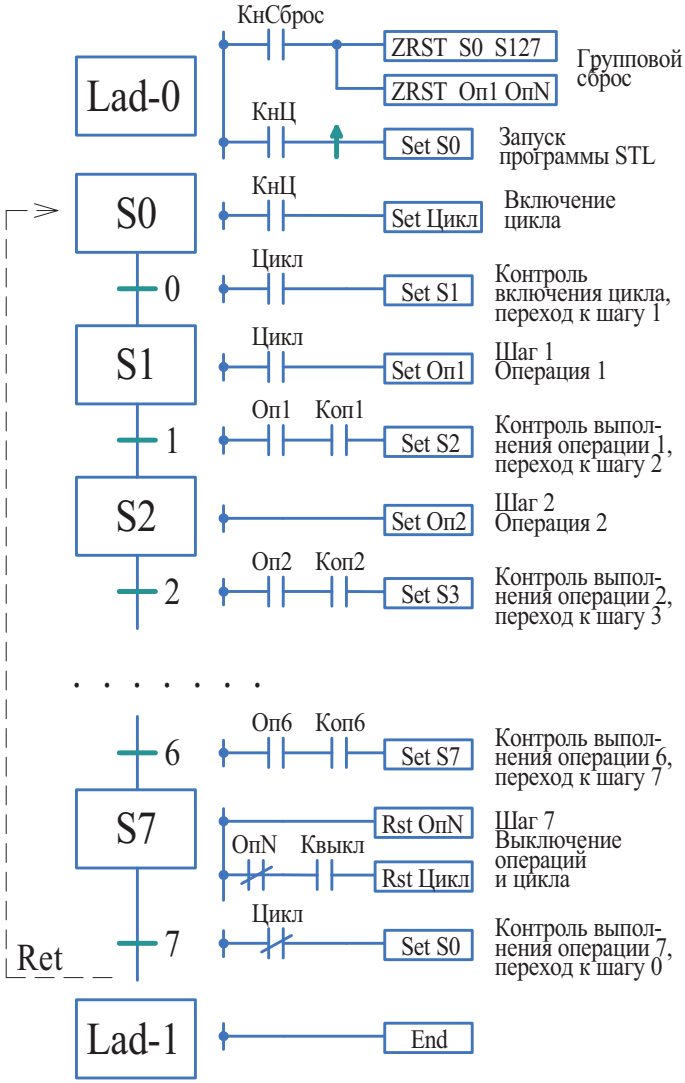


Рис. 4.1. Блок-схема цикла в формате SFC

В зависимости от структуры построения программы существуют следующие правила кодирования шагов:

– для начального шага программ отведены коды S0–S9, таким образом можно сформировать 10 независимых параллельных программ. Для простейшей одиночной линейной структуры с начальным шагом S0

допускается кодировать остальные шаги, начиная с S1 и т. д. по порядку;

– если программируется несколько параллельных программ, то рабочие шаги следует кодировать, начиная с S10;
– если в программах используются команды прерывания IST, то для них резервируются коды S10–S19, а рабочие шаги начинаются с S20.

Для того чтобы никогда не было никаких накладок, следует принять за правило:

– коды S0–S9 использовать только для начальных шагов;
– коды S10–S19 использовать для прерываний;
– для кодирования рабочих шагов использовать S20–S1023.

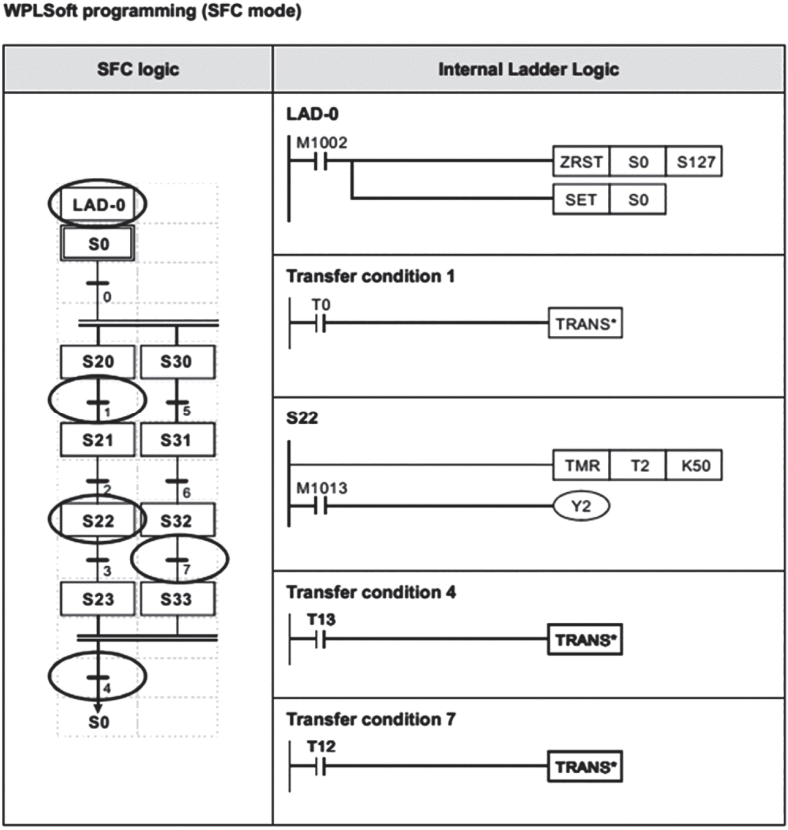


Рис. 4.2. Пример SFC-программы

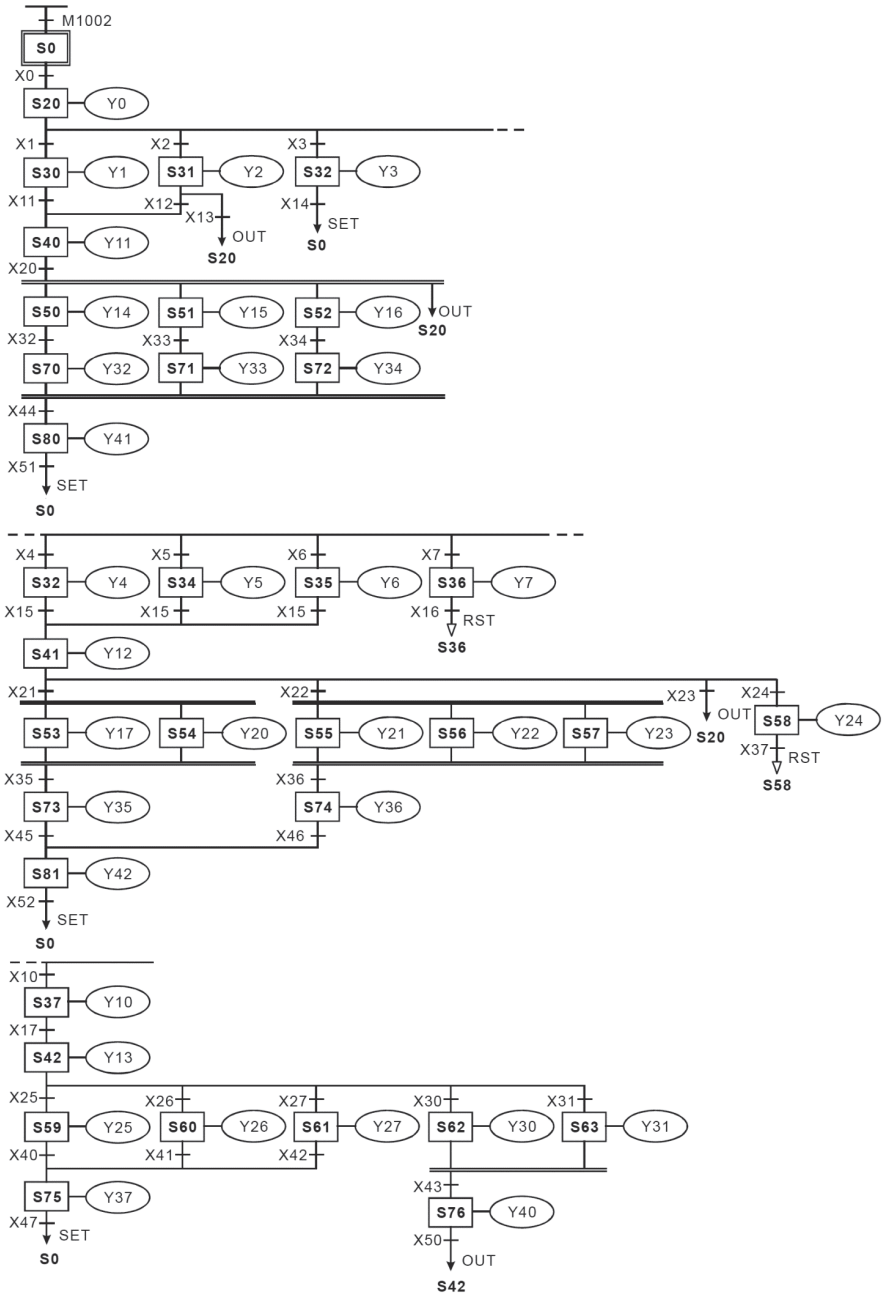


Рис. 4.3. Пример SFC-программы

Для сброса шаговых операндов используется инструкция ZRST.

Еще важное замечание. Если язык SFC не используется, то битовую область операнда S можно использовать как дополнительную область памяти пользователя, т. е. параллельно с зоной памяти операнда M.

Следует сразу сказать, что, несмотря на рекламируемую простоту, набор, отладка и чтение таких программ довольно сложная процедура и особого энтузиазма не вызывают. На рис. 4.2 и 4.3 приведены примеры программ из документации фирмы. Кажущаяся структурная простота обманчива, набор сложен, при отладке нужно постоянно открывать текущие шаги и переходы. При первом обращении к документации возникает много непонятных вопросов.

На рис. 4.4–4.7 приведены фрагменты рабочих программ, сделанных в среде WPL Soft. Полагаю, что начинающему пользователю в них мало что будет понятно.

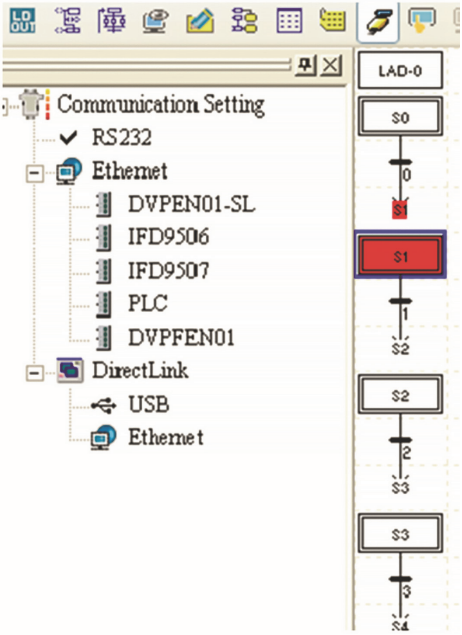
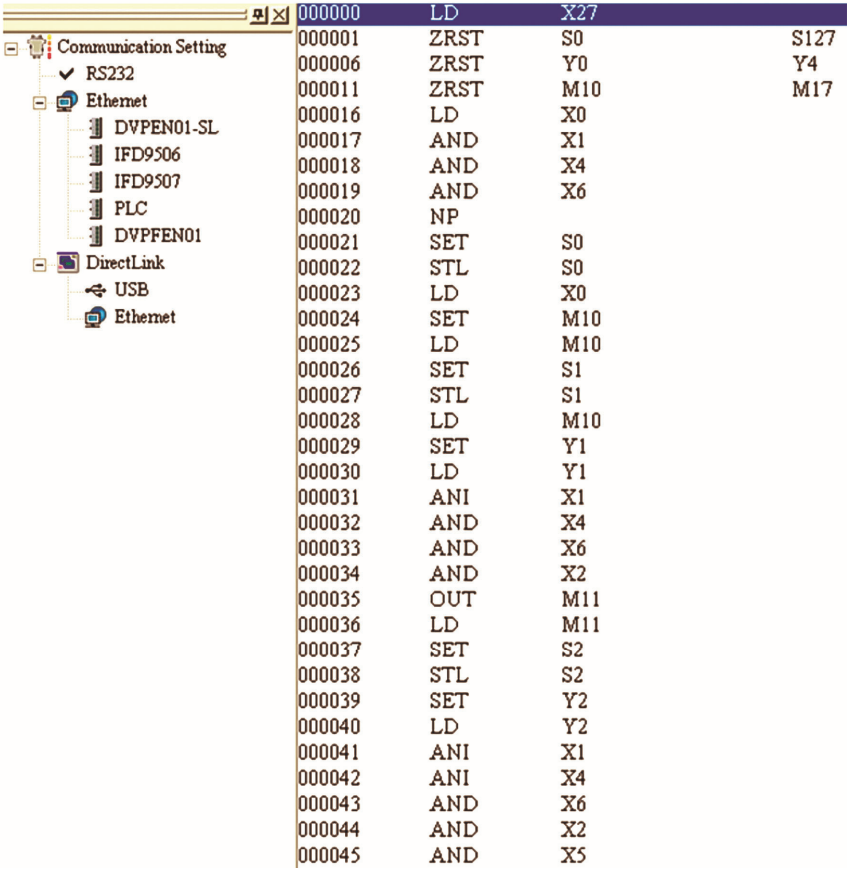


Рис. 4.4. Пример линейной рабочей программы SFC

Рис. 4.5. Пример раскрытия внутреннего содержания шага SFC

Рис. 4.6. Пример раскрытия внутреннего содержания перехода SFC



000000	LD	X27	
000001	ZRST	S0	S127
000006	ZRST	Y0	Y4
000011	ZRST	M10	M17
000016	LD	X0	
000017	AND	X1	
000018	AND	X4	
000019	AND	X6	
000020	NP		
000021	SET	S0	
000022	STL	S0	
000023	LD	X0	
000024	SET	M10	
000025	LD	M10	
000026	SET	S1	
000027	STL	S1	
000028	LD	M10	
000029	SET	Y1	
000030	LD	Y1	
000031	ANI	X1	
000032	AND	X4	
000033	AND	X6	
000034	AND	X2	
000035	OUT	M11	
000036	LD	M11	
000037	SET	S2	
000038	STL	S2	
000039	SET	Y2	
000040	LD	Y2	
000041	ANI	X1	
000042	ANI	X4	
000043	AND	X6	
000044	AND	X2	
000045	AND	X5	

Рис. 4.7. Пример конвертированной программы SFC в язык мнемокода

Можно себе представить, с какими трудностями столкнется наладчик электроавтоматики при отладке программы, приведенной ранее на рис. 4.3. Программу можно конвертировать в мнемокод (рис. 4.7), но и она не проще. Тем не менее, опытный инженер должен знать основы языка SFC, хотя бы для того, чтобы суметь наладить импортное оборудование, электроавтоматика которого написана на этом языке.

К нашему большому удовлетворению, в языке предусмотрен второй вариант набора и представление таких программ, это STL – Step Ladder Programming, где все значительно проще и понятнее. Язык очень похож на обычные лестничные диаграммы и использует следующие операнды:

STL – начало программы;

S – номер шага;

RET – конец программы. Остальное аналогично обычному программированию.

Чтобы упростить программирование, необходимо создать аналоги типовых решений на обоих языках. Так мы и поступим дальше. Здесь же скажем, что программная среда WPL Soft позволяет конвертировать программы SFC и STL друг в друга.

Ниже приводятся некоторые фрагменты таких аналогов.

Расходимость по И (рис. 4.8)

В программе **SFC**:

Если активен шаг S20 и выполнено условие перехода X1, то осуществляется переход к параллельному независимому выполнению ветвей с начальными шагами S21, S31 и S41.

В программе **STL** те же условия изображены в виде простой и понятной лестничной диаграммы: Если активен шаг S20 и включен бит X1, то одновременно инструкцией SET включаются те же шаги с адресами S21, S31 и S41. Набор программы осуществляется по обычным правилам и затруднений не вызывает.

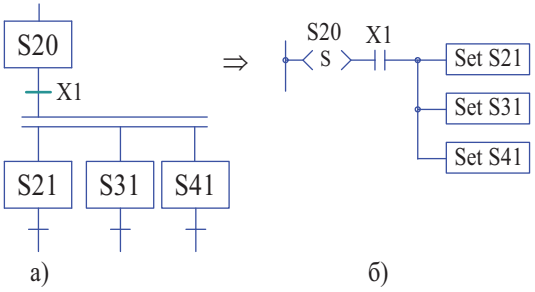


Рис. 4.8. SFC (а) и STL (б) аналоги расходимости по логическому И

Сходимость по И (рис. 4.9)

В программе **SFC**: Условием перехода из параллельных ветвей к линейному является завершение работы всех трех ветвей, т. е. активное состояние

шагов S27, S37 и S47. Дальнейший переход к шагу S50 произойдет при условии активного состояния операнда перехода X20.

В программе **STL** та же процедура выполняется простейшей логической цепочкой Set S50 = S27·S37·S47·X20.

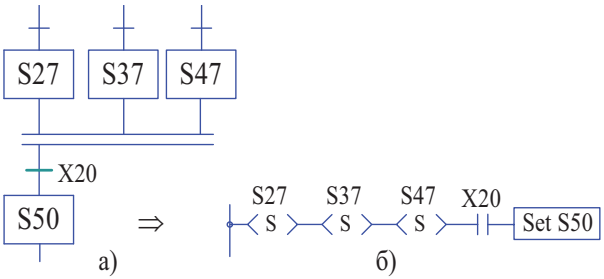


Рис. 4.9. SFC (а) и STL (б) аналоги сходимости по логическому И

Расходимость по ИЛИ (рис. 4.10)

В программе **SFC**: Если активен шаг S20, то по условиям переходов X1, X2 или X3, можно независимо активизировать запуск работы любой параллельной ветви S21, S31 и S41.

В программе **STL** те же условия изображены в виде лестничной диаграммы, соответствующей логическим уравнениям:

Set S21 = S20·X1, Set S31 = S20·X2, Set S41 = S20·X3.

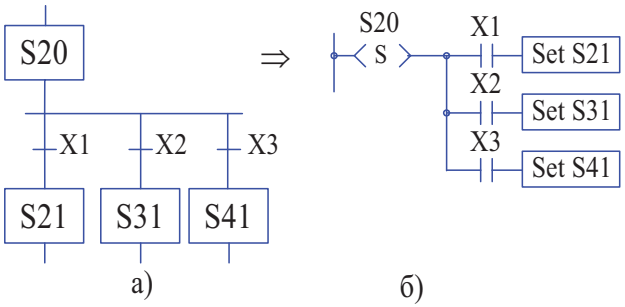


Рис. 4.10. SFC (а) и STL (б) аналоги расходимости по логическому ИЛИ

Сходимость по ИЛИ (рис. 4.11)

В программе **SFC**: Активизация шага S50 осуществляется при завершении работы любой их параллельных ветвей, признаком чего является активное состояние любого перехода X21, X22 или X23.

В программе **STL** те же условия изображены в виде лестничной диаграммы, соответствующей логическим уравнениям:

Set S50 = S27·X21 или Set S50 = S37·X22 или Set S50 = S47·X23.

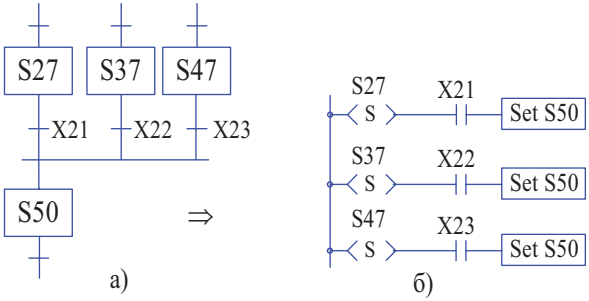


Рис. 4.11. SFC (а) и STL (б) аналоги сходимости по логическому И

В табл. 4.1 приведены используемые в языке служебные маркеры и регистры.

Таблица 4.1

Специальные реле и регистры

Устройство	Описание
M1040	Запрещение шаговых переходов
M1041	Старт шагового перехода. Флаг команды IST
M1042	Импульсный старт шагового перехода. Флаг команды IST
M1043	Возвращение в нулевую точку завершено. Флаг команды IST
M1044	Нахождение в нулевой точке. Флаг команды IST
M1045	Запрещение сброса всех выходов. Флаг команды IST
M1046	Выполнение STL-режима. M1046 = ON при начавшем выполнение любом шаге
M1047	Выключение мониторинга работы STL-режима
D1040	№ 1-го активного шага
D1041	№ 2-го активного шага
D1042	№ 3-го активного шага
D1043	№ 4-го активного шага
D1044	№ 5-го активного шага
D1045	№ 6-го активного шага
D1046	№ 7-го активного шага
D1047	№ 8-го активного шага

Далее приводим несколько рабочих программ, написанных на языке **SFC/STL**.

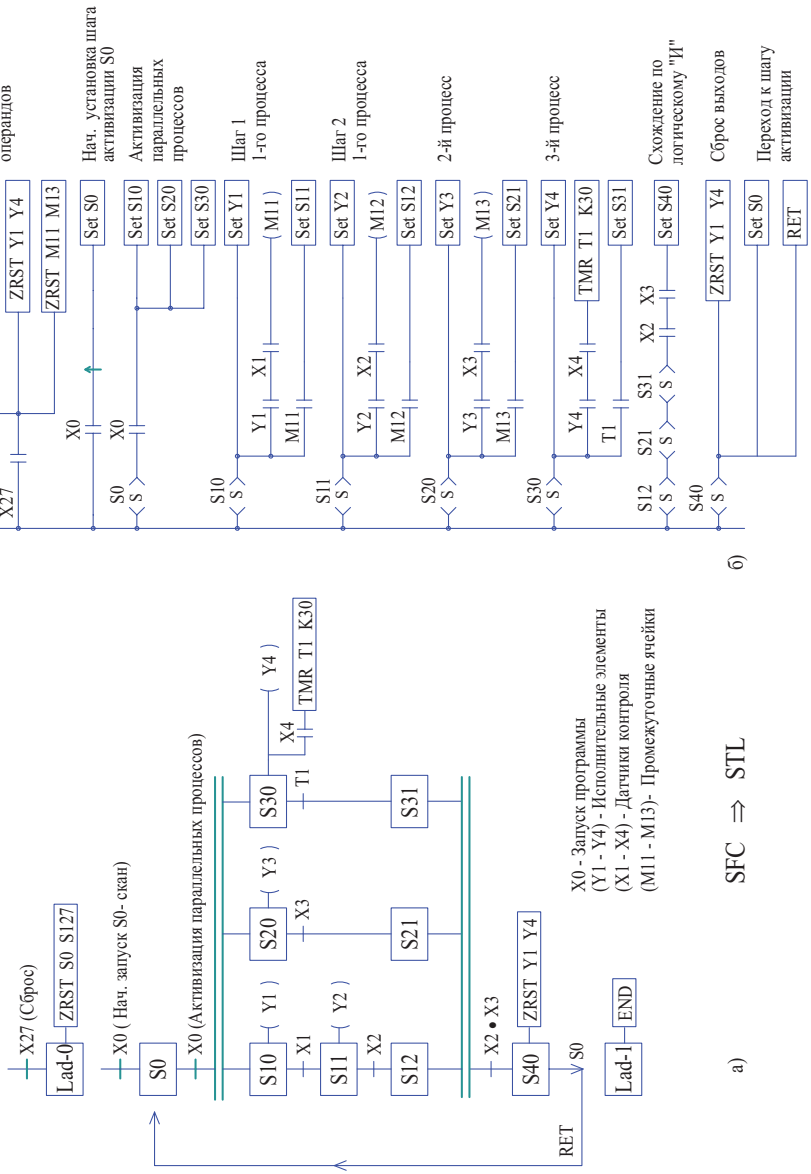


Рис. 4.12. Пример SFC(а) и STL(б) программ расходимости / входимости по логическому И

4.2. Типовые программы

Программа 1 (рис. 4.12 и 4.13). Разветвление с тремя параллельными ветвями по схеме И.

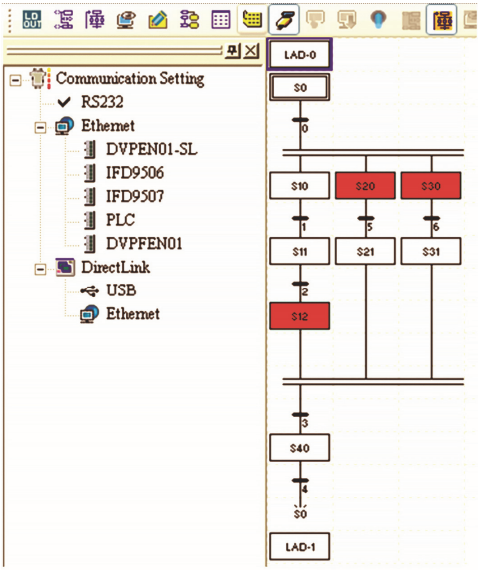


Рис. 4.13. Отображение программы SFC в среде WPL Soft

Программа включает следующие условные блоки:
Блок сброса Lad-0 всех операндов при включении контроллера и от кнопки X27.
Первый шаг запуска программы S0 от кнопки X0.
Три параллельных независимых процесса, также запускаемых от кнопки X0.

Процесс 1: S10 – шаг 1, включает выход Y1.
S11 – шаг 2, запускается при условии активного состояния выхода Y1 и сигнала контроля его выполнения M11=Y1·X1, включает выход Y2;
S12 – шаг 3, запускается при условии M12 = Y2·X2 и служит признаком завершения работы 1-го процесса.

Процесс 2: S20 – шаг 1, включает выход Y3;
S21 – шаг 2, запускается при условии M13 = Y3·X3 и служит признаком завершения работы 2-го процесса.

Процесс 3: S30 – шаг 1, включает выход Y4, при условии Y4·X4 запускает таймер T1;

S31 – шаг 2, запускается по окончании выдержки времени таймера T1 и служит признаком завершения работы 3-го процесса.

Линейный шаг S40 – запускается при условии завершения работы всех трех параллельных процессов и наличия сигналов контроля, т. е. при S12·S21·S31·X2·X3. Выключает выходы Y1–Y4 и переводит программу к начальному шагу активизации программы.

Блок Lad-1.

Конвертированная программа в среде WPL Soft приведена на рис. 4.13.

Программа 2 (рис. 4.14 и 4.15). Разветвление с тремя параллельными ветвями по схеме ИЛИ.

Включает следующие условные блоки:

Блок сброса Lad-0 всех операндов при включении контроллера и от кнопки X27.

Первый шаг запуска программы S0 от кнопки X0.

Линейный шаг S10, запускаемый операндом X1.

Три параллельных независимых процесса, запускаемых, соответственно, операндами X2, X3 и X4 при условии активного состояния шага S10.

Процесс 1: S20 – шаг 1, запускается операндом X2, включает выход Y1;

S21 – шаг 2, запускается по условию M11 = Y1·X5, включает выход Y2 и при наличии сигнала контроля X6 по условию M12 = Y2·X6 активизирует линейный шаг S50.

Процесс 2: S30 – шаг 1, запускается операндом X3, включает выход Y3 и по условию M13 = Y3·X7 активизирует линейный шаг S50.

Процесс 3: S40 – шаг 1, запускается операндом X4, включает выход Y4, сигналом контроля X20 запускает таймер T0 и по истечение выдержки времени активизирует линейный шаг S50.

Линейный шаг S50 – запускается при условии завершения работы любого из трех параллельных процессов, включает выход Y5, по условию Y5·X21 запускает таймер T1.

Линейный шаг S51 запускается выходным сигналом таймера T1, сбрасывает выходы Y1–Y5 и дает команду на возврат к первому шагу программы.

Блок Lad-1.

Конвертированная программа SFC в среде WPL Soft приведена на рис. 4.15.

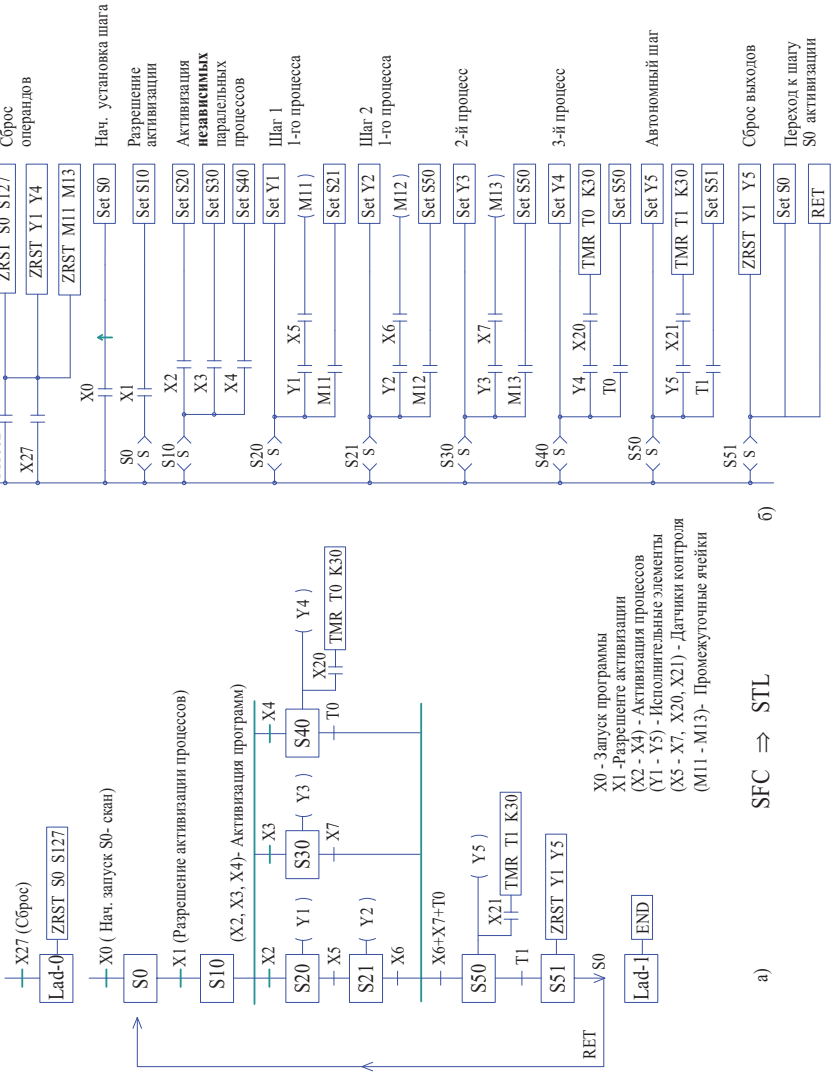


Рис. 4.14. Пример SFC(а) и STL(б) программ расходимости / входимости по логическому ИЛИ

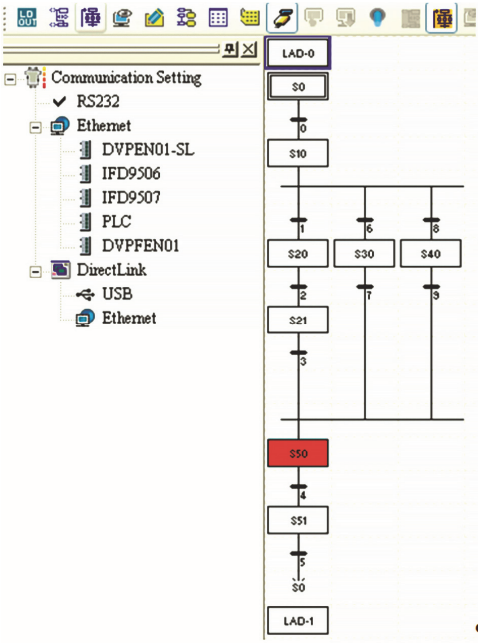


Рис. 4.15. Отображение программы в среде WPL Soft

Программа 3 (рис. 4.16, 4.17 и 4.18). Линейная программа с условным переходом **вперед**.

Программа включает следующие условные блоки:

Блок сброса Lad-0 всех операндов при включении контроллера и от кнопки X27.

Первый шаг запуска программы S0 от кнопки X0.

Линейный шаг S20. Запускается операндом X1, при условии X2 = 1 включает выход Y1, при условии Y1·X3 запускает следующий шаг S21.

Линейный шаг S21. Включает выход Y2, при условии Y2·X5 запускает следующий шаг S22.

Линейный шаг S22. Включает выход Y3, при условии Y3·X6 запускает следующий шаг S23.

Линейный шаг S23 является признаком завершения программы. Сбрасывает выходы Y1–Y3, отключает шаги S20–S23, дает команду на переход к начальному шагу программы.

Ветвь условного перехода вперед от шага S20 к шагу S22 при условии активизации операнда X4 вместо операнда X3.

Блок Lad-1.

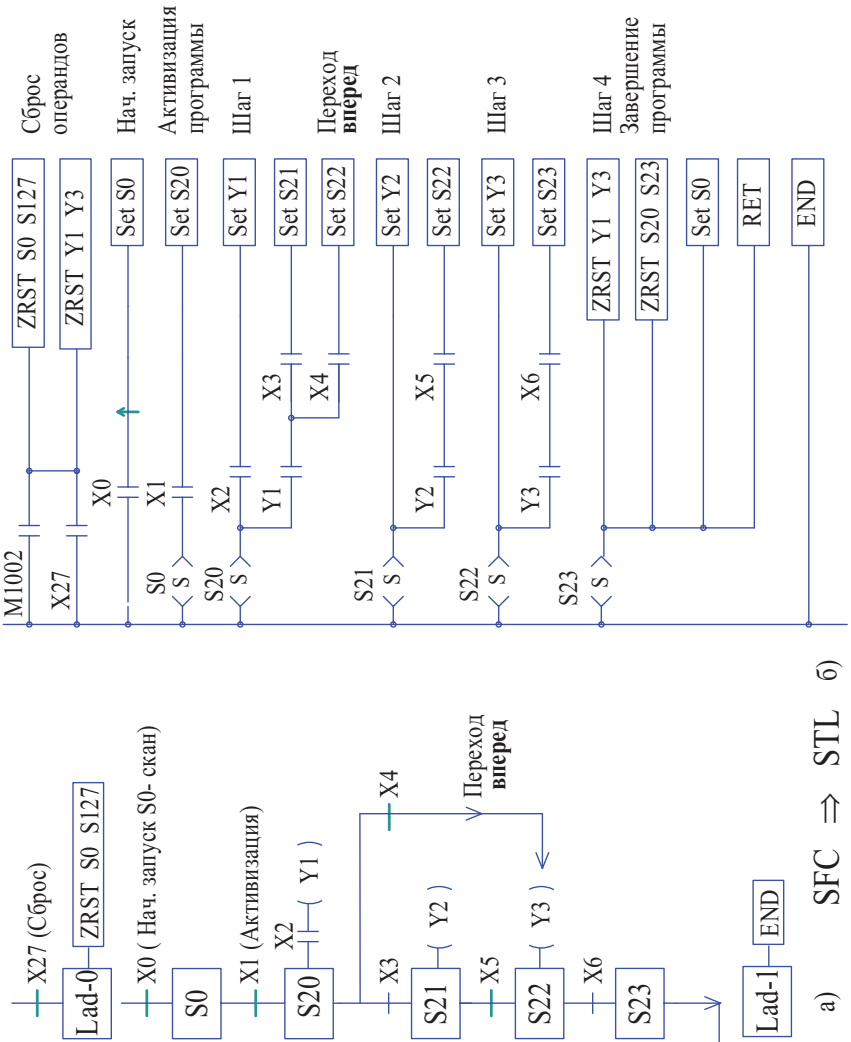


Рис. 4.16. Пример SFC(а) и STL(б) программ с условным переходом вперед

На рис. 4.16 и 4.18, соответственно, показаны рабочая программа STL и конвертированная программа SFC в среде WPL Soft.

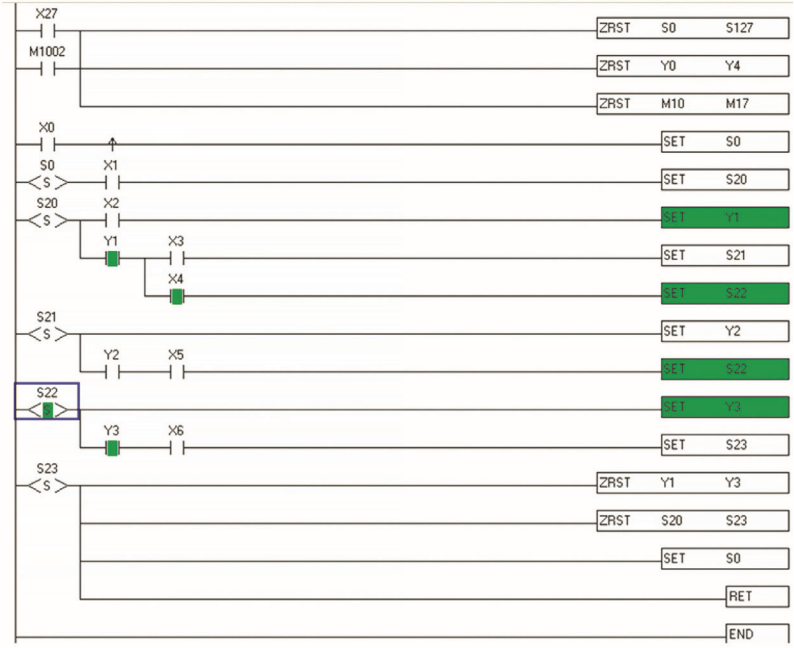


Рис. 4.17. Отображение STL-программы в среде WPL Soft

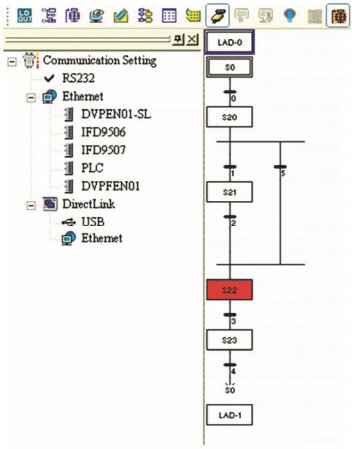


Рис. 4.18. Отображение SFC-программы в среде WPL Soft

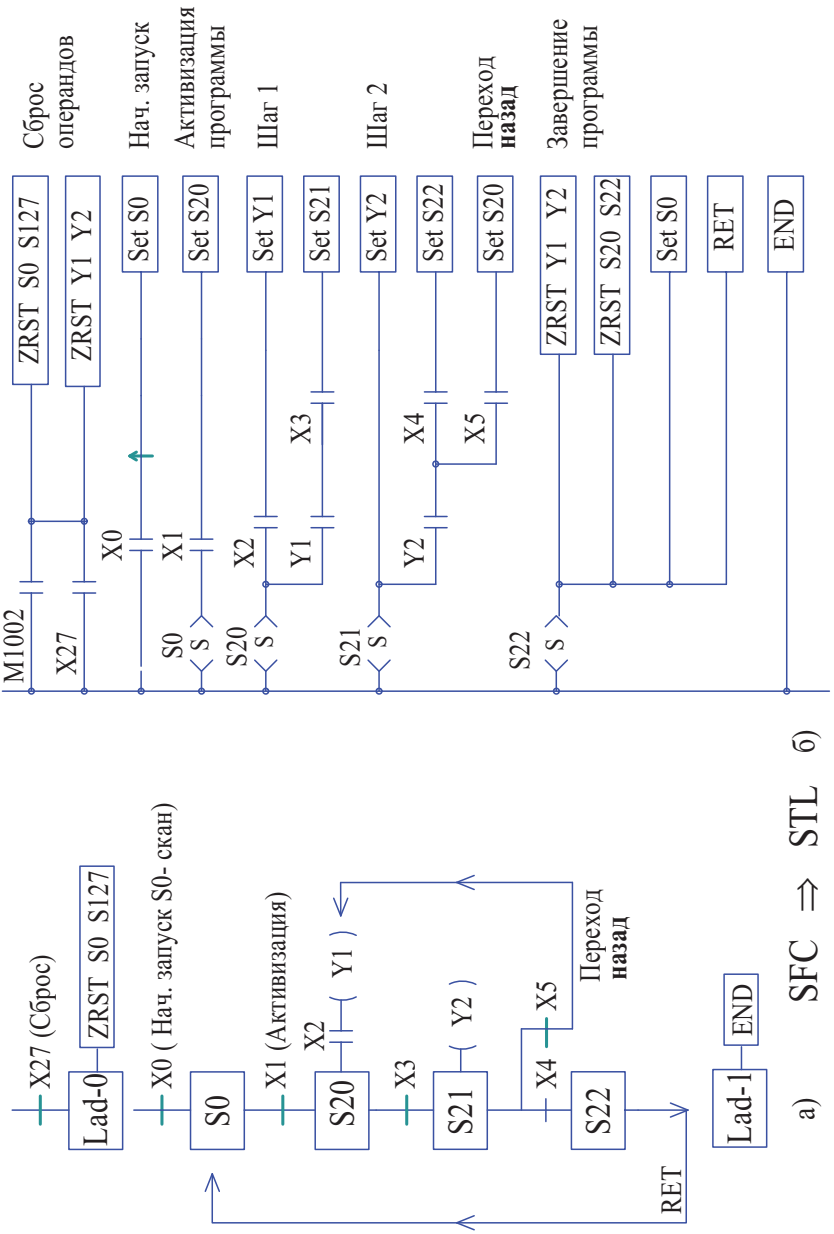


Рис. 4.19. Пример SFC(а) и STL(б) программ с условным переходом назад

Программа 4 (рис. 4.19, 4.20 и 4.21). Линейная программа с условным переходом **назад**.

Программа включает следующие условные блоки:

Блок сброса Lad-0 всех операндов при включении контроллера и от кнопки X27.

Первый шаг запуска программы S0 от кнопки X0.

Линейный шаг S20. Запускается операндом X1, при условии X2 = 1 включает выход Y1, при условии Y1·X3 запускает следующий шаг S21.

Линейный шаг S21. Включает выход Y2, при условии Y2·X4 запускает следующий шаг S22

Линейный шаг S22 является признаком завершения программы. Сбрасывает выходы Y1–Y2, отключает шаги S20–S22, дает команду на переход к начальному шагу программы.

Ветвь условного перехода назад от шага S21 к шагу S20 при условии активизации операнда X5 вместо операнда X4.

Блок Lad-1.

На рис. 4.20 и 4.21, соответственно, показаны рабочая программа STL и конвертированная программа SFC в среде WPL Soft.

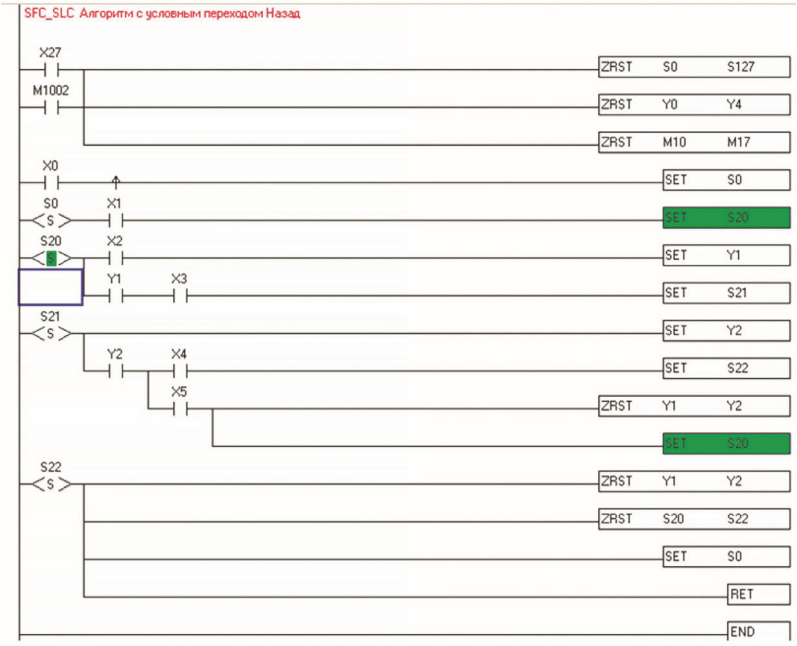


Рис. 4.20. Отображение STL-программы в среде WPL Soft

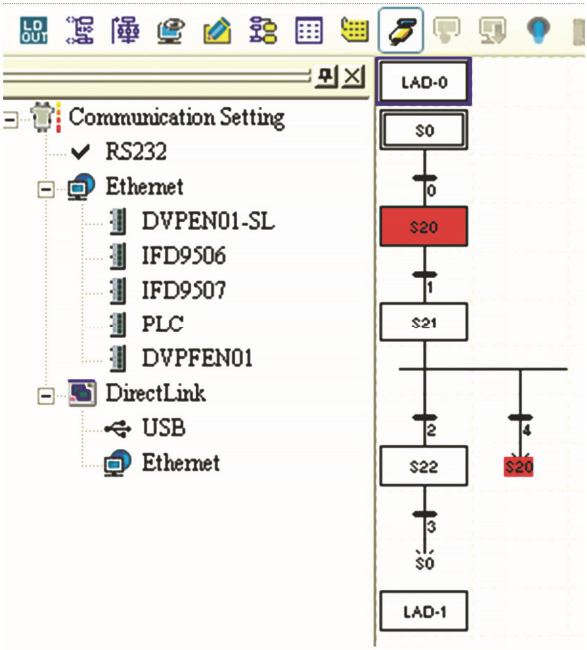


Рис. 4.21. Конвертированная SFC-программы в среде WPL Soft

Программа 4 (рис. 4.22 и 4.23). Две параллельные независимые структуры с возможностью перехода из первой программы во вторую. Начальный шаг первой программы S0, а второй S1.

Первая программа включает следующие условные блоки:

Блок сброса Lad-0 всех операндов при включении контроллера и от кнопки X27.

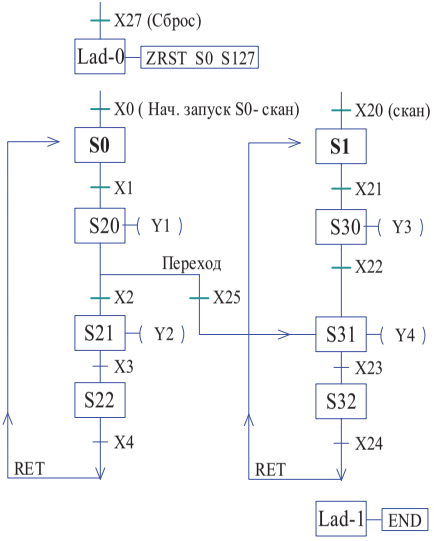
Первый шаг запуска программы **S0** от кнопки X0.

Линейный шаг S20. Запускается операндом X1, включает выход Y1, при при условии Y1·X2 запускает следующий шаг S21.

Линейный шаг S21. Включает выход Y2, при условии Y2·X3 запускает следующий шаг S22.

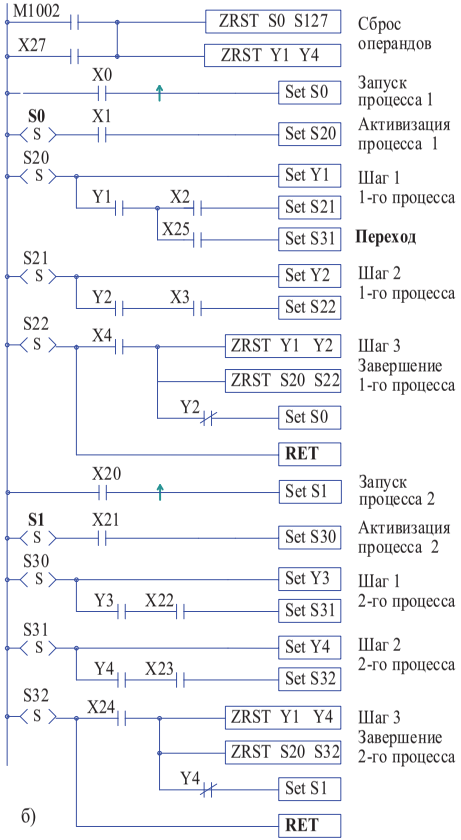
Линейный шаг S22 является признаком завершения программы. При условии X4 = 1 сбрасывает выходы Y1–Y2, отключает шаги S20–S22, дает команду на переход к начальному шагу программы **S0**.

Ветвь условного перехода от шага S21 к шагу S31 другой программы при условии активизации операнда X25 вместо операнда X2.



а)

SFC ⇒ STL



б)

Рис. 4.22. Пример SFC(а) и STL(б) программ с условным переходом в другой процесс

Вторая программа включает следующие условные блоки:
Первый шаг запуска программы **S1** от кнопки X20.
Линейный шаг S30. Запускается операндом X21, включает выход Y3, при условии Y3·X22 запускает следующий шаг S31.
Линейный шаг S31. Включает выход Y4, при условии Y4·X2 запускает следующий шаг S32.
Линейный шаг S32 является признаком завершения второй программы. Сбрасывает выходы Y1–Y4, отключает шаги S20–S32, дает команду на переход к начальному шагу программы **S1**.
Блок Lad-1.

На рис. 4.23 показана конвертированная программа SFC в среде WPL Soft.

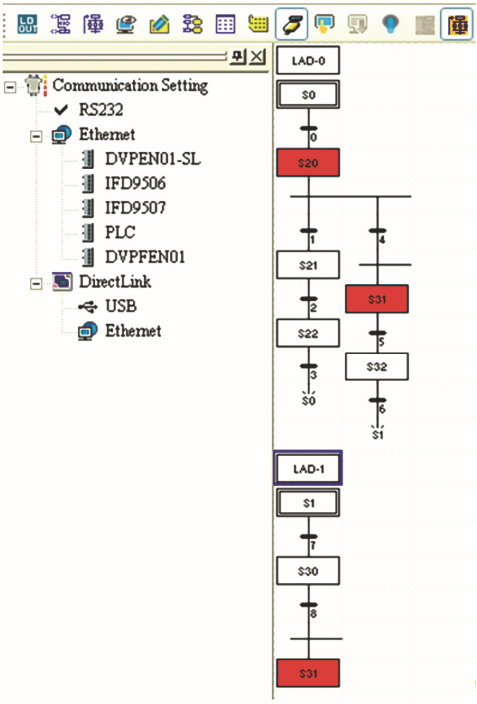


Рис. 4.23. Конвертированная SFC-программа в среде WPL Soft

Обращаем внимание, что независимые программы программируются *последовательно*, одна за другой.

4.3. Управление манипулятором смены инструмента на языке SFC / STL

Такой проект уже был выполнен в нескольких вариантах в главе 2 с использованием стандартных Ladder-диаграмм. Для удобства изложения и понимания материала повторим здесь рисунок цикла смены и фрагмент принципиальной схемы.

Движение манипулятора (рис. 4.24) в цикле автоматической смены инструментов станка с ЧПУ, а также операции отжима и зажима инструмента в шпинделе производятся за счет включения соответствующих **электромагнитов** $У_1 \ldots У_4$, управляющих **гидравлическими золотниками.**

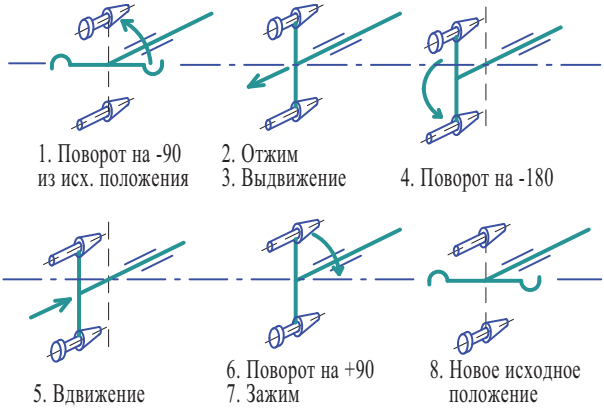


Рис. 4.24. Движения манипулятора в процессе смены инструмента

Работа над проектом всегда включает *следующие этапы*:

1. Изучение принципа работы механизма (рис. 4.24) и разработка аппаратной принципиальной схемы. Принципиальная схема (рис. 4.25) включает в себя:
   - асинхронный привод насоса гидравлики;
   - схему подключения дискретных входов и выходов;
   - схемы управления электромагнитами на постоянном токе и включения насоса гидравлики на переменном токе (не показано).

   Здесь же приводятся принятые адреса входных и выходных сигналов.

2. Формализация работы механизма при помощи циклограммы (рис. 4.26). Рабочие сигналы в циклограмме показываются в порядке их появления в процессе выполнения цикла и всегда в прямом виде, т. е. включенному состоянию элемента соответствует высокий (единичный) уровень сигнала, а отключенному – низкий (нулевой).

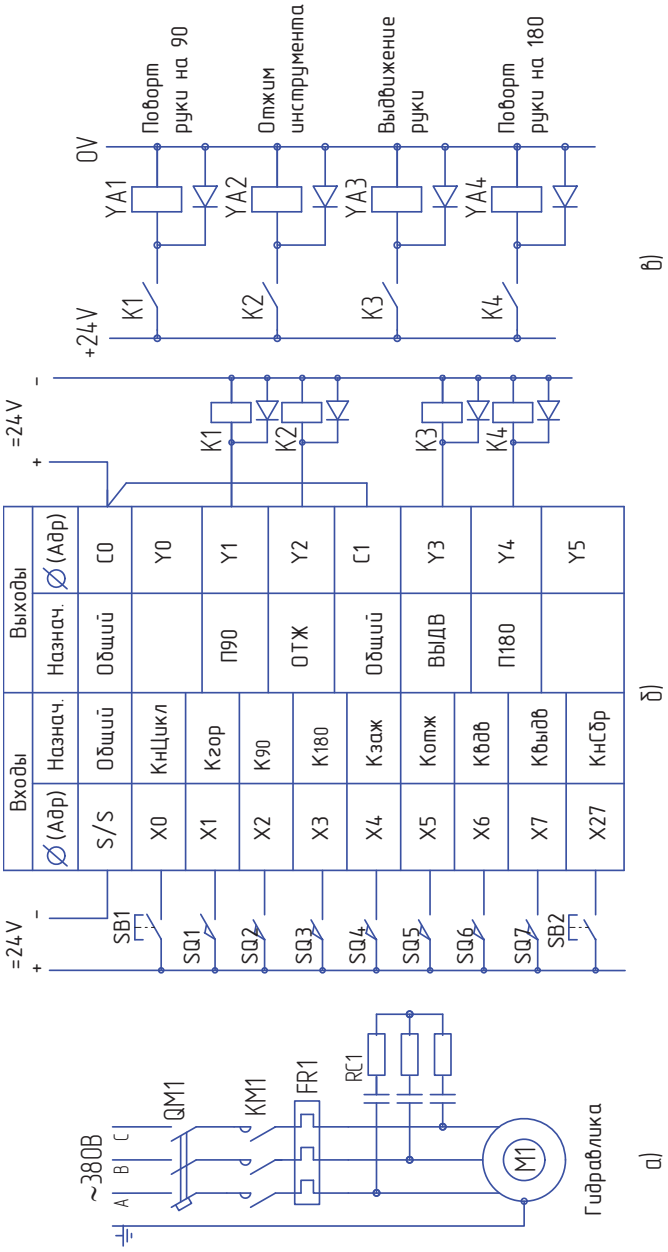


Рис. 4.25. Принципиальная аппаратная схема:

а – двигатель гидравлики; *б* – дискретные входы и выходы; *в* – электромагниты

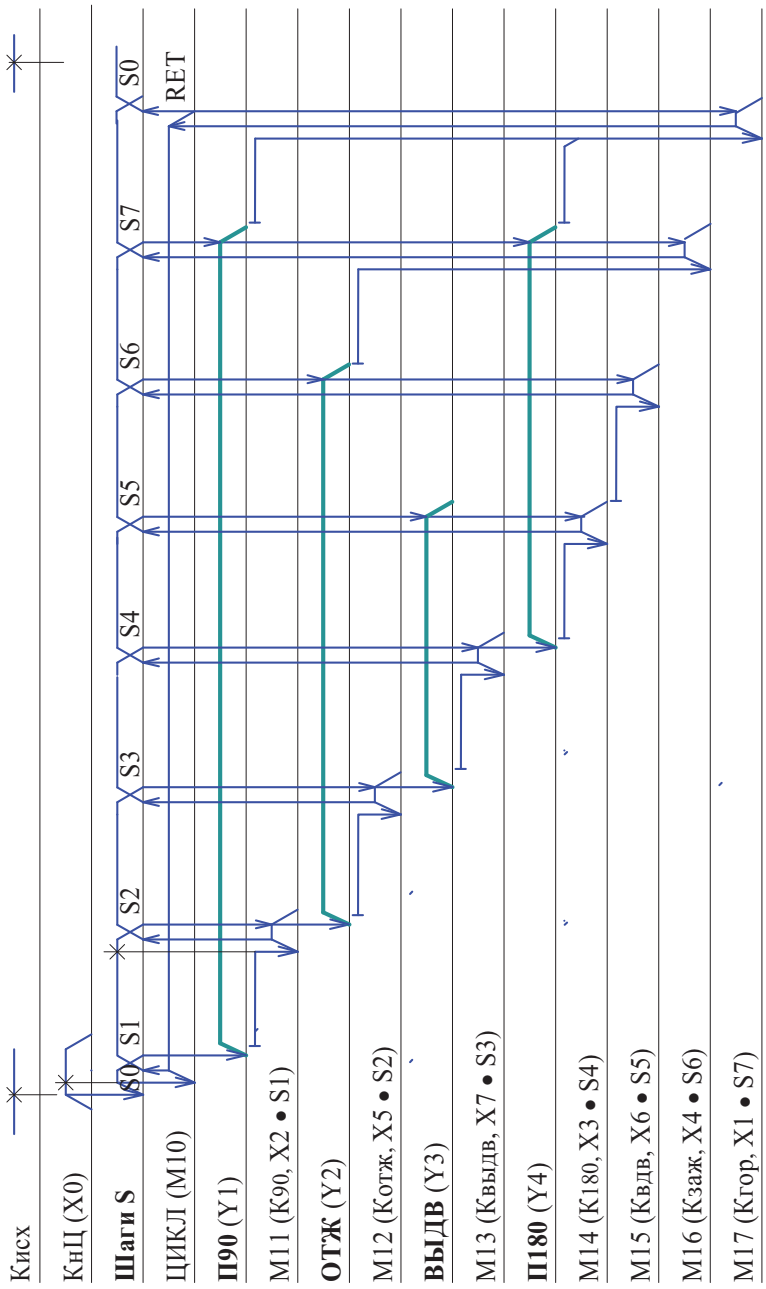


Рис. 4.26. Циклограмма смены инструмента в формате STL

Присвоение логическим переменным легко запоминающихся условных обозначений упрощает процесс синтеза и его чтение другими лицами.

Контроль выполнения операций осуществляется бесконтактными путевыми выключателями.

3. Синтез по циклограмме логических уравнений, описывающих поэтапную работу механизма.

4. Формальное вычерчивание релейно-контактного алгоритма, адаптированного к конкретному языку программируемого контроллера.

Что касается написания алгоритмов управления, то любую задачу можно решить различными методами, каждому из них будут присущи свои преимущества и недостатки. В главе 2 настоящей книги приведены примеры решения данной задачи тремя различными способами;

- на основе циклограммы рис. 2.57, описывающей реальное состояние выходных и контролирующих элементов на каждый момент времени и аппарата алгебры логики (см. гл. 2);
- методом счетчика последовательности (аналог SFC) с общим сбросом памятей сигналов в конце цикла (см. рис. 2.61);
- методом счетчика последовательности с логическим отдельным сбросом памяти каждого сигнала (см. рис. 2.64).

Ниже приводится решение задачи на языке последовательных лестничных цепей STL.

Основой для проектирования является таблица соответствия исполнительных и контролирующих элементов управления манипулятором (табл. 4.2).

Таблица 4.2

Состояние электромагнитов и датчиков контроля в цикле смены

Операция	Электромагниты				Датчики контроля						
	Y1	Y2	Y3	Y4	Кгор X1	Кзаж X4	Квдв X6	К90 X2	Котж X5	Квыдв X7	К180 X3
Исх	0	0	0	0	1	1	1	0	0	0	0
П90	**1**	0	0	0	**0**	1	1	**1**	0	0	0
ОТЖ	1	**1**	0	0	0	**0**	1	1	**1**	0	0
Выдв	1	1	**1**	0	0	0	0	1	1	**1**	0
П180	1	1	1	**1**	0	0	0	1	1	1	**1**
ВДВ	1	1	**0**	1	0	0	**1**	1	1	**0**	1
ЗАЖ	1	**0**	0	1	0	**1**	1	1	**0**	0	1
Исх	**0**	0	0	**0**	**1**	1	1	**0**	0	0	**0**

Такая таблица всегда приводится в гидравлической схеме. На ее основе разрабатывается циклограмма (рис. 4.26) и программа (рис. 4.27) управления манипулятором на языке STL.

Программа STL включает следующие этапы (шаги):

Шаг S0 – начальная инициализация программы от кнопки X0 (Пуск цикла), включение сигнала M10- цикл, при условии исходного положения манипулятора: X1(рука горизонтально), X4 (инструмент зажат), X6 (рука вдвинута). Запуск шага S1.

Шаг S1 – включение электромагнита Y1, поворот манипулятора на 90 градусов, захват манипулятором инструментов в шпинделе и магазине. Формирование сигнала контроля поворота M11. Ключевой сигнал X2 (контроль поворота на 90 градусов). В программе сделаны дополнительные блокировки. Запуск шага S2.

Шаг S2 – включение электромагнита Y2, отжим инструмента в шпинделе. Формирование сигнала контроля отжима M12. Ключевой сигнал X5 (контроль отжима инструмента). В программе сделаны дополнительные блокировки. Запуск шага S3.

Шаг S3 – включение электромагнита Y3, выдвижение руки, т. е. вытаскивание инструментов из шпинделя и магазина. Формирование сигнала контроля выдвижения M13. Ключевой сигнал X7 (контроль выдвинутого положения руки). В программе сделаны дополнительные блокировки. Запуск шага S4.

Шаг S4 – включение электромагнита Y4, поворот манипулятора на 180 градусов. Формирование сигнала контроля поворота M14. Ключевой сигнал X3 (контроль поворота на 180 градусов). В программе сделаны дополнительные блокировки. Запуск шага S5.

Шаг S5 – выключение электромагнита Y3, выдвижение руки, т. е. установка нового инструмента в шпиндель и возврат отработавшего инструмента в магазин. Формирование сигнала контроля вдвинутого положения руки M15. Ключевой сигнал X6 (рука вдвинута). В программе сделаны дополнительные блокировки. Запуск шага S6.

Шаг S6 – включение электромагнита Y2, зажим инструмента в шпинделе. Формирование сигнала контроля зажима M16. Ключевой сигнал X4 (контроль зажима инструмента в шпинделе). В программе сделаны дополнительные блокировки. Запуск шага S7.

Шаг S7 – выключение электромагнитов Y1 и Y4, возврат руки в исходное горизонтальное положение. Формирование сигнала контроля исходного положения M17. Ключевой сигнал X1 (рука горизонтально). В программе сделаны дополнительные блокировки. Выключение сигнала M10-цикл.

Переход к начальному шагу программы S0.

Таким образом, программа, написанная на языке SFC (STL), является аналогом программы, написанной при помощи метода счетчика последовательности и принципиально отличается от программы, написанной на основе объективной формализации. Следует признать, что ее синтез проще, но программа не является

универсальной и не может быть формально заимствована при программировании на другом языке.

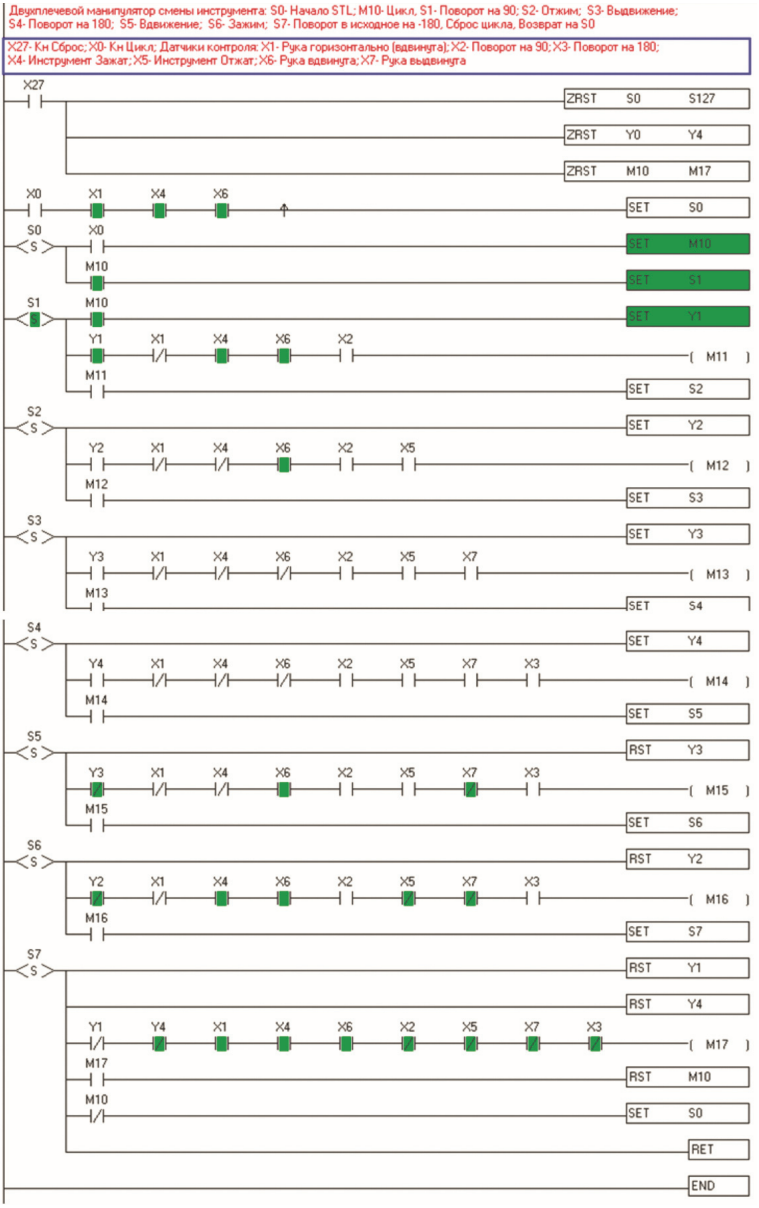


Рис. 4.27. STL-программа управления манипулятором

ГЛАВА 5. ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР ТИПА AS324MT-A

Контроллеры серии AS300 фирмы Дельта являются высокопроизводительными ПЛК модульного типа и выпускаются в различных модификациях.

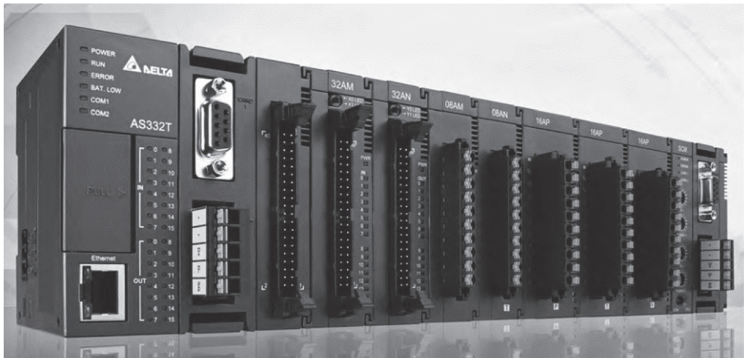


Рис. 5.1. Общий вид контроллера серии AS300

В данной главе рассматривается ПЛК типа AS324MT-A.

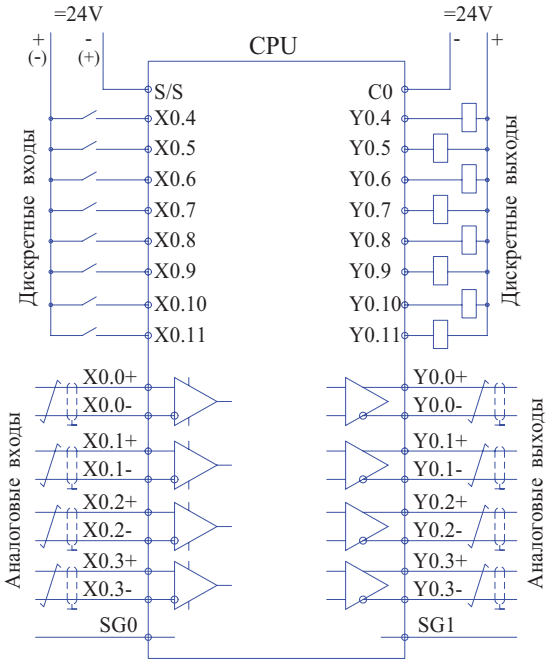
5.1. Основные технические характеристики

Основные технические характеристики:

- общее питание = 24 В;
- процессорный модуль CPU A300
  дискретные входы – 8 (= 24 В, 5 мА);
  дискретные выходы – 8 (= 24 В, NPN);
  дифференциальные входы – 4 (= 5 В, 5 мА);
  дифференциальные выходы – 4 (5 мА);
  расширение до 1016 I/O;
  128 k шагов, наличие скоростных блоков;
  Ethernet, 2*RS485, 1*USB, 1*microSD;
- модуль расширения на 16 дискретных входов (= 24 В, 5 мА);
- модуль расширения на 16 дискретных релейных выходов (≈ 240 В/ = = 24 В, 2 А (выход), 8А(одновременно);
- модуль расширения на 4 аналоговых выхода (+/−10 В, 0–10 В, +/−20 мА, 0–20 мА);
- программное обеспечение ISP Soft V3.11.

5.2. Подключение

На рис. 5.2 приведена принципиальная схема подключения дискретных и аналоговых входов и выходов контроллера. Указанные на всех последующих схемах подключения адреса присваиваются при выполнении процедуры конфигурации контроллера. Общее питание блока = 24 В подключается через отдельный разъем и на схеме не показано.



№	Назначение		№
40	S/S	S/S	39
38	X0.11	X0.10	37
36	X0.9	X0.8	35
34	X0.7	X0.6	33
32	X0.5	X0.4	31
30	SG0	SG0	29
28	X0.3-	X0.3+	27
26	X0.2-	X0.2+	25
24	X0.1-	X0.1+	23
22	X0.0-	X0.0+	21
20	C0	C0	19
18	Y0.11	Y0.10	17
16	Y0.9	Y0.8	15
14	Y0.7	Y0.6	13
12	Y0.5	Y0.4	11
10	SG1	SG1	9
8	Y0.3-	Y0.3+	7
6	Y0.2-	Y0.2+	5
4	Y0.1-	Y0.1+	3
2	Y0.0-	Y0.0+	1

б)

Рис. 5.2. Принципиальная схема (а) и схема подключения (б) входов и выходов центрального процессора

Дискретные входы с адресами X0.0…X0.11 подключаются через источник питания = 24 В *любой* полярности, однако всегда рекомендуется принятие какой-либо постоянной системы. Как правило, общий провод входов следует подключать к положительному выводу ИП.

Дискретные выходы с адресами Y0.4…Y0.11 также подключаются через источник постоянного тока = 24 В, однако их общий провод *обязательно* должен быть подключен к положительному выводу ИП, так как в модуле используется NPN – выходной транзистор (Sink-система).

Аналоговые входы с адресами X0.0…X0.3 должны иметь уровень входных сигналов равный 5 В. Максимально допустимая частота зависит от типа используемых при программировании счетчиков.

Аналоговые выходы с адресами Y0.0…Y0.3 так же имеют уровень сигналов, равный 5В и их максимально допустимая частота зависит от типа используемых при программировании счетчиков.

При использовании аналоговых входов и выходов обязательна попарная скрутка и экранирование информационных проводов. Так же обязательно подключение выводов SG0 и SG1 на землю.

Двойные выводы S/S, C0, SG0, SG1 объединены внутри контроллера.

На рис. 5.3 приведена принципиальная схема подключения дискретных входов и выходов модулей расширения, а также топология их разъемов.

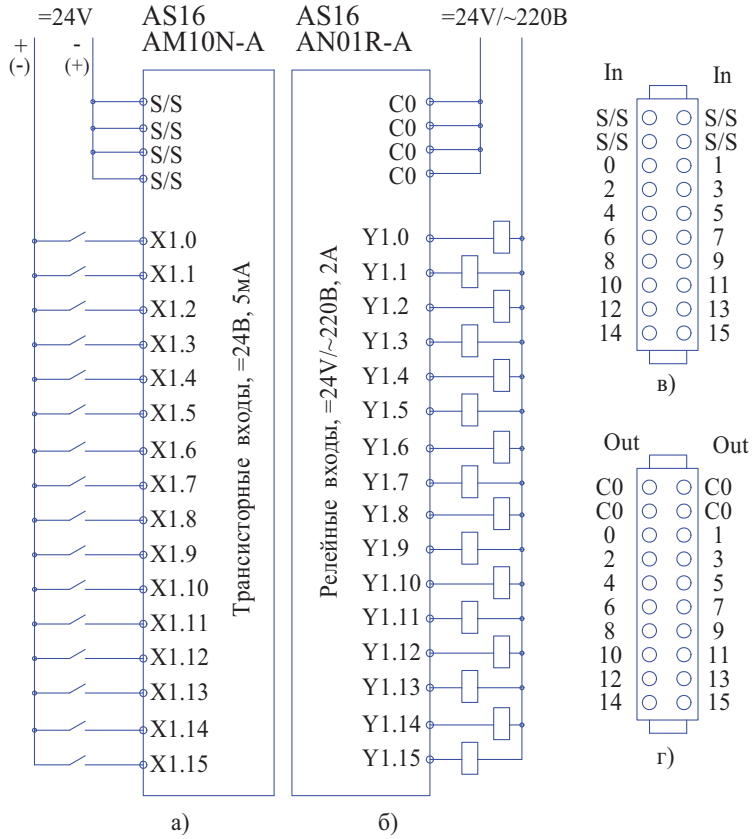


Рис. 5.3. Принципиальные схемы (а, б) и топологии разъемов (в, г) модулей расширения дискретных входов и выходов

Модуль дискретных входов обеспечивает подключение 16 входов с адресами X1.0…X1.15.

Модуль релейных выходов обеспечивает подключение 16 выходов с адресами Y1.0…Y1.15. В связи с тем, что их подключение осуществляется через «сухой» контакт встроенного в модуль реле, то допускается подключение нагрузки как постоянного, так и переменного тока.

Адреса входов и выходов также назначаются при конфигурации системы.

Для обоих модулей используются «втычные» разъемы.

На рис. 5.4 приведена принципиальная схема подключения модуля аналоговых выходов и топология его разъема.

Модуль требует отдельного питания от внешнего источника = 24 В.

Выходы VO1…VO4 предназначены для формирования выходного аналогового напряжения +/−10 В или (0…10) В, а выходы IO1…IO4 для формирования выходного токового сигнала +/−20 мА или (0…20) мА.

При использовании аналогового входа обязательна попарная скрутка и экранирование информационных проводов, а также заземление экрана.

Как видно их выходной характеристики аналогового выхода, максимальному выходному напряжению +/−10 В соответствует цифровой входной код +/−32000. Формирование кода осуществляется в программе электроавтоматики и будет показано ниже.

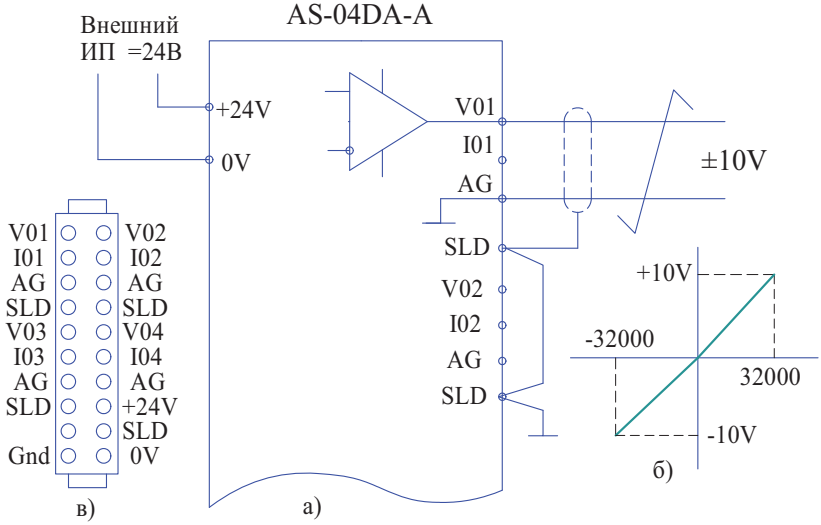


Рис. 5.4. Топология разъема (а), принципиальная схема подключения (б) и выходная характеристика (в) модуля аналоговых выходов

Выходные адреса каналов ЦАП устанавливаются при конфигурации системы:

Output CH1 → D28002…D28003
Output CH2 → D28004…D28005
Output CH3 → D28006…D28007
Output CH4 → D28008…D28009,

а посылка кода на выход осуществляется напрямую инструкцией DMOV.

Важно! Указанные выше адреса и принцип их активизации были определены при реальной отладке программы на контроллере. Первоначальный теоретический проект выполнялся по скаченной на сайте технической документации. Как выяснилось, она была не исправлена и не соответствовала данному типу контроллера. Проект оказался неработоспособным. Нужно быть готовым к подобным ситуациям, они возникают довольно часто.

5.3. Синтаксис языка электроавтоматики программируемого контроллера типа AS324MT-A

Состав доступных программных операндов контроллера и их система адресации приведена в табл. 5.1.

В распоряжении разработчика электроавтоматики имеются следующие фиксированные адреса и часы реального времени:

SM400 – бит, всегда равный единице (= 1);
SM401 – бит, всегда равный нулю (= 0);
SM402 – единичный импульс 1-го скана программы;
SM403 – нулевой импульс 1-го скана программы;
SM404 – генератор импульсов с периодом T = 10 мс;
SM405 – генератор импульсов с периодом T =100 мс;
SM406 – генератор импульсов с периодом T = 200 мс;
SM407 – генератор импульсов с периодом T = 1 с.

Часы реального времени

BCD	D	Назначение
SR220	SR391	Год
SR221	SR392	Месяц
SR222	SR393	День
SR223	SR394	Часы
SR224	SR395	Минуты
SR225	SR396	Секунды
SR226	SR397	Дни недели

Таблица 5.1

Доступные операнды

Тип	Наименование объекта		Количество объектов	Диапазон
Битовые объекты	Входное реле	X	1024	X0.0~X63.15
	Выходное реле	Y	1024	Y0.0~Y63.15
	Регистр данных	D	48,0000	D0.0~D29999.15
		W	48,0000	W0.0~W29999.15 *4
	Вспомогательное реле	M	8192	M0~M8191
	Специальное вспомогательное реле	SM	2048	SM0~SM4095
	Шаговое реле	S	2048	S0~S2047
	Таймер	T	512	T0~T511
	Счетчик	C	512	C0~C511
	32-битный счетчик	HC	256	HC0~HC255
Словные объекты	Входное реле	X	64	X0~X63
	Выходное реле	Y	64	Y0~Y63
	Регистр данных	D	30000	D0~D29999
		W	30000	W0~W29999 *4
	Специальное вспомогательное реле	SR	2048	SR0~SR2047
	Файловый регистр	FR	65536	FR0~FR65535
	Таймер	T	512	T0~T511
	Счетчик	C	512	C0~C511
	32-битный счетчик	HC	256 (512 words)	HC0~HC255
	Индексный регистр	E	10	E0~E9
			5	E10~E14 *4
Константы	Десятичный формат	K	16 бит: -32768~32767 32 бит: -2147483648~2147483647	
Константы	Шестнадцатеричный формат	16#	16 бит: 16#0~16#FFFF 32 бит: 16#0~16#FFFFFFFF	
	С плавающей точкой	F	32 бит: $\pm1.17549435^{-38}$~$\pm3.40282347^{+38}$	
Символьная переменная	Символьная переменная	“$”	1~31 символов	

*1: В списках объектов в Главах 5 и 6 настоящего Руководства по программированию десятичные значения обозначаются буквой K. Например, для K50 в ISPSoft может быть задан только номер 50.

*2: В списках объектов в Главах 5 и 6 настоящего Руководства по программированию числа с плавающей запятой обозначаются F / DF, а в ISPSoft они представлены десятичными точками; для F500 с плавающей запятой следует ввести 500,0.

*3: В Главах 5 и 6 настоящего Руководства по программированию строки обозначаются символом «$», например, в ISPSoft для строки 1234, следует ввести «1234».

*4: Используется только для редактирования в ISPSoft.

Все многообразие команд или инструкций, используемых при программировании электроавтоматики, вне зависимости от типа языка, разбивается на две группы: *базовые* и *функциональные* команды.

5.3.1. Базовые команды

К *базовым* командам относятся логические инструкции, реализующие основные функции алгебры логики И, ИЛИ, НЕ, равнозначность, а также команды считывания первоначальной информации и посылки результата выполнения операции в выходной модуль или память. Эти команды отличаются простотой и доступностью техническому персоналу средней квалификации, и предназначены для программирования алгоритмов управления, аналогичных обычным комбинационным релейно-контактным или бесконтактным схемам электроавтоматики.

К базовым следует отнести также инструкции программирования таймеров и простейших счетчиков.

Любой программируемый контроллер имеет стандартный набор базовых инструкций, отличающихся, как правило, только аббревиатурой.

Основные базовые логические команды приведены на рис. 5.5:

а) считывание первого в логической цепи прямого или инверсного операнда;
б) логическое умножение на прямой или инверсный операнд;
в) посылка результата логической операции на выход;
г) команды установки и сброса логического выхода;
д) команда инверсии логической цепи.

Процедура набора РКС-цепей здесь не рассматривается.

Синтаксис языка РКС позволяет набирать разветвленные логические схемы (рис. 5.6) Такие схемы очень удобны для формирования простых законченных блоков управления тем или иным механизмом, например, охлаждением или включением освещения.

Таймер с задержкой на включение выходного сигнал приведен на рис. 5.6.

Аббревиатура инструкции таймера:

TMR для таймеров с дискретностью Δt = 100 мс и TMRH для таймеров с дискретностью Δt = 1 мс.

Операнд таймера обозначается буквой T.

Общее число таймеров 512 (T0…T511).

Максимальная уставка выдержки времени – практически не ограничена.

Инструкция TMR имеет три входа:

En – вход активизации таймера (Enable);

S1 – номер таймера T0–T511;

S2 – уставка счета выдержки времени (SV – Set Value).

Выходом таймера являются замыкающийся или размыкающийся контакт, указываемый в отдельной цепи и имеющий адрес номера таймера.

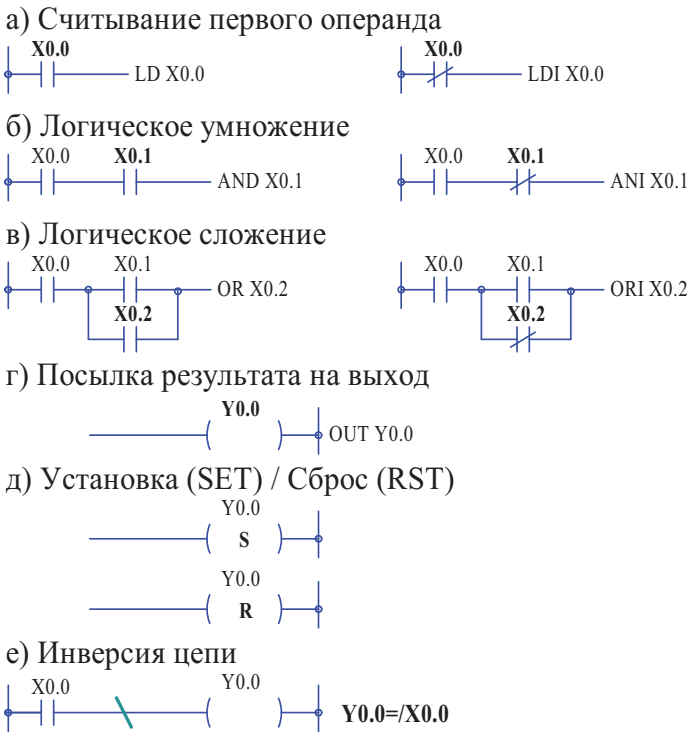


Рис. 5.5. Основные базовые логические команды

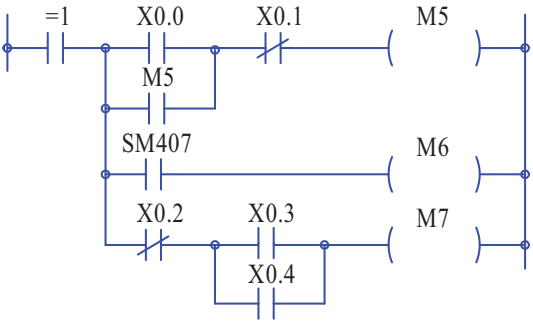


Рис. 5.6. Разветвленная логическая цепь

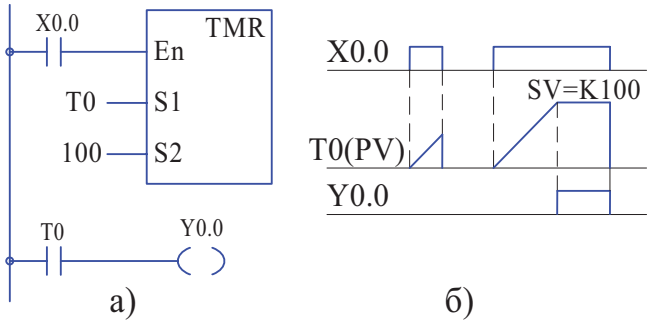


Рис. 5.7. Стандартный таймер:
а – графическое отображение; *б* – диаграмма работы

Если за время активизации таймера по входу En текущее состояние таймера (PV – Previous Value) не достигнет величины уставки, его значение сбрасывается в ноль и отсчет времени начинается с нулевого состояния.

Если за время активизации текущее состояние таймера достигнет величины уставки (SV), то активизируется выходной бит таймера T0…T511 и остается включенным до момента снятия входа активизации.

Данный тип таймера является базой для построения любой временной схемы или алгоритма (см. гл. 2).

На рис. 5.8 показан *накапливающий (ил аккумуляторный) таймер*, обозначаемый аббревиатурой ST0…ST511. Его отличительной особенностью является то, что при прерывании сигнала активизации текущее состояние таймера не сбрасывается и запоминается. При повторной активизации продолжается счет заданной уставки времени и при ее достижении активизируется выходной бит таймера.

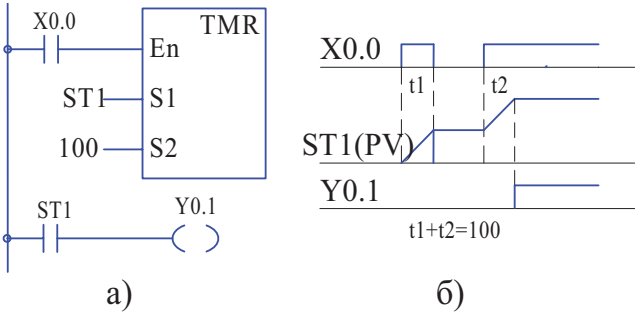


Рис. 5.8. Аккумуляторный таймер:
а – графическое отображение; *б* – диаграмма работы

5.3.2. Функциональные команды

Функциональные команды расширяют возможности контроллера и позволяют простыми средствами формировать таймеры, счетчики, регистры, преобразователи кодов, схемы сравнения многоразрядных кодов, работать со стековой памятью и таблицами, выполнять арифметические операции и многое другое.

Состав функциональных команд резко отличается от контроллера к контроллеру. Рассмотрим наиболее распространенные инструкции, применяемые для автоматизации металлорежущих станков.

Простейший унитарный нереверсивный счетчик (рис. 5.9)

Аббревиатура инструкции счетчика **CNT** (Counter).

Операнд счетчика обозначается буквой С.

Общее число счетчиков 512 (С0…С511).

Максимальная уставка счета: 32767.

Счетчик начинает считать с нулевого значения.

Инструкция CNT имеет три входа:

En – вход активизации счетчика (Enable);

S1 – номер счетчика С0–С511;

S2 – уставка счета (SV – Set Value).

Выходом счетчика являются замыкающийся или размыкающийся контакт, указываемый в отдельной цепи и имеющий адрес номера счетчика.

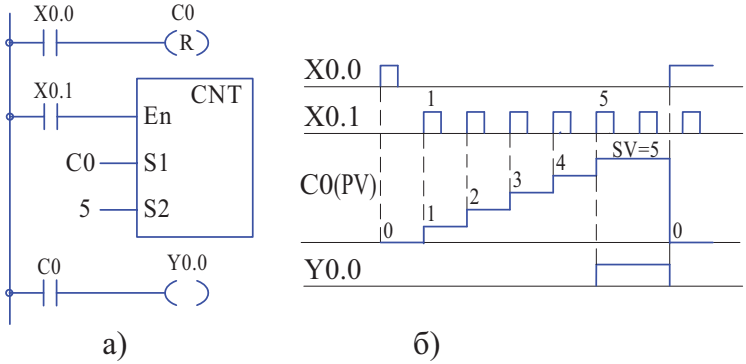


Рис. 5.9. Простейший унитарный счетчик:
а – схема набора; *б* – диаграмма работы

При достижении уставки счета выходной контакт замыкается и остается включенным до момента принудительного сброса. Так же не изменяется текущее состояние счетчика, не реагируя на последующие входные импульсы.

Сброс счетчика в нулевое положение осуществляется внешним сигналом Reset, воздействующим на его катушку. Обращаем внимание, что и контакт и катушка счетчика обозначаются одинаково буквой С с номером счетчика.

32-разрядные счетчики

Счетчики имеют аббревиатуру DCNT (Double Counter).

Операнд счетчика обозначается буквой HC.

Общее число счетчиков 255 (HC0…HC255). Такие счетчики разделяются на три группы:

(HC0…HC63) – реверсивные (Up/Down) счетчики;

(HC64…HC199) – нереверсивные (Up) счетчики;

(HC200…HC255) – высокоскоростные реверсивные (Up/Down) счетчики.

Максимальная уставка счета: двойное слово, т. е. 2 147 483 648.

Нереверсивные счетчики начинают считать с нулевого значения.

Используя инструкции сравнения можно получить доступ к текущему состоянию счетчика.

Инструкция CNT имеет три входа:

En – вход активизации счетчика (Enable);

S1 – номер счетчика HC0–HC255;

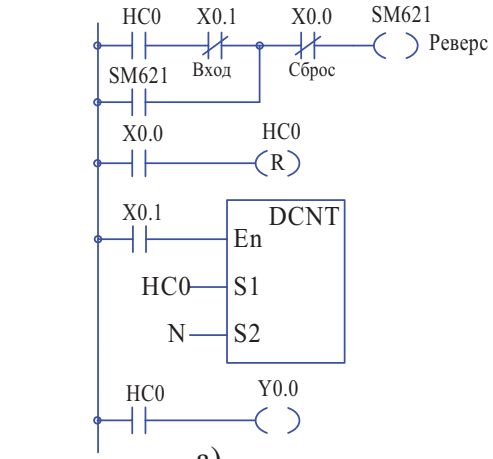
S2 – уставка счета.

Выходом счетчика являются замыкающийся или размыкающийся контакт, указываемый в отдельной цепи и имеющий адрес номера счетчика.

На рис. 5.10 приведен пример счетчика, работающего с положительной уставкой счета, равной 5 и автоматическим реверсом при достижении уставки.

По умолчанию или после принудительного сброса счетчик считает на сложение (Up). Выходной бит счетчика активизируется при достижении текущим состоянием счетчика положительной уставки PV = SV(= 5) и остается включенным до нарушения этого равенства или момента принудительного сброса. Так же не изменяется текущее состояние счетчика.

В данном *учебном* алгоритме при достижении уставки счета организован автоматический реверс. Реверс счета осуществляется активизацией служебных сигналов (SV621…SV684), однозначно привязанного к номеру счетчика (см. табл. рис. 5.9). Счетчик начинает считать на вычитание и после перехода через ноль продолжает считать в отрицательной зоне.



Сигналы реверса

Счетчик	Реверс
HC0	SV621
HC1	SV622
HC2	SV623
HC3	SV624
⋮	⋮
HC63	SV684

б)

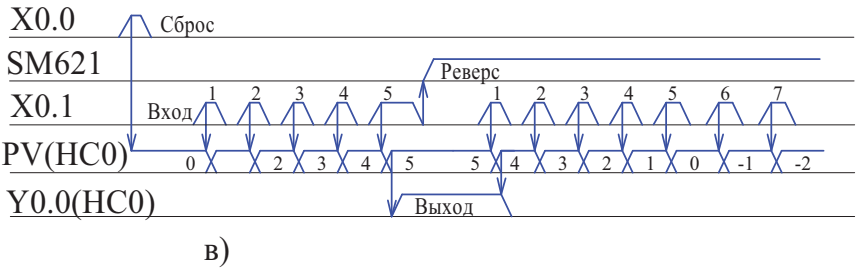


Рис. 5.10. Простейший реверсивный счетчик:
а – схема набора; *б* – параметры реверса; *в* – диаграмма работы

Работа счетчика с отрицательными значениями уставок автором не изучалась. При необходимости, это нужно делать на стенде перед разработкой реальных программ. Теоретически, по исходной технической документации понять это вряд ли удастся.

Сброс счетчика в нулевое положение осуществляется внешним сигналом Reset, воздействующим на его катушку. Обращаем внимание, что и контакт и катушка счетчика обозначаются одинаково буквами HC с номером счетчика.

Базируясь на опыте разработки предыдущего счетчика легко организовать счетчик с автоматическим сбросом для любой уставки счета (рис. 5.11).

Счетчик, построенный по данному алгоритму, после достижения заданной уставки автоматически возвращается в исходное нулевое состояние и цикл счета может быть повторен.

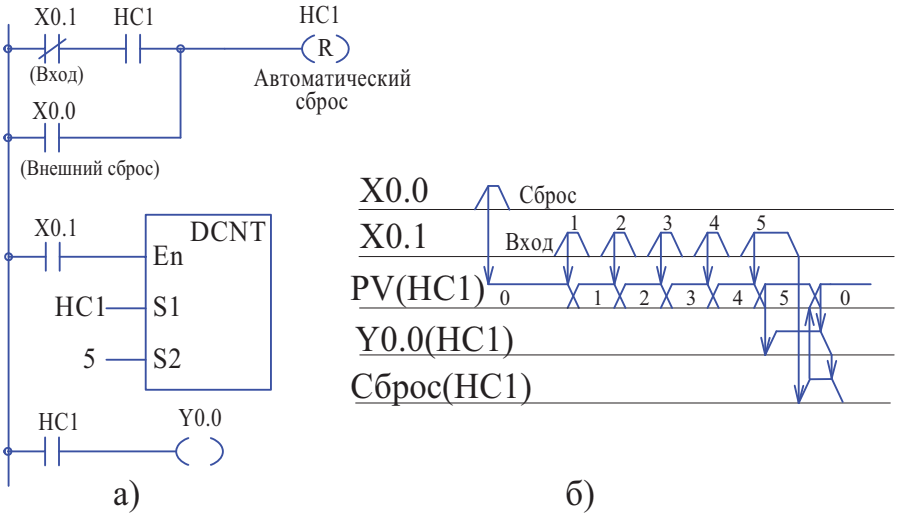


Рис. 5.11. Счетчик с самосбросом:
а – топология набора; *б* – диаграмма работы

Принципы построения высокоскоростных счетчиков и примеры их применения будут рассмотрены ниже.

Формирователи тактов

Формирователи тактов используются для формирования импульсов длительностью в один вычислительный цикл (такт или скан) по переднему, заднему или по обоим фронтам входного сигнала. С помощью тактирования входных сигналов создаются самые разнообразные специальные алгоритмы.

Как правило в синтаксисе языка предусматривается несколько способов формирования сканов.

*Инструкции **PLS** и **PLF*** (рис. 5.12)

Обращаем внимание, что адрес выходного тактированного сигнала указывается на выходе аппликации инструкции, а его рабочий контакт для последующего использования в отдельной цепи.

На рис. 5.13 приведены варианты формирования тактовых сигналов по результатам *вычисления групповых логических цепей*. Для этой цели в конечной логической цепи используются стрелочные символы:

↑ – формирование такта при условии логической единицы;

↓ – формирование такта при условии логического нуля.

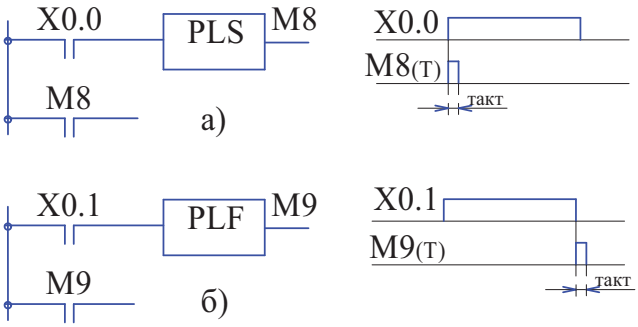


Рис. 5.12. Формирователи тактов:
а – PLS – по переднему фронту; *б* – PLF – по заднему фронту

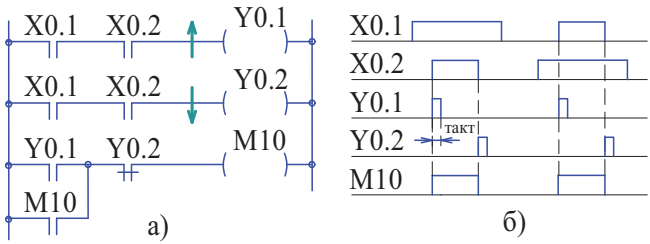


Рис. 5.13 Групповые формирователи тактов:
а – РКС-алгоритм; *б* – диаграмма работы

На выходе Y0.1 формируется тактовый сигнал при условии формирования переднего фронта результирующего сигнала, определяемого логическим уравнением (X0.1*X0.2) = 1, а на выходе Y0.2 при условии формирования заднего фронта того же уравнения, т. е. при (X0.1*X0.2) = 0. Для примера, эти сигналы управляют памятью, реализованной на ячейке M10.

Синтаксис языка данного контроллера позволяет так же формировать сканы непосредственно в битовых операндах (рис. 5.14).

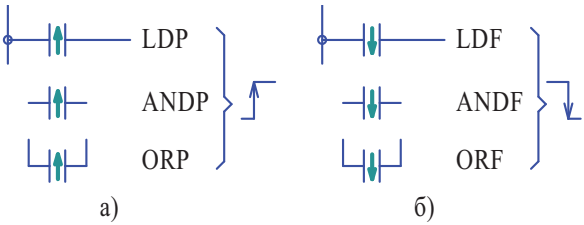


Рис. 5.14. Формирование тактов в битовых операндах

При необходимости, тактирование сигналов легко сделать при помощи *базовых* инструкций. На рис. 5.15 приведены классические алгоритмы, пригодные для любого типа контроллера:

а) формирование такта Y(т) по переднему фронту входного сигнала Y;
б) формирование такта Y(т) по заднему фронту входного сигнала Y;
в) формирование такта Y(т) по обоим фронтам входного сигнала Y.

Данные алгоритмы требуют использования промежуточных сигналов α.

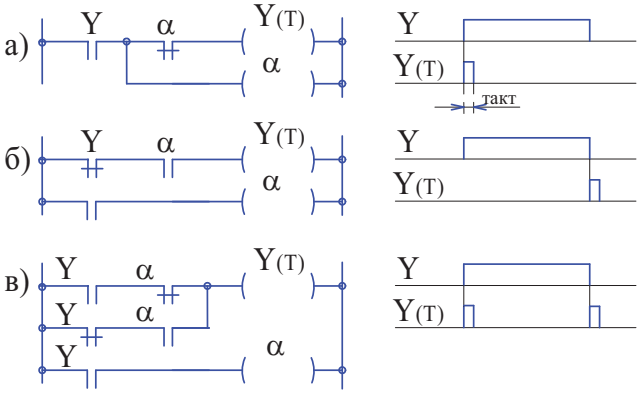


Рис. 5.15. Формирователи тактов на базовых командах

Инструкции пересылки и обработки данных (рис. 5.16)

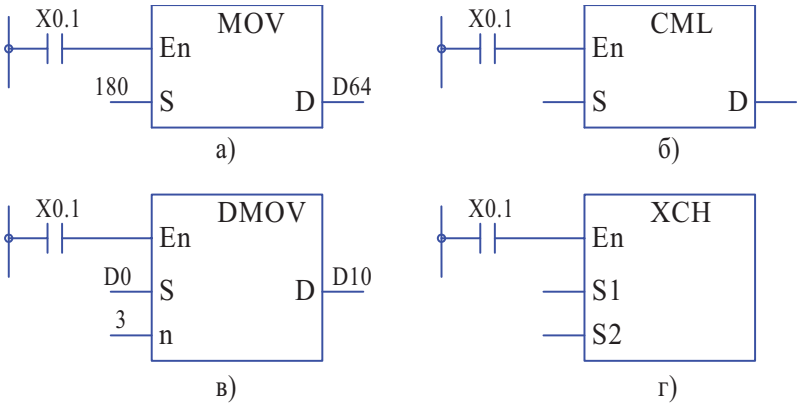


Рис. 5.16. Команды пересылки и обработки данных

К данному классу команд относятся:

а) пересылка слов S → D
   MOV 16-разрядные слова;
   DMOV 32-разрядные слова;

б) групповая пересылка n-слов S → D здесь, при n = 3
   D0 → D10
   D1 → D11;
   D2 → D12;

в) обмен данными S1 ↔ S2
   XCH 16-разрядные слова;
   DXCH 32-разрядные слова;

г) побитная инверсия данных D = /S
   CML 16-разрядные слова;
   DCML 32-разрядные слова.

Инструкции преобразования кодов (рис. 5.17)

К данному классу команд относятся:

а) BCD-преобразование двоичного кода в двоично-десятичный, т. е. S(BIN) → D();

б) BIN-преобразование двоично-десятичного кода в двоичный, т. е. S(BCD) → D();

в) GRY-преобразование двоичного кода в код Грея, т. е. S(BIN) → D(Grey);

г) GBIN-преобразование кода Грея в двоичный код, т. е. S(Grey) → D(BIN).

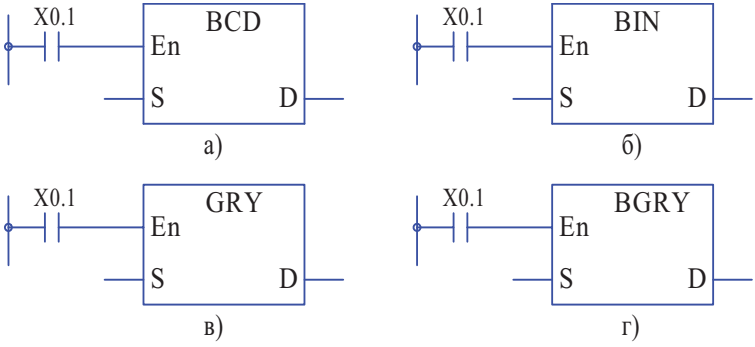


Рис. 5.17. Команды преобразования кодов

Инструкции инкремента и декремента (рис. 5.18)

Инструкция INC – инкремент при подаче на вход En тактированного входного сигнала выполняет операцию увеличения на единицу содержимого рабочего регистра, т. е. D = D + 1.

Инструкция DEC – декремент при подаче на вход En тактированного входного сигнала выполняет операцию уменьшения на единицу содержимого рабочего регистра, т. е. D = D – 1.

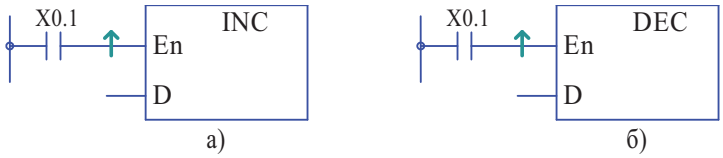


Рис. 5.18. Команды инкремента (а) и декремента (б)

Инструкции сравнения (рис. 5.19)

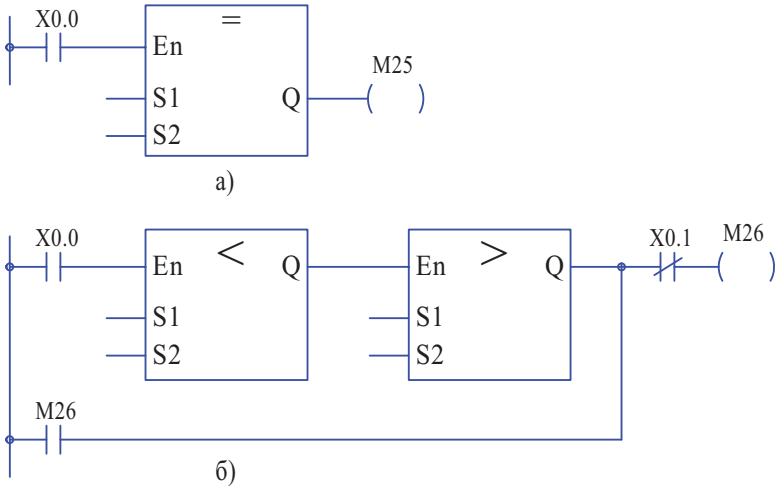


Рис. 5.19. Инструкции сравнения:
а – топология команды; *б* – пример разветвленной цепи

Инструкции сравнения имеют три входа:

En – вход активизации (Enable);

S1 – 1-е сравниваемое слово;

S2 – 2-е сравниваемое слово.

Выходом инструкции Q является катушка реле, активизируемая при выполнении условий сравнения (табл. 5.2).

Таблица 5.2

Инструкции сравнения

Инструкция	Аббревиатура	Действие
Равно	=, D=	S1 = S2
Не равно	< >, D< >	S1 ≠ S2
Больше	>, D>	S1 > S2
Больше или Равно	>=, D>=	S1 ≥ S2
Меньше	<, D<	S1 < S2
Меньше или Равно	<=, D<=	S1 ≤ S2

Характерной особенностью синтаксиса языка является возможность использования команд сравнения в качестве битовых инструкций в разветвленных базовых цепях. Так, например, РКС-алгоритм рис. 5.19, *б* реализует следующее логическое уравнение:

$$M26 = (X0.0*(S1 < S2)*(S1 > S2) + M26)*/X0.1.$$

Арифметические инструкции (рис. 5.20)

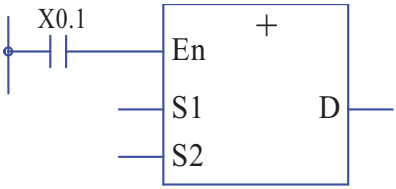


Рис. 5.20. Топология арифметических инструкций

Арифметические инструкции имеют три входа:

En – вход активизации (Enable);

S1 – 1-й операнд;

S2 – 2-й операнд.

Адрес результат вычисления записывается на выводе с адресом D.

Арифметические инструкции представлены в табл. 5.3 и выполняются в двоичном коде. В зависимости от аббревиатуры арифметические вычисления могут производиться как с 16-разрядными, так и с двойными словами.

Таблица 5.3

Арифметические инструкции

Инструкция	Аббревиатура	Действие	Примечание
ADD – сложение	+, D +	S1 + S2 = D	ст. бит: 0 – плюс 1 – минус
SUB – вычитание	--, D --	S1 – S2 = D	ст. бит: 0 – плюс 1 – минус
MUL – умножение	*, D *	S1 * S2 = D	ст. бит: 0 – плюс 1 – минус
DIV – деление	/, D /	S2 : S2 = D	ст. бит: 0 – плюс 1 – минус, остаток → (D+1)

Регистры

В синтаксисе языка имеется большое число регистровых и сдвиговых инструкций. При необходимости следует обратиться к сопроводительной технической документации.

На рис. 5.21 приведен классический кольцевой сдвиговый регистр вправо ROR. В варианте с использованием флага регистр имеет аббревиатуру RCR.

Регистр имеет три входа:

En – вход активизации (Enable);

D – адрес регистра;

n – число сдвигаемых бит.

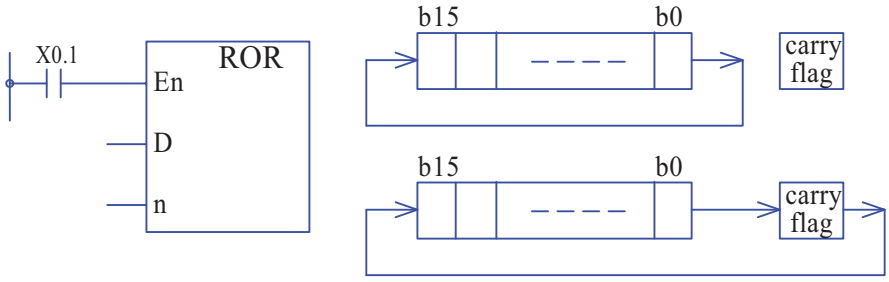


Рис. 5.21. Кольцевой регистр со сдвигом информации вправо

Аналогично строятся регистры со сдвигом информации влево:

ROL – без использования флага;

RCL – с флагом переноса.

Шифраторы и дешифраторы

Дешифратор DECO (рис. 5.22) имеет три входа и один выход:

En – вход активизации (Enable);

S – начальный адрес дешифрируемого слова;

n – число дешифрируемых разрядов (бит) входного слова S;

D – начальный адрес выходного слова, в бит которого записывается результат выполнения операции.

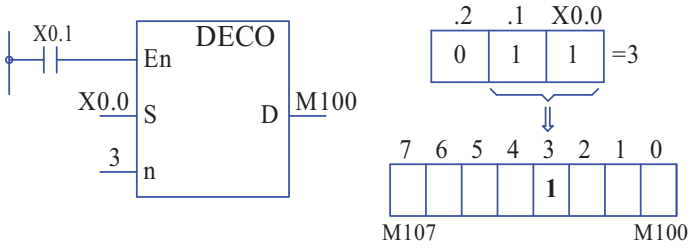


Рис. 5.22. Инструкции дешифратора

Шифратор ENCO (рис. 5.23) также имеет три входа и один выход:

En – вход активизации (Enable);

S – начальный адрес шифрируемого слова;

n – шифрируемый разряд (бит) входного слова S;

D – адрес выходного слова, в которое записывается результат выполнения операции.

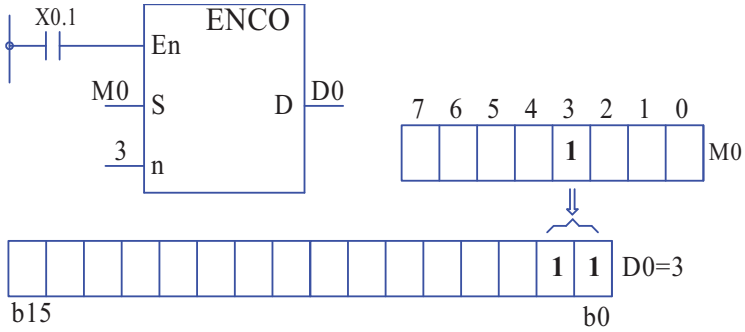


Рис. 5.23. Инструкции шифратора

Инструкция группового сброса ZRST (рис. 5.24) предназначена для одновременного группового сброса блока данных и имеет три входа:

En – вход активизации (Enable);
D1 – начальный адрес блока данных;
D2 – конечный адрес блока.

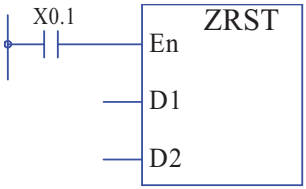


Рис. 5.24. Инструкция группового сброса

Сбрасываться могут операнды с индексами Y, M, S, T, C, D, E, F.
Обязательные условия:

– операнды, указываемые на входах начального D1 и конечного D2 адресов, должны иметь одинаковую структуру;
– конечный адрес должен иметь больший номер, чем начальный, т. е. D2 > D1.

5.3.3. Высокоскоростные инструкции

DPLSY – HSpeed pulse Output
DPLSR – HSpeed pulse Output
DPWM – Pulse width modulation
DJOG – Jog Output
DZRN – Zero Return
DPLSV – Adjustable Pulse Output
DDRVI – Relative Position Control
DDRVA – Absolute Position Control
DHSZ – HSpeed Input Zone Comparison
DCNT – 32-bit Counter
DHSCS – Setting HS-comparison
DHSCR – Resetting HS-comparison
DSPD-Detecting Speed
DRST – Devise to Reset

Перечень используемых при программировании служебных команд (параметров) приведен ниже в табл. 5.4 и 5.5.

Таблица 5.4

Служебные команды SM

Назначение параметра	Номер параметра по выходным каналам			
	Y0.0	Y0.1	Y0.2	Y0.3
Outputting (Busy)	SM460		SM480	
Output is complete	SM461		SM481	
Reversing Output Direction		SM462		SM482
Stopping the Output	SM463		SM483	
Enabling the «+» max. value	SM464		SM484	
Alarm: «+» Limit	SM465		SM485	
Enabling the «-» max. value	SM466		SM486	
Alarm: «+» Limit	SM467		SM487	
Enabling fixed ramp Up/Down	SM469		SM489	
Auto-reset for Output is complete	SM470		SM490	
Interrupt I500/I501 when pulses end	SM471		SM491	
Outputting		SM472		SM492
Output is complete		SM473		SM493
Stopping the Output		SM474		SM494
Auto-reset for Output is complete		SM475		SM495
Output Stop when disabled	SM476	SM477	SM496	SM497
Change the target position	SM478		SM498	
Change the target position		SM479		SM499

Таблица 5.5

Служебные команды SR

Назначение параметра	Номер параметра по выходным каналам			
	Y0.0	Y0.1	Y0.2	Y0.3
Position	SR460, 461		SR480, 481	
Output Mode	SR462		SR482	
Starting / Ending «f»	SR463		SR483	
Acceleration Time	SR464		SR484	
Deceleration Time	SR465		SR485	
Jog Frequency	SR466		SR486	
Numerator	SR468		SR488	
Denominator	SR469		SR489	
Position of the Machine	SR470, 471		SR480, 481	
Target frequency for fixed Slope	SR472, 473		SR482, 483	
Position		SR474, 475		SR484, 485
Starting / Ending «f»		SR476		SR486
Acceleration / Deceleration		SR477		SR487
Backlash Compensation	SR478	SR479	SR488	SR489
Current Output Speed (Hz)	SR610, 611		SR612, 613	

Инструкция генерирования пачки импульсов DPLSY (HSpeed Pulse Output), предназначенная для выдачи на дифференциальный выход контроллера заданных числа и частоты импульсов амплитудой 5 В *без формирования разгона и замедления*.

Инструкция (рис. 5.25) имеет три входа:

En – вход активизации (Enable);

S1 – Pulse Output Frequency (частота выходных импульсов);

S2 – Number of pulses to Output (Число импульсов).

На выходе инструкции D задается адрес дифференциального выхода (в рассматриваемом контроллере имеется 4 дифференциальных выхода Y0.0–Y0.3).

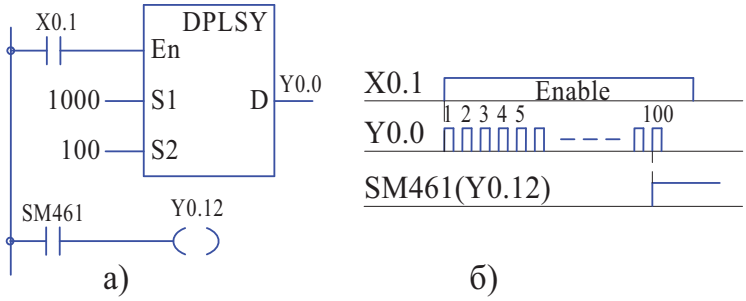


Рис. 5.25. Инструкция DPLSY

По окончании выдачи импульсов активизируется служебный бит SM461 (Output is complete – выдача завершена). В данном примере генерируется 100 импульсов с частотой 1000 Гц.

Инструкция генерирования пачки импульсов DPLSR (HSpeed Pulse Output), предназначенная для выдачи на дифференциальный выход контроллера заданных числа и частоты импульсов амплитудой 5 В *с формированием разгона и замедления*.

Инструкция (рис. 5.26) имеет четыре входа:

En – вход активизации (Enable);

S1 – Pulse Output Frequency (частота выходных импульсов);

S2 – Naumber of pulses to Output (Число импульсов);

S3 – Ramp Up / Down Time (Время разгона и торможения).

На выходе инструкции D задается адрес дифференциального выхода.

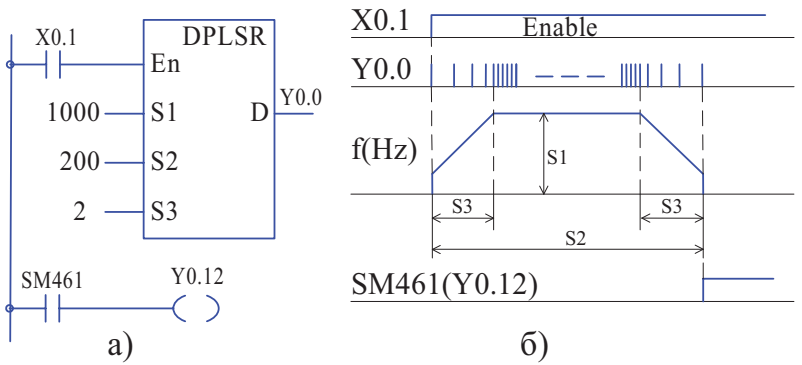


Рис. 5.26. Инструкция DPLSR

По окончании выдачи импульсов активизируется служебный бит SM461 (Output is complete – выдача завершена). В данном примере генерируется 200 импульсов с частотой 1000 Гц и временем разгона и торможения 2 мс.

Инструкция генератора импульсов с изменяемой скважностью PWM (Pulse Width Modulation), предназначенная для выдачи на дифференциальный выход контроллера непрерывного потока импульсов заданных длительности и периода амплитудой 5 В.

Инструкция (рис. 5.27) имеет три входа:

En – вход активизации (Enable);

S1 – Pulse Output Width (длительность импульса);

S2 – Pulse Output Cycle (Период).

На выходе инструкции D задается адрес дифференциального выхода.

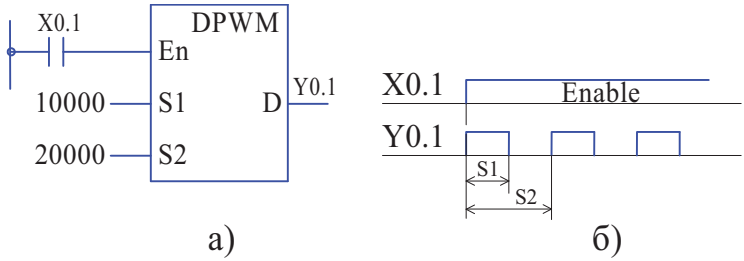


Рис. 5.27. Инструкция DPWM

В данном примере, при подаче разрешающего сигнала, на выходе D генерируются импульсы длительностью 10000 мкс с периодом 20000 мкс.

Инструкция формирования безразмерного задания DJOG (Jog – толчок), предназначена для управления через дифференциальный выход контроллера непрерывным безразмерным вращением двигателя, управляемого от регулируемого привода.

Инструкция (рис. 5.28) имеет четыре входа:

En – вход активизации (Enable);

S1 – Ramp Up Time (Время ускорения);

S2 – Target Output Frequency (Рабочая частота задания);

S3 – Ramp Down Time (Время замедления)

и два выхода:

D1 – Pulse Output Device (Рабочий выход);

D2 – Auxiliary Output Device (Дополнительный выход).

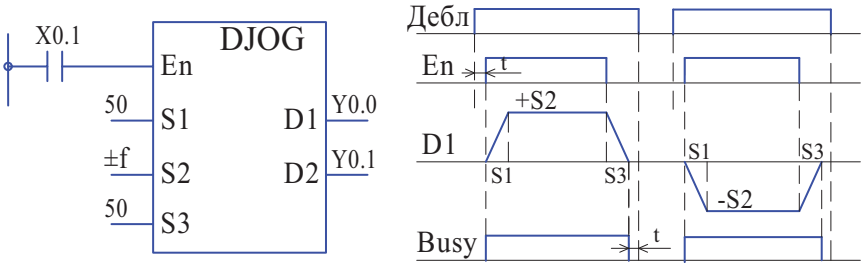


Рис. 5.28. Инструкция JOG

Дискретность задания времени ускорения и замедления $\Delta t = 10$ ms.

При использовании инструкции DJOG в программе электроавтоматики, в зависимости от задачи, могут быть использованы следующие служебные биты (параметры):

SR462 / SR482 = 0 (Puls + Dir);

= 1 (A / B);

SM462 / SM482 – Reverse (Реверс);

SM460 / SM480 – Busy (Канал занят);

SM461 / SM481 – Complete (Выполнено);

SM463 / SM483 – Отключение выхода;

SR460 / SR480 – Реальное перемещение.

Важно! Работа инструкции никак не синхронизирована с работой аппаратного электропривода. При подаче сигнала активизации Enable инструкция сразу начинает отрабатывать заданное на ее входах перемещение, не зависимо от реального состояния управляемого электропривода. Если электропривод при этом не деблокирован или будет деблокирован с задержкой, то произойдет

сбой. Это обстоятельство следует **учитывать** при разработке алгоритма управления.

На рис. 5.29 приведена идеология подключения дифференциальных выходов контроллера ко входам задания регулируемого электропривода типа E100.

В приводе установлены следующие ключевые параметры:

(P1–00) = 1, что соответствует установке режима позиционирования;

(P1–02) = 0, что соответствует принципу управления «Pulse + Dir», т. е. «Импульсы + Направление».

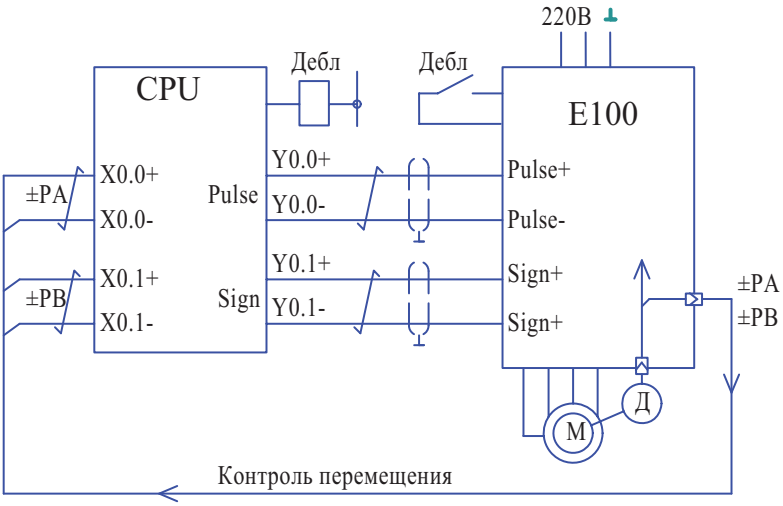


Рис. 5.29. Идеология подключения дифференциальных выходов PLC к электроприводу

При разработке программы электроавтоматики необходимо обеспечить следующую последовательность управления:

- установка параметров (атрибутов) инструкции;
- деблокировка электропривода;
- активизация En инструкции DJOG и подача на ее вход рабочей частоты.

Отрицательное направление рекомендуется формировать умножением положительного задания на −1;

- отключение активизации En, замедление до остановки;
- отключение деблокировки электропривода после пропадания сигнала Busy.

Безразмерное перемещение будет осуществляться до момента отключения сигнала активизации инструкции DJOG.

Если требуется организовать контроль величины перемещения, то следует использовать дифференциальные входы контроллера и выходные транзитные TTL-сигналы от электропривода. Естественно, в программе электроавтоматики нужна нормализация сигналов в соответствии с кинематической схемой и параметрами выходных сигналов привода.

Инструкция формирования размерного задания DDRVI (Relative Position Control), предназначена для управления через дифференциальный выход контроллера размерным перемещением оси (вращением двигателя, управляемого от регулируемого привода).

Инструкция (рис. 5.30) имеет три входа:

En – вход активизации (Enable);

S1 – Number of Output Pulses (Число импульсов управления);

S2 – Pulse Output Frequency (Рабочая частота задания)

и два выхода:

D1 – Pulse Output Device (Выход задания импульсов управления);

D2 – Pulse Direction Output Device (Выход задания направления).

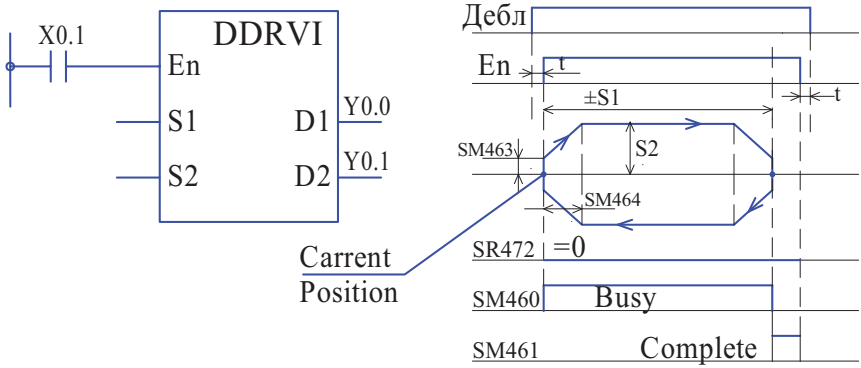


Рис. 5.30. Инструкция DDRVI

Схема подключения, установка параметров и особые условия организации алгоритма управления одинаковы с инструкцией DJOG. Величина и скорость размерного перемещения задаются на панели оператора и обрабатываются программой электроавтоматики. Еще раз подчеркнем, что при этом должна соблюдаться последовательность работы, показанная на диаграмме рис. 5.30.

Формально, при применении инструкции DDRVI можно не использовать обратную связь по положению, так как она работает аналогично управлению разомкнутого шагового привода. Однако организация контроля необходима по двум причинам:

– необходимости вывода на панель оператора информации о фактическом перемещении и
– реальной проверке выхода механизма в заданную позицию.

Такой контроль осуществляется при помощи инструкции быстрого счетчика (DCNT и HCn), на вход которого через дифференциальные входы контроллера подаются транзитные TTL-импульсы с электропривода. Схема подключения та же (см. рис. 5.29). Все реализуется программно.

Высокоскоростные аппаратные счетчики **HCn** (High Speed Counter)

Высокоскоростные счетчики организованы **аппаратно** и имеют аббревиатуру HC200…HC255. Номера счетчиков жестко привязаны к дифференциальным входам контроллера (в рассматриваемом PLC имеется 4 входа X0.0–X0.3) и разбиты на группы по **три** счетчика, см. табл. 5.6.

Таблица 5.6

Быстрые счетчики

Счетчик	Аналоговые входы						
	X0.0	**X0.1**	**X0.2**	**X0.3**	X0.4	X0.5	и т. д.
HC200	P						
HC201	P	D					
HC202	A	B					
HC204			P				
HC205			P	D			
HC206			A	B			
и т.д.							

Назначение (тип) входов следующее:

P – Single-phase pulse input;
D – Direction Signal input;
A/B – Two-phase two-input;
R – Reset Signal input.

Из одной группы может быть использован только один счетчик.

Счетчики типа A/B автоматически определяют направления счета.

Пояснение. Счетчик HC200 считает обезличенную последовательность импульсов по входу X0.0. Реверс следует организовывать, например, сигналами привода. Счетчик HC201 по входу X0.0 считает обезличенную последовательность импульсов, а вход X0.1 определяет направление счета. Счетчик HC202 по входам X0.0/X0.1 считает сдвинутые на 90 градусов стандартные TTL-сигналы A/B и автоматически определяет направление счета.

В Ladder-диаграмме быстрый счетчик не изображается в виде функционального блока, а присутствует *только в виде адресов*. Выход счетчика с адресом HCn может использоваться как вход функциональной инструкции, как катушка реле или битовый операнд.

В таблице 5.7 приведены специальные параметры быстрых счетчиков.

Таблица 5.7

Параметры быстрых счетчиков

Counter	Up / Down Function			Starting the Reset function	Reversing Direction	Counting Mode
HC200	SM300	R / W	Show / Set	SM291	нет	SR190
HC201	SM301	R	Show	SM291	SM281	SR190
HC202	SM302	R	Show	SM291	нет	SR190
HC204	SM304	R / W	Show / Set	SM292	нет	SR191
HC205	SM305	R	Show	SM292	SM282	SR191
HC206	SM306	R	Show	SM292	нет	SR191

Назначение параметра SR190/SR191:

= 1 – Time frequency input by default

= 2 – Удвоение частоты импульсов

= 4 – Учетверение частоты импульсов

Важно! Если в проекте не используется контроль перемещения, то счетчик можно не использовать. В случае необходимости контроля используются высокоскоростные программные инструкции DCNT или DHSCS.

Высокоскоростные программные счетчики **DCNT** и **DHSCS**

Инструкция счетчика DCNT предназначена для сравнения текущего состояния высокоскоростного счетчика с заданной уставкой и имеет три входа (рис. 5.31, *а*):

En – вход активизации (Enable);

S1 – Counter Value (Текущее состояние счетчика HCn);

S2 – Setting Value (Уставка счетчика).

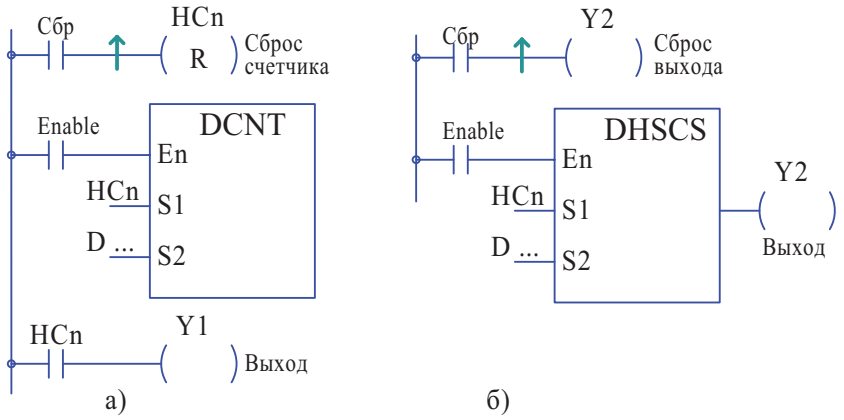


Рис. 5.31. Быстрые программные счетчики DCNT и DHSCS

При достижении текущим состоянием счетчика значения уставки активизируется выходной бит счетчика HCn, обрабатываемый в программе электроавтоматики отдельной строкой. Важно также помнить, что счетчик работает только при наличии сигнала активизации En.

Сброс счетчика осуществляется выключением его катушки HCn, также программируемой отдельной строкой.

Еще раз обращаем внимание, что счетчик HCn, это аппаратный счетчик и как такового его в программе нет. *Под одним и тем же именем* используются выходная катушки, выходной бит и текущее состояние.

Инструкция счетчика DHSCS также предназначена для сравнения текущего состояния высокоскоростного счетчика с заданной уставкой, но в быстрой логике.

Инструкция имеет те же три входа (рис. 5.31, *б*):

En – вход активизации (Enable);

S1 – Counter Value (Текущее состояние счетчика HCn);

S2 – Setting Value (Уставка счетчика) и катушку выхода, единичное состояние которой мгновенно запоминается.

Данную инструкцию можно использовать при управлении механизмами, где не требуется абсолютно точная остановка. В этом случае привод управляется по аналоговому заданию и дискретными сигналами направления вращения (рис. 5.32), требуется лишь максимально быстрый останов. Например, при перемещении рамы пилы при резании и возврате в исходное положение, не имеет большого значения, насколько переместится рама после завершения процесса и выключения команды на перемещение. Будет это 0,1 мм или 0,5 мм, значения

не имеет. Нужно просто включить реверс или выключить привод. Вариант алгоритма такого управления с использованием обоих типов инструкций обработки сигналов счетчиков приведен на диаграмме рис. 5.33, *б*):

- привод программируется на аналоговое управление скоростью, т. е. (P1-00) = 0, что соответствует установке управления скоростью, (P1-08) = 1, что соответствует управлению +/−10В по входу AT1;
- задание на привод осуществляется от модуля аналоговых выходов;
- деблокировка привода и реверс осуществляются дискретными входными сигналами ON и Реверс;
- органы управления приводом (задание величины и скорости перемещения, команды Вперед, Назад и Стоп) формируются на панели оператора;
- для контроля за фактической величиной перемещения используются транзитные TTL-сигналы из привода. Используется счетчик HC204. Они же могут использоваться для индикации фактического перемещения на панели оператора;
- останов перемещения при движении Вперед осуществляется при помощи инструкции DCNT, формируемой бит HC204 при достижении уставки. При этом отключается деблокировка привода и он останавливается с перебегом, определяемым быстродействием аппаратов управления и самого привода;
- останов перемещения при движении Назад осуществляется при помощи инструкции DHSCS, мгновенно активизирующей, не дожидаясь конца вычислительного цикла, выходную катушку при HC204 = 0 (уставка инструкции), отключающую деблокировку привода. Таким образом максимально сокращается время перебега.

Важно! При реализации таких решений следует проанализировать возможность постоянного накопления ошибки, связанной с перебегом и принять меры по ее обнулению.

Инструкции определения зон перемещения DHSZ и DZ

Инструкция DHSZ (HSpeed input Zone Comparisson) по информации счетчика HCn позволяет определить три зоны величины перемещения в зависимости от заданных уставок (рис. 5.33, *а*):

- меньше минимальной уставки (здесь Y0.10),
- нахождение в заданной зоне (здесь Y0.11) и
- больше допустимой уставки (здесь Y0.12).

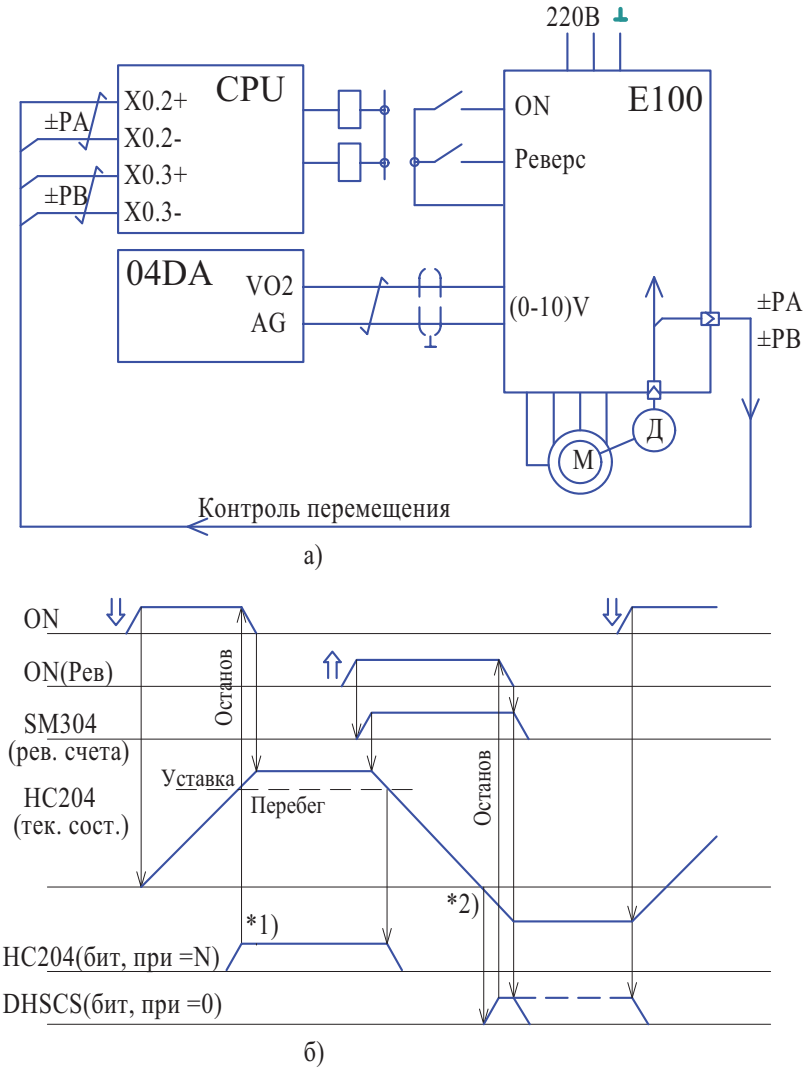


Рис. 5.32. Аналоговое управление приводом

Назначение выводов инструкции следующее:

S1 – Counter Number (номер счетчика);

S2 – Lower Zone (нижняя граница);

S3 – Upper Zone (верхняя граница);

D – Comparison Result (начальный адрес результата).

Универсальная инструкция **DZ** предназначена для сравнения различных величин с формированием одного битового сигнала при выполнении условий сравнения (рис. 5.33, *б*).

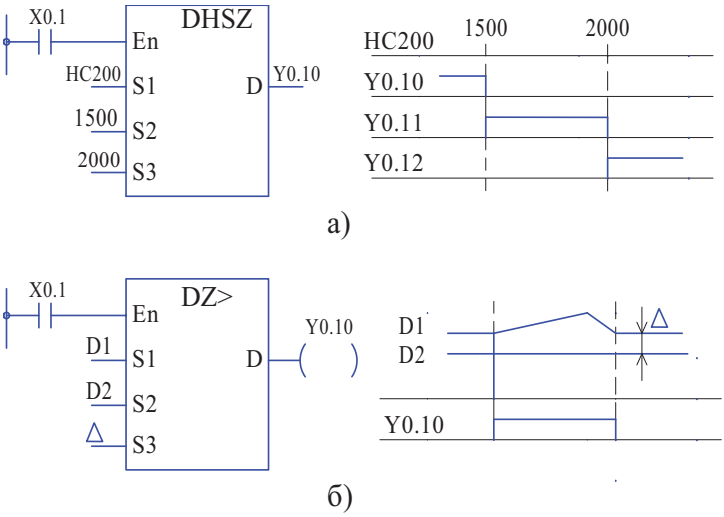


Рис. 5.33. Инструкция определения зон скорости

Доступные варианты сравнения:

DZ < – сравнение на меньшее значение;
DZ > – сравнение на большее значение;
DZ =< – сравнение на меньше или равно;
DZ => – сравнение на больше или равно;
DZ < > – сравнение на неравенство.

Назначение выводов инструкции следующее:

S1 – 1-е сравниваемое значение;
S2 – 2-е сравниваемое значение;
S3 – уставка сравнения;
D – катушка вывода результата сравнения.

Данная инструкция *не является* быстродействующей.

5.4. Пример проектирования электроавтоматики

Ниже рассматривается фрагмент применения контроллера AS324 для управления подачей заготовки автоматической пилы фирмы МЕБА.

Подробный проект управления пилой приведен в [64].

Разработка проекта выполняется в следующей последовательности:

1. Изучение объекта управления, составление технического задания, выбор основных комплектующих изделий.

Принимаем решение:

- программируемый контроллер AS324;
- электропривод подачи заготовки типа E100;
- управление от панели оператора типа DOP-110WS;
- режимы работы: наладочный ручной и непрерывный с заданием размера перемещения.

2. Изучение аппаратного подключения комплектующих, системы параметров электропривода, синтаксиса языка программирования контроллера и панели оператора, организации связи.

3. Разработка принципиальной электрической схемы и органов управления.

4. Конфигурация контроллера, разработка программы электроавтоматики, ввод программы в память и отладка.

Очевидно, что в полном проекте электрооборудования разрабатываются также чертежи на изготовление электрошкафов, расположения и разводок электрических проводов и кабелей, разрабатываются различного рода спецификации, конструктивные чертежи, сопроводительная документация и другое. Эти вопросы здесь не рассматриваются.

Ниже приведены фрагменты принципиальной схемы, относящиеся к управлению подачей заготовки.

Рис. 5.34. Схема подключения электропривода E100 фирмы Sinea.

Рис. 5.35. Подключение дискретных и дифференциальных входов.

Рис. 5.36. Подключение дискретных и дифференциальных выходов.

Рис. 5.37. Эскиз панели управления.

Управление приводом осуществляется по принципу «Импульсы плюс Направление» и задается параметрами.

Программирование (набор программы) осуществляется на персональном компьютере, на который необходимо поставить рекомендованное фирмой-изготовителем программное обеспечение. Перед началом работы следует изучить процедурные вопросы конфигурирования проекта и процедуры набора программы. В процессе конфигурации автоматически назначаются адреса входов и выходов.

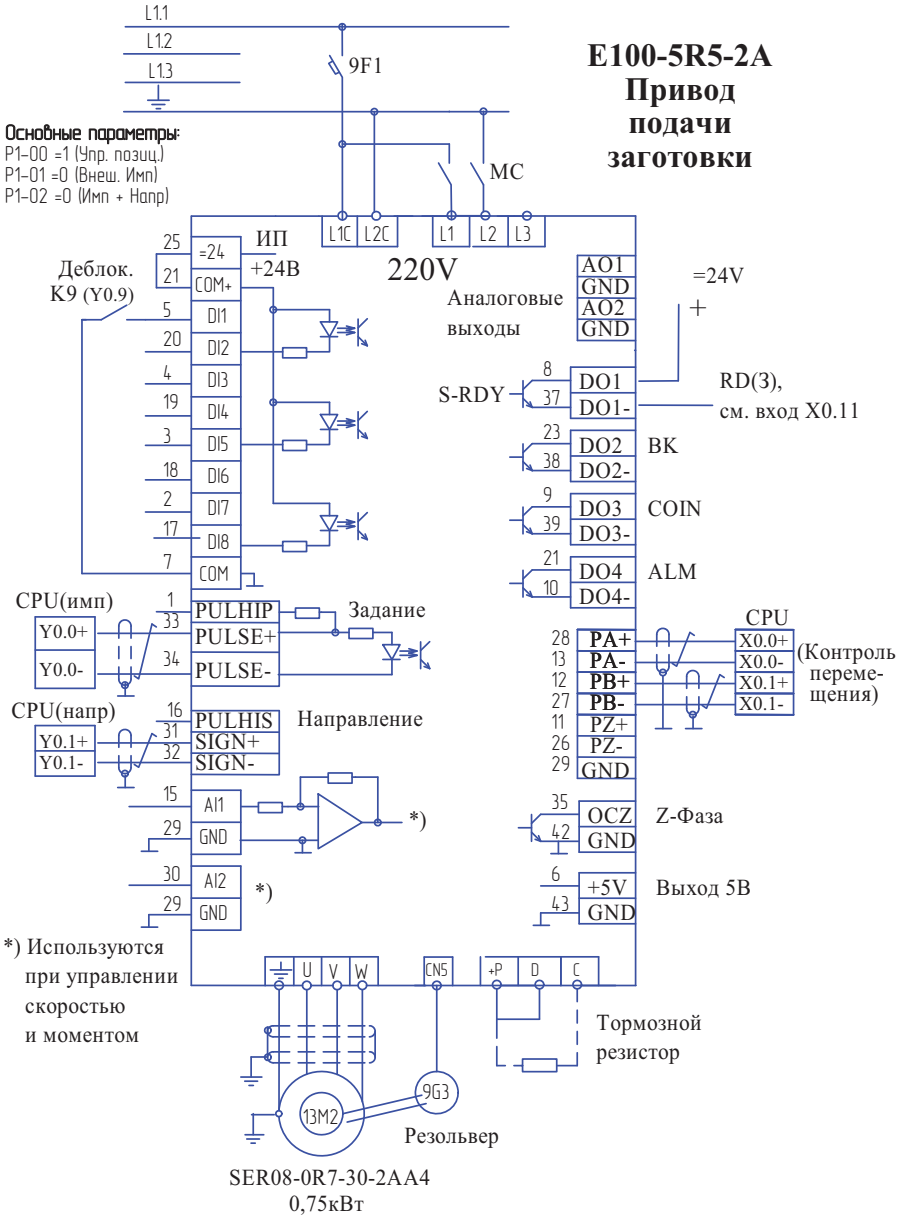
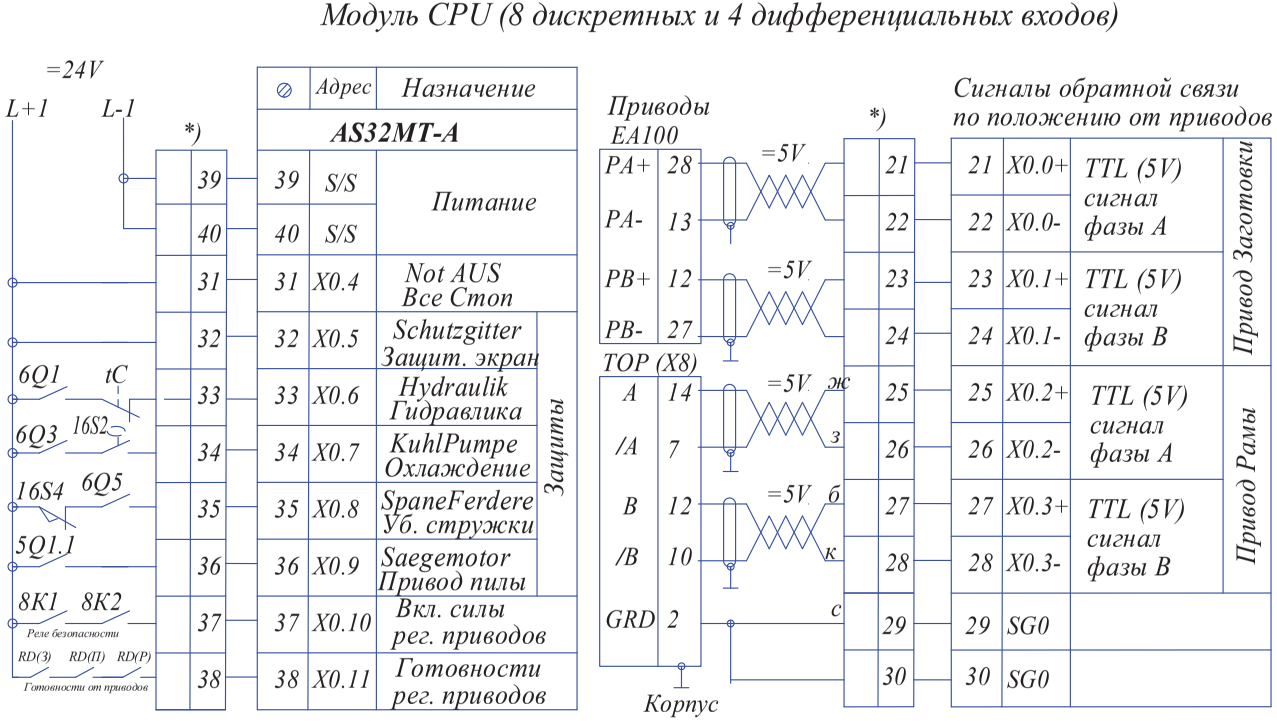


Рис. 5.34. Подключение электропривода подачи заготовки



*)- втычной переходник UB-10-IO32D

Топология ножек разъема и адресов сохраняются

Рис. 5.35. Подключение дискретных и дифференциальных входов контроллера

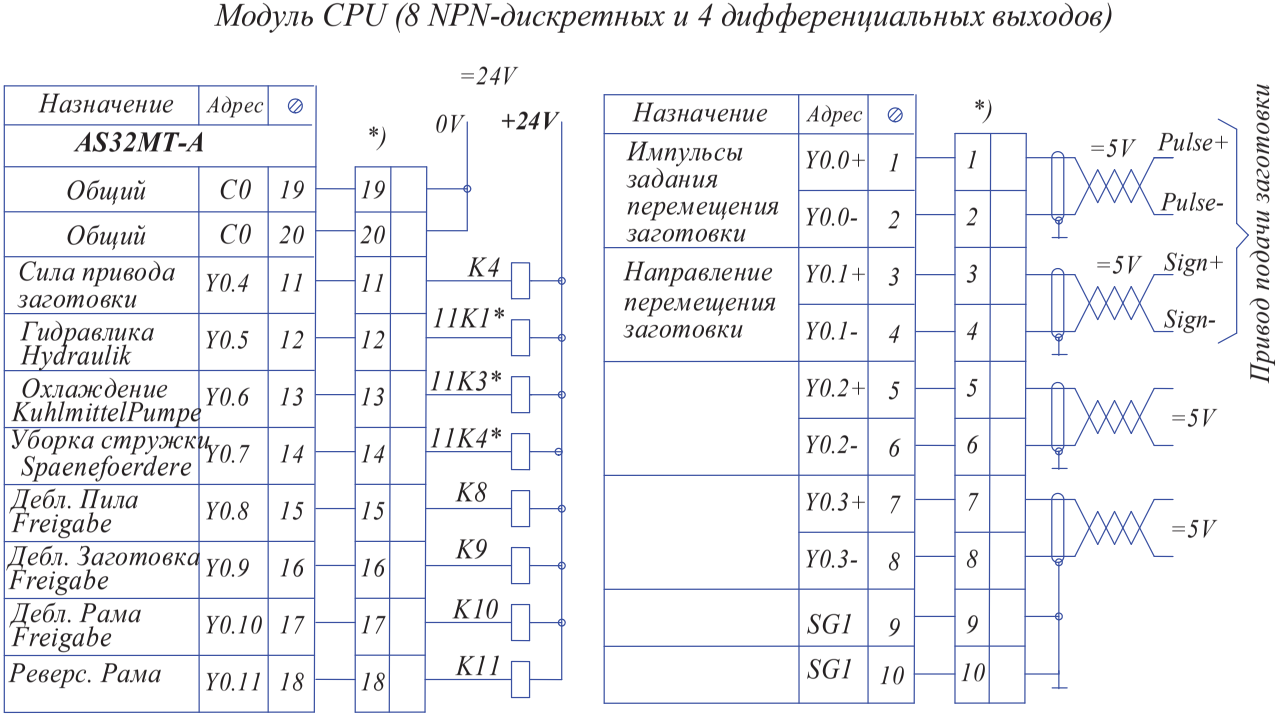


Рис. 5.36. Подключение дискретных и дифференциальных выходов контроллера

CPU	DA	Input	Output
X0.0 ... X0.15 Y0.0 ... Y0.15	D2802 (Ch1) D2804 (Ch2)	X1.0 ... X1.15	Y1.0 ... Y1.15

Перед разработкой программы необходимо заготовить чистую таблицу для заполнения используемых адресов операндов в процессе программирования после набора каждого блока программы, это позволит избежать двойных адресов и, как следствие, упростит отладку, а дальнейшем и эксплуатацию объекта автоматизации.

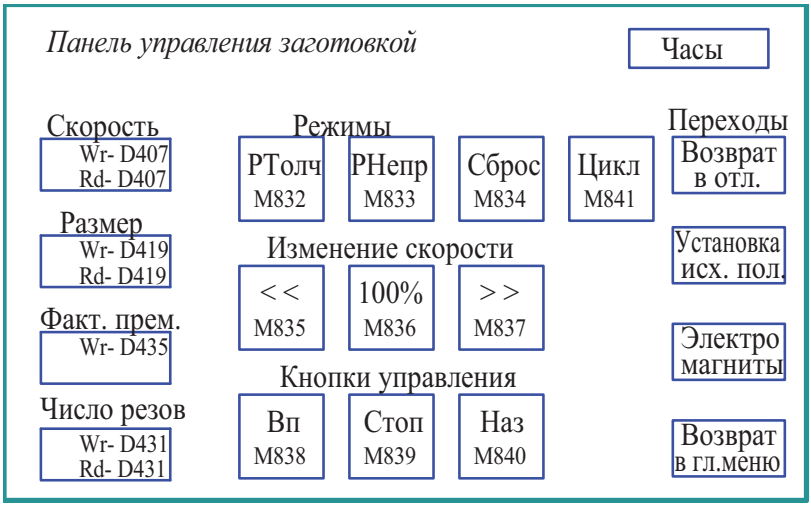


Рис. 5.37. Эскизный проект панели управления

В панели предусмотрены:

Органы Задания скорости и величины перемещения заготовки, а также числа резов в автоматическом цикле.

Индикатор фактического перемещения заготовки.

Кнопки выбора Толчкового или Непрерывного режимов управления движением заготовки в наладке.

Кнопки оперативного изменения скорости перемещения в процессе работы.

Командные кнопки перемещения заготовки: Вперед, Стоп и Назад.

Кнопки старта наладочного Цикла и Сброса электроавтоматики при зависаниях.

Часы реального времени.

Кнопки перехода на другие панели управления.

Естественно, что в процессе проектирования и наладки в эскизный проект каждой панели могут вноситься изменения, вплоть до полной переработки идеологии.

Здесь же следует указать адреса операндов, которые будут использоваться в программе электроавтоматики. Адреса присваиваются в процессе разработки общей программы и их следует немедленно заносит в таблицу адресного пространства.

Варианты реальных панелей управления заготовкой приведены ранее на рис. 1.22, рис. 3.107 и рис. 3.111.

Прежде чем приступить к изложению материала по проектированию алгоритмов электроавтоматики еще раз напомним, что общие принципы и методология проектирования не зависят от типа и фирмы-изготовителя контроллера, но для разных контроллеров, даже одной фирмы, всегда отличаются:

- система конфигурации проекта;
- система подключения и адресации входов и выходов;
- синтаксис языка программирования;
- программное обеспечение для набора программы и связи с компьютером;
- общие процедурные вопросы.

Это наглядно видно, если сравнить синтаксис языков программирования контроллеров SX2 и AS300 фирмы «Дельта», рассмотренных, соответственно, в гл. 4 и гл. 5.

Что касается алгоритмов управления механизмами, то если их разработать в общем универсальном виде, то они не будут зависеть, ни от способа подключения, ни от адресации входов и выходов и, практически, от синтаксиса языка контроллера. Это очень важно понять и применять на практике. Главное, это знать, какие вопросы и в какой последовательности нужно решать, одним словом, с чего начать!

Методология проектирования изложена в главе 2 настоящей книги.

Ниже с комментариями приводится фрагмент программы управления подачей заготовки в наладочном режиме от кнопок панели оператора.

Управление осуществляется от органов управления панели оператора, показанных на рис. 3.103 (задание параметров) и рис. 3.107 (режим наладки).

Любой локальный алгоритм (в данном случае управления заготовкой) должен начинаться с формирования блокировочных сигналов, определяющих запрет или разрешение на его работу.

Очевидно, что работа локального алгоритма должна быть *запрещена*, например, если не включен станок, не включена гидравлика, включено перемещение рамы, в ручном режиме работают другие несовместимые циклы и др.

Разрешение работы формируется при условии, что отсутствует сигнал общего запрета и присутствуют разрешающие сигналы, относящиеся только к данному механизму, например, включен привод и есть сигнал о его готовности, соблюдаются какие-либо специальные начальные условия и др.

Разрешающий сигнал формируется в отдельном блоке программы, здесь ему присвоен адрес M467.

Режимы работы заготовки

Предусмотрены следующие режимы:

– толчковое управление в наладке от кнопок;
– непрерывное управление от кнопок;
– автоматический наладочный цикл.

Алгоритм режимов управления от кнопок приведен на рис. 5.38.

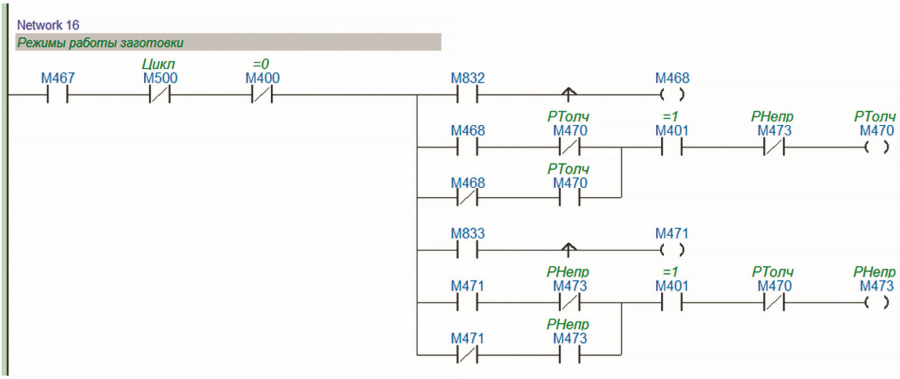


Рис. 5.38. Толчковый и непрерывный режимы работы

Работа алгоритма разрешена, если есть результирующий разрешающий сигнал M467 и отсутствует сигнал активизации автоматического цикла M500.

Сигнал M400 (всегда = 0) введен для оперативного внесения, при необходимости, дополнительной блокировки алгоритма. Для той же цели введены два сигнала M401 (всегда = 1).

Режимы РТолч (M470, кнопка M832) и РНепр (M473, кнопка M833) несовместимы между собой, поэтому принято управление от одной кнопки, первое нажатие кнопки – включение, второе – выключение. Для обеспечения такого способа управления сигналы кнопок тактируются (M468 и M471), тогда

РТолч = (КнТ(т)≠РТолч) или М470 = (М468≠М470) и

РНепр = (КнН(т)≠РНепр) или М473 = (М471≠М473).

Начальная установка скорости перемещения (рис. 5.39)

Для задания скорости перемещения заготовки в относительных единицах (здесь в процентах) принято слово D407.

При включении контроллера и наличии разрешения (М467) автоматически формируется импульс локального «Скана» М474 по которому инструкцией MOV в слово D407 записывается число 100, т. е. 100 %. Это число переписывается в индикатор задания скорости панели оператора и далее в программе электроавтоматики нормализуется в реальное задание в мм/мин для электропривода.

Кнопка М836 (100 %) позволяет установить номинальную скорость в любой момент времени.

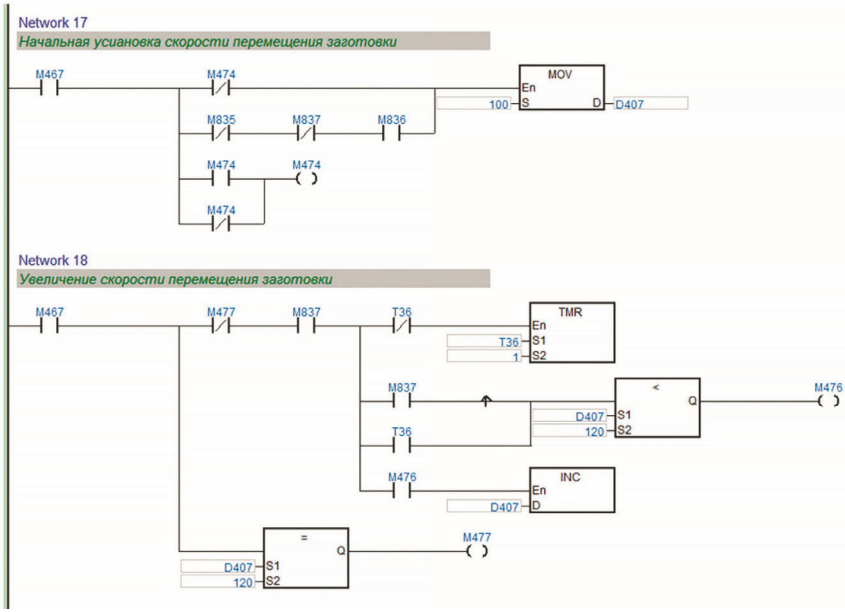


Рис. 5.39. Начальная установка скорости перемещения (блок 18) и увеличение скорости от кнопки (блок 19)

Изменение скорости перемещения от кнопок

Задание скорости перемещения заготовки может быть осуществлено следующими способами:

– автоматической начальной установкой на величину 100 %;

– предварительным заданием в режиме преднабора на панели оператора в пределах установленных ограничительными параметрами. Для работы в данном режиме достаточно коснуться пальцем по иконке;
– при помощи кнопок «Больше» и «Меньше», выведенных на панель управления. Алгоритм управления для увеличения скорости приведен на рис. 5.39, блок 19.

При нажатии кнопки M837 (> >) запускается генератор тактовых импульсов, выполненный на таймере T36. Сигналы генератора при помощи инструкции INC (Инкремент) увеличивают числовое значение слова D407, а, следовательно и скорости. При достижении верхнего ограничения скорости (здесь 120 %) работа генератора блокируется.

Аналогично организуется алгоритм уменьшения величины скорости. При этом используется кнопка M835 (< <) и инструкция DEC (Декремент). Ограничение скорости вниз осуществляется на уровне 70 %.

Перерасчет кода скорости и размера заготовки

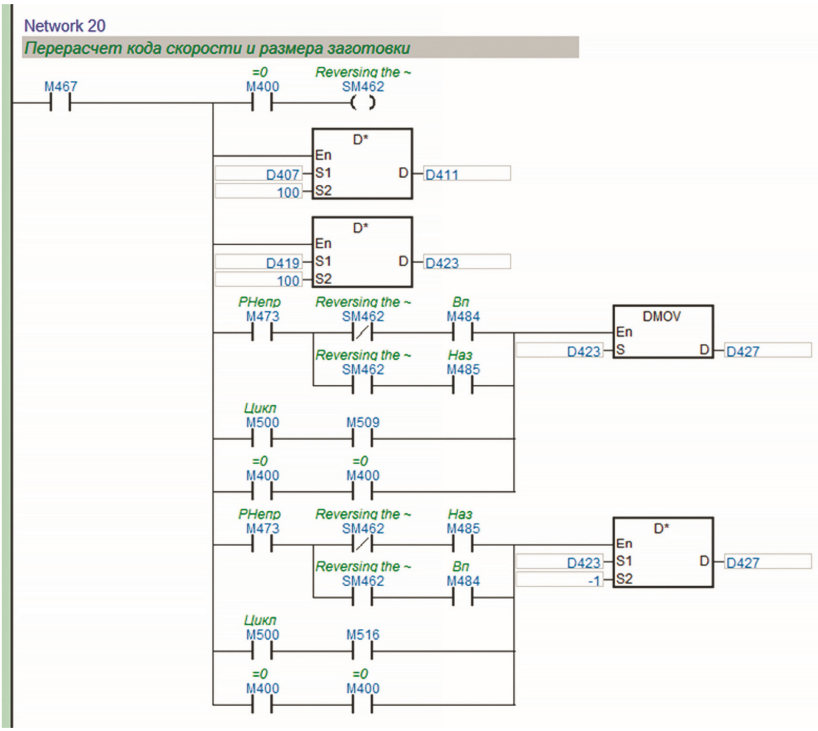


Рис. 5.40. Масштабирование кода скорости и величины перемещения

На рис. 5.40 показан алгоритм перерасчета (нормирования) кодов скорости и размера заготовки с учетом кинематики станка и дискретности TTL-импульсов.

Коды скорости (D407) и длины (D419), задаваемые на панели оператора арифметическими инструкциями умножаются на 100. Пересчитанные значения D411 и D423 используются в дальнейшем для формирования соответствующих кодов со знаком, D415 – скорость для направления перемещения вперед или назад в режиме управления от кнопок (см. рис. 5.43) и D427 – величина задания перемещения в режиме непрерывного перемещения (см. рис. 5.40 и рис. 5.43), а также в циклах.

Контроль задания исходных данных для движения

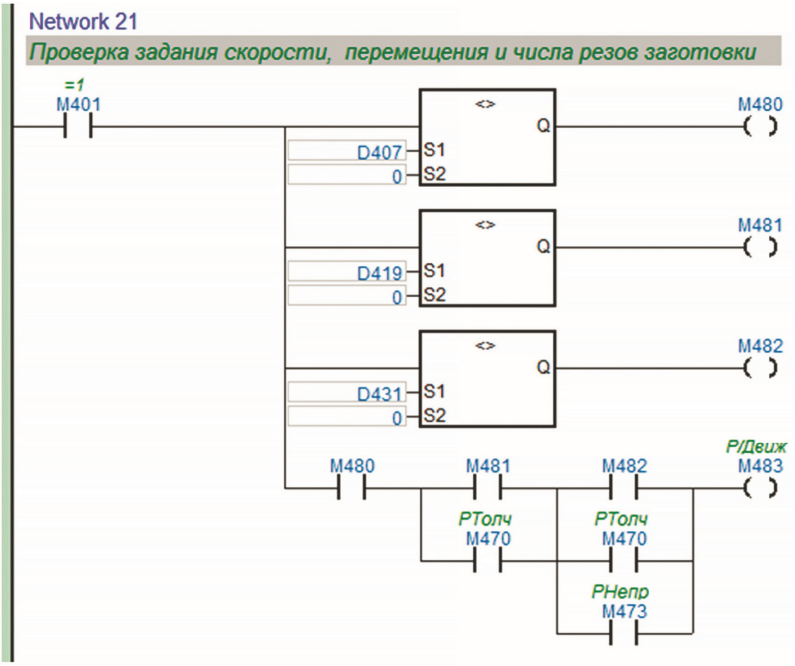


Рис. 5.41. Контроль задания исходных данных движения

Алгоритм рис. 5.41 предназначен для контроля задания исходных величин:

– скорости перемещения M480 во всех режимах;
– величины перемещения M481 при непрерывном перемещении;
– числа резов M482 в автоматических циклах.

Выходным сигналом контроля является катушка с адресом M483.

Формирование командных сигналов «Вперед» и «Назад» при управлении от кнопок (рис. 5.42, блок 22)

Выходные катушки алгоритма M484 и M485 активизируются мгновенно кнопками управления M838 и M840, соответственно, при условии наличия сигнала разрешения движения M483. При непрерывном движении катушки встают на память, и останов движения происходит при нажатии кнопки «Стоп» M839, срабатывании ограничительных конечников или при достижении заданного размера (SM461, таймер T38).

Сигналы M484 и M485 включают деблокировку привода и определяют направление движения. Их отключение в режимах задания размерного перемещения осуществляется с задержкой после формирования сигнала о завершении движения SM461 – Complete (см. работу инструкций DJOG и DDRVI).

Задание уставок (параметров) движения

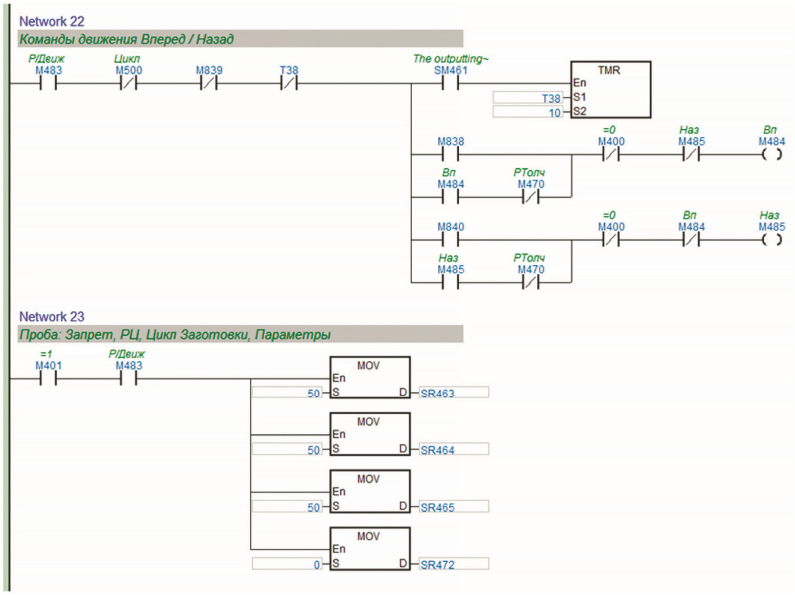


Рис. 5.42. Формирование сигналов Вперед и Назад (блок 22) и установка параметров движения (блок 23)

Блок 23 рис. 5.42 отражает установку параметров движения:

SR463 – стартовая частота управляющих импульсов;

SR464 – время ускорения;
SR465 – время замедления;
SR472 – активизация рампы (= 0).

Управление движением в толчковом и непрерывном режимах (рис. 5.43)

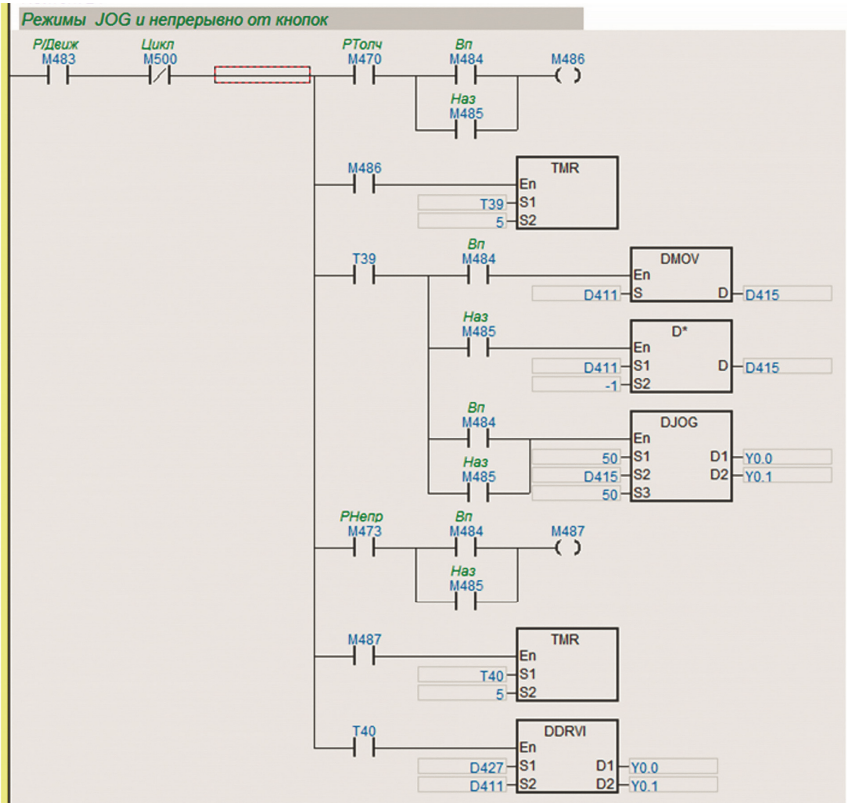


Рис. 5.43. Управление в толчковом и непрерывном режимах от кнопок

Управление в *толчковом* режиме РТолч осуществляется при помощи инструкции JOG. Назначение операндов следующее:

M470 – режим Толчок;
M484 – команда Вперед;
M485 – команда Назад;
M486 – деблокировка привода;
D411 и D415 – задание скорости без и со знаком, соответственно.

Знак скорости формируется умножением на −1

Y0.0 – выход задания частоты импульсов движения;

Y0.1 – выход направления движения.

Управление в *непрерывном* режиме РНепр осуществляется при помощи инструкции DDRVI. Назначение операндов следующее:

M473 – режим Непрерывно;

M484 – команда Вперед;

M485 – команда Назад;

M487 – деблокировка привода;

D427 – задание величины перемещения со знаком;

D411 – задание скорости;

Y0.0 – выход задания частоты импульсов движения;

Y0.1 – выход направления движения.

Обращаем внимание, что здесь применено виртуальное программирование.

Каждый блок работает независимо друг от друга, так как активизируется только своим режимом РТолч или РНепр. Виртуальные сигналы, например, деблокировки привода M486 (толчок) и M487 (непрерывно) затем объединяются в общий выходной сигнал Y по логической функции ИЛИ.

Кроме того, инструкции DJOG и DDRVI не могут работать вместе, так как у них разные сигналы активизации En.

Индикация фактического перемещения заготовки (рис. 5.44)

Индикация фактического перемещения заготовки осуществляется путем посылки транзитных TTL-импульсов с электропривода подачи заготовки через высокоскоростной счетчик HC202 в слово D439 (D435), выводящего его значение на индикатор панели оператора. Инструкция DCNT осуществляет сравнение заданной величины перемещения с фактической и при их равенстве формирует выходной бит HC202 (M521).

Задание и расчет числа резов в автоматическом режиме (рис. 5.45)

При включении контроллера и формирования сигнала разрешения движения M467 в слово D431 автоматически записывается и выводится на экран число равное единице.

При необходимости, уставка счета задается в режиме преднабора непосредственно на экране. В автоматическом режиме при помощи сигнала M518, счетчика C1 (инструкция CNT) и инструкции DEC – декремент производится подсчет сделанных резов и их динамичное отображение на экране, в данном решении – число оставшихся резов.

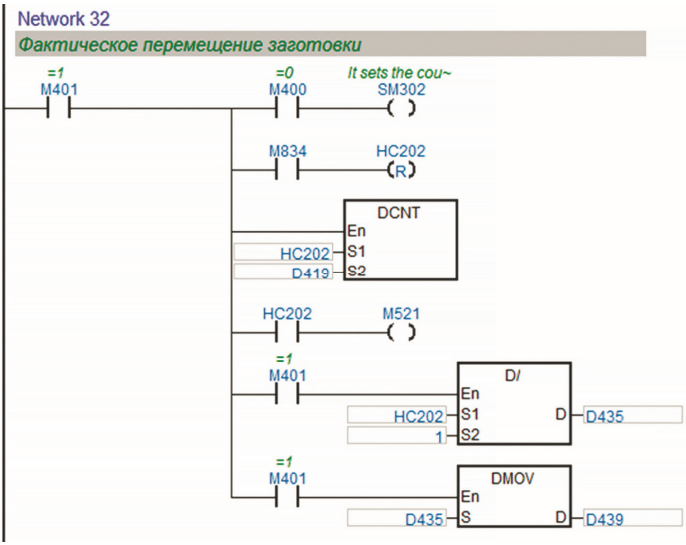


Рис. 5.44. Фактическое перемещение заготовки

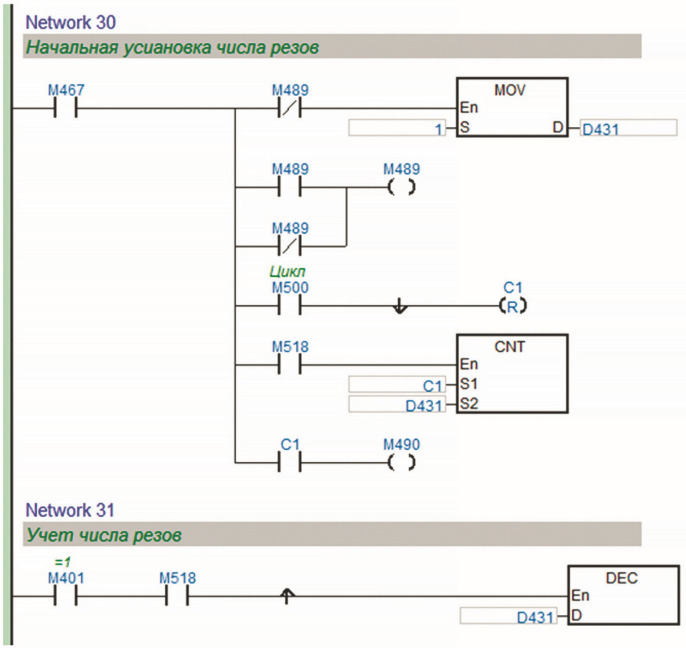


Рис. 5.45. Учет числа резов

В заключение скажем, что приведенные выше фрагменты управления приводом перемещения заготовки являются чисто наладочными, так как в них не учтено реальное состояние механизмов зажима и отжима заготовок. Для этой цели формируется отдельное окно и наладчик сам должен отслеживать ситуацию. Блокировочные сигналы несовместимых операций, как было сказано в начале раздела, учитываются при формировании результирующего сигнала разрешения управления приводом.

5.5. Процедурные вопросы

5.5.1. Инструкция по записи проекта электроавтоматики управления станком в память PLC типа AS324MT-A фирмы Дельта

Аппаратные и программные средства

Компьютер: hp, Windows 7 максимальная, SP1, 64-х разрядная ОС.

Контроллер: AS324MT-A (расширение – AS04DA-A, AS16AM10N-A, AS16AN01R-A).

Программное обеспечение: ISP Soft V3.**04.**

HWCONFIG V4.01 (Конфигуратор аппарата).

COMMGR V1.12 (Коммуникационный менеджер).

Общие сведения

1. Разработка проекта электроавтоматики управления станком для ПЛК типа AS324MT осуществляется при помощи программного обеспечения типа ISP Soft V3.**04** или V3.11, предварительно загружаемого в персональный компьютер.

2. Программа проекта имеет расширение «.isp».

3. Загрузка проекта в панель может быть выполнена несколькими способами:

– по каналу RS-485

– через mini USB-порт;

– через Ethernet.

Предварительные действия

1. Загрузить в компьютер программное обеспечение ISP Soft V3.04 или V3.11.

2. Выполнить конфигурацию проекта, получить адреса входов и выходов.

Разработать проект электроавтоматики, сделать сохранение (**File → Save**);

3. Выполнить операцию компиляции (**Compile → Compile All**), проверив проект на отсутствие ошибок.

Проверить в папке hp-PC наличие портов связи (мой компьютер → свойства системы → диспетчер устройств).

Загрузка по каналу RS-485

1. Подготовить кабель связи:

купить на фирме Дельта кабель IFD6500W953019 (Delta Electronics Inc) с адаптером USB или купить адаптер и распаять кабель самостоятельно разъем ПЛК – Разъем RJ45).

2. На компьютер поставить драйвер IFD6500-Drievers.**exe** (поставляется на диске при покупке фирменного кабеля).

3. На компьютере открыть проект электроавтоматики, выполнить компиляцию.

4. Включить контроллер.

5. Выполнить аппаратное соединение по каналу USB.

6. В меню диспетчера устройств ПК наблюдать появление папки Порты (COM + LPT) с файлом **Silicon Labs CP210X USB to UART Brige (COM6)**. ПК увидел контроллер.

7. На ПК присвоить имя драйвера связи:

Tools / Communication Setting
Driver **Driver 1** (или DRV_RS)
Station Address 1
IP Address -
нажать **OK**!

Открыть COMMGR (менеджер связи), активизировав иконку (**open**/close).

В открывшемся окне **Drive Properties** установить параметры протокола передачи:

Driver Name **Driver 1** (или DRV_RS)
Type **RS232/422/485**
Virtual COM Name **6**
COM Port **COM6**
Data Length 8 **ASCII**
Parity e RTO
Stop Bit 2 Auto-detect
Band Rate 9600 Default
Connect Retries 3
Connection Time Out 30
нажать **OK**!

В открывшемся окне **COMMGR** наблюдать параметры протокола:

Name Description ate

Driver 1 RS232/422/485,COM6,ASCII,Protocol-9600,8,e,2.RetryOK (Start)

Закрыть окно, щелкнув мышкой вне его пределов.

Внизу экрана наблюдать Offline Driver1, [RS232: COM6] AS324MT.

Активизировать **RUN** и режим **Online**. Ранее загруженная PLC-программа электроавтоматики активизируется в режиме связи.

Выключить режим **Online**, исправить (или написать новую) программу, сохранить и откомпилировать.

Активизировать процесс загрузки программы из ПК в ПЛК – **Download to PLC**, наблюдать появление окна загрузки **PC => PLC Download** с параметрами передачи, отмеченными «Галочкой».

Щелкнуть на клавишу **Transfer**. Если ПЛК не переведен в состояние **Stop**, то появится предупреждение о невозможности выполнения операции (однако все работает и режиме RUN).

Перевести ПЛК в режим **Stop** и вновь нажать **Transfer again**, наблюдать процесс загрузки.

Активизировать режим **Online.** Программа заработает на экране.

Активизировать режим **RUN**. Программа заработает в контроллере. Кабель связи можно отключить.

Загрузка через USB-порт

Подготовить стандартный покупной кабель связи **USB → mini USB**.

Записать на компьютер папку **DVP-SE-USB-driver EN** с программой драйвера **delta-iabu.inf,** например в C:\ Chernov\ ПилаМеба\Драйверы.

На компьютере открыть проект электроавтоматики, выполнить компиляцию.

Включить контроллер.

Сделать аппаратное соединение по каналу **USB**.

В меню диспетчера устройств ПК наблюдать появление папки **Порты (COM + LPT)** с файлом **Delta PLC (COM5)**. ПК увидел контроллер.

Установить драйвер на компьютер, для чего выполнить следующие операции:

Щелкнуть мышкой на файл Delta PLC (COM5) в диспетчере устройств.

В появившемся окне «свойства: Delta PLC (COM5)» выполнить или проверить:

Общее → Устройство работает нормально.

Параметры порта: 9600, 8, нет, 1, нет.

Драйвер:

а) обновить;

б) выполнить поиск драйверов в ручную:
строку C:\User\hp\Downloads\CHS41SER в режиме **Обзор (Browse)** заменить на адрес папки, где находиться файл драйвера, например, диск C:\ Chernov\ ПилаМеба\Драйверы;

в) нажать Далее (Next). Наблюдать сообщение о завершении установки;

г) нажать Закрыть (Close).

Драйвер поставлен.

PS. *Процедура установки драйверов описана в следующей документации:*

ISPSoft User Manual (Приложение А.1. Установка USB драйверов для контроллеров AS);

AS Series Operation Manual (Приложение А. Установка USB драйверов).

На ПК присвоить имя драйвера связи:

Tools / Communication Setting
Driver **Driver 2** (или DRV_USB)
Station Address 1
IP Address -
нажать **OK**!

PS. 1. Если используется только один канал связи, то нужно указать Driver1, если два канала, то Driver2.

Открыть **COMMGR** (менеджер связи), активизировав иконку (**open** /close).

В открывшемся окне Drive Properties установить параметры протокола передачи:

Driver Name **Driver 2** (или DRV_RS)
Type **USB (Virtual COM)**
COM Port **COM5**
Connect Retries 3
Connection Time Out 30
нажать **OK**!

В открывшемся окне **COMMGR** наблюдать параметры протокола:

Name Description State
Driver 2 USB,COM5,Retry=3,Time Out=3000 OK(Start)

Закрыть окно, щелкнув мышкой вне его пределов.

Внизу экрана наблюдать Offline Driver2, [USB: COM5] AS324MT.

Активизировать **RUN** → Да.

Активизировать **Online**. Связь установлена. PLC-программа на экране работает.

Исправить (написать новую) программу.

Выключить режим **Online**, перевести ПК в режим Стоп.

Активизировать передачу проекта в ПК: Download to PLC → Transfer again.

Активизировать режим **Online.** Программа заработает на экране.

Активизировать режим **RUN**. Программа заработает в контроллере. Кабель связи можно отключить.

Загрузка через Ethernet

Подготовить стандартный покупной **Ethernet** кабель связи (Распайка прямая).

1. Включить компьютер и программируемый контроллер.

В компьютере посмотреть состав сетевых адаптеров (плат).

Путь поиска: Мой компьютер / Свойства системы / Диспетчер устройств.

Открыть папку «Сетевые адаптеры»:

Сетевые адаптеры:

Realtek PCIe FE Family Controller

Адаптер мини-порта …

…….

Плата адаптера 1*1 11 b/g/n Wieless LAN PCI Exper Half Mini Card

……

Устройство Bluetooth

Задать сетевой адрес компьютера, для чего:

открыть окно «Просмотр основных сведений о системе и настроек подключений»;

соединить ПК и ПЛК Ethernet-кабелем;

наблюдать процесс Идентификации и ее результат «Неопознанная сеть»;

в окне «Просмотр… » щелкнуть по надписи «Подключение по локальной сети»;

наблюдать появление окна «Состояние – подключение по локальной сети (Ethernet)»;

в открывшемся окне щелкнуть «Свойства»;

наблюдать появление окна «Подключение по локальной сети (Ethernet) – Свойства»; В строке «Подключение через:» на данный момент может стоять имя какого-либо сетевого адаптера;

В перечне «Компоненты, используемые при подключении» поставить галочку и отметить:

Протокол Интернет в версии 4 (TCP/IPv4);

щелкнуть «Свойства»;

наблюдать появление окна «Свойства: Протокол интернета версии 4 (TCP/IPv4);

в открывшемся окне активизировать (*) и установить IP-адреса:

Получить IP-адрес автоматически

* Использовать следующие IP-адреса

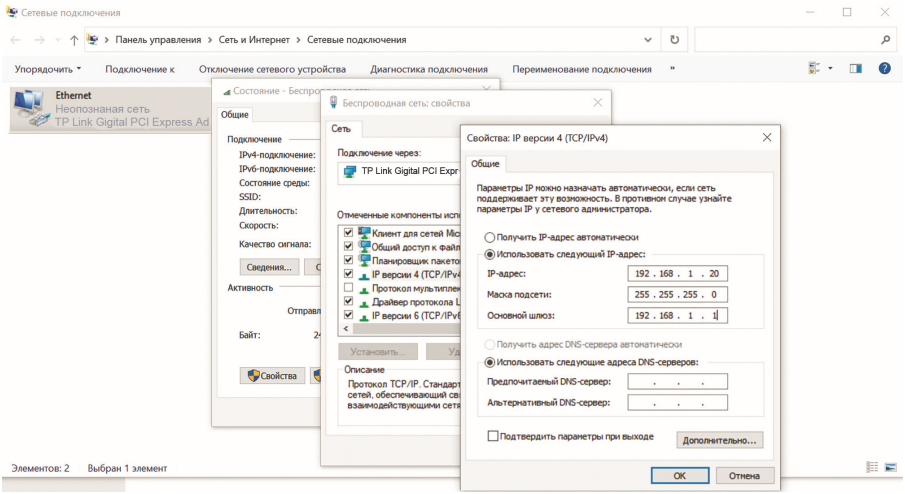
IP-адрес: 192.168.1.20

Маска: 255.255.255.0

Шлюз: 192.168.1.1

после установки адресов щелкнуть «OK!);

закрыть (свернуть) все окна.



Открыть на ПК ISP Soft V3.04 и рабочую PLC-программу.

В окне Tools / Communication Setting установить номер драйвера и IP-адрес контроллера

Driver Driver3 (или Drv_Eth)

Station Address 0

IP Address 192.168.1.5

и нажать **ОК!**

PS.

1. Если используется только один канал связи, то нужно указать Driver1, если два канала, то Driver2, если три канала, то Driver3.

2. В IP-адресе ПЛК от IP-адреса ПК должна отличаться только последняя цифра.

Открыть COMMGR → Open.

Наблюдать появление окна Drive Propetis.

В открывшемся окне указать:

Drive name Drive3

Typ Eternet

Ethernet Card Realtek PCIe FE Family Controller

и нажать **OK**!

Наблюдать

Description Realtek PCIe FE Family Controller

192.168.1.20

и нажать **OK!**В открывшемся окне **COMMGR** наблюдать параметры протокола:

Name Description State

Driver 3 Ethernet, Realtek PCIe FE Family Controller, Local IP Adrress OK(Start)

Закрыть окно, щелкнув мышкой вне его пределов.

Внизу экрана наблюдать Offline Driver3, [Ethernet] AS324MT

Активизировать **RUN** → Да.

Активизировать **Online**. Связь установлена. PLC-программа на экране работает.

Исправить (написать новую) программу.

Выключить режим **Online**, перевести ПК в режим Стоп.

Активизировать передачу проекта в ПК: Download to PLC → Transfer.

Активизировать режим **Online.** Программа заработает на экране.

Активизировать режим **RUN**. Программа заработает в контроллере. Кабель связи можно отключить.

PS. Если при многоканальной связи возникнут проблемы, то следует попереключать каналы в окне «Communication Setting».

5.5.2. Организация связи между контроллером и панелью оператора

После того как автономно разработаны, загружены и отлажены программы электроавтоматики программируемого контроллера и панели оператора, следует установить связь между контроллером и панелью. Ниже приводится компактная инструкция, основанная на подробных инструкциях связи контроллера и панели с компьютером.

Последовательность работы следующая:

1. Выполнить соединения согласно рис. 5.46

Соединение в кабеле Ethernet прямое.

2. На компьютере, контроллере и панели оператора установить параметры (проверить заданные ранее) в соответствии с приведенной ниже табл. 5.8.

Обращаем внимание, что реально, в зависимости от программных продуктов, установленных на конкретном компьютере, могут быть незначительные отличия в действиях при установке связи.

Таблица – это ваш путеводитель, задающий последовательность действий.

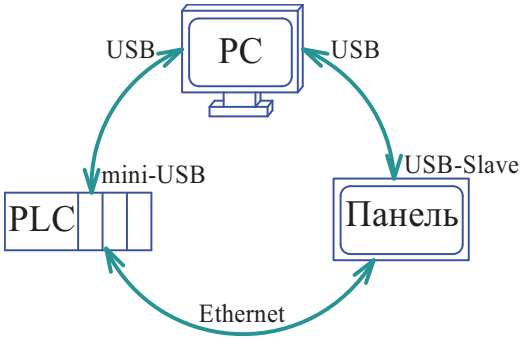


Рис. 5.46. Организация связи между PLC, ПО и PC

Таблица 5.8

Процедурные действия разработчика при установке связи

Контроллер	Компьютер	Панель
Tool / Communication Driver 2 St. Address 0	Option / Environment *USB	MISC → CDC
COMGMR → Open Driver 2 Ethernet Realtek 192.168.1.20	Configuration / Main Meba 1 DOP-110WS 65536 Main / Others CDC	Сеть: LAN1 Static 192.168.1.7 255.255.255.0 0.0.0.0 00:18:23: и т. д.
	Configuration Setting Device Ethernet ← COM2 00 – EtherLinc Delta As Series PLC TCP HMI St. 0 Cont IP: 192.168.1.5 1,12345678,0,1000,2	Сетевое окружение VNC: OFF 12345678 5900
	LocalHost Overwrite IP HMI IP Addr 192.168.1.7 Mask 255.255.255.0 Gate Way 192.168.1.1 DNS 0. 0.0.0	

3. После установки связи между контроллером и панелью, кабели, соединяющие контроллер и панель с компьютером, могут быть отключены. Контроллер и панель могут работать автономно. Связь сохраняется при выключении питания.

Процедуры организации связи значительно отличаются для разных типов контроллеров и при первом освоении являются достаточно трудоемкой работой, занимающей много времени. Как правило, в технической документации организация связи описывается непонятно и недостаточно. Разработчики считают, что это всем известно, но это не так. Поэтому совет: не тратьте время на самостоятельное освоение, не стесняйтесь спрашивать сервисных инженеров.

И помните, что основная ваша работа заключается в разработке алгоритмов управления, а не в установке связи. Разработка сложных алгоритмов управления, например, универсальными фрезерными станками приведена в [64].

Адаптировать алгоритм на любой язык электроавтоматики проблемы не составит. Исключение составляет решение редко встречающихся специальных задач, где специфика языка играет значение.

ГЛАВА 6. ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР CP1E-E400DR-A

Программируемые контроллеры типа CP1E являются самыми простыми, компактными, дешевыми из линейки контроллеров японской фирмы «Омрон» (CP1E → CP1L → CP1H → серия CJ). В то же время он обладает большими функциональными возможностями, достаточными для автоматизации оборудования средней сложности, в том числе и универсальных металлорежущих станков.

6.1. Основные технические характеристики

Основные технические характеристики:

- общее питание (110–240) В переменного тока;
- процессорный модуль на 20, 30 и 40 входов/выходов;
- возможность подключения модулей расширения;
- наличие аналоговых модулей;
- связь с компьютером стандартным кабелем USB типа B;
- совместимое с другими контроллерами программное обеспечение.

Общий вид контроллеров серии CP1 приведен на рис. 6.1.

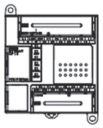
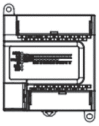
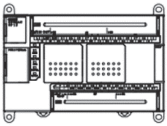
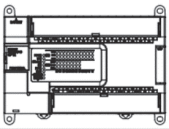
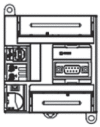
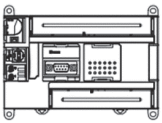
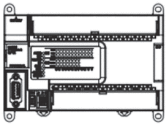
	Basic Models		CP1E Application Models		
	E□□(S)-type CPU Units		N□□(S□)-type CPU Units		NA□□-type CPU Units
	CPU with 10, 14 or 20 I/O Points	CPU Unit with 30, 40 or 60 I/O Points	CPU with 14 or 20 I/O Points	CPU Unit with 30, 40 or 60 I/O Points	CPU Unit with 20 I/O Points
Appearance	E□□-type  E□□S-type 	E□□-type  E□□S-type 	N□□-type 	N/NA□□-type  N□□S(1)-type 	

Рис. 6.1. Общий вид контроллеров серии CP1

6.2. Аппаратное подключение

Рис. 6.2 – аппаратное винтовое подключение процессорного модуля CP1E-E400DR-A на 24 дискретных входа и 16 релейных выходов.

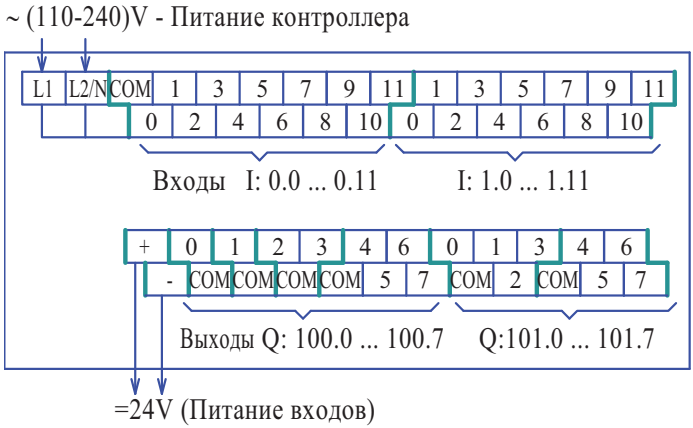


Рис. 6.2. Клеммы подключения процессорного модуля

Рис. 6.3 – принципиальная схема подключения процессорного модуля.

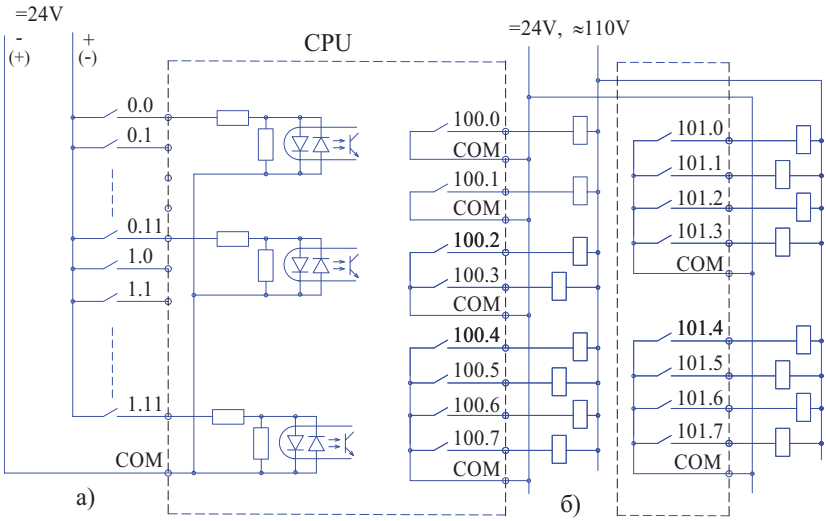


Рис. 6.3. Принципиальная схема подключения процессорного модуля

Рис. 6.4 – аппаратное винтовое подключение модуля расширения CP1W-40EDR на 24 дискретных входа и 16 релейных выходов.

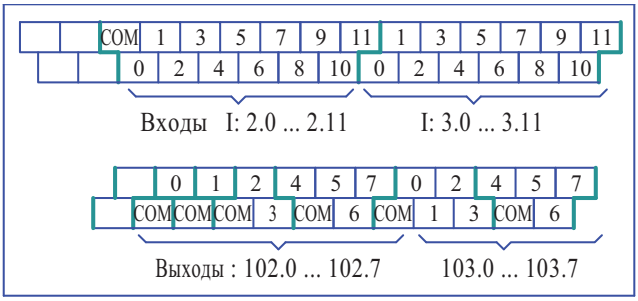


Рис. 6.4. Клеммы подключения модуля расширения

Назначение клемм подключения следующее:

L1, L2 – силовое питание процессорного модуля;

COM – общие групповые выводы входов и выходов;

0, 1,…, 11 – клеммы подключение входов и выходов;

(+/-) – внутренний источник питания = 24 В.

Принципиальная схема подключения расширительного модуля построена аналогично.

6.3. Адресация входов и выходов

Адресация входов и выходов *процессорного модуля*:

I: 0.0…0.11 – первая группа клемм;

I: 1.0 … 1.11 – вторая группа клемм;

Q: 100.0 … 100.7 – первая группа клемм;

Q: 101.0 … 101.7 – вторая группа клемм.

Примечания.

Латинские буквы I: и Q: при наборе числовых адресов устанавливаются автоматически.

Обе группы входов имеют общий провод COM. Допускается подключение напряжения = 24 В любой полярности. Для питания входов можно использовать внутренний источник питания (клеммы =/−).

Общие провода выходов COM сгруппированы по группам:

Q: 100.0 – отдельно;

Q: 100.1 – отдельно;

Q: 100.2 и 100.3 – общий;

Q: 100.4 … 100.7 – общий;
Q: 101.0 … 101.3 – общий;
Q: 101.4 … 101.7 – общий.

Адресация входов и выходов *модуля расширения*:
I: 2.0…2.11 – первая группа клемм;
I: 3.0…3.11 – вторая группа клемм;
100.0…100.7 – первая группа клемм;
101.0…101.7 – вторая группа клемм.

Примечания.

Латинская буква I: входов при наборе числовых адресов устанавливается автоматически.

Буква Q: для выходов расширителя не устанавливается.

Обе группы входов имеют общий провод COM. Допускается подключение напряжения =24 В любой полярности. Для питания входов также можно использовать внутренний источник питания (клеммы =/–).

Общие провода выходов COM сгруппированы по группам:
102.0 – отдельно;
102.1 – отдельно;
102.2 и 102.3 – общий;
102.4…102.7 – общий;
103.0…103.3 – общий;
103.4…103.7 – общий.

6.4. Конфигурация проекта

Последовательность работы при первоначальной конфигурации проекта следующая:

1. Установить на компьютер программное обеспечение Programmer V9.3.
2. Сделать аппаратное подключение и соединить компьютер и контроллер стандартным кабелем USB типа B (рис. 6.5).
3. Включить контроллер и установить драйвер связи.
4. Проверить, что компьютер увидел контроллер:
Мой компьютер / Оборудование / Диспетчер устройств / Контроллеры → «Omron SYSMAC PLC Device».
5. Открыть Programmer V9.3.
6. В падающем меню File / New открыть окно конфигурации Change PLC и установить параметры контроллера:

Device Name: Proba1
Device Type: CP1E
Setting: E40
Network Type: USB

7. Нажать клавишу ОК, наблюдать открытие окна набора программы.
8. Набрать нужную программу и выполнить следующие действия:
   Program / Compile – компиляция программы;
   Щелчок по иконке «Треугольник» – Установка связи, ответ ДА;
   PLC / Transfer to PLC, ОК, Да, ДА – запуск процедуры загрузки программы в контроллер, ожидание сообщения Download Succesful, ОК, Да.
9. Программа загружена, связь установлена, можно производить отладку. Видна исключительная простота процедуры установки связи по сравнению с другими типами контроллеров, в том числе и фирмы Омрон.

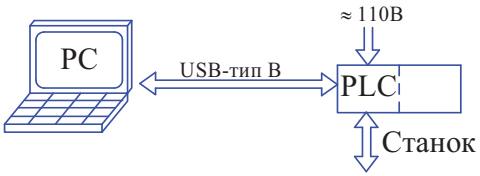


Рис. 6.5. Схема соединения компьютера и контроллера

6.5. Синтаксис языка электроавтоматики

Ниже рассматриваются основные команды релейно-контактного языка, достаточные для программирования электроавтоматики управления универсальными станками.

Ниже приведено *распределение адресного пространства* и доступные операнды:

Дискретные входы I: 0…99, т. е. 0.0, 0.1, 0.2 и т. д.
Дискретные выходы Q: 100…199, т. е. 100.0, 100.1, 100.2 и т. д.
ОЗУ пользователя 200…299, т. е. 200.0, 200.1, 200.2 и т. д.
Таймеры T0…T255.
Счетчики C0…C255.
Слова W0…W 99, т. е. W0.0, W0.1, W0.2 и т. д.
Регистры D0…D2047.
Ячейки долговременной памяти H0…H99
Вспомогательные операнды системы A0…A753

Базовые логические команды:

LD, LD NOT – считывание 1-го операнда логической цепи;
AND, AND NOT – логическое умножение;
OR, OR NOT – логическое сложение;
OUT – посылка результата на выход;
NOT – инверсия;
AND LD – безадресная работа со стеком (И);
OR LD – безадресная работа со стеком (ИЛИ).

Набор базовых команд в Ladder-диаграмме осуществляется путем перетаскивания из меню соответствующей иконки на место установки курсора, набора и подтверждения адреса оператора.

Разрешается набор разветвленных логических цепей (рис. 6.6), при этом базовое математическое обеспечение автоматически использует стековые команды и разработчику электроавтоматики об этом задумываться не нужно.

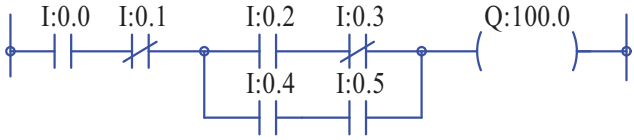


Рис. 6.6. Разветвленная релейная цепь с использованием стека

Программа в кодах будет выглядеть так:

LD I:0.0 – считывание;
AND NOT I:0.1 – логическое умножение;
LD I:0.2 – проталкивание в стек и считывание;
AND NOT I:0.3 – логическое умножение;
LD I:0.4 – проталкивание в стек и считывание;
AND I:0.5 – логическое умножение;
OR LD – логическое сложение со стеком;
AND LD – логическое умножение на стек;
OUT Q:100.0 – посылка результата на выход;

Инструкция инверсии NOT (рис. 6.7)

Выходная ячейка 100.0 будет активизирована при условии нулевого значения логического уравнения, т. е. при (I:0.0 + I:0.1) * I:0.2 = 0 выход будет равен единице, т. е. 100.0 = 1, и наоборот.

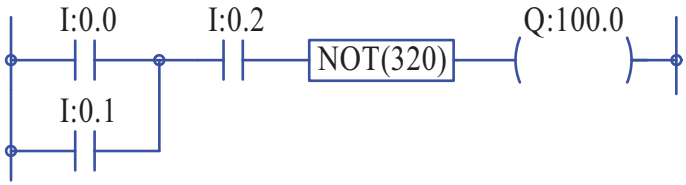


Рис. 6.7. Базовая инструкция инверсии

Формирователи тактов (рис. 6.8)

Предусмотрено несколько вариантов формирования сигналов длительностью в один вычислительный цикл по переднему или заднему фронтам входного сигнала:

UP – по переднему фронту. Инструкция требует начертания выходной катушки;

DOWN – по заднему фронту. Инструкция требует начертания выходной катушки;

DIFU – по переднему фронту. Инструкция не требует начертания выходной катушки;

DIFD – по заднему фронту. Инструкция не требует начертания выходной катушки.

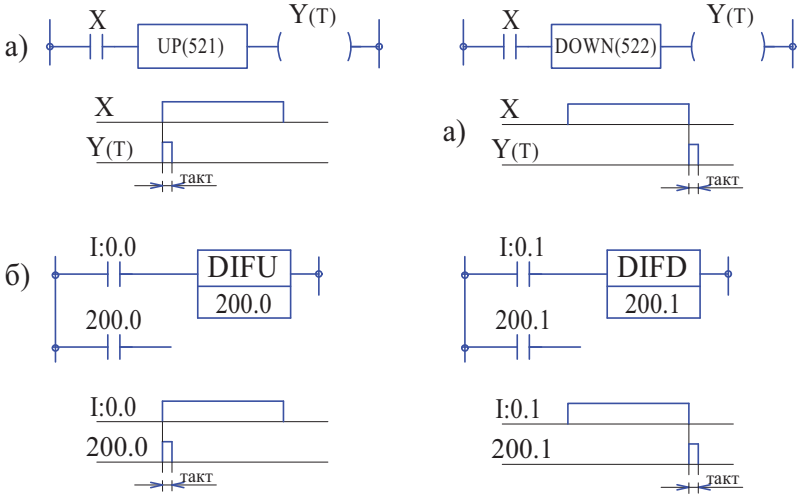


Рис. 6.8. Формирователи тактов

При желании, тактовые сигналы можно формировать самостоятельно при помощи базовых команд (см. рис. 3.29 для PLC Дельта).

Также имеется возможность тактировать операнд, записывая внутрь контакта стрелку (см. дальше рис. 6.16).

Инструкция Keep-реле (рис. 6.9)

Данная инструкция принадлежит к классу RS-триггеров и каких-либо пояснений не требует.

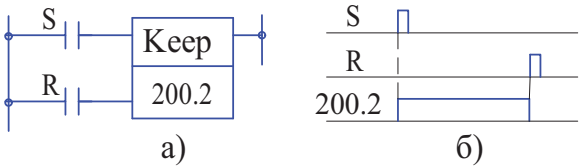


Рис. 6.9. Принципиальная схема (а) и диаграмма работы (б) инструкции Keep

Инструкция Set / Reset (рис. 6.10)

Инструкция позволяет раздельно устанавливать или сбрасывать тот или иной операнд, т. е. это тоже RS-триггер. Отличительной особенностью подобных инструкций является отсутствие реакции на дребезг входного сигнала активизации.

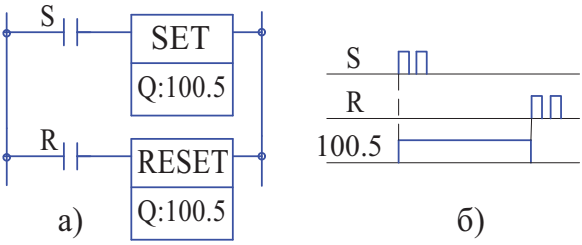


Рис. 6.10. Принципиальная схема (а)
и диаграмма работы (б) инструкций Set / Reset

Инструкция Timer (рис. 6.11)

Синтаксис языка контроллера включает несколько типов таймеров:

TIM – таймер с дискретностью 0,1 сек и заданием в BCD-коде. Диапазон уставок #0…#9999;

TIMX – таймер с дискретностью 0,1 сек и заданием в BIN-коде. Диапазон уставок &0…&65536;

TIMH – таймер с дискретностью 10 мс и заданием в BCD-коде. Диапазон уставок #0…#9999;

TIMHX – таймер с дискретностью 10мс и заданием в BIN-коде. Диапазон уставок &0…&65536;

TML – таймер с большой выдержкой времени и заданием в BCD-коде. Диапазон уставок #…#9999 9999;

TMLX – таймер с большой выдержкой времени и заданием в BIN-коде. Диапазон уставок &0…&4 294 967 294.

Общее число таймеров 256.

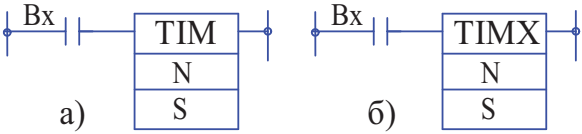


Рис. 6.11. Условное обозначение таймеров, кодируемых в BCD (а) и BIN (б) кодах

Здесь:

N – номер таймера (0–255);

S – уставка выдержки времени в дискретах.

Уставки в BCD-коде задаются со знаком #, а в BIN-коде со знаком &.

Последовательность набора инструкции следующая:

– установить курсор в нужное место;
– перетащить мышкой знак функциональной инструкции на место курсора;
– щелкнуть мышкой и в появившемся поле набрать с пробелом TIM 0 &10 (имя инструкции, номер инструкции и величина уставки);
– при необходимости, набрать комментарий;
– щелкнуть OK, на экране появится изображение инструкции таймера (см. рис. 6.12). Оно позволяет в реальном времени отслеживать текущее значение выдержки времени.

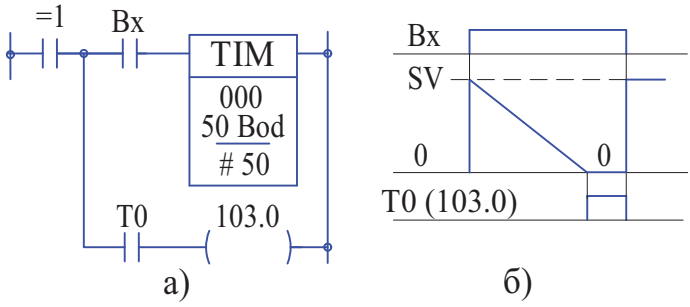


Рис. 6.12. Условное обозначение (а) и диаграмма работы (б) таймера TIM

Здесь:
000 – номер таймера;
50 Bod – текущее значение;
#50 – величина уставки времени SV (Set Value).

Обращаем внимание, что текущее состояние таймера изменяется в сторону уменьшения, от величины уставки до нуля.

Если за время входного сигнала активизации таймер не успеет отсчитать заданную выдержку времени, то текущее значение устанавливается в SV и отсчет начинается сначала. При достижении нулевого значения активизируется выходной бит таймера и остается включенным до снятия сигнала активизации.

Доступ к текущему состоянию таймера, при необходимости, осуществляется косвенным путем, используя инструкции пересылки и сравнения. Обращаем внимание, что текущее состояние и выходной бит в Ladder-диаграмме обозначаются одинаково.

Примечание. Сигнал = 1 позволяет набирать разветвленную цепь в одной строке программы.

Данной классической инструкции таймера вполне достаточно, чтобы спроектировать любой временной алгоритм (см. раздел 2.11 главы 2).

Инструкция Counter (рис. 6.13)
Синтаксис языка контроллера включает следующие типы счетчиков.
Нереверсивные:
CNT – вычитающий счетчик с заданием в BCD-коде. Диапазон уставок #0…#9999;
CNTX – вычитающий счетчик с заданием в BIN-коде. Диапазон уставок &0…&65535.
Реверсивные:
CNTR – реверсивный счетчик с заданием в BCD-коде. Диапазон уставок #0…#9999;
CNTRX – реверсивный счетчик с заданием в BIN-коде. Диапазон уставок &0…&65535.
Общее число счетчиков 256.
На рис. 6.13:
N – номер счетчика (0 -255). Нумерация счетчиков разного типа – сквозная;
S – уставка счета в BCD – коде (#) или в BIN – коде (&) в зависимости от типа счетчика.

Счетчик имеет два входа:
In – счетный вход и
Reset – вход сброса.

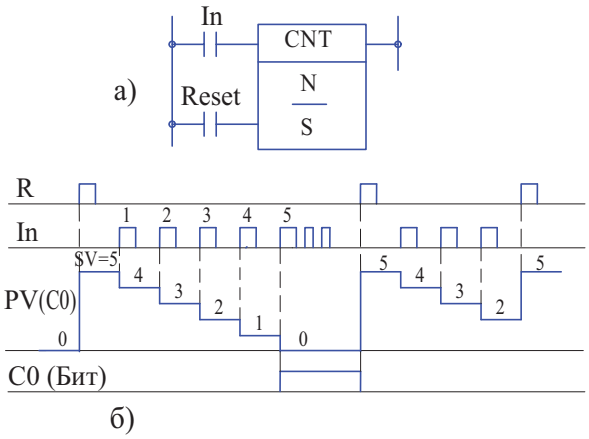


Рис. 6.13. Условное обозначение (а) и диаграмма работы (б) счетчика CNT

Доступ к текущему состоянию осуществляется через инструкции сравнения и перемещения.

Счетчик считает на вычитание, при достижении нулевого значения активизируется и остается включенным выходной бит счетчика. Текущее значение отображается на экране дисплея.

При активизации входа сброса в счетчик записывается величина уставки, а битовый выход обнуляется.

Обращаем внимание, что отображение текущего состояния и битового выхода на Ladder-диаграмме одинаково (буква С, сопровождаемая номером счетчика).

На рис. 6.14 приведен пример организации счетчика с автоматическим сбросом при обнулении уставки.

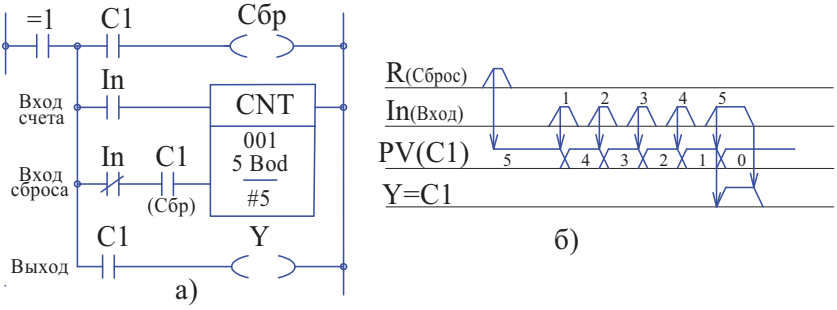


Рис. 6.14. Алгоритм (а) и диаграмма работы (б) счетчика с автоматическим сбросом

Здесь:
001 – номер счетчика;
5 Bod – текущее значение;
#5 – величина уставки счета SV (Set Value).

В формировании логического уравнения для входа автоматического сброса R = /In*C1 можно использовать как прямой (C1), так и косвенный сигнал сброса (Сбр).

Реверсивные счетчики CNTR и CNRX (рис. 6.15)
Реверсивные счетчики имеют три входа:
In+ счетный вход на сложение;
In – счетный вход на вычитание;
R – вход сброса.

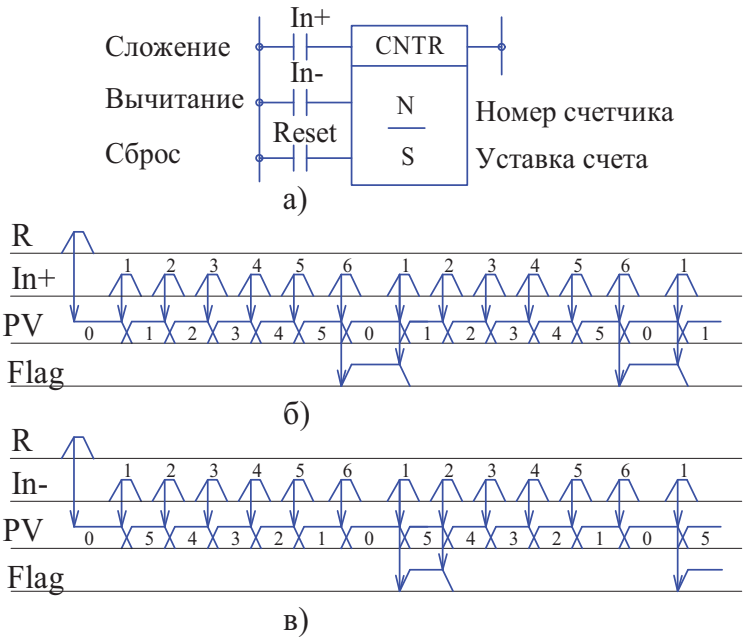


Рис. 6.15. Диаграммы работы реверсивного счетчика

На рис. 6.15, *б*, *в* приведены диаграммы работы реверсивного счетчика, снятые с действующего стенда. Они весьма своеобразны при формировании выходного битового сигнала (Flag). При счете вперед он формируется при текущем значении равном нулю, а при счете назад при значении уставки.

Это логично, но дело в том, что при начальном сбросе счетчик всегда устанавливается в значение ноль. По этой причине при старте счета с нуля и уставке, равной 5, активизация Флага при счете вперед происходит на 6-й импульс, а при счете назад на 7-й. Это обстоятельство следует учитывать при проектировании кольцевых реверсивных счетчиков. Для управления универсальными станками такая задача, как правило, не стоит.

Инструкции пересылки **MOVE**

Синтаксис языка контроллера включает следующие типы инструкций:

MOV – пересылка слов;
MOVL – пересылка двойных слов;
MVN – пересылка слов с побитовой инверсией;
XCHG – обмен словами (Data Exchange);
MOVB – пересылка отдельных битов (Move Bit);
MOVD – пересылка отдельных чисел (Move Digit).

На рис. 6.16 приведены изображения в Ladder-диаграмме инструкции MOV.

Команда выполняет следующую операцию: при активизации входного сигнала осуществляется пересылка слова источника в слово приемника, т. е.

Источник → Приемник.

Содержимое Источника при этом не изменяется.

Пересылать можно дискретные выходы, ОЗУ, слова, таймеры, счетчики, числа и др.

Естественно, что приемником не могут быть дискретные входы и числа.

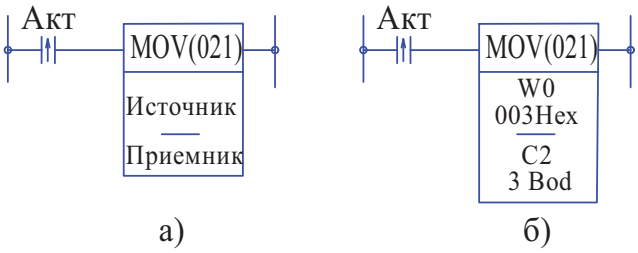


Рис. 6.16. Учебная (а) и реальная (б) пиктограммы инструкции MOV

Рис. 6.16, *б* показывает пример начальной установки счетчика C2 при помощи команды пересылки MOV.

Здесь:

W0 – слово источника;

003 Hex – содержимое слова источника;
C2 – слово приемника (счетчик №2);
3 Bod – содержимое приемника;
#5 – величина уставки счета SV (Set Value).

Для осуществления однократности операции переписи входной сигнал активизации ACT необходимо тактировать, как показано на рис. 6.16. При длительном сигнале активизации перепись будет осуществляться постоянно, пока он активен.

Примечание. Прямое тактирование сигнала Act(↑) при наборе программы осуществляется одновременным нажатием клавиш (Shift + @). Если необходимо тактирование по заднему фронту Act(↓), то следует нажать (Shift + %).

Инструкции сравнения (рис. 6.17)

Синтаксис языка контроллера включает следующие типы инструкций:

= сравнение на равенство;
< > сравнение на неравенство;
< сравнение на меньшее значение;
< = сравнение на равенство или меньшее значение;
> сравнение на большее значение;
> = сравнение на равенство или большее значение.

При программировании необходимо адресовать:

– вход активизации;
– сравниваемые величины S1 и S2;
– битовый выход Y.

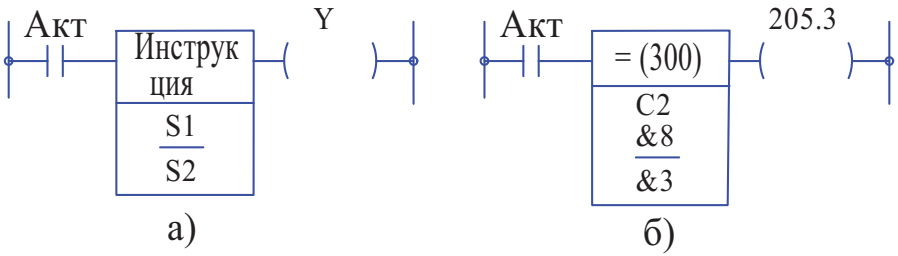


Рис. 6.17. Учебная (а) и реальная (б) пиктограммы инструкции сравнения

При активном входе выход Y устанавливается в логическую единицу, т. е. Y = 1, если выполняются заданные условия сравнения:

= S1 = S2;

< > S1 ≠ S2;

< S1 < S2;

< = S1 ≤ S2;

> S1 > S2;

> = S1 ≥ S2.

Сравниваемые величины должны иметь одинаковый формат.

Инструкции сравнения можно программировать как биты при работе с базовыми командами, т. е. логически умножать и складывать (рис. 6.18).

Выходной бит 205.4 будет активен, если значение счетчика C2 будет находиться в диапазоне больше 3 и меньше 7, т. е.

$$205.4 = Act*(C2 > 3)*(C2 < 7).$$

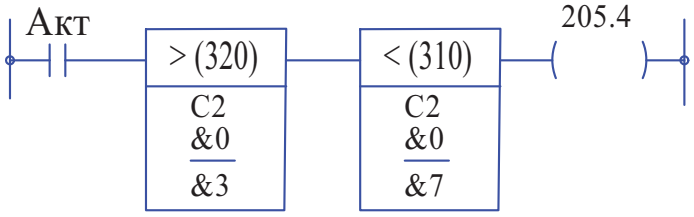


Рис. 6.18. Использование инструкций сравнения в виде логического умножения

Инструкции преобразования кодов **BCD** и **BIN** (рис. 6.19)

Синтаксис языка контроллера предусматривает преобразование двоичного кода BIN в двоично-десятичный BCD, и наоборот. Преобразовывать можно как 16-тиричные, так и двойные слова (L).

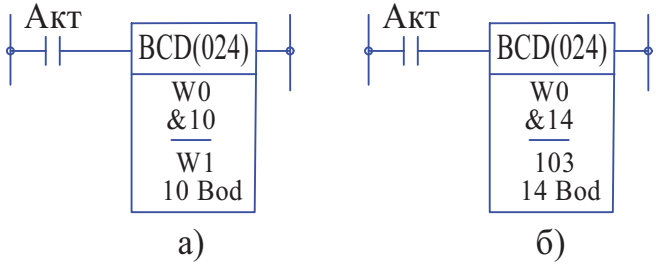


Рис. 6.19. Примеры преобразования и пересылок:
а – слово W0 (BIN) в слово W1 (BCD); *б* – слово W0 (BIN) на выход 100 (BCD)

Инструкции BCD и BCDL преобразуют двоичный код источника в двоично-десятичный и переписывают его в слово приемника, т. е. BIN → BCD. Содержимое слова источника не изменяется.

Инструкции BIN и BINL преобразуют двоично-десятичный код источника в двоичный и переписывают его в слово приемника, т. е. BCD → BIN.

Содержимое слова источника не изменяется.

Инструкция рис. 6.19, *а* преобразует двоичное значение 10 слова

W0 = 0000 1010

в двоично-десятичное и пересылает его в слово

W1 = 0001 0000.

Набор команды осуществляется следующим образом: BCD W0 W1.

Инструкция рис.6.19, *б* преобразует двоичное значение 14 слова

W0=0000 1110

в двоично-десятичное и пересылает его на выход

103=0001 0100.

Набор команды осуществляется следующим образом: BCD W0 103.

Инструкции **Increment** *и* **Decrement** (рис. 6.20)

Команды Инкремента и Декременты могут работать в двоичном и двоично-десятичном коде, как с 16-разрядными, так и с двойными словами:

Increment: ++, ++L (BIN)
++B, ++BL (BCD)
Decrement: – -, – -L (BIN)
– -B, – -BL (BCD)

Для обеспечения однократности выполнения операции входной сигнал должен быть тактирован.

При каждом входном тактируемом сигнале содержимое рабочего слова для инструкции Инкремент увеличивается на единицу, а для инструкции Инкремент уменьшается.

На экране дисплея показывается текущее состояние рабочего слова.

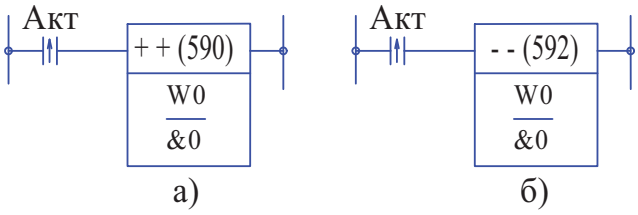


Рис. 6.20. Инструкции Инкремент (а) и Декремент (б)

Регистровые инструкции

В синтаксисе языка предусмотрены различные сдвиговые и регистровые инструкции, например:

ROL – Rotate Left;
ROR – Rotate Right;
SFT – Shift Left;
SFTR – Reversive Shift Register и др.

Инструкции работают по классическим принципам, для их изучения, при необходимости, адресуем читателя к сопроводительной документации на контроллер.

Изучив основные инструкции языка, можно переходить к разработке рабочей программы.

6.6. Примеры набора программ электроавтоматики

На рис. 6.21 и 6.22 в качестве иллюстрации приведено изображение программ на экране компьютера в программной среде Programmer V9.3.

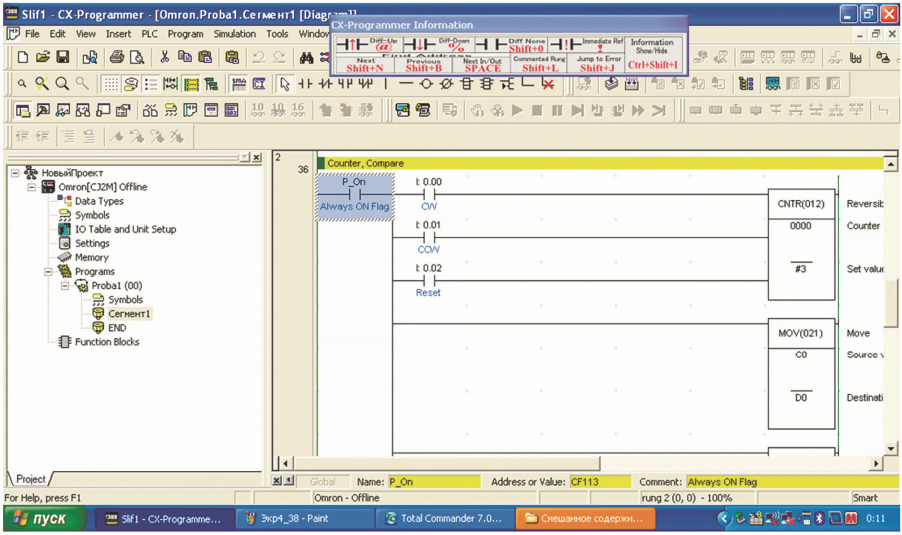


Рис. 6.21. Инструкции счетчика и пересылки ПЛК фирмы «Омрон»

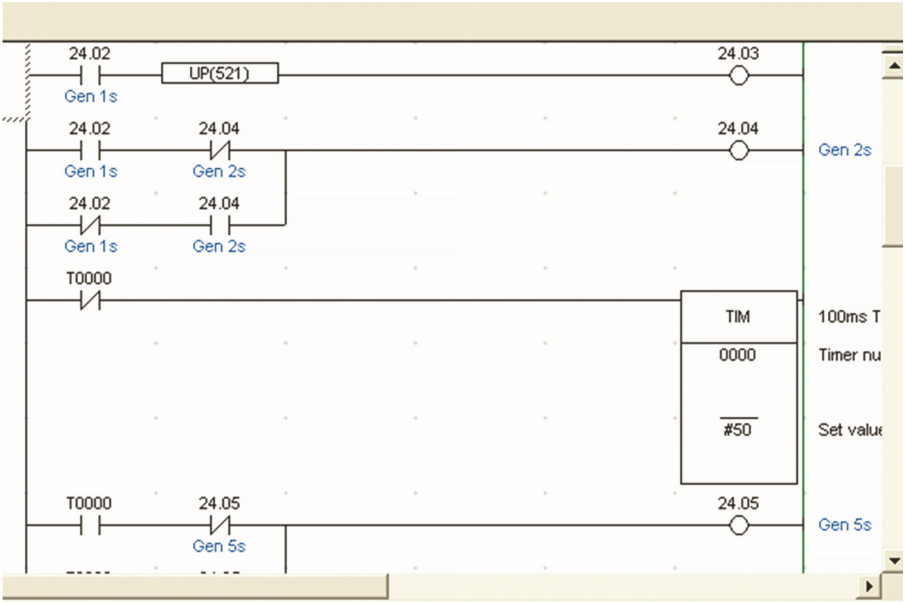


Рис. 6.22. Инструкции генераторов и таймера ПЛК фирмы «Омрон»

ГЛАВА 7. ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР ТИПА TM221

7.1. Общие сведения

Фирма Шнайдер Электрик выпускает широкую гамму программируемых контроллеров, различающихся техническими возможностями, конфигурацией, стоимостью (рис. 7.1).

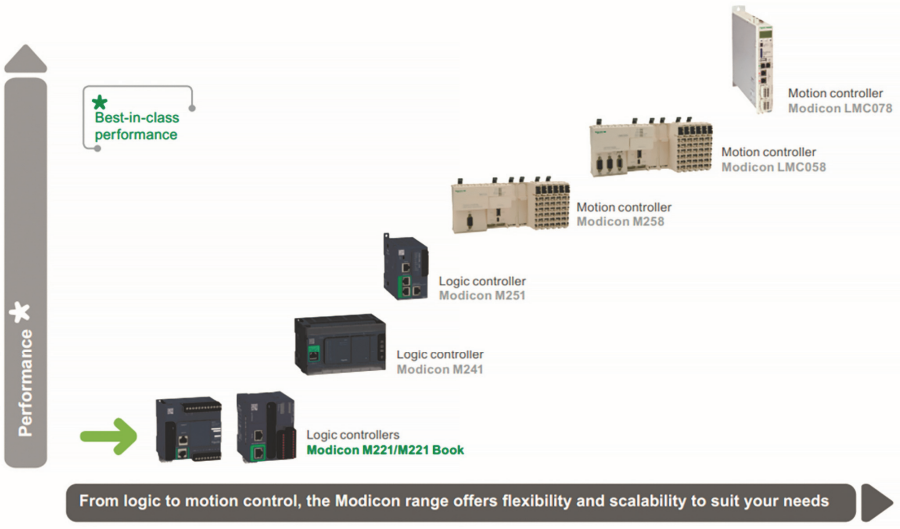


Рис. 7.1. Гамма контроллеров фирмы Шнайдер Электрик

В настоящей главе рассматривается контроллер типа TM221, выпускаемый в трех модификациях (рис. 7.2) на 16, 24 и 40 дискретных входов и выходов.

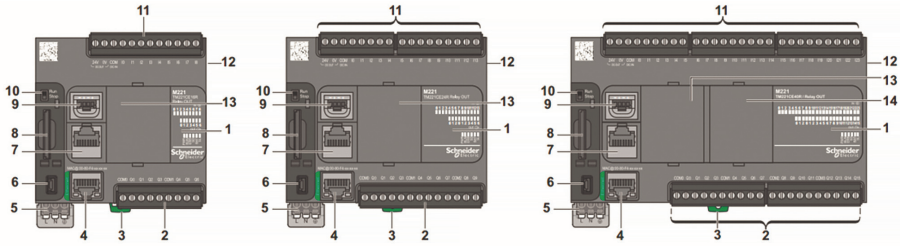


Рис. 7.2. Общий вид контроллеров TM221 на 16, 24 и 40 Вх/Вых

Основные технические характеристики контроллеров TM221 приведены в табл. 7.1.

Исполнения:

T – транзисторные выходы;

R – релейные выходы;

E – исполнение с Ethetrnet.

Таблица 7.1

Основные технические характеристики контроллера TM221

TM221C16x		**TM221C24x**		**TM221C40x**	
Исполн. R	Исполн. T	Исполн. R	Исполн. T	Исполн. R	Исполн. T
~(100-240)В	=24В	~(100-240)В	=24В	~(100-240)В	=24В
9 вх / 7 вых		14 вх / 10 вых		24 вх / 16 вых	
95 * 90 * 70		110 * 90 * 70		163 * 90 * 70	
Два аналоговых входа ± 10В					
1 порт Ethernet, исполнение E					
1 мини-USB порт					
Контроль положения в исп. CxxT и CExxT					
Подключение модулей расширения TM3 (до 264 вх / вых)					
Вариант книжного исполнения (Book)					
ПО – So Machine Basic					
Языки программирования: Ladder Diagram / Instruction List					
Совместимость: Windows 7 и Windows XP / SP3					
Сайт: www. schneider-electric.com					

7.2. Подключение процессорного блока

Исполнение 16R / Входы (рис. 7.3)

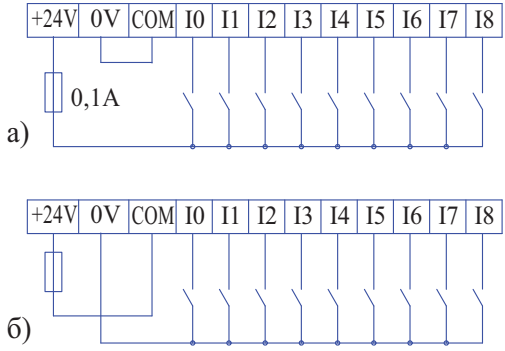


Рис. 7.3. Подключение входов исп. 16R:
а – общий плюс; *б* – общий минус

При подключении входов используется *внутренний источник питания*. Допускается любая полярность подключения общего провода входов.

Исполнение 16R / Выходы (рис. 7.4)

Выходы разбиты на две группы с общими выводами COM0 (Q0–Q3) и COM1 (Q4–Q6).

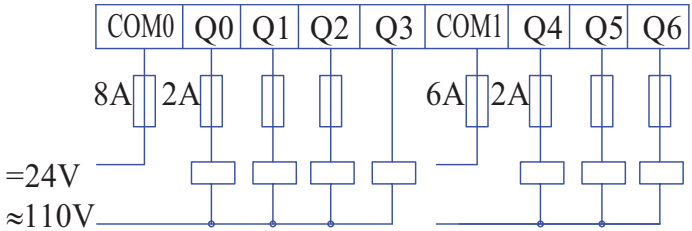


Рис. 7.4. Подключение выходов исп. 16R

Исполнение 16T: Входы / Выходы

Схема подключения приведена на рис. 7.5. Полярность подключения входов любая, выходов – строго фиксирована.

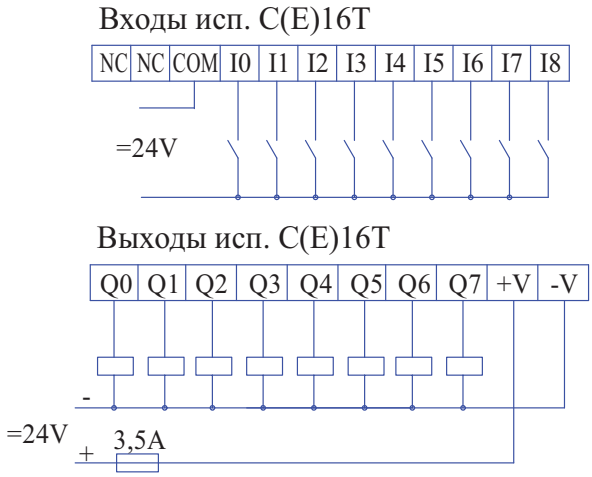


Рис. 7.5. Подключение входов и выходов исп. 16T

Исполнение 24R и 40R: Входы / Выходы

На рис. 7.6 приведена топология клемм подключения дискретных входов и выходов для исполнений 24R и 40R. Проводное подключение аналогично

рис. 7.3 и 7.4. Входы обоих исполнений имеют один общий провод, а выходы – три: COM0, COM1 и COM2.

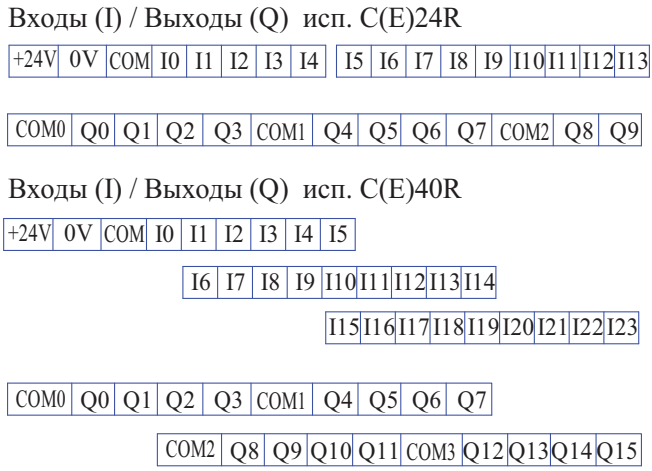


Рис. 7.6. Клеммы подключения входов и выходов исп. 24R и 40R

Исполнение 24T и 40T: Входы / Выходы

На рис. 7.7 приведена топология клемм подключения дискретных входов и выходов для исполнений 24T и 40T. Проводное подключение аналогично рис. 7.5.

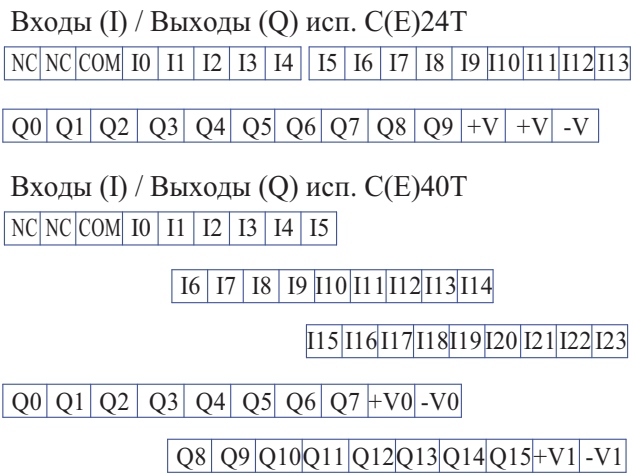


Рис. 7.7. Клеммы подключения входов и выходов исп. 24T и 40T

7.3. Подключение модулей расширения

Общий вид и маркировка расширительных модулей приведена на рис. 7.8.
Расшифровка маркировки:

TM3 Dxxx (X)

! ! !____ G – съемный клеммный блок
! ! К – разъем типа HE10
! ! A, U, без буквы – соединение под винт
! !_____ Модификация блоков:
! DI8 – 8 входов (= 24 В)
! DI16 – 16 входов (= 24 В)
! DQ8R – 8 выходов (Реле)
! DQ8T – 8 выходов (Транзистор)
! DQ16R – 16 выходов (Реле)
! DQ16T – 16 выходов (Транзистор)
! DM8R/G – 4 вх / 4вых (Реле)
! DM24R/G – 16 вх / 8вых (Реле)
! DI32K – 32 входа (= 24 В)
! DI8A – 8 входов (~ 120 В)
!_____ Расширительный модуль

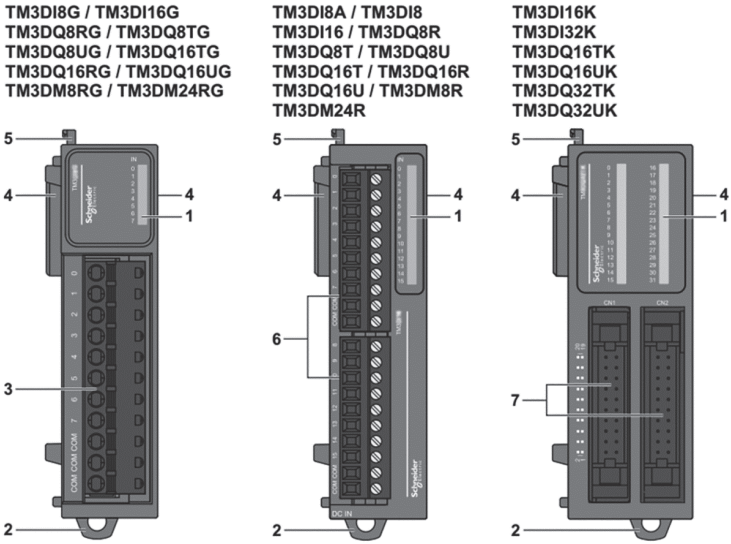


Рис. 7.8. Общий вид модулей расширения

Подключение различных типов расширительных модулей приведено на рис. 7.9 (входы) и рис. 7.10 (выходы).

Полярность питания входов – любая (± 24 В).

Система питания выходов индивидуальна и зависит от типа модуля.

Во избежание ошибок, следует внимательно изучить маркировку клемм и всегда обращаться к исходной технической документации. В документации так же могут быть ошибки и при малейших сомнениях нужно обращаться в сервисный центр. Это касается всех типов и производителей контроллеров.

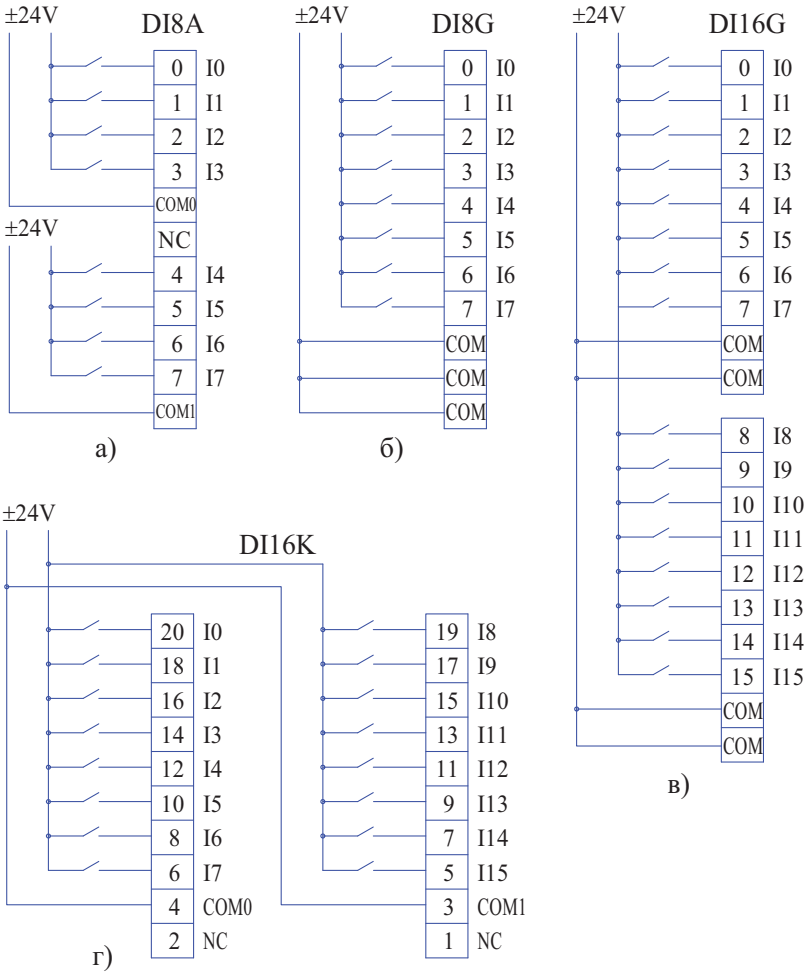


Рис. 7.9. Подключение дискретных входов расширительных модулей:
а – DI8A; *б* – DI8G; *в* – DI16G; *г* – DI16K

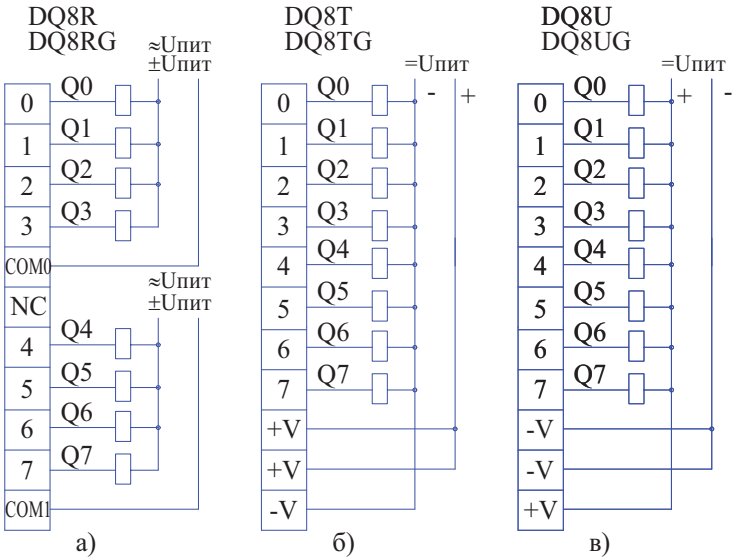


Рис. 7.10. Подключение дискретных выходов расширительных модулей:
а – DQ8R/DQ8RG; *б* – DQ8T/DQ8TG; *в* – DQ8U/DQ8UG

Обозначение адресов входов и выходов во всех приведенных выше схемах подключения условно и всегда начинается с нуля и далее по порядку. Реальные адреса присваиваются автоматически при конфигурации проекта.

На рис. 7.11 приведена топология реального подключения контроллера TM221C40R с двумя расширительными блоками TM3DI16 и TM3DI8 для универсального фрезерного станка.

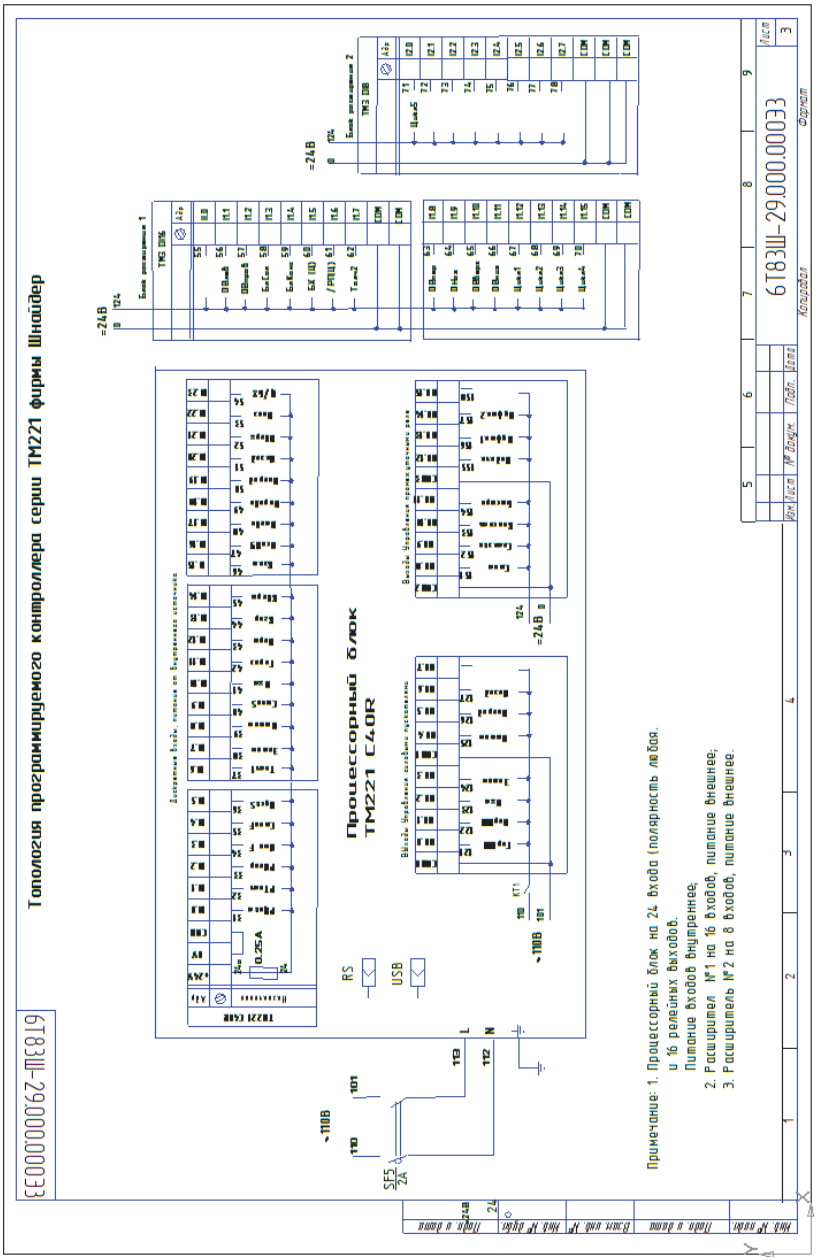


Рис. 7.11. Топология подключения контроллера консольно-фрезерного станка

7.4. Конфигурация проекта

Процедура конфигурации:

– установить на компьютер и запустить программное обеспечение SoMachine Basic (рис. 7.12);
– активизировать «Создать новый проект». Наблюдать появление рабочего окна программного обеспечения;
– активизировать окно конфигурации (Configuration);
– конфигурировать проект, в нашем случае (рис. 7.13):
  процессорный блок TM221C40R;
  расширительный блок TM3DI16 на 16 входов;
  расширительный блок TM3DI8 на 8 входов.

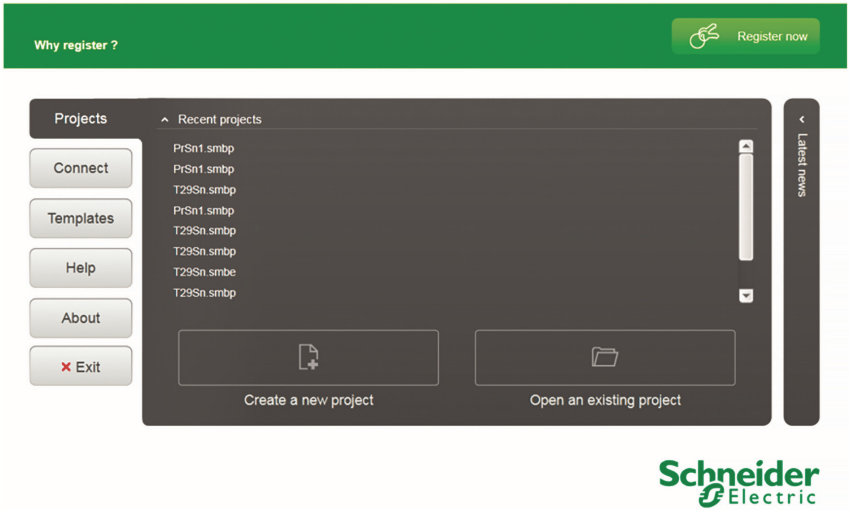


Рис. 7.12. Главная страница SoMachine Basic

При этом автоматически присваиваются адреса для каждого блока:
Процессор C40R: I0.0…I0.23 24 входов;
Q0.0…Q0.15 16 выходов;
Расширитель I16: I1.0…I1.15 16 входов;
Расширитель I8: I2.0…I2.7 8 входов.

При наборе программы перед буквами I (вход) и Q (выход) автоматически устанавливается знак %.

Номер слова (0, 1, 2) определяется последовательностью установки блоков на динрейке.

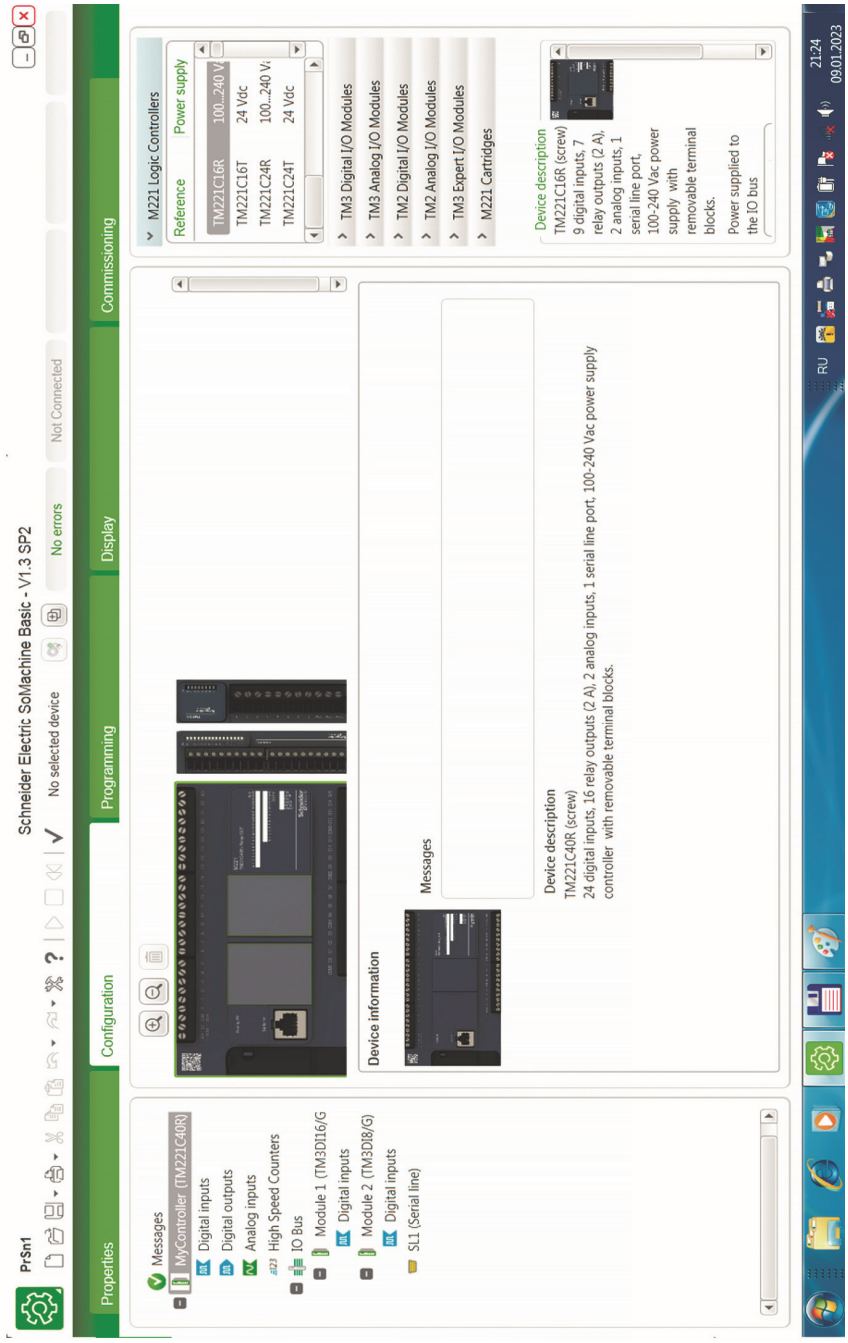


Рис. 7.13. Рабочее окно конфигурации проекта SoMachine Basic

Число бит в слове определяется физическим числом входов и выходов на конкретном блоке.

После завершения конфигурации можно набирать Ladder-диаграмму, предварительно изучив систему команд и разработав теоретические алгоритмы.

7.5. Набор и структура программы

Программирование осуществляется в рабочем окне Programming (рис. 7.14) при помощи набора иконок меню ToolBar. Доступны два языка: Ladder-диаграммы и Programming List (Мнемокод).

Назначение основных иконок Ladder-диаграммы и инструкций Instruction List приведено в табл. 7.2.

На рис. 7.15 приведен фрагмент рабочей программы с раскрытием иконки набора таймеров счетчиков, регистров и других инструкций.

Доступные операнды:

%Ix.x – дискретные входы;
%Qx.x – дискретные выходы;
%Ix.x – дискретные входы;
%Ix.x – дискретные входы;
% IW – входное слово;
% QW – выходное слово;
%Mx – бит оперативной памяти;
%MWx – слово 16 бит;
%MDx – слово 32 бита;
%MFx – слово с плавающей запятой;
%KWx – константа 16 бит;
%KDx – константа 32 бита;
%KFx – константа с плавающей запятой;
%Sx – системный бит;
%SWx – системное слово;
%PLSx – формирователь такта;
%Cx – счетчик;
%FCx – быстрый счетчик;
%HFCx – высокоскоростной счетчик;
%PWMx – широтно-импульсный модулятор;
%TMx – таймер;
%SBRx – сдвиговый регистр;
%Rx – стековая инструкция;
%PIDx – ПИД – регулятор;
%MSGx – сообщение.

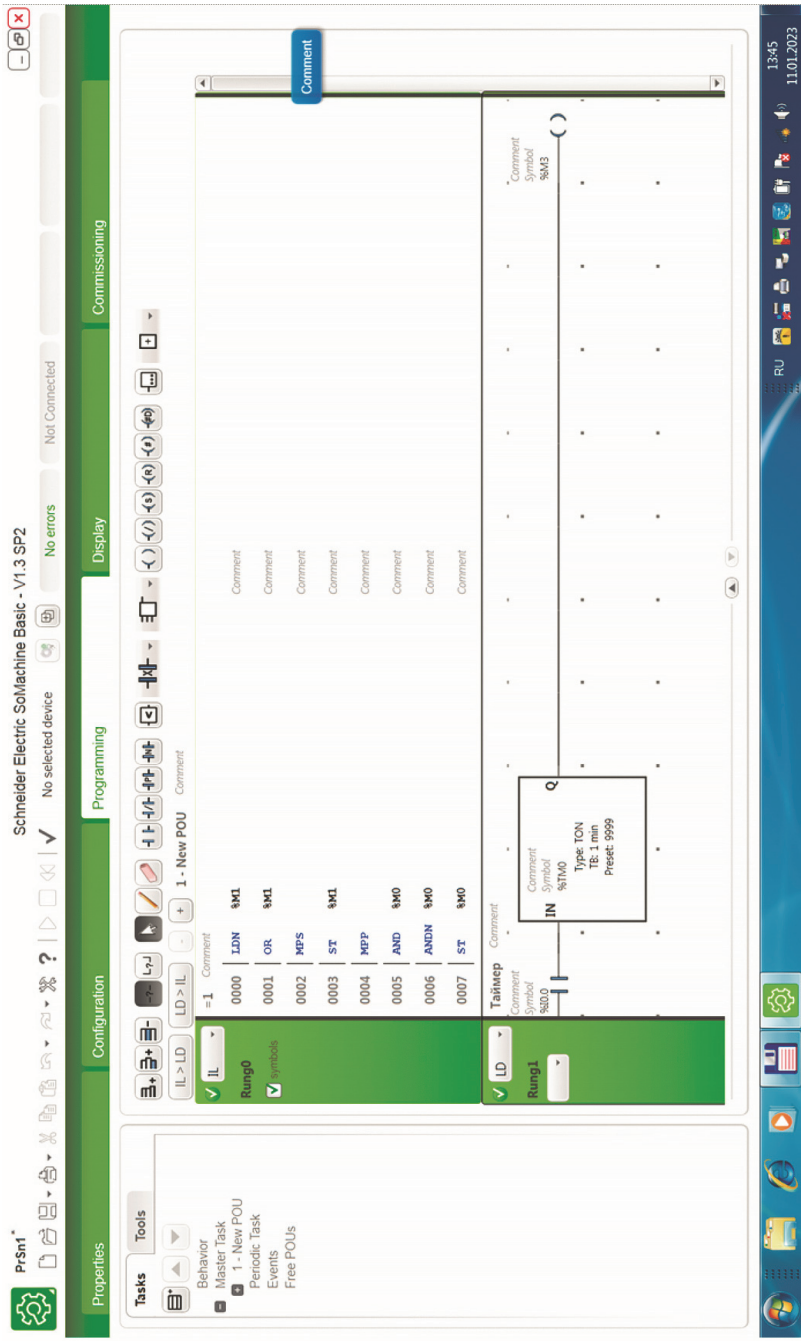
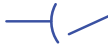
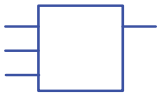
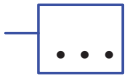
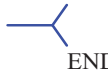


Рис. 7.14. Рабочее окно программирования SoMachine Basic. Пример мнемокода и Ladder-диаграммы

Все операнды предваряются специальным знаком %. При наборе дискретных входов и выходов знак % устанавливается автоматически, в остальных случаях его нужно набирать отдельно.

Таблица 7.2

Назначение иконок набора программ

Ladder	IL	Назначение
	LD	Прямой операнд
	LDN	Инверсный операнд
	LDR	Такт по переднему фронту
	LDF	Такт по заднему фронту
	XOR	Исключающее ИЛИ
	ST	Катушка реле
	STN	Инверсная катушка реле
	S	Установка реле на память
	R	Выключение памяти
		Таймер, Счетчик, Регистр
		Инструкции Сравнения
		Функциональный блок
	N	Логическая Инверсия *)
		Разрыв цепи *)
		Замыкание цепи *)
	END	Конец Программы *)

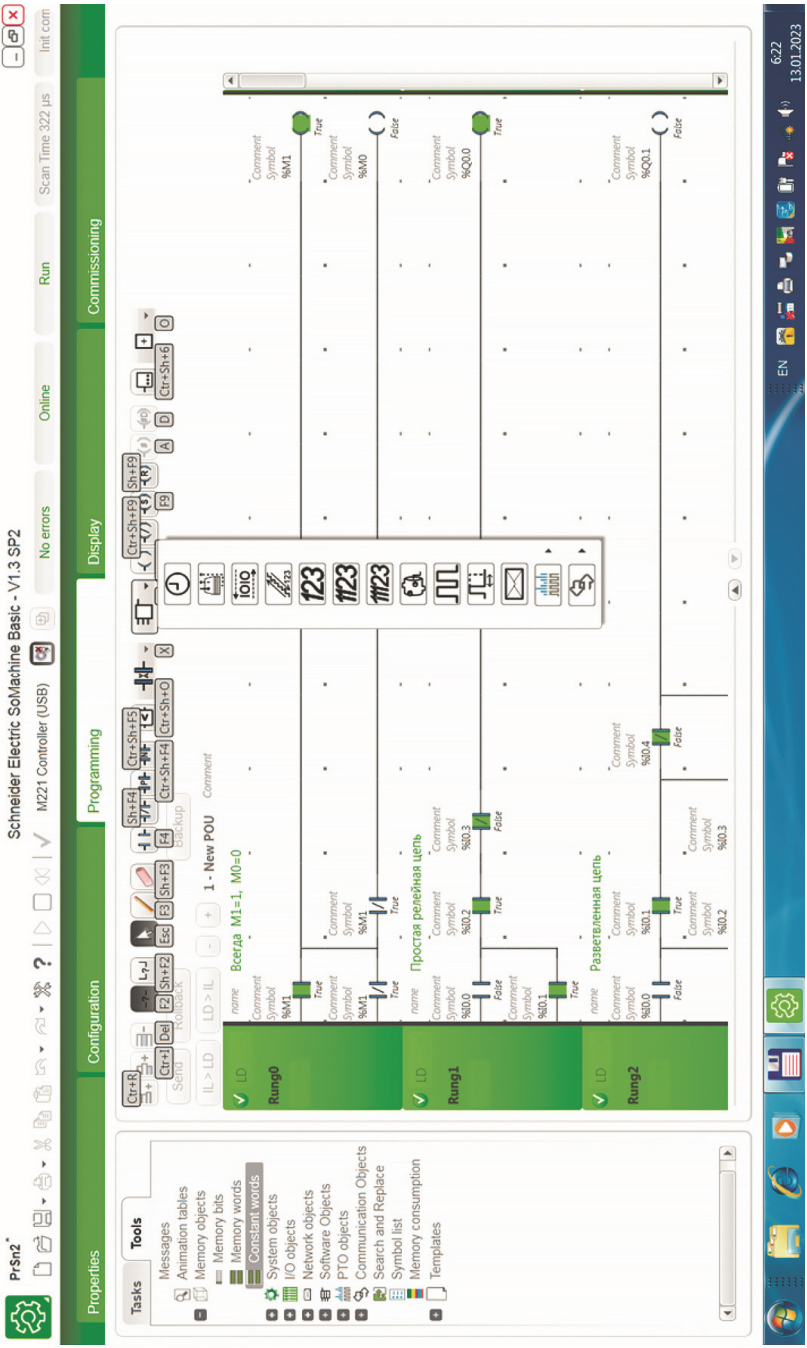


Рис. 7.15. Рабочее окно программирования

Адресация битовых операндов входов (I) и выходов (Q) осуществляется по принципу «номер слова . номер бита».

Адресация адресного пространства битов промежуточной памяти (M), 16-разрядных (MW) и двойных (MD) слов, таймеров (TM), счетчиков (C), регистров – сквозная, т. е. указывается только номер слова или, например, таймера.

Битовая память M и память слов (MW, MD), это независимые адресные пространства, т. е. можно программировать, например, M12, MW12 и MD12.

Общая программа состоит из одной или нескольких подпрограмм POU (Program Operation Unit) и последовательных ступеней (Rang).

Число ступеней Rang зависит от сложности программы.

Структура программы следующая:

Master Task Главная программа
POU_0 Блок (Подпрограмма) 0
RUNG_0 Ступень 0
RUNG_1 Ступень 1
RUNG_2 Ступень 2
. . . .
RUNG_N Ступень N
POU_1 Блок (Подпрограмма) 1
RUNG_0
RUNG_1
. . . . и т. д.

Изучение инструкций и синтаксиса языка на начальной стадии работы с контроллером рекомендуется проводить на компьютере без подключения контроллера с помощью встроенного «Симулятора». Запуск симулятора осуществляется специальной иконкой в горизонтальном меню ПО или из окна Commissioning (рис. 7.16) клавишей Launch Simulator. После ответов на вопросы Софта (Да) и получения квитанции «PLC and Controller application are **identical**» клавишей Start Controller запускается работа симулятора.

Активизация управляющих битов осуществляется «щелчком» мыши компьютера на номер дискретного входа в специальном окне (см. рис. 7.17).

Обладая базовыми знаниями можно выполнять эту же работу, используя аппаратный реальный контроллер, для чего следует выполнить следующие действия:

- поставить на компьютер программное обеспечение SoMachine Basic;
- соединить компьютер и контроллер стандартным кабелем USB (сторона компьютера) – mini USB (сторона контроллера);
- включить питание контроллера;

– включить питание компьютера. При необходимости, в диспетчере устройств компьютера (libusb-win32 devices / M221 Controller) убедится, что компьютер увидел контроллер;
– запустить разработанную программу электроавтоматики, например, PrSn2.smbp и сделать необходимые изменения. Если связь между компьютером и контроллером установилась, то в середине горизонтального меню рабочего окна появится запись M221 Controller (USB), см. рис. 7.16;
– запомнить и выполнить компиляцию программы (кнопка √ в горизонтальном меню);
– кнопкой Online горизонтального меню или клавишей Login меню Commissioning попытаться активизировать работу программы. Если программы на компьютере и в памяти контроллера не совпадают («PLC and Controller application are **differnt**»), то клавишей Download активизировать процесс загрузки программы с компьютера в контроллер. На появляющиеся вопросы отвечать Да;
– при успешной записи и появлении сообщения «PLC and Controller application are **identical**» нажать клавишу Start controller. В горизонтальном меню наблюдать информацию о статусе системы: Online / Run. Связь установлена, контроллер запущен, можно производить проверку работы программы.

Естественно, что для реальной проверки необходимо предварительно собрать стенд и подключить к контроллеру необходимое количество дискретных входов. Выходы можно контролировать по установленным на контроллере светодиодам.

Для внесения исправлений и дополнений в программу клавишей Stop Controller остановить работу и перейти в меню Programming.

Процедура установки связи между PC и PLC для данного контроллера удивительно проста, чего не скажешь о других, рассмотренных нами контроллерах.

7.6. Синтаксис языка электроавтоматики

Данная серия контроллеров имеет достаточно большой набор инструкций программирования электроавтоматики и **сложный** синтаксис написания программ. Инструкции по программированию в большей части не дают четких рекомендаций. По этой причине все приведенные далее фрагменты программ являются рабочими и сняты с экрана в режиме Print Screen.

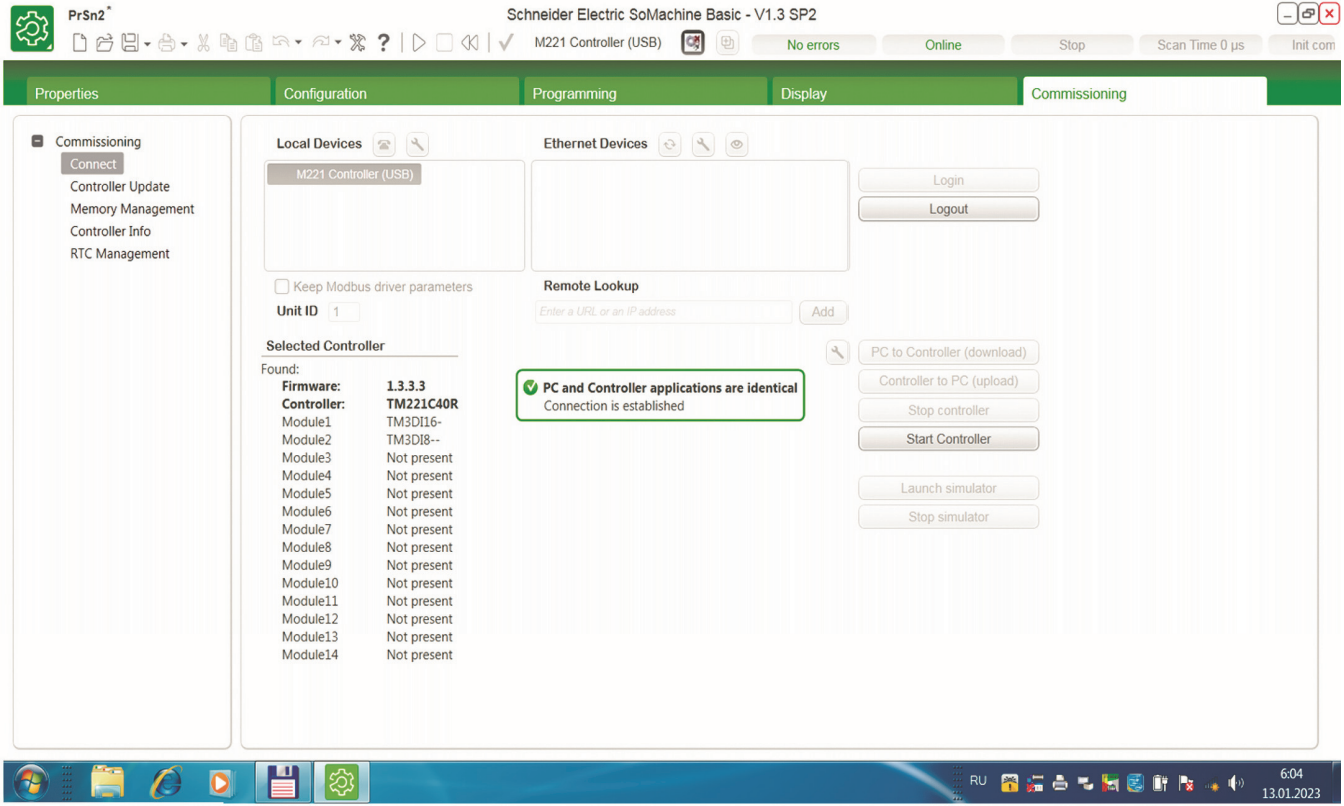


Рис. 7.16. Рабочее окно Commissoning

Автор имеет разовый опыт применения данного контроллера для автоматизации консольно-фрезерного станка, выполненный несколько лет тому назад, поэтому ниже приводятся только основные инструкции языка. Если для решения какой-либо новой задачи потребуются дополнительные знания, то предварительно следует изучить и проверить на стенде или в режиме симулятора необходимые для этого инструкции.

Доступно много готовых системных операндов, например:

%S0 – 1-й скан;
%S4 – генератор T = 10 мс;
%S5 – генератор T = 100 мс;
%S6 – генератор T = 1 с;
%S7 – генератор T = 1 мин;
% S12 = 1 – PLC Run,
= 0 – PLC Stop.

7.6.1. Базовые релейные цепи

Язык позволяет набирать как простые, так и разветвленные релейные цепочки, используя иконки базовых команд (рис. 7.17).

В Rang_0 рис. 7.17 приведена простая цепочка с использованием битовых команд считывания начального операнда, логического сложения, скобок и логического умножения, т. е. Q0.0 = (I0.0 + I0.1)·I0.2·/I0.3.

Здесь знак «/» обозначает инверсию.

При начертании Ladder-диаграмм используются:

– иконки «Карандаш» и «Ластик» для рисования и стирания линий;
– иконки для добавления, вставки и удаления новых блоков Rang;
– стандартные компьютерные команды Cut, Copy, Paste;
– команды Delete Line, Inset Line и др.

Возможно использование следующих комментариев:

Блок Rang: Name – заголовок, например, *Зажим;*
Comment – дополнительная информация, например, *Гл. шпиндель;*
Битовый Операнд: Comment – назначение, например, *Кзаж*.

В Rang_1 рис. 7.17 приведена разветвленная релейная цепь, выполненная в режиме симулятора.

Рис. 7.18 показывает написание тех же программ на языке мнемокода, откуда ясно следует, что использовать этот язык при реальном проектировании не следует, это сложно и трудно отлаживать. Если контроллер позволяет проектировать программы на языке Ladder-диаграмм, то следует использовать исключительно его. Закраска зеленым цветом замкнутого состояния операндов позволяет сразу видеть причину, почему не включается тот или иной выходной операнд.

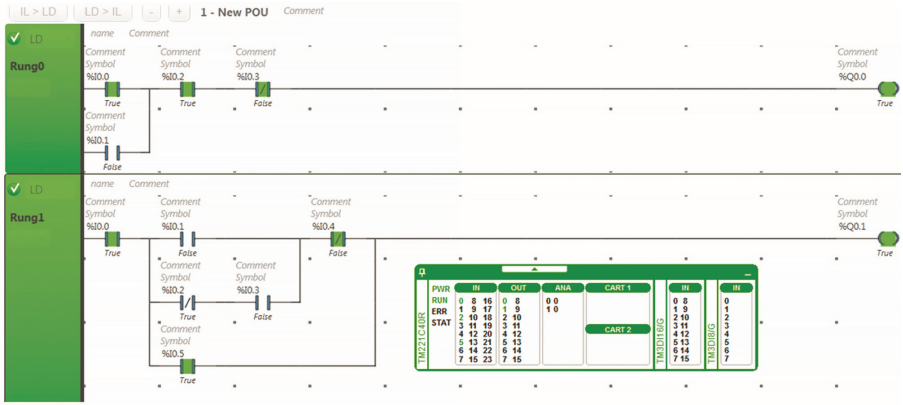


Рис. 7.17. Базовые команды в Ladder

Для того чтобы в одном Rang можно было набирать несколько выходных катушек реле, в начале цепи следует установить один общий операнд. Это позволит одним блоком набирать законченные релейные алгоритмы, относящиеся к управлению конкретного механизма. Если логически такой операнд отсутствует, то нужно использовать вспомогательный = 1 (всегда равно единице). На рис. 7.19 приведен пример алгоритма управления электромеханическим зажимом инструмента универсального фрезерного станка модели 6Т13 горьковского завода фрезерных станков.

Работа механизма зажима инструмента запрещена, если:

– включен главный (Шп1, Кшп1) или второй (Шп2, Кшп2) шпиндель;
– Включено торможение второго шпинделя (Торм2);
– Работает автоматический цикл (Цикл), т. е.

$$Запр=Шп1+Кшп1+Шп2+Кшп2+Торм2+Цикл.$$

Для выполнения операции зажима двигатель от кнопки *КнЗаж* при условии отсутствия запрета включается в направлении вращения по часовой стрелке, а выключается по датчику контроля зажима *Дзаж*.

Для отжима инструмента кнопкой КнОтж двигатель включается в противоположную сторону. Кнопка удерживается до окончания операции отжима, при этом оператор должен не дать инструменту упасть на рабочий стол.

Таким образом, процесс описывается следующими логическими уравнениями:

$$ЗАЖ: S = КнЗаж$$

$$R = Запр + Кзаж + ОТЖ \text{ и } /R = /Запр*/Кзаж*/ОТЖ$$

Здесь и далее:
S – сигнал включения памяти (Set);
R – сигнал выключения памяти (Reset).

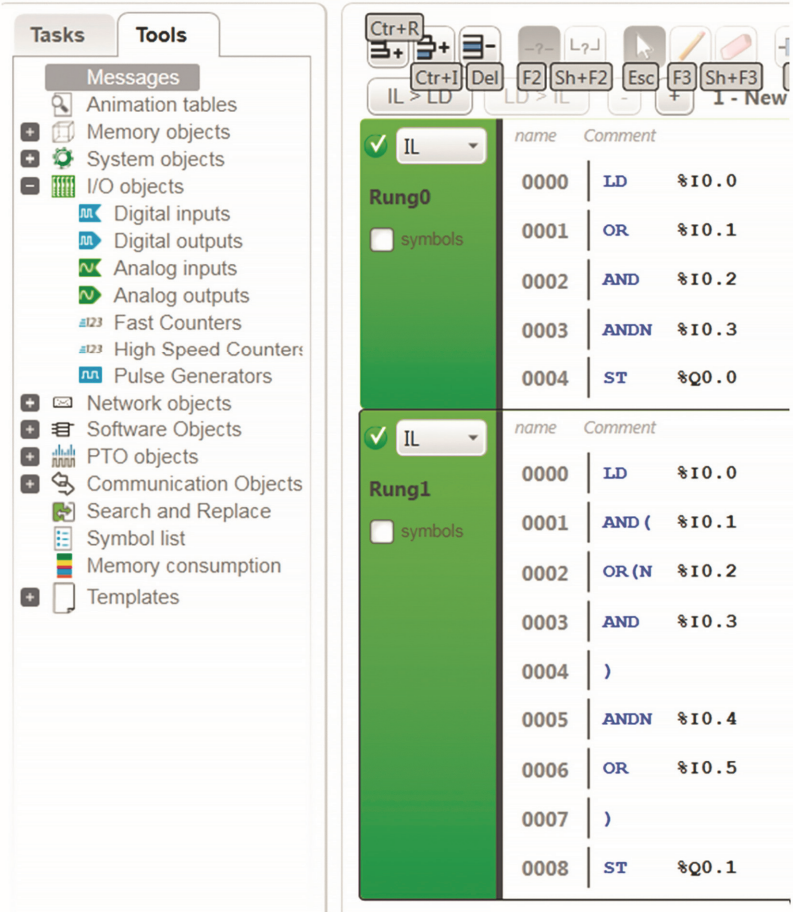


Рис. 7.18. Базовые команды в Instruction List

Для правильного начертания алгоритма сигнал выключения памяти R необходимо проинвертировать, так как релейная RS-память выключается инверсным сигналом.

ОТЖ = КнОтж*/Запр + *ЗАЖ

Кзаж: S = ЗАЖ*Дзаж

$$R = ОТЖ \text{ и } /R=/ОТЖ$$

$$T2 = /ЗАЖ*Котж$$

$$Кзаж(t) = /ЗАЖ*Котж*T2(t)$$

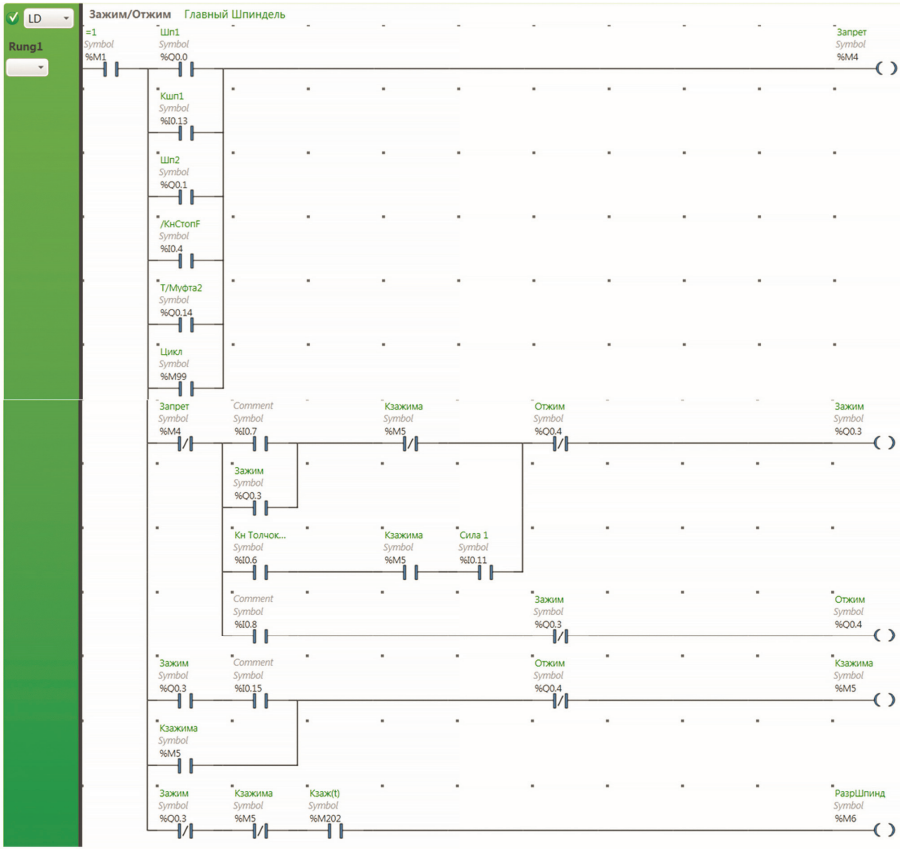


Рис. 7.19. Алгоритм управления электромеханическим зажимом инструмента

Сигнал Кзаж(t) разрешает включение шпинделя по окончании процесса зажима. Его формирование **обязательно**, так как в противном случае, при включении шпинделя, инструмент может «вылететь» из шпинделя со всеми вытекающими последствиями. Формирование этого сигнала на рис. 7.19 не показано, так как синтаксис языка разрешает набирать инструкции таймеров только в отдельном Rang, т. е его нужно набирать отдельно.

Во избежание *проворота* шпинделя при выполнении операций зажима и отжима инструмента следует также в отдельном Rang предусмотреть его

удержание в неподвижном состоянии при помощи тормозной муфты, т. е. реализовать следующее уравнение

Т/Муфта1 = /Шп1*/Кшп1*(ЗАЖ*/Кнт1+ОТЖ).

7.6.2. Таймер

Предусмотрено три типа таймера:

TON – выдержка времени при включении входного сигнала;

TOF – выдержка времени при выключении входного сигнала;

TP – одновибратор, т. е. формирователь импульса заданной длительности по переднему фронту входного сигнала в независимости от его длительности.

Таймер набирается в отдельной строке Rang. Дополнительные логические цепи запрещены. На рис. 7.20 приведен пример таймера на включение TON.

Набор таймера осуществляется из вертикального меню иконки функциональных инструкций (см. рис. 7.15). В шаблоне инструкции указывается:

– %TM – номер таймера;
– тип таймера (TON, TOF или TP);
– величина дискреты времени TB: 1 ms, 10 ms, 100 ms, 1 s или 1 m;
– число дискрет Preset.

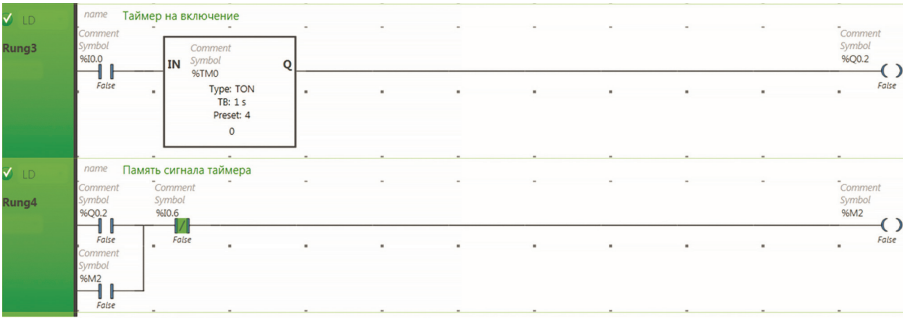


Рис. 7.20. Таймер на включение TON

Для установки параметров необходимо щелкнуть мышкой по номеру таймера &TMx и заполнить открывшуюся таблицу.

Для дальнейшей обработки доступны:

%TMx.P – величина уставки (Present Value);

%TMx.V – текущее состояние (Carrent Value);

Прямой и инверсный контакты выходного реле таймера.

Комментарий к рис. 7.20:

При активизации входного сигнала I0.0 через выдержку времени 4 секунды включается выходное реле таймера Q0.2 (Rang3), активизируется и встанет на самопитание катушка реле M0.2 (Rang4). При выключении входного сигнала I0.0 таймер и его катушка сбрасываются в исходное состояние.

Если входной сигнал отключится раньше заданной уставки, то текущее состояние времени не сохраняется и при новой активизации входа отсчет времени начинается с нуля.

7.6.3. Счетчик

В контроллере предусмотрено три типа счетчиков:

%Cx – классический универсальный счетчик (Counter), x = (0–255);

%FCx – быстрый счетчик (Fast Counter), x = (0–3);

%HFCx – высокоскоростной счетчик (High Fast Counter) , x = (0–3).

Пример стандартного счетчика Cx показан на рис. 7.21.

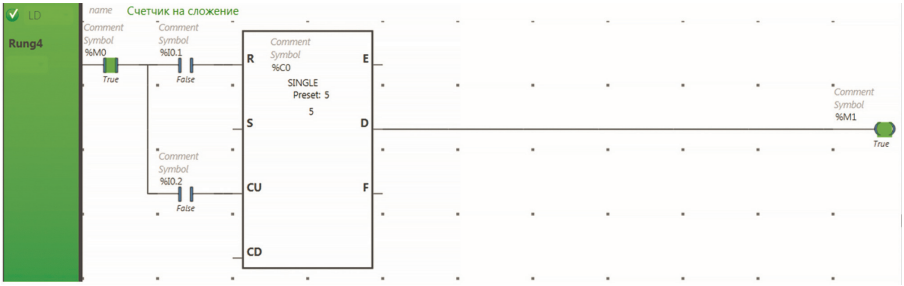


Рис. 7.21. Счетчик на сложение

Счетчик имеет четыре входа:

R (Reset) – сброс текущего состояния счетчика, Cx.V = 0 при R = 1;

S (Set) – принудительная установка текущего состояния, Cx.V = Preset при S = 1;

CU (Counter Up) – вход счета на увеличение, Cx.V = (Cx.V + 1) при CU = 1;

CD (Counter Down) – вход счета на вычитание, Cx.V = (Cx.V – 1) при CD = 1.

Счетчик имеет три выхода:

E (Empty) – счетчик в нуле, Cx.E =1 при Cx.V = 0;

D (Count Done) – уставка счета достигнута, Cx.D = 1 при Cx.V = Cx.P;

F (Counter Full) – счетчик переполнен, Cx.F = 1 при Cx.V = max.

Максимальная уставка счета – 9999.
Повтор счета после сброса.

7.6.4. Триггеры

На рис. 7.22 приведено программирование следующих типов триггеров:

RS – триггер (M5) с использованием инструкций S (Set) и R (Reset);

T – триггер (M6) с использованием команды

XOR – «Исключающее ИЛИ»;

T – триггер (M8) с тактированием входного сигнала M7 и битовой схемой неравнозначности.

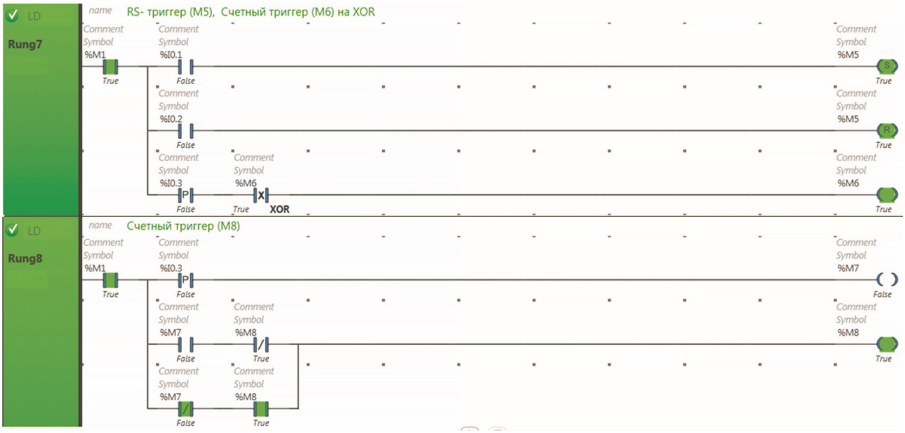


Рис. 7.22. Триггеры

7.6.5. Присвоение

На рис. 7.23 приведен синтаксис наиболее распространенных вариантов присвоения (аналоги команды пересылки MOVE):

(%M40:8 := %I0.0:8) – пересылка 8-ми бит входов I0.0–I0.7 в 8 ячеек битовой памяти M40–M47;

(%MW41 := 50) – запись десятичного числа 50 в 16-разрядное слово MW41;

(%MW42 := 16#300) – запись 16-теричного числа 300 в 16-разрядное слово MW42;

(%MW43 := %KW0) – запись десятичной константы из памяти KW0 в 16-разрядное слово MW43.

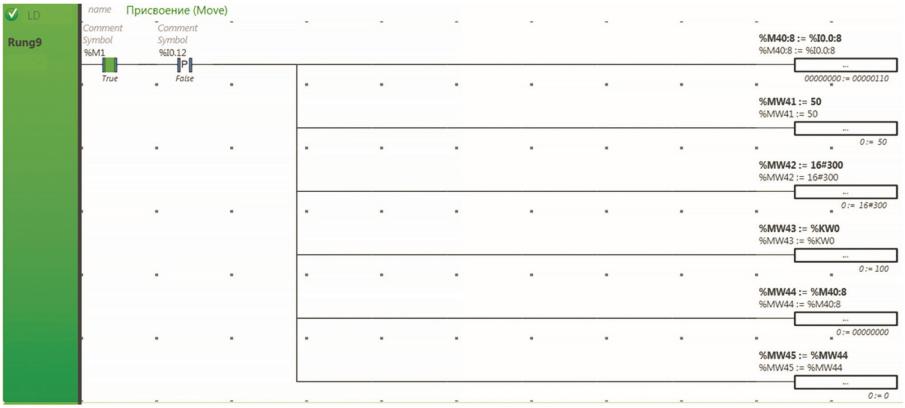


Рис. 7.23. Присвоение

Запись констант в память осуществляется предварительно через левое вертикальное меню окна Programming;

(%MW44 := %M40:8) – запись содержимого массива памяти M40-M47 в 16-разрядное слово MW44;

(%MW45 := %MW44) – перепись содержимого слова MW44 в слово MW45.

Общее замечание: содержимое источника при переписи не изменяется.

7.6.6 Сравнение

Примеры программирования сравнения приведены на рис. 7.24:

M10 = 1, если содержимое слов MW46 и MW45 одинаково;

M11 = 1, если содержимое слова MW42 больше содержимого слова MW41;

M12 = 1, если содержимое слова MW41 меньше константы, записанной в слове KW0;

M13 = 1, если текущее значение счетчика C0 меньше 5-ти;

M14 = 1, если текущее значение таймера TM0 больше 3-х.

Обращаем внимание, что топология изображения инструкций сравнения всегда одинакова (со знаком <), вне зависимости от типа сравнения. Какая операция выполняется реально, следует определять по синтаксической записи над прямоугольником инструкции. В режиме Online отображается содержимое сравниваемых величин.

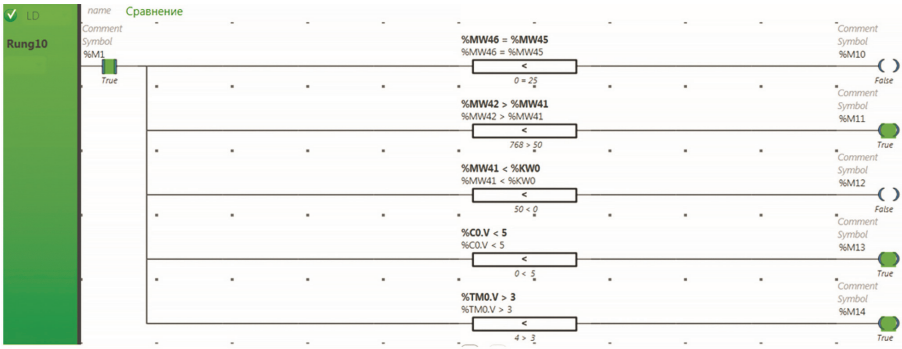


Рис. 7.24. Инструкции сравнения

7.6.7. Арифметические инструкции

На рис. 7.25 в качестве простейших примеров приведены инструкции арифметических вычислений сложения, вычитания, умножения, деления и извлечения квадратного корня с целыми десятичными числами и использовании 16-тиричных слов. Каких-либо пояснений здесь не требуется. Внимательно просмотрите синтаксис записи инструкций. При сложных расчетах используется косвенная адресация и двойные слова.

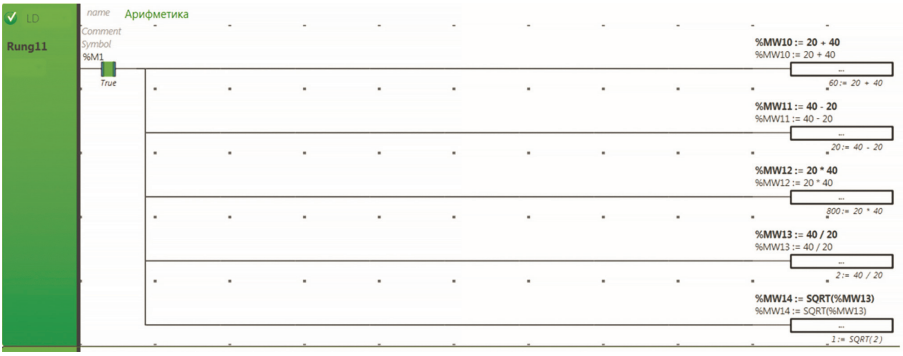


Рис. 7.25. Арифметические инструкции

7.6.8. Инструкции INC / DEC

Пример программирования инструкций Инкремента и Декремента приведен на рис. 7.26. Обязательным условием является тактирование входных сигналов.

Рис. 7.26. Инструкции INC_DEC

7.6.9. Инструкции ITB / BTI

Инструкция ITB выполняет преобразование двоичного кода (BIN) в двоично-десятичный код (BCD).

Инструкция BTI выполняет обратное преобразование двоично-десятичного кода (BCD) в двоичный код (BIN).

Комментарий к рис. 7.27:

– в слове MW16 импульсом I1.0 установлен в единицу бит I0.5 (0000 0000 0010 0000), что соответствует числу 32 в двоичном коде или числу 20 в двоично-десятичном;
– инструкция %MW17 := ITB (%MV16) преобразует двоичное число 32 слова источника MW16 в двоично-десятичное число 50 (0000 0000 0011 0010, т. е. 32 + 16 + 2) и записывает его в слово приемника MW17;
– инструкция %MW18 := BTI (%MV16) преобразует двоично-десятичное число 32 слова источника MW16 в двоично-десятичное число 20 и записывает его в слово приемника MW18;

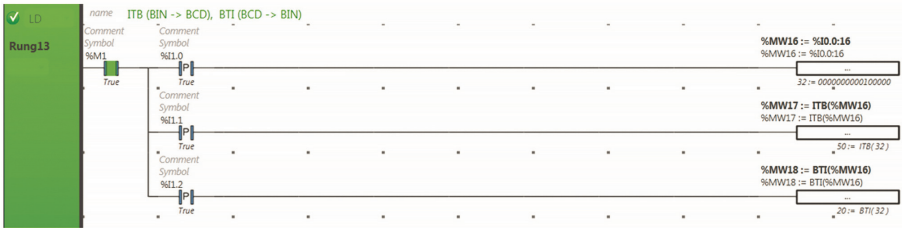


Рис. 7.27. Инструкции ITB_BTI

7.6.10. Инструкции SHR / ROR

Инструкция SHR, это выталкивающий сдвиговый регистр вправо.
Синтаксис регистра %MW19 := SHR (%MW19,1).
Инструкция ROR, это кольцевой сдвиговый регистр вправо.
Синтаксис регистра %MW20 := ROR (%MW20,1).

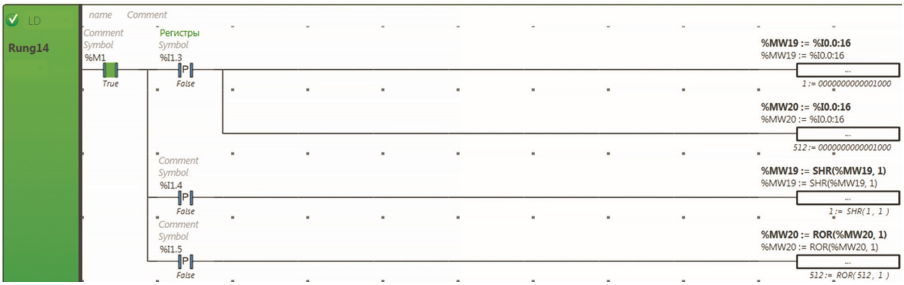


Рис. 7.28. Инструкции SHR_ROR

Комментарий к рис. 7.28:

- импульсом начальной установки I1.3 в регистры SHR (MW19) и ROR (MW20) предварительно записывается начальная информация, в данном случае число 8 (0000 0000 0000 1000);
- в инструкциях регистров, через запятую, указывается число сдвигаемых разрядов, в данном примере 1;
- при каждом тактированном входном сигнале информация в регистрах сдвигается на один разряд (в примере вправо);
- при последнем сдвиге выталкивающий регистр MW19 обнуляется, а в регистре MW20 информация сдвигается по кольцу.

Инструкции сдвига влево SHL и ROL работают аналогично.

Приведенный в главе материал это только часть языковых средств контроллера. За кадром остались, например, инструкции логических операций со словами, условные переходы, подпрограммы, стековые инструкции, тригонометрия, работа с числами с плавающей запятой и др. Заинтересованного читателя отправляем к сопроводительной документации на контроллер.

ГЛАВА 8. ПАНЕЛИ ОПЕРАТОРА

8.1. Общие сведения, классификация

Графические панели оператора предназначены для управления различным промышленным оборудованием и осуществления в реальном времени диагностики работы технологического процесса, причем отображение процесса может осуществляться как в статическом, так и в динамическом виде. Ввод и отображение различных параметров на экране может выполняться в цифровом или графическом виде. Они с успехом применяются и для управления универсальными металлорежущими станками. Язык программирования панелей включает большой набор команд, позволяющих создавать удобный интерфейс между оператором и объектом управления. На нем можно отобразить любые кнопки управления, сигнальные лампы, программировать графики, осуществлять индикацию различных технических параметров, например, напряжение, ток, скорость, задавать и контролировать размерные перемещения координат, диагностировать состояние дискретных датчиков.

Таким образом, панели оператора являются основным средством для организации удобных интерфейсов между человеком и машиной, т. е. HMI-интерфейсом.

Обращаем внимание, панель оператора не является изделием, позволяющим автономно управлять металлорежущим станком или каким-либо другим объектом.

При помощи панели оператора формируются удобные, отвечающие требованиям заказчика, органы управления. Это важная часть общей системы электроавтоматики, включающей программируемый контроллер, электроприводы, прочее низковольтное электрооборудование (рис. 8.1).

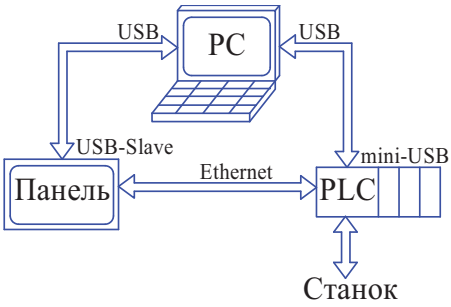


Рис. 8.1. Блок-схема управления с использованием панели оператора

Последовательность освоения и проектирования панелей следующая:

- изучить технические характеристики объекта управления и выбрать панель, отвечающую его требованиям;
- разобраться с аппаратным подключением панели;
- изучить синтаксис языка программирования;
- поставить на компьютер требуемую программную среду;
- сформулировать задачу и написать программу, позволяющую отобразить необходимые органы управления на экранах панели;
- выполнить предварительную проверку проекта на симуляторе компьютера;
- организовать связь панели с компьютером, загрузить в панель разработанную программу и ее отладить.

Это только часть общего проекта электроавтоматики. После разработки логической части PLC-программы необходимо связать панель с программируемым контроллером и осуществить наладку их совместной работы.

Различными фирмами выпускается большая номенклатура панелей оператора, различающаяся цветностью и размерами экрана, способами и протоколами связи с PLC и компьютером, объемом памяти и языком программирования.

Различают:

- текстово-графические терминалы со стандартными или многофункциональными панелями с кнопочным управлением без встроенного PLC;
- сенсорные панели, различаемые по размеру экрана, без встроенного PLC;
- текстово-графические и сенсорные панели со встроенным PLC;
- специализированные сенсорные панели с большим размером экрана;
- компактные выносные пульты управления с сенсорным экраном и др.

Ниже рассматривается процедура программирования на примере панели оператора типа DOP-110WS фирмы «Дельта».

Основные характеристики панели:

- габаритные размеры;
- жидкокристаллический экран 10.1 дюйма;
- разрешающаяся способность 1024×600 пикселей;
- 65536 цветов;
- питание напряжением = 24 В;
- поддержка USB-порта;
- наличие двух COM-портов, поддержка связи по RS485;
- наличие порта Ethernet;

– центральный процессор – ARM Cortex-A8 (800 Mhz);
– оперативная память 512 Мб;
– возможность использования русского языка;
– поддержка основных протоколов локальных систем управления;
– наличие большой базы данных аппликаций различных элементов управления, задания и контроля систем управления;
– прочный корпус с классом защиты IP65.

8.2. Последовательность создания проекта

1. Поставить на компьютер и открыть специальное программное обеспечение Soft V4.0.8.

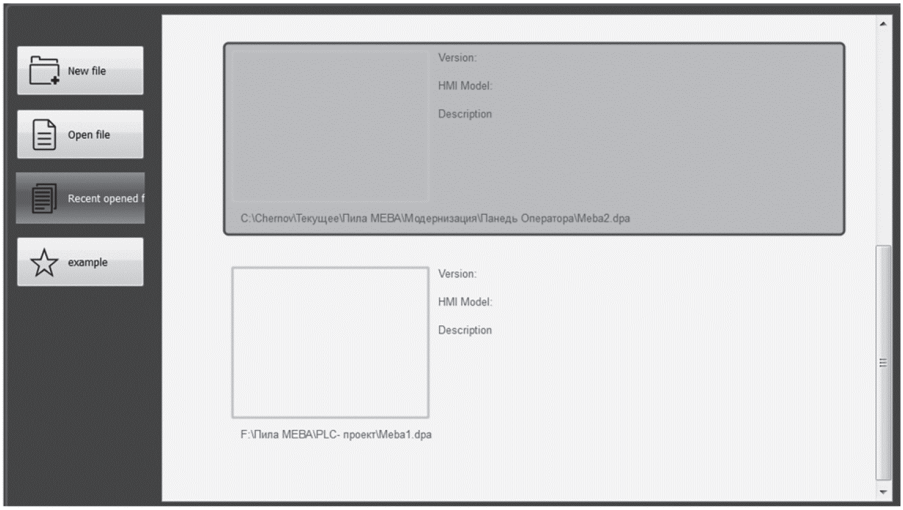


Рис. 8.2. Начальное окно Soft V4.0.8

2. В открывшемся начальном окне (рис. 8.2) щелкнуть по клавише New file.

3. В окне Project Wizard:
выбрать серию (DOP-100) и тип (110WS) панели оператора,
установить имя проекта (Том4_HMI),
имя (Screen1) и номер (1) первой панели,
язык (Russia) общения (рис. 8.3).

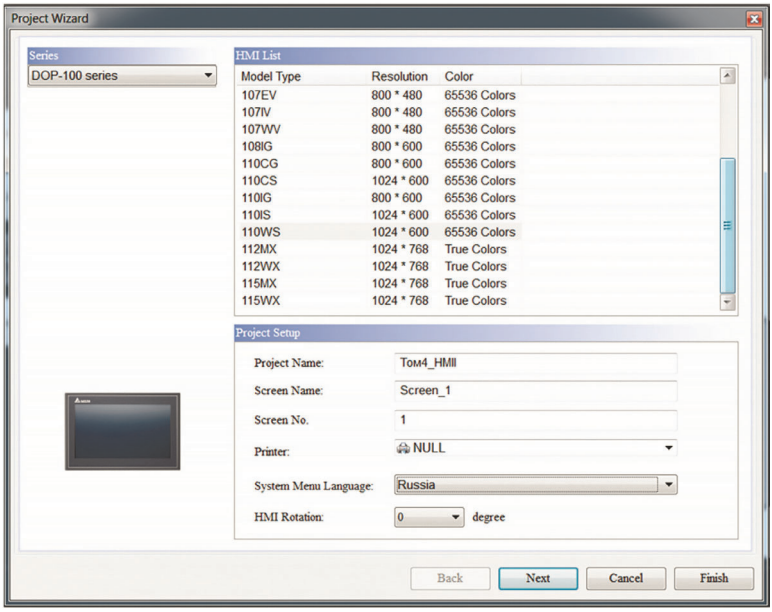


Рис. 8.3. Окно установки типа панели

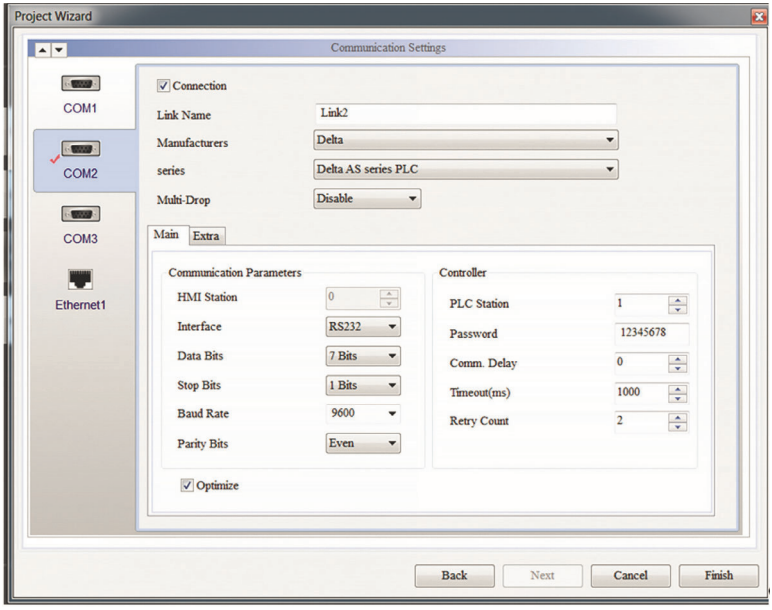


Рис. 8.4. Окно параметров коммуникации

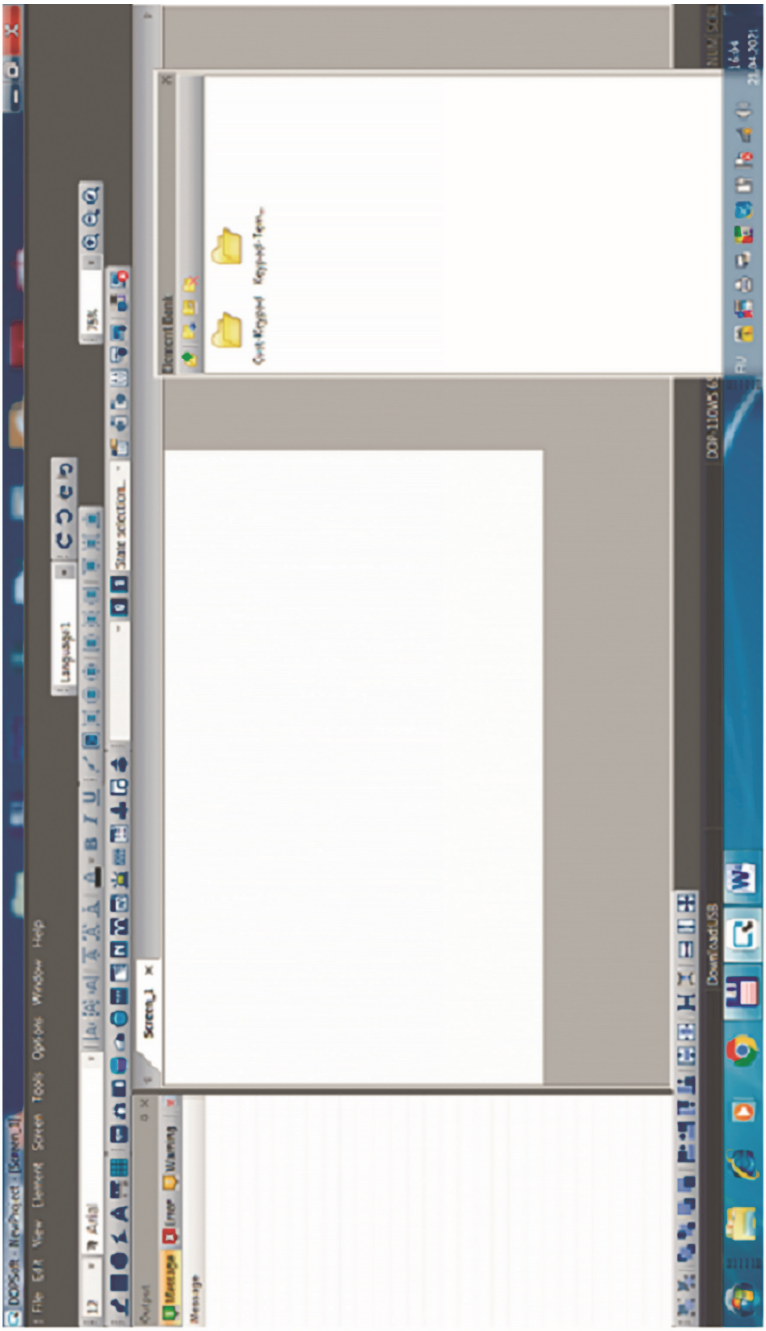


Рис. 8.5. Окно проектирования первой панели

4. Нажать клавишу Next и в следующем окне установить параметры коммуникации (рис. 8.4):

Link Name Link 2
Manufactures Delta
Series Delta AS Series PLC
Multi_drop Disable

Прочие параметры на начальном этапе оставить без изменения.

5. Нажать клавишу Finish. На экране персонального компьютера появится окно проектирования первой панели (рис. 8.5).

6. Спроектировать панель, используя заранее подготовленный черновик органов управления, располагаемых в данном окне и инструментарий, предоставленный разработчику электроавтоматики фирмой-изготовителем панели. Расширение имени проекта – «**.dpa**».

8.3. Синтаксис языка проектирования панели

Синтаксис языка проектирования панелей предоставляет пользователю огромные возможности и подробно (и часто непонятно) изложен в сопроводительной технической документации DOP Soft User Manual, состоящей из 2244 страниц. Но не нужно пугаться. Это чисто техническая работа. Для проектирования органов управления универсальными станками нужно знать не так уж и много. Можно во всем разобраться методом проб, ошибок и их исправления. Кроме того, в программном обеспечении проектирования предусмотрен режим симуляции, позволяющий проверять основные моменты проекта непосредственно на компьютере без использования аппаратной панели.

Главное – это грамотно разработать идеологию управления станком, т. е. определить количество панелей, состав их органов управления, последовательность переключения, т. е. продумать удобный и простой для оператора интерфейс. Нужно по минимуму ограничить номенклатуру используемых аппликаций и как можно меньше применять сложные рисунки, тогда все будет в порядке.

Обратимся к основному окну Soft V4.0.8 (см. рис. 8.5).

В нем предусмотрен стандартный набор падающих меню:

- File (Открытие, сохранение, печать файлов);
- Edit (Редактирование файлов);
- View (Просмотр, конфигурация);
- Element (Выбор элементов);
- Screen (Создание и открытие окон);
- Tools (Компиляция, загрузка в панель, симулятор);
- Options (Конфигурация, изменение среды);

– Window (Работа окнами);
– Help (Подсказки).

Оперативные чертежные иконки
– Line (линия);
– Rectangle (прямоугольник);
– Circle (окружность);
– Polygon (полилинии);
– Static Text (текст);
– Scale (шкалы);
– Table (таблицы).

Оперативные иконки вызова аппликаций
– Button (кнопки);
– Meter (измерители);
– Bar Graf (разные наборы);
– Pipe Graf (трубы) и др.

Каждая иконка вызова аппликаций имеет вложенную разветвленную входимость. Так при активизации аппликации иконки «Button – кнопка» появляется возможность выводить на экран различные типы кнопок, например:

Set to On (установка включенного состояния);
Set to Off (установка выключенного состояния);
Momentary (установка включенного состояния только при нажатии);
Maintained (установка единичного состояния с запоминанием);
Multistate (многофункциональная кнопка);
Goto Screen (кнопка вызова другого окна) и многие другие.

Аналогично по всем другим иконкам.

Прежде чем делать реальный проект следует изучить предлагаемый инструмент и выбрать из него наиболее подходящие аппликации, записав их местоположение в базе данных.

Теперь рассмотрим процедуру переноса аппликаций кнопок из базы данных в окно проектируемой панели:

– раскрыть меню кнопок и выбрать тип кнопки;
– перетащить мышкой выбранный тип кнопки в любое место экрана и сделать двойной «Клик». На экране появится едва заметный квадратик размером 1×1 мм и окно атрибутов;
– увеличить размер квадрата. Это можно сделать двумя способами: произвольно растащить квадратик мышкой или установить его конкретные размеры в параметрах. На данном этапе работы, на экране будет изображен обезличенный серый квадрат (см. рис. 8.6);

– перетащить квадрат в удобное для дальнейшей работы место;
– в меню Picture выбрать из базы необходимый тип кнопки, например, для рис. 8.7: Picture Bank Name – $3D Button.pib → BUTTON_55.bmp;
– в меню Main выбрать атрибуты кнопки:
  Style → Raised
  Foreground Color → серый
  Write Adress → M832, Device → Bit
  Read Adress → M470
  State: 0 (желтый)
  1 (зеленый);
– в меню Text задать текст на кнопке, его размер и стиль;
– в меню Coordinates установить размеры (Width*Height) и координаты (X, Y) аппликации. **Внимание!** Размеры и координаты автоматически устанавливаются, если их задавать мышкой.

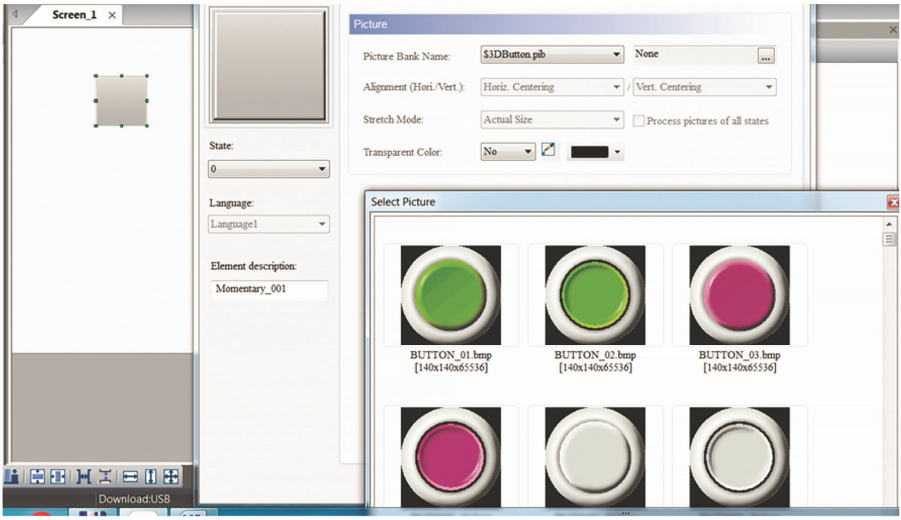


Рис. 8.6. Окна проектирования первой панели

Таким образом, мы спроектировали прямоугольную кнопку с серым обрамлением установки режима работы «Толчок». Кнопка имеет адрес M832 и ее цвет в исходном положении – желтый. При кратковременном нажатии кнопки через электроавтоматику PLC активизируется установка режима «Толчок» (адрес M470) и цвет кнопки становится зеленым. Выключение режима осуществляется вторичным нажатием на эту же кнопку.

Аналогично проектируются любые другие органы управления.

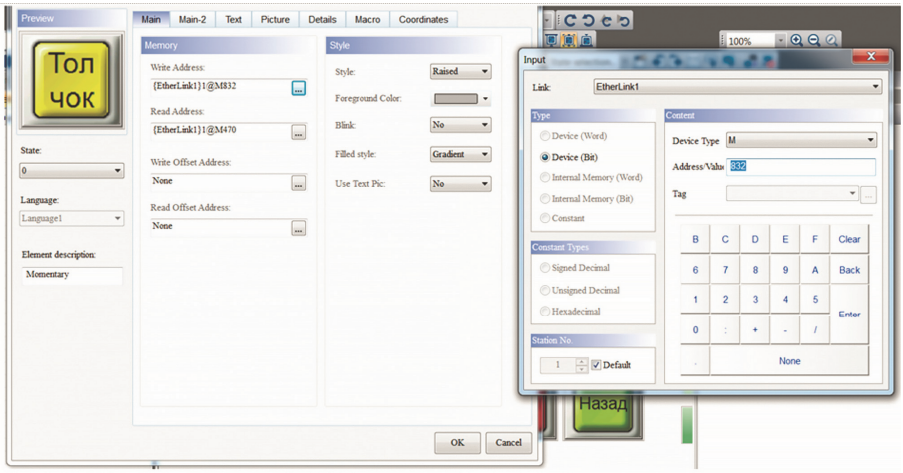


Рис. 8.7. Окна установки атрибутов кнопки

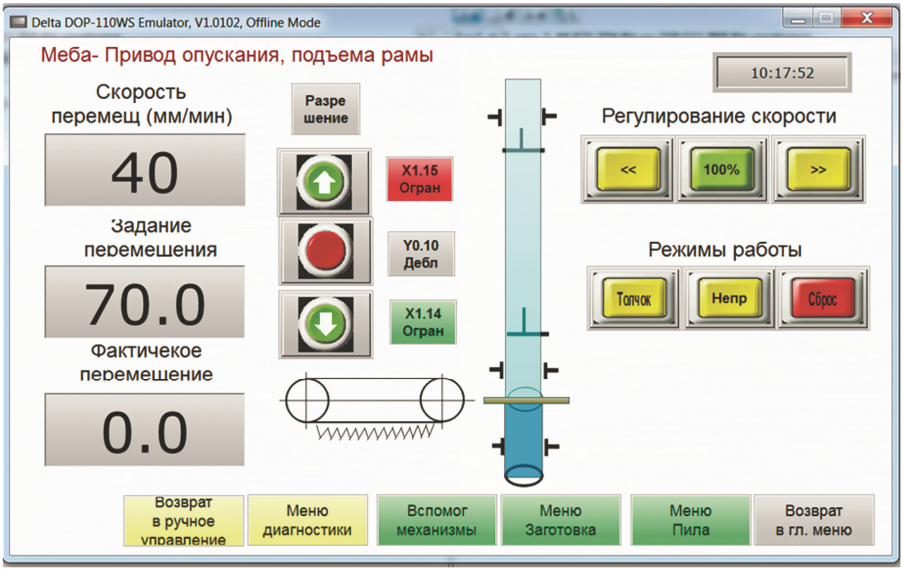


Рис. 8.8. Экран управления приводом перемещения пилы

На рис. 8.8 приведен пример рабочего экрана управления перемещением рамы автоматической пилы фирмы «Меба». Экран предусматривает следующие органы управления:

– задание скорости перемещения рамы при резании;
– задания величины и индикации фактического перемещения рамы;
– выбор режима работы: толчковое или непрерывное перемещение рамы;
– кнопки оперативного изменения скорости;
– кнопки задания перемещения Вверх и Вниз;
– индикация состояния ограничительных конечных выключателей и сигналов деблокировки привода;
– кнопки перехода в меню отладки и главное меню;
– часы времени.

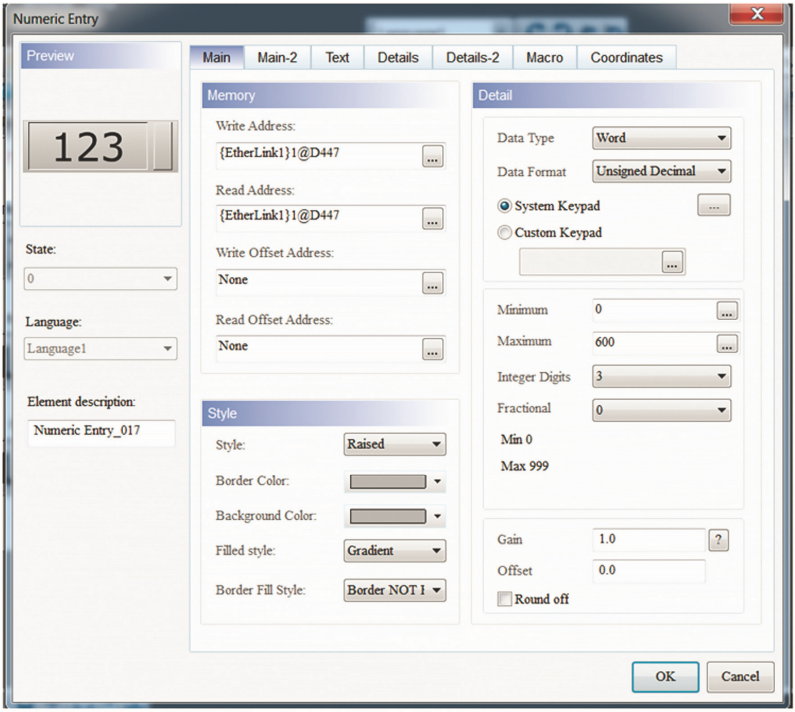


Рис. 8.9. Окно установки атрибутов скорости

Доступ к атрибутам задания скорости, величины перемещения и его контроля (рис. 8.9) осуществляется по следующему пути

Element → Input → Numeric Entry;

Доступ к атрибутам стандартных кнопок

Element → Button → Momentary;

Доступ к атрибутам индикации

Element → Indicator → Multistate Indicator
Доступ к атрибутам текста
Element → Drawing → Text
Доступ к атрибутам установки времени
Element → Data Display → Time Display
Доступ к атрибутам кнопок перехода в другие окна (рис. 8.10)
Element → Button → Goto Screen

Создание нового окна осуществляется путем открытия окна его атрибутов по пути Screen → New Screen, где необходимо установить имя, номер и тип окна.

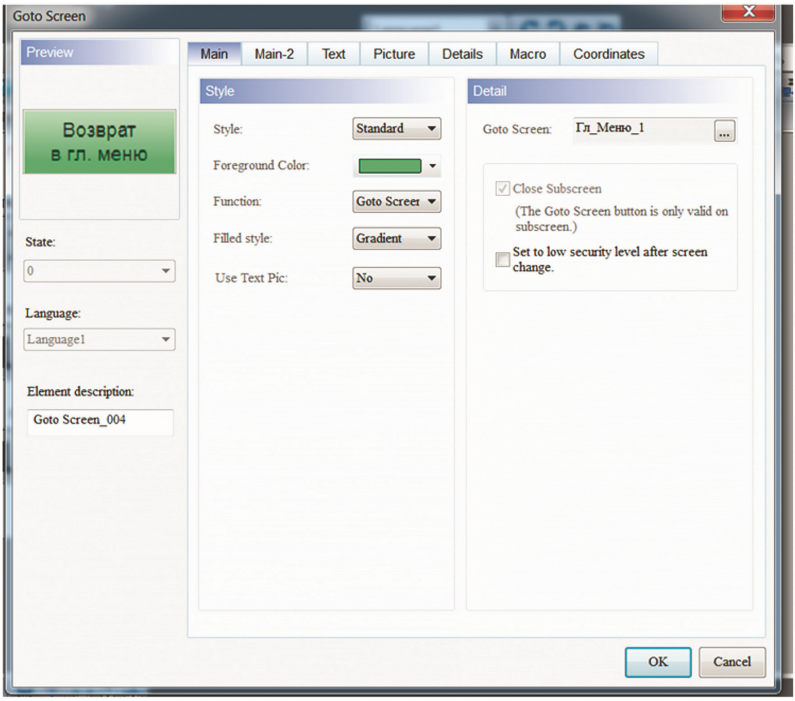
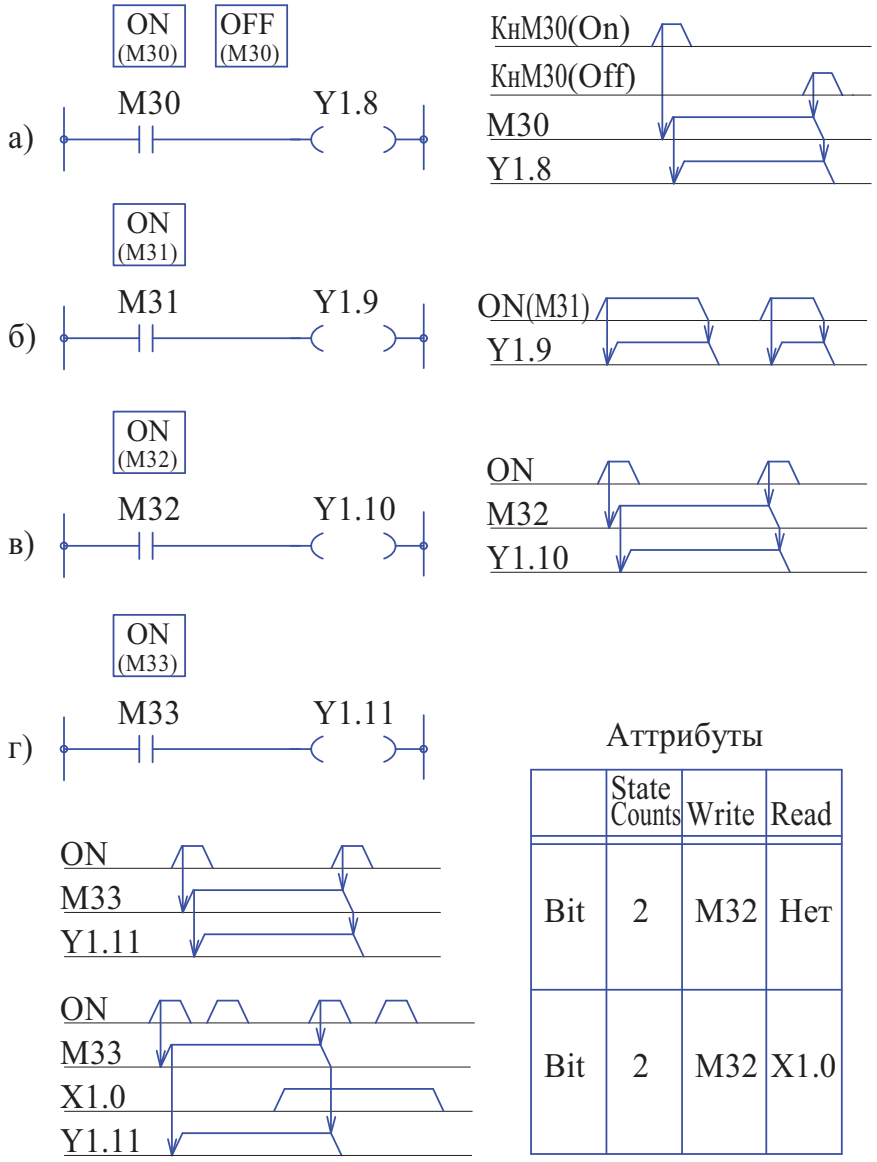


Рис. 8.10. Окно установки атрибутов кнопок перехода

Разработанные ранее органы управления можно переносить из окна в окно, используя стандартные команды Cut, Copy и Paste.

При необходимости, разработчик может рисовать на экране любые пользовательские рисунки.



Аттрибуты

	State Counts	Write	Read
Bit	2	M32	Нет
Bit	2	M32	X1.0

Рис. 8.11. РКС-алгоритмы и циклограммы работы кнопок управления:
а – SetON / OFF; *б* – Momentery; *в* – Mainteaied; *г* – Multystate

На рис. 8.11 приведены циклограммы, поясняющие работу наиболее распространенных кнопок управления:

а) кнопки типа Set ON / OFF предназначены для упрощенного формирования памяти типа RS-триггера. На панели устанавливаются две раздельные кнопки включения (On) и выключения (Off) с одинаковыми адресами (здесь M30) какого-либо дискретного выходного сигнала (Y1.8), но с разными атрибутами (параметрами). При кратковременном нажатии кнопки On выход Y1.8 включается, а при нажатии кнопки Off выключается;

б) кнопки типа Momentary, это аналог стандартной кнопки управления, контакт которой замыкается при нажатии и размыкается при отпускании. Такие кнопки наиболее распространены при проектировании, так как позволяют работать привычным образом;

в) кнопки типа Maintained предназначены для упрощенного формирования памяти типа счетного Т-триггера;

г) кнопки типа Maintained, это многофункциональные кнопки, изменяющие способ функционирования в зависимости от установленных параметров.

Отдельно отметим, что в базе данных имеется большое разнообразие кнопок управления, различающихся не только по принципу работы, но и по внешнему оформлению и цвету. Форма кнопок может быть простой прямоугольной, прямоугольной с обрамлением, круглой и т. д. Кроме того, форму кнопок, их размеры, координаты расположения на экране, цвет, адреса кнопок, надписи на кнопках, специальное назначение и многое другое можно оперативно изменять при помощи специальных окон установки атрибутов (параметров). Такие окна вызываются путем кратковременного «клика» мышкой на аппликацию кнопки.

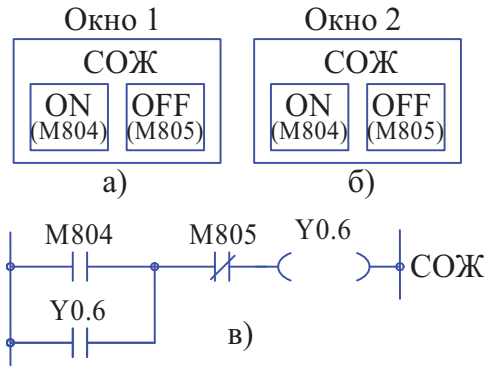


Рис. 8.12. Управление памятью из разных окон

Приведем еще очень важную информацию:

1. При дублировании органов управления одной и той же операцией в разных окнах (рис. 8.12), в программе электроавтоматики этого делать не нужно.
2. Синтаксис языка рассматриваемой панели оператора позволяет формировать ячейки признаков окна (рис. 8.13), что очень удобно для реализации различного рода блокировок (рис. 8.14).

Процедура формирования таких признаков приведена далее

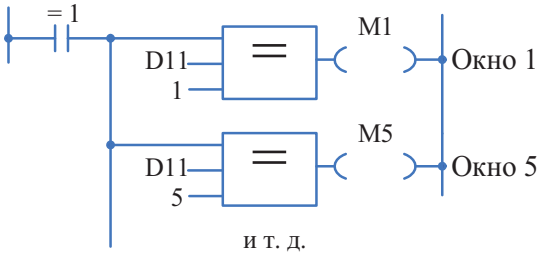


Рис. 8.13. Формирование признака окна

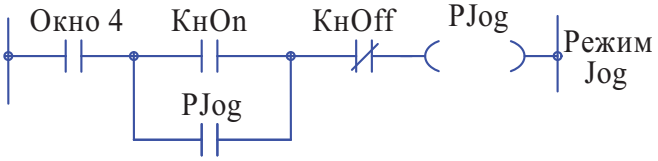


Рис. 8.14. Применение признака окна

Процедура формирования признаков окон (панелей):

1. В программе DOP Soft выполнить следующее:
   - выйти в меню конфигурации по цепи Options / Configuration;
   - активизировать окно Control Block и в открывшемся меню последовательно, слева – направо поставить «√ – галочки». Система сама поставит рабочие адреса.

Control Block	Static Block
√ Screen № → D0 √ General Contr. → D1 ……..	√ General Control → D10 √ Screen № → **D11** **……..**

Здесь **D11** – это адрес, который будет автоматически записываться номер открытой активной панели.

2. В программе электроавтоматики ISP написать программу дешифрации номера панели и выходным ячейкам присвоить рабочие адреса (рис. 8.13), например M1 – первое окно, M2 – второе и т. д.

3. Во избежание наложения адресов зарезервировать для рассмотренной процедуры адреса M0 – M19 и D0 – D19 и не использовать их в программе электроавтоматики для других целей.

4. Использовать сигналы признаки панелей при необходимости выполнения различных блокировок. Например, на рис. 8.14 показан алгоритм, разрешающий включение режима JOG только из 4-го окна. Кроме того если оператор забудет отключить данный режим, то при переходе в другое окно отключение произойдет автоматически.

Ниже приведены инструкции по записи проекта панелей из компьютера в память панели.

8.4. Инструкция по записи проекта панелей управления станком в память Панели Оператора типа DOP-110WS фирмы «Дельта»

Аппаратные и программные средства

Компьютер: hp, Windows 7 максимальная, SP1, 64-х разрядная ОС;
Панель оператора: DOP110WS;
Программное обеспечение: DOP Soft V4.00.08;
Контроллер: AS324MT-A.

Общие сведения

1. Разработка проекта сенсорных панелей управления станком для панелей оператора типа DOP-110WS осуществляется при помощи программного обеспечения типа DOP Soft V4.00.08, предварительно загружаемого в персональный компьютер.

2. Программа проекта имеет расширение «.dpa».

3. Загрузка проекта в панель может быть выполнена несколькими способами:

- через USB-порт (разъем Slave);
- через Ethernet (разъем LAN1);
- через COM-порты (разъемы COM1 и COM2/3);
- через SD Card (одноименный слот).

Предварительные действия

1. Загрузить в компьютер программное обеспечение DOP Soft V4.00.08.

2. Разработать проект панелей управления, сделать сохранение (**File → Save**).

3. Выполнить операцию компиляции (**Tools → Compile All**), проверив проект на отсутствие ошибок (Message, Error, Warning).

4. Из падающего меню **Tools → On/Off Line Simulator** или иконками панели управления активизировать Симулятор. Проверить правильность работы проекта, щелкая мышкой по кнопкам управления.

Загрузка через USB-порт

1. Выполнить аппаратное соединение между ПК и Панелью, используя стандартный покупной кабель USB (ПК) **→ Slave** (Панель).

2. В компьютере выполнить следующие действия:

2.1. в падающем меню **Options** открыть окно Environment, активизировать USB и установить драйвер «Reinstall HMI USB Driver»;

2.2. в падающем меню **Options** открыть окно **Configuration / Main / Others** и установить USB Up/Down **→ CDC** (Communication Divice Class).

3. В панели управления выполнить следующие действия:

3.1. пальцем коснуться кнопки «System Setting»;

3.2. в появившемся окне с активной кнопкой MISC установить: USB Comm. Mode **→ CDC** (коснувшись пальцем в месте установки).

4. В компьютере выполнить следующие действия:

4.1. в падающем меню **Tools** командой **Firmware Update** проверить наличие связи (выполнять по подсказкам**);**

4.2. в падающем меню **Tools** командой **Download All Data** запустить процесс загрузки проекта из компьютера в панель оператора.

5. Выключить ПК и Панель, отсоединить кабель.

6. Включить автономно Панель оператора и проверить ее работу. Возникающие при этом ошибки исчезнут **(?)** при настройке связи Панели оператора с ПЛК.

Для возврата к меню «System Setting» для внесения изменений в настройки панели необходимо выполнить следующие действия:

- коснуться пальцем и подержать некоторое время любое свободное место экрана. В левом верхнем углу панели наблюдать появление меню системных настроек;
- активизировать верхнюю иконку «System Setting» (шестеренки);
- подтвердить переход в системное меню (Да / Ок). Переход завершен.

Для нового перехода из системного меню в рабочий проект необходимо в правом верхнем углу нажать кнопку «Возврат».

Загрузка через Ethernet

1. Подготовить Ethernet-кабель (Распайка прямая).
2. Открыть DOP Soft 4.00.08 и рабочий проект панелей.
3. На компьютере выполнить следующие действия:

3.1. в падающем меню **Options** открыть окно Environment, активизировать **Ethernet**

Upload / Download

( ) USB (***) Ethernet**

( ) PC COM Port

и щелкнуть **ОК**!

3.2. в падающем меню **Options** открыть окно **Communication Setting**;

3.3. в открывшемся окне щелкнуть по **Ethernet**;

3.4. в открывшемся окне установить:

Device

Link Name 00 – Ether Linr1

Detail

Communication Parameter

Controller: **Delta AS Series PLS TCP**

HMI Station: 0

IP: ComPort **192.168.1.3** : 44818

Main

PLC Station: 1

PassWord: 12345678

Delay Time: 1000

Retry Count: 2

и щелкнуть **ОК**!

PS. Первые три цифры IP-адреса (192.168.1.) должны совпадать с IP-адресом компьютера, а последняя отличаться (произвольно).

4. Соединить ПК и ПО кабелем Ethernet (можно на начальной стадии работы).

5. Удерживая палец на свободном месте экрана панели активизировать режим настройки.

6. Коснуться пальцем иконки в верхнем левом углу панели (Шестеренки) и перейти в системное меню панели.

7. Нажать клавишу «Настройки системы».

8. Клавишей «> – дальше» переключить меню и в окне Настройки системы – Сеть установить:

LAN1

Имя панели: HMI

Mode: **Static**

IP: **192.168.1.5**
Маска: 255.255.255.0
Шлюз: 192.168.1.1
DNS: 0.0.0.0
MAC: 00:18:23:75:7F:BB
и нажать клавишу «**Обновить**»

9. Открыть на компьютере рабочий проект панели, выполнить сохранение и компиляцию.

10. Активизировать меню **Tool / Download All data**.

11. В открывшемся окне «**IP Address**» убрать «галочку» с команды **Auto Search**, установить IP-адрес панели:
Static IP 192.168.1.3
и нажать клавишу OK!

12. В окне «**IP Address**» наблюдать запись
HMI Model Type Source IPA Port
HMI DOP-110WS 192.168.1.3 12346
и нажать клавишу **OK**!

13. На экране панели наблюдать процесс записи проекта из ПК в Панель.

8.5. Панели оператора фирмы Kinco

В качестве альтернативы приведем основные стадии проектирования панели типа MT4532 TE HMI. Идеология работы с панелями разных фирм одинакова, различаются синтаксические свойства и, особенно, процедурные вопросы конфигурации, набора графики и организации связи. Нужно обращаться к сопроводительной документации, но при наличии опыта большинство вопросов довольно легко решаются «опытным» путем.

Основные технические характеристики:

- размеры 193×280×35,5 мм;
- диагональ экрана 11.1 дюйма;
- питание = 24 В, 6 Вт;
- 32-х битовый процессор RISC;
- операционная система Linux;
- загрузка через USB-кабель или Flash-накопитель;
- программное обеспечение Kinco HMIWare V2.5.

Процедура конфигурации:

- поставить на компьютер и открыть софт Kinco HMIWare V2.5;
- через падающее меню Файл (F) создать новый проект, присвоив ему имя, например, ch1 и нажать клавишу OK;

– наблюдать открытие начального меню ch1.wpj;
– с левой стороны экрана появится «Окно графических элементов»:
  Коммуникации
  HMI
  ПЛК
  Графические элементы
  Функциональные компоненты
  Базы данных
– с правой стороны экрана появятся меню «Файлы проекта» и «Структура проекта»;
– в меню HMI выбрать тип панели и перетащить иконку в наборное поле (в примере MT4532TE);
– в меню ПЛК выбрать тип контроллера и перетащить иконку в наборное поле (в примере Delta DVP);
– в меню коммуникации выбрать тип связи (в примере COM-порт) и соединить панель с контроллером (рис. 8.15);
– сохранить (файл / сохранить) и откомпилировать (инструменты / компилировать) проект.

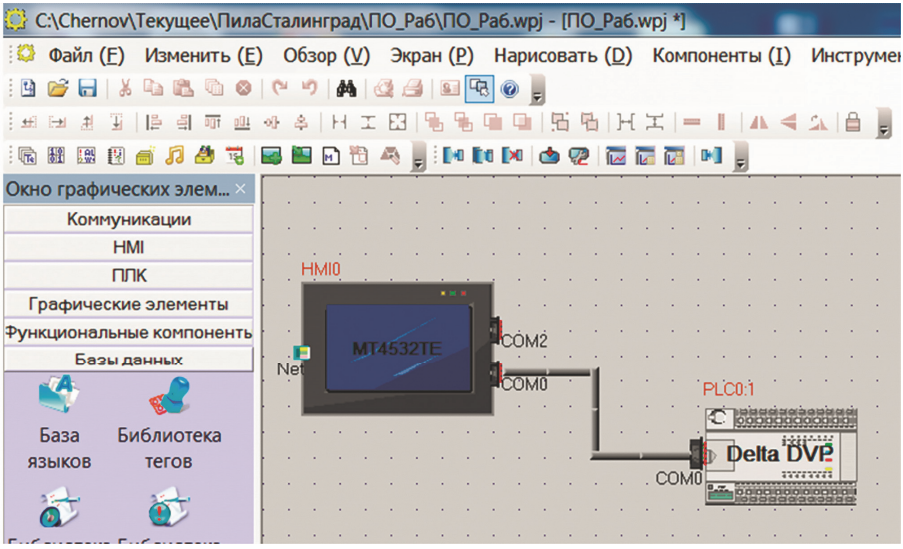


Рис. 8.15. Конфигурация проекта

Проект сохранится в C: \ Kinco \ Kinco HMIWare V2.5 \ Projects \ ch1 \ ** переписать **весь проект** в нужный каталог компьютера и в копии на на флешку.

Следующие стадии проекта:

- ознакомиться с синтаксисом языка;
- разработать первую панель, перетаскивая нужные графические элементы на экран (рис. 8.16). При этом можно использовать, как левое «Окно графических элементов», так и падающее меню «Компоненты». Для каждого элемента установить необходимые атрибуты. Сохранить проект панели;
- аналогично спроектировать следующие панели, одновременно разработав систему их переключения. Например, серые кнопки рис. 8.16 Наладка, Параметры, Карта, Лист, Статистика и Аларм предназначены для вызова соответствующих рабочих панелей, желтые кнопки В/Пульт, Наладка, Лист, Карта активизируют и одновременно, меняя цвет, индикатируют режимы работы, кнопки СОЖ и Шнек – это оперативные кнопки включения / выключения приводов;
- установить физическую связь между панелью и компьютером, загрузить проект в память панели. На этом же этапе можно установить атрибуты (параметры) самой панели;
- установить физическую связь между панелью и программируемым контроллером, обеспечив тем самым их автономную работу.

Совет: выполнив конфигурацию проекта, методично, одно за другим, открыть падающие меню Файл(F), Изменить(E), Обзор(V), Экран(P), Нарисовать(D), Компоненты(I), Инструменты(I), Опция(O), Окно(W). Справка(H) и внимательно познакомится с их содержимым. Это позволит быстро найти нужные для выполнения проекта команды.

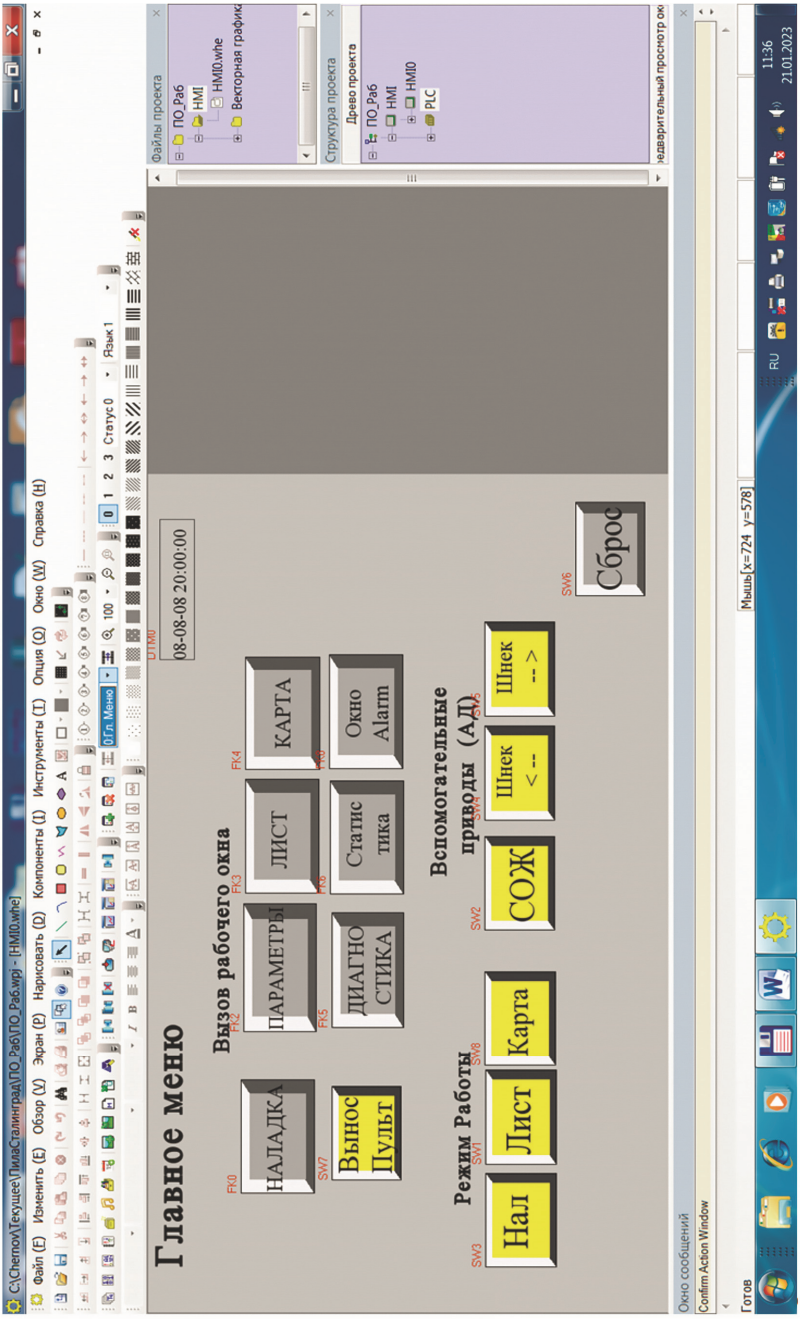


Рис. 8.16. Окно набора панели

ГЛАВА 9. ПРОГРАММИРУЕМЫЕ КОНТРОЛЛЕРЫ СИСТЕМ ЧПУ

9.1. Общие сведения

Программируемые контроллеры систем числового программного управления (СЧПУ) в большинстве случаев непосредственно встроены в систему ЧПУ. Отдельными блоками поставляются модули расширителей дискретных входов и выходов.

При проектировании электроавтоматики PLC станков с ЧПУ нужно учитывать следующее:

1. Перечень решаемых задач существенно шире и сложнее. Кроме стандартных задач локального циклового управления добавляется следующее:
   - управление электроприводами осей координат (приводы подачи), предназначенных для формообразования траектории движения инструмента (или детали) при резании;
   - управление электроприводом главного движения, предназначенного для физической обработки детали;
   - управление автоматическим поиском и сменой инструмента, а в обрабатывающих центрах и деталей;
   - решение различных специальных задач, например, коррекции инструмента, измерений, обработки M, T и S кодов, формирования ответов, задание режимов работы, программного ограничения перемещений, выхода в ноль, управления синхронными осями и т. д.

Как следствие, дополнительно необходимо разобраться с большим набором обменных сигналов и параметров, используемых при программировании управлением приводами подачи, главного движения, сменой инструмента и при решении специальных технологических задач [62, 63].

2. Широкая возможность компоновочных решений, и как следствие, большое количество электронных устройств, входящих в общий комплекс электрооборудования, которые необходимо аппаратно и программно связать между собой.

3. Возможность создания собственных оболочек (пультов) управления и применения промышленных выносных пультов.

4. Большое разнообразие протоколов связи (SSB, IO-Link, FFSB, Modbus и др.) в зависимости от типа УЧПУ.

5. Большое разнообразие номенклатуры обменных сигналов и параметров.

6. Существенное отличие синтаксиса языков программирования.

Приведем далеко не полный перечень фирм, системы ЧПУ которых широко применяются в России: ООО «Балт-Систем», «Модмаш-Софт», «Маяк», «Микрос», 4СК, «Фанук», «Сименс», «Хайденхайн», «Фагор», «Мицубиси», «Дельта» и др. Везде все по-разному, а наш отечественный универсал электрик-программист должен суметь разработать проект, написать программу и наладить станок с любой системой. И ведь мы, россияне, это делаем. Для начинающих скажем, что нужно просто знать с чего начать и в какой последовательности выполнять работу. Сделав один проект и поняв, что к чему, работа с другой системой ЧПУ (другим контроллером) будет *делом техники*. Просто нужно будет определенное время и упорство. Я это многократно испытал на себе.

Действительно, чтобы, например, управлять движением координатной оси необходимо: связать электропривод с УЧПУ, поставить его на слежение, выбрать режим работы, выбрать ось, направление и скорость движения, проверить разрешающие блокировки и дать стартовую команду. Разве это зависит от типа системы ЧПУ? Конечно нет. Значит, для каждой системы в технической документации необходимо найти соответствующие обменные сигналы и параметры, отвечающие за то или иное действие и написать блок программы в соответствии с синтаксисом языка.

Итак, с чего начать? *Первый* шаг, это изучение объекта управления, выбор типа системы, электроприводов, прочего электрооборудования, разработка органов управления и составление технического задания.

Второй шаг, изучение основных комплектующих изделий, системы ЧПУ и встроенного в нее PLC, электроприводов. Результатом этой работы, как минимум, должно быть понимание, как следует аппаратно соединить блоки между собой, как адресовать дискретные входы и выходы, чтобы нарисовать принципиальную схему. Здесь же получить начальные сведения об обменных сигналах и параметрах.

Третий шаг, разработка принципиальной электрической схемы. На данном этапе вполне допустимы неточности и ошибки. Они будут исправляться при написании программы электроавтоматики и при заключительной отладке объекта автоматизации.

Перед написанием программы нужно разобраться со способом управления электроприводом и системой его параметров. Если привод допускает автономную наладку, то эту работу желательно проделать предварительно на стенде.

Четвертый шаг, автономное включение системы ЧПУ и ее конфигурация. Выполняется на стенде.

Пятый шаг, установка на компьютер необходимого для разработки программного обеспечения и набор программы. По мере проектирования можно углублять знания по обменным сигналам и параметрам, необходимым для текущего блока программы.

Шестой шаг, ввод программы в память системы ЧПУ. Если есть возможность, то предварительно ее отладить на стенде. Это позволит избежать многих ошибок и неприятностей при реальном запуске объекта управления.

Однозначное мнение автора, такая возможность должна быть всегда. Один раз изготовить универсальный стенд с имитацией входов тумблерами и выходов двух-полярными светодиодами никакого труда не составляет. В зависимости от типа системы меняются только соединительные кабели.

9.2. Аппаратное подключение

В качестве примера приведем примеры организации связи между УЧПУ, приводами и периферийными блоками:

Рис. 9.1 – блок-схема подключения системы ЧПУ типа NC-210 для управления фрезерным станком модели ГФ2171. Это моноблочная система с возможностью управления от ЦАП 4-ми координатными осями и шпинделем. Число входов – 64, выходов – 48. Отдельный разъем для подключения штурвала.

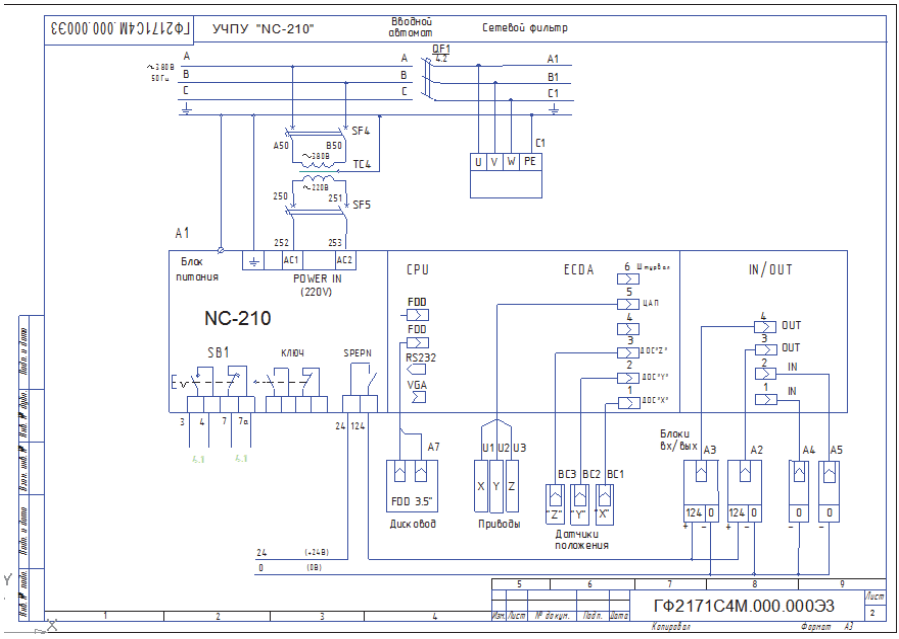


Рис. 9.1. Блок-схема подключения системы ЧПУ типа NC-210

Рис. 9.2 – блок-схема подключения системы ЧПУ типа Fanuc Series 0i Model TF для управления 2-х суппортным карусельным станком модели SC27Ф3. Это многоблочная система с возможностью управления по оптоволоконному кабелю 4-ми координатными осями и шпинделем по каналу ЦАП. Отдельный модуль связи с линейными датчиками перемещения. Выносные блоки входов и выходов с управлением по каналу IO-Link (48вх / 32вых). Отдельный стационарный программируемый пульт электроавтоматики, выносной пульт со встроенным штурвалом.

Рис. 9.3 – блок-схема подключения системы ЧПУ типа NC-310 для управления фрезерным станком модели ИР500ПМФ4. Это многоблочная система с вставляемыми наборными платами управления по каналу ЦАП приводами подачи, главного движения и дискретными входами и выходами. Отдельный стационарный программируемый пульт электроавтоматики со встроенным штурвалом.

Дальнейший материал излагаем, ориентируясь на систему ЧПУ типа NC-210 производства С.-Петербургского ООО «Балт-Систем».

Схема подключения дискретных входов и выходов рис. 9.4 показывает, насколько важно строго придерживаться технической документации.

Внимание!

Полярность подключения входных и выходных сигналов строго фиксирована. Это важно не только для правильной связи с источником питания, но и при выборе входных бесконтактных датчиков.

Подключение входных сигналов осуществляется не по порядку.

После 16-й ножки входов идет 20-я ножка, а ножки 17, 18, 19 используются для подключения питания.

Подключение выходных сигналов осуществляется так же не по порядку.

После 12-й ножки входов идет 14-я ножка, а ножка 13 используются для подключения последнего адреса.

К разъему выходов не подключается нулевой провод питания, поэтому блок не может работать автономно, только вместе с блоком входов.

Такие «особенности» есть практически во всех системах ЧПУ и об этом нужно помнить.

9.3. Система адресации

Система адресации входных, выходных и обменных сигналов использует буквенно цифровые коды. Она **одинакова** для всех типов устройств. Различие состоит только в конкретной адресации номеров разъемов физических входов / выходов и их привязки к конкретным ножкам разъемов, что необходимо уточнять по конкретной технической документации.

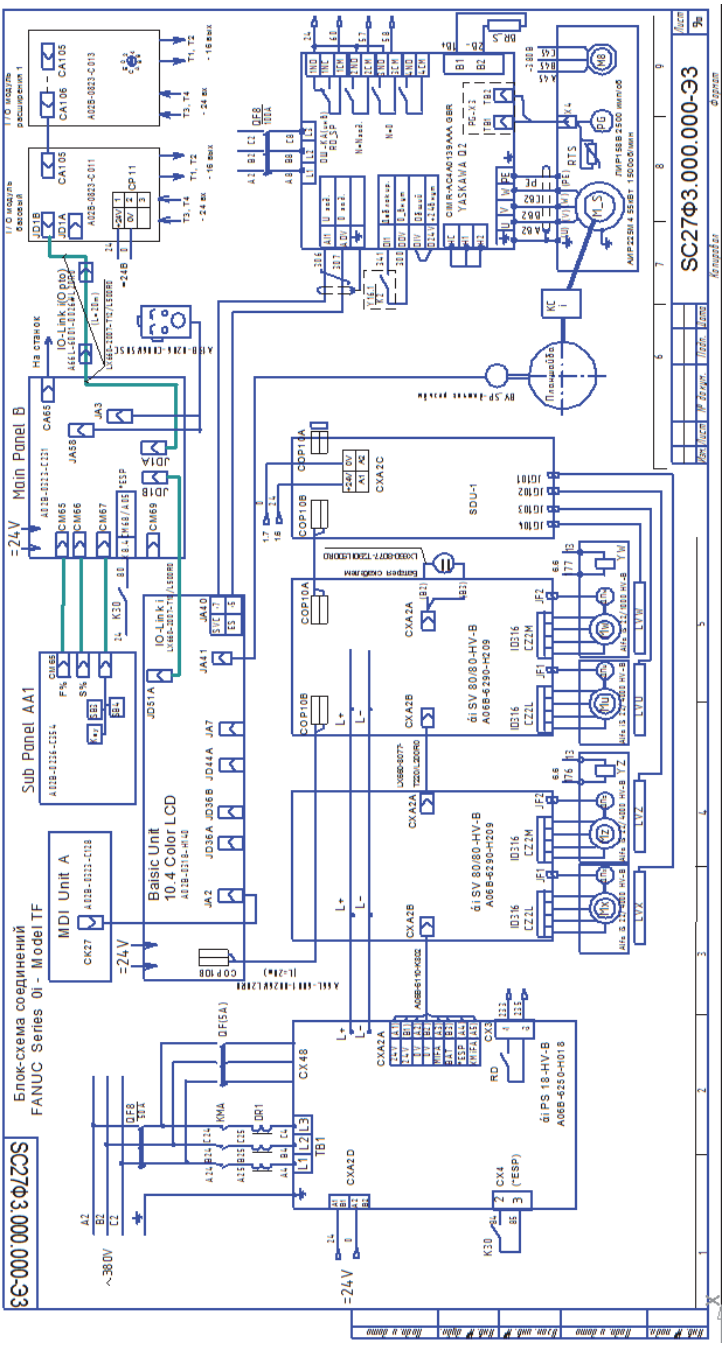


Рис. 9.2. Блок-схема подключения системы ЧПУ Фанук

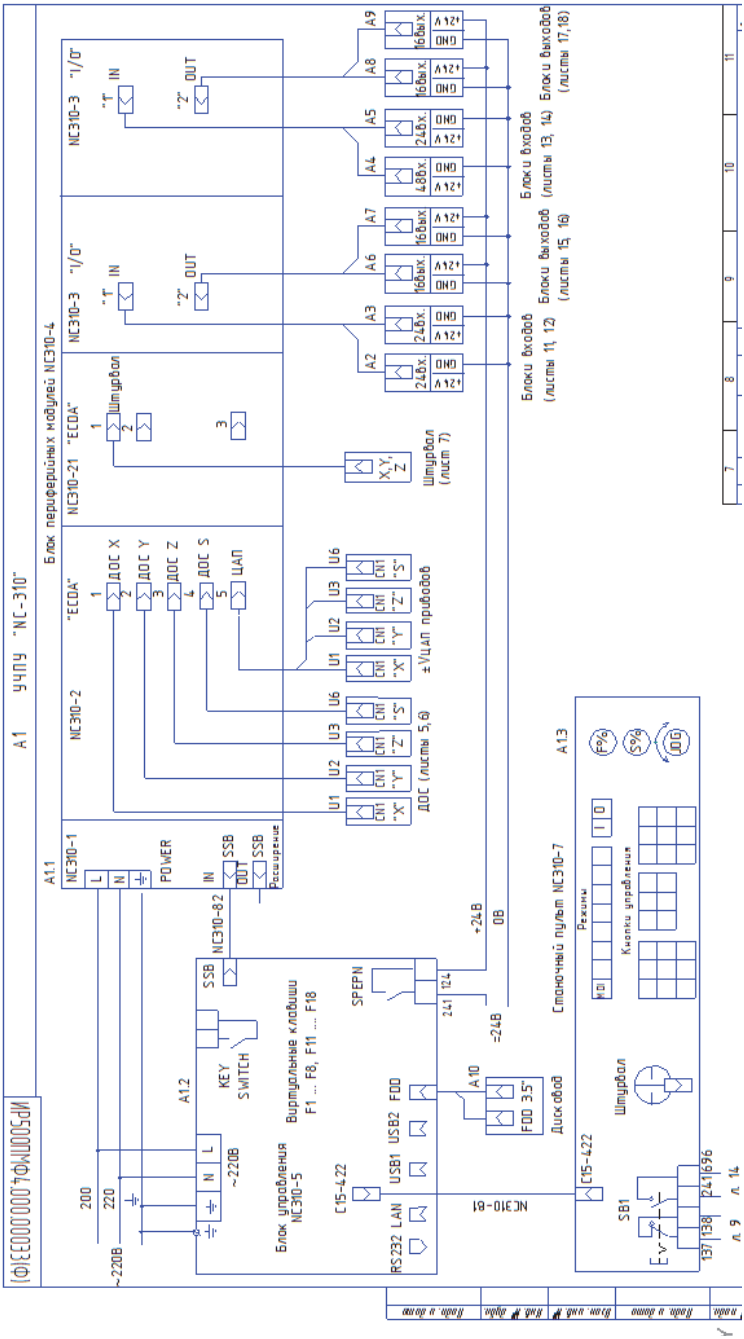


Рис. 9.3. Блок-схема подключения системы ЧПУ типа NC-310

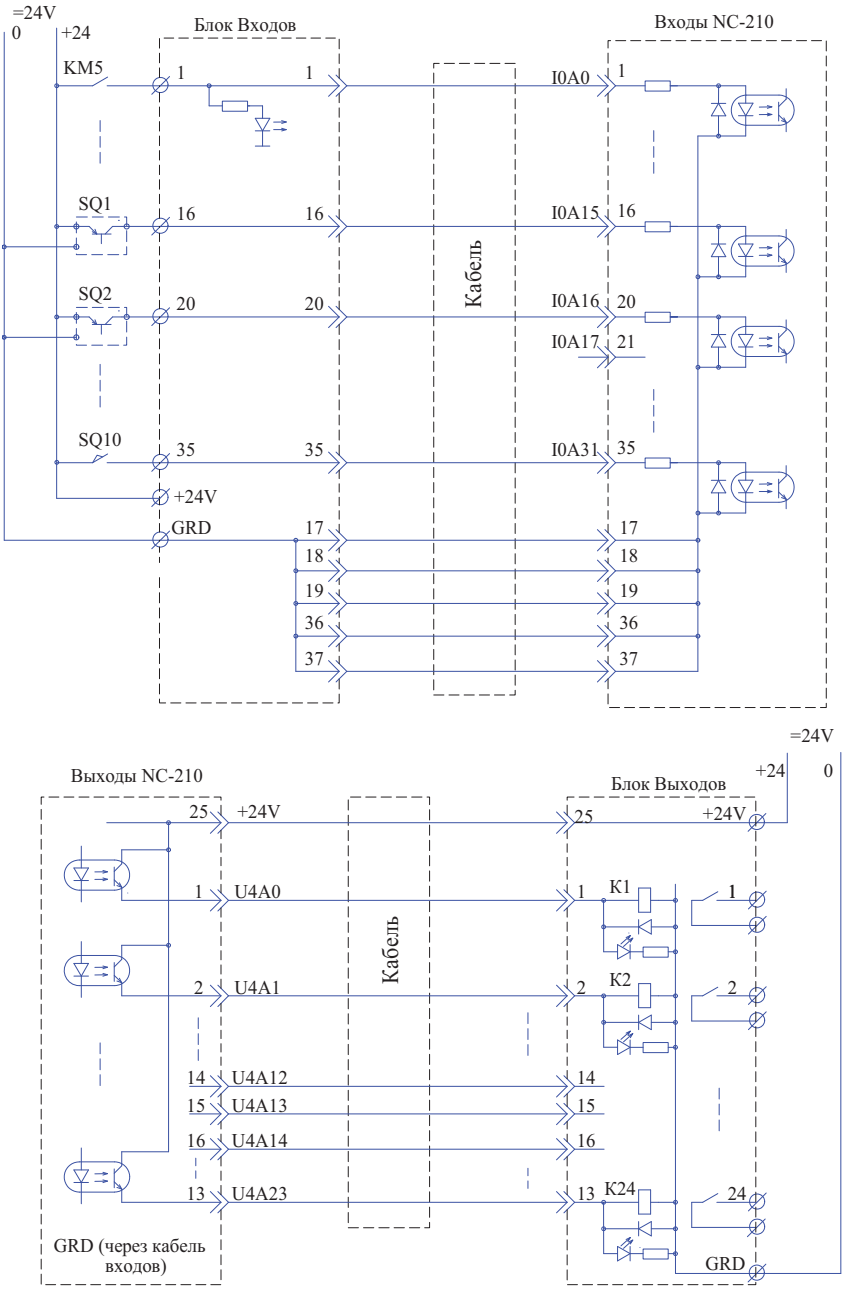


Рис. 9.4. Подключение дискретных входов и выходов

Назначение букв: I – бит входа, U – бит выхода, W – восьмиразрядное слово, K – принадлежность к оперативной памяти, N – принадлежность к 2-му массиву памяти, A – принадлежность к физическому входу или выходу.

Базовое ПМО формально (ограничение по аппаратной части) позволяет осуществлять управление пятью независимыми процессами (каналами) одновременно, для чего в пакете памяти адреса **К** выделены пять дублирующих групп слов (0К–22К, 26К–48К, 52К–74К, 78К–100К, 104К–126К) для системных фиксированных обменных сигналов.

Неиспользуемые группы системных слов можно произвольно использовать в электроавтоматике. Аппаратная часть систем NC-210 реально позволяет управлять одновременно только **двумя** процессами, поэтому для нужд электроавтоматики пользователю рекомендуется использовать адреса, *начиная* с 50К.

Свободные адреса пакета **N** для нужд электроавтоматики использовать **не следует**, так как разработчик системы постоянно добавляет в этот пакет новые системные команды и может произойти *дублирование* назначения адресов. Свободной памяти пакета **К** вполне *достаточно* для любого сложного станка.

Адресация входных сигналов:

```
I  0  A  0
!  !  !  !___ номер бита (0…31);
!  !  !_____ A – признак физического входа;
!  !        K, N – признак операнда памяти;
!  !________ номер слова (разъема): 0, 1 – входы;
!           (0…255) – память;
!_____________ признак входного сигнала.
```

Адресация выходных сигналов

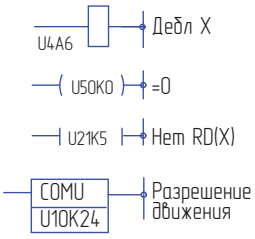
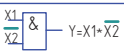
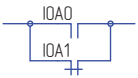
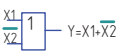
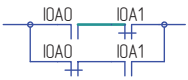
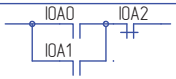
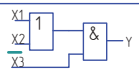
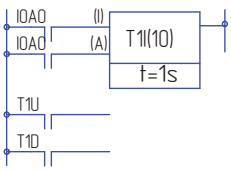
```
U  4  A  0
!  !  !  !___ номер бита (0…31);
!  !  !_____ A – признак физического входа;
!  !        K, N – признак операнда памяти;
!  !________ номер слова (разъема): 4, 5 – входы;
!           (0…255) – память;
!_____________ признак выходного сигнала.
```

Адресация 8-разрядных слов (байтов)

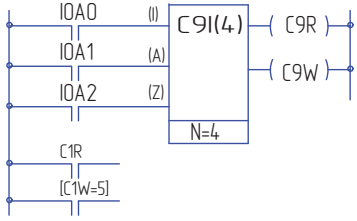
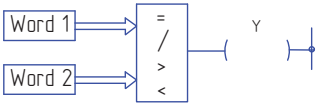
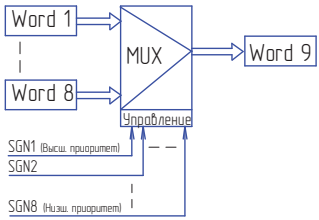
```
W  0  K  0
!  !  !  !___ номер байта (0…3);
!  !  !_____ A – признак физического входа;
!  !        K, N – признак операнда памяти;
!  !________ номер слова (разъема): 0, 1 – входы;
!           4, 5 – выходы;
!           (0…255) – память;
!_____________ признак 8-разрядного слова.
```

Таблица 9.1

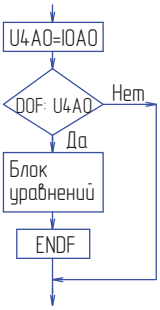
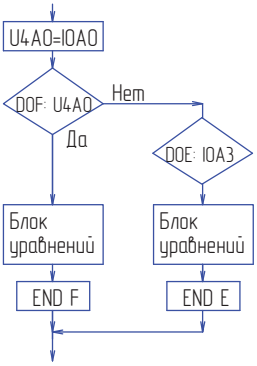
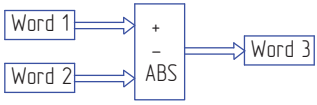
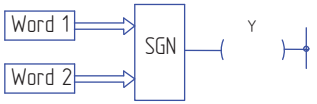
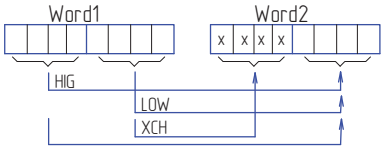
Команды языка электроавтоматики

Базовые инструкции			
Название инструкции	Условное обозначение		Запись в логическом уравнении
	РКС	Функциональный план	
Присвоение: Вых. Реле Ячейка ОЗУ Сообщение Системная команда			U4A6= U50K0= U21K5= U10K24=
Прямой операнд			U65K14
Инверсный опранд			/U59K2
Логическое умножение Y=X1*X2			I0A0*/I0A1
Логическое сложение Y=X1+X2			I0A0+/I0A1
Исключающее ИЛИ Y=(X1/=X2) Y=X1*X2+X1*X2			I0A0&I0A1
Скобки Y=(X1+X2)*X3			(I0A0+I0A1)*/I0A2
Таймер (0 … 99) T1 – таймер №1 I(10) △t=1s (I) – вход (A) – запрет U – △t на вкл. D – инв. выход	 T0–T47, T64–T99 –△=0,1s. T48–T63 –△=0,01s.		T1I(10)=I0A0 T1A=I0A1 U4A0=T1U U4A1=/T1U U4A2=T1d U4A3=/T1D

Продолжение таблицы 9.1

Функциональные инструкции		
Название инструкции	Условное обозначение	Запись в логическом уравнении
Счетчик (0 … 255) C1 –счетчик №1 I(10) N=10 (A) –слож/вычит (Z) –сброс R – выход W – тек. сост.	 Работает в **двоичном** коде	C1I(10)=I0A0 C1A=I0A1 C1Z=I0A2 U4A1=C1R U4A2=[C1W=5] C1W=24
Генератор такта (P0 – P99)		P2=I0A0 U4A1=P2 U115K5=P2
Компараторы: Y = [=] Y = /[=] Y = [>] Y = /[>] Y = [<] Y = /[<]		Y=[W0A2=W0A3] Y=/[W1A0=W70K2] Y=[W70K2>W71K3] и т.д.
Шифратор (D–>BCD)		Word2=ENC(Word1)
Дешифратор(BCD–>D)		Word2=DEC(Word1)
Преобразователи кодов		Word2=BCD(Word1) Word2=BIN(Word1)
Мультиплексор		Word9= MUX(W1,W2,...,W8), (SGN1,SGN2,...,SGN8)

Окончание таблицы 9.1

Условные переходы		
Условный переход **DOF**:		U4A0=I0A0 **DOF**: U4A0 U4A1=I0A1 – – – – – U4A2=I0A2 **ENDF** U4A20=U60K3 – – – – –
Условный переход **DOE**:		U4A0=I0A0 **DOF**: U4A0 U4A1=I0A1 – – – – – U4A2=I0A2 **ENDF** **DOE**: I0A3 U4A4=I0A4 – – – – U4A5=I0A5 **ENDE** U4A6=I0A6 – – – – –
Арифметические инструкции		
Сложение Вычитание Взятие модуля (все в **BIN** коде)		Word3=[Word1+Word2] Word3=[Word1–Word2] Word3= =[ABS(Word1+Word2)]
Выделение знака "–" (в BIN коде)		Y=SGN(Word1–Word2)
Преобразование полуслов (байтов) xxxx– обнуляются при HIG/LOW		Word2=HIG(Word1) Word2=LOW(Word1) Word2=XCH(Word1)

9.4. Синтаксис языка

Доступны два языка программирования электроавтоматики:

– язык логических уравнений;
– язык Ladder-диаграмм [94].

Команды языка логических уравнений приведены в табл. 9.1. Несмотря на небольшую численность инструкций, он позволяет решать любые задачи управления самыми сложными станками.

К *базовым* командам относятся:

– присвоение. Прямая запись выходного сигнал со знаком «равно», например, U4A5=, U10K24=;
– считывание первого прямого операнда. Прямая запись, например, U65K14, I0A16;
– логическое умножение. Обозначается знаком «*», например, I0A2*I0A3, /U65K14*I0A16;
– логическое сложение. Обозначается знаком «+», например, I0A2+I0A3, U65K14+/I0A16;
– исключающее ИЛИ. Обозначается знаком «&», например, I0A2&I0A3, U80K1&* U80K2;
– скобки (…) позволяют не задумываясь программировать любые разветвленные релейные цепочки, например,

U70K30 = I0A0*((I0A1*/I0A2+I0A3)*/I0A4) + I0A5).

Базовые команды, в принципе, позволяют решить любую задачу, но это будет сложно и нерационально. Функциональные инструкции позволяют делать это гораздо проще. Следует сказать, что автор склонен относить к базовым и команду формирования таймера, так как без него немыслим любой реальный алгоритм.

Приведем более подробное описание функциональных команд.

Таймер

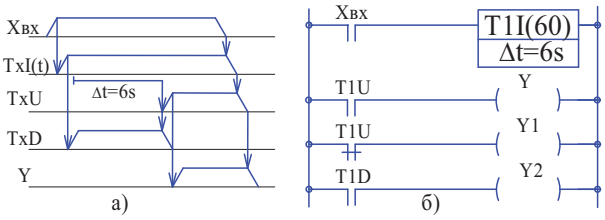


Рис. 9.5. Циклограмма работы (а) и алгоритм (б) таймера

Язык электроавтоматики систем ЧПУ серии NC предусматривает реализацию классического таймера с задержкой на включение выходного сигнала (рис. 9.5).

Синтаксис таймера:

Таймер обозначается латинской буквой T.

```
T     n     I     (Δt)    =Xвх
!     !     !       !       !  ____ Вход активизации (бит)
!     !     !       !  _____________ Уставка выдержки времени (1–255)
!     !     !_____________________ Input (Вход)
!     !_____________________________ Номер таймера (0–99)
!_____________________________________ Инструкция Таймер
```

Всего предусмотрено 99 таймеров:

T0–T47 с дискретой Δt = 0,1 сек;

T48–T 63 с дискретой Δt = 0,01 сек;

T64–T 99 с дискретой Δt = 0,1 сек.

Максимальное число дискрет равно 255, т. е.

$$\Delta t(\text{макс}) = 0{,}1 \times 255 = 25{,}5 \text{ сек.}$$

Счетчик

В синтаксисе языка электроавтоматики реализован классический двоичный реверсивный счетчик (рис. 9.6).

Синтаксис счетчика:

```
C     n     I     (m)     =Xвх
!     !     !       !       !  ____ Счетный Вход (бит)
!     !     !       !  _____________ Уставка счета (1–255)
!     !     !_____________________ Input (Вход)
!     !_____________________________ Номер счетчика (0–99)
!_____________________________________ Инструкция Счетчик
```

Счетчик обозначается латинской буквой C.

Всего предусмотрено 100 счетчиков (C0–C99).

Максимальная уставка счета 255.

Счетчик имеет три входа:

I – счетный вход;

CnA – вход реверса;

CnZ – вход сброс

и два выхода:

CnR – битовый выход, активизируемый при достижении уставки счета;
CnW – выход текущего состояния счетчика (слово).

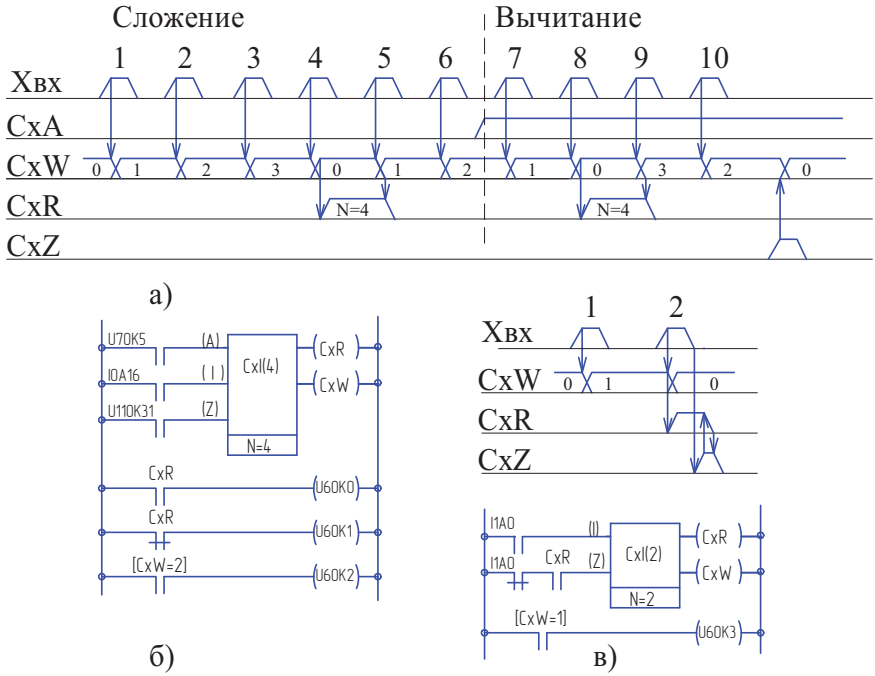


Рис. 9.6. Принцип работы инструкции «Счетчик» системы ЧПУ типа NC-210:
а – циклограмма; *б* – блок-схема алгоритма; *в* – счетчик на 2 с автоматическим сбросом

Выходные сигналы CnR и CnW могут быть проинвертированы. Отключение выходных сигналов осуществляется по завершении счета первым новым импульсом Хвх.

Синтаксис языка: C5I(100) = I0A10 – счетный вход
C5A = I0A11 – вход реверса;
C5Z = I0A12 – вход сброса;
U70K10 = C5R – считывание выходного бита;
U70K11 = [C5W = 2] – считывание текущего состояния;
W70K2 = C5W – перепись текущего состояния;
C5W = 24H – запись числа в счетчик.

Из приведенной диаграммы видны следующие особенности, которые необходимо учитывать при разработке алгоритмов на базе инструкции:

– счет начинается с нуля;
– максимальное текущее состояние на единицу меньше уставки счета;
– при достижении уставки счета счетчик сбрасывает текущее состояние в ноль;
– выходной бит счетчика сбрасывается в ноль только с приходом следующего входного импульса.

Данная инструкция, совместно с другими функциональными командами языка позволяет формировать самые разнообразные типы счетчиков.

Например, синтезировав счетчик с автоматическим сбросом при достижении уставки счета, и сделав преобразование в BCD-код, получим двоично-десятичный счетчик.

Формирователь такта

В синтаксисе языка реализован формирователь импульса (такта, скана) длительностью один вычислительный цикл по переднему фронту входного сигнала (рис. 9.7).

Формирователь обозначается латинской буквой P.

Всего предусмотрено 100 формирователей (P0–P99).

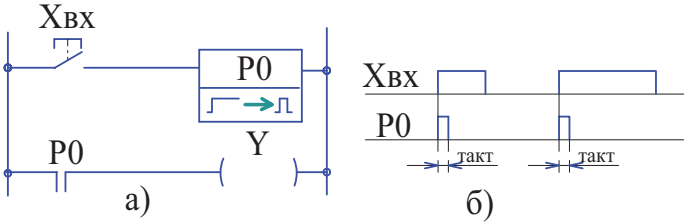


Рис. 9.7. Формирователь тактируемых импульсов по переднему фронту

При необходимости формирования тактируемых импульсов по заднему или по обоим фронтам следует использовать битовые команды.

Всего предусмотрено 100 формирователей (P0–P99).

Сравнение многоразрядных кодов в PLC следует выполнять при помощи специальных функциональных инструкций, всегда присутствующих в языке любого контроллера. Нужно не забывать, что сравнивать следует однотипные коды и, при необходимости, предварительно делать необходимые преобразования.

На рис. 9.8 приведена блок-схема инструкции сравнения системы ЧПУ NC210.

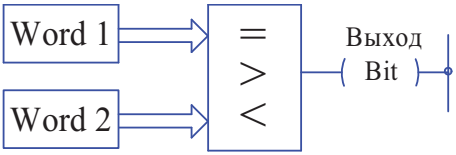


Рис. 9.8. Блок-схема инструкции сравнения УЧПУ типа NC210

Инструкция работает с 8-разрядными словами (байтами). Выполняемые операции заключаются в квадратные скобки. Синтаксис инструкции следующий:

Y = [Word 1 = Word 2] – Y = 1 если слово Word1 равно слову Word2;

Y = [Word 1 > Word 2] – Y = 1 если слово Word1 больше слова Word2;

Y = [Word 1 < Word 2] – Y = 1 если слово Word1 меньше слова Word2;

Если в уравнении использовать знак инверсии /, то можно расширить возможности сравнения. Знак / ставится перед квадратными скобками:

Y = /[Word 1 = Word 2] – Y = 1 если слово Word1 не равно слову Word2;

Y = /[Word 1 > Word 2] – Y = 1 если слово Word1 не больше слова Word2;

Y = /[Word 1 < Word 2] – Y = 1 если слово Word1 не меньше слова Word2.

Дешифратор (рис. 9.9).

Команда DEC преобразует 4-разрядный двоичный код входного слова в 8-разрядный позиционный код, являющийся неким аналогом десятичного. Система работает с 8-разрядными словами (байтами).

Синтаксис команды: Вых.Слово = DEC(Вх.Слово).

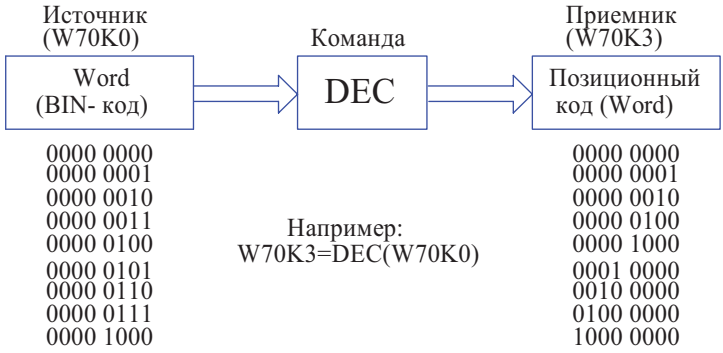


Рис. 9.9. Функциональная команда дешифратора DEC

Шифратор (рис. 9.10)

Команда ENC преобразует 8-разрядный позиционный код (некий аналог десятичного) входного слова в 4-разрядный двоичный код.

Система работает с 8-разрядными словами (байтами).
Синтаксис команды: Вых. Слово = ENC(Вх. Слово).

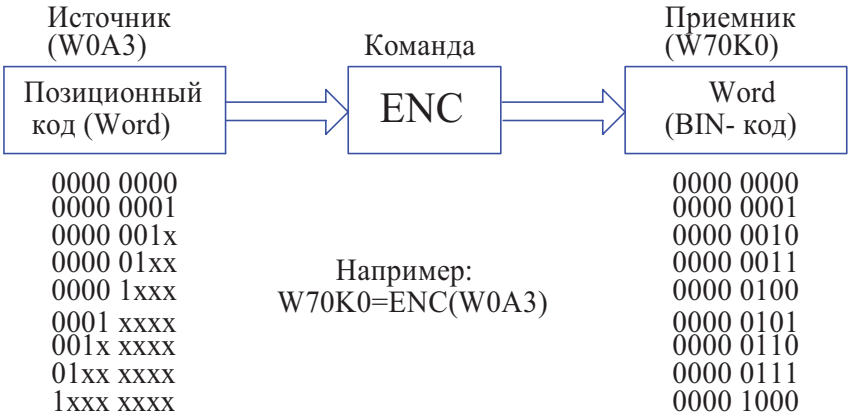


Рис. 9.10. Функциональная команда шифратора ENC

В сущности – это приоритетный шифратор выделения старшей единицы из всех единиц, присутствующих во входном коде. Знак «x» в таблице означает, что допустимо любое значение переменной 0 или 1.

Преобразователи кодов

На рис. 9.11 показаны структуры преобразователей кодов системы ЧПУ типа NC210.

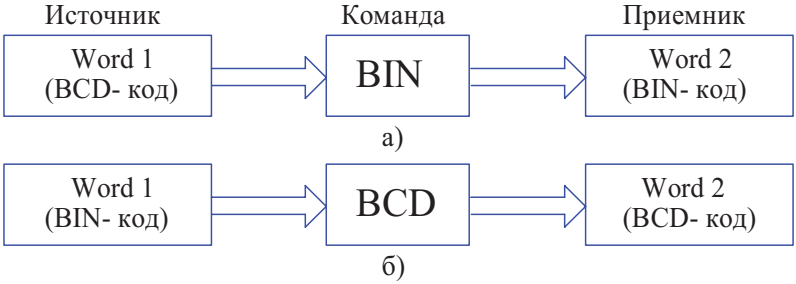


Рис. 9.11. Функциональные команды преобразователей кодов системы ЧПУ NC210: BCD → BIN (а), BIN → BCD (б)

Синтаксис языка электроавтоматики системы предусматривает работу только с 8-разрядными словами (байтами), поэтому максимальное допустимое значение преобразуемых кодов равно 99.

Синтаксис преобразователя BCD → BIN:

Word 2 = BIN(Word 1)

Синтаксис преобразователя BIN → BCD:

Word 2 = BCD(Word 1)

Здесь Word 1 – слово Источника, а Word 2 – слово Приемника.

Мультиплексор

На рис. 9.12 приведена блок-схема функциональной инструкции «Мультиплексор» системы ЧПУ типа NC210.

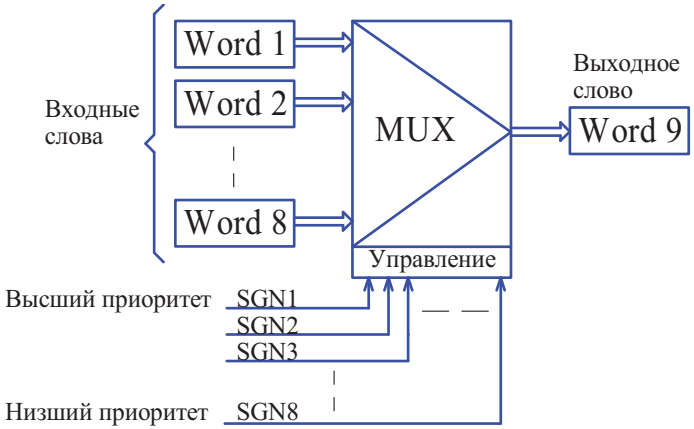


Рис. 9.12. Блок-схема инструкции мультиплексора системы СПУ типа NC210

Инструкция представляет собой 8-разрядный групповой мультиплексор, работающий с 8-разрядными словами.

Синтаксис инструкции следующий:

Word9 = MUX(Word1,Word2,Word3,…,Word8),

(SGN1,SGN2,SGN3,…,SGN8)

Сигналы управления SGNi-битовые.

Если только один из 8-ми управляющих сигналов SGNi равен единице, то в выходное слово Word9 передается содержимое, соответствующего ему входного слова Wordi, например, если SGN1 = 1, то Word9 = Word1.

Управляющий сигнал с меньшим индексом имеет *больший приоритет* над сигналом с большим индексом, поэтому если одновременно два управляющих сигнала равны единице, например, SGN1 = 1 и SGN2 = 1, то все равно Word9 = Word1.

Если SGN1 = 0, а SGN2 = 1, то Word9 = Word2.

Слова Word1…Word8 могут содержать адреса реальных входных, выходных и промежуточных слов, а также двоичных (1…255) и двоично-десятичных чисел (1H…99H) в пределах допустимых значений 8-ми разрядного слова.

При необходимости иметь больше, чем восемь передаваемых слов, инструкцию MUX можно задавать несколько раз подряд, например:

W75K2 = MUX(1H,2H,3H,…,8H),(I0A16,I0A17,I0A18,…,I0A23)

W75K2 = MUX(9H,10H,11H,12H),(I0A24,I0A25,I0A26,I0A27)

Команда MUX является универсальным средством для решения разнообразных задач электроавтоматики. С ее помощью можно выполнять преобразование кодов, делать пересылки, формировать таблицы диагностики и др.

Пример преобразования 12-тиразрядного унитарного кода (I0A16…I0A27) в двоично-десятичный (W70K2):

W70K2=MUX(1H,2H,3H,4H,5H,6H,7H,8H),(I0A16,I0A17,I0A18,I0A19,
I0A20,I0A21,I0A22,I0A23)

W70K2=MUX(9H,10H,11H,12H),(I0A24,I0A25,I0A26,I0A27)

Здесь «H» – признак двоично-десятичного кода.

Используя команду BIN можно преобразовать двоично-десятичный код в двоичный, например, W70K3 = BIN(W70K2).

Условные переходы

Предусмотрены две команды условного перехода:

DOF / ENDF и DOE / DOFE.

Принцип работы условного перехода следующий:

– если условие не выполняется Px = 0, то осуществляется переход к команде ENDF и далее по программе;
– если условие выполняется Px = 1, то выполняется часть программы, расположенная между DOF и ENDF, далее по программе.

Пример 1. Управление охлаждением от одной кнопки (рис. 9.13). Обязательным условием является тактирование кнопки. При каждом нажатии кнопки КнОхл осуществляется «заход» на один скан внутрь команды DOF / ENDF и инвертирование выхода Охл.

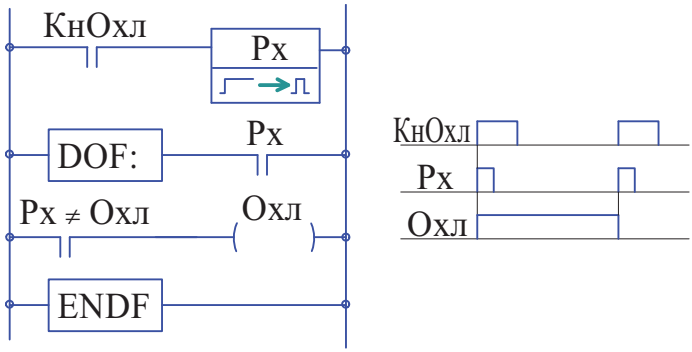


Рис. 9.13. Управление охлаждением от одной кнопки с использованием команды условного перехода

Пример 2. Тактируемый сдвиговый регистр

Тактируемые сдвиговые регистры могут применяться, например, при построении командоаппаратов выбора инструментов или при смене назначения кнопок управления для прямого управления различными механизмами. Идеология сдвига приведена на рис. 9.14.

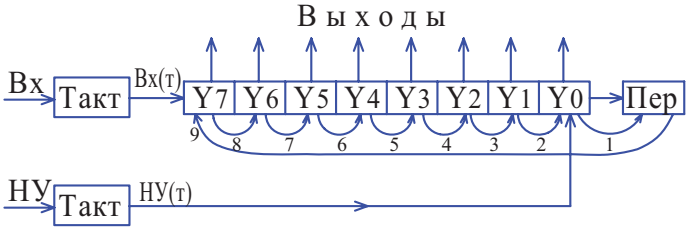


Рис. 9.14. Идеология сдвига

Предварительно в регистр записывается начальная информация.

В качестве выходов могут использоваться любые битовые ячейки, произвольно расположенные в области памяти контроллера, т. е. это регистры, которые не используют классические слова с определенным начальным адресом. Естественно, что для облегчения чтения программы, адреса следует назначать в упорядоченной последовательности.

Число разрядов таких регистров неограниченно.

Главным условием построения приведенных здесь регистров является правильная последовательность переписи информации (рис. 9.15).

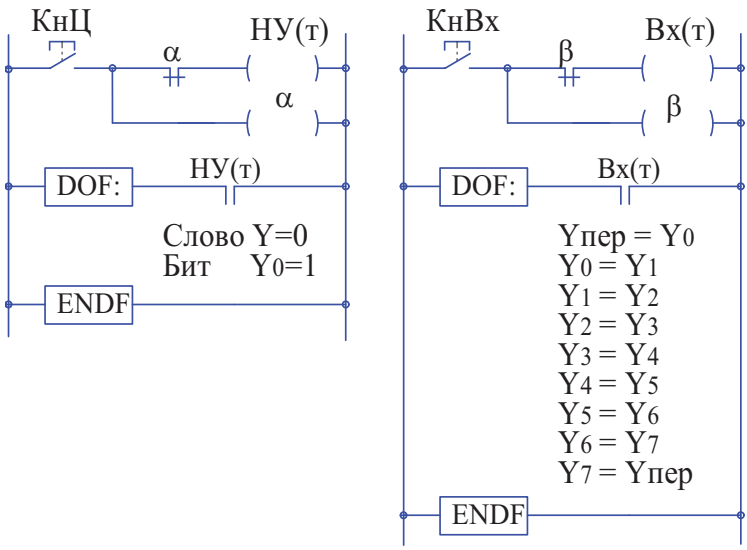


Рис. 9.15. Регистр с использованием команды условного перехода

Арифметические команды

Предусмотрены следующие арифметические команды:

Сложение. Синтаксис команды Word3 = [Word1 + Word2];

Вычитание. Синтаксис команды Word3 = [Word1 – Word2];

Взятие модуля. Синтаксис команды Word3 = [ABS(Word1 + Word2)];

Определение знака операции. Синтаксис команды Y = SGN(Word1 + Word2).

Обращаем внимание на употребление квадратных и круглых скобок.

Все арифметические операции выполняются в *двоичном* коде.

В качестве примера приведем решение задачи определения кратчайшего направления вращения инструментального магазина станка с ЧПУ.

Определение кратчайшего направления вращения

Существует много способов определения кратчайшего направления вращения инструментальных магазинов многооперационных станков с ЧПУ, основанных на разных принципах построения алгоритмов [17]. В связи с тем, что во всех современных контроллерах имеются арифметические инструкции самым правильным решением следует считать алгоритмы, построенные по следующей универсальной формуле:

$$По = (Т \geq КА) \cdot [(Т - КА) \leq N/2] + (Т < КА) \cdot [(КА - Т) > N/2]$$

$$Против = \overline{По}$$

На рис. 9.16 приведен алгоритм, построенный на базе данных уравнения для систем ЧПУ серии NC фирмы «Балт-Систем».

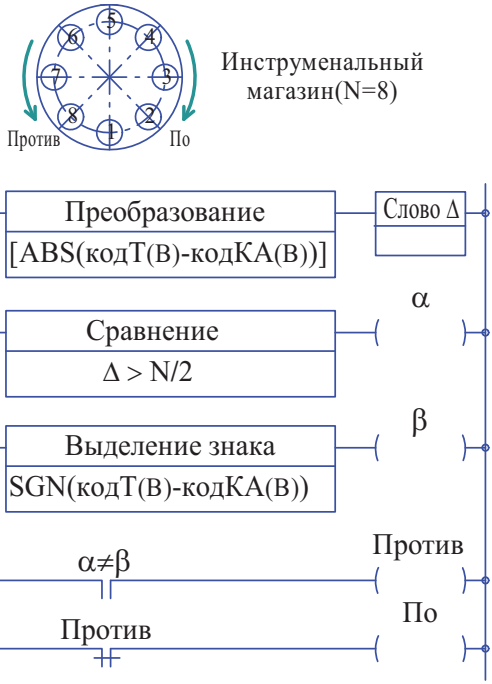


Рис. 9.16. Алгоритм определения кратчайшего направления вращения

Для решения задачи сделаны следующие преобразования, адаптирующие исходное уравнение в языковую среду контроллера:

$$\Delta = (T - KA) = (KA - T) = |T - KA| = ABS(T - KA)$$

$$\beta = (\Delta \le \frac{N}{2}) \text{ и } \overline{\beta} = (\Delta \le \frac{N}{2})$$

$$\alpha = (T \ge KA) = SGN(T - KA) \text{ и } \overline{\alpha} = (T < KA)$$

$$\text{По} = \alpha \cdot \overline{\beta} + \overline{\alpha} \cdot \beta = (\alpha \ne \beta) = (\alpha \& \beta),$$

Здесь:

ABS – функциональная инструкция выделения модуля;

SGN – функциональная инструкция выделения знака;

& – инструкция «Исключающее ИЛИ»;

α, β, Δ – промежуточные переменные для упрощения записи.

Все вычисления производятся в *двоичном коде*.

Рабочая PLC-программа для магазина с N = 30 запишется следующим образом:

```
; Определение Направления вращения
W74K1=[ABS(W71K1-W71K2)]
U74K16=[W74K1>15]
U74K17=SGN(W71K1-W71K2)
U74K18=(U74K16&U74K17)
U74K19=/U74K18
```

Здесь:

W71K1 – двоичный код Т;

W71K2 – двоичный код КА;

W74K1 – слово выделения модуля Δ;

U74K16 – промежуточный операнд β;

U74K17 – промежуточный операнд α;

U74K18 – бит направления «По часовой»;

U74K19 – бит направления «Против часовой».

Преобразование полуслов (тетрад)

Предусмотрены следующие команды:

Перемещение старшей тетрады. Синтаксис команды Word2 = = HIG(Word1). Старшая тетрада слова Word1 переписывается в младшую тетраду слова Word2, при этом старшая тетрада слова приемника Word2 обнуляется.

Перемещение младшей тетрады. Синтаксис команды Word2 = = LOW(Word1). Младшая тетрада слова Word1 переписывается в младшую тетраду слова Word2, при этом старшая тетрада слова приемника Word2 обнуляется.

Смена тетрад. Синтаксис команды Word2 = XCH(Word1). Старшая тетрада слова Word1 переписывается в младшую тетраду слова Word2, а младшая в старшую.

Практическое применение данных команд см. в [62, 63].

Приведем еще несколько полезных алгоритмов работы со словами.

Формирование такта при изменении кода (рис. 9.17).

Обмен информацией между словами (рис. 9.18).

Содержимое слов А и В меняется местами.

Ограничение содержимого слова (рис. 9.19).

Содержимое слова А ограничивается на уровне содержимого слова В.

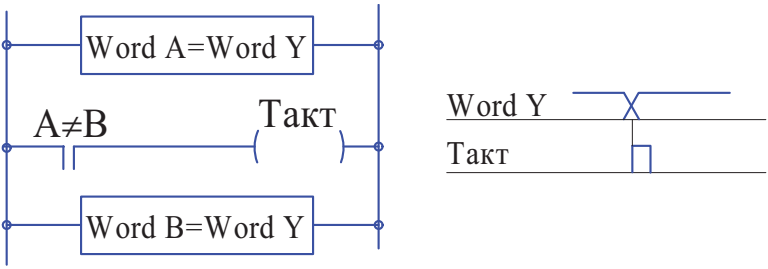


Рис. 9.17. Формирование такта при изменении кода

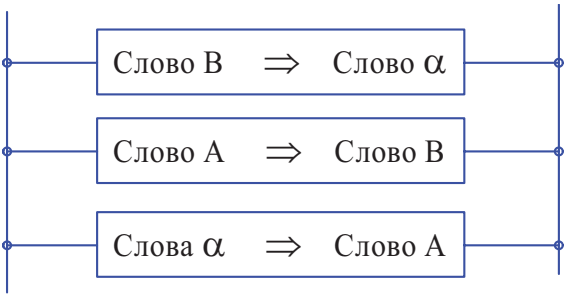


Рис. 9.18 Обмен информацией между словами

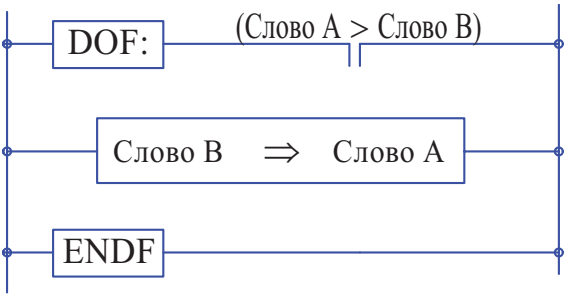
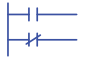
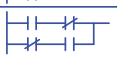
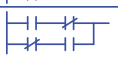
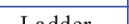
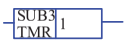
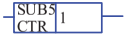


Рис. 9.19. Ограничение содержимого слова на заданном уровне

В таблице 9.2 приведена система адресации операндов встроенных контроллеров для различных систем ЧПУ. Синтаксис написания адресов кардинально отличается, а решаемые задачи те же.

Таблица 9.2

Сводная таблица адресации различных УЧПУ

Системы ЧПУ		Балт-Систем	Маяк	Микрос	FMS	Фанук	Сименс	Хайденхайн
Адресация	Входы	I0A0 ... I0A31 I1A0 ... I1A31	Ax.x бит 0-7 байт17-99	Xx.x бит 0-7 байт 0-9	Ix.x бит 1-8 байт 1-64	Xx.x бит 0-7 байт0-127	Ix.x бит 0-7 байт 0-...	Ix бит 0-23, 128-143
	Выходы	U4A0...U4A31 U5A0...U5A31	Ax.x бит 0-7 байт 1-16	Yx.x бит 0-7 байт 1-6	Ux.x бит 1-8 байт 1-64	Yx.x бит 0-7 байт0-127	Qx.x бит 0-7 байт 0-...	Ox бит 0-23,
	ОЗУ	U53K0...U53K31 U54K0...U54K31	D115-D299	Lx.x бит 0-7 байт 0-15 BLx WLx	Mx.x бит 1-8 байт 1-64	Rx.x бит 0-7 байт ...	Mx.x бит 0-7 байт 0-...	Mx бит 0-999
Базовые инструкции	Считывание	I0A0 /I1A0	A17.0 !A17.0	X0.0 (X0.0^1)	I1.1 /I1.1			L (Load) LN
	Инверсия	/	!	~	/ или -			N
	Лог. умнож.	I0A0*I0A6	A17.0&&A17.1	X0.0*X0.1	I1.1*I1.2			A или AN
	Лог. слож.	I0A0+I0A6	A17.0IIA17.1	X0.0IX0.1	I1.1+I1.2			O или ON
	Исключ. ИЛИ	I0A0&I0A6	A17.0##A17.1	X0.0^X0.1	Нет			XO или XON
	Присвоение	U4A0=	A1.0=	L8.0=	M50.1=			=M1005
	Скобки	(,), [,]	(,)	(,), [,]	(,), [,]	Ladder	Ladder	[,]
	Таймер	T5I(255)= T5U, T5D	my TM1= TM1::next	PAUSE DURING	Txx=△t TSxx, /TSxx		T0 ... T127 TON, TOF	T0 ... T47
Счетчик		C5I(255)= C5Z, C5W	my CTR1= CTR1::next	Средствами языка СИ	Cxx=N Cxx.I, Cxx.D		C0 ... C63 CTU, CTD	[,]
Комментарий		;	/	// или /*...*/	,	Ladder	Ladder	;

9.5. Обменные сигналы

Кроме решения стандартных задач цикловой автоматики контроллеры систем ЧПУ предназначены для решения специальных задач, связанных с управлением координатными осями, главным приводом, сменой инструмента и т. д. Для этой цели предусматриваются фиксированные обменные сигналы и параметры. Подробно данные вопросы изложены в [62, 63]. Здесь, для понимания сути проблемы, приведем лишь несколько примеров.

Программирование из электроавтоматики режимов работы устройства ЧПУ

На системе ЧПУ уже установлены органы управления, позволяющие автономно управлять станком:

1. Переключатель режимов работы.
2. Кнопки «Пуск» (I8K7) и «Стоп».
3. Клавиши (курсоры) выбора координат в ручном режиме.
4. Переключатель выбора скорости и направления JOG.
5. Корректор величины подачи F%.
6. Корректор скорости вращения шпинделя S%.

Однако, в ряде случаев, гораздо удобнее для оператора, если бы нужный режим включался автоматически. Для этой цели предусмотрены обменные сигналы, сведенные в табл. 9.3.

Таблица 9.3

Адреса режимов работы пульта управления

Режим		**Адрес**		
Условное обозначение	**Назначение**	**Активизация**	**Ответ о включении**	**Положение переключателя**
MDI	Преднабор	U15K8	I8K24	I1N24
AUTO	Автомат	U15K9	I8K25	I1N25
STEP	Кадр	U15K10	I8K26	I1N26
MANU	Наладка (безразмерное перемещение)	U15K11	I8K27	I1N27
MANJ	Наладка (фиксированное перемещение)	U15K12	I8K28	I1N28
PROF	Возврат на профиль	U15K13	I8K29	I1N29
HOME	Выход в ноль	U15K14	I8K30	I1N30
RESET	Сброс	U15K15	I8K31	I1N31

Адреса режимов работы можно использовать, например, для:

– формирования глобальных режимов «Ручной» и «Автоматический», которые используются для блокировок в различных подпрограммах управления станком;
– прямого включения нужного режима, если станок не работает по программе, например, MANU (Наладка) при нажатии любой кнопки движения оси. При этом программируется также запрет работы аппаратного переключателя режимов, установленного на системе.

Для запрета органов управления системы ЧПУ предусмотрено слово W15K0, осуществляющее необходимые запреты в двоичном коде:

W15K0 = 1 Запрет кнопки ПУСК;
= 2 Запрет кнопки СТОП;
= 4 Запрет переключателя режимов работы;
= 8 Запрет корректора JOG;
= 16 Запрет корректора F%;
= 32 Запрет корректора S%;
= 64 Запрет выбора оси;
= 128 Запрет выбора JOG.

Постановка приводов подачи на слежение

Вариант алгоритма постановки приводов на слежение представлен на рис. 9.20. Он обеспечивает следующую последовательность работы:

1. Формирование сигнала «Разрешение» при условии:
   – готовности УЧПУ к обслуживанию осей;
   – включенного станка и силового питания приводов;
   – выключенного состояния тепловой защиты;
   – готовности электроприводов к работе;
   – наличия прочих специфичных условий;
   – отсутствия сигнала аварии от диагностики станка.

2. Формирование импульса включения с задержкой 2–8 секунд, в зависимости от типа привода, после включения станка. Сигнал предназначен для ожидания появления готовности всех электроприводов и обеспечения нулевой защиты при срабатывании защит. Повторная постановка на слежение возможна только при выключении и повторном включении станка.

3. Включение на самопитание сигналов деблокировки горизонтальных осей (здесь X и Y) и сигнала отжима «падающей» оси Z. В формировании данных сигналов на рис. 9.20 могут быть включены конечные выключатели ограничения перемещения. Такое решение возможно, если запас хода выше паспортного мал и не позволяет установить раздельные ограничительные и аварийные конечные выключатели, и они совмещаются.

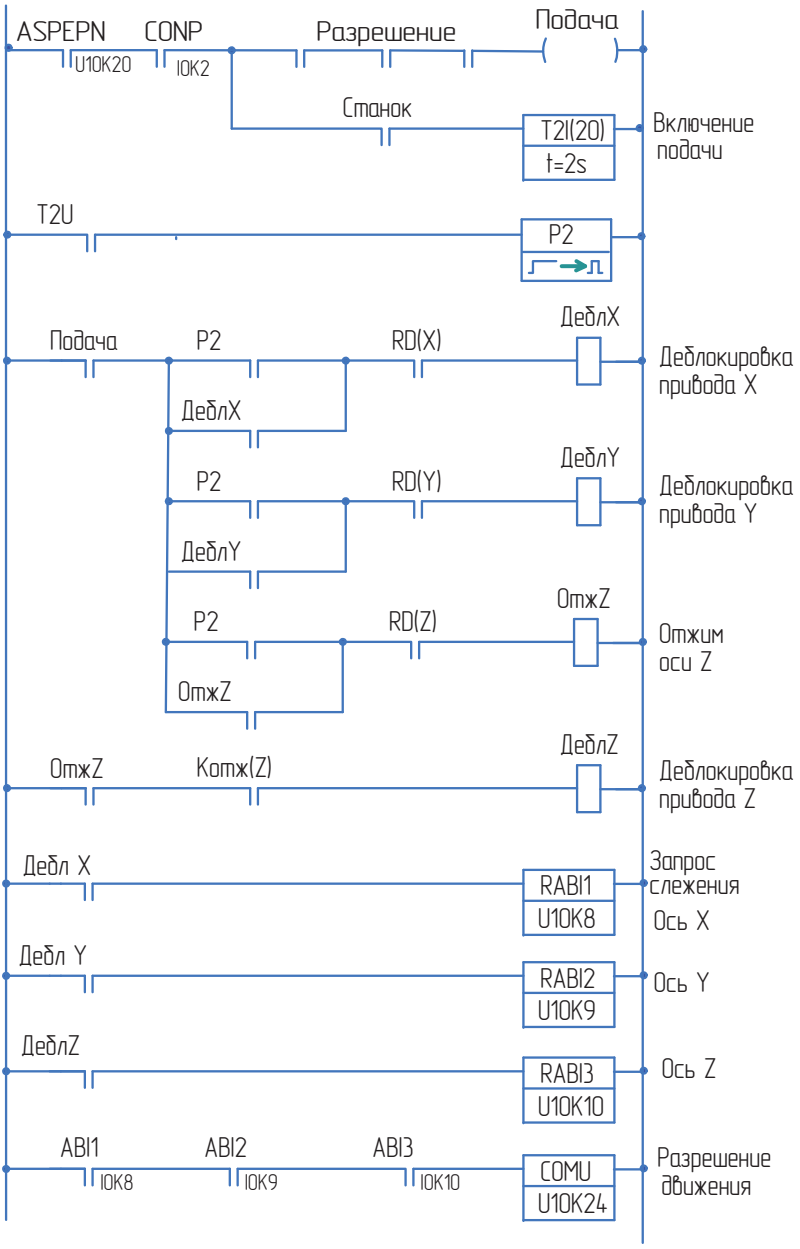


Рис. 9.20. РКС-алгоритм постановки приводов на слежение

4. Включение деблокировки оси Z после срабатывания датчика отжима.

Примечание. В некоторых конструкциях станков, чтобы исключить падение оси, требуется обратная последовательность выдачи сигналов сначала деблокировки, а затем отключения тормоза.

Если ось падает при выключении станка, следует задержать отключение тормоза.

5. Формирование сигналов запроса постановки приводов на слежение в устройство ЧПУ:

RABI1 (U10K8) – первая ось;
RABI2 (U10K9) – вторая ось;
RABI3 (U10K10) – третья ось.

6. Получение ответа от УЧПУ о постановке приводов на слежение:

ABI1 (I0K8) – первая ось;
ABI2 (I0K9) – вторая ось;
ABI3 (I0K10) – третья ось.

7. Формирование сигнала разрешения движения **COMU (U10K24)**.

Сигнал COMU разрешает движение координатных осей во всех режимах работы. Кроме того необходимо установить параметры, связанные с движением осей координат, в данном случае сформировать файлы характеризации осей AXCONF, технологии PGCONF и логики IOCONF.

9.6. Установка параметров

В системах ЧПУ производства ООО «Балт-Систем» установка параметров осуществляется при помощи файловой структуры и называется характеризацией.

На рис. 9.21 приведен принцип характеризации осевых электроприводов.

Здесь:

NAS = X Заголовок оси X;

TPA = 1, 1 – признак координатной оси;

NTC = 1, 1 – номер канала ДОС, номер канала ЦАП.

Для каждой координатной оси необходимо дополнительно конфигурировать следующие инструкции файла характеризации осей **AXCFIL**:

RAP, GAS, PAS, MCZ, SRV, MAN, GN0, LOP, MFC.

Прочие инструкции указываются при специальных требованиях к электроприводу.

Для вывода информации на экран в файле **PGCFIL** следует задать инструкцию MAS = XYZ с перечнем используемых осей.

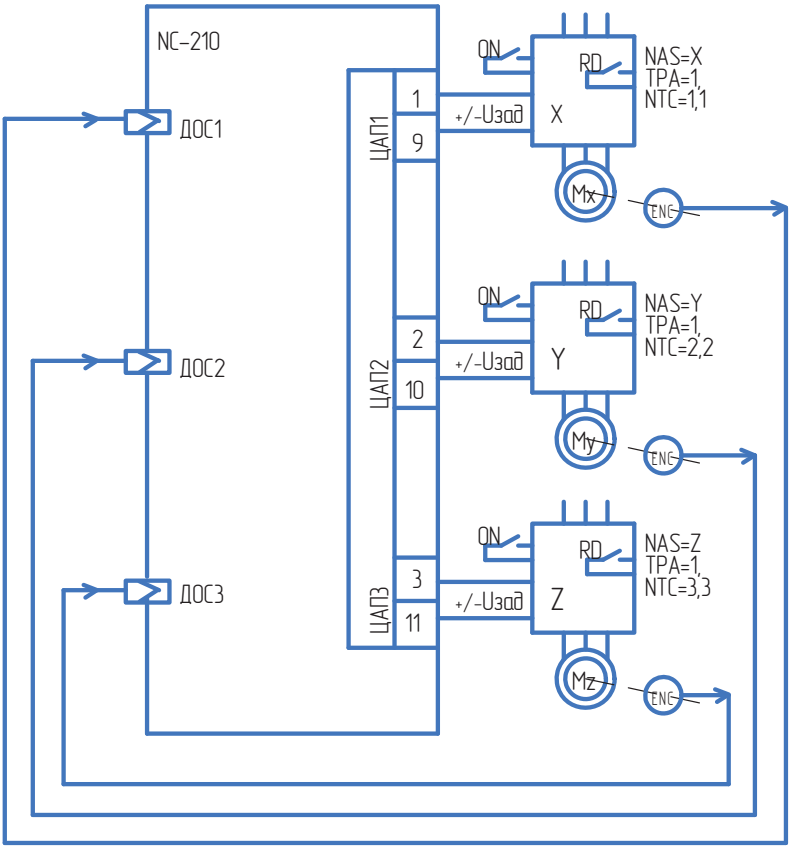


Рис. 9.21. Характеризация электроприводов

Пример файла характеризации осей AXCFIL:

```
*1
NBP = 1,
TIM = 2,0,0,0,0
PRO = 1
IN1 = 1,XYZ,S,2,16
CAS = 1,XYZS,2
```

```
*2
PRO = 1
; координатная ось X
NAS = X                 Имя оси
TPA = 1,                Тип оси
NTC = 1,1               1-й ДОС, 1-й ЦАП
RAP = 3000,100          Быстрый ход
GAS = 0.0,0             Компенсация люфта
PAS = 4000,1            Установка дискретности
POS = 0.05,5            Допуск позиционирования
SRV = 0.8,5,12          Сервоошибки
ZNO = 1.25,,,,          Смещение нуля
MAN = 3000,100          Скорость ручного перемещения
GM0 = 3000,7.5,16       Макс. скорость, Uцап, добротность
MCZ = I0A18,1,1000      Выход в ноль
MFC = I0A12,I0A13       Ограничение перемещения
LOP = 800.1,−800.1      Программные конечники
FBF = ,,U250K0,1        Ошибки датчика

; координатная ось Y
NAS=Y
TPA = 1,
NTC = 2,2
RAP = 3000,150
GAS = 0.0,0
PAS = 4000,1
POS = 0.05,5
SRV = 0.8,5,12
MAN = 3000,150
GM0 = 3000,7.5,18
MCZ = I0A19,1,1000
MFC = I0A14,I0A15
LOP = 350.1,-350.1
ZNO = −0.7,,,,
FBF = ,,U250K0,1
; координатная ось Z
NAS = Z
TPA = 1,
NTC = 3,3
RAP = 2400,120
GAS = 0,0
PAS = 4000,1
POS = 0.02,5
SRV = 0.8,5,12
```

```
MAN = 2400,120
GM0 = 2400,6.3,16
MCZ = I0A20,1,500
MFC = I0A16,I0A17
LOP = 310.1,−310.1
ZNO = 0.3,,,,
FBF = ,,U250K0,1
; шпиндель S
NAS = S
TPA = 10,
NTC = 0,4
GAS = 0,0
POS = ,
GM1 = 315,7.5,15
GM2 = 630,7.5,0
GM3 = 1250,7.5,0
GM4 = 2000,7.5,0
TSM = 5,17
; POM = 0.1,100
RAP = ,20
; ASM = X
*3
```

Примечания.

1. Программная смена полярности выходного напряжения ЦАП осуществляется *сменой знака напряжения* Uцап инструкции:

GM0 = nmax, −Uцап, Kv

2. Программная фазировка датчика по положению ДОС осуществляется *сменой знака механического шага* инструкции:

PAS = эл.шаг, – мех. шаг

Станок готов к работе, можно управлять электроприводами подачи от пульта системы ЧПУ. Для управления от станочного пульта необходимо написать специальную программу электроавтоматики, используя следующие обменные сигналы:

Выбор перемещаемой оси. Осуществляется при помощи сигналов слова **W15K2**:

U15K16 – 1-я ось, обычно X;
U15K17 – 2-я ось, обычно Y;
U15K18 – 3-я ось, обычно Z;
U15K19 – 4-я ось;
U15K20 – 5-я ось;

U15K21 – 6-я ось;
U15K22 – 7-я ось;
U15K23 – 8-я ось.

Следует иметь в виду, что в системах ЧПУ рассматриваемой серии в ручном режиме одновременно можно перемещать только одну ось.

Выбор направления перемещения. По умолчанию задается положительное направление перемещения, для задания отрицательного направления перемещения следует активизировать последний бит слова **W15K3**, т. е. установить **U15K31 = 1**.

Активизация направления перемещения осуществляется кнопками:

Кн (+X, +Y, +Z) – направление плюс;

Кн (−X, −Y, −Z) – направление минус, адреса которых присваивает разработчик программы электроавтоматики.

При помощи информации слова W9K1 (сигналы BILA1, BILA2…BILA8) можно в реальном времени получить информацию о направлении движения координат по осям, например:

I9K1 = 1 – ось X движется в отрицательном направлении,

I9K1 = 0 – ось X движется в положительном направлении.

Выбор скорости перемещения. Осуществляется при помощи сигналов слова **W15K3** в процентах от скорости быстрого хода, определенной в файле характеризации осей AXCFIL (секция *2, инструкция RAP):

U15K24 – 1 % от скорости Б/Х;
U15K25 – 2 % от скорости Б/Х;
U15K26 – 4 % от скорости Б/Х;
U15K27 – 8 % от скорости Б/Х;
U15K28 – 16 % от скорости Б/Х;
U15K29 – 32 % от скорости Б/Х;
U15K30 – 64 % от скорости Б/Х.

Сигналы **пакета N** дают доступ к положениям задатчика скорости (I1N0…I1N6) и направления (I1N7) **JOG**, а также к положениям корректора подачи **F%** (I1N8…I1N14).

Выбор величины фиксированного перемещения (JOG). Осуществляется при помощи сигналов слова **W16K1:**

U16K16 – величина перемещения 0,001 мм;
U16K17 – величина перемещения 0,01 мм;
U16K18 – величина перемещения 0,1 мм;
U16K19 – величина перемещения 1,0 мм;
U16K20 – величина перемещения 10,0 мм;
U16K21 – величина перемещения 100,0 мм.

Формирование сигнала пуска цикла CYST (U10K4)

Формирование сигналов индикации перемещения

Факт нахождения той или иной координаты в движении можно индикатировать, используя сигналы слова **W0K2 (MOV i):**

I0K16 – 1-я ось (здесь X);
I0K17 – 2-я ось (здесь Y);
I0K18 – 3-я ось (здесь Z);
I0K19 – 4-я ось;
I0K20 – 5-я ось;
I0K21 – 6-я ось;
I0K22 – 7-я ось;
I0K23 – 8-я ось.

Аналогично, в системе обменных сигналов присутствуют фиксированные адреса и параметры для управления главным приводом, поиском и сменой инструмента, многими специальными режимами. Подробное рассмотрение данных вопросов выходит за рамки настоящей книги. Адресуем читателя к сопроводительной документации и к [61–63].

СПИСОК ЛИТЕРАТУРЫ

1. Паничев Н. А. Разлом. Записки министра СССР. – М.: Русская новь, 2004. – 400 с.
2. Федотов В. А. Украденные победы моего поколения. Москва, 2008. – 446 с.
3. Гаврилов М. А. Теория релейно-контактных схем. М.: АН СССР, 1959, – 303 с.
4. Проектирование бесконтактных управляющих логических устройств промышленной автоматики / Г. Р. Грейнер, В. П. Ильяшенко, В. П. Май и др. М.: Энергия, 1977. – 384 с.
5. Разыграев А. Н. Построение релейных электросхем металлорежущих станков. М. – Л.: Энергия, 1964. – 72 с.
6. Юрасов А. Н. Теория построения релейных схем. М. – Л.: Госэнергоиздат, 1962. – 120 с.
7. Будинский. Я. Логические цепи в цифровой технике: Пер. с чешск. М.: Энергия, 1978. – 456 с.
8. Трачик В. Дискретные устройства автоматики: Пер. с польск. М.: Энергия, 1978. – 456 с.
9. Червонный А. Л. Реле и элементы промышленной автоматики. Практическое пособие для инженеров. – М.: РадиоСофт, 2012. – 208 с.
10. Электрооборудование кузнечно-прессовых машин: Справочник / В. Е. Стоколов, Г. С. Усышкин, В. М. Степанов и др. – 2-е изд., перераб. и доп. – М.: Машиностроение, 1981. – 304 с.
11. Иванова О. И., Лазарев В. Г., Пийль Е. И. Синтез электронных схем дискретного действия. М.: Связь, 1964. – 176 с.
12. Микро-ЭВМ: Пер. с англ. Под ред. А. Дирксена. М.: Энергоатомиздат, 1982. – 382 с.
13. Минскер Э. И., Сушев М. И. Разработка релейно-контактных схем управления производственных механизмов. М.: Энергия, 1972. – 136 с.
14. Мишель Ж. Программируемые контроллеры: Архитектура и применение / Пер. с фр. М.: Машиностроение, 1992. – 320 с.
15. Титце У., Шенк К. Полупроводниковая схемотехника: Пер. с нем. М.: Мир, 1983. – 512 с.
16. Хоровиц П., Хилл У. Искусство схемотехники: В 2-х томах. Пер. с англ. М.: Мир, 1983, 598 с (т. I). – 590с (т. II).
17. Чернов Е. А. Проектирование станочной электроавтоматики. М.: Машиностроение, 1989. – 304 с.
18. Чернов Е. А. Электропривод и электрооборудование в автоматизированном производстве. М.: Машиностроение, 1992. – 304 с.

19. Вешеневский С. Н. Характеристики двигателей в электроприводе. Изд. 6-е, исправленное. М.: Энергия. 1977. – 432 с.
20. Хализев Г. П., Серов В. И. Расчет пусковых, тормозных и регулировочных устройств для электродвигателей. М.: Высшая школа. 1966. – 308 с.
21. Горбачев Г. Н., Чаплыгин Е. Е. Промышленная электроника: Учебник для вузов / Под ред. В. А. Лабунова. М.: Энергоатомиздат, 1988. – 320 с.
22. Электронные промышленные устройства: Учеб. для студ. вузов спец. «Пром. электрон» / В. И. Васильев, Ю. М. Гусев, В. Н. Миронов и др. М.: Высш.шк. 1988. – 303 с.
23. Марголин Ш. М., Гуров А. С. Функциональные узлы схем автоматического управления: Справочное пособие. М.: Энергоатомиздат, 1983. – 168 с.
24. Юдицкий С. А., Тагаевская А. А., Ефремова Т. К. Проектирование дискретных систем автоматики. – М.: Машиностроение, 1980. – 232 с.
25. Цифровые вычислительные машины (элементы, узлы и устройства машин). Лабораторный практикум. Под ред. Г. Н. Соловьева. Учебное пособие для вузов. М.: Атомиздат, 1977. – 312 с.
26. Лазарев В. Г., Маркин Н. П., Лазарев Ю. В. Проектирование дискретных устройств автоматики. Учеб. пособие для вузов связи. – М.: Радио и связь, 1985. – 168 с.
27. Кофлин Р., Дрискол Ф. Операционные усилители и линейные интегральные схемы. М.: Мир, 1979. – 360 с.
28. Потемкин И. С. Функциональные узлы цифровой автоматики. – М.: Энергоатмиздат, 1988. – 320 с.
29. Пухальский Г. И., Новосельская Т. Я. Проектирование дискретных устройств на интегральных микросхемах: Справочник. – М.: Радио и связь, 1990. – 304 с.
30. Федорков Б. Г., Телец В. А. Микросхемы ЦАП и АЦП: функционирование, параметры, применение. М.: Энергоатомиздат, 1990. – 320 с.
31. Шило В. Л. Популярные цифровые микросхемы. Справочник. 2-е изд., испр. – Челябинск: металлургия, Челябинское отд., 1989. – 352с (массовая радиобиблиотека. Вып. 1111).
32. Klockner Moeller, Wiring Manual, 1998.
33. Schematheque Elecctrotechnique, Telemecanique, Editons CITEF, Rueil-Maimaison, 1986.
34. Шалыто А. А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. – СПб. : Наука, 2000. – 780 с.
35. Петров И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. – М.: СОЛОН-Пресс, 2004. – 256 с. – (Серия «Библиотека инженера»).
36. Босинзон М. А. Системы числового программного управления: учеб. пособие / М. А. Босинзон, Г. М. Мартинов. – М.: Логос, 2005. – 296 с.

37. Ловыгин А. А. Современный станок с ЧПУ с CAD/CAM – система / А. А. Ловыгин, Л. В. Теверовский. – М.: ДМК Пресс, 2012. – 279 с.
38. Гусев Н. В. Системы цифрового управления многокоординатными следящими электроприводами: учеб. пособие / Н. В. Гусев, В. Г. Букреев. – Томск.: Изд-во ТПУ, 2010. – 213 с.
39. Сосонкин В. Л. Системы числового программного управления; учеб. пособие / В. Л. Сосонкин, Г. М. Мартинов. – М.: Логос, 2005. – 296 с.
40. Дюбей Г. К. Основные принципы устройства электроприводов: [пер. с англ.] / Г. К. Дюбей. – М.: Техносфера, 2009. – 479 с.
41. Усынин Ю. С. Системы управления электроприводов: учеб. пособие / Ю. С. Усынин. – Челябинск: Изд-во ЮУрГУ, 2004. – 328 с.
42. Фираго Б. И. Теория электропривода: учеб. пособие / Б. И. Фираго, Л. Б. Павлячик. – Мн.: Техноперспектива, 2004. – 527 с.
43. Фираго Б. И. Регулируемые электроприводы переменного тока / Б. И. Фираго, Л. Б. Павлячик. – Мн.: Техноперспектива, 2006. – 363 с.
44. Чернов Е. А. Электроприводы подач станков с ЧПУ: Справочное пособие / Е. А. Чернов, В. П. Кузьмин, Синичкин С. Г. – Горький, ВВКИ, 1986. – 272 с.
45. Чернов Е. А. Комплектные электроприводы станков с ЧПУ: Справочное пособие / Е. А. Чернов, В. П. Кузьмин. – Горький, ВВКИ, 1989. – 320 с.
46. Чернов Е. А. Станочные электроприводы переменного тока: Справочное пособие / Е. А. Чернов. – Москва, Вираж-Центр, 1997. – 232 с.
47. Чернов Е. А. Станочные электроприводы постоянного тока: Справочное пособие / Е. А. Чернов. – Нижний Новгород, из-е автора, 1998. – 310 с.
48. Чернов Е. А. Комплектные станочные электроприводы: Справочное пособие / Е. А. Чернов. – Нижний Новгород, из-е автора, 1999. – 330 с.
49. Чернов Е. А. Управление поисковыми станочными механизмами: справочное пособие / Е. А. Чернов. – Н. Новгород: Промэлектронсервис, 2001. – 98 с.
50. Чернов Е. А. Электроавтоматика револьверных головок токарных станков с ЧПУ: справочное пособие / Е. А. Чернов. – Н. Новгород: Промэлектронсервис, 2011. – 232 с.
51. Чернов Е. А. Проектирование электроавтоматики на базе устройства ЧПУ типа FMS-3000: справочное пособие / Е. А. Чернов. – Н. Новгород.: Промэлектронсервис, 2007. – 180 с.
52. Чернов Е. А. Инструкция по проектированию электроавтоматики на базе устройства ЧПУ типа МАЯК-600: справочное пособие / Е. А. Чернов. – Н. Новгород: Промэлектронсервис, 2007. – 136 с.
53. Чернов Е. А. Проектирование электроавтоматики на базе устройства ЧПУ типа МИКРОС 12Т(Ф): справочное пособие / Е. А. Чернов. – Н. Новгород.: Промэлектронсервис, 2009. – 148 с.

54. Чернов Е. А. Проектирование электроавтоматики на базе устройства ЧПУ типа FANUC: справочное пособие / Е. А. Чернов. – Н. Новгород: Промэлектронсервис, 2009. – 296 с.
55. Чернов Е. А. Типовые схемы релейно-контактного управления асинхронными двигателями / Е. А. Чернов, И. Н. Филатов. – Н. – Новгород, НГТУ им. Р. Е. Алексеева, 2015. – 142 с.
56. Чернов Е. А. Управление подачей металлорежущих станков / Е. А. Чернов, И. Н. Филатов, В. Л. Мельников. – Н. – Новгород, НГТУ им. Р. Е. Алексеева, 2019. – 266 с.
57. Терехов В. М. Системы управления электроприводов / В. М. Терехов, О. И. Осипов. – Москва, Академия, 2005. – 304 с.
58. Онищенко Г. Б. Теория электропривода / Г. Б. Онищенко. – Москва, ООО «Образование и исследование», 2013. – 352 с.
59. FMS in Japan. Japan Machinery exporters Association, 1980. – 157 с.
60. Рогинский В. Н. Основы дискретной автоматики (Статика и динамика дискретных автоматов), М., «Связь», 1975, – 432 с.
61. Чернов Е. А. Электроавтоматика металлорежущих станков. Том 1. Подготовительный курс: монография. Москва; Вологда: Инфра-Инженерия, 2021. – 512 с.
62. Чернов Е. А. Электроавтоматика металлорежущих станков. Том 2. Основной курс (базовая электроавтоматика): монография. Москва; Вологда: Инфра-Инженерия, 2021. – 472 с.
63. Чернов Е. А. Электроавтоматика металлорежущих станков. Том 3. Основной курс (электроавтоматика многооперационных станков): монография. Москва; Вологда: Инфра-Инженерия, 2021. – 464 с.
64. Чернов Е.А., Костенко А.И. Электроавтоматика универсальных и программных.: монография. Москва; Вологда: Инфра-Инженерия, 2022. – 524 с.
65. Чернов Е. А. Программируемые контроллеры в промышленной электроавтоматике : Учебное пособие / Е. А. Чернов. – Горький, ГпТИ им. А. А. Жданова, 1990. – 80 с.
66. Автоматика: учебник и практикум для вузов / А. С. Серебряков, Д. А. Семенов, Е. А. Чернов; под общ. ред. А. С. Серебрякова. – 2-е изд. – Москва: Издательство Юрайт, 2022. – 476 с.
67. Технические каталоги фирмы «СКБ ИС».
68. Технические каталоги фирмы «Балт-Систем».
69. Технические каталоги фирмы «Дельта».
70. Технические каталоги фирмы «Омрон».
71. Технические каталоги фирмы «Мицубиси».
72. Технические каталоги фирмы «Шнайдер».

ОГЛАВЛЕНИЕ

Книги почтой

Заказ можно сделать на сайте издательства

www.infra-e.ru

№ п/п	Наименование книги
1	Автоматизация производственных процессов в машиностроении. Робототехника, робототехнические комплексы. Практикум
2	Автоматизированные системы кузнечно-штамповочного производства.
3	Бесконтактные опоры высокоскоростных роторных систем. Эксплуатация и проектирование
4	Блочно-матричный метод математического моделирования поверхностей
5	Вибродиагностика: теория и практика
6	Вибромониторинг промышленных машин
7	Гидропневмопривод. Теория и практика
8	Детали машин
9	Детали машин и основы конструирования
10	Запорные клапаны на высокие параметры: исследования и проектирование
11	Защита интеллектуальной собственности в машиностроении
12	Инженерная графика
13	Инженерная и компьютерная графика. Издание 2-е
14	Инновационное проектирование цифрового производства в машиностроении
15	Интеллектуальная система автоматизированного проектирования процессов резания при токарной обработке материалов
16	Исправление погрешностей и стабилизация геометрических параметров изделий малой жесткости. Микродинамический метод
17	Исследование механических свойств конструкционных материалов в разных эксплуатационных условиях
18	Комплектование шариковых подшипников по критериям минимальной нагрузки на тела качения и требуемой величины зазора
19	Компьютерная графика в машиностроении
20	Конструкционная прочность
21	Контактные задачи в герметологии неподвижных соединений
22	Лабораторный практикум по технологии конструкционных материалов
23	Математическое моделирование и оптимизация механической обработки
24	Математическое обеспечение чертежа при конструировании деталей в машиностроении
25	Материаловедение
26	Материаловедение и технология конструкционных материалов
27	Материалы в современном машиностроении
28	Метод подобия в технологии машиностроения
29	Метод стохастического комплектования и сборки шариковых подшипников
30	Механика машин и конструирование привода: курсовое проектирование
31	Моделирование и оптимизация перспективных схем электрохимической обработки
32	Модернизация двигателей внутреннего сгорания: цилиндропоршневая группа нового поколения
33	Модернизация станочного парка промышленных предприятий
34	Начала технических знаний: введение в основы устройства и работы машин и механизмов
35	Непрерывное литьё заготовок. Кристаллизаторы и зона вторичного охлаждения
36	Обеспечение качества продукции в машиностроении
37	Оптимальное проектирование затворов трубопроводной арматуры

38	Организация и методология научных исследований в машиностроении
39	Основы вибродиагностики и средства измерения вибрации
40	Основы инженерии поверхностей трения
41	Основы механической обработки деталей. Точение и фрезерование
42	Основы программирования для станков с ЧПУ в CAM-системе
43	Основы технологии машиностроения
44	Основы технологии машиностроения
45	Основы технологии производства металлорежущего инструмента
46	Особенности лезвийной механической обработки труднообрабатываемых материалов
47	Особенности формообразующих операций обработки корпусных деталей из коррозионностойких металлов
48	Отказы деталей машин. Анализ причин, техническая диагностика и профилактика
49	Оценка точности зубофрезерных станков
50	Перспективные направления развития материалов и методов их обработки
51	Перспективные промышленные технологии лазерной обработки
52	Повышение срока службы чугунных деталей гидроцилиндров
53	Повышение срока службы чугунных деталей зубчатых и червячных передач
54	Подшипники с газовой смазкой для турбомашин
55	Прикладные методы теории надежности технических объектов и технологических систем
56	Прогрессивные технологии обработки деталей газотурбинных двигателей
57	Проектирование и процессы формообразования фрезерного инструмента
58	Проектирование сложнопрофильного режущего инструмента
59	Проектирование сменных многогранных пластин для токарных резцов
60	Процессы и операции формообразования
61	Прочностная надежность и долговечность деталей машин и конструкций
62	Расчет параметров и показателей процесса резания
63	Расчет припусков и проектирование заготовок
54	Расчет режимов резания для операций механической обработки
65	Расчет режимов резания при точении с учетом виброустойчивости технологической системы
66	Расчёт режимов резания. Курсовое и дипломное проектирование по технологии машиностроения
67	Режущий инструмент. Зуборезные долбяки с оптимальными параметрами
68	Ресурсосбережение в машиностроении и других отраслях при использовании закрученных потоков газов и жидкостей
69	Системные подходы в задачах динамики машин, приборов и аппаратуры
70	СЛЕСАРНОЕ ДЕЛО. Слесарные работы при изготовлении и ремонте машин. Книга 1
71	СЛЕСАРНОЕ ДЕЛО. Механическая обработка деталей на станках. Книга 2
72	СЛЕСАРНОЕ ДЕЛО. Сборка производственных машин. Книга 3
73	Сопротивление коррозионной усталости технологически обработанных металлов и сплавов
74	Сплавы в машиностроении
75	Справочник конструктора. Книга 1. Машины и механизмы. Издание 3-е, испр. и доп.
76	Справочник конструктора. Книга 2. Проектирование машин и их деталей. Издание 3-е, испр. и доп.
77	Справочник мастера машиностроительного производства. Издание 2-е, испр. и доп.
78	Теория и практика производства червячных передач общего вида. Издание 2-е

79	Теория механизмов и машин. Курсовое проектирование.
80	Техническая диагностика механического оборудования
81	Техническая механика
82	Технологии ремонта деталей авиационных двигателей
83	Технологические основы восстановления промышленного оборудования современными полимерными материалами
84	Технологические процессы и их контроль
85	Технологическое оборудование машиностроительных заводов
86	Технология конструкционных материалов. Производство заготовок
87	Технология машиностроения
88	Технология машиностроения
89	Технология машиностроения. Проектирование технологии изготовления деталей
90	Технология машиностроения. Специальная часть
91	Технология металлообрабатывающего производства. Введение в специальность
92	Технология ремонта машин
93	Технология упрочняющей механико-термической обработки
94	Токарная обработка деталей из коррозионностойких сплавов
95	Токарная обработка. Издание 9-е
96	Управление качеством машин и технологий
97	Управление производственными системами
98	Физические основы технологических процессов в машиностроении
99	Экспертные методы управления технологичностью промышленных изделий
100	Эксплуатация подшипников качения
101	Электроискровые толстослойные покрытия повышенной сплошности
102	Электрофизические и электрохимические методы обработки в машиностроении

Учебное издание

Чернов Евгений Александрович

ПРОГРАММИРУЕМ PLC

Учебное пособие

ISBN 978-5-9729-1474-6

Подписано в печать 31.03.2023
Формат 60×84/16. Бумага офсетная.
Гарнитура «Таймс».

Издательство «Инфра-Инженерия»
160011, г. Вологда, ул. Козленская, д. 63
Тел.: 8 (800) 250-66-01
E-mail: booking@infra-e.ru
https://infra-e.ru