МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

МЕЖДУНАРОДНЫЙ ИНСТИТУТ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

Кафедра «Информационные системы и технологии»

Ю. Е. Лившиц
В. И. Лакин
Ю. И. Монич

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ДЛЯ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

Учебно-методическое пособие и лабораторные работы

Часть 1

Минск
БНТУ
2014

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

МЕЖДУНАРОДНЫЙ ИНСТИТУТ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

Кафедра «Информационные системы и технологии»

Ю. Е. Лившиц
В. И. Лакин
Ю. И. Монич

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ДЛЯ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

Учебно-методическое пособие и лабораторные работы
для студентов всех форм обучения специальностей

1-53 01 01 «Автоматизация технологических процессов и производств»,
1-53 01 06 «Промышленные роботы и робототехнические комплексы»,
1-40 01 01 «Программное обеспечение информационных технологий»,
1-40 01 02 «Информационные системы и технологии»

В 2 частях

Ч а с т ь 1

Минск
БНТУ
2014

УДК 004.382+004.31(076.5)(075.8)
ББК 32.97я7
Л55

Рецензенты :
канд. техн. наук, доцент, заведующий кафедрой РЭС БГУИР
И. Н. Цырельчук ;
канд. техн. наук, доцент БГУИР *В. А. Алексеев*

Лившиц, Ю. Е.

Л55 Программируемые логические контроллеры для управления технологическими процессами : учебно-методическое пособие и лабораторные работы для студентов всех форм обучения специальностей *1-53 01 01 «Автоматизация технологических процессов и производств, 1-53 01 06 «Промышленные роботы и робототехнические комплексы», 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 01 02 «Информационные системы и технологии»*: в 2 ч. / Ю. Е. Лившиц, В. И. Лакин, Ю. И. Монич. – Минск : БНТУ, 2014. – Ч. 1. – 206 с.

ISBN 978-985-550-022-4 (Ч. 1).

В первой части учебно-методического пособия рассмотрены структура, основные характеристики программируемых логических контроллеров (ПЛК). Показаны возможности конфигурации системы управления на базе ПЛК Mitsubishi серии MELSEC FX0S, FX2N.

Приведена классификация языков программирования ПЛК по стандарту МЭК 1131-3 и основные правила составления программ на языке релейно-контактных схем (LD).

В приложениях дан краткий обзор стандартов сетевого взаимодействия ПЛК и SCADA-систем.

Во второй части приведен курс лабораторных работ для получения практических навыков создания автоматизированных систем управления на базе ПЛК.

Авторы глубоко признательны научно-производственному объединению «Техникон» за информационную и техническую поддержку, которая позволила подготовить данное издание.

УДК 004.382+004.31(076.5)(075.8)
ББК 32.97я7

ISBN 978-985-550-022-4 (Ч. 1)
ISBN 978-985-550-024-8

Оглавление

Введение

Необходимость использования контроллеров назрела в начале 1960-ых, когда промышленность начала предъявлять высокие требования к эффективному использованию производственных мощностей, а существующие решения на основе релейно-контактных схем не могли обеспечить гибкое и эффективное управление технологическими процессами, так как изменение технологических циклов требовало замены большого числа элементов управления и контроля. Громоздкость и ограниченный срок службы реле требовали создания сложных систем контроля, а поиск неисправности среди 1000 реле требовал содержания большого числа специалистов. Создание промышленных контроллеров позволило объединить сотни, тысячи реле, таймеров, счетчиков в единый и компактный модуль. Возможность перепрограммирования позволила предприятиям быстро перестраивать производство в соответствии с требованиями рынка. Требования к управлению на расстоянии начали появляться приблизительно в 1973. С момента, когда Программируемые Логические Контроллеры (ПЛК) получили возможность управлять другим ПЛК и могли находиться далеко от оборудования, которым они управляли, вопрос о необходимости перехода на повсеместное использование контроллеров стал очевидным для всех.

ПЛК может использоваться повсеместно там, где есть производство – любая задача, которая требует использования электрических устройств управления, имеет потребность в ПЛК. Например: предположим, что при включении выключателя нам необходимо запустить электропривод на 15 секунд, а затем выключить независимо от того, как долго выключатель включен. С помощью таймера мы можем легко решить эту задачу, но если для решения технологического процесса необходимо включить 10 выключателей и электроприводов? Нам потребуется 10 таймеров, а для расчета числа циклов включения-выключения нам понадобится такое же количество внешних счетчиков. Использование одного контроллера позволит легко решить эту простую задачу, а возможность изменения программы даст возможность максимально быстро менять технологический процесс в зависимости от текущей задачи.

ПЛК ориентированы на длительную работу в *условиях промышленной среды*. Это обуславливает определенную специфику схемотехнических решений и конструктивного исполнения.

Хороший контроллер обладает мощной, совместимой и инту-

итивно понятной системой программирования, удобен в монтаже и обслуживании, обладает высокой ремонтопригодностью, имеет развитые средства самодиагностики и контроля правильности выполнения прикладных задач, средства интеграции в единую систему, а также надежен и неприхотлив.

РАЗДЕЛ ПЕРВЫЙ.

АППАРАТНАЯ ЧАСТЬ КОНТРОЛЛЕРА

Глава 1. Общие сведения. Введение в ПЛК

1.1 Назначение и структура программируемого контроллера

Программируемый логический контроллер (ПЛК) − специализированное микропроцессорное устройство со встроенным аппаратным и программным обеспечением, которое используется для выполнения функций управления технологическим оборудованием. Прародителями ПЛК были релейные схемы автоматики. Это "родство" до сих пор проявляется в виде жесткой цикличности выполнения программы и своеобразного языка программирования. ПЛК – устройство, доступное для программирования неспециалисту в области информатики и предназначенное для управления последовательными логическими процессами в условиях промышленной среды в реальном масштабе времени. ПЛК циклически опрашивает входы, к которым подключены выключатели, датчики и.т.д., и в зависимости от их состояния («включено» − 1, «выключено» − 0), включает-выключает выходы, а следовательно и подключенные к выходам исполнительные механизмы. Функциональная схема системы управления (СУ) на базе контроллера показана на рисунке 1.1. Используя программное обеспечение, пользователь имеет возможность программировать контроллер или вносить изменения в уже существующую программу.

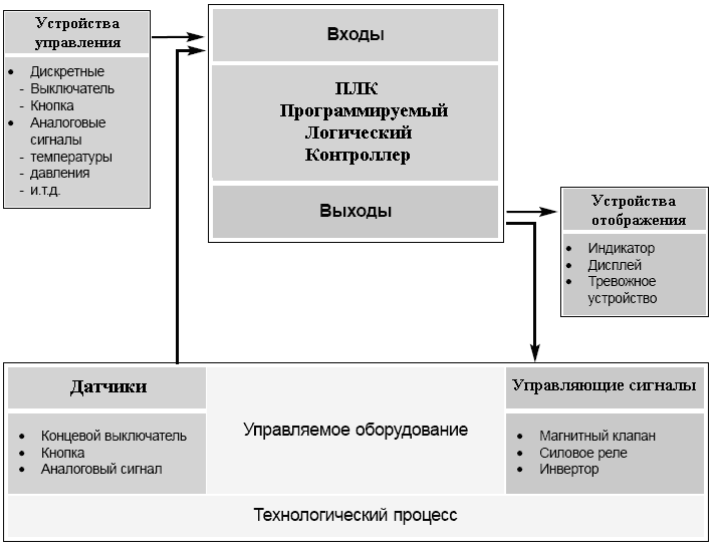


Рисунок 1.1 – Функциональная схема СУ на базе ПЛК.

Программируемый логический контроллер, главным образом состоит из центрального процессора (ЦП), области памяти и функций обработки сигналов ввода/вывода (т.е., входов и выходов). Условно можно назвать такой контроллер основным, или базовым блоком (модулем). Можно считать, что ПЛК − это сотни или тысячи отдельных реле, счетчиков, таймеров и память. Все эти счетчики, таймеры моделируются ЦП и осуществляют логику работы согласно заложенной программы. Структурная схема контроллера показана на рисунке 1.2.

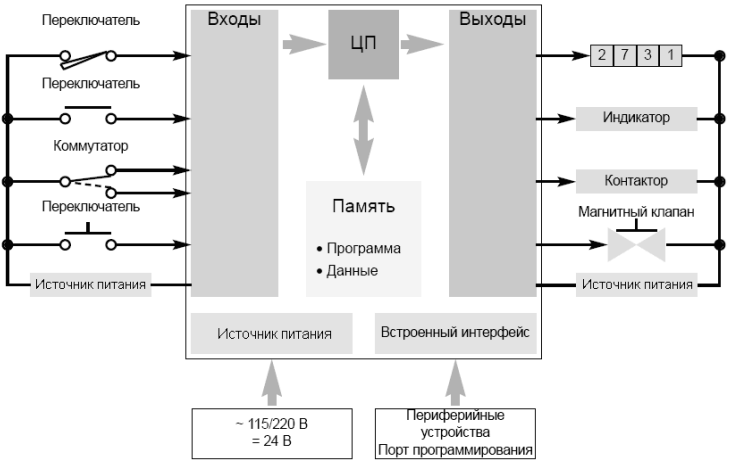


Рисунок 1.2 – Структурная схема контроллера.

•*ВХОДЫ* обеспечивают связь с внешними устройствами. Физически существуют и получают сигналы от выключателей, датчиков, и т.д. Различают аналоговые и цифровые входы, предназначенные для работы с аналоговыми и цифровыми сигналами соответственно.

•*ЦП* – «мозг» ПЛК, осуществляющий логику работы системы. Это процессор, обрабатывающий команды программы и управляющий всеми внутренними элементами контроллера: входами, выходами, счетчиками, таймерами, внутренними реле, регистрами и т.д. На рисунке 1.2 счетчики, таймеры и внутренние реле не показаны отдельно, они входят в состав микросхемы ЦП. Т.е. каждый контроллер обладает фиксированным набором таких элементов, которые приводятся в спецификации.

•*ВНУТРЕННИЕ РЕЛЕ (МЕРКЕРЫ)* предназначены для обеспечения работы программы, т.к. являются своего рода единицами хранения информации (смотри раздел 11.2 «Программирование внутреннего реле»). Наряду с обычными меркерами существуют также и служебные меркеры, несущие специальную смысловую и функциональную нагрузку (например, установка разрешающего флага для запуска высокоскоростных счетчиков). Назначение каждо-

го конкретного служебного меркера приводится в документации к контроллеру.

•*СЧЕТЧИКИ* предназначены для различного рода счета. Отдельно выделяют высокоскоростные счетчики. Как правило, имеются ограничения на скорость счета и значение, до которого ведется счет, для чего необходимо обращаться к документации конкретного контроллера.

•*ТАЙМЕРЫ* предназначены, как правило, для установки времени задержки включения/выключения и т.п. Различаются в основном точностью отсчета времени и, как следствие, назначением.

•*ПАМЯТЬ* – контроллер обладает некоторым объемом памяти, которая в различных контроллерах может иметь различную организацию. Как правило, память делится на рабочую область (ОЗУ), куда загружается программа непосредственно во время работы контроллера, и область данных (EEPROM, MMC и т.п), где хранится программа и различные данные. Часто объем рабочей области измеряется в килобайтах, а объем области данных – в количестве шагов программы.

•*ВСТРОЕННЫЙ ИНТЕРФЕЙС* обеспечивает подключение ПЛК к компьютеру или программатору для обмена данными, в том числе и для перепрограммирования контроллера. В основном это RS-232C (COM-port), RS-422, RS-485 и т.п.

•*ВЫХОДЫ* обеспечивают связь с внешними устройствами, т.е. обеспечивают включение/ выключение исполнительных механизмов. Существуют два варианта исполнения: релейные, полупроводниковые (транзисторные и симисторные). Различают аналоговые и цифровые выходы, предназначенные для работы с цифровыми и аналоговыми сигналами соответственно.

•*ИСТОЧНИК ПИТАНИЯ* предназначен для обеспечения работы контроллера. Могут использоваться внешние источники питания, как постоянного тока +12/24 В, так и переменного – ~110/220 В. Многие контроллеры обладают встроенными сервисными источниками питания (обычно +12/24 В), которые используются для подачи питания на датчики или другие устройства, подключенные к контроллеру для упрощения входных и выходных цепей..

Последнее время имеется тенденция к расширению функциональных возможностей контроллеров за счет реализации встроенных ПИД-регуляторов, часов реального времени, объединения контроллеров в сеть и использования возможностей подключения блоков расширения. В любом случае структура контроллера остается неизменной, и выбор модели определяется только требованиями технологического процесса, а широкий ряд моделей позволяет подобрать контроллер с оптимальным соотношением цена/производительность. Вопросы, связанные с выбором контроллера рассмотрены в разделе 6 «Вопрос выбора ПЛК».

Для понимания работы контроллера на рисунке 1.3 приведен алгоритм его работы.

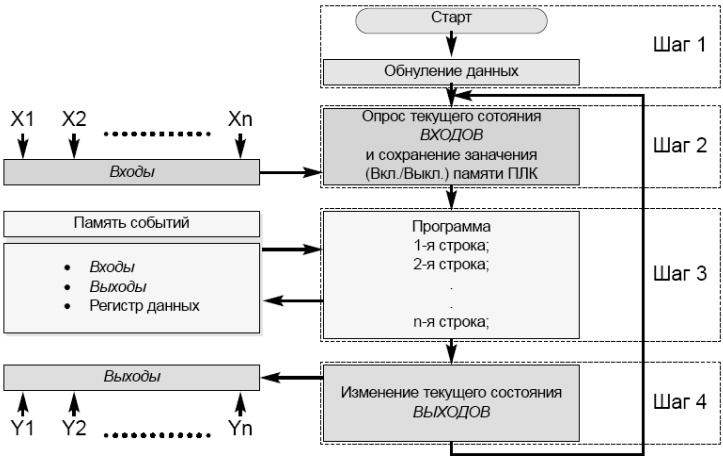


Рисунок 1.3 – Схема алгоритма работы контроллера.

В процессе работы ПЛК непрерывно опрашивает текущее состояние входов X1, X2…Xn и в соответствии с требованиями производственного процесса изменяет состояние выходов Y1, Y2…Yn (вкл./выкл). Можно разделить этот цикл на четыре основных шага.

Шаг первый – инициализация системы. Необходимо помнить, что в случае сбоев по питанию или при выключении контроллера система обязана вернуться в исходное состояние. Не следует недооценивать важности этой части программного кода, так как в противном случае это может привести к сбоям и поломкам оборудования.

Шаг второй – проверка текущего состояния входов. ПЛК проверяет текущее состояние входов и в зависимости от их состояния («вкл». или «выкл.») выполняет последовательные действия, указанные в программе. Состояние любого из входов сохраняется в памяти (в области данных) и может в дальнейшем использоваться при обработке третьего шага программы.

Шаг третий – выполнение программы. Будем считать, что в ходе технологического процесса переключился вход (X1) с «выключено» на «включено», и в соответствии с технологическим процессом нам необходимо изменить текущее состояние выхода (Y1) с «выключено» на «включено». Так как ЦП опросил текущее состояния всех входов и хранит их текущее состояние в памяти, то выбор последующего действия обусловлен только ходом технологического процесса.

Шаг четвертый – изменение текущего состояния выхода. ПЛК изменяет текущее состояние выходов в зависимости от того, какие входы являются выключенными, а какие включенными, исходя из алгоритма записанной в память программы, которая была отработана на третьем шаге. То есть контроллер, физически переключил выход (Y1*)* и включились исполнительные механизмы: лампочка, двигатель и т.д. После этого следует возврат на второй шаг.

1. 2 Классификация контроллеров

Как правило, ПЛК объединяет в себе базовый блок и широкий спектр модулей расширения, позволяющих сконфигурировать оптимальную систему непосредственно под каждую конкретную задачу. В зависимости от данной задачи можно выбрать как небольшие и недорогие не наращиваемые контроллеры (которые не поддерживают подключение модулей расширения), так и масштабируемые решения с возможностью подключения дополнительных модулей с обширным набором возможностей. Невозможно сказать, какой из контроллеров лучше, а какой хуже; широкий ряд контроллеров позволяет решать задачи с оптимальным соотношением цена-производительность, и выбор конкретной модели определяется только требованиями, которые предъявляются решаемой задачей (память, быстродействие, возможность расширения, необходимость создания сети).

Мощное вычислительное ядро современных ПЛК делает их очень похожими на компьютеры. Однако ПЛК это не «железо», а технология. Она включает специфическую аппаратную архитектуру, принцип циклической работы и специализированные языки программирования. Программирование ПЛК осуществляется людьми, хорошо знающими прикладную область, но не обязанными быть специалистами в математике и программировании.

В настоящее время в промышленности используется несколько типов логических контроллеров.

1) **Встроенный**, являющийся неотъемлемой частью агрегата, машины, прибора. Такой контроллер может управлять станком с ЧПУ, современным интеллектуальным аналитическим прибором, автомашинистом и другим оборудованием. Выпускается на раме без специального кожуха, поскольку монтируется в общий корпус оборудования. Такой контроллер может быть как однокристальным, так и представлять собой набор плат и интегральных схем без собственного корпуса.

2) **Автономный модуль**, реализующий функции контроля и управления небольшим изолированным технологическим узлом, как, например, районные котельные, электрические подстанции, резервуарные парки. Автономные контроллеры помещаются в защитные корпуса, рассчитанные на разные условия окружающей среды. Почти всегда эти контроллеры имеют порты для соединения в режиме «точка-точка» (peer-to-peer) с другой аппаратурой и интерфейсы, связывающие отдельные устройства через сеть с другими средствами автоматизации. В контроллер встраивается или подключается к нему специальная панель интерфейса для оператора, состоящая из алфавитно-цифрового дисплея и набора функциональных клавиш.

В этом классе следует выделить специальный тип локальных контроллеров, предназначенных для систем противоаварийной защиты. Такие устройства отличаются особенно высокой надежностью и быстродействием. В них предусматриваются различные варианты полной текущей диагностики неисправностей, вплоть до диагностики неисправностей каждой отдельной платы; защитные коды, предохраняющие информацию от искажений во время передачи и хранения; резервирование как отдельных компонентов, так и всего устройства в целом.

Контроллеры, предназначенные для цепей противоаварийной защиты, должны иметь специальный сертификат, подтверждающий их высокую надежность и устойчивость к внешним воздействиям.

3) Сетевой комплекс контроллеров (PLC, Network).

Сетевые решения на базе ПЛК наиболее широко применяются для управления производственными процессами во всех отраслях промышленности. Минимальный состав данного класса ПТК подразумевает наличие следующих компонентов:

− набор контроллеров;

− несколько дисплейных рабочих станций операторов;

− системную (промышленную) сеть, соединяющую контроллеры между собой и контроллеры с рабочими станциями.

Контроллеры каждого сетевого комплекса, как правило, имеют ряд модификаций, отличающихся друг от друга быстродействием, объемом памяти, возможностями по резервированию, способностью работать в разных условиях окружающей среды, числом каналов входа/выхода. Так что можно подобрать контроллер для каждого узла автоматизируемого агрегата с учетом особенностей и выполняемых функций последнего и использовать один и тот же комплекс для управления разными производственными объектами.

Следует выделить телемеханический тип сетевого комплекса контроллеров, предназначенный для автоматизации объектов, распределенных по большой области пространства.

Системная сеть с характерной структурой и особые физические каналы связи (радиоканалы, выделенные телефонные линии, оптоволоконные кабели) позволяют интегрировать узлы объекта, отстоящие друг от друга на многие десятки километров, в единую систему автоматизации.

Рассматриваемый класс сетевых комплексов контроллеров имеет верхние ограничения как по сложности выполняемых функций, так и по объему автоматизируемого объекта. Обычно телемеханические комплексы решают типовые задачи измерения, контроля, учета, регулирования и блокировки, учитывая до нескольких десятков тысяч измеряемых и контролируемых величин.

Чаще всего сетевые комплексы применяются на уровне цехов машиностроительных заводов, агрегатов нефтеперерабатывающих, нефтехимических и химических производств (правда, не самых мощных), а также цехов предприятий пищевой промышленности. Телемеханические сетевые комплексы контроллеров используются для управления газо- и нефтепроводами, электрическими сетями, транспортными системами.

4) Распределенные маломасштабные системы управления (DCS, Smaller Scale).

Маломасштабные распределенные контроллерные средства в среднем превосходят большинство сетевых комплексов контроллеров по мощности и гибкости структуры, а следовательно, по объему и сложности выполняемых функций. В целом, этот класс еще имеет ряд ограничений по объему автоматизируемого производства и набору реализуемых функций. Однако данная категория средств отличается от предшествующего класса тем, что имеет развитую многоуровневую сетевую структуру. Так, нижний уровень может выполнять связь контроллеров и рабочей станции компактно расположенного технологического узла, а верхний уровень поддерживать взаимодействие нескольких узлов друг с другом и с рабочей станцией диспетчера всего автоматизируемого участка производства. На верхнем уровне (уровне рабочих станций операторов) эти комплексы, по большей части, имеют достаточно развитую информационную сеть. В некоторых случаях расширение сетевой структуры идет в направлении применения стандартных цифровых полевых сетей, соединяющих отдельные контроллеры с удаленными от них блоками ввода/вывода и интеллектуальными приборами. Подобная простая и дешевая сеть соединяет по одной витой паре проводов контроллер с множеством интеллектуальных полевых (заводских) приборов, что резко сокращает длину кабельных сетей на предприятии и уменьшает влияние возможных помех, поскольку исключается передача низковольтной аналоговой информации на значительные расстояния.

Мощность контроллеров, применяемых в этом классе средств, позволяет в дополнение к типовым функциям контроля и управления реализовывать более сложные и объемные алгоритмы управления (например, самонастройку алгоритмов регулирования, адаптивное управление).

Маломасштабные распределенные системы управления используются для автоматизации отдельных средних и крупных агрегатов предприятий непрерывных отраслей промышленности, а также цехов и участков дискретных производств и цехов заводов черной и цветной металлургии.

5) Полномасштабные распределенные системы управления (DCS, Full Scale).

Это наиболее мощный по возможностям и охвату производства класс контроллерных средств, практически не имеющий границ ни по выполняемым на производстве функциям, ни по объему авто-

матизируемого производственного объекта. Нередки примеры использования одной такой системы для автоматизации производственной деятельности целого крупномасштабного предприятия.

Описываемая группа контроллерных средств отличается:

− развитой многоуровневой структурой, предусматривающей выделение трех уровней: информационного, системного и полевого, причем для организации отдельных уровней могут использоваться разные варианты построения сетей;

− клиент-серверным режимом работы;

− выходом на корпоративную сеть предприятия, систему управления бизнес-процессами, глобальную сеть Интернет, а также на уровень интеллектуальных приборов;

− широким модельным рядом применяемых контроллеров, различающихся по числу входов/выходов, быстродействию, объему памяти разного типа, возможностям по резервированию, наличию встроенных и удаленных интеллектуальных блоков ввода/вывода на все виды аналоговых и дискретных сигналов;

− широким диапазоном рабочих станций;

− мощным современным программным обеспечением, в состав которого входят:

а) интерфейсы операторов с системой управления, предусматривающие различные варианты построения на разных уровнях управления;

б) набор технологических языков с объемными библиотеками типовых программных модулей для решения задач контроля, логического управления и регулирования;

в) универсальные прикладные пакеты программ, реализующие типовые функции управления отдельными агрегатами, диспетчерское управление участками производства, технический учет и планирование производства в целом,

г) системы автоматизированного проектирования и конструкторского документооборота для разработки системы автоматизации.

Полномасштабные распределенные системы управления устанавливаются на электростанциях, крупных агрегатах типа «котел-турбина», нефтеперерабатывающих заводах для управления крекинг-процессами, охватывают все производство на химических и нефтехимических заводах и т. д.

К перечисленным выше видам контроллеров можно добавить также то, что существуют программы, имитирующие работу ПЛК на компьютере, так называемые SoftPLC (программные ПЛК). В этом

случае, удается совместить на одной машине контроллер, средства программирования и визуализации. Недостатком такого решения является значительное время восстановления при сбоях и повреждениях. Перезагрузка операционной системы (ОС) и запуск прикладной задачи может занимать несколько минут. Переустановка и настройка ОС, драйверов оборудования и прикладных программ требует значительного времени и высокой квалификации обслуживающего персонала, тогда как системное программное обеспечение ПЛК расположено в постоянной памяти в адресном пространстве центрального процессора и всегда готово к работе. По включению питания, ПЛК готов взять на себя управление системой уже через несколько миллисекунд.

Глава 2. Основные характеристики и параметры ПЛК

В методическом пособии рассмотрены вопросы практического использования контроллеров для автоматизации в различных областях техники на примере контроллеров Mitsubishi серии MELSEC FX2N и FX0S, как типичных представителей недорогих и широко используемых контроллеров. MELSEC FX0S − наиболее простой и недорогой логический контроллер, включающий в себя все преимущества системы ПЛК в компактном корпусе. Такое устройство представляет собой экономичную, с точки зрения вложений, альтернативу стандартным контакторным и релейным решениям. В свою очередь, MELSEC FX2N имеет более мощный процессор и обладает большей функциональностью, а также возможностью построения более сложных систем (т.е. с возможностью подключения модулей расширения и с поддержкой сетевого взаимодействия).

Для наглядности в таблице 2.1 приведены данные двух типичных спецификаций контроллеров.

Таблица 2.1 – Технические характеристики контроллеров

Электрические параметры		FX2N -16MR-UA1/UL	FX0S-30MR-DS
1		2	3
Питание		~100…240 В	= 12…24 В
Кол-во входов-выходов		16	30
Потребляемая мощность		30 Вт	8 Вт
Пиковый ток при включении		макс. 40А < 5мс/ ~100 В макс. 60А < 5мс/ 200 В	макс 60А, <1,5 мс ,=24 В
Защита от КЗ		внешними цепями	внешними цепями
Предохранитель		3,15А	3,15А
Допустимый провал питания		10 мс	5 мс
Ток источника питания внутренней шины 5V DC		290 мА	шина отсутствует
Ток сервисного источника питания 24V DC		460 мА, пульсации при макс нагрузке: ≤ ±5%	нет сервисного источника
Входы:			
Количество		8	16
Входной ток и напряжение (макс)		4,7 мА / ~100 В /50Гц 6,2 мА/ ~110 В /60Гц	=24В / 8,5 мА
Ток переключения ВЫКЛ→ ВКЛ / ВКЛ → ВЫКЛ		80В 3,8мА / 30В 1,7мА мин ток лог1/ макс ток лог0	>18В >4,5мА / <4В <1,5мА мин ток лог1/ макс ток лог0
Быстродействие		25 мс	0,1…15 мс, регулируемое
Входное сопротивление (импеданс)		21 кОм / 50 Гц 18 кОм / 60 Гц	
Гальваноразвязка		опторазвязка между входами и питанием	
Выходы:			
Количество		8	14
Тип выхода		транзистор	реле
Уровень коммутируемого напряжения (макс)		< ~240 В < =30 В	5…30 В
Макс. выход-ной ток	- на канал	0,5 А	2 А
	- на группу	0,8 А	8 А

Продолжение таблицы 2.1

1		2	3
Коммутируемая мощность	индуктивная нагрузка	12 Вт (0,5А / =24 В)	80 ВА, ~120/ 240 В
	активная нагрузка	1,5 Вт (0.0625А / =24 В)	100 Вт (1.17 А / ~85 В 0.4 А / ~250 В)
Минимальная нагрузка			5 мА, при < =24 В
Ток утечки		<0.1 мА/ = 30 В	
Быстродействие		<0,2 мс	10 мс
Гальваноразвязка		реле	реле
Срок службы контактов (число коммутаций)		Бесконтактная коммутация	3.000.000 при 20 ВА 1.000.000 при 35 ВА 200.000 при 80 ВА
Механические параметры			
Масса		0,35 кг	0,5 кг
Размеры (Ш х В х Г), мм		100 х 90 х 75	90 х 90 х 80
Условия эксплуатации			
Диапазон рабочих температур		0 … +55˚ С	0 … +55˚ С
Допустимая влажность воздуха		35…85 % (без конденсата)	35…85 % (без конденсата)
Виброустойчивость (2 часа в 3-х направлениях)	на винтах	10..55 Гц / 0,5 мм / Макс 2g	10..55 Гц / 0,5 мм / Макс 1g
	на DIN рейке	10..55Гц / 0,5 мм / Макс 0,5g	10..55 Гц / 0,5 мм / Макс 0,5g
Ударопрочность (3 цикла в 3 направлениях)		10g	10g
Помехоустойчивость от генератора помех		1000 Vpp, 1мс, при частоте 30..100 Гц	1000 Vpp, 1мс @ 30..100 Гц
Напряжение пробоя изоляции		~1500 В, 1 мин =500 В, 1 мин	~1500 В, 1 мин =500 В, 1 мин
Сопротивление изоляции		5 МОм, U= 500В	5 МОм, U= 500В
Заземление		Класс 3	Класс 3
Высота местности		До 2000 м	До 2000 м
Класс оборудования		II	II
Класс защиты		IP20	IP20
Степень загрязнения окр. среды		2	2
Окружающая среда		избегать сред, содержащих коррозийные газы и электропроводящую пыль и мусор недопустимо воздействие активной среды, минимальное воздействие пыли	

Окончание таблицы 2.1

1	2	3
Программные параметры		
I/O (адресное пространство)	256	128 (+4 опционально)
Диапазон адресов	макс 248 входов (X0-X367) макс 248выходов(Y0-Y367)	макс 84 входа (X1-X123) макс 64 выхода (Y0-Y77)
Память программы	2000 шагов – EEPROM +опциональные носители	800 шагов – EEPROM
Быстродействие	0,1..0,7мкс/лог.инструкцию	0,55…1мкс/лог. инструкцию
Операнды		
Количество инструкций (команд)	Базовых: 27 Прикладных: 128	Базовых: 29 Прикладных: 85
Внутреннее реле	3072	512
Спец. реле	256	256
Step-реле	1000	74
Таймеры	256	84
Задание установок внешними потенцио-метрами	2 потенциометра	1 потенциометр
Счетчики	235	18
Входы быстрого счета импульсов	1-фазн. счет: 6 вх. до 60кГц 2-фазн. счет: 2 вх. до 30кГц	7 вх. до 7 кГц 3 вх. до 14 кГц
Часы реального времени	год/месяц/день/часы/ мин/сек/день недели	год/месяц/день/часы/ мин/сек/день недели
Регистры данных	8000	8000
Файловые регистры	Макс 7000 (всего ≤8000)	Макс 7000 (всего ≤8000)
Индексные регистры	16	2
Спец. регистры	256	256
Указатели	128	64
Допустимое число вложений в программе	8	8
Входы прерываний	6	4
Дополнительно		
Масштабируемость (расширяемость)	да	нет
Число функциональных блоков в системе	8 (ограничено адресным пространством I/O и током внутр. =5В шиной питания)	не поддерживает расширение
Поддержка работы по сети	используя сетевые модули расширения	не поддерживает сетевое взаимодействие
Встроенный интерфейс	RS-485	RS-485

Многие из приведенных параметров очевидны и не возникает каких-либо вопросов по их расшифровке и оценке, некоторые были

рассмотрены ранее или еще будут рассмотрены в рамках данного пособия, однако разберем некоторые из них более подробно.

2.1 Питание

2.1.1 Внешний источник питания

Так как работа ПЛК подразумевается в промышленных условиях, где то и дело приходится сталкиваться с резкими перепадами напряжения питающей сети, он должен быть нечувствительным к колебаниям напряжения. Так, допускается некоторое отклонение напряжения питания от номинальной величины (при питании переменным током ~100-240В +10%,-15% , 50/60 Гц ±10% или при питании постоянным током =24В +20%,-15%), а также полное кратковременное исчезновение (провал) питания на несколько миллисекунд. В это время приоритетно отрабатывается процедура защиты информации, и происходит пересылка содержимого регистров ЦП в энергонезависимую память. В некоторых ПЛК предусмотрена возможность автоматического пуска при восстановлении питания (либо с самого начала программы, либо с прерванной команды с восстановлением ситуации).

Для защиты контроллера при подключении питания необходимо использовать предохранитель, значение которого указано в спецификации. Смотри также раздел 3.4 «Подключение источника питания».

2.1.2 Пиковый ток при включении

Пиковый ток при включении – максимальный ток, который может быть необходим контроллеру во время пуска. Таким образом, необходимо выбирать источник питания достаточной мощности.

2.1.3 Внутренний источник питания

Внутренний (сервисный) источник питания постоянного тока =24 В предназначен для питания внешних устройств – датчиков и специальных модулей расширения, что делает удобным их подключение, однако следует учитывать значения потребляемого тока этими устройствами. (смотри раздел 5 «Расчет энергопотребления»).

Для примера, сервисный источник питания постоянного тока контроллера FX2N 24 В, 460 мА; пульсации при максимальной нагрузке ≤ ±5 %.

2.1.4 Внутренняя шина =5 В

Шина предназначена для питания подключаемых модулей расширения (смотри раздел 5 «Расчет энергопотребления»). В контроллерах, не поддерживающих расширение, она отсутствует.

2.2 Входы ПЛК

2.2.1 Дискретные входы

Один дискретный вход ПЛК способен принимать высокий или низкий уровень электрического сигнала, описываемый двумя состояниями – включено или выключено. На уровне программы – это один бит информации: 1 или 0, соответственно значение логической (булевой) переменной «ИСТИНА» или «ЛОЖЬ».

Кнопки, выключатели, контакты реле, датчики обнаружения предметов и множество приборов с выходом типа «реле» или «открытый коллектор» непосредственно могут быть подключены к дискретным входам ПЛК.

Некоторые первичные приборы систем промышленной автоматики имеют более 2-х состояний. Для их подключения используют несколько дискретных входов. Например, автоматические электронные весы способны контролировать пороги допуска. Они имеют 2 выхода – «меньше нормы» и «больше нормы». Вес объекта определяется двумя битами информации: 01 – «меньше», 00 – «норма», 01 – «больше», 11 – «неисправность прибора». Используя n отдельных входов можно закодировать 2^n состояний. Как правило, в прикладной программе ПЛК соответствующие биты объединяют в отдельную «дискретную» переменную.

Системное программное обеспечение ПЛК включает драйвер, автоматически считывающий физические значения входов в оперативную память. Благодаря этому, прикладному программисту нет необходимости разбираться с внутренним устройством контроллера.

С точки зрения прикладного программиста дискретные входы это наборы бит, доступные для чтения.

Все дискретные входы контроллеров рассчитаны на прием стандартных сигналов согласно спецификации конкретного контроллера. Устройство входа включает индивидуальный светодиодный индикатор, гальваническую развязку и, как правило, защиту от ошибочного подключения. Так, для контроллера MELSEC FX0S (смотрите таблицу 2.1) уровень входного сигнала более 18 В считается логической единицей, а менее 4 В – логическим нулем. Типовое максимальное значение тока одного дискретного входа (при входном напряжении 24В) составляет около 10мА (8.5мА).

У многих контроллеров светодиодные индикаторы включены до гальванической развязки, что позволяет проводить диагностику работы внешних цепей, даже не включая контроллер.

Каждый дискретный вход имеет аналоговый фильтр «срезающий» высокочастотные помехи и верхние гармоники спектра входного сигнала. Частота среза фильтра согласована с программным быстродействием, определяющимся типовым временем рабочего цикла ПЛК. Длительность импульса, который можно надежно зафиксировать дискретным входом общего назначения, составляет 2…3 мс.

Все современные датчики, базирующиеся на разнообразных физических явлениях (емкостные, индуктивные, ультразвуковые, оптические и т.д.), как правило, поставляются со встроенными первичными преобразователями и не требуют дополнительного согласования при подключении к дискретным входам ПЛК.

Не смотря на внешнюю простоту дискретного входа, его схемотехническое решение и элементная база постоянно совершенствуются.

2.2.2 Аналоговые входы

Поскольку ПЛК является цифровой вычислительной машиной, аналоговые входные сигналы обязательно подвергаются аналого-цифровому преобразованию (АЦП). В результате образуется дискретная переменная определенной разрядности. Как правило, в ПЛК применяются 8…12 разрядные преобразователи, т.е. учитываются только 8…12 старших разрядов преобразованного аналогового сигнала. АЦП более высокой разрядности не оправдывают себя, в

первую очередь из-за высокого уровня индустриальных помех, характерных для условий работы контроллеров.

Для аналоговых входов наиболее распространены стандартные диапазоны постоянного напряжения -10..+10 В и 0..+10 В. Для токовых входов это 0..20 мА и 4..20 мА. Для достижения хороших результатов измерений решающую роль играет качество выполнения монтажа внешних аналоговых цепей.

ПЛК может обладать встроенными аналоговыми входами или использовать дополнительно подключаемые модули расширения или адаптеры (смотри раздел 4.2).

Особые классы аналоговых входов представляют входы, предназначенные для подключения термометров сопротивления и термопар. Здесь требуется применение специальных технических решений (трех-точечное включение, источники образцового тока, схемы компенсации холодного спая, схемы линеаризации и т.д.).

Практически все модули аналогового ввода являются многоканальными. Входной коммутатор подключает вход АЦП к необходимому входу модуля. Управление коммутатором и АЦП выполняет драйвер системного программного обеспечения ПЛК. Прикладной программист работает с готовыми значениями аналоговых величин в памяти контроллера аналогично дискретным входам.

2.2.3 Специальные входы

Стандартные дискретные входы ПЛК способны удовлетворить абсолютное большинство потребностей систем промышленной автоматики. Несоответствие физических значений напряжений и токов датчиков решается применением нормирующих преобразователей или заменой нестандартных датчиков. Здесь изготовление специализированных входов не оправдано. Необходимость применения специализированных входов возникает в случаях, когда непосредственная обработка некоторого сигнала программно затруднена. Достаточно часто первичный сигнал содержит избыточную информацию, а программная фильтрация сложна или требует много времени.

Наиболее часто ПЛК оснащаются специализированными счетными входами для измерения длительности, фиксации фронтов и подсчета импульсов.

Например, при измерении положения и скорости вращения вала очень распространены устройства, формирующие определенное количество импульсов за один оборот – квадратурные шифраторы.

Частота следования импульсов может достигать нескольких мегагерц. Даже если процессор ПЛК обладает достаточным быстродействием, непосредственный подсчет импульсов в пользовательской программе будет весьма расточительным по времени. Здесь желательно иметь специализированный аппаратный входной блок, способный провести первичную обработку и сформировать, необходимые для прикладной задачи, величины.

Вторым распространенным специализированным типом входов являются входы способные очень быстро запускать заданные пользовательские задачи с прерыванием выполнения основной программы.

2.3 Выходы ПЛК

2.3.1 Дискретные выходы

Один дискретный выход ПЛК способен коммутировать один электрический сигнал. Также как и дискретный вход, с точки зрения программы это один бит информации, принимающий состояния «ИСТИНА» или «ЛОЖЬ» (смотрите раздел 2.2 «Входы ПЛК»)

Простейший дискретный выход ПЛК выполняется в виде контактов реле. Такой выход достаточно удобен в применении и прост. Однако он обладает характерными недостатками реле – ограниченный ресурс, низкое быстродействие, разрушение контактов при работе на индуктивную нагрузку. Альтернативным решением дискретного выхода является электронный силовой элемент, который выполняется по бесконтактной схеме (транзистор – для нагрузки постоянного тока, симистор – для нагрузки переменного тока). Схема ключа, как правило, содержит индивидуальную светодиодную индикацию, гальваническую развязку и элементы защиты от ошибочного включения и короткого замыкания нагрузки.

Практика эксплуатации доказала нецелесообразность сосредоточения в корпусе ПЛК большого числа силовых коммутирующих элементов. Оптимальным решением является установка силовых коммутирующих приборов максимально близко к нагрузке. В результате, сокращается длина силовых монтажных соединений, снижается стоимость монтажа, упрощается обслуживание, уменьшается уровень электромагнитных помех. Поэтому наиболее широким

спросом пользуются дискретные выходы средней мощности (до 1А, 24В).

Светодиодные индикаторы включения выходов питаются от ПЛК. Это упрощает отладку программы управления без подключения оборудования.

Благодаря применению специальных узлов защиты, дискретный выход контроллера обладает очень высокой надежностью.

2.3.2 Аналоговые выходы

Как и в случае с аналоговыми входами для управления аналоговыми сигналами ПЛК может обладать встроенными аналоговыми выходами или использовать дополнительно подключаемые модули расширения или адаптеры (смотри раздел 4.2).

Примечание

− Практика показывает, что аналоговые выходы менее надежные, чем дискретные, поэтому при конфигурации системы полезно иметь некоторый запас по количеству аналоговых выходов (то же можно сказать и о входах).

− Также полезно иметь запас по количеству дискретных входов/выходов, т.к. часто бывает необходимо подключить уже на стадии эксплуатации дополнительное оборудование, не прибегая к покупке дополнительных модулей расширения. Таким образом, разумно иметь примерно 20% запас свободных каналов.

2.4 Время реакции – быстродействие

2.4.1 Факторы, влияющие на быстродействие

Быстродействие – это основополагающий фактор, влияющий на выбор ПЛК. Примером быстродействия может служить то время, которое требуется человеку для осмысления ответа на поставленный вопрос или для принятия решения. Так и ПЛК необходимо некоторое время для обработки текущего состояния. Для некоторых приложений этот параметр может не являться критическим, и выбор того или иного значения быстродействия должен сопоставляться с реальными требованиями приложения. Так и для контроллера время реакции зависит, как от числа опрашиваемых входов/выходов, так и от производительности самого процессора.

Давайте попробуем проанализировать эту ситуацию. Факторы, влияющие на время реакции контроллера, изображены в виде схемы на рисунке 2.1.

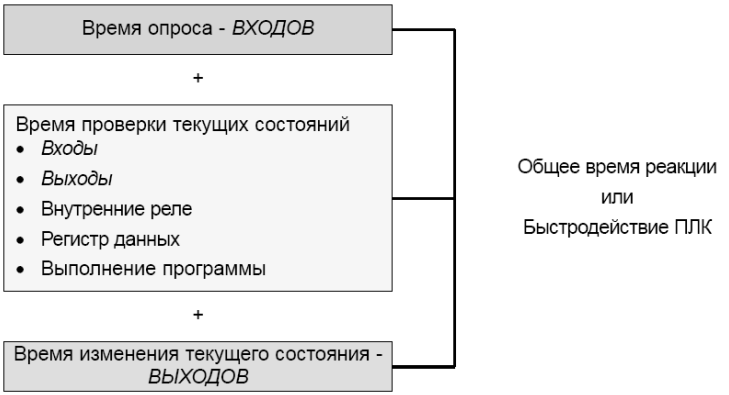


Рисунок 2.1 – Время реакции или быстродействие ПЛК

2.4.2 Действительное быстродействие ПЛК

Теперь, когда мы получили представление о том, как работает контроллер, давайте посмотрим, что в действительности происходит в ПЛК при обработке текущих состояний входов, и как это влияет на время срабатывания выходов. Разберем три типичных случая переключения входов ПЛК, приведенных на рисунке 2.2.

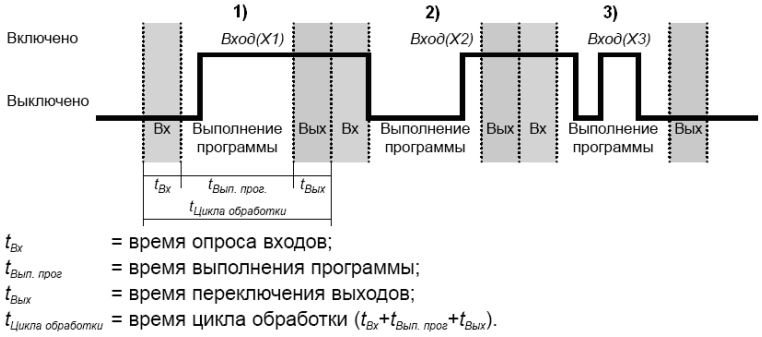


Рисунок 2.2 – Пример переключения входов ПЛК

1) ПЛК не зафиксировал переключение входа (X1). Это произошло потому, что переключение произошло после обработки контроллером состояний входов. А значит, контроллер сможет выполнить операцию, соответствующую входу (X1) только на следующим цикле опроса.

2) ПЛК не зафиксировал переключение входа (X2). Следовательно, если котроллер должен был выключить выход (Y1) при одновременном включении входа (X1) и входа (X2), то это событие не было обработано контроллером и операция не выполнена.

3) ПЛК не зафиксировал переключение входа (X3). И, возможно, не сможет обработать изменение состояния данного входа (X3) никогда.

Есть три варианта решения этой проблемы:

1) Использовать контроллер с более высоким быстродействием;

2) Воспользоваться функцией задержки времени «выкл». Эта функция увеличит длительность входного сигнала (рисунок 2.3)

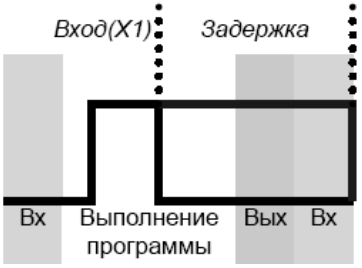


Рисунок 2.3 – Задержка времени включения

3) Использовать функцию обработки прерываний (рисунок 2.4). То есть, как только вход включен, то независимо от того этапа, на котором в настоящий момент находится программа ПЛК, немедленно останавливает выполнение основной программы и выполняет подпрограмму прерыва-

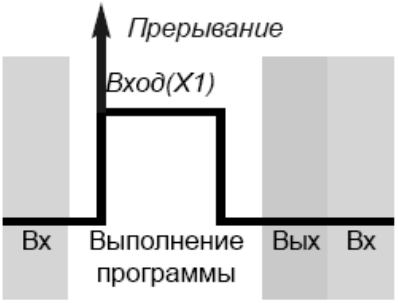


Рисунок 2.4 – Использование прерывания

ния.

Таким образом, из рисунка 2.5 следует: можно считать, что оптимальным временем продолжительности импульса будет являться время прохода одного цикла программы.

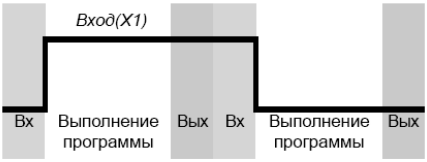


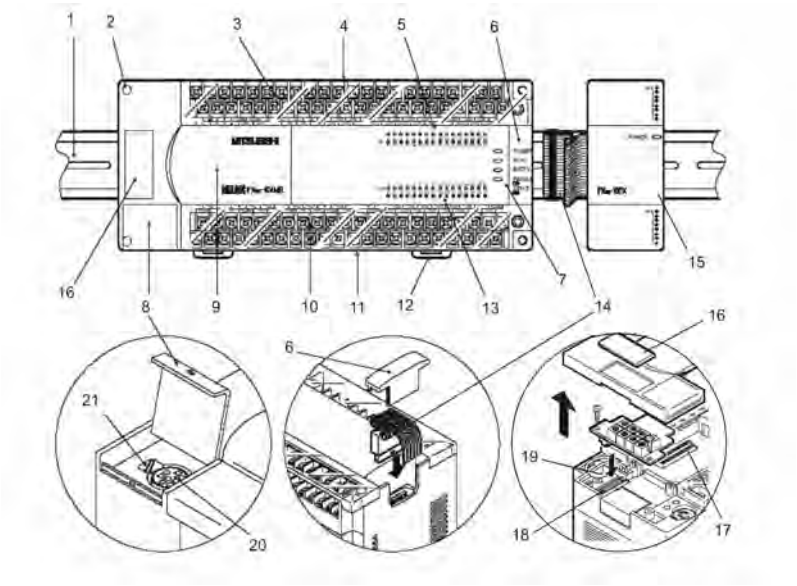
Рисунок 2.5 – Временная диаграмма цикла работы ПЛК

В технических характеристиках ПЛК приводится типовое время рабочего цикла. (в нашем случае в таблице 2.1 приведены составляющие величины рабочего цикла – быстродействие входов, выходов и время исполнения логических инструкций программы). При его измерении пользовательская программа должна содержать 1К (1024) простых логических команд (на языке списка инструкций (IL) по стандарту МЭК 1131-3). Сегодня ПЛК имеют типовое значение времени рабочего цикла, измеряемое единицами миллисекунд и менее. События, требующие быстрой реакции, выделяются в отдельные задачи, приоритетность и период выполнения которых можно изменять.

Глава 3. Установка и подключение ПЛК

3.1 Конструктивные элементы ПЛК

Рассмотрим основные конструктивные элементы контроллера на примере того же Mitsubishi MELSEC FX2N (рисунок 3.1).



1 - DIN-рейка 35мм (1.38 дюйма);
2 - Установочные отверстия;
3 - Входные клеммы;
4 - Заглушка входных клемм;
5 - Индикатор состояния входов;
6 - Заглушка шины расширения;
7 - Индикаторы состояния;
8 - Заглушка порта программирования;
9 - Передняя панель;
10 - Выходные клеммы;
11 - Заглушка выходных клемм;
12 - Фиксатор DIN-рейки;
13 - Индикатор состояния выходов;
14 - Шлейф для подключения модуля расширения;
15 - Модуль расширения;
16 - Заглушка разъема адаптера расширения;
17 - Порт для установки кассеты памяти;
18 - Разъем для адаптера расширения;
19 - Батарея для сохранения данных при отсутствии питания;
20 - Порт программирования;
21 - Переключатель ВКЛ/ВЫКЛ (RUN/STOP);

Рисунок 3.1 – Общий вид контроллера MELSEC FX2N

Обычно блоки промышленных контроллеров устанавливаются на 35 мм (1,38 дюйма) DIN-рейку. Крепление блоков на DIN-рейке осуществляется при помощи специальной защелки. Возможно также крепление на панель с помощью винтов в отверстия.

Подключение модулей расширения, специальных функциональных модулей к базовому процессорному модулю выполняется при помощи стандартного плоского шлейфного кабеля либо специальных выводных корпусных контактов.

3.2 Размещение

Как правило, на промышленных объектах контроллеры устанавливаются в специальные шкафы управления, в которых предусмотрены стандартные разъемы и DIN рейки для крепления основных и периферийных блоков и модулей, что позволяет легко и удобно компоновать целые системы управления. На рисунке 3.2 приведен пример расположения контроллера в шкафу управления.

Однорядное расположение

Двухрядное расположение с использованием дополнительного шлейфного кабеля

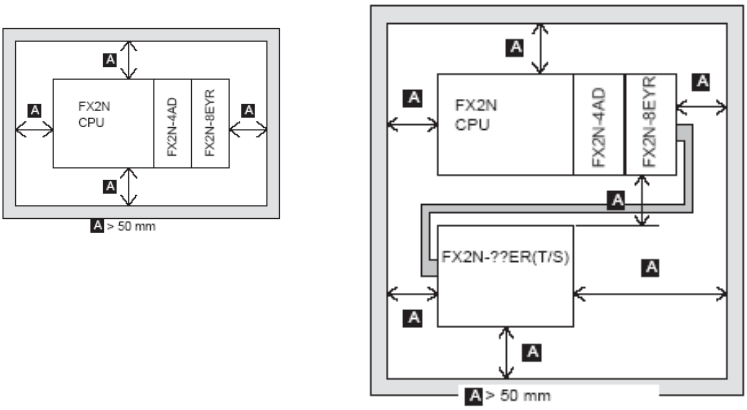


Рисунок 3.2 – Пример расположения контроллеров в шкафу управления

При установке контроллера необходимо обратить особое внимание на соблюдение всех условий эксплуатации, приведенных в техническом паспорте контроллера. В основном различают три типа факторов воздействия агрессивной среды:

▪ *Физические и механические факторы*:

К ним относятся диапазон рабочих температур, атмосферное давление (высота над уровнем моря), влажность, вибрация и удары. Расположение в непосредственной близости от нагревателей, химических реакторов, доменных печей, а также тяжелые климатические условия могут негативно отразиться на работоспособности элементов устройств управления (откуда вытекает необходимость создания систем естественной или принудительной вентиляции ПЛК). Высокий уровень влажности (более 80%) вызывает конденсацию паров и ускоряет коррозию. Уровень влажности менее 35% способствует возникновению электростатических зарядов, вызывающих случайные срабатывания логических схем. При установке устройств рядом с источниками вибрации и ударов сварные, контактные и другие соединения подвергаются опасным воздействиям. Так, рассматриваемые нами контроллеры способны работать при температурах 0…+55˚ С и влажности 35…85 %, выдерживают вибрацию с частотой от 10 до 55Гц с амплитудой 0,5 мм и максимальным ускорением от 0,5 до 2g в течение 2-х часов в каждом направлении X, Y, Z. Также данные контроллеры выдерживают ударную нагрузку с ускорением 10g (3 цикла в каждом направлении X, Y, Z).

▪ *Химические факторы:*

К ним относят вызывающие коррозию газы (Cl_2, H_2S, SO_2), углеводородные пары, металлическая (литейный, сталеплавильный цеха) или минеральная (цементный завод) пыль. Являющаяся действием этих факторов коррозия поражает контакты и микросхемы. Для ее предотвращения чаще всего покрывают лаком платы с печатными схемами и устанавливают фильтры, препятствующие попаданию пыли или агрессивных газов. Иногда ПЛК делают полностью герметичными.

▪ *Электрические факторы:*

Обычно энергия электрической промышленной помехи может достигать 100 мкДж. Энергия переключения схемы ТТЛ при напряжении 5 В, токе 2 мА и продолжительности переключения 10 нс равна 10^{-4} мкДж, т.е в 10^6 раз меньше. Отсюда следует, что уровень помех должен быть снижен на 120 дБ. Основными источниками помех являются: термоЭДС (эффект Пельтье) в несколько милливольт,

разность потенциалов в зоне контакта металлов с различной химической активностью, электростатические и электромагнитные влияния, вызываемые включением индуктивностей и емкостей (расположение вблизи трансформаторов, сварочных агрегатов и др.). Помехи от источников двух первых типов могут внести погрешность в результаты измерений аналоговых величин низкого уровня или вызвать коррозию элементов; для защиты от помех от источников двух последних типов необходимо соответствующее исполнение входов-выходов, например обеспечение эффективной гальванической развязки (оптрон, реле, разделительный трансформатор).

Приведенные в таблице 2.1 данные указывают, что рассматриваемые контроллеры имеют гальваническую развязку входов и выходов и обладают помехозащищенностью от импульсного напряжения амплитудой 1000 В от генератора помех длительностью 1 мс при 30…100 Гц (1000 Vpp, 1 мс при частоте 30…100 Гц).

3.3 Общие рекомендации по электробезопасности

3.3.1 Изоляция

Как и все электротехнические устройства ПЛК должен соответствовать определенным требованиям электробезопасности. Обычно качество изоляции тестируется между всеми рабочими точками устройства а также корпусом и землей (методика проведения испытаний по МЭК 61131-2 Программируемые контроллеры. Общие технические требования и методы испытаний).

3.3.2 Заземление

Используется проводник определенного сечения (обычно не менее 2 мм^2). Сопротивление заземления должно быть менее 100 Ом (класс 3). Заземляющий проводник должен быть присоединен к тому же контуру заземления, что и силовые цепи. Эти требования также указываются приведенным в предыдущем пункте стандартом.

3.3.3 Класс оборудования

Условный номер, присвоенный группе оборудования, для которой применяются определенные средства, используемые для обес-

печения защиты от поражения электрическим током при нормальной эксплуатации и в условиях единичного дефекта функционирования установленного оборудования.

Таким образом, оборудование класса II (смотри таблицу 2.1) — оборудование с прочным и по существу непрерывным корпусом из изоляционного материала, который окружает все проводящие части, за исключением небольших частей типа фабричных марок, винтов и заклепок, отделенных от частей с опасным напряжением изоляцией, эквивалентной, по крайней мере, усиленной изоляции. Подробнее смотри стандарт МЭК 61131-2.

3.3.4 Класс защиты

Код IP показывает степень защиты, которую обеспечивает оболочка (корпус). Расшифровку конкретного кода можно найти в справочнике или в ГОСТ 14254-96.

Так, IP20 (смотри таблицу 2.1) расшифровывается следующим образом:

2 – ПЛК защищен от проникновения внешних твердых предметов диаметром >12,5 мм (это же означает, что человек (оператор) не может пальцами достать до опасных частей ПЛК).

0 – ПЛК не имеет защиты от вредного воздействия в результате проникновения воды.

3.3.5 Степень загрязнения окружающей среды (микросреды)

Условный номер, присвоенный для оценки изолирующих качеств воздушных зазоров и частей поверхности изоляции с целью установления величины загрязнения микросреды:

1) **степень загрязнения 1:** Отсутствие загрязнений или наличие только сухих, непроводящих загрязнений. Загрязнения не существенны;

2) **степень загрязнения 2:** Обычно имеют место только непроводящие загрязнения. Иногда может ожидаться временная проводимость, вызванная конденсацией влаги;

3) **степень загрязнения 3:** Имеют место проводящие загрязнения. Сухие непроводящие загрязнения могут стать проводящими из-за конденсации влаги

3.3.6 Разводка кабелей

Все входящие и выходящие кабели присоединяются к ПЛК к специальному разъему «под винт». Кабельный ввод имеет крышку для защиты от коротких замыканий и нарушения контакта.

ПРЕДУПРЕЖДЕНИЕ

1. Не используйте для входных и выходных сигналов один и тот же провод.
2. Не используйте для входных и выходных сигналов жилы одного и того же многожильного кабеля.
3. Не располагайте сигнальные кабели вблизи силовых. Низковольтные кабели должны быть отделены или изолированы от силовых.
4. При значительной длине проводников входных/выходных сигналов необходимо учитывать потерю напряжения и шумовые помехи.

3.4 Подключение источника питания

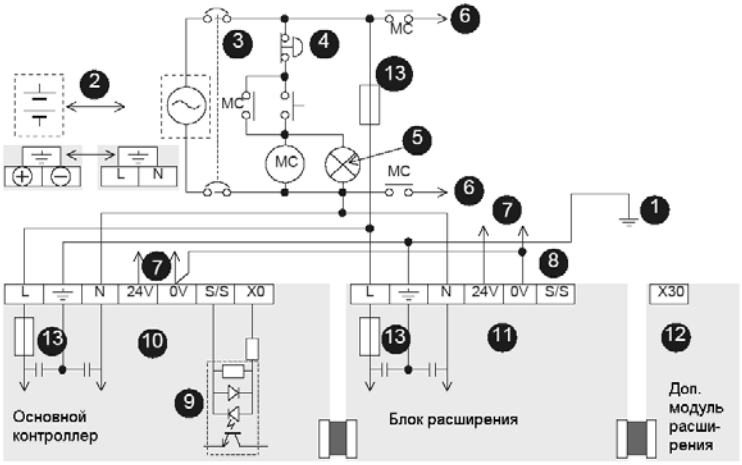
При использовании источника питания постоянного тока необходимо соединить «+» клемму источника питания с «+» клеммой контроллера, а «−» клемму источника питания с «−» клеммой контролера. Ни какое другое подключение недопустимо.

При использовании источника питания переменного тока:

− линия *фазы* должна быть связана с клеммой ***L*** а линия ***0*** с клеммой ***N***. Во избежание поражения электротоком не соединяйте линию *фазы* ***L*** с клеммой ***N***.

− подключение заземления обязательно. Сопротивление заземления должно быть R < 100 Ом

Пример подключения источника питания показан на рисунке 3.3.



1 - Заземление: класс 3;
2 - Источник питания;
3 - Защитное устройство;
4 - Аварийный стоп;
5 - Индикатор наличия питания;
6 - Питание нагрузки;
7 - Встроенный источник питания;
8 - Переключатель типа коммутации входов
9 - Оптопара;
10 - Базовый блок;
11 - Блок расширения;
12 - Модуль расширения (нпр. аналоговый вход);
13 - Предохранитель.(на 3 А)

Рисунок 3.3 – Пример подключения источника питания к ПЛК

ВАЖНЫЕ ЗАМЕЧАНИЯ:

Встроенный источник питания:

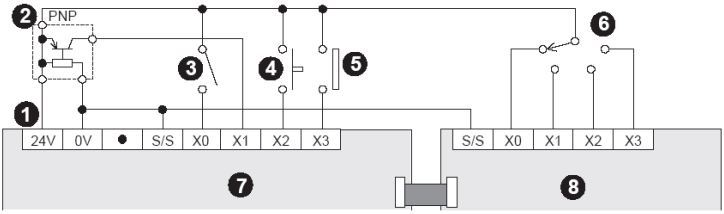
− Если в системе используется встроенный источник питания как базового блока, так и блока расширения, то клеммы «0» должны быть соединены.

− В то же время НЕ СОЕДИНЯЙТЕ клеммы «24 В» базового блока и блока расширения между собой;

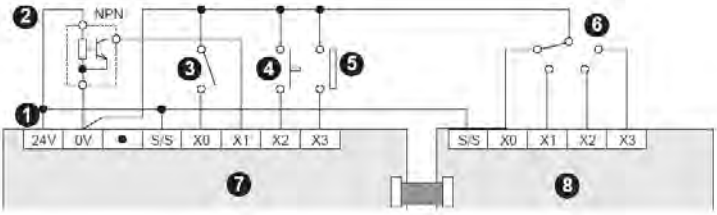
− НИКОГДА не подключайте внешний источник питания к клеммам «24 В».

3.5 Подключение входов

В зависимости от внутренней реализации входов ПЛК может быть два типа подключения: коммутация общим плюсом (source) , либо общим минусом (sink). Пример подключения показан на рисунке 3.4. Многие контроллеры поддерживают оба типа подключения. В нашем примере выбор типа подключения осуществляется клеммой «S/S».



коммутация общим плюсом (source)



коммутация общим минусом (sink)

1 - Источник питания постоянного тока;
2 - Датчик приближения PNP (NPN);
3 - Переключатель;
4 - Кнопка;
5 - Контакт;
6 - Поворотный переключатель;
7 - Базовый блок;
8 - Модуль расширения.

Рисунок 3.4 – Пример подключения входов контроллера

Особенности подключения входов ПЛК к внешним устройствам (например, датчикам) в зависимости от способа их питания показаны на рисунках 3.5...3.7.

Подключение датчика, использующего встроенный в ПЛК источник питания

Подключение датчика, использующего независимый источник питания

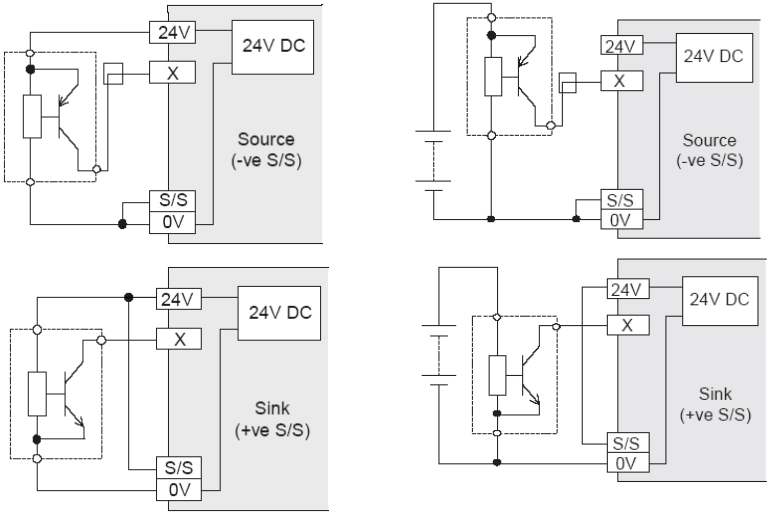


Рисунок 3.5 – Схема подключения датчика к входу ПЛК в зависимости от способа питания датчика и способа коммутации входов ПЛК

Последовательное подключение диодов и входов.

На рисунке 3.6 показано последовательное подключение диода и входа ПЛК. В этом случае при нажатии на кнопку диод загорается. Важно учитывать, что максимальное падение напряжения на диоде – 4 В. Последовательно может быть подключено не более 2 светодиодов.

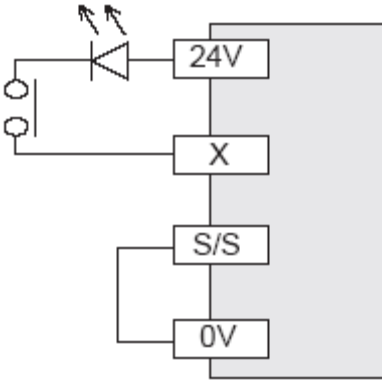


Рисунок 3.6 – Последовательное подключение диода

Также учитывается, что максимальный входной ток для входа ПЛК (например, 8,5 мА), а потребление тока диодом примерно 2,1 мА. При этом необходимо обеспечить корректное срабатывание входа, т.е. значение тока логической «1» и «0». (например, для FX0S: более 4,5 мА и более 1,5 мА для «1» и «0» соответственно, смотри таблицу 2.1).

Параллельное подключение резисторов и входов. Параллельное подключение светодиода

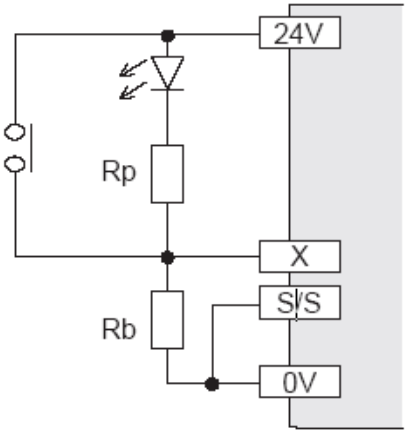


Рисунок 3.7 – Параллельное подключение диода

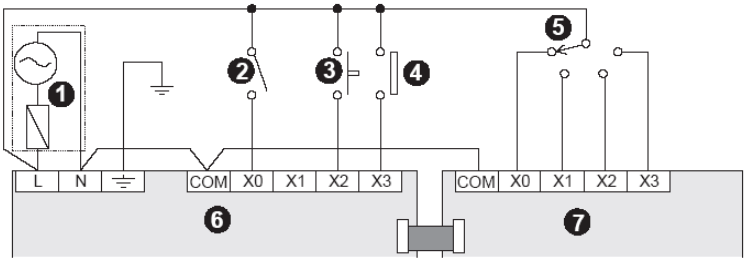
При таком подключении (рисунок 3.7) светодиод при нажатой кнопке погасает. Для сохранения уровня входного тока, поступающего на входы ПЛК, и сохранения работоспособности светодиода необходимо параллельно подключить сопротивление Rp: для FX2N = = 15 кОм. Если сопротивление Rp меньше установленного значения, то добавляется сопротивление Rb. Для определения значения Rb смотрите уравнение

$$Rb \le \frac{4Rp}{15 - Rp}.$$

Ток переключения из состояния логической «1» в «0» для FX2N = 1,5 мА (смотри таблицу 2.1). Если ток больше установленного значения, то ставится дополнительный резистор *Rb* (подтяжка к нулю). Таким образом, вход будет корректно обрабатывать сигнал. Для определения значения Rb смотрите уравнение

$$Rb \le \frac{6}{I - 1,5}.$$

Для закрепления вышеизложенного материала на рисунке 3.8 приведен пример подключения входов контроллера и модуля расширения, работающего от источника переменного тока, к дискретным элементам СУ.



1 - Источник питания переменного тока;
2 - Переключатель;
3 - Кнопка;
4 - Контакт;
5 - Поворотный переключатель;
6 - Базовый блок;
7 - Модуль расширения.

Рисунок 3.8 – Пример подключения входов контроллера

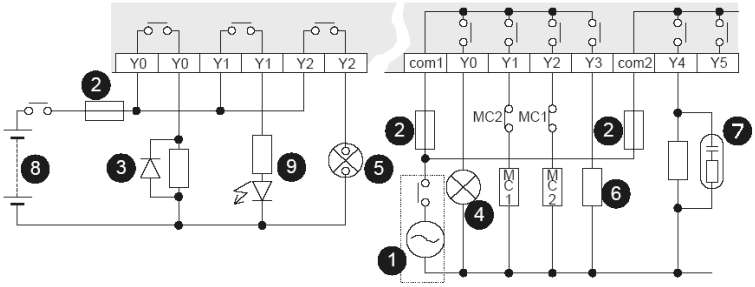
3.6 Подключение выходов

При подключении нагрузки к выходам ПЛК необходимо учитывать следующие параметры ПЛК :

– номинальное напряжение (резистивная нагрузка), В
– номинальный ток / N точек (резистивная нагрузка), А
– максимальная индуктивная нагрузка, ВА
– максимальная активная нагрузка, Вт
– минимальная нагрузка, (необходимая нагрузка при определенном малом напряжении для релейных выходов)
– время срабатывания, мс
– ток утечки (для транзисторных выходов)
– изоляция контура

В зависимости от типа выходов ПЛК подключение осуществляется разными способами. На рисунках 3.9…3.11 приведены соответственно схемы подключения для релейных, транзисторных и симисторных выходов. Следует обратить внимание, что транзисторные

выходы рассчитаны на нагрузку постоянного тока, симисторные – на нагрузку переменного тока, релейные выходы бывают для обоих типов нагрузки (обычно различаются расположением выходных клемм, смотри рисунок 3.9).



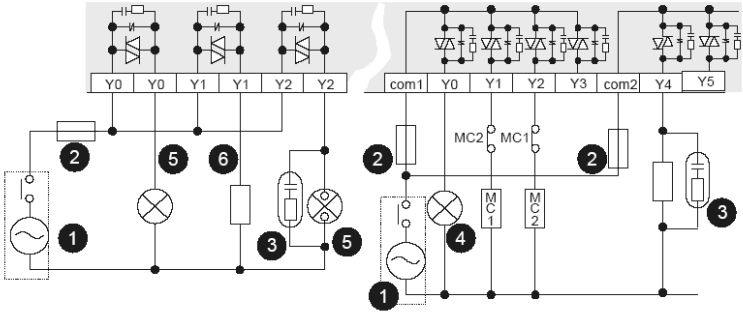
Подключение выхода к нагрузке с источником питания постоянного тока

Подключение выхода к нагрузке с источником питания переменного тока

1 – Источник питания переменного тока;
2 – Предохранитель (1-2 А на каждую группу из 4 входов для защиты выходных цепей ПЛК);
3 – Токоограничивающий диод (продлевает срок службы реле);
4 – Лампа накаливания;
5 – Неоновая лампа;
6 – Резистор;
7 – Цепь помехозащиты (уменьшает шумы при индуктивной нагрузке ~) (C = 0.1 мкФ, R = = 100…120 Ом)
8 – Источник питания постоянного тока;
9 – Светодиод;

Рисунок 3.9 – Схема подключения релейных выходов

Схема подключения симисторных выходов ПЛК показана на рисунке 3.10.



1 – Источник питания переменного тока;
2 – Предохранитель;
3 – Цепочка помехозащиты;
4 – Лампа накаливания;
5 – Неоновая лампа;
6 – Резистор

Рисунок 3.10 – Схема подключения симисторных выходов

Подключение транзисторных выходов также может производиться двумя способами:

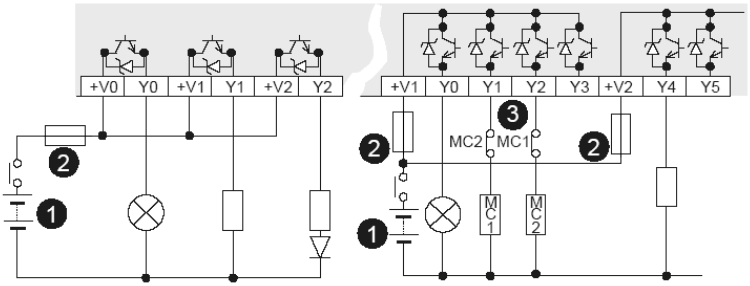
− коммутация общим плюсом (Source);

− коммутация общим минусом (Sink).

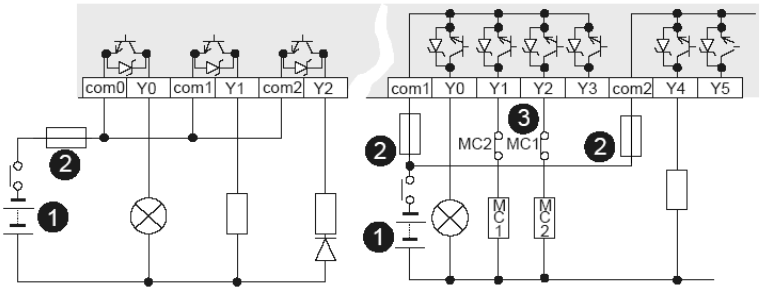
Пример схем таких подключений выходов ПЛК к различного рода нагрузке показан на рисунке 3.11.

Для достижения оптимального времени срабатывания транзисторного выхода рекомендуется использовать в выходных внешних цепях добавочный резистор R, как показано на рисунке 3.12. Значение добавочного резистора выбирается таким, чтобы значение выходного тока было не менее 0,2 А.

Пример схемы подключения источника питания к контроллеру FX0S, а также подключение его входов и выходов к технологическому оборудованию (датчикам, исполнительным механизмам) приведен на рисунке 3.13. В данном примере контроллер управляет движением конвейеров и толкателя, а также подает управляющие сигналы на СУ роботом.



Коммутация общим плюсом (Source)



Коммутация общим минусом (Sink)

1 – Источник питания постоянного тока;
2 – Предохранитель;
3 – Внешняя механическая блокировка логически исключающих друг друга выходов (например, вращение двигателя вперед/назад)

Рисунок 3.11 – Схема подключения транзисторных выходов

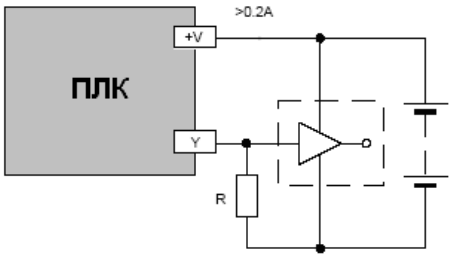


Рисунок 3.12 – Подключение добавочного резистора к выходу ПЛК

Контроллер питается от источника постоянного тока 24 В, входы ПЛК подключены к датчикам по схеме коммутации общим плюсом (source), релейные выходы подключены к нагрузке постоянного тока. Для каждой группы из 4 выходов имеется отдельный предохранитель.

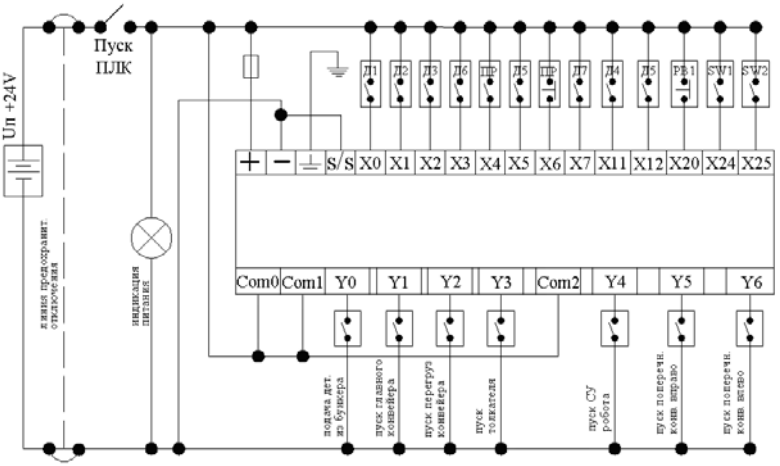
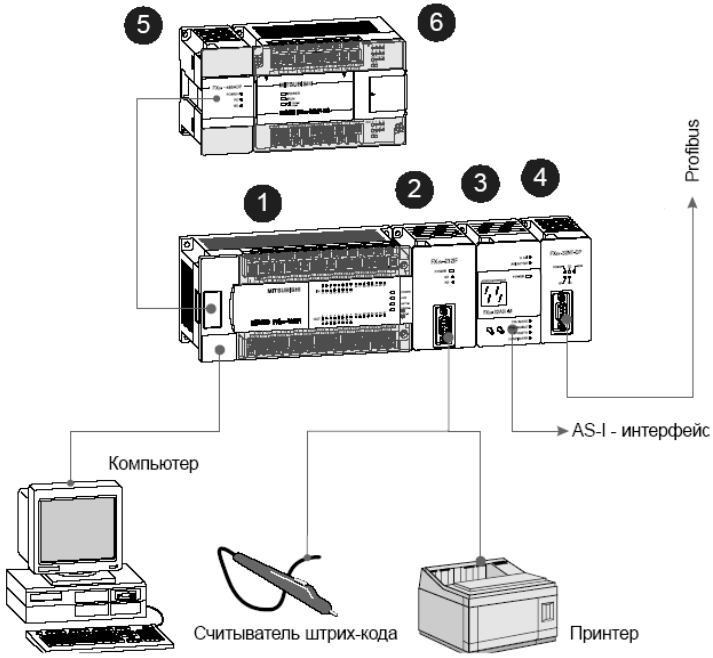


Рисунок 3.13 – Схема подключения ПЛК FX0S к источнику питания и технологическому оборудованию

Глава 4. Конфигурация системы

Часто возникают задачи, требующие расширения функциональности контроллера и построения более мощной и гибкой системы управления с набором специальных возможностей, но на базе одного основного блока. Быстро нарастить и расширить возможности контроллера позволяют дополнительно подключаемые блоки и модули (модули расширения). Пример такой системы показан на рисунке 4.1.



1 - Основной блок
2 - Интерфейсный модуль RS232C
3 - Интерфейсный модуль AS-I
4 - Интерфейсный модуль PROFIBUS DP
5 - Интерфейсный модуль RS485
6 - Ведомый основной блок (slave)

Рисунок 4.1 – Пример многофункциональной расширенной системы управления на базе одного основного блока

4.1 Наращивание количества входов/выходов

Если возникает необходимость задействовать в проекте большее количество входов/выходов, то ПЛК может быть расширен с помощью соответствующих устройств. На примере контроллеров Mitsubishi серии MELSEC это может быть либо компактный блок расширения, модуль расширения, либо адаптеры (рисунок 4.2).

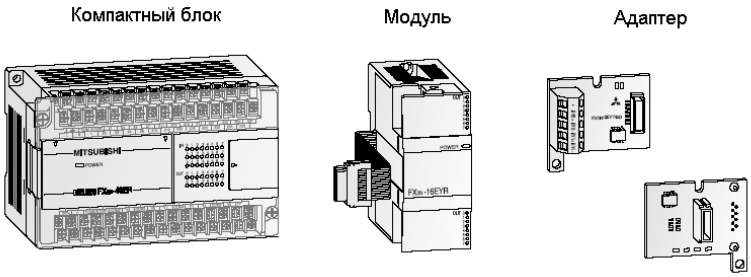


Рисунок 4.2 – Средства наращивания количества входов-выходов

Компактные блоки имеют 32 или 48 входов/выходов (релейные и транзисторные модификации) со светодиодной индикацией состояния входов/выходов и встроенным источником питания (что очень важно при расчете энергопотребления – см. ниже).

Модули расширения имеют 4, 8 или 16 входов/выходов (релейные и транзисторные модификации), также оснащены светодиодной индикацией, однако не имеют собственного источника питания.

Адаптеры расширения на 4 входа и 2 выхода устанавливаются непосредственно в базовые модули контроллеров и не вызывают изменения их габаритных размеров. Адаптеры не занимают адресного пространства контроллера, для входов и выходов предусмотрены отдельные адреса, имеют светодиодную индикацию состояния.

4.2 Модули аналоговых входов/выходов

Аналоговые модули могут оснащаться разным числом входов/выходов (В/В), которые преобразуют цифровые величины контроллеров в аналоговые сигналы (ЦАП), и наоборот, аналоговые сигналы (например, с датчиков температуры или давления) – в цифровые величины для дальнейшей обработки контроллером (АЦП). Наряду с этим имеются готовые модули со встроенной термопарой и ПИД-регулятором для управления температурой, имеющие определенное число каналов, управляемых раздельно. Модули аналоговых В/В показаны на рисунке 4.3.

Аналоговый адаптер расширения представляет собой ЦАП и АЦП с одним входом/выходом и может выдавать сигнал как по току, так и по напряжению. Устанавливается в слот расширения базового модуля, дополнительное питание не требуется.

Рисунок 4.3 – Средства для обработки аналоговых величин

4.3 Модули позиционирования

1-осевые или 2-осевые модули позиционирования представляют собой модули с импульсным выходом, обеспечивающими позиционирование шаговых и сервоприводов (рисунок 4.4). Такие модули обладают высокой гибкостью за счет реализации различных режимов работы, возможности позиционирования в абсолютных и относительных координатах и т.п.

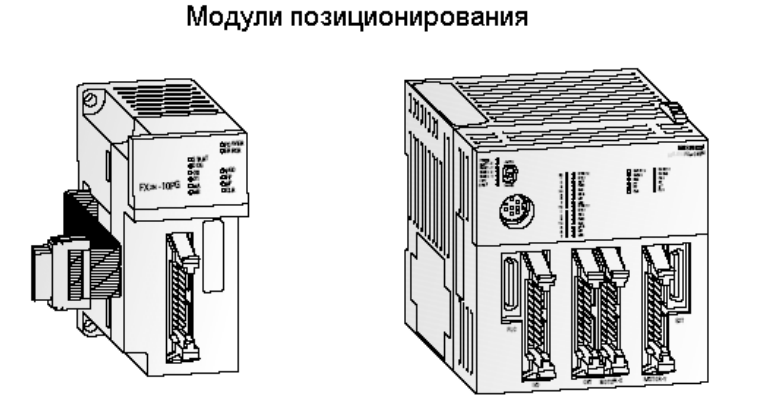


Рисунок 4.4 – Модули расширения ПЛК
для позиционирования электроприводов

4.4 Аппаратные средства программирования

Все процессоры имеют стандартизированный программный интерфейс для присоединения к программатору или к персональному компьютеру (рисунок 4.5).

Простые проекты могут быть запрограммированы непосредственно с ручного программатора или модульной панели управления. Кроме того, процессоры могут программироваться с помощью специальных программных пакетов, которые запускаются на обычном ПК. Это, как правило, мощная оболочка для создания больших проектов удовлетворяющих стандарту МЭК 1131-3, более подробно данная тема будет рассмотрена в разделе 8 «Языки программирования. Пакеты ПО».

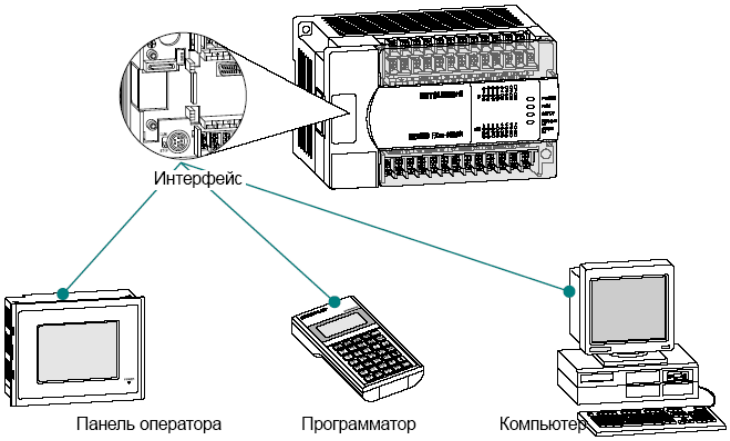


Рисунок 4.5 – Аппаратные средства программирования ПЛК через встроенный интерфейс

4.5 Средства визуализация процесса

Для отображения технологического процесса на панели оператора используются специальные системы визуализации, которые позволяют контролировать протекание процесса, а также реализуют удобную для человека систему оповещения о тревогах, ведут учет различных параметров ТП. Существуют как аппаратные, так и про-

граммные решения с частичной или полной графической поддержкой (рисунок 4.6). Средства визуализации могут включать в себя различные операторские терминалы, а также мощные программные пакеты визуализации типа SCADA (обзор возможностей SCADA смотри в приложении Б).

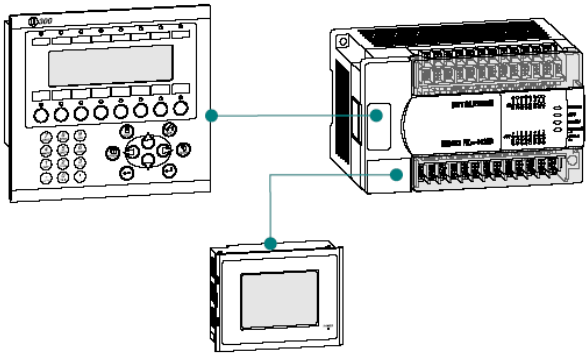


Рисунок 4.6 – Средства визуализации процесса

4.6 Коммуникационные модули

Для подключения различного рода устройств обмена данными, а также построения различных типов сетей имеются соответствующие коммуникационные модули под необходимый интерфейс (рисунок 4.7).

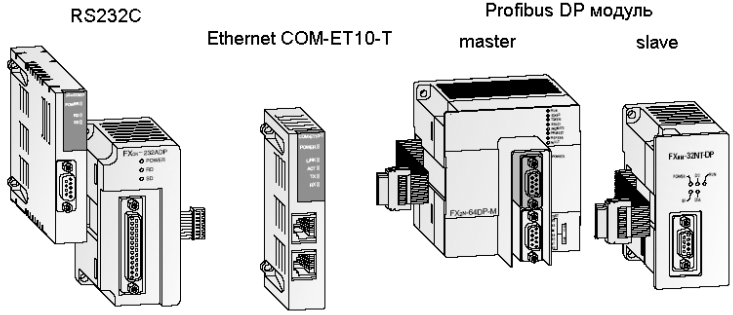


Рисунок 4.7 – Коммуникационные модули

В современном производстве часто возникает необходимость объединения промышленного оборудования в единую сеть, подобно офисным и домашним компьютерным сетям со всеми из этого вытекающими преимуществами, эффективностью и удобством. Промышленная сеть – Fieldbus (от англ. – полевая шина) объединяет в себе различное технологическое оборудование: промышленные компьютеры, контроллеры, датчики, частотные приводы и т.п. Такая сеть должна отвечать специфическому набору требований:

− жесткая детерминированность (предсказуемость) поведения;

− обеспечение функций реального времени;

− работа на длинных линиях с использованием недорогих физических сред (например, витая пара);

− повышенная надежность физического и канального уровней передачи данных для работы в промышленной среде (например, при больших электромагнитных помехах);

− наличие специальных высоконадежных механических соединительных компонентов.

На сегодняшний день имеется несколько наиболее используемых стандартов промышленных сетей. Как выбрать именно то, что нужно часто становится некоторой проблемой. Предпочтительность того или иного сетевого решения как средства передачи данных можно оценить по следующей группе критериев:

− объем передаваемых полезных данных;

− время передачи фиксированного объема данных;

− удовлетворение требованиям задач реального времени;

− максимальная длина шины;

− допустимое число узлов на шине;

− помехозащищенность;

− денежные затраты в расчете на узел.

Часто улучшение по одному параметру может привести к снижению качества по другому, то есть при выборе того или иного протокольного решения необходимо следовать принципу разумной достаточности. В зависимости от области применения весь спектр промышленных сетей можно разделить на два уровня (таблица 4.1):

− Field level (промышленные сети этого уровня решают задачи по управлению процессом производства, сбором и обработкой данных на уровне промышленных контроллеров);

− Sensor/actuator level (задачи сетей этого уровня сводятся к опросу датчиков и управлению работой разнообразных исполнительных механизмов).

Таблица 4.1 – Стандарты промышленных сетей

Типичные открытые сенсорные (датчиковые) сети:	Типичные открытые сети для обоих уровней применения:
ASI (Actuator/Sensor Interface) ControlNet Interbus PROFIBUS-DP (Profibus for Distributed Periphery)	CAN (Controller Area Network) DeviceNet FIPIO (Factory Instrumentation Protocol) Interbus LON (Local Operating Network) Modbus Plus, RTU & TCP PROFIBUS (Process Field Bus) Profinet

Основные технические характеристики наиболее распространенных стандартов промышленных сетей приведены в приложении А.

На рисунке 4.8 представлена обобщенная сетевая структура, показывающая в общем виде возможное использование того или иного протокола на определенных уровнях условного промышленного предприятия.

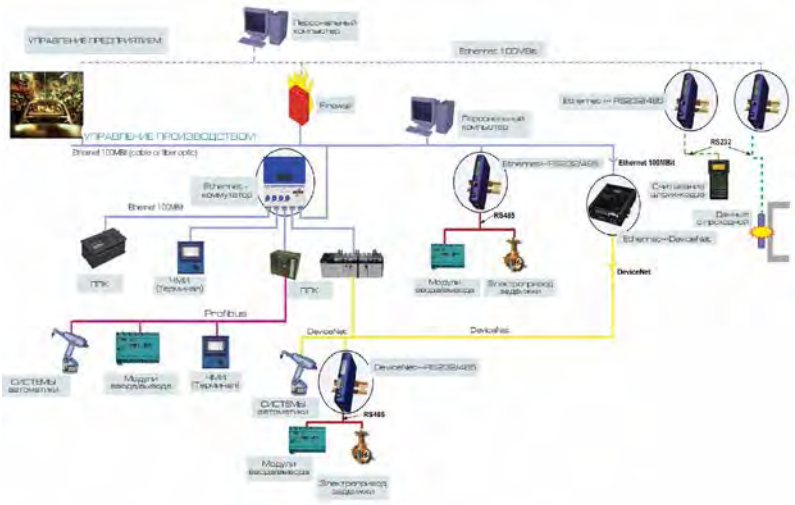


Рисунок 4.8 – Сетевая структура технологического оборудования предприятия

Глава 5. Расчет энергопотребления

При построении системы с использованием дополнительных модулей расширения, необходимо производить расчет энергопотребления по шинам питания контроллера.

Расчет потребления энергии с 5 В шины модулями специального назначения осуществляется с учетом энергопотребления этих модулей и максимального допустимого тока нагрузки основного контроллера. Эти значения приводятся в технической документации к каждому конкретному модулю.

Например, для FX2N-16MR-UA1/UL выдаваемый ток источником питания по 5 В шине равен 290 мА, а потребление подключаемого модуля позиционирования FX2N-1PG-E по 5 В шине равно 60 мА. Каждый тип модулей расширения имеет свое значение потребляемого тока, которое приводится в технической документации на данный модуль расширения.

Значение остаточного тока от сервисного источника ровно 24 В**,** в зависимости от количества подключенных дополнительных входов/выходов (мА) показано в таблице 5.1.

Таблица 5.1 – Значение тока с 24 В источника, в зависимости от количества подключенных дополнительных Входов/Выходов

Количество дополнительных выходов	**48**	10								
	40	85	35							
	32	160	110	60	10					
	24	235	185	135	85	35				
	16	310	260	210	160	110	60	10		
	8	385	335	285	235	185	135	85	35	
	0	460	410	360	310	260	210	160	110	60
		0	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**60**
		Количество дополнительных входов								

Модули специального назначения должны быть запитаны от дополнительного источника, если их суммарное энергопотребление больше допустимой нагрузки тока контроллера.

Пример расчета допустимой нагрузки

Проектируемая система управления состоит из основного вычислительного блока FX2N-80MR-ES, к которому подключены модули расширения, а именно: модуь АЦП FX2N-4AD, модуль ЦАП FX2N-4DA и модуль сетевого взаимодействия по протоколу RS232C–

FX2N-232IF. Таблица 5.2 показывает пример расчета энергопитания ПЛК-системы. Значения энергопотребления для модулей специального назначения взяты из спецификаций к каждому конкретному модулю. Сравнение с максимальным значением нагрузки (+460 мА и +290 мА для 24 В шины и 5 В шины соответственно), приведенным в таблице, показывает, что нагрузка на 5В шину лежит в допустимой области (за вычетом потребления тока всеми блоками остаток тока по шине +100 мА). Однако, как показывает расчет энергопотребления с 24 В шины, все блоки должны быть запитаны от внешнего источника питания напряжением 24 В (остаток тока меньше 0 (-170 мА)).

Таблица 5.2 – Пример расчета допустимой нагрузки ПЛК-системы

Модуль	**Кол-во модулей**	**По шине 24 В**		**По шине 5 В**	
		Ток (мА)	**Итого (мА)**	**Ток (мА)**	**Итого (мА)**
FX2N-80MR-ES (ПЛК)	1	460	+460	+290	+290
FX2N-4AD (АЦП)	3	50	-150	30	-90
FX2N-4DA (ЦАП)	2	200	-400	30	-60
FX2N-232IF (RS232C)	1	80	-80	40	-40
Результат			**-170**		**+100**

Помимо расчетов значений тока по шинам питания, необходимо учитывать ограничение адресного пространства ПЛК – т.е. то максимальное количество входов/выходов, которое он в состоянии обрабатывать (для FX2N – 256). Также ПЛК может иметь ограничение на количество подключаемых дополнительных модулей (например, для FX2N – до 8).

Глава 6. Вопрос выбора ПЛК

6.1 Из чего выбирать

Приведем краткую оценку объема рынка контроллерных средств. Лидируют на нем такие компании, как: ABB (распространяющая также контроллерные средства фирм Baily Controls и Gartman & Braun), Emerson, General Electric Fanuc Automation, Foxboro, Honeywell, Metso Automation, Mitsubishi Electric, Moore Products, Omron, Rockwell Automation, Siemens, Yokogawa, Schneider Automation и др. Всего порядка 15 фирм, каждая из которых предлагает от двух до пяти контроллерных средств разных классов.

Около 20 западных и азиатских производителей меньшего масштаба имеют на местных рынках своих дилеров, внедряющих их контроллерные средства на предприятиях (Koyo Electronics, Tornado, Triconex, PEP, Trey, Control Microsystems, GF Power Controls и др.).

Более 20 российских предприятий конкурируют с зарубежными производителями в разных классах контроллерных средств («Автоматика», «ДЭП», «Импульс», «Инсист Автоматика», «Интеравтоматика», «Квантор», «НИИ-теплоприбор», «НВТ-Автоматика», «ПИК», «Прогресс», «Саргон», «Системотехника», «ТЕКОН», «Электромеханика», «ЭМИКОН» и др.).

Спектр продукции, предлагаемой сегодня, чрезвычайно широк. Но как же выбрать нужный контроллер, если контроллеры имеют равные функциональные возможности, близкие технические и эксплуатационные характеристики и даже почти одинаковые размеры (рисунок 6.1). В такой ситуации необходимо определить критерии оценки и выбора ПЛК, удовлетворяющего поставленной задаче.

Рисунок 6.1 –Контроллеры одного функционального назначения различных производителей

6.2 Как выбирать

Учитывая специфику устройств, критерии оценки можно разделить на три группы (рисунок 6.2):

- технические характеристики;
- эксплуатационные характеристики;
- потребительские свойства.

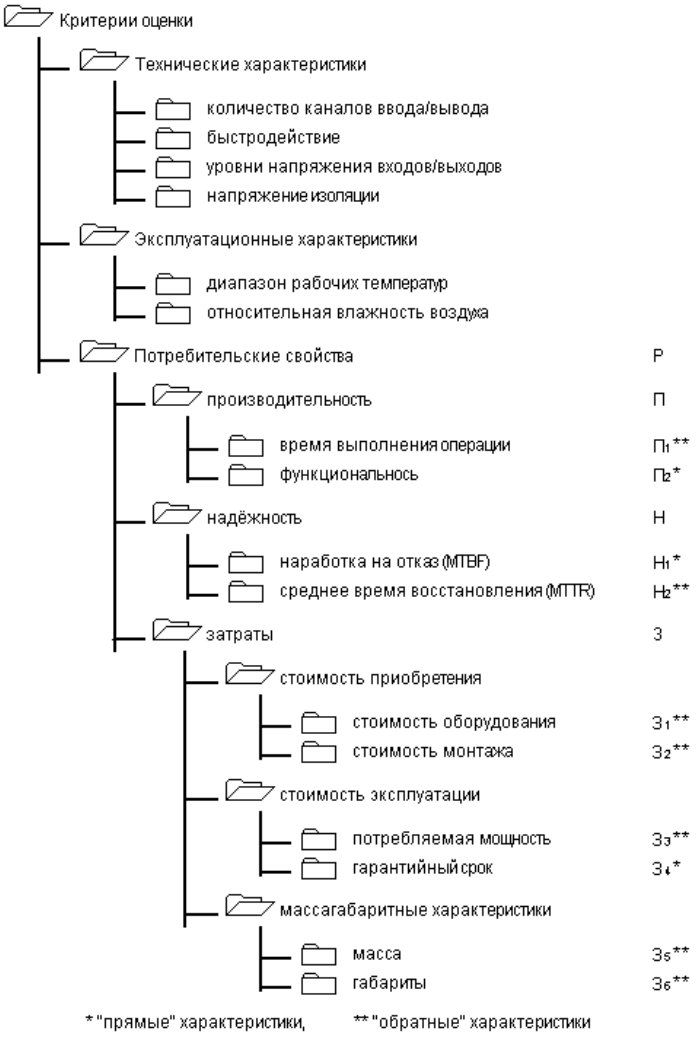


Рисунок 6.2 – Критерии оценки ПЛК

При этом критериями выбора можно считать потребительские свойства, т.е. соотношение показателей затраты/производительность/надежность, а технические и эксплуатационные характеристики – ограничениями для процедуры выбора.

Кроме того, необходимо разделить характеристики на прямые (для которых положительным результатом является её увеличение) и обратные (для которых положительным результатом является её уменьшение).

Так как характеристики между собой конфликтны, т.е. улучшение одной характеристики почти всегда приводит к ухудшению другой, необходимо для каждой характеристики Ki (Р, П, $П_1$, $П_2$, Н и т.д. , смотри рисунок 6.2) определить весовой коэффициент a_i, учитывающий степень влияния данной характеристики на полезность устройства, другими словами – поставить ей оценку.

Терминология и состав критериев оценки ПЛК приведены в соответствии с основными положениями квалиметрии и стандартами качества (ГОСТ 15467-79).

Выбор аппаратуры производится в четыре этапа (рисунок 6.3):

1) определение соответствия технических характеристик предъявленным требованиям;

2) определение соответствия эксплуатационных характеристик предъявленным требованиям;

3) оценка потребительских свойств выбираемой аппаратуры;

4) ранжирование изделий.

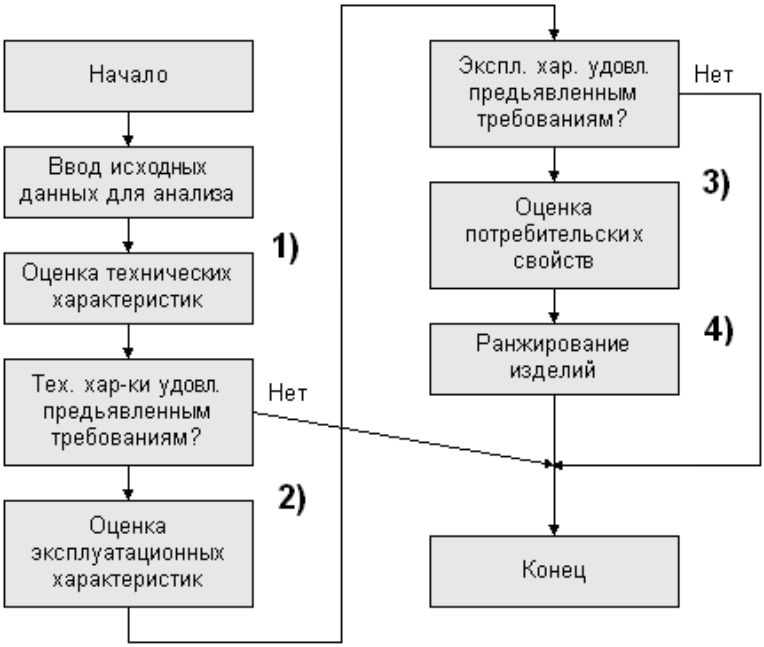


Рисунок 6.3 – Алгоритм выбора ПЛК

На первом этапе каждая техническая характеристика анализируемого изделия сравнивается с предъявленными к проектируемой системе требованиями, и если данная характеристика не удовлетворяет этим требованиям, изделие снимается с рассмотрения.

Такой же анализ проводится на втором этапе с эксплуатационными характеристиками, и только если технические и эксплуатационные характеристики соответствуют поставленной задаче и предъявленным требованиям, проводится оценка потребительских свойств ПЛК. Для этого используется аддитивный метод оценки, когда суммарная оценка каждого свойства вычисляется по следующей формуле

$$K = \sum_{i=1}^{n} \frac{K_i}{K'_i} \alpha_i + \sum_{i=1}^{m} \frac{1}{\frac{K_j}{K'_j}} \alpha_j \ ,$$

где K_i , K_j – прямая и обратная характеристики выбираемого изделия;

K'_i , K'_j – соответствующие характеристики аналога;

α_i , α_j – весовые коэффициенты характеристик;

n, m – количество прямых и обратных характеристик.

Деление на характеристики аналога необходимо для приведения всех свойств к относительным величинам.

Определение весовых коэффициентов для характеристик ПЛК является одной из самых ответственных задач, т.к. именно от их правильной величины зависит достоверность результатов анализа. Для нахождения усредненной оценки каждого коэффициента может быть рекомендована следующая методика экспертных оценок и программа их расчета.

Программа расчета по методике экспертных оценок:

1) Составляется матрица эксперты-коэффициенты (таблица 6.1), в которой проставляются полученные от каждого эксперта оценки коэффициентов по шкале от 0 до 10.

Таблица 6.1 – Матрица эксперты - коэффициенты

	Коеффициент					
Эксперт	**1**	**2**	**3**	**4**	**5**	**6**
1						
2						
…						
j						
…						
m						

2) Рассчитывается относительная значимость (W_{ij}) всех коэффициентов в отдельности для каждого эксперта. С этой целью оценки, полученные от каждого эксперта, суммируются (по горизонтали), а затем нормируются:

$$W_{ij} = \frac{\alpha_{ij}}{\sum_{i=1}^{n} \alpha_{ij}}, \text{ при } j = \text{const},$$

3) Вычисляется усредненная оценка, данная всеми экспертами каждому коэффициенту. Для этого нормированные оценки, полученные в предыдущем шаге, суммируются (по вертикали), а затем рассчитывается среднее арифметическое для каждого коэффициента:

$$\alpha_i = \frac{\sum_{j=1}^{m} W_{ij}}{m}, \text{ при } i = \text{const};$$

4) В результате анализа потребительских свойств аппаратуры составляется матрица изделия-потребительские свойства (таблица 6.2), которая содержит исходные данные для выбора ПЛК.

Таблица 6.2 - Матрица изделия - потребительские свойства

	Потребительские свойства		
Изделия	**П**	**Н**	**З**
1			
2			
…			
j			
…			
N			

Ранжирование изделий, т.е. расположение их в порядке возрастания (или убывания) соотношения показателей затраты/производительность/надежность целесообразно проводить по формуле

$$P = П + Н + З.$$

Необходимо отметить, что применение данной методики допускает варьирование характеристик в зависимости от конкретной ситуации. Это может быть обусловлено как объективными, так и субъективными причинами.

Проведенный анализ не претендует на полноту охвата всех показателей в основном по субъективным причинам. Однако даже в таком виде можно сделать вывод о том, что данная методика позволяет провести оценку и принять решение о выборе ПЛК с достаточно высокой степенью достоверности.

РАЗДЕЛ ВТОРОЙ.

ПРОГРАММИРОВАНИЕ КОНТРОЛЛЕРА

Глава 7. Основы программирования ПЛК. Реле и контроллер

Теперь, когда мы имеем представление об основных принципах работы ПЛК, как он обрабатывает входы, выходы, и как выполняется программа, мы готовы начать писать программу. Но для начала вспомним о том, как работает реле, и что реле из себя представляет, так как целью использования контролера и является физическое замещение реле.

В принципе реле – это выключатель, подобно тем выключателям, которыми мы пользуемся повседневно, включая-выключая свет в комнатах. Только, если дома мы используем физическую силу для включения-выключения, то промышленное реле использует электромагнитное поле. При подачи напряжение на катушку возбуждается магнитное поле – это магнитное поле притягивает контакты реле, следствием чего является соединение двух контактов. В сущности, основное назначение реле – дать возможность току протекать между двумя контактами.

Давайте рассмотрим простой пример, в котором необходимо включить электродвигатель, при нажатии на выключатель вход (X1).

В результате мы получим устройство (рисунок 7.1), которое можно условно разделить на три основные части:

- выключатель вход (X1);
- реле;
- электродвигатель.

Всякий раз, когда выключатель вход (X1) соединяет контакты (включается), цепь замыкается и ток, создавая электромагнитное поле в реле, переключает контакты реле, т.е. пускает электродвигатель.

В данном примере использовано типичное индустриальное реле постоянного тока для управления включением-выключением устройств. Пока вход (X1) открыт, ток не может течь через катушку реле и электродвигатель не работает. Но как только вход (X1) за-

крыт, ток создает электромагнитное поле, которое заставляет контакты реле соединиться. В результате ток, протекающий через контакты реле, вращает электродвигатель .

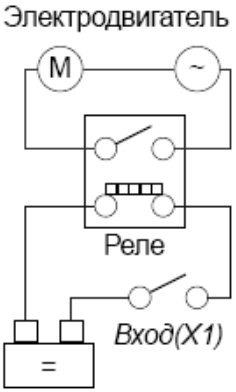


Рисунок 7.1 – Пример использования реле

Попробуем заменить реле контроллером. Конечно, данный пример не является показателем эффективности использования ПЛК в категории цена-производительность. Но он позволит понять, в каких случаях и для чего можно эффективно использовать контроллер. Первое, что необходимо осознать, каким образом мы можем объяснить ПЛК то, какую задачу он должен выполнить.

Одним из базовых языков программирования ПЛК остается язык *Релейно-контактных схем*. (наиболее удобный для понимания сути в нашем примере)

Шаг первый:

Необходимо переопределить все составляющие оборудования, которые мы используем, в символы, понятные для контроллера, так как ПЛК ничего не знает о существовании таких вещей, как выключатель, реле, электродвигатель и т.д. ПЛК может работать с символами вход, выход. Для контроллера совершенно не важно, что из себя физически представляет вход или выход. контроллер обрабатывает только текущее состояние входа (включено-выключено). Все остальные действия выполняются последовательно и только в строгом соответствии с алгоритмом, заложенным в контроллер программой, – т.е. вами.

Шаг второй:

Сначала заменяем источник питания. Для языка *Релейно-контактных схем* - этим символом будет являться две параллельные прямые с левой и правой стороны диаграммы. Можно считать, что левая линия является «+», а правая линия «–».

Шаг третий:

Присваиваем символы входам. В нашем примере имеем два входа:

Вход (X1) – нормально открытый контакт;

Вход (X2) – нормально закрытый контакт.

Шаг четвертый:

присваиваем символ выходам.

Выход (Y1) – символ катушки реле.

В результате мы получили программу, которая может быть выполнена ПЛК (рисунок 7.2).

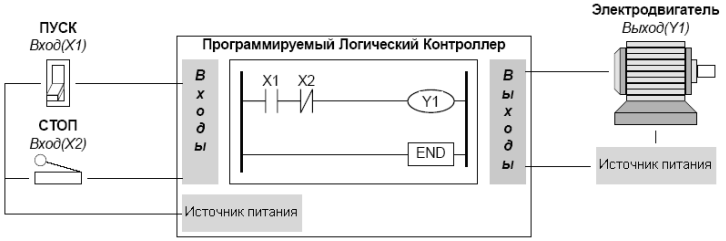


Рисунок 7.2 – Пример программы для ПЛК

Глава 8. Языки программирования, пакеты ПО

В 1993 году Международной Электротехнической Комиссией (МЭК) был опубликован стандарт МЭК 61131-3 (IEC 61131-3). Этот международный стандарт входит в группу стандартов МЭК 1131, которые охватывают различные аспекты использования программируемых логических контроллеров. Назначение МЭК 61131-3 – стандартизация существующих языков программирования ПЛК, а, вер-

нее, базовая платформа для такой работы в национальных комитетах стандартизации.

Стандарт МЭК 61131-3 оказался настолько актуален, что ждать его адаптации не хватило сил: функции поддержки и внедрения стандарта на рынке взяла на себя независимая организация PLCOpen, состоящая из производителей и пользователей программного обеспечения (ПО), ориентированного на МЭК 61131-3. В результате деятельности PLCOpen на рынке ПО появилась серия сертифицированных средств программирования ПЛК,– средств, которые достаточно широко и небезуспешно внедряются в промышленности.

Список инструментальных программных систем, реализующих стандарт МЭК 61131-3, превышает два десятка (таблица 8.1).

Таблица 8.1 – Примеры инструментальных систем реализующих МЭК 61131-3

Название программного пакета	Производитель
CoDeSys	Smart Software Solutions, Германия
ACCON-ProSys	Deltalogic, Германия
OpenDK	Infoteam Software, Германия
PUMA	KEBA, Австрия
SUCOsoft S340	Klonker-Moeller, Германия
NAIS CONTROL	Matsushita AC, Германия
PDS7	Philips, Нидерланды
SELECONTROL	Selection Lyss, Швейцария
Soft Control	Softing, Германия
ISaGRAF	CJ International, Франция

Глава 9. Организация PLCopen и уровни совместимости

Общую координацию деятельности производителей и пользователей «61131»-систем (инструментальные системы, реализующие стандарт МЭК 61131-3 на пять языков программирования контроллеров) осуществляет независимая международная Ассоциация PLCopen. Она занимается популяризацией и информационной поддержкой стандарта МЭК 61131-3 с целью его использования в промышленных системах контроля и управления.

В задачи PLCopen не входит поддержка разработки универсального инструмента программирования для любого типа контроллеров. Основное направление деятельности ассоциации – поддержка некоторого множества языков программирования, использование которых позволит пользователям различных контроллеров обмениваться своими наработками.

Одна из важнейших задач PLCopen – выработка системы и принципов сертификации программных продуктов на предмет их соответствия стандарту МЭК 61131-3. PLCopen определяет три уровня совместимости инструментальных систем (рисунок 9.1):

− Базовый уровень (Base Level)
− Уровень переносимости функций (Portability Level)
− Уровень переносимости приложений (Application Level)

Базовый уровень предполагает, что системы должны быть совместимы на некотором подмножестве базовых компонентов, определяемых стандартом (типы переменных, языковые конструкции и т. п.). Это первый этап, который должны пройти разработчики контроллеров с возможностями подключения к «61131»-системе.

Следующий уровень совместимости определен как уровень переносимости функций и функциональных блоков между различными системами. Для этой цели был введен специальный формат файла обмена.

Последний, третий уровень, определяет степень совместимости и переносимости на уровне завершенных приложений.

Хорошо проработаны и ведутся процедуры сертификации инструментальных «61131»-систем по первым двум уровням. Спецификация по третьему уровню совместимости (а это мечта любого системного интегратора) пока разрабатывается.

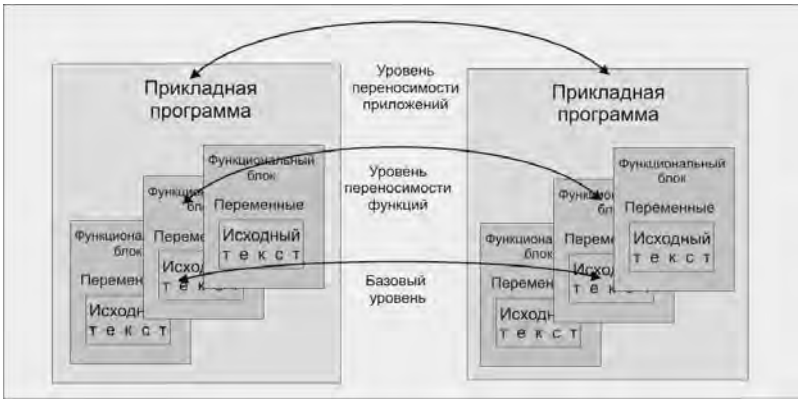


Рисунок 9.1 – Уровни совместимости «61131» – систем

Глава 10. Классификация языков по стандарту МЭК 61131-3

Стандарт МЭК 61131-3 определяет языки для программируемых контроллеров таким образом, что части прикладной программы могут быть запрограммированы на любом языке и скомпонованы в единую исполняемую программу. При разработке стандарта было найдено так много вариаций языков для программируемых контроллеров, что было невозможно выбрать одну из существующих вариаций в качестве общего языка. Новый стандарт включает структурное программирование, абстрактные типы данных, выделение данных и процедур в блок (инкапсуляцию) в сочетании с сохранением тесной связи с классическими языками для программируемых контроллеров.

Опыт показывает, что выбор используемых языков чаще всего определяется личными предпочтениями пользователей и мало связан с автоматизируемым технологическим процессом. Действительно, представленные в стандарте языки в большинстве случаев взаимозаменяемы. Это значит, что при разном уровне подготовленности в области чистого программирования пользователи могут создавать программы равной функциональности.

Языки программирования стандарта МЭК 61131-3 включают в себя 3 визуальных языка (FBD, SFC, LD), ориентированных на ин-

женеров и бизнес-аналитиков и 2 текстовых (ST, IL), ориентированных на программистов.

10.1 Язык релейно-контактных схем (LD)

Язык релейных диаграмм, или релейной логики (Ladder Diagrams) применяется для описания логических выражений различного уровня сложности и использует в качестве базовых элементов программирования графические элементы «контакты» (contacts) и «катушки» (coils), связанные с входными и выходными каналами соответственно (рисунок 10.1)

Присутствие в стандарте языка LD определяется, скорее всего, данью традициям: для релейной техники было разработано огромное количество оборудования и алгоритмов. Сегодня, имея типовой набор цифрового ввода/вывода, можно создавать управляющие системы на отлаженной годами алгоритмической базе.

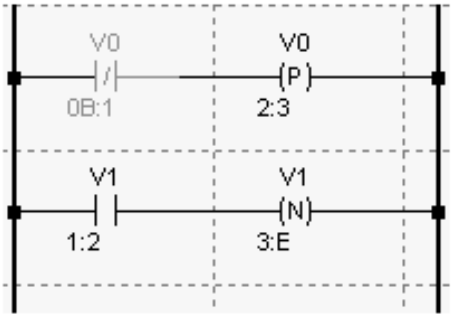


Рисунок 10.1 – Пример кода программы на языке релейно-контактных схем

10.2 Язык последовательных функциональных схем (SFC)

Язык последовательных функциональных схем (Sequential Function Charts, или Grafcet) позволяет формулировать логику программы на основе чередующихся процедурных шагов и транзакций (условных переходов), а также описывать последовательно-параллельные задачи в понятной и наглядной форме (рисунок 10.2).

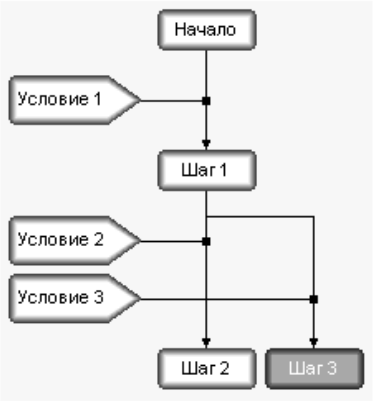


Рисунок 10.2 – Пример кода программы на языке последовательных функциональных схем

Строго говоря, SFC не является языком программирования. Это средство проектирования прикладного программного обеспечения, которое всегда является комплексом большого числа программных единиц: программ, функциональных блоков, функций. Обеспечение параллельности выполнения программ, установление и контроль состояния порожденных процессов, обеспечение синхронизации по приему и обработке данных, описание однозначно понимаемых и заказчиком, и исполнителем состояний автоматизируемого процесса – все это возможно при использовании языка программирования SFC.

Основные достоинства SFC можно определить следующим образом:

− Высокая выразительность. Язык SFC имеет те же возможности, что и диаграммы состояний, и является наиболее подходящим средством для описания динамических моделей.

− Графическое представление. Благодаря графической мнемонике SFC максимально прост в использовании и изучении. Вместе с тем, он является наглядным средством представления логики на разных уровнях детализации.

− Предварительное проектирование ПО. Использование языка SFC на ранних этапах проектирования прикладного ПО позволяет снять многочисленные непонимания между заказчиком, проектировщиком ПО и программистом.

10.3 Язык функциональных блоков (FBD)

Язык функциональных блоков (Function Block Diagrams) позволяет создать программную единицу практически любой сложности на основе стандартных кирпичиков (арифметические, тригонометрические, логические блоки, ПИД-регуляторы, блоки, описывающие некоторые законы управления, мультиплексоры и т.д.). На рисунке 10.3 показан пример типичных элементов. Это языковое средство использует технологию инкапсуляции алгоритмов обработки данных и законов регулирования. Все программирование сводится к «склеиванию» готовых компонентов. В результате получается максимально наглядная и хорошо контролируемая программная единица.

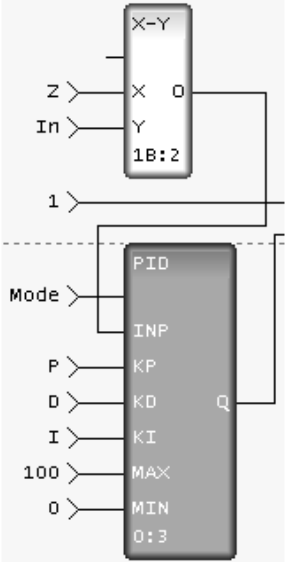


Рисунок 10.3 – Пример кода программы на языке функциональных блоков

10.4 Язык списка инструкций (IL)

В «достандартные» времена (до 1993 года) практически каждый программируемый контроллер сопровождался своим Ассемблером. Выросли целые поколения программистов, ориентированных на определенные кланы микропроцессоров. Освоение новой техники сталкивалось с проблемой освоения очередного языка программирования под новый кристалл. Отдельные мнемонические конструкции Ассемблеров были похожи, но о каком-либо стандарте не было и речи.

Появление языка инструкций (Instruction List) в наборе стандартных языков – это унификация интерфейса языка программирования низкого уровня, неориентированного на какую-либо микропроцессорную архитектуру. У языка IL есть очень важное качество: на его основе можно создавать оптимальные по быстродействию программные единицы. Пример участка кода на языке инструкций приведен на рисунке 10.4.

```
ADD VAR_000 2.6
LT VAR_000 VAR_001
JMPC label1
GT VAR_001 20
JMPC label2
LD 278
label1:
CAL FUNCTION1(VAR_000, 3)
label2: ST VAR_001
```

Рисунок 10.4 – Пример кода программы на языке списка инструкций

10.5 Язык структурированного текста (ST)

Язык структурированного текста (Structured Text) относится к классу текстовых языков высокого уровня (пример кода программы на рисунке 10.5). Этот язык уходит корнями в такие известные языки программирования, как Ada, Pascal и C. На его основе можно создавать гибкие процедуры обработки данных. Язык структурированного текста является основным для программирования последовательных

шагов и транзакций языка SFC. Кроме этого, он имеет «выходы» во все остальные языки, что делает его универсальным в применении разными категориями пользователей.

```
PROGRAM
    VAR_INPUT ARG_000 : REAL; END_VAR
    VAR_INOUT ARG_001 : REAL; END_VAR

WHILE ARG_000 > 2 DO ARG_000 = ARG_000-1;
ARG_001 = ARG_001 + 2;
END_WHILE;

END_PROGRAM
```

Рисунок 10.5 – Пример кода программы на языке структурированного текста

В многих инструментальных «61131»-системах существует возможность «смешивать» программы/процедуры, написанные на разных языках, а также вставлять кодовые последовательности из одного языка в коды, написанные на другом языке. Любое функциональное расширение возможно за счет поддержки СИ-интерфейса, которая считается сегодня обязательной.

Рисунок 10.6 демонстрирует место каждого из языков на различных этапах разработки прикладного ПО.

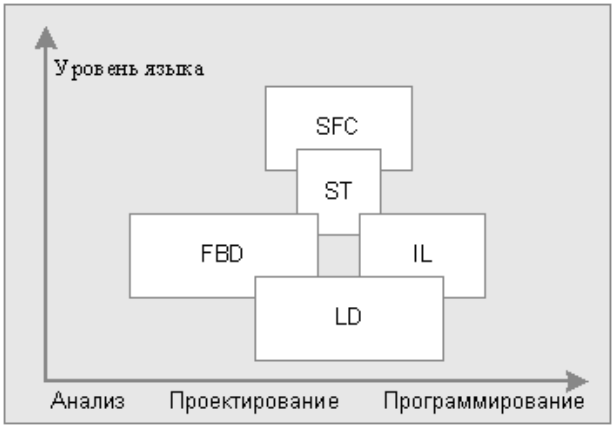


Рисунок 10.6 – Место языков на этапах разработки программы для ПЛК

В итоге МЭК 61131-3 (IEC 61131-3) содержит:

Богатый набор стандартных функций:

− Функции для работы с битовыми строками;
− Числовые функции;
− Функции преобразования типов;
− Функции выбора;
− Функции сравнения;
− Функции, определяемые производителем и пользователем.

Функциональные блоки:

− Блоки синхронизации состояний;
− Блоки для дифференцирования переднего и заднего фронта;
− Счетчики;
− Таймеры;
− Функциональные блоки, определяемые пользователем и производителем.

С помощью этих функций и блоков выполняется множество операций над большим количеством стандартных типов данных, таких как битовые строки, целые, беззнаковые целые, вещественные, временные, строки символов, массивы, структуры, а также типы данных, определяемые производителем и пользователем.

Глава 11. Язык релейно-контактных схем (LD)

Рассмотрим язык релейно-контактных схем, дающий наглядное представление о работе контроллера за счет того, что изначально контроллер создавался для замены реле. Таким образом, данный язык нагляден и понятен программисту, который только начинает знакомиться с программной логикой.

Структура команды на языке релейно-контактных схем показана на рисунке 11.1.

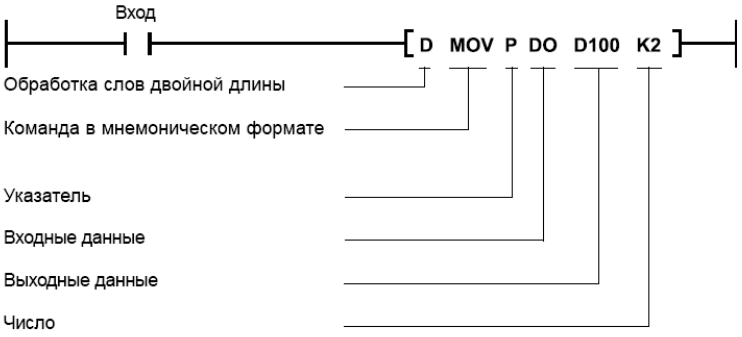
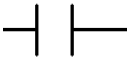


Рисунок 11.1 – Структура команды

11.1 Основные команды

11.1.1 Команда (LD) - нормально открытый контакт



Прочитав этот сигнал, контроллер начинает постоянно проверять состояние входа, норме которого указан программистом. (Далее для простоты изложения будут указаны конкретные номера, они выбраны произвольно). И как только контролер определяет изменение состояния входа (X1), с выключено на включено, следует включение выхода (Y1). Данный символ может относиться не только к физическим входам контроллера, а также и к внутренним (вспомогательным) реле. Число и ограничения по использованию внутренних реле определяется только возможностями контроллера.

11.1.2 Команда (LDI) - нормально закрытый контакт

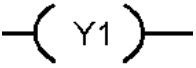
Прочитав этот сигнал, контроллер начинает постоянно проверять состояние входа (X2). И как только контроллер определяет изменение состояния входа (X2), с включено на выключено, следует включение выхода (Y1). Данный символ может относиться не только

к физическим входам контроллера, а также и к внутренним (вспомогательным) реле. Логика работы соответствует таблице 11.1.

Таблица 11.1 – Таблица логического состояния входов

Логическое состояние	*LD* - нормально открытый контакт	*LDI* – нормально закрытый контакт
0	Ложь	Истина
1	Истина	Ложь

11.1.3 Команда (OUT) - инициализация ***Выхода***



Прочитав эту команду, контроллер изменит состояние выхода (Y1), с выключено на включено. Так же, как и в случае с входами данный символ может относиться не только к физическим выходам контроллера, но и к внутренним (вспомогательным) реле. Число и ограничения по использованию внутренних реле определяется только возможностями контроллера. Логика работы соответствует таблице 11.2.

Таблица 11.2 – Таблица логического состояния выходов

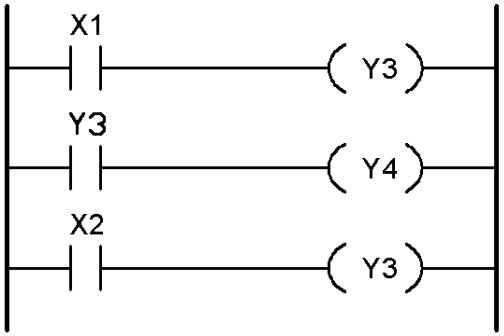
Логическое состояние	*OUT* – Выход
0	**Ложь**
1	**Истина**

Примечание:

Избегайте двойной записи выходов (double coil) , так как это может привести к помехам при отработке программы.

Пример двойной записи приведен на рисунке 11.2 а – ошибочная, б – верная запись:

а)



б)

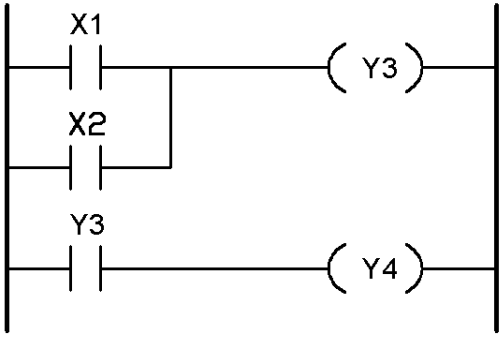


Рисунок 11.2 – Пример двойной записи выхода

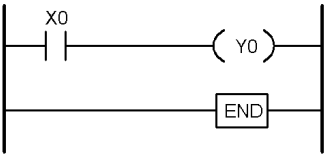
Исходим из того, что вход X1 включен (сигнал «1»), а вход X2 отключен (сигнал «0»).

Первая запись выхода Y3 активизируется включенным входом X1, в отображении процесса выходов Y3 включен, соответственно активизируется также выход Y4.

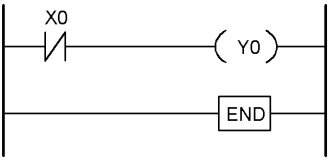
Эта программная последовательность имеет следствием то, что когда выключится X1, Y3 останется включенным, при условии, что был включен X2.

Для исправления такой ошибки, необходимо использовать оператор «ИЛИ». (см. далее)

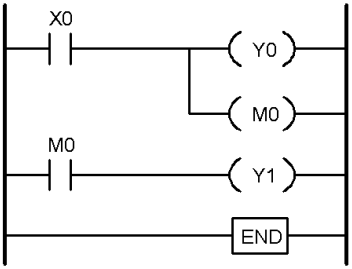
Отработка зависимости выходов от значения входов показана на рисунке 11.3:



Y0 будет в состоянии «истина», когда вход X0 будет включен («истина») , т.е. замкнут.



Y0 будет в состоянии «истина», когда вход X0 будет включен («истина») , т.е. замкнут.

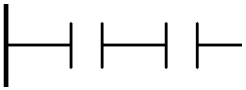


Вместе с Y0 будет включено внутреннее реле M0 (см. далее), замыкание которого повлечет в свою очередь установление выхода Y1 в состояние 1 («истина»).

Рисунок 11.3 – Зависимость отработки выходов от значения входов

11.1.4 Команды логических связей процесса (AND/ANI/OR/ORI)

Команда (AND) - логическое умножение



Операция логического умножения. В языках программирования и языках запросов обозначается символами

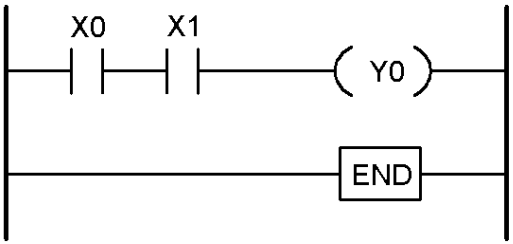


Рисунок 11.4 - Логическое умножение

AND, И, & и другими способами. Результатом операции является «истина», если оба операнда принимают значение «истина», и «ложь» – в остальных случаях. Пример записи команды показан на рисунке 11.4.

Команда (ANI) – отрицание логического умножения

По аналогии с предыдущим операндом результатом операции является «истина», если X0 будет замкнут, и X1 – разомкнут, т.е. оба входа примут значение «истина», в остальных случаях Y0 будет «ложь». Пример записи команды показан на рисунке 11.5.

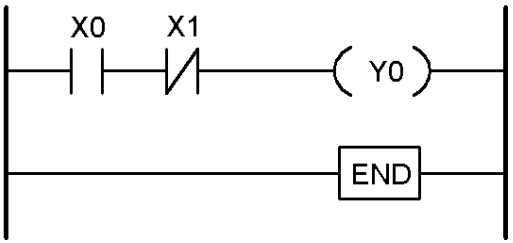
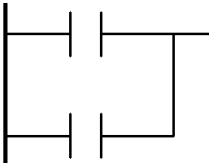


Рисунок 11.5 – Отрицание логического умножения

Команда (OR) - логическое сложение



Операция логиче-

ского сложения. В языках программирования и языках запросов обозначается символами OR и другими способами. Результатом операции является «истина», если оба или один из операндов принимают значение «истина», и «ложь» – в остальных случаях. Пример записи команды показан на рисунке 11.6.

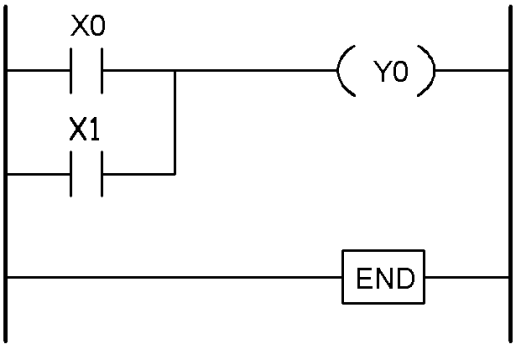
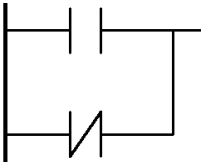


Рисунок 11.6 – Логическое сложение

Команда (ORI) - отрицание логического сложения



Аналогично предыдущему варианту. Пример записи команды показан на рисунке 11.7.

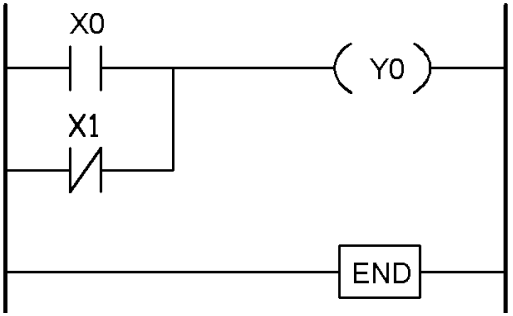
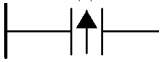


Рисунок 11.7 – Отрицание логического сложения

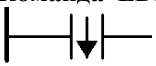
11.1.5 Команды (LDP) и (LDF) – управление по фронтам входных сигналов

При необходимости использовать фронты сигналов входов (передний или задний) сигнал входа будет иметь вид

Команда LDP (управление по переднему фронту)



Команда LDF (управление по заднему фронту)



Пример использования управления по фронтам сигналов показан на рисунке 11.8.

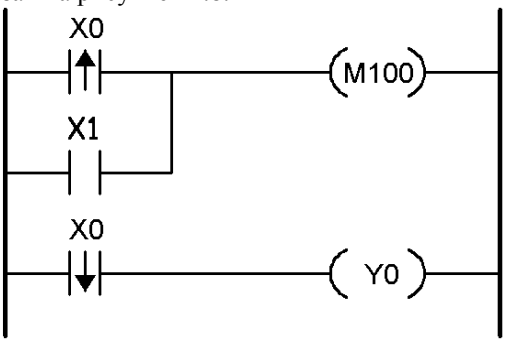


Рисунок 11.8 – Управления по фронтам сигналов

Внутреннее реле M100 (меркер) включается на время включения X1 или при положительном фронте X0 (моменте его включения).

Выход Y0 включается при отрицательном фронте X0 (моменте его отключения).

11.1.6 Команды SET(Установить)/RST(Сбросить)

Пример, объясняющий функциональность команд прямой установки и сброса, показан на рисунке 11.9.

Состояние сигнала операнда с помощью «SET/RST» команд (включение/выключение) может устанавливаться непосредственно. С помощью «SET/RST» могут устанавливаться в «1»/«0» (включаться/выключаться) соответствующие операнды, например: Y (выход) , M (внутреннее реле) или S (состояние шагов).

Также «RST» применяется для обнуления регистров и счетчиков.

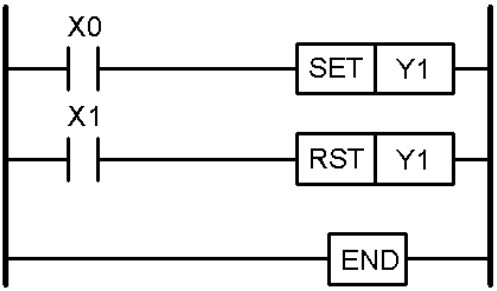
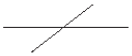


Рисунок 11.9 – Команды прямой установки

Примечание: команда «RST» преобладает над командой «SET».

11.1.7 Команда (INV) – Инверсия результата обработки



Инвертирует состояние сигнала результата стоящей впереди команды. Полученный согласно обработки сигнал «1», после инверсии становится «0», и соответственно наоборот «0» становится «1».

11.1.8 Команда (NOP) – Пустая строка в программе

Можно создать пустую строку без логических функций, которая позднее может быть использована для каких-либо команд, например, при окончательном изготовлении программы, при отладке оборудования. После успешного завершения отладки программы «NOP»-команды должны быть удалены, так как в противном случае они бесполезно удлиняют время цикла программы. «NOP» – команда может использоваться для создания паузы нужной длительности при отработке программы ПЛК.

11.1.9 Команда (END) – конец программы

Окончание программы ПЛК и переход к началу программы (шаг 0). Каждая программа ПЛК должна завершаться командой

«END» . На этом месте оканчивается обработка программы. Последующие области программы не принимаются во внимание. После обработки «END»–команды выполняется обработка выходов. Чтобы организовать участки программы для пошаговой проверки, можно вводить «END»–команду также внутри программы. Эта дополнительная «END»–команда должна после окончания проверки удаляться.

11.2 Программирование внутреннего реле

Давайте вернемся к предыдущему примеру и посмотрим, как контроллер обрабатывает состояния входов-выходов, и хранит полученные значения. Приведем вновь рисунок, упрощенно изображающий ПЛК для управления технологическим процессом и его программу (рисунок 11.10).

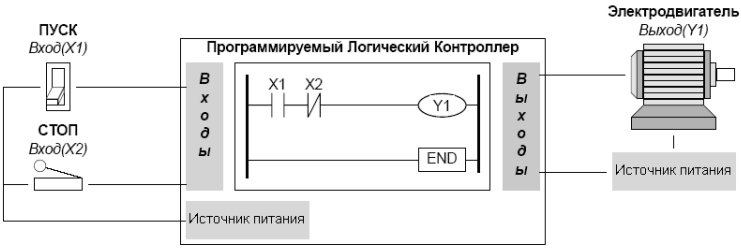


Рисунок 11.10 – Пример программы для ПЛК

Таблица 11.3 дает представление о том, как контроллер хранит полученные значения в регистрах

Таблица 11.3 – Пример состояния регистров входов и выходов ПЛК

Регистр входов (X0…X15)															
X15	***X14***	***X13***	***X12***	***X11***	***X10***	***X9***	***X8***	***X7***	***X6***	***X5***	***X4***	***X3***	***X2***	***X1***	***X0***
													1	**0**	

Регистр выходов (Y0…X15)															
Y15	***Y14***	***Y13***	***Y12***	***Y11***	***Y10***	***Y9***	***Y8***	***Y7***	***Y6***	***Y5***	***Y4***	***Y3***	***Y2***	***Y1***	***Y0***
														0	

Регистр входов:

– в регистре X1 находится значение «0» (бит «0»), то есть вход (X1) – выключен;

– в регистре X2 находится значение «1» (бит «1») следовательно, вход (X2) – включен.

Регистр выходов:

– в регистре Y1 находится значение «0» (Бит «0»), то есть выход (Y1) – выключен.

В действительности во всех остальных ячейках находится значение «0», но они не отображены, чтобы сосредоточиться на примере.

Рассмотрим то, как изменятся значения в регистрах после того, как включится выключатель – вход (X1). Значения регистров выходов приведены в таблице 11.4.

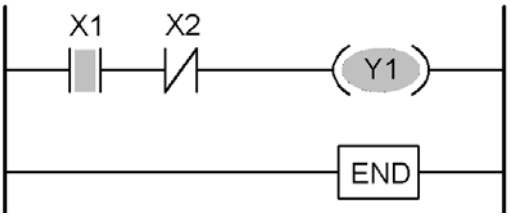


Рисунок 11.11 – Реакция выхода на изменение состояния входа

Таблица 11.4 – Пример состояние регистров входов и выходов ПЛК

Регистры входов (X0…X15)															
X15	***X14***	***X13***	***X12***	***X11***	***X10***	***X9***	***X8***	***X7***	***X6***	***X5***	***X4***	***X3***	***X2***	***X1***	***X0***
													1	**1**	

Регистры выходов (Y0…X15)															
Y15	***Y14***	***Y13***	***Y12***	***Y11***	***Y10***	***Y9***	***Y8***	***Y7***	***Y6***	***Y5***	***Y4***	***Y3***	***Y2***	***Y1***	***Y0***
														1	

Согласно схемы после включения входа (X1) включился и выход (Y1) – электродвигатель заработал и технологический процесс пошёл. Рисунок 11.11 иллюстрирует это.

Таблица 11.5 описывает все возможные состояния для двух входов и одного выхода.

Таблица 11.5 – Таблица возможных состояний входов и выходов ПЛК

Входы (X1/X2)		***Выход (Y1)***	***Значение в регистрах***		
LD (X1)	***LDI (X2)***	***OUT (Y1)***	***LD (X1)***	***LDI (X2)***	***OUT (Y1)***
Ложь	Истина	Ложь	0	0	0
Истина	Истина	Истина	1	0	1
Истина	Ложь	Ложь	1	1	0
Ложь	Ложь	Ложь	0	1	0

Из таблицы 11.5 можем видеть, что контроллер выполнит операцию включения выхода (Y1) тогда, когда вход (X1) и вход (X2) будут в состоянии «истина».

Данный пример показывает (рисунок 11.12), как работает логика «AND» («И»), т. е. выход (Y1) изменит свое состояние «0» на «1» тогда и только тогда, когда вход (X1) и вход (X2) будут в состоянии «истина». Во всех остальных случаях выход (Y1) будет в состоянии «0», т. е. «выкл.».

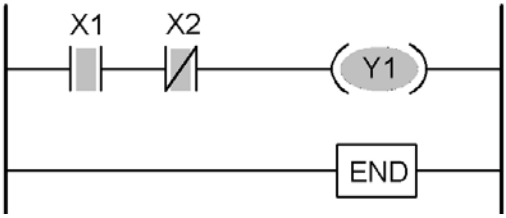


Рисунок 11.12 – Пример работы логики «И»

Использование внутреннего реле (меркера) – контроль уровня.

Предположим нам необходимо постоянно поддерживать определенный уровень в баке с водой (рисунок 11.13). Для решения необходимо:

- датчик верхнего уровня – вход (X2) (переходит в состояние «включено», когда уровень воды падает);
- датчик нижнего уровня – вход (X1) (переходит в состояние «включено», когда уровень воды падает);
- насос – выход (Y1).

К сожалению, использование логики для двух входов не даст желаемого результата, так как если мы будем включать насос, когда оба входа окажутся в состоянии «включено», то насос будет поднимать уровень только до нижнего датчика, или уровень воды никогда не опустится до уровня нижнего датчика, т.е. начнет работать в очень жестком режиме постоянного включения-выключения.

Для обеспечения работы насоса в оптимальном режиме нам понадобится включение в задачу дополнительного элемента, который должен обеспечить:

- *Включение* насоса при достижении нижнего уровня;
- *Выключение* насоса при достижении верхнего уровня.

Для этого мы будем использовать внутренне реле контроллера M1, также называемое меркером, в результате чего получим схему, изображенную на рисунке 11.13.

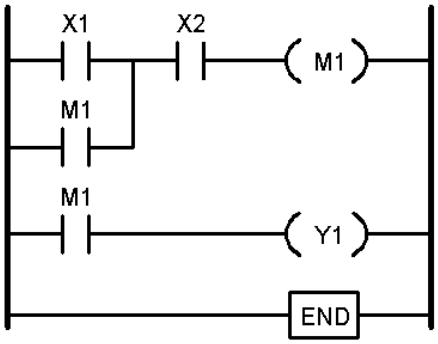


Рисунок 11.13 – Схема программы для включения насоса

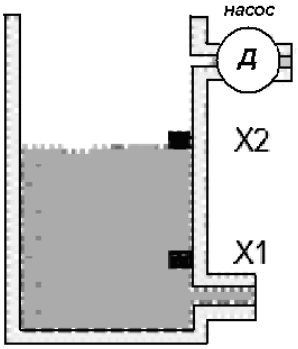


Рисунок 11.14 – Бак с водой

Когда вода опустится ниже уровня второго датчика, то оба входа (вход (X1) и вход (X2)) изменят свое состояние с «0» на «1», за этим последует изменение состояния внутреннего реле M1 с «0» на «1» и включение насоса.

Когда вода поднимется до уровня второго датчика, то вход (X2) будет в состоянии «1», а вход (X1) изменит свое состояние с «1» на «0», и в результате насос будет продолжать работать, так как внутреннее реле будет в состоянии «1» до тех пор, пока вода не достигнет уровня верхнего датчика и не изменит состояние входа (X2) с «1» на «0». За этим последует изменение состояния регистра M1 с «1» на «0» и выключение насоса.

Данный пример демонстрирует необходимость и важность использования внутренних реле контроллера, количество которых зависит только от модели контроллера. Внутренние реле позволяют решать задачи управления с минимальным числом внешних устройств, что и является основной задачей контроллера.

11.3 Программирование счетчика. Команда COUNTER

Счетчик – самое простое устройство в контроллере, так как предназначен только для одного – считать. Конечно, в реальности в зависимости от модели контроллер может поддерживать различные варианты использования данной функции:

- суммирующие счётчики (прямой счет 1, 2, 3 ...);
- обратные счётчики (счет вниз 3, 2, 1 ...);
- счет вверх-вниз (счет вниз 1, 2, 3, 4, 5, 4, 3, 2, 3, 4, 5 ...).

Возможность использования того или другого типа счетчика относится только к возможностям конкретной модели контроллера.

Дополнительно счетчики делятся по способу обработки импульсов на два основных типа:

- Программные – напрямую зависят от быстродействия контроллера и не могут работать быстрее скорости обработки двух программных циклов (при использовании программного счетчика допускается, что быстродействии счетчика не превышает «*Времени цикла обработки*» X2. В противном случае необходимо использовать высокоскоростные аппаратные счетчики);
- Аппаратные – не зависят от быстродействия контроллера и могут работать быстрее времени обработки одного программного

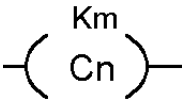
цикла (с частотой до 100 кГц). Мы можем считать, что данный тип счетчиков физически существует;

Независимо от того – программные или аппаратные счетчики, выбор должен определяться только тем, с какой скоростью будет работать *счетный вход* и позволяет ли быстродействие контроллера обработать сигналы с датчика или нет.

Для правильного использования счетчиков необходимо определить для себя всего 3 вещи:

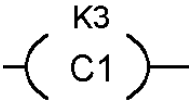
- **С какой частотой мы хотели бы считать.** Так как обычно, использование *программного счетчика* допустимо для любого из входов, то при выборе *аппаратного счетчика* мы можем использовать только те входы, которые служат для высокоскоростного счета. Для определения возможности использования того или иного входа в качестве высокоскоростного необходимо сверяться с руководством пользователя или с руководством по программированию;
- **До какого значения мы собираемся считать импульсы.** Диапазон, в котором может считать контролер 0 ... 32.767, -32.768 ... -32.768, 0 ... 65535, ... зависит только от конкретной модели контроллера;
- **По какому условию мы можем остановить счет.**

Команда (OUT Cn Km) – Инициализация счетчика.



C – обозначение счетчика;
n – число от 1 до 256 – номер счетчика;
K – обозначение константы;
m – число от 1 - 2.147.483.648, до которого будет вестись счет.

Например, при n = 1, m = 3 :



Вспомним наш первый пример и попробуем решить следующую задачу. Пусть насос (выход (Y1)) включится только после того, как мы три раза включим-выключим вход (X2) (см. рисунок 11.14).

- То есть при включении входа (X1) – счетчик (C1) установится в значение «0»;
- Первое нажатие на вход (X2) – счетчик (C1) сравнит «1» = K3 («3») – если нет, то запомнит «1»;

▪ Второе нажатие на вход (X2) – счетчик (C1) прибавит «1», сравнит «2» = K3(«3») – если нет, то запомнит «2»;

▪ Третье нажатие на вход (X2) – счетчик (C1) прибавит «1» сравнит «3» = K3(«3») – если да, то сменит состояние C1 с «0» на «1» и насос заработает;

▪ Повторное включение Входа (X1) – счетчик (C1) установится в значение «0» и цикл начнется с начала.

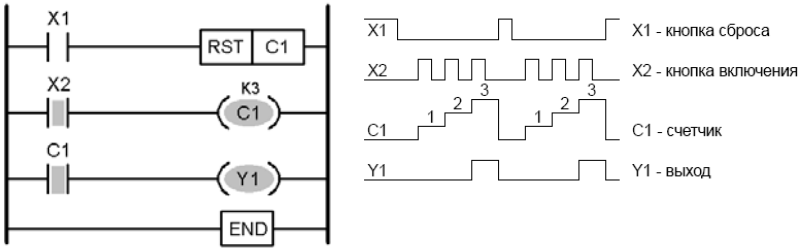


Рисунок 11.15 – Схема и диаграмма работы счетчика

11.4 Программирование таймера. Команда TIMER

Попробуем классифицировать, какими бывают таймеры:

▪ ***с задержкой по включению.*** Другими словами, после того, как Вы нажали на выключатель (т.е. придя поздно вечером домой и нажав на выключатель, вы с удивлением обнаруживаете, что свет загорается только через одну минуту после включения выключателя);

▪ ***с задержкой по выключению.*** Иначе, после того, как Вы выключили свет (т.е. уходя поздно вечером из дома и нажав на выключатель, вы с удивлением обнаруживаете, что свет в квартире погас только спустя некоторое время);

▪ ***Накапливающий таймер*** – Этот тип таймера позволит Вам выключить свет только после того, как суммарное время включения достигнет определенного значения. Данный тип таймера требует обязательного использования двух входов.

Что мы должны определить для себя – это всего лишь 2 вещи:

▪ Какой из входов запустит таймер;

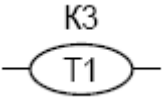
▪ Какую задержку времени установить, т.е. сколько времени пройдет, прежде чем включится-выключится выход.

С того момента, когда все команды перед символом таймера примут значение – «истина», таймер начинает отсчёт времени и по

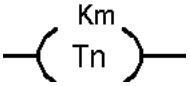
достижении установленного значения переключит состояния выхода с «включено» на «выключено» или наоборот. В любой момент времени работы таймера есть возможность отображать его текущее состояние. Диапазон, в котором может работать контроллер (0 ... 32.767, -32.768 ... -32.768, 0 ... 65535) зависит только от конкретной модели контроллера.

Правила и возможность использования определяются документацией по программированию контроллера. Так же, как и в случае со счетчиками, некоторые модели контроллеров позволяют использовать *высокоскоростные таймеры.*

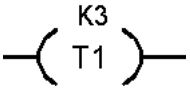
Команда (OUT Tn Km) – Инициализация таймера.



T – обозначение таймера;
n – число от 1 до 256-номер таймера;
K – обозначение константы;
m – число от 1 до 32768, до которого будет вестись
отсчет времени.

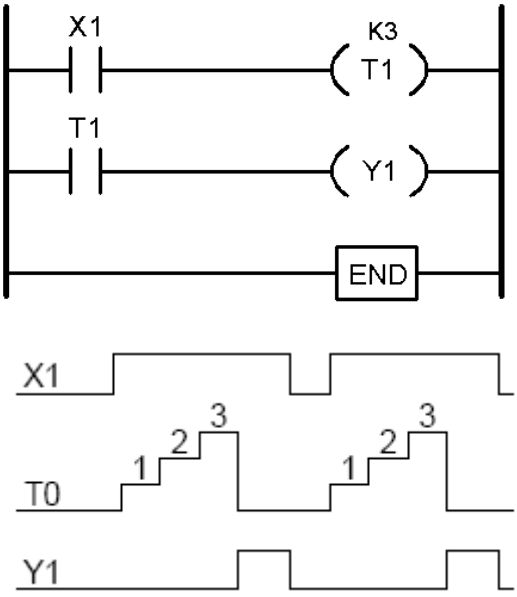


Например, при n = 1, m = 3:



Пример 1: Таймер с задержкой по включению.

Вспомним наш пример с уровнем воды в баке и решим следующую задачу: пусть насос (выход (Y1)) включится через три секунды после включения входа (X1). Схема работы контроллера показана на рисунке 11.16.



- Вход (X1) включился – таймер (T1) начал отсчет времени;
- Прошло три секунды – выход (Y1) включился;
- Вход (X1) выключился – выход (Y1) выключился.
- Вход (X1) включился – таймер (T1) начал отсчет времени;
- Прошло три секунды – выход (Y1) включился;
- Вход (X1) выключился – выход (Y1) выключился.

Рисунок 11.16 – Схема и временная диаграмма программирования таймера с задержкой по включению

Понятно, что шаг мог бы быть и 0,1 мс или 0,01 мс и т. д. Аналогично работает таймер с задержкой по выключению.

Пример 2: Таймер с накоплением

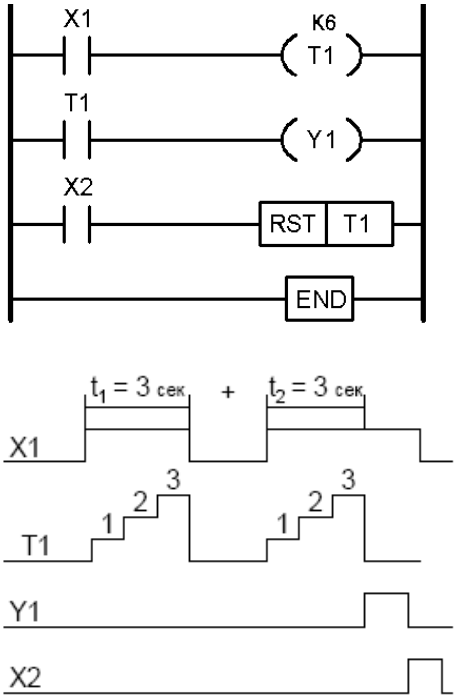


Рисунок 11.17 – Схема и временная диаграмма программирования таймера с накоплением

Пусть насос (выход (Y1)) включится только после того, как вход (X1) отработает цикл включено-выключено-включено, общей длительностью 6 секунд. Схема работы контроллера показана на рисунке 11.17.

- Вход (X1) включился – таймер (T1) начал отсчет времени;
- Прошло три секунды – выход (X1) выключился;
- Таймер запомнил время работы входа(X1) – $t_1 = 3$ с;
- Вход (X1) включился – таймер (T1) продолжил отсчет времени, как только $T = t_1 + t_2 = 6$ с, то выход (Y1) включился;
- Вход (X2) включился – таймер (T1) изменил свое состояния с «1» на «0» и выход (Y1) выключился.

Пример 3: Таймер с памятью

Наряду с уже описанными видами таймеров имеются также таймеры с памятью, которые после отключения управляющей логической связи сохраняют уже накопленное значение времени. Действительное значение времени в таймере записывается в память, содержимое которой сохраняется и при отключении напряжения. Пример работы таймеров с памятью показан на рисунке 11.8.

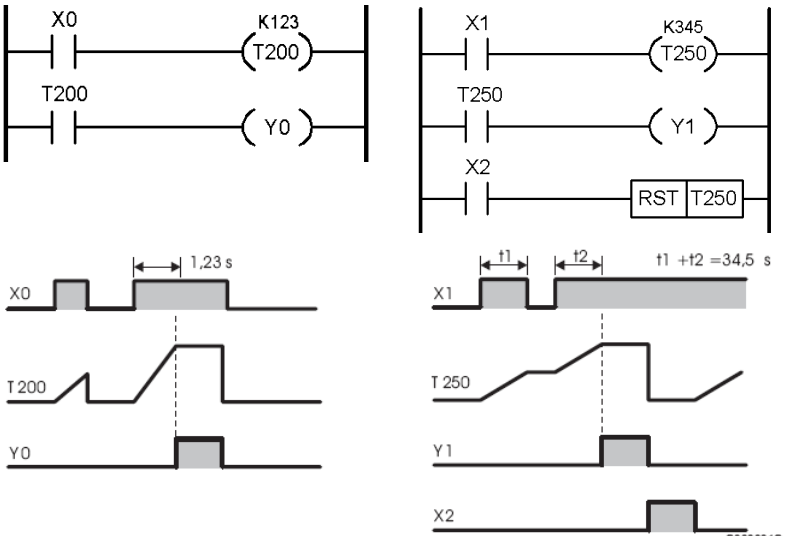


Рисунок 11.18 – Схемы и временные диаграммы работы таймера с памятью

TIMER – погрешность

Теперь вернемся немного назад и вспомним, как работает контроллер – что такое быстродействие? Конечно, если использовать таймер с шагом 1 секунда, то можно не беспокоиться, но когда используются таймеры, которые работают с шагом от 0,001 секунды, просто необходимо помнить о быстродействии контроллера.

Рассмотрим два типичных случая, в которых обычно не учитывается погрешность при использовании таймера:

- погрешность по входу;
- погрешность по выходу.

Погрешность по входу (рисунок 11.19) означает, что с момента, когда включится вход (X1), до начала работы таймера пройдет время, равное:

$$t_{Вып.прог} \cdot 2 + t_{Вх} + t_{Вых} = t_{Погр.по\ Вых.}$$

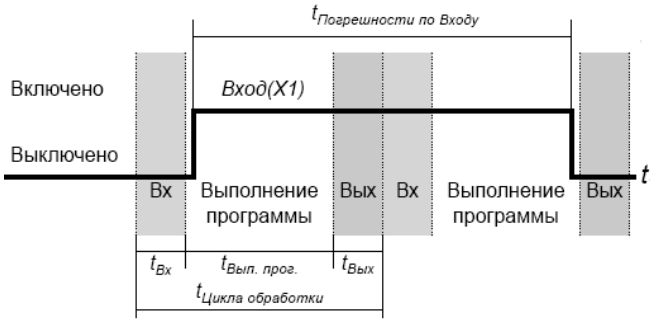


Рисунок 11.19 – Временная диаграмма погрешности таймера по входу

Погрешность по выходу (рисунок 11.20) означает, что с момента, когда выключится вход (X1), до начала работы таймера пройдет время, равное

$$t_{Вып.прог} \cdot 2 + t_{Вх} + t_{Вых} = t_{Погр.по\ Вых.}$$

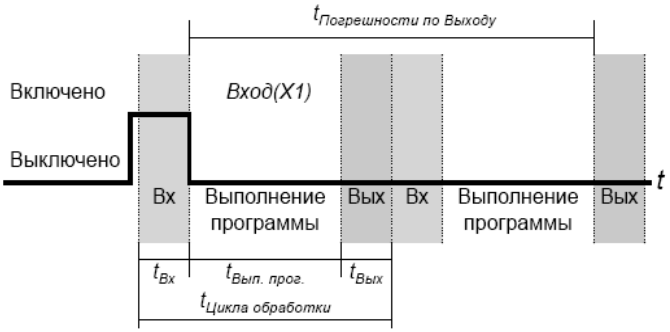
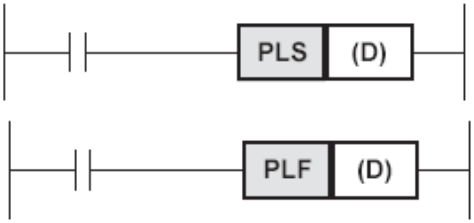


Рисунок 11.20 – Временная диаграмма погрешности таймера по выходу

Таким образом, если контроллер имеет $t_{Цикла\ обработки}$ = 5 мк и поддерживает работу таймера с шагом 1 мсек., то минимальный шаг,

с которым может корректно работать таймер контроллера, должен быть больше, чем 10 мсек. В действительности, кроме «программной погрешности», мы должны принимать во внимание и существование «аппаратной погрешности», т. е. времени, необходимого контроллеру на проверку действительности срабатывания входа. В реальных условиях возможен шум или скачок, который контроллер может принять за включение входа, хотя этого и не произошло. Обычно производители предусматривают настройку данного параметра в диапазоне от 0 до 10 мсек., в зависимости от «чистоты» линии.

11.5 Программирование одиночных импульсов. Команды (PLF) и (PLS)



Генерация одного импульса – опознание фронта сигнала независимо от продолжительности входного сигнала для включения соответствующего операнда (выхода Y или внутреннего реле M). Операнд остается включенным на протяжении одного цикла программы (скана).

PLS – генерация одиночного импульса по возрастающему фронту входного сигнала.

PLF – генерация одиночного импульса по спадающему фронту входного сигнала.

Пример применения однократных импульсов показан на рисунке 11.21:

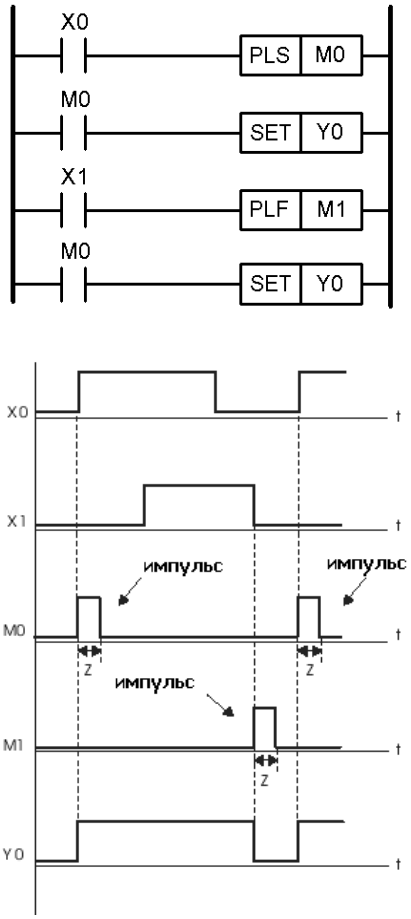


Рисунок 11.21 – Схема и временная диаграмма применения команды временных импульсов

При возрастании входного сигнала на входе X0 с «0» до «1» (возрастающий фронт) внутреннее реле M0 благодаря «PLS»-команде получает импульс (включается на время одного цикла программы). С помощью этого импульса по контакту реле M0 включается выход Y0. Лишь когда на входе X1 пройдет смена сигнала с «1» на «0» (падающий фронт), выход Y0 снова отключится (см. рисунок 11.21).

Генерация одиночного импульса по возрастающему фронту входного сигнала

Генерация одиночного импульса по спадающему фронту входного сигнала

Z – Время цикла программы (время скана)

Глава 12. Инструкции процесса отработки программы

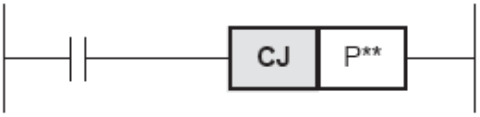
12.1 Структуризация программы

Программы для ПЛК составляются из трех основных элементов: главная программа, подпрограммы (необязательные) и программы обработки прерываний (необязательные).

- Главная программа. В этой основной части программы располагаются операции, управляющие всем приложением. Операции главной программы в каждом цикле обрабатываются последовательно. Для окончания главной программы используется инструкция абсолютного завершения программы («END»).
- Подпрограммы. Эти необязательные компоненты программы обрабатываются только тогда, когда они вызываются из главной программы. Подпрограммы располагаются после главной программы (после «FEND»-инструкции и перед «END»-инструкцией). Каждая подпрограмма завершается командой «SRET», «вернуться».
- Программы обработки прерываний. Эти необязательные компоненты программы обрабатываются только тогда, когда появляется событие прерывания. Программы обработки прерываний располагаются также после главной программы (после «FEND»-инструкции и перед «END»-инструкцией). Каждая программа обработки прерываний завершается операцией «IRET», «вернуться из программы обработки прерываний».

Возможно располагать подпрограммы и программы обработки прерываний после главной программы в смешанной последовательности. Однако, если Вы хотите, чтобы программа была легко читаемой и понятной, то необходимо присоединить все подпрограммы непосредственно к главной программе, а затем расположить все программы обработки прерываний вслед за подпрограммами.

12.2 Переход внутри программы (CJ)



С помощью «CJ»-инструкции можно «перепрыгивать» через часть программы. При применении этой инструкции время программы может уменьшиться. Цель (конец) перехода определяется установкой маркировки (маркировка точки) в программе. Для маркировки используют точки P0…P63.

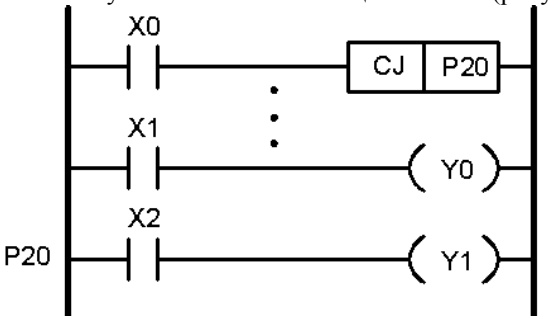
Если внутри подпрограммы перехода программируется инструкция сброса (отключения) для счетчика с запоминанием, то процесс сброса (стирание накопленного значения) имеет место тогда, когда перепрыгивается цепь схемы катушки счетчика.

Примечание:

При дублировании записи выходов необходимо следить за тем, чтобы оба выхода никогда не были активными в одно и тоже время. Это может привести к ошибочной отработке программы.

Маркировка точки в программе:

При программировании на языке контактных схем маркировка точки указывается слева от цепи схемы (рисунок 12.1).



Если включается X0, то выполняется переход к маркировке точки P20.

Рисунок.12.1 – Пример программирования CJ-инструкции

Пример двукратной вставки в программу адреса точки P9 показан на рисунке 12.2.

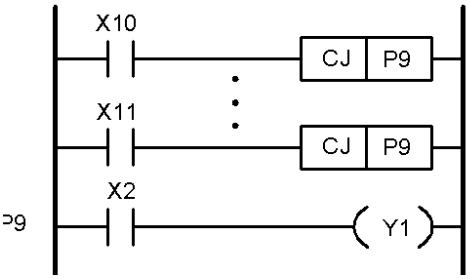


Рисунок 12.2.– Пример двукратной вставки в программу адреса точки P9.

Если X10 включен, то выполняется переход к промаркированной точке P9.
Если X10 выключен, а X11 включен, то все равно произойдет переход к точке P9.

Примечание:
Одинаковая маркировка точек не должна многократно использоваться в программе, т.к. может создаться ошибка в работе программы.

Маркировка точки перед «CJ»-инструкцией перехода:

Обратный переход (вверх программы) также может выполняться внутри программы.

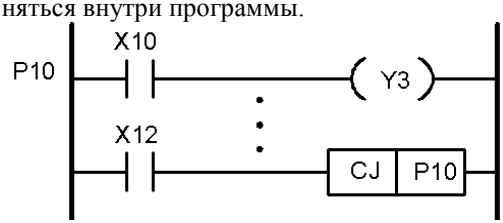
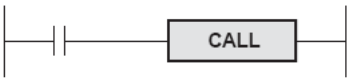


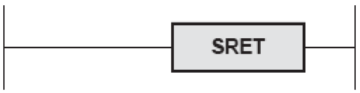
Рисунок.12.3 – Маркировка точки перед «CJ»-инструкцией

Примечание:
Если входной сигнал для «CJ»-инструкции держится больше 200 мс, то появляется ошибка времени работы (Watch-Dog-Timer).

12.3 Вызов подпрограммы (CALL / SRET)



Вызов подпрограммы



Конец подпрограммы

С помощью «CALL»-инструкции вызывается подпрограмма, промаркированная с помощью точек (P0...P62). Подпрограмма программируется после «FEND»-инструкции и перед «END»-инструкцией. В конце подпрограммы должна находиться «SRET»-инструкция. Если активируется «CALL»-инструкция, то выполняется переход к указанной точке маркировки. После отработки «SRET»-инструкции выполняется обратный переход к инструкции, следующей за «CALL»-инструкцией. Активированные в подпрограмме операнды остаются таковыми после отработки подпрограммы до новой ее обработки. В подпрограмме должны использоваться таймеры T192...T199 и T246...T249. Те же точки могут использоваться с любым числом CALL-инструкций. Внутри подпрограммы могут вызываться другие подпрограммы. Возможно максимум 4 уровня ветвления.

Пример применения подпрограмм показан на рисунке 12.4.

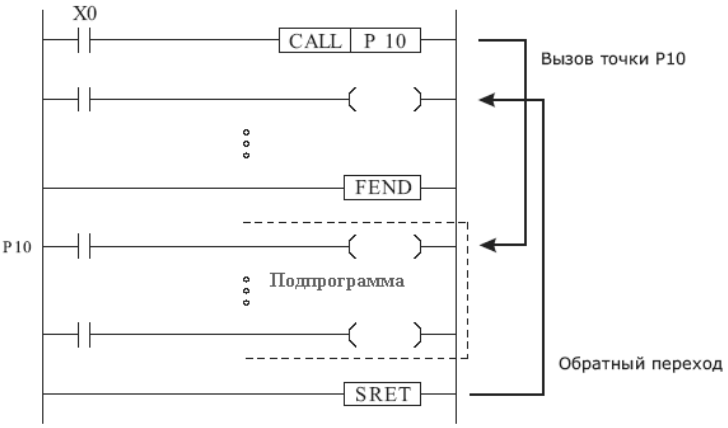
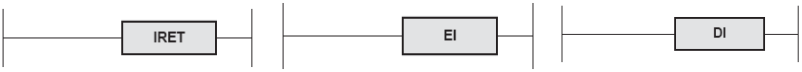


Рисунок 12.4 – Пример программирования с применением «CALL»- и «SRET»-инструкций

12.4 Ввод прерывания программы (IRET, EI, DI)



Окончание программы прерывания

Импульсный запуск/активизация программы прерывания

Деактивация программы прерывания

Принцип функционирования.

Вызов, Окончание, Активизация и Деактивация программы прерывания.

Вызов программы прерывания.

При вызове программы прерывания останавливается основная программа и выполняется переход к программе прерывания. После окончания программы прерывания выполняется возврат к основной программе. Начало программы прерывания определяется установкой маркировки (точки прерывания). Конец программы прерывания определяется «IRET»-инструкцией. Входы X0...X5 образуют входы прерывания. Сигналы прерывания должны иметь ширину импульса минимум в 200 мкс.

Примечание:

Входы X0...X5 не могут применяться одновременно для обработки сигналов прерывания и для обработки сигналов высокоскоростного счетчика.

Адресация точек прерывания.

Точка прерывания: I X 0 x

X – адрес 0...5, соответствующий входам X0...X5

x = 0 – прерывание при падающем фронте входного сигнала.

x = 1 – прерывание при возрастающем фронте входного сигнала.

Примечание:

Адрес прерывания может использоваться только один раз

Применение «EI»- и «DI»-инструкций.

С помощью «EI»-инструкции могут активироваться инструкции прерывания. Это означает, что после отработки «EI»-инструкции, смена сигнала, которая появляется на одном из входов X0...X5, обрабатывается как сигнал прерывания в программе.

С помощью «DI»-инструкции могут деактивироваться инструкции прерывания. Это означает, что после отработки «DI»-инструкции, смена сигнала, которая появляется на одном из входов X0...X5, не обрабатывается больше как сигнал прерывания в программе.

Примечание:

Если ни одна из обеих инструкций «EI» или «DI» не программируется, режим прерывания не активизируется, т.е. тогда не может обрабатываться никакой сигнал прерывания.

Отработка программы прерывания.

Во время исполнения программы прерывания не может вызываться никакая другая программа прерывания. Однако может программироваться два уровня разветвления.

Несколько, одна за другой следующие, программы прерывания обрабатываются в последовательности их вызова. Если одновременно вызываются несколько программ прерывания, то вначале обрабатывается программа прерывания с более низким адресом точки.

Выключение прерывания.

Любое прерывание может повременно или постоянно выключаться посредством включения соответствующего специального меркера (внутреннего реле). Для FX0S первым специальным меркером является M8050, который выключает прерывание IOab. Соответствующие специальные меркеры указываются в руководстве пользователя.

Пример адресации точки прерывания показан на рисунке 12.5.

Точка I001 – вход прерывания X0, прерывание при возрастающем фронте входного сигнала (смена сигнала с «0» на «1»).

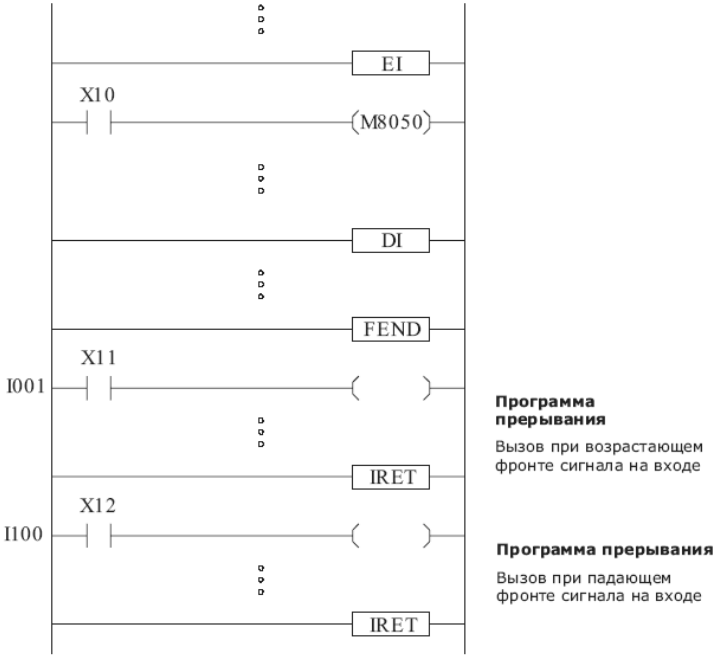


Рисунок 12.5 – Пример программирования при использовании инструкций «EI», «DI» и «IRET»

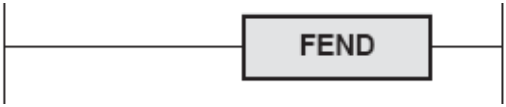
Если вход X0 устанавливает сигнал прерывания во время выполнения шага программы внутри области от EI-инструкции до DI-инструкции, то имеет место переход к программе прерывания I001. Программа прерывания выполняется и происходит возврат в основную программу.

Программа прерывания I001 не выполняется, если активизирован специальный меркер M8050 (вход X10 включен).

Если вход X1 устанавливает сигнал прерывания во время выполнения шага программы внутри области от EI-инструкции до «DI»-инструкции, то имеет место переход к программе прерывания I100. Программа прерывания выполняется и происходит возврат в основную программу.

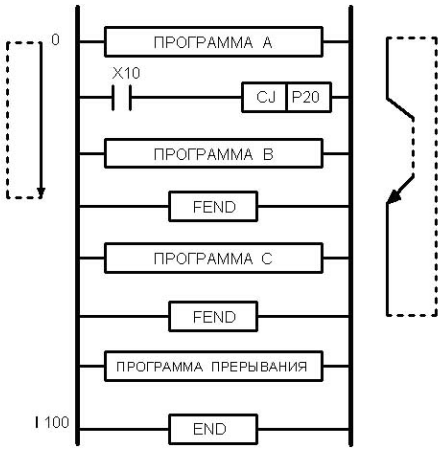
Если появляются одновременно сигналы X0 и X1, то вначале обрабатывается программа прерывания I001, а затем программа прерывания I100.

12.5 Конец области программы (FEND)



С помощью «FEND»-инструкции определяется конец области программы. В программе возможно применение нескольких «FEND»-инструкций. После отработки «FEND»-инструкции выполняется обработка выходов, затем выполняется возврат к программному шагу 0, после чего обновляется обработка входов и время установки контроля цикла программы. Пример работы программы с применением «FEND» показан на рисунке 12.6.

Работа программы, если X10 не включен

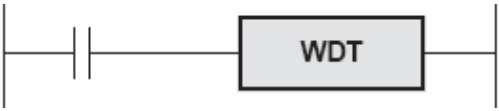


Работа программы, если X10 включен

Область программы В перепрыгивается

Рисунок12.6 – Пример программирования «FEND»-инструкции

12.6 Обновление таймера времени работы программы (WDT)



С помощью «WDT»-инструкции можно длинные программы разделить на отдельные отрезки программ. Время цикла программы (скана) определяется для каждого отдельного отрезка программы самим ПЛК (WDT обновляется после каждого отрезка программы). С помощью «WDT»-инструкции можно обрабатывать программу, время цикла которой превышает 200 мс. На рисунке 12.7 показана типичная ситуация, в которой необходимо использовать «WDT»-инструкцию.

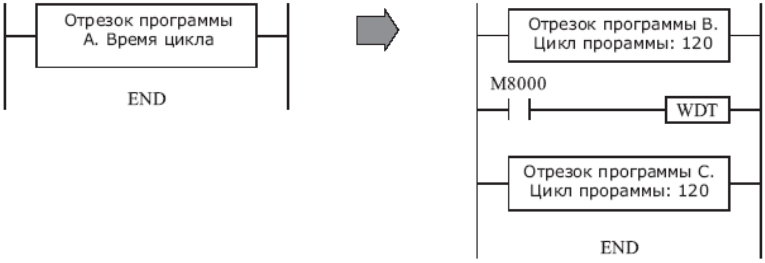


Рисунок 12.7 – Пример программирования при использовании «WDT»-инструкции

Время обработки для отрезка программы A превысило значение 200 мс. Поэтому отрезок программы A был разделен с помощью «WDT»-инструкции на два отрезка программ (B, C). Отрезки программ B и C требуют соответственно только по 120 мс времени цикла.

Значение времени цикла программы изменяется в специальном регистре D8000. Если время цикла программы постоянно превышает значение 200 мс, можно изменить значение максимально допустимого времени цикла в специальном регистре D8000 (смотри пример на рисунке 12.8)

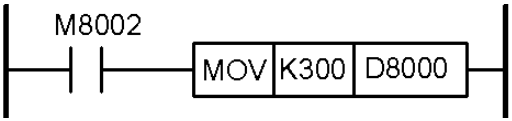
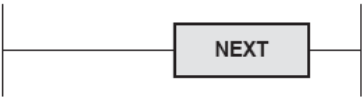


Рисунок 12.8 – Пример изменения длительности скана программы

Установка максимально допустимого времени цикла программы в регистре данных D8000 на значение 300 мс.

12.7 Повторение части программы, задание цикла (FOR, NEXT)

Начало повторения программы



Конец повторения программы

Программирование повторений программы (петля программы). Эти инструкции позволяют, чтобы часть программы между «FOR»- и «NEXT» повторялась «n» раз. После завершения «FOR» выполняется шаг программы после «NEXT»-инструкции. Значение «n» может находиться внутри следующей области значений: от +1 до +32 767. Если для «n» указано значение между 0 и -32 767, то петля «FOR-NEXT» отрабатывается только один раз. Можно программировать до пяти «FOR-NEXT»-уровней вложения.

Примечание:

«FOR»- и «NEXT»-инструкции могут программироваться только попарно. К каждой инструкции «FOR» должна программироваться соответственно «NEXT»-инструкция.

Источники ошибок:

В следующих случаях появляются ошибки в работе программы:

• «NEXT»-инструкция запрограммирована перед «FOR»-инструкции.

• «NEXT»-инструкция запрограммирована после «FEND»-инструкции или «END»-инструкции.

• Количество «NEXT»-инструкций не соответствует количеству «FOR»-инструкций.

Пример применения «FOR»- и «NEXT»-инструкций показан на рисунке 12.9.

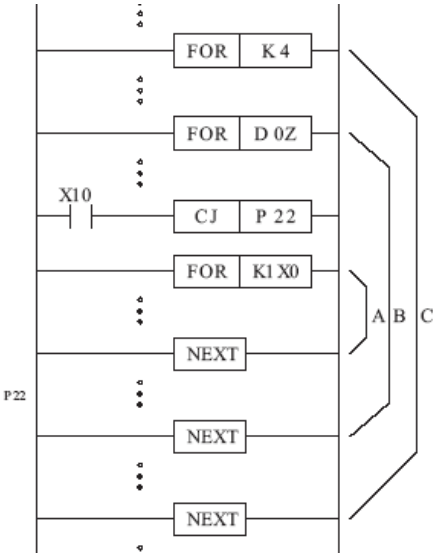


Рисунок 12.9 – Использование «FOR»-
и «NEXT»-инструкций

В примере запрограммированы три входящие друг в друга «FOR»- и «NEXT»-уровня вложения.

• Отрезок программы С обрабатывается четыре раза (здесь К4 константа). В конце обработки последний программный шаг выполняется после третьей «NEXT»-инструкции.

• При каждом исполнении отрезка С отрезок программы В обрабатывается шесть раз, если в регистре данных D0Z записано число 6.

• Поэтому отрезок В обрабатывается $6 \cdot 4 = 24$ раза.

• Если вход Х10 включен, то «FOR-NEXT»-петля (отрезок программы) пропускается (не обрабатывается) с помощью «CJ»-инструкции.

• Если вход Х10 выключен и содержание К1Х0 (блок К1 - первых 4 бита – в слове Х0) равно 7, то при каждом выполнении отрезка В отрезок программы обрабатывается семь раз.

• Поэтому отрезок А обрабатывается 168 раз ($6 \cdot 4 \cdot 7$).

12.8 Программирование STL-инструкций

12.8.1 STL-инструкция. Шаговое управление

STL-инструкция применяется совместно со статусом шага (операндами шаговых состояний S) и обеспечивает комфортное программирование шагового управления. Пример, объясняющий работу STL-инструкции, показан на рисунке 12.10.

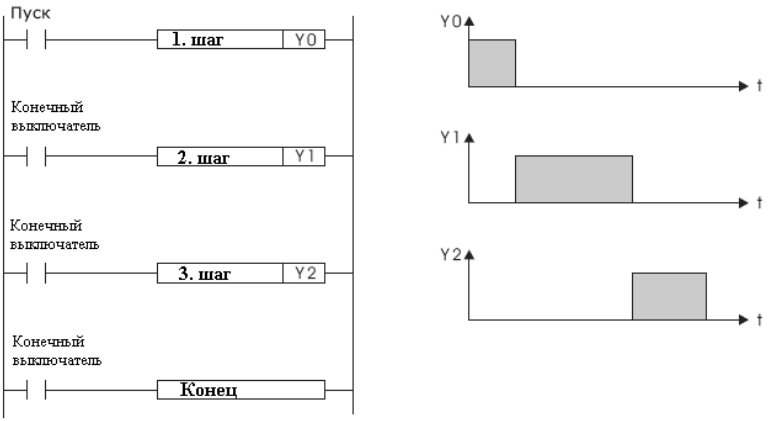


Рисунок 12.10 – Схематичный процесс шагового управления

Из рисунка 12.9 видно, что 2-й рабочий шаг включается, как только заканчивается 1-й шаг и включается соответствующий ему конечный выключатель. Это означает, что все состояния операндов внутри первого шага отключились. Конец 2-го шага означает одновременно пуск 3-го шага. С помощью включения 3-го конечного выключателя достигается окончание шаговой последовательности (4-й рабочий шаг).

Такое описание процесса очень удобно на практике, где часто приходится иметь дело с конечными состояниями различного технологического оборудования (робота, станка и т. п.).

12.8.2 Представление управления процессом в диаграмме блоков

На рисунке 12.11 представлена схема линейного управления процессом в диаграмме блоков – последовательных функциональ-

ных схемах, что соответствует представлению программы на языке SFC. Такое представление программы дает более наглядное понимание протекания процесса управления – процессов ветвления алгоритма и переходов в разные места программы.

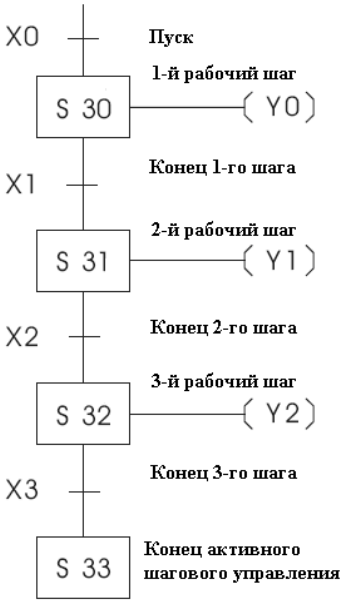
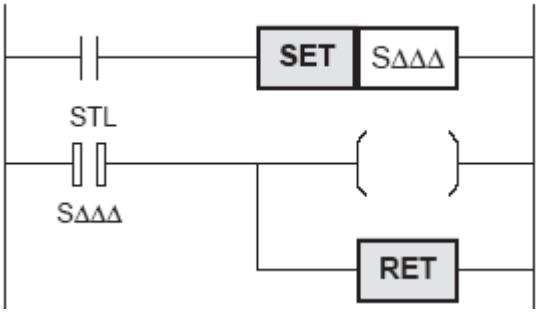


Рисунок 12.11 – Пример представления программы
в виде диаграммы блоков

На диаграмме представлено управление процессом упрощенно, независимо от позднейшей реализации программы ПЛК.

12.8.3 Программирование STL-инструкции



Как уже говорилось выше, STL-инструкция применяется для программирования управления процессом. Она применяется совместно с операндом шагов S, который может программироваться со следующими инструкциями набора базовых команд: LD, LDI, AND, ANI, OR, ORI, OUT, SET, RST. Внутри контактной схемы STL-контакт появляется на левой сборной («питающей») шине и может рассматриваться как «главный контакт», включающий на обработку стоящую в этой ветке часть программы. Как только STL-контакт отключается, следующая цепь не может больше обрабатываться. Область STL-программы (состояния шагов) обязательно заканчивается с помощью RET-инструкции.

Примечание:

▪ В программе без шагового управления шаговые операнды S могут применяться также как обычные внутренние реле.

▪ STL-инструкция не может применяться в программе прерывания.

▪ Не применяйте никаких инструкций переходов внутри шаговых состояний.

Пример программирования STL –инструкции показан на рисунке 12.12.

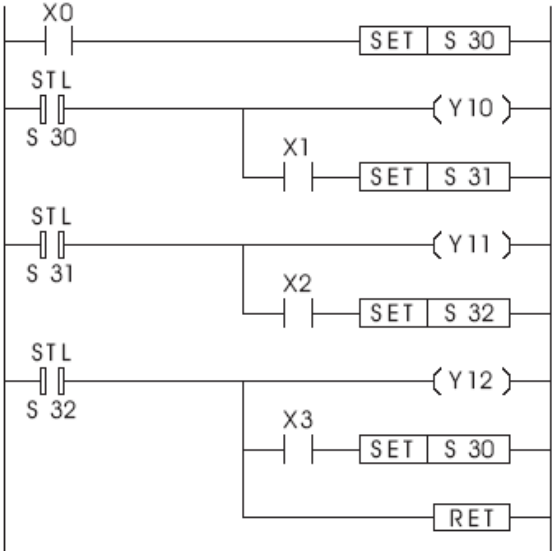


Рисунок 12.12 – Пример программирования STL-, RET-инструкций

Каждое состояние шага требует инициализации. Для этого имеются, например, инициализирующие операнды S0...S9 (шаг инициализации показывается на диаграмме двойной рамкой). С помощью инициализирующих операндов можно выполнить различные шаги процессов внутри STL-программы, чтобы реализовать, например, разные процессы работы (наладочный и автоматический режимы, подход к нулевой точке и т. д.).

Пример инициализация шаговых состояний показан на рисунке 12.13.

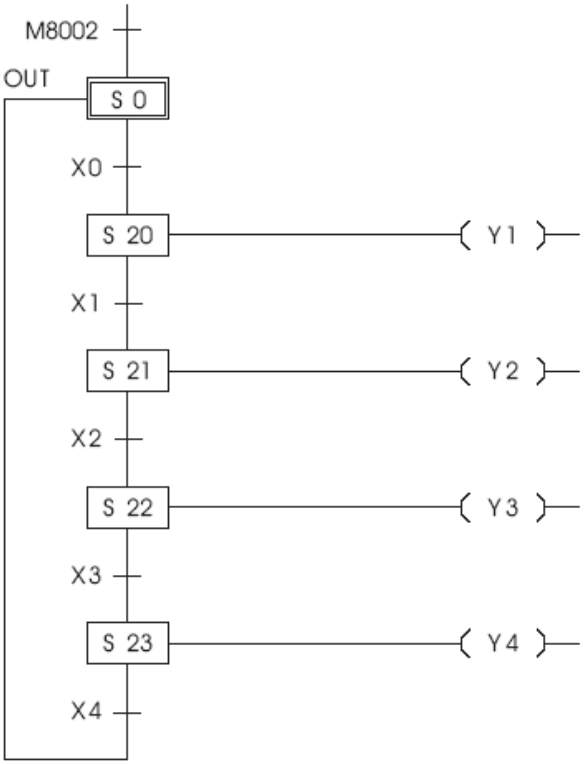


Рисунок 12.13 – Пример программирования инициализации шаговых состояний

Меркер M8002 (служебное внутреннее реле) задействует при включении ПЛК определенное системное состояние. Инициализация шаговой цепи определится включением S0. Условия шагов для каждого последующего шага выполняются уже описанным способом. Чтобы осуществить новый пуск или повторение шаговой цепи, снова должен включиться S0.

12.8.4 STL-разветвления

Программируемое управление на ПЛК могут обрабатывать различные, друг от друга независимые процессы состояний и разветвления. Нужно различать процессы:

- простой (линейный) процесс;
- селективное разветвление;
- параллельное (одновременное) разветвление;
- переходное разветвление.

12.8.4.1 Простой (линейный) процесс

При простом процессе шаговые состояния обрабатываются последовательно (один за другим). Последовательность обработки определяется только положением шагового состояния в простом процессе и благодаря независимости от адреса шагового состояния. На рисунке 12.14 и 12.15 показаны программа и временная диаграмма простого процесса.

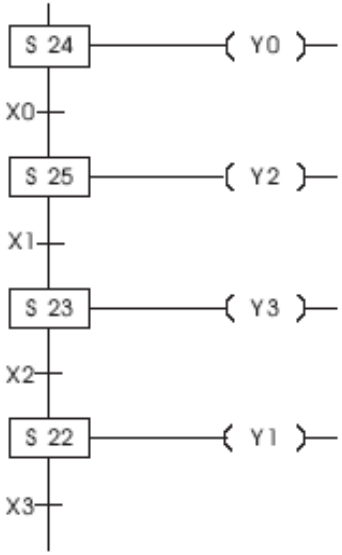


Рисунок 12.14 – Пример диаграммы блоков простого процесса

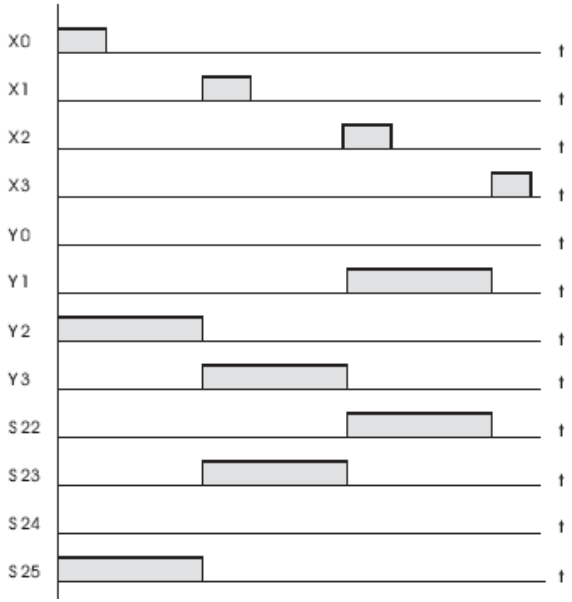


Рисунок 12.15 – Временная диаграмма простого процесса

12.8.4.2 Селективное разветвление

При селективном разветвлении имеется возможность произвести в этой операции выбор среди двух или более процессов состояний. Из одного шагового состояния разветвление может создавать несколько процессов состояний. В зависимости от соответственно примененных входных условий производится выбор, какой процесс состояний должен активизироваться в программе. Пример программирования селективного разветвления программы показан на рисунках 12.16 и 12.17.

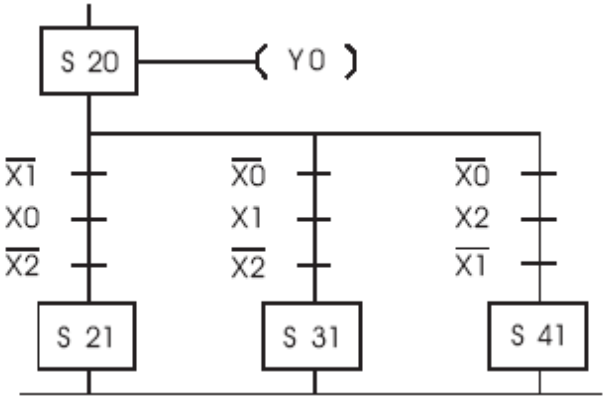


Рисунок 12.16 – Пуск (начало) селективного разветвления

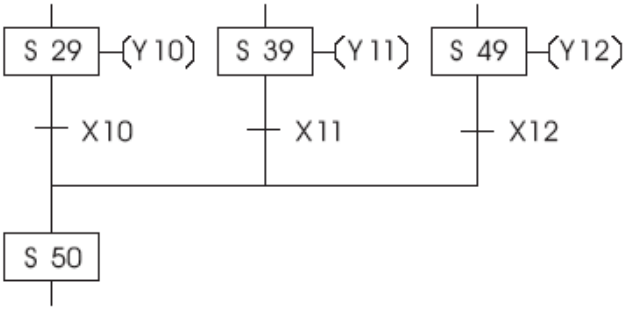
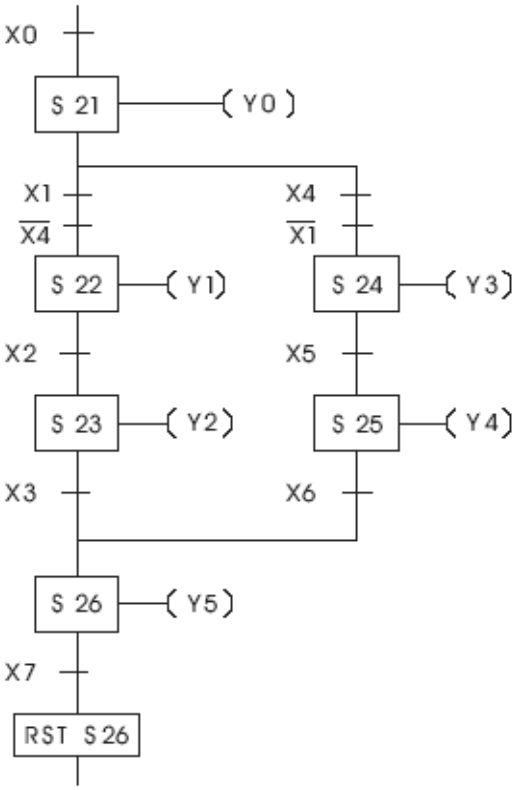


Рисунок 12.17 – Сборка (окончание) селективного разветвления

Примечание:

Для ПЛК семейства FX может программироваться максимум 8 разветвлений, выходящих из одного шагового операнда. Общее количество всех селективных разветвлений не должно превышать 16.

Пример, приведенный на рисунке 12.18, показывает запись разветвления программы ПЛК как в виде диаграммы блоков, так и в виде релейной схемы. Наглядность и удобство использования диаграммы блоков в этом случае очевидно.


X0
S 21
Y0
X1
X4
X4
X1
S 22
Y1
S 24
Y3
X2
X5
S 23
Y2
S 25
Y4
X3
X6
S 26
Y5
X7
RST S26

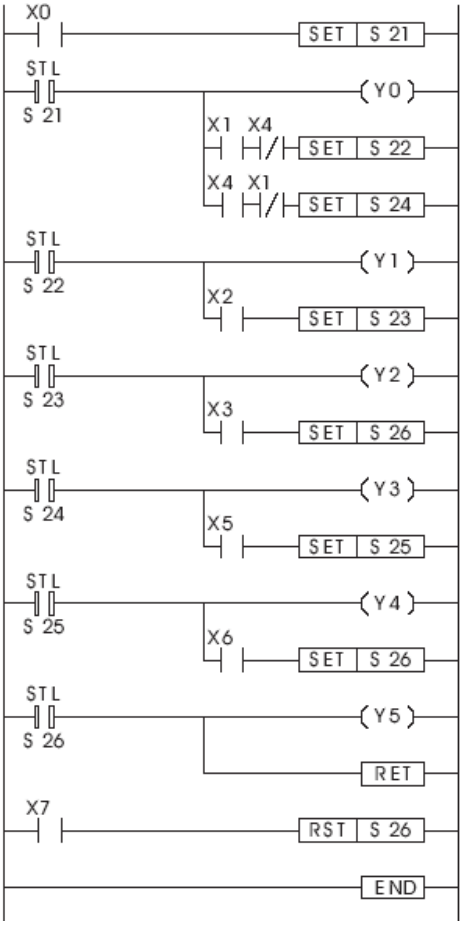


Рисунок 12.18 – Пример программирования
селективного разветвления

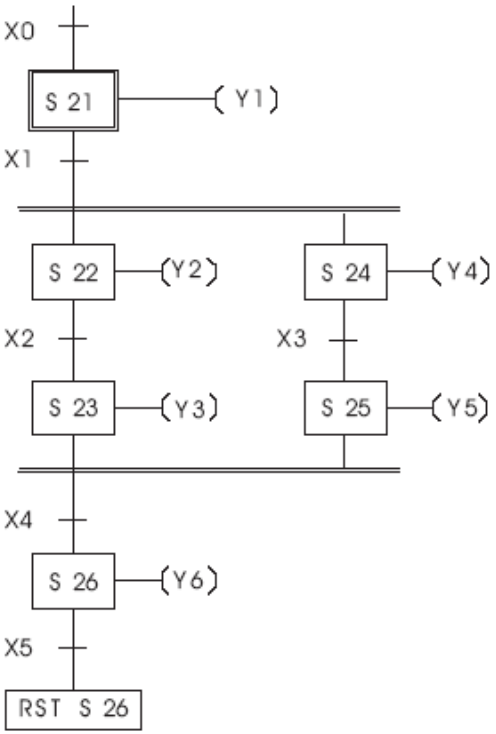
Здесь всегда можно выполнять только одну функцию. Это определяется тем, что S21 автоматически выключится, если не включится ни S22 ни S24. S26 включится по шагам S23 и S25. Соответственно этому при включении S26 отключается или S23 или S25.

12.8.4.3 Параллельное разветвление

При параллельном разветвлении два или несколько процессов состояний обрабатываются одновременно. Из одного состояния разветвление может создавать несколько (максимум 8) процессов состояний.

В зависимости от соответственно примененных входных условий выполняется разветвление на отдельные ветви. В противоположность к селективному разветвлению при параллельном разветвлении могут одновременно обрабатываться несколько процессов состояний. Включенные операнды параллельных шагов отключаются лишь тогда, когда обработаются шаги, лежащие после объединения параллелей.

Блочная диаграмма и контактная схема параллельного разветвления показаны на рисунке 12.19.



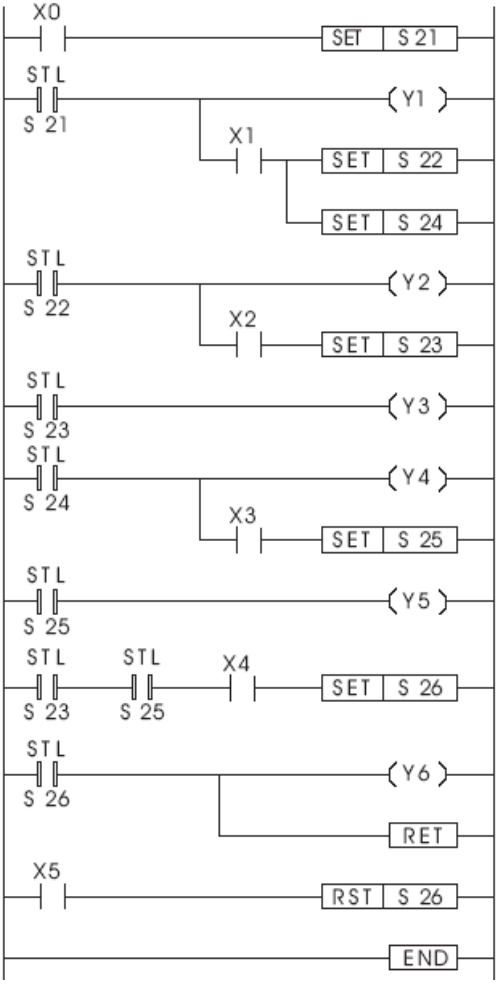


Рисунок 12.19 – Параллельное разветвление

Шаг S26 выполняется в зависимости от X4 лишь после выполнения шагов S23 и S25.

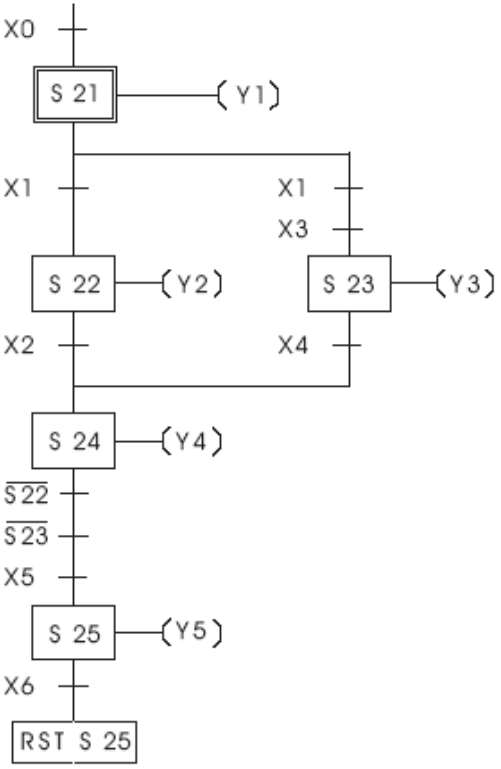
Примечание:

После разветвления (начало) и перед объединением (концом) не допустимы никакие логические связи.

Внутри параллельного разветвления нельзя программировать никаких селективных разветвлений, комбинированное применение таких разветвлений приведено в следующем пункте этой главы 12.8.4.4.

12.8.4.4 Комбинация из селективного и параллельного разветвления

Селективное и параллельное разветвление могут комбинироваться в одной программе STL следующим образом (рисунок 12.20).



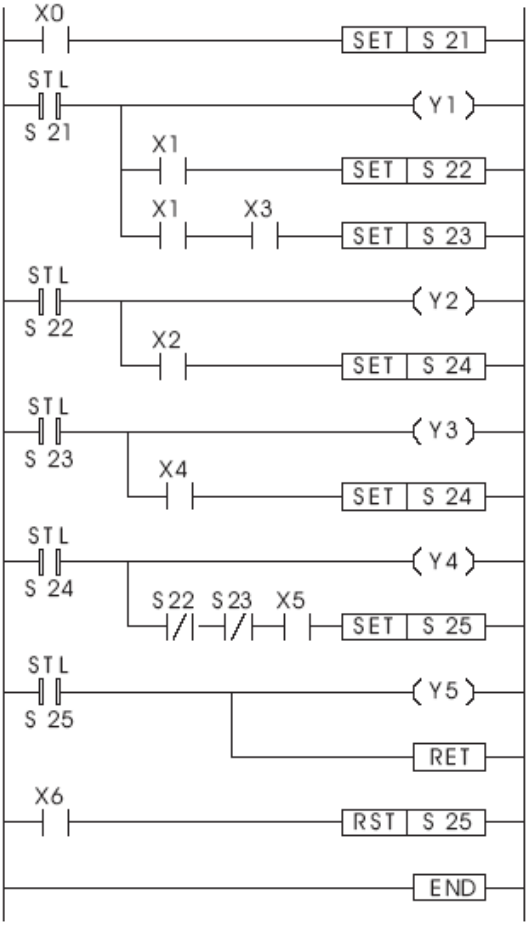
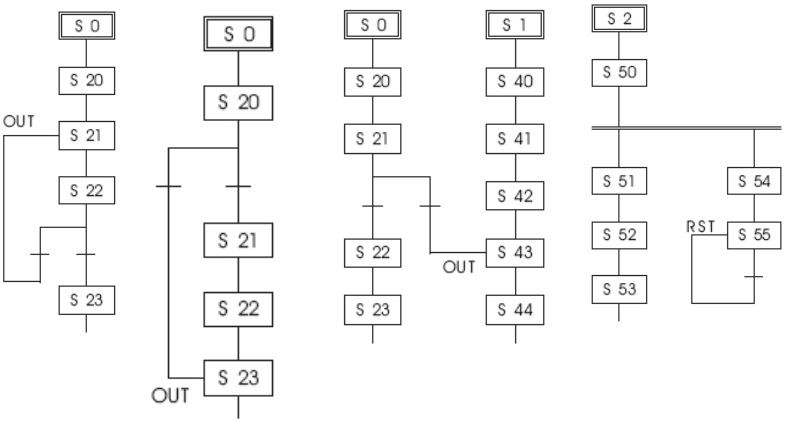


Рисунок 12.20 – Комбинация из селективного
и параллельного разветвления

Если в примере X3 включится, выполняться условия для параллельного разветвления. Если X3 не включится, выполнится селективная программа обработки, т. е. S24 сможет включиться только через S22. S24 включится только тогда, если S22 или S23 отключится. S25 включится только тогда, если S22 и S23 отключатся.

12.8.4.5 Переходное разветвление

Имеется возможность перескочить через часть области (схемы) последовательности состояния или многократно выполнить петлю программы. Пример таких переходов программы показан на рисунке 12.21.



Частичное повторение | Скачок | Скачок в другую STL-программу | Программирование возврата

Рисунок 12.21 – Пример программирования различных возможностей переходного разветвления

12.8.5 Примеры программ с использованием STL-инструкции

12.8.5.1 Пример контроля загрузки и выгрузки

В этом примере передвижной контейнер для транспортировки сыпучих грузов перемещается по жестко заданному отрезку и на определенных местах загружается и разгружается. Схема движения контейнера и его выгрузки показана на рисунке 12.22.

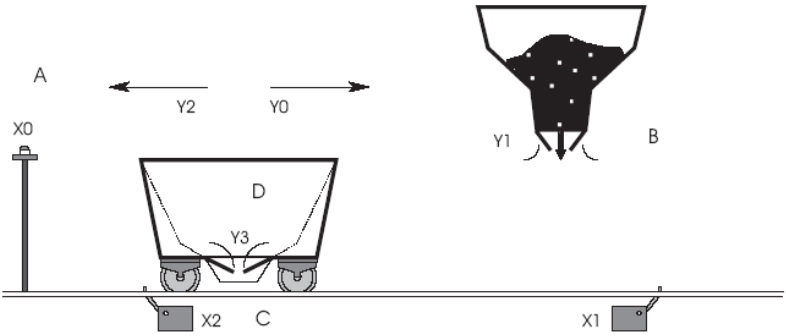


Рисунок 12.22 – Пример контроля загрузки и выгрузки контейнера

A. После воздействия на пусковую кнопку X0 тележка перемещается в направлении места загрузки и останавливается на конечном выключателе X1.

B. Загрузчик силоса открывается на семь секунд (Y1).

C. Тележка едет назад и останавливается на конечном выключателе X2 в месте разгрузки.

D. Клапан разгрузки тележки открывается на 5 секунд (Y3).

Ниже приведен пример программы работы тележки в двух видах – диаграмме блоков и релейно-контактной схемы (рисунок 12.23).


Пуск
Пусковая кнопка задействована?
S 21
Левый конечный выключатель включен ?
X20
X22
X23
Клапан разгрузки включен ?
S 22
Y20
Перемещение вперед
X21
S 24
Y21
Клапан силоса открыт
K70
T1
T1
S 22
Y22
Перемещение назад
X22
S 25
Y23
Разгрузочный клапан
K50
T2
T2

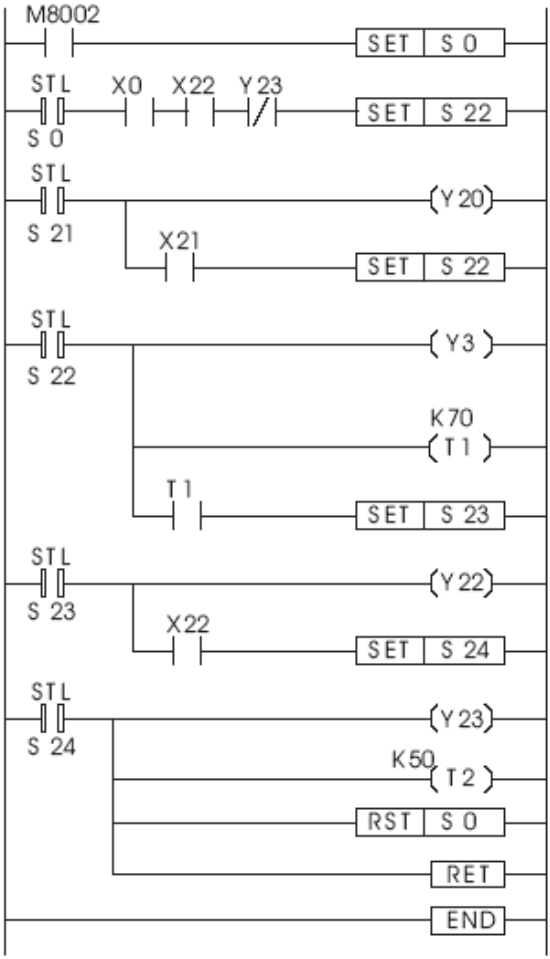


Рисунок 12.23 – Программа процесса контроля загрузки
и разгрузки контейнера

12.8.5.2 Пример транспортировки и сортировки

Этот пример отражает механизм управления, в котором разные по величине стальные шары поднимаются из одного лотка и

транспортируются по транспортеру. В конце транспортного пути шары сортируются в соответствующие сосуды в зависимости от их величины. Схема движения транспортера и перемещения шаров показана на рисунке 12.24.

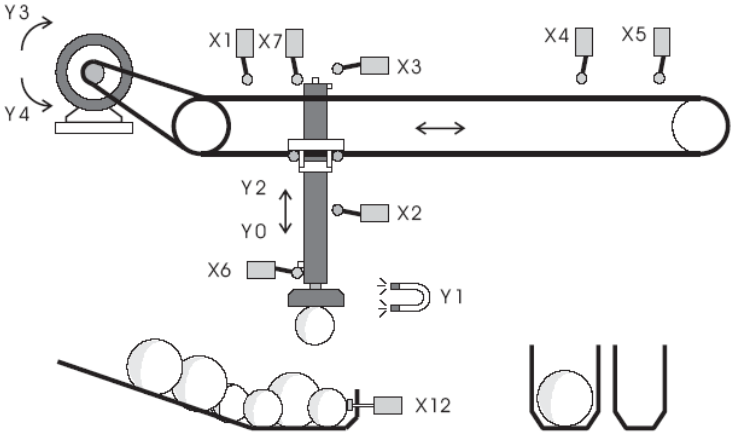


Рисунок 12.22 – Устройство сортировки шаров

A. Манипулятор опускается из исходного положения (Y0 = вкл)

B. Если нижняя граница не достигнута, то в шахте забора шара находится большой шар (X2=выкл, X6 = вкл). Контакт X2 замкнут, если в шахте забора не лежит малый шар.

C. Включается электромагнит (Y1 = вкл), и шар захватывается.

D. Манипулятор поднимается (Y2 = вкл). Рука останавливается при достижении верхней границы.

E. Манипулятор перемещается вправо (Y3 = вкл)

F. Если был взят малый шар, двигатель останавливается при достижении конечного выключателя X5. Если был взят большой шар, останов мотора выполняется при достижении конечного выключателя X4.

G. Манипулятор опускается (Y0 = вкл).

H. После достижения нижнего положения (X6) магнит выключается (Y1 = выкл).

I. Манипулятор поднимается до верхней границы (X3) (Y2 = = выкл).

J. Манипулятор перемещается в исходную позицию (Y4 = вкл).
K. Достигнута исходная позиция (X7 = вкл).

Пример программы работы устройства сортировки приведен на рисунке 12.25.

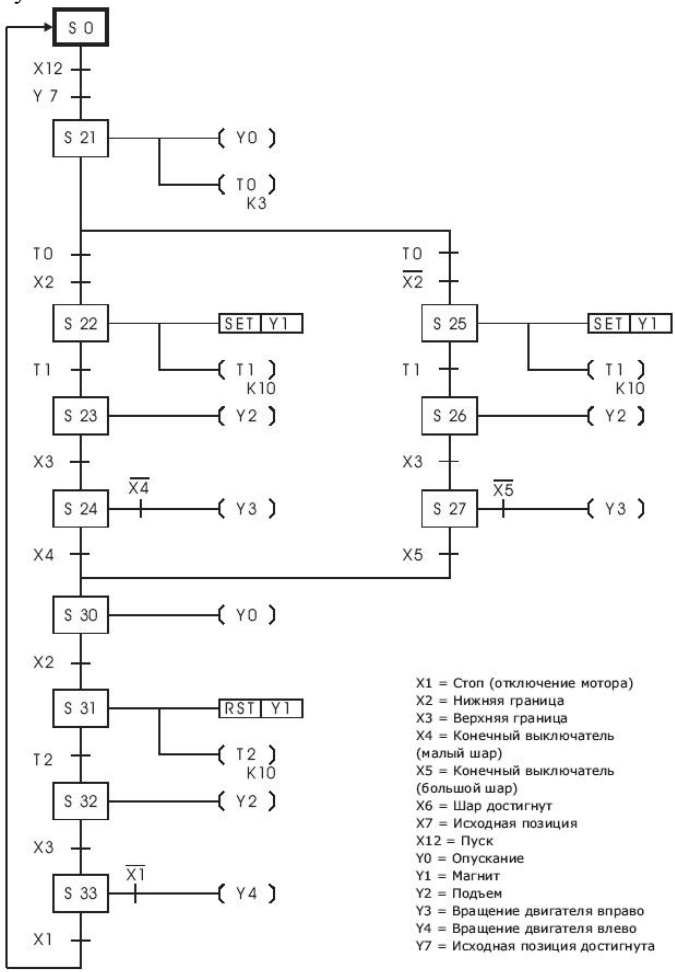


Рисунок 12.25 – Пример программирования устройства сортировки

Глава 13. Высокоскоростные инструкции

13.1 Обновление входов и выходов (REF)

Операнд (D) должен быть кратным 10: X0, X10, X20, и т. д. n должно быть кратным 8: 8, 16, 24, и т. д.

Обновление входов и выходов. Обработка программы у ПЛК серии FX выполняется по методу отображения процесса управления. Перед обработкой программы ЦП ПЛК считывает состояние сигналов входов и сохраняет их в специальной области памяти – регистрах отображения входов. Таким образом, обрабатываются не реальные входы, а регистр отображения входов.

После обработки программы считываются данные (записанные при обработке программы) из регистров отображения выходов и передаются на реальные выходы. С помощью «REF»-инструкции входы во время цикла работы программы могут опрашиваться и обновляться содержание регистра отображения. Можно применять «REF»-инструкцию, чтобы прочесть последнюю информацию входов, во время выполнения операции, а также можно выдавать результаты операции непосредственно после ее выполнения. Пример применения «REF»-инструкции показан на рисунках 13.1 и 13.2.

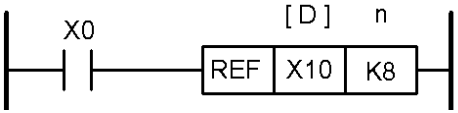


Рисунок 13.1 – Пример программирования «REF»-инструкции; обновление входов

Обновляется 8 адресов, т.е. входов X0...X7.

Если входы активированы примерно за 10 мс (время задержки) перед обработкой «REF»-инструкции, то входной регистр отображения активизируется, если выполнится «REF»-инструкция.

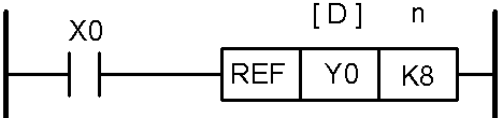
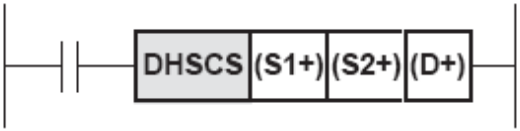


Рисунок 13.2 – Пример программирования «REF»-инструкции; обновление выходов

Обновляется 8 адресов, т.е. выходов Y0...Y7.

Если выходы включены, то включатся принадлежащие к ним регистры отображения выходов после выполнения «REF»-инструкции. Зажим реального выхода активизируется по истечению времени ответа. Время ответа является физически обусловленным временем включения активизированного выхода.

13.2 Использование высокоскоростного счетчика (DHSCS, DHSCR)



Включение от высокоскоростного счетчика

Отключение от высокоскоростного счетчика

Включение и отключение операндов от команды высокоскоростного счетчика. Операнды включаются или выключаются сразу по окончанию выполнения инструкции до окончания цикла программы. Высокоскоростной счетчик считает изменение состояния на входах счетчика в режиме прерывания. Каждому высокоскоростному счетчику определены жесткие входы счета с жестко определенными функциями.

С помощью «DHSCS»- и «DHSCR»-инструкции могут включаться / выключаться операнды от команды высокоскоростного счетчика. Операнд, записанный в (D+), включается / выключается, как только будет достигнуто установленное значение счета. Операнды включаются до окончания цикла программы непосредственно после выполнения инструкции. Инструкция выполняется, если данные в (S1+) согласованы с данными в (S2+). При этом активизация должна выполняться или по импульсу на счетном входе или на входе сброса.

Инструкция не выполняется, если согласование данных между (S1+) и (S2+) произведено посредством косвенного изменения данных в (S1+). Если, например, в S1+ находился регистр данных D0 и значения данных в D0 были изменены инструкцией «MOV», то высокоскоростная инструкция не выполнится. Пример применения «DHSCS»-, «DHSCR»-инструкций показан на рисунке 13.3.

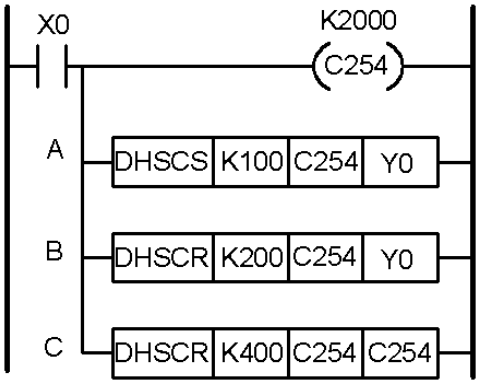


Рисунок 13.3 – Пример программирования «DHSCS»-, «DHSCR»-инструкций

Счетным входом высокоскоростного счетчика C254 является X0 (A-фаза) и X1 (B-фаза). Входом сброса является X2, а входом запуска X3.

A) Если накопленное значение счетчика C254 изменяется с 99 на 100 или со 101 на 100, то сразу включится Y0.

B) Если накопленное значение счетчика C254 изменяется с 199 на 200 или с 201 на 200, то сразу отключится Y0.

С) Если накопленное значение счетчика С254 изменяется с 399 на 400 или со 401 на 400, то сразу отключится счетчик С254.

Примечание:

В программе может использоваться не больше 6 «DHSCS» и «DHSCR» инструкций.

Выходы изменяются согласно их физическому времени включения.

Применение точки прерывания счетчика

Счетчик-прерывания может применяться как операнд для включения («HSCS»-инструкция) или отключения («HSCR» »-инструкция). Для отключения счетчика-прерывания нужно включать меркер М8059.

Пример программирования счетчика-прерывания точки-прерывания I030 показан на рисунке 13.4.

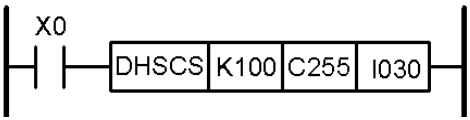
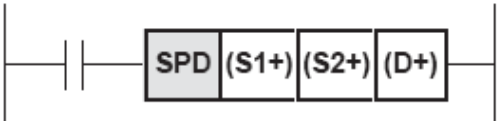


Рисунок 13.4 – Пример программирования счетчика-прерывания.

Программа-прерывания, вызванная точкой-прерывания I030, выполняется как только значение высокоскоростного счетчика С255 достигнет заданного значения по константе К100.

13.3 Определение скорости (SPD)



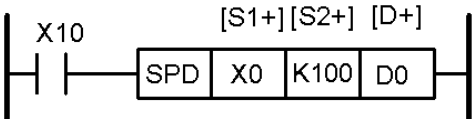
Команда выполняет фиксирование числа импульсов в течении заданного времени. Импульсы на (S1+) подсчитываются за время в (S2+) в мс и результат записывается в (D+). Задействуются операнды (D+), ((D+)+1), ((D+)+2).

(D+) – сумма импульсов после отсчета времени;

((D+)+1) – текущее значение времени внутри интервала времени;

((D+)+2) – остающееся отсчитываемое время.

Пример программирования «SPD»-инструкции с временной диаграммой показан на рисунке 13.5.



a: действительное текущее набираемое значение импульсов (D1)

b: накопленное значение счетчика за установленный интервал времени (D0)

c: остающееся в (D2) время до переброса данных из (D1) в (D0)

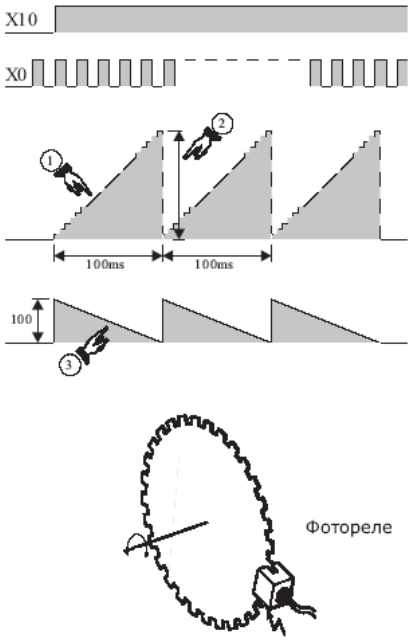


Рисунок 13.5 – Пример программирования «SPD»-инструкции

В примере D1 считает количество включений X0. После 100 мс результат счета сохраняется в D0.

D1 отключается и начинает вновь счет включений X0.

В D2 соответственно измеряется оставшееся время.

С помощью этого значения можно определить число оборотов привода.

$N = 60 \cdot D_0 \cdot 10^3$ (об/мин);

$n \cdot t$,

где n – число импульсов на оборот;

N – скорость;

t: интервал времени (мс), который указывается в S2+.

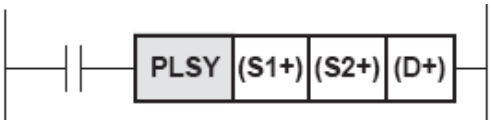
Примечание:

После отсчета времени содержание ((D+)+1) передается в (D+), а само ((D+)+1) отключается.

Входы высокоскоростного счетчика, используемые в инструкции, не могут применяться в других высокоскоростных операциях.

Для каждого высокоскоростного входа можно задать максимум одну «SPD»-инструкцию.

13.4 Выдача определенного числа импульсов (PLSY, DPLSY)



В (S1+) определяется частота.

В (S2+) указывается число создаваемых импульсов.

В (D+) определяется адрес выхода.

Команда осуществляет выдачу определенного числа импульсов с жестко заданной частотой и жестким соотношением ширины импульса 50:50.

Диапазон (S1+) в зависимости от модели ПЛК может быть:

1 ... 32767 Гц (FX1S/FX1N, PLSY);

1 ... 100 кГц (FX1S/FX1N, DPLSY);

2 ... 20 кГц (FX2N)

Диапазон значений (S2+):

16 битовые инструкции – от 1 до 32767 импульсов;

32 битовые инструкции – от 1 до 2 147 483 647 импульсов.

Если указано значение 0, то создается последовательный ряд импульсов.

Данные в (S1+) (частота) могут изменяться во время выполнения инструкции. Однако изменение данных в (S2+) (число импульсов) может применяться лишь тогда, если инструкция уже отработана. Пример применения «PLSY»-инструкции формирования группы импульсов и временная диаграмма процесса показаны на рисунке 13.6.

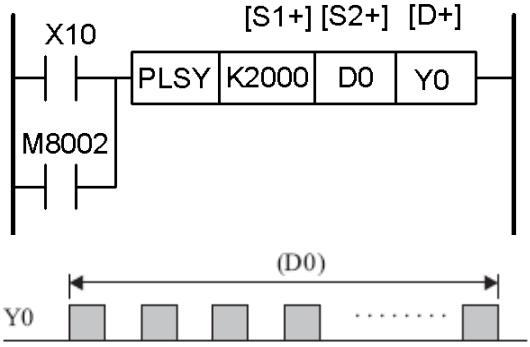


Рисунок 13.6 – Пример программирования «PLSY»-инструкции

Если включено X10, то создаются импульсы с частотой 2000 Гц. Всегда создается столько импульсов, сколько указано в регистре данных D0 (S2+).

Создание импульсов прекращается, если выключается X10. Если X10 включается снова, то операция начинается вновь. Если X10 не включено, то выключается Y0.

Примечание:

Включенное и отключенное состояния выдаются непосредственно в режиме прерывания.

При применении «DPLSY»-инструкции число импульсов задается в двух следующих друг за другом регистров данных.

Если желаемое число импульсов создано, то включается специальный меркер M8029 (инструкция полностью отработана). M8029 отключается, если деактивируется «DPLSY»-инструкция.

В программе допускается использование не более 2 инструкций PLSY.

Возможно применение одновременно инструкций PLSY и PLSR, если данные инструкции подключены к разным выходам.

Импульсы могут выдаваться только на выходах Y0 и Y1.

Возможно использование в подпрограммах нескольких инструкций PLSY, тем не менее должна остановиться запущенная команда, прежде чем в подпрограмме начинается следующая команда PLSY .

ПЛК должен иметь транзисторные выходы.

Для стабильной работы выходов, на максимальных частотах, ток нагрузки для ПЛК серии FX2N составляет минимум 200мА, для ПЛК серии FX1S-1N - от 10 до 100мА.

13.5 Выдача импульсов с модуляцией ширины импульса [ШИМ] (PWM)

В (S1+) устанавливается ширина импульса.

В (S2+) устанавливается продолжительность периода.

В (D+) указывается адрес выхода.

Команда формирует на выходе последовательный ряд импульсов с жестко заданной шириной импульса и продолжительностью периода.

t – ширина импульса (мс) (t = от 1 до 32767 мс);

$T0$ – продолжительность периода (мс) ($T0$ = от 1 до 32767 мс);

$f = 1/T0$ – частота (кГц).

Пример применения «PWM»-инструкции моделирования импульсов показан на рисунке 13.7.

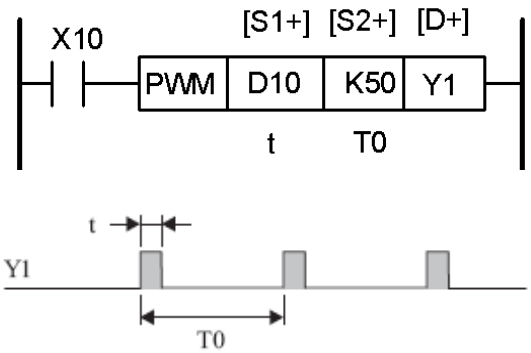


Рисунок 13.7 – Пример программирования «PWM»-инструкции

Благодаря изменению данных в регистре D10 в области от 0 до 50 можно варьировать относительную ширину импульса *T*0 от 0 до 100 %. Если значение D10 устанавливается на 0, то не выдается никаких импульсов. Если значение D10 изменяется на 50, то Y1 вводится для всего цикла. Y1 выключается, если выключается X10.

Примечание:

Ширина импульса должна лежать в пределах $1 < t < T0$.

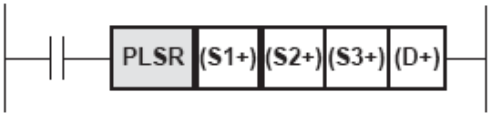
Инструкция может использоваться в программе только один раз.

Импульсы могут выдаваться только на выходы Y0 и Y1.

ПК должно иметь транзисторные выходы, чтобы избежать износа контактов.

Для стабильной работы выходов на максимальных частотах ток нагрузки для ПК серии FX2N составляет минимум 200мА, для ПК серии FX1S-1N - от 10 до 100мА.

13.6 Выдача определенного числа импульсов (PLSR)



«PLSR»-инструкция создает на выходе заданное число импульсов (S2+) с заданной частотой (S1+). Частота по десять шагов изменяется вверх в начале роботы инструкции (разгон) и вниз в кон-

це за заданное время (S3+). Для ПЛК серий FX1S или FX1N частота может быть в пределах от 10 до 100 000 Гц. Указанная частота должна делиться на 10. Если задаваемая частота не делится на 10, она округляется до соответствующего значения. Ширина шага наклонной составляет 1/10 указанной выходной частоты (при применении шагового двигателя это нужно учитывать).

Максимальное количество импульсов:

16-битовые инструкции: от 110 до 32.767 импульсов;

32-битных инструкций: от 110 до 2.147.483.647 импульсов.

Примечание:

Время подъема рампы (выхода на заданную частоту) должно соответствовать ниже описанным граничным значениям.

В качестве выходов могут программироваться только Y0 и Y1.

В одной программе в то же самое время могут применяться две «PLSR»-инструкции с передачей импульсов на Y0 и Y1. Возможно также применение «PLSY»-инструкции и PLSR-инструкции в одном цикле с передачей импульсов на Y0 и Y1. Многократное применение может реализовываться по подпрограмме или с помощью подобных методов.

При задании менее 110 импульсов правильная работа не гарантируется. Если число указанных импульсов недостаточно, частота срезается. Специальный меркер M8029 включается после выдачи указанного числа импульсов. Отключение меркера выполняется при отключении условий выполнения «PLSR»-инструкции.

Ограничение времени наклона

Время наклона (S3+) ограничивается 5 000 мс. Граничное значение времени наклона в зависимости от частоты и числа выходных импульсов рассчитывается следующим образом:

•Значение в (S3+) должно быть минимум в 10 раз больше, чем время цикла программы (D8012).

• Минимальное значение для (S3+) рассчитывается по уравнению: (S3+) = (9.000 / (S1+)) · 5.

• Максимальное значение для (S3+) рассчитывается по уравнению: (S3+) = ((S2+) / (S1+)) · 818.

• Если параметр выходит за расчетные границы, то значение (S1+) (частота) уменьшается.

• Подъем выходной частоты происходит в 10 шагов.

Пример применения «PLSR»-инструкции показан на рисунке 13.8.

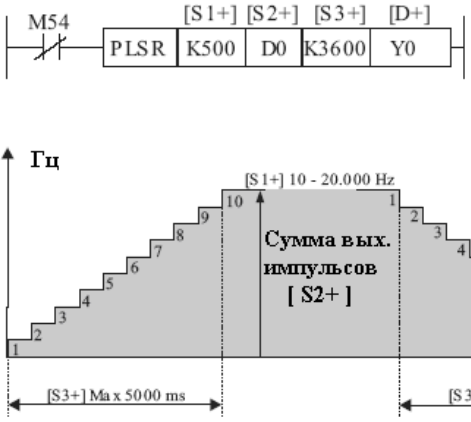


Рисунок 13.8 – Пример программирования «PLSR»-инструкции

При размыкании внутреннего реле M54 указанное в D0 (S2+) число импульсов выдается на Y0 (D+). Выходная частота составляет 500 Гц (S1+). Возрастание частоты до 500 Гц (S1+) и снижение частоты до 0 выполняется соответственно за 3600 мс (S3+) шагами по 50 Гц (S1+ / 10).

Глава 14. Регистры

Регистры представляют память данных внутри ПЛК. В регистре можно собирать числовые значения и следующую друг за другом двоичную информацию. Для этого возможно, например, состояние сигналов нескольких входов запомнить вместе и в программе обработать.

Данные сохраняются в 16-битном формате. Благодаря совместному включению (рассмотрению) двух 16-битных регистров можно образовать «двойной регистр». В двойном регистре данные хранятся в 32-битном формате.

14.1 Классификация регистров

Имеются следующие типы регистров:

▪ РЕГИСТР ДАННЫХ (не буферизован).

Регистр без сохранения данных при отключении напряжения ПЛК.

▪ РЕГИСТР ДАННЫХ (буферизован).

Регистр с сохранением данных при отключении напряжения ПЛК. Данные хранятся в энерго независимой памяти.

▪ ИНДЕКСНЫЙ РЕГИСТР.

Этот регистр служит для запоминания промежуточных результатов и для индифицирования операндов.

▪ СПЕЦИАЛЬНЫЙ РЕГИСТР.

Для определенных контрольных и проверочных функций предусмотрен ряд специальных регистров.

▪ РЕГИСТР ФАЙЛОВ.

Для сохранения параметров или рецептуры необходимы регистры файлов. Для ПЛК некоторых серий области памяти этих регистров устанавливаются пользователем. Эти регистры файлов являются частью регистров файлов с запоминанием. (Для хранения файлов используется основная память ПЛК, таким образом, сокращается возможное количество шагов основной программы.)

14.2 Структура регистра

Каждый регистр состоит из бита знака числа и нескольких битов данных. Структура регистра показана на рисунке 14.1.

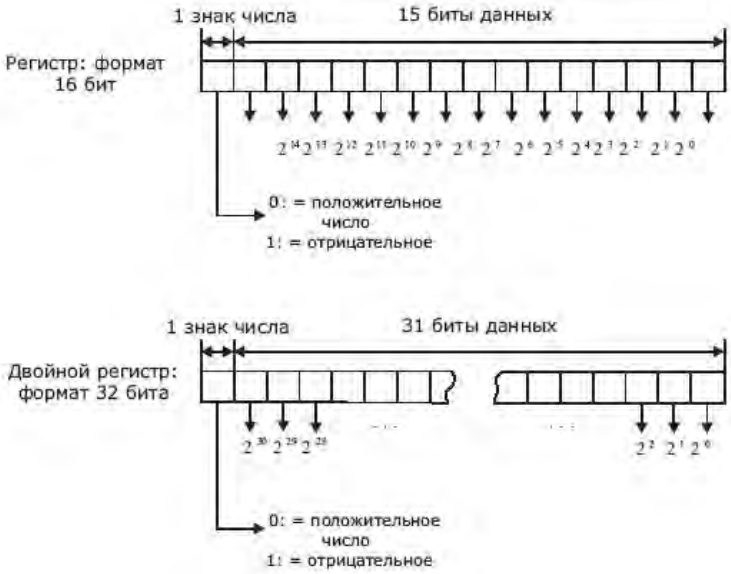


Рисунок 14.1 – Структура регистра (16 бит)
и двойного регистра (32 бита)

14.3 Применение индексных регистров

Индексные регистры применяются для того, чтобы для инструкций передачи и сравнения к адресам операндов добавить значение индекса.

Индексный регистр является 16-битовым регистром.

В 32-битовых инструкциях индексные регистры V (V0...V7) и Z (Z0...Z7) применяются комбинированно. Z содержит 16 младших бит, V запоминает 16 старших бит. В качестве адреса назначения указывается индексный регистр Z. Индексный регистр не может самостоятельно индицироваться. Пример передачи данных от регистра данных D5V к регистру данных D10Z показан на рисунке 14.2.

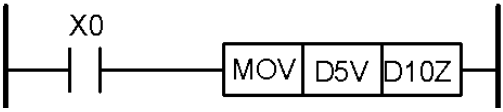


Рисунок 14.2 – Пример схемы передачи данным от регистра D5V к регистру D10Z

Расчет адреса выхода D5V:

V = 8

5 + 8 = 13 → D13

Расчет адреса пересылки D10Z:

Z = 14

10 + 14 = 24 → D24

Следовательно, имеет место пересылка данных от регистра данных D13 к регистру данных D24.

14.4 Применение регистров файлов

Регистры файлов записываются в блоках 500 адресами в программной области памяти (EPROM или EEPROM) ПЛК. Количество блоков устанавливается в параметрах. Доступ к регистрам файлов возможен через программаторы и терминалы обслуживания.

Если используются регистры файлов, то для программы ПЛК сокращается полезная область памяти. Для каждого блока, состоящего из 500 регистров файлов, сокращается количество полезных программных шагов примерно на 500 шагов. Количество регистров файлов варьируется в зависимости от типа ПЛК.

Чтение регистров файлов происходит во время работы ПЛК посредством «BMOV»-инструкций. Что касается записи, то в ПЛК регистры файлов могут записываться только с программатора, с персонального компьютера с соответствующим программным обеспечением, а также могут изменяться в программе ПЛК при применении «BMOV»-инструкции.

Примечание:

Изменение данных регистра файлов возможно в «RUN»-РЕЖИМЕ только для «RAM»-регистров или для регистров файлов во внутренней памяти. Регистры файлов, которые находятся в оперативной памяти (RAM), во внутренней памяти или на EEPROM кассе-

те памяти, могут изменяться в «STOP»-РЕЖИМЕ. Регистры файлов, которые находятся на EPROM кассете памяти, изменяться не могут.

14.5 Регистры данных

Области значений.

Если в регистре запоминаются числа в двоичном коде, то область значений чисел ограничивается на базе граничных величин регистров.

• Десятичные числа
16 бит: -32 768 ... +32 767;
32 бит: -2 147 483 648 ... +2 147 483 647.
• Шестнадцатеричные числа
16 бит: 0 ... FFFF;
32 бит: 0 ... FFFFFFFF.

ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

Отрицательные числа представляются как 2-е дополнение. При образовании 2-го дополнения инвертируется двоичное число (1-е дополнение) и прибавляется двоичное значение числа 1.

Пример:
0101101 (двоичное) → +45 (десятичное)
1010010 (двоичное) → 1-е дополнение
1010011 (двоичное) → 2-е дополнение
1010011 (двоичное) → -45 (десятичное)

Регистр данных заносит значение негативным, если в высшем разряде бита (знак числа) стоит число 1. Пример хранения чисел в регистрах данных показан на рисунке 14.3.

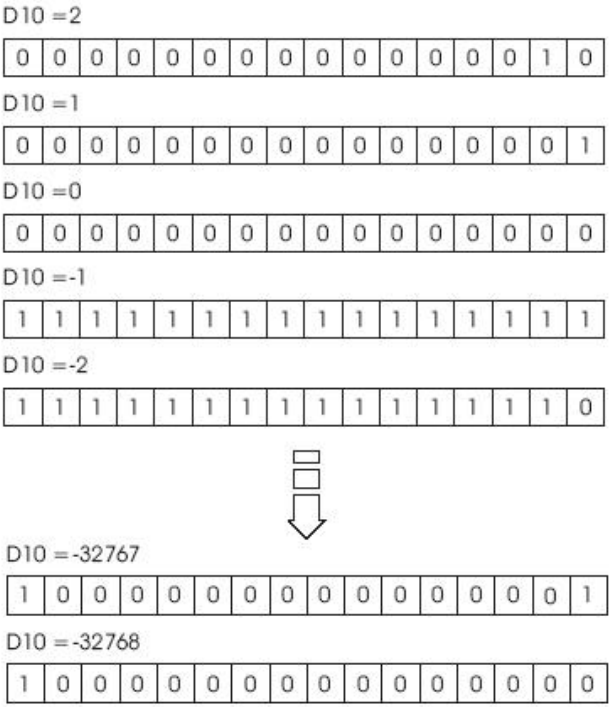


Рисунок 14.3 – Пример хранения чисел в регистрах

14.6 Представление чисел

ПЛК имеют, как правило, возможность работать со значениями чисел в следующих представлениях (форматах):

- десятичные числа,
- числа в научном формате,
- числа с плавающей запятой,
- двоичные числа,
- шестнадцатеричные числа,
- формат числа в двоично-десятичном коде (BCD),
- битовый пример.

Десятичная система чисел
База: 10
Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Область значений:
– 16-битный формат: -32 768 ... +32 767
– 32-битный формат: -2 147 483 648 ... +2 147 483 647

Пример:
351 (десятичное) = $3 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0$

Числа в научном формате

Этот формат рассчитан на научное представление очень больших и очень малых чисел. Представление выполняется в 32-х битном формате с плавающей запятой.

Формат: Мантисса $\cdot 10^{\text{Экспонента}}$

Пример
Скорость света:
– как десятичное число: 299 792 458 м/с
– в научном формате: $2998 \cdot 10^5$ м/с

Здесь 2998 является мантиссой и 5 – экспонентой. В регистре данных число сохраняется, например, в форме D120 $\cdot 10^{D121}$ (рисунок 14.4).

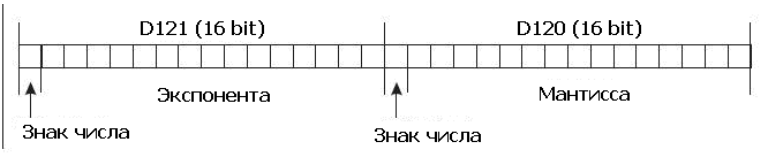


Рисунок 14.4 – Расположение числа в научном формате в регистре данных

Система чисел с плавающей запятой

Операции с числами очень быстро превышают допустимые значения областей, серия FX предлагает дополнительное представление очень больших и очень малых чисел в формате с плавающей запятой, как это применяется в персональных и микрокомпьютерах.

Формат системы чисел с плавающей запятой запоминает мантиссу и экспоненту как двоичные числа в 32-битовых двойных словах, где мантисса имеет 23 бита, а экспонента 8 бит (рисунок 14.5).

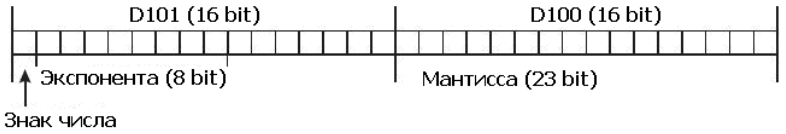


Рисунок 14.5 – Расположение числа с плавающей запятой в регистре данных

Формат: $\pm$Мантисса $\cdot$ $2^{\text{Экспонента}}$

Пример:

D101 = 16592 = 40D0HEX

D100 = 0 = 0000HEX

Размещение бит в регистре данных (мантиссы и экспоненты) показано на рисунке 14.6.

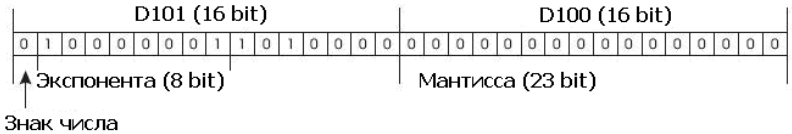


Рисунок 14.6 – Размещение бит в регистре данных

Бит знака числа равен 0 – положительное значение.

Экспонента записана числом 10000001, это соответствует:

$(1 \cdot 2^7 + 0 \cdot 2^6 + ... + 1 \cdot 2^0) - 127 = (128 + 0 + ... + 1) - 127 = 2$;

Мантисса представлена числом 10100000000000000000000 – это соответствует:

$1{,}101_2$ или $1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + ... + 0 \cdot 2^{-23} = 1{,}625$.

В результате число равно $+ 1{,}625 \cdot 2^2 = 6{,}5$.

Двоичная система чисел

База: 2

Цифры: 0, 1

Пример:

11001 (двоичное число) $= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

11001 (двоичное число) = 16 + 8 + 1 = 25 (десятичное)

Двоичное кодирование: десятичное число → двоичное число

Пример:

30 (десятичное)
30 : 2 = 15 остаток 0
15 : 2 = 7 остаток 1
7 : 2 = 3 остаток 1
3 : 2 = 1 остаток 1
1 : 2 = 0 остаток 1
30 (десятичное) = 11110 (двоичное)

Кодирование: двоичное число → десятичное число

Пример:

111000 (двоичное) $= 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$
111000 (двоичное) = 32 + 16 + 8 = 56 (десятичное)

Восьмеричная система счисления

База: 8
Цифры: 0, 1, 2, 3, 4, 5, 6, 7

Пример:

245 (восьмеричное) $= 2 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 128 + 32 + 5 =$
$=$ 165 (десятичное)

Кодирование: десятичное число → восьмеричное число

Пример:

30 (десятичное)
30 : 8 = 3 остаток 6
3 : 8 = 0 остаток 3
30 (десятичное) = 36 (восьмеричное)

Кодирование: восьмеричное число → десятичное число

Пример:

374 (восьмеричное) $= 3 \cdot 8^2 + 7 \cdot 8^1 + 4 \cdot 8^0 = 192 + 56 + 4 =$
$=$ 252 (десятичное)

Шестнадцатеричная система счисления

База: 16
Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
(A=10, B=11, C=12, D=13, E=14, F=15)

Пример:

1E (шестнадцатеричное) = $1 \cdot 16^1 + 14 \cdot 16^0 = 16 + 14 = 30$ (десятичное).

Кодирование: десятичное число → шестнадцатеричное число

Пример:

63 (десятичное)

63 : 16 = 3 остаток 15 R F (шестнадцатеричное)

3 : 16 = 0 остаток 3 R 3 (шестнадцатеричное)

63 (десятичное) = 3F (шестнадцатеричное)

Кодирование: шестнадцатеричное число → десятичное число

Пример:

7A (шестнадцатеричное) = $7 \cdot 16^1 + 10 \cdot 16^0 = 112 + 10 = 122$ (десятичное).

BCD - формат (двоично-десятичный код)

В «BCD»-формате (двоично-десятичном коде) каждая цифра десятичного числа представляется четырехбитным двоичным числом. При четырехбитном представлении имеется возможность двоично кодировать десятичные цифры от 0 до 15. Однако в «BCD»-формате допустимо кодирование только десятичных цифр от 0 до 9.

Кодирование: десятичное число → BCD - формат

Пример:

67 (десятичное) <=> Цифры: 6, 7.

Глава 15. Работа с регистрами с помощью языка релейно-контактных схем (LD)

15.1 Основные команды

Для правильного выполнения команд с регистрами необходимо обращать особое внимание на формат данных в этих регистрах, иначе результат может быть неверным.

15.1.1 Передача данных. Команды (MOV) и (DMOV)

Команда служит для передачи данных от источника данных (S+) к данным цели (D+). Данные в источнике данных (S+) при выполнении «MOV»-команды автоматически интерпретируются как двоичные значения. Пример применения «MOV»-команды (рисунок 15.1).

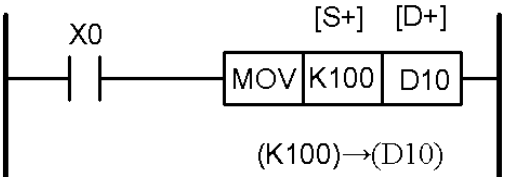


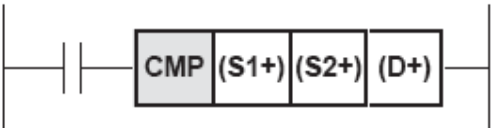
Рисунок15.1 – Пример программирования с использованием MOV-инструкции

Если входное условие X0 включено, то выполняется переход данных от (S+) к (D+). Если X0 выключен, не выполняется никакой переход. Константа K100 при выполнении «MOV»-команды автоматически интерпретируется двоичным значением (рисунок 15.1).

Примечание:

Инструкции выполняются в каждом цикле программы. Этого можно избежать благодаря использованию вставленной впереди импульсной функции («PLS»- или «PLF»-инструкции или же параметра «P»).

15.1.2 Сравнение числовых данных. Команды (CMP) и (DCMP)



Команды выполняют сравнение между двумя числовыми значениями данных (больше, меньше, равно) по следующему принципу:

▪ Данные в обоих источниках (S1+) и (S2+) сравниваются друг с другом.

▪ Результат сравнения (больше, меньше, равно) отображается (индицируется) с помощью внутреннего реле M, операнда состояния шага S или выхода Y. Определение, какой из этих операндов должен задействоваться, выполняется по адресу цели (D+).

(S1+) > (S2+) (D+)
(S1+) = (S2+) ((D+)+1)
(S1+) < (S2+) ((D+)+2)

▪ Данные в S1+ и S2+ обрабатываются как двоичные данные.

Пример применения «CMP»-команды показан на рисунке 15.2.

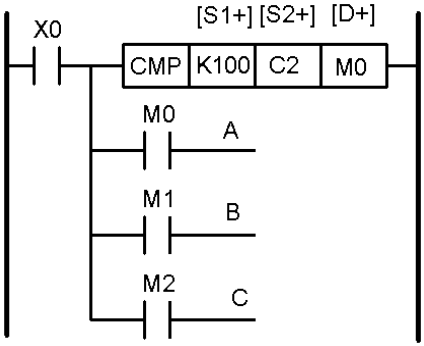


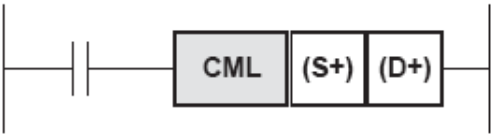
Рисунок 15.2 – Схема использования команды CMP

В адресе цели (D+) в этом примере указано внутреннее реле M0. Соответствующие результаты сравнения автоматически присваиваются приращиваемым на 1 последующих адресов внутренних реле M0, M1, M2 и имеют следующие значения:

- A) M0 включено, если K100 > накопленного в счетчике C2 значения;
- B) M1 включено, если K100 = накопленному в счетчике C2 значению;
- C) M2 включено, если K100 < накопленного в счетчике C2 значения.

M0, M1, M2 не изменяются, если входное условие X0 выключено.

15.1.3 Копирование и инвертирование. Команда (CML)



Функция этой команды – образование 1-го дополнения двоичного числа. Двоичное значение числа в (S+) преобразовывается в свое 1-е дополнение и записывается в данные цели (D+). Пример применения «CML»-инструкции (релейная схема и регистры) показан на рисунке 15.3.

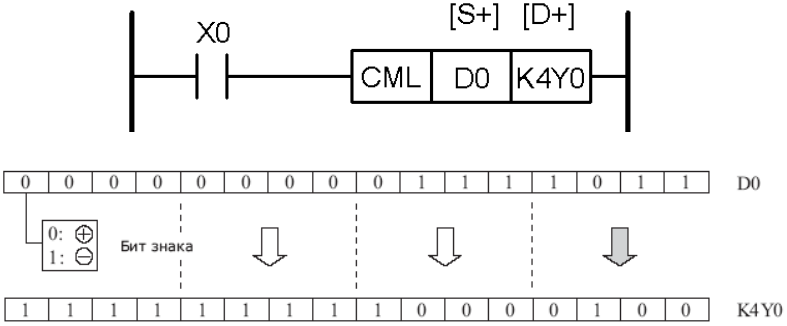


Рисунок 15.3 – Схема использования команды «CML» (инвертирование и перенос). Значения бит в регистрах D0 и K4Y0 при выполнении команды «CML»

Примечание:

Если адрес цели располагает большим числом бит, чем адрес источника, то все неиспользуемые биты включаются.

15.1.4 Обмен данными. Команда (XCH)

Команда предназначена для осуществления обмена данными между двумя операндами (регистрами). Обмениваются данные в регистрах (D1+) и (D2+). Пример применения «XCH»-команды показан на рисунке 15.4.

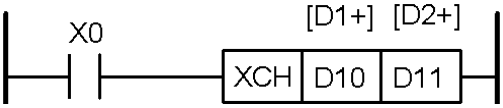


Рисунок 15.4 – Пример записи схемы команды «XCH» (обмена данными)

Значения перед выполнением инструкции:
D10 = 5, D11 = 7
Значения после выполнения инструкции:
D10 = 7, D11 = 5

Примечание:

Процесс обмена выполняется в каждом цикле, если не программируются никакое управление по фронту.

15.1.5 Сдвиг регистра вправо/влево. Команды (ROR) и (ROL)

Команды выполняют сдвиг регистра на n мест вправо «ROR» / влево «ROL». Битовое отображение в регистре (D+) сдвигается вправо/влево на n мест при каждом исполнении одной из команды.

Состояние последнего сдвигаемого бита копируются в флаг передачи – «Carry». Пример применения команд «ROR» и «ROL» показаны на рисунках 15.5 и 15.6 соответственно:

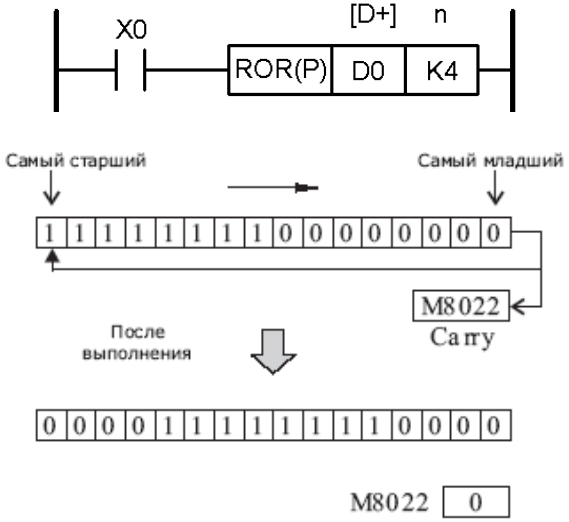


Рисунок 15.5 – Сдвиг регистра вправо

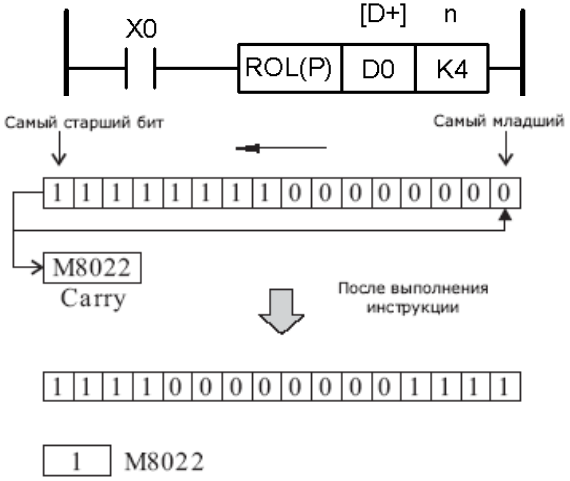


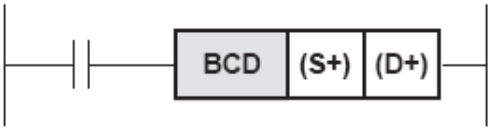
Рисунок 15.6 – Сдвиг регистра влево

Битовые данные в регистре данных D0 каждый раз сдвигаются вправо/влево на 4 бита (K4), когда вход X0 переходит из состояния ОТКЛ. в состояние ВКЛ. Значение последнего сдвигаемого бита запоминается в флаге передачи (рисунки 15.5, 15.6).

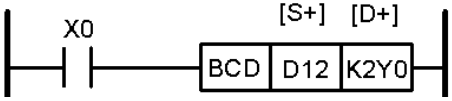
Примечание:
Если не программируется никакого опознания фронта, то сдвиг битового отображения повторяется в каждом цикле программы.

Для различных моделей ПЛК может существовать целый ряд дополнительных расширенных и специальных инструкций для работы с регистрами, количество которых зависит от конкретной серии и модели ПЛК. Для получения детального описания возможностей ПЛК необходимо обращаться к инструкции пользователя.

15.1.6 Двоично-десятичное преобразование. Команды (BCD) и (DBCD)



Команды выполняют конвертирование двоичных данных в «BCD»-формат (двоично-десятичный). Обычно внутри ПЛК обрабатываются только двоичные данные. Благодаря применению «BCD»-команды могут выдаваться данные также в «BCD»-формате (например, для управления семисегментным устройством отображения). Двоичные данные источника (S+) конвертируются в «BCD»-данные и передаются по адресу цели (D+). Пример выполнения «BCD»-команды приведен на рисунке 15.7.



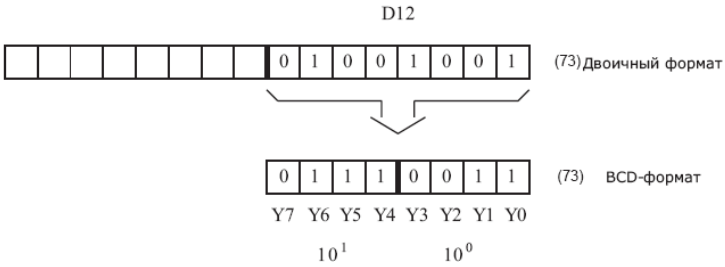


Рисунок 15.7 – Пример преобразования двоичных данных в «BCD»-формат

Двоичные данные из регистра данных D12 конвертируются в «BCD»-формат и затем выдаются по выходам Y0...Y7 (рисунок 15.7). В этом примере: 73 (десятичное число).

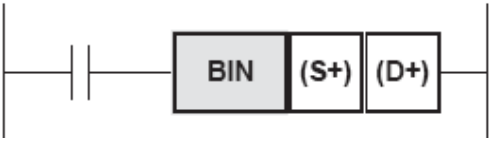
Примечание:

Важно, чтобы результат «BCD»-конвертирования находился в допустимой области:

- 16-битовая-инструкция (BCD): от 0 до +9 999;
- 32-битовая-инструкция (DBCD): от 0 до +99 999 999.

Если результат «BCD»-конвертирования находится вне допустимой области, то появляется ошибка обработки программы и команда не выполняется.

15.1.7 Двоичное преобразование. Команды (BIN) и (DBIN)



Команда «BIN» выполняет конвертирование из «BCD»-данных в двоичный формат. Это опять же связано с тем, что внутри ПЛК обрабатываются только двоичные данные. С помощью «BIN»-команды по входам могут считываться данные также в «BCD»-формате. «BCD»-данные источника (S+) конвертируются в двоичные данные и передаются по адресу цели (D+).

Пример применения команды конвертирования (рисунок 15.8).

«BCD»-данные на входах X0...X7 конвертируются в формат двоичных данных по адресу цели D13 и затем выдаются на выходы Y0...Y7. В этом примере 73 – десятичное число.

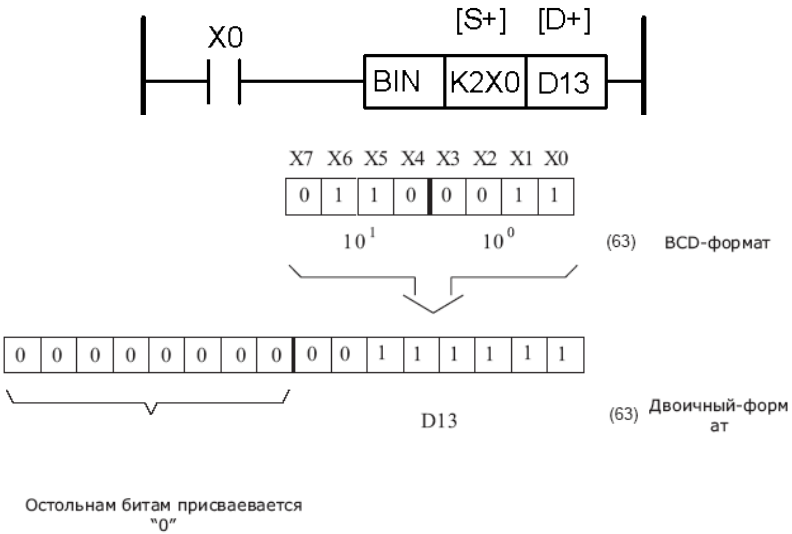


Рисунок 15.8 – Пример преобразования данных в «BCD»-формате в двоичную форму

Примечание:

Данные в (S+) должны находиться внутри допустимой области:

• 16-битовая-инструкция (BIN): от 0 до +9 999;

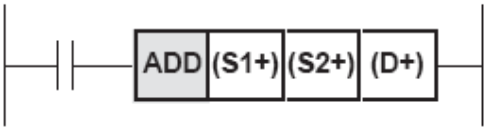
• 32-битовая-инструкция (DBIN): от 0 до +99 999 999.

Если данные в (S+) не в «BCD»-формате, появляется ошибка.

Конвертирование данных из одного формата в другой играет очень важную роль, так как именно несоответствие типов данных является наиболее частым источником ошибок. Выше были рассмотрены команды конвертирования «ходовых» форматов, Остальные команды конвертирования смотри в инструкции пользователя или инструкции по программирования контроллера.

15.2 Арифметические инструкции

15.2.1 Сложение числовых данных. Команды (ADD) и (DADD)



Команда «ADD» выполняет сложение двух числовых данных. Результат сложения хранится по адресу цели. В адресе источника (S+) и адресе цели (D+) нужно также указывать одинаковые типы операндов. Двоичные данные в адресах источников (S1+) и (S2+) суммируются, при этом результат суммирования запоминается в адресе цели (D+). Таким образом: (S1+) + (S2+) = (D+). В старшем бите запоминается знак числа суммирования (0 – знак положительного числа, 1 – знак отрицательного числа). Пример применения «ADD»-команды показан на рисунке 15.9.

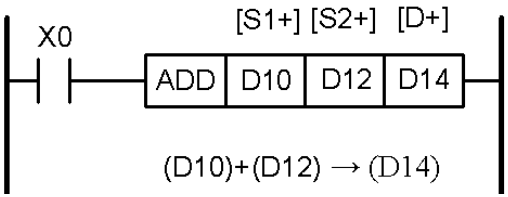


Рисунок 15.9 – Пример применения «ADD»-команды

Если включен X0, то суммируются значения данных в регистрах D10 и D12. Результат суммирования запоминается в регистре данных D14.

Примечание:

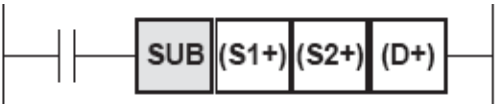
При определенных результатах счета после исполнения команды включается специальное внутреннее реле (флаг).

- Если результатом сложения является 0, включается флаг нуля.
- Если результатом сложения явилось число меньше -32 767 (16-битовая операция) или же -2 147 483 648 (32-битовая операция), включается флаг заимствования.

▪ Если результатом сложения явилось число выше +32 767 (16-битовая операция) или же ±2 147 483 647 (32-битовая операция), включается флаг переноса.

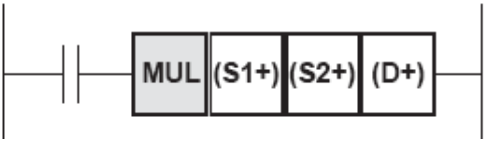
При выполнении 32-битной инструкции (DADD) в ней указывается операнд слова младших 16-бит. Следующий за ней операнд является операндом слова старших 16-бит. Рекомендуется при задании адреса применять четные числа, чтобы не запрограммировать по ошибке наложение адресов.

15.2.2 Вычитание числовых данных. Команды (SUB) и (DSUB)



Команда «SUB» выполняет вычитание двух числовых данных. Результат вычитания хранится по адресу цели. Все полностью аналогично предыдущей функции – (см. выше) команда сложения «ADD».

15.2.3 Умножение числовых данных. Команды (MUL) и (DMUL)



Команда «MUL» выполняет умножение двух числовых данных. Результат умножения хранится по адресу цели. Данные в S1+ и S2+ перемножаются между собой. Результат умножения запоминается по адресу операнда указанного в D+ и в следующем за ним адресе операнда: (S1+) x (S2+) = (D+) . В старшем бите запоминается знак результата перемножаемых чисел (0 – знак положительного числа; 1 – знак отрицательного числа).

Умножение 16-битных данных (MUL-команда).

Результат 16-битного умножения оказывается 32-битным числом. Это число запоминается как 32-битное значение. Младшие 16-бит записываются по адресу операнда, заданному в (D+). Стар-

шие 16 бит записываются по следующему за ним адресу операнда. Пример применения «MUL»-инструкции показан на рисунке 15.10.

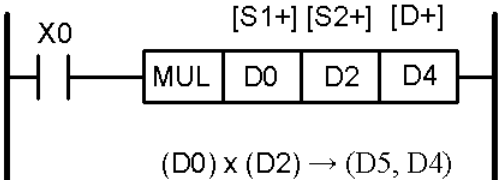


Рисунок 15.10 – Пример применения «MUL»-команды

Результат умножения записывается как 32-битное значение данных в регистры данных D4 и D5. В D4 стоят младшие 16 бит, а в D5 – старшие 16 бит. Знак числа стоит в 15-м бите D5.

Умножение 32-х битных данных (DMUL-команда).

Результат 32-х битного умножения запоминается 64 значением данных. Младшие 16 бит запоминаются по адресу операнда, заданному в (D+). Старшие биты записываются по следующему за ним адресу операнда. Пример применения «DMUL»-инструкции показан на рисунке 15.11:

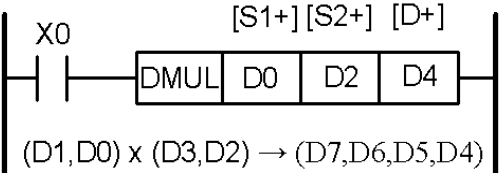


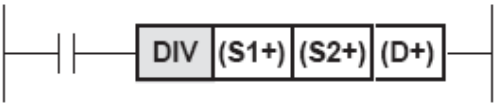
Рисунок 15.11 – Пример применения «DMUL»-команды

Результат умножения записывается как 64-х битное значение данных в регистры данных D4, D5, D6 и D7. В D4 стоят младшие 16 бит, а в D5, D6 и D7 – старшие биты.

Примечание:

При 32-битном операнде для (D+) нельзя применять Z(V).

15.2.4 Деление числовых данных. Команды (DIV) и (DDIV)



Команда «DIV» выполняет деление двух числовых данных. Результат деления хранится по адресу цели. Выполняется деление данных в (S1+) и (S2+). Данные в (S1+) соответствуют делимому, в (S2+) – делителю. Результат деления запоминается по адресу операнда указанного в (D+) и в следующем адресе операнда. Остаток делимого запоминается в следующем адресе операнда: (S1+) : (S2+) = = (D+). В старшем бите запоминается знак числа результата деления (0 – знак положительного числа; 1 – знак отрицательного числа). При работе программы появляется ошибка, если значение делителя равно 0 и команда не обрабатывается.

Деление 16-битных данных (DIV-команда).

Результат 16-битного деления запоминается по адресу операнда в заданном D+. Остаток от деления запоминается в следующем за ним адресе операнда. Пример применения «DIV»-команды показан на рисунке 15.12.

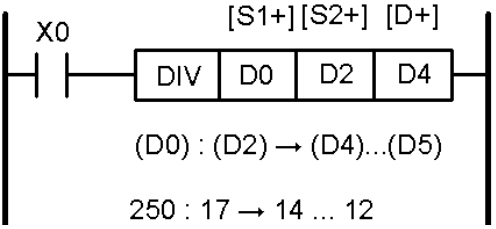


Рисунок 15.12 – Пример применения «DIV»-команды

Результат деления 14 записывается в регистр данных D4. Остаток от деления 12 записывается в следующий регистр данных D5.

Деление 32-битных данных (DDIV-команда).

При делении 32-битных данных для делимого, делителя, результата и остатка от деления имеется соответственно по два следу-

ющих друг за другом регистра данных. В «DDIV»-команде должны указываться соответственно регистры данных с нижними адресами операндов. Пример применения «DDIV»-команды показан на рисунке 15.13.

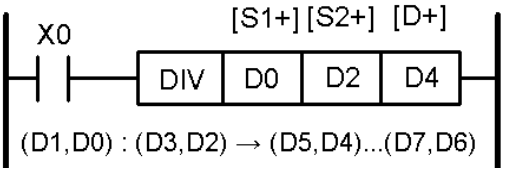
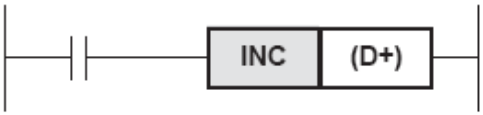


Рисунок 15.13 – Пример применения «DDIV»-команды

Результат деления записывается в регистры данных D4 и D5. Остаток от деления записывается в следующие регистры данных D6 и D7.

При 32-битном операнде для (D+) нельзя применять Z(V).

15.2.5 Команда приращения (INC) и (DINC)



Команда выполняет приращение к значению числа, имеющемуся в D+, – прибавляется число 1, как только выполнится входное условие. Команда выполняется в каждом цикле программы. Этого можно избежать благодаря введению впереди функций импульса («PLS»- или «PLF»-команды) или применив командный параметр «P» (для MELSEC FX). Пример применения «INC»-команды показан на рисунке 15.14.

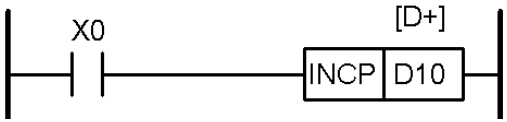


Рисунок 15.14 – Пример применения «INC»-команды

Значение данных в регистре данных D10 при наличии входного сигнала X0 повысится на число 1. Инструкция активизируется благодаря подключенной впереди функции импульса. Это важно, чтобы процесс суммирования не выполнялся в каждом цикле программы.

Примечание:

16-битовый операнд (INC-команда)

Если при 16-битовом операнде значение 1 добавится к +32 767, то запишется значение - 32 768. Не появляется никакого флага.

32-битовый операнд (DINC-команда)

Если при 32-битовом операнде значение 1 добавится к числу +2 147 483 647, то запишется значение -2 147 483 648. Не появляется никакого флага.

15.2.6 Команда уменьшения (DEC) и (DDEC)

Команда вычитает 1 от числового значения данных, находящихся в регистре (D+). Аналогично команде приращения, команда уменьшения выполняется в каждом цикле программы.

Примечание:

16-битовый операнд (DEC-команда)

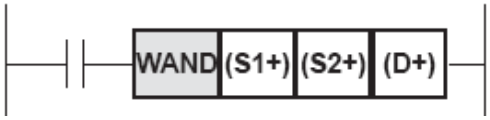
Если при 16-битовом операнде значение 1 отнимется от числа -32 768, то запишется значение +32 768. Не появляется никакого флага.

32-битовый операнд (DDEC -команда)

Если при 32-битовом операнде значение 1 отнимется от числа -2 147 483 648, то запишется значение +2 147 483 647. Не появляется никакого флага.

15.3 Логические инструкции

15.3.1 Логическая связь «И». Команды (WAND) и (DAND)



Команда «WAND» осуществляет логическую связь «И» двоичных данных, которая выполняется по отдельным битам. Данные в (S1+) и (S2+) побитно логически связываются друг с другом, согласно таблице истинности 15.1. Результат связи сохраняется в (D+). Пример применения «И»-связи показан на рисунке 15.15.

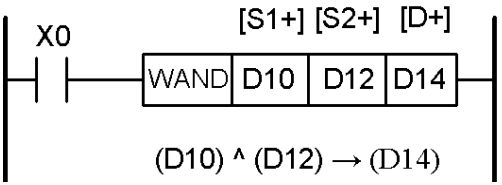


Рисунок 15.15 – Пример применения «WAND»-команды

Таблица 15.1 – Таблица истинности «И»

(S1+)	(S2+)	(D+)
1	1	1
1	0	0
0	1	0
0	0	0

Состояние регистров D10, D12, D14 при выполнении команды логической «И»-связи показано на рисунке 15.16.

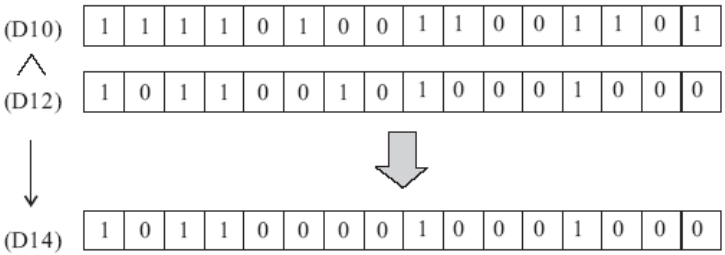
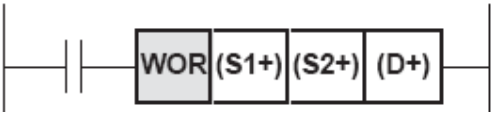


Рисунок 15.16 – Состояние регистров при выполнении команды «И»-связи

15.3.2 Логическая связь «ИЛИ». Команды (WOR) и (DWOR)



Команда «WOR» работает аналогично команде «WAND», осуществляя логическую связь «ИЛИ». Данные в (S1+) и (S2+) побитно логически связываются друг с другом, согласно таблице истинности 15.2. Результат связи сохраняется в (D+). Пример применения «ИЛИ»-связи показан на рисунке 15.17.

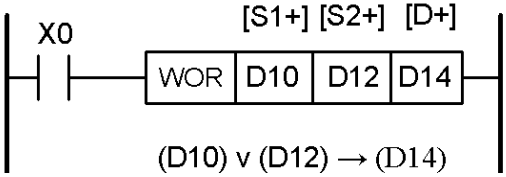


Рисунок 15.17 – Пример применения «WAND»-команды

Таблица 15.2 – Таблица истинности «ИЛИ»

(S1+)	(S2+)	(D+)
1	1	1
1	0	1
0	1	1
0	0	0

Состояние регистров D10, D12, D14 при выполнении команды логической «И»-связи показано на рисунке 15.18:

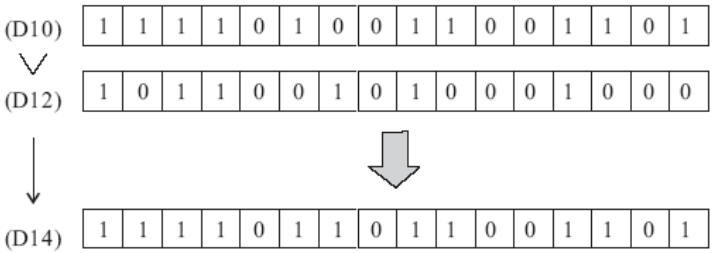
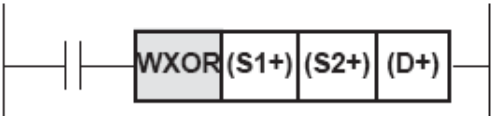


Рисунок 15.18 – Состояние регистров при выполнении команды «ИЛИ»-связи

15.3.3 Логическая связь «исключающее или». Команды (WXOR) и (DXOR)



Команда «WXOR» осуществляет логическую связь «ИСКЛЮЧАЮЩЕЕ ИЛИ» двоичных данных, которая выполняется по отдельным битам. Данные в (S1+) и (S2+) побитно логически связываются друг с другом, согласно таблице истинности 15.3. Результат связи сохраняется в (D+). Пример применения логической связи «ИСКЛЮЧАЮЩЕЕ ИЛИ» показан на рисунке 15.19.

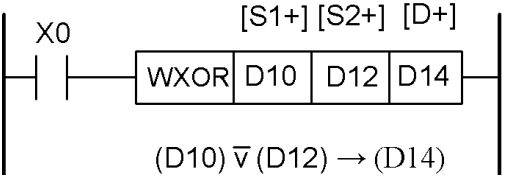


Рисунок 15.19 – Пример применения «WXOR»-команды

Таблица 15.3 – Таблица истинности «ИСКДЮЧАЮЩЕЕ ИЛИ»

(S1+)	(S2+)	(D+)
1	1	0
1	0	1
0	1	1
0	0	0

Состояние регистров D10, D12, D14 при выполнении команды логической связи «ИСКЛЮЧАЮЩЕЕ ИЛИ»- показано на рисунке 15.20.

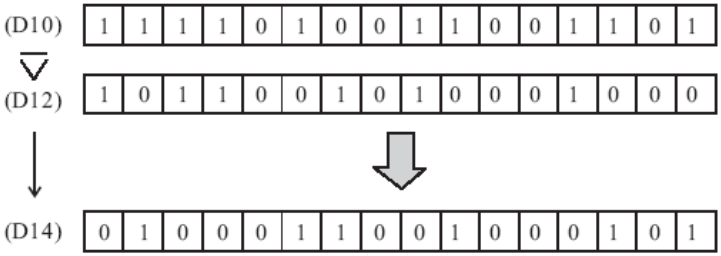


Рисунок 15.20 – Состояние регистров при логической связи “ИСКЛЮЧАЕЮЩЕЕ ИЛИ”

15.3.4 Инверсия данных. Команда (NEG)

Команда «NEG» образует второе дополнение значения данных, записанных в (D+), и оно сохраняется в том же (D+). Пример применения «NEG»-команды показан на рисунке 15.21.

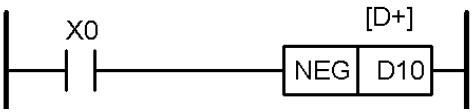


Рисунок 15.21 – Пример применения «NEG»-команды

Двоично: D10 + 1 D10

Состояние регистра D10 при выполнении «NEG»-команды показано на рисунке 15.22.

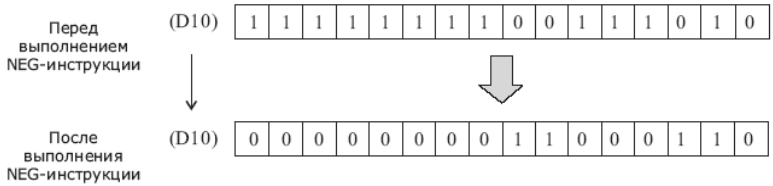


Рисунок 15.22 – Состояние регистра при команде «NEG»

Примечание:

Если не программируется никакого опознания фронта, то образование дополнения повторяется в каждом цикле программы.

Глава 16. Рекомендации по проектированию системы с ПЛК

Кажущаяся простота ПЛК может привести к ложному выводу о том, что его использование не требует тщательной подготовки. Но это далеко не так. Общеизвестно, что легкость достигается решением весьма сложных задач.

Может создаться положение, когда такой совершенный аппарат, как ПЛК, при неправильном использовании станет менее эффективным и более дорогим, чем более простое устройство.

Есть много методов проектирования систем с микроконтроллером. Этот раздел предоставляет некоторые общие рекомендации, которые могут применяться ко многим процессам проектирования. В таблице 16.1 указаны некоторые основные шаги процесса проектирования.

Таблица 16.1 – Основные шаги процесса проектирования системы на базе ПЛК

*	Расчленение процесса или машины на простые компоненты
*	Создание функциональной спецификации модулей.
*	Проектирование аппаратных схем защиты.
*	Задание станций операторов.
*	Разработка чертежей конфигурации ПЛК.
	Создание списка соглашений о символических именах сигналов - (необязательно)

Расчленение процесса или машины

Разложите ваш процесс или машину на сегменты, не зависящие друг от друга. Эти сегменты определяют границы между контроллерами и влияют на описания функциональных областей и назначение ресурсов.

Описание функциональных областей

Приведите описания работы каждого сегмента процесса или машины. Включите следующие пункты:

- Входы/выходы.
- Описание функционирования.
- Условия деблокировки (состояния, которые должны достигаться перед разрешением действия) для каждого исполнительного механизма (соленоиды, двигатели, приводы и т. д.).
- Описание интерфейса оператора.
- Интерфейсы с другими сегментами процесса или машины.

Проектирование схем защиты

Определите оборудование, требующее для обеспечения безопасности аппаратно-реализованной логики. Устройства управления могут выходить из строя опасным образом, вызывая неожиданный запуск или изменение в работе машинного оборудования. Там, где неожиданная или неправильная работа машинного оборудования может привести к физической травме людей или значительному материальному ущербу, нужно уделить внимание использованию электромеханических блокировок, которые работают независимо от

ПЛК, чтобы предотвратить опасные операции. В проектирование схем защиты должны включаться следующие задачи:

- Выявление ненадлежащей или неожиданной работы исполнительных механизмов, которая может оказаться опасной.
- Определение состояний, которые гарантировали бы, что работа не опасна, и выяснение того, как обнаруживать эти состояния независимо от ПЛК.
- Определение влияния ПЛК посредством его входов/выходов на процесс при подаче и выключении питания и при обнаружении ошибок. Эта информация должна использоваться только для проектирования нормального и ожидаемого аварийного режимов работы и не должна использоваться для целей безопасности.
- Проектирование ручных или электромеханических блокировок, которые блокируют опасную операцию независимо от ПЛК.
- Предоставление в ПЛК надлежащей информации о состоянии от независимых цепей тока, чтобы программа и любые интерфейсы оператора имели необходимую информацию.
- Определение любых других связанных с безопасностью требований для безопасного протекания процесса.

Задание станций оператора

Основываясь на требованиях из описаний функциональных областей, разработайте чертежи станций оператора. Включите следующие пункты:

- Обзор, показывающий местоположение каждой станции оператора относительно процесса или машины.
- Механическая компоновка устройств станции оператора (дисплей, переключатели, лампы и т.д.).
- Электрические чертежи ПЛК или модулей расширения с соответствующими входами-выходами (схемы подключения).

Разработка чертежей конфигурации

Основываясь на требованиях из описаний функциональных областей, разработайте чертежи конфигурации аппаратуры управления. Включите следующие пункты:

- Обзор, показывающий местоположение каждого ПЛК относительно процесса или машины.
- Механическая компоновка ПЛК и модулей расширения входов-выходов (включая стойки и другое оборудование).

▪ Электрические чертежи для каждого ПЛК и модуля расширения входов-выходов (включая номера моделей устройств, коммуникационные адреса и адреса входов-выходов).

Разработка списка символических имен

Если вы используете для адресации символические имена, то разработайте список символических имен для абсолютных адресов. Включите не только сигналы физических входов-выходов, но также и другие элементы, которые использованы в вашей программе.

Глава 17. Примеры программ

17.1 Штамповочная машина

Задача – Запрограммировать контроллер, управляющий штамповочной машиной. Написать часть программы, отвечающей за выход готового изделия и подготовку матрицы к следующему циклу. Схема конвейера показана на рисунке 17.1.

Данный пример демонстрирует работу только части кода программы отвечающей за:

• Открытие формы (перемещение штампа вправо);

• Очистка матрицы (подача воздуха и перемещение из точки A (5 с) в точку B (5 с);

• Смазка матрицы (подача смазки и перемещение из точки A (5 с) в точку B (5 с).

Примечание:

Остальной код, описывающий ход технологического процесс по штамповке изделия, в настоящем примере не представлен.

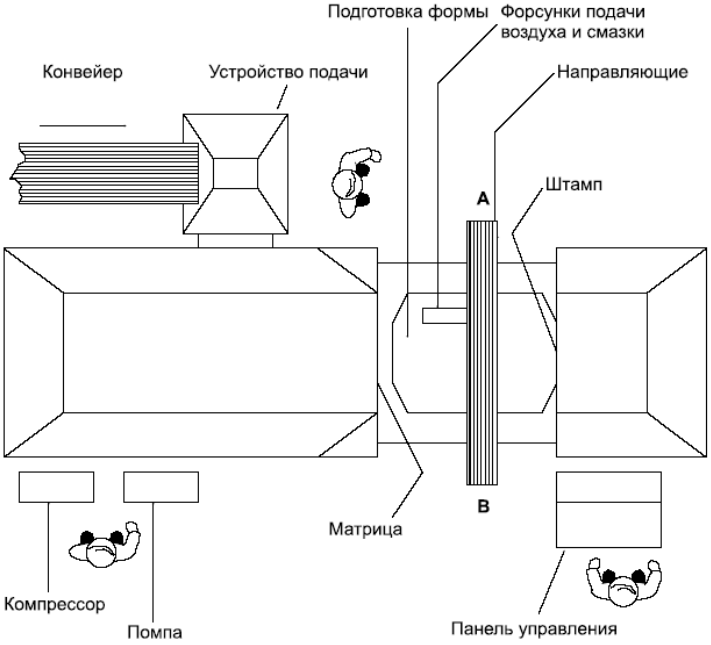


Рисунок 17.1 – Схема конвейера

Рассмотрим алгоритм управляющей программы (рисунок 17.2).

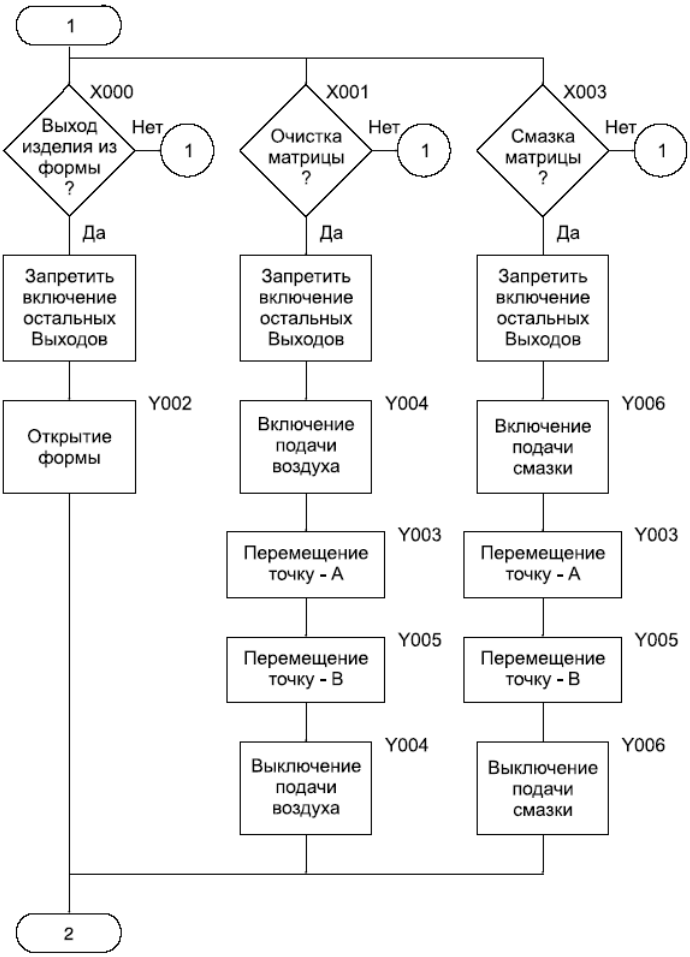
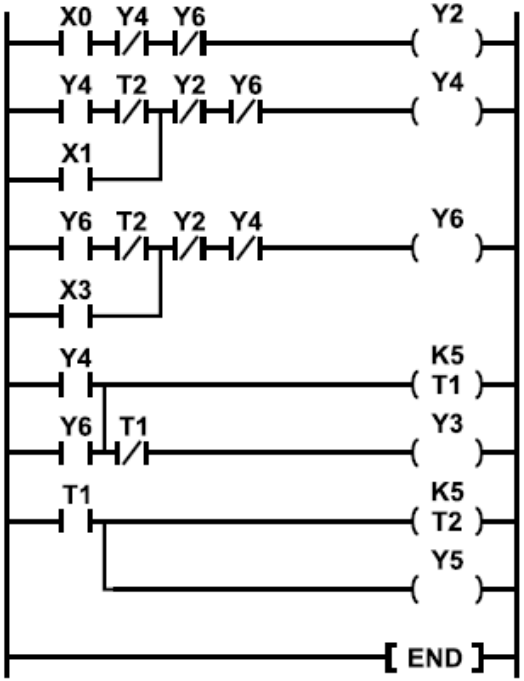


Рисунок 17.2 – Алгоритм управляющей программы

В соответствии с приведенным алгоритмом программа на языке релейно-контактных схем будет иметь вид, как показано на рисунке 17.3.



Комментарии:
Открытие формы

Включение подачи воздуха

Включение подачи смазки

При достижении таймера 5 с, перемещение в точку А

При достижении таймера 5 с, перемещение в точку В

Рисунок 17.3 – Схема управления штамповочной машиной

Штамповочная машина - Настройка времени работы штампа

Задача – Настроить время работы штампа.

• по нажатию на кнопку *Обучение* (вход X0*)* запомнить время работы штампа;

• по нажатию на кнопку *Старт* (вход X1) выполнять операцию используя время, полученное на этапе обучения. Штамповочный пресс показан на рисунке 17.4. Схема управления штамповочным прессом показана на рисунке 17.5.

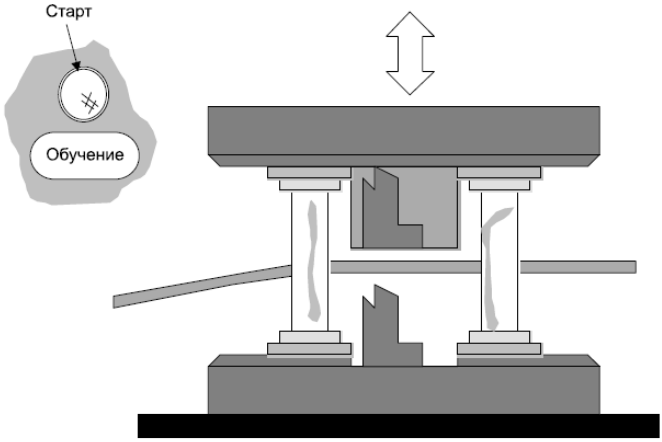


Рисунок 17.4 – Штамповочный пресс

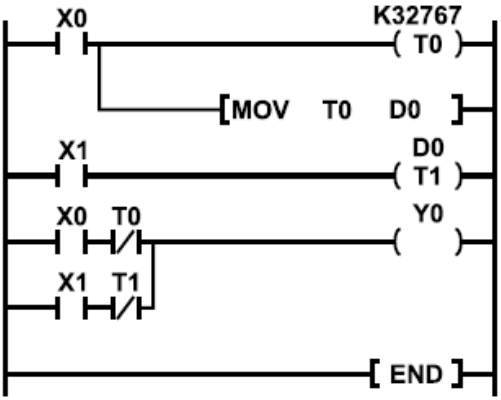


Рисунок 17.5 – Схема управления штамповочным прессом

17.2 Конвейер – Разделение потоков

Задача – Считать число бутылок и переключать направление промышленного конвейера (рисунок 17.6) .

• считать число бутылок;

• по достижению заданного значения D0 переключить направление с «конвейер1» на «конвейер2»;

• по достижению заданного значения D2 переключить направление с «конвейера2» на «конвейер1»;

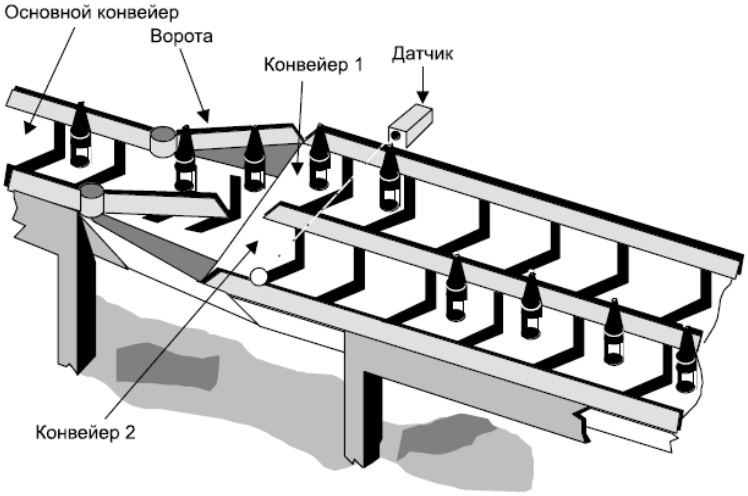
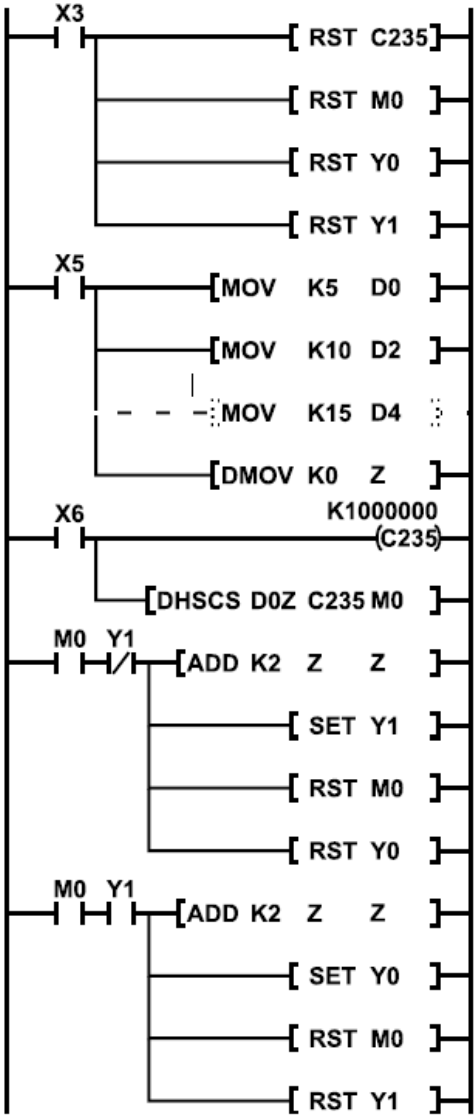


Рисунок 17.6 – Схема конвейерной линии

В соответствии с требованиями задания ниже приведен участок программы на языке релейно-контактных схем (рисунок 17.7).



Комментарии:

Сброс состояний счетчика, внутреннего реле и выходов

Запись значений счета в регистры

Инициализация счета высокоскоростным счетчиком

Выполнение переключения конвейеров и суммирования результатов счета по условиям состояния M0 и Y1.

Рисунок 17.7 – Схема управления конвейерной линией

Литература

1. «Руководство по программированию. Программируемый контроллер семейства MELSEC-FX». MITSUBISHI ELECTRIC EUROPE B.V. 03/2003 (издание 3-е, версия C).
2. «Программирование ПЛК». MITSUBISHI ELECTRIC EUROPE B.V.
3. «Программируемые логические контроллеры серии MELSEC FX1S/ FX1N/ FX2N/ FX2NC. Технический каталог». MITSUBISHI ELECTRIC EUROPE B.V. 10/2003 (Издание 3е, версия C)
4. «Как выбирать контроллерные средства» − Эммануил Ицкович (профессор, заведующий лабораторией методов автоматизации производства Института проблем управления им. В. А. Трапезникова РАН, д-р техн. наук). Статья опубликована в журнале «Оборудование» (прилож. к еженедельнику «Эксперт»)
5. «Как выбрать программируемый логический контроллер» − канд. техн. наук Г.П.Митин (МГТУ Станкин). Статья опубликована в журнале «Мир компьютерной автоматизации».
6. «Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы» − В.Е. Зюбин (Институт автоматики и электрометрии СО РАН). Статья опубликована в журнале “Промышленные АСУ и контроллеры”. – 2005.– №11. – С.31-35
7. «На стандартном пути /ISaGRAF-PRO – мощный инструмент разработчика» − Любашин А.Н., ЗАО «РТСофт», PC Week, 15/2000 (URL: www.rtsoft.ru)
8. «Знакомство со стандартом на языки программирования PLC: IEC 1131-3 (МЭК 1131-3)» − Джеймс Х. Христенсен, главный инженер компании Allen-Bradly. Статья опубликована в журнале «PLCopening», 08/1994.
9. «SCADA-системы, или муки выбора» − Надежда Куцевич, ЗАО РТСофт (URL: http://www.asutp.ru/go/?id=600055&url=www.rtsoft.ru)
10. Программируемые контроллеры. Общие технические требования и методы испытаний : ГОСТ Р 51841-2001 (МЭК 61131-2-92).
11. Степени защиты, обеспечиваемые оболочками» : ГОСТ 14254-96.
12. Сетевые интерфейсы для промышленного применения. Каталог продукции HMS INDUSTRIAL NETWORKS. − М. : ООО «АКОМ», 2004.

ПРИЛОЖЕНИЕ А. Обзор стандартов сетевого взаимодействия ПЛК

AS-InterFace

AS интерфейс (AS-I, Actuator Sensor Interface) оптимизирован для передачи дискретных сигналов и требует минимальных действий по установке и конфигурации. По этой причине, применение AS-I во многих случаях становится более целесообразным, чем применение более «мощных» полевых шин.

Еще одним преимуществом является то, что AS интерфейс обеспечивает питание устройств 24В по сигнальной паре проводов.

Пример участка сети AS-I приведен на рисунке А.1, а ее основные технические характеристики в таблице А.1

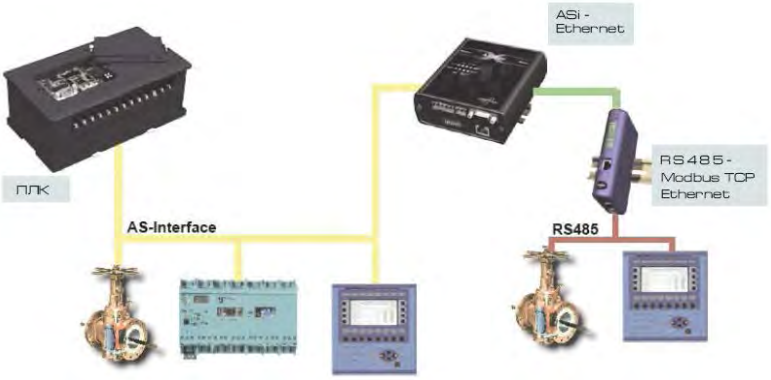


Рисунок А.1 – Пример участка сети AS-I

Таблица А.1 – Технические характеристики сети AS-I

Размер сети	До 62 сетевых узлов
Скорость передачи данных	167 кбит/с
Протяженность линии связи	До 100 м без повторителей; До 300 м с повторителями;
Обмен	Master/Slave
Топология сети	Шина, древовидная, звезда
Функции	Возможность передачи значений аналоговых сигналов; Автоматическая установка адреса в сети; Питание и передача сигналов по одной паре проводов; Нет необходимости в согласовании линии связи (на концах подбирать резисторы-терминаторы); По сравнению с другими сетями применение AS-I позволяет уменьшить затраты на кабельную проводку на 15…40%.

CANopen

CANopen – это промышленный стандарт сети, основанный на протоколе CAN (Control Area Network), который позволяет одновременно обеспечить доступ к параметрам сетевых устройств и передачу данных, критичных ко времени доставки. Функции сетевого управления (network management) значительно упрощают разработку и реализацию проекта, обеспечивая диагностику сети и коррекцию ошибок передачи. Через кабель CANopen возможно производить питание сетевых устройств. Основные технические характеристики сети CANopen приведены в таблице А.2.

Таблица А.2 – Технические характеристики сети CANopen

Размер сети	До 127 сетевых узлов
Скорость передачи данных	10 кбит/с …1 Мбит/с
Протяженность линии связи	25…5000 м (в зависимости от скорости)
Обмен	Master/Slave, peer-to-peer (прямой обмен), Multi-cast и Multi-Master
Топология сети	Шина, сегментированная
Функции	Удаление элементов сети без разрыва передающей среды; Передача данных по условиям «запрос/ответ»; Передача данных, критичных ко времени доставки; Фрагментация данных при пересылке больших объемов информации

CC-Link

Стандарт промышленной сети CC-Link (Control & Communication Link) был разработан компанией Mitsubishi и применяется в Японии. CC-Link – это мощная, скоростная полевая шина, которая обеспечивает как передачу больших объемов данных, так и распределенное управление процессами. Возможность запитывания сетевых устройств по сигнальному кабелю значительно упрощает кабельную разводку. Основные технические характеристики сети CC-Link приведены в таблице А.3.

Таблица А.3 – Технические характеристики сети CC-Link

Размер сети	До 354 сетевых узлов
Скорость передачи данных и протяженность линии связи	156 кбит/с – 1200 м; До 625 кбит/с – 900 м; До 2500 кбит/с – 400 м; До 5 Мбит/с – 160 м; До 10 Мбит/с – 100 м;
Обмен	Широковещательный режим по запросу
Топология сети	Шина (линейная – trunkline/dropline)
Функции	Питание устройств по сигнальной линии;

ControlNet

ControlNet – это первая полевая шина для построения распределенных систем управления реального времени. Сеть ControlNet обеспечивает высокую скорость передачи критичных ко времени данных, передачу больших массивов данных, включая удаленную выгрузку/загрузку управляющих программ, и прямой обмен данными (peer-to-peer). Детерминированность и повторяемость времени передачи сообщений при высокой скорости обмена (5 Мбит/с) позволяют использовать эту сеть в ответственных приложениях. Пример участка сети ControlNet приведены на рисунке А.2, а технические характеристики – в таблице А.4.

Рисунок А.2 – Пример участка сети ControlNet

Таблица А.4 – Технические характеристики сети ControlNet

Размер сети	До 48 сетевых узлов (до 99 с повторителями)
Скорость передачи данных и протяженность линии связи	5 Мбит/с; Коаксиальный кабель: 2 сетевых узла – до 1000 м; 48 сетевых узлов – до 250 м; 99 сетевых узлов с повторителями – до 5000 м; Оптоволоконный кабель: До 3000 м без повторителей; До 30 км – с повторителями
Обмен	Точка-точка (peer-to-peer), Multi-Master, Master/Slave
Топология сети	Шина, древовидная, звезда
Функции	«Горячая» замена/добавление устройств в рабочей сети; Детерминированность, повторяемость; Автоопределение дублирования сетевых адресов; Возможность работы во взрывоопасных зонах; Выделение различных рабочих частот для одновременной передачи данных, управления в реальном времени и удаленного программирования.

DeviceNet

DeviceNet – распространенная во всем мире сеть приборного уровня для задач промышленной автоматизации. В основном применяется для обеспечения связи между устройствами низовой автоматики (дискретные датчики, управляется клапана, софт-стартеры, частотные преобразователи, терминалы управления) и ПЛК (компьютером). Основан на протоколе CAN. Разработан компанией Rockwell Automation, в силу чего очень распространен в Северной Америке. Пример участка сети DeviceNet показан на рисунке А.3, а технические характеристики – в таблице А.5.

Рисунок А.3 – Пример участка сети DeviceNet

Таблица А.5 – Технические характеристики сети DeviceNet

Размер сети	До 64 сетевых узлов
Скорость передачи данных и протяженность линии связи	До 125 кбит/с – 500 м; До 250 кбит/с – 250 м; До 500 кбит/с – 100 м;
Обмен	Multi-Master, Master/Slave, Широковещательный режим, по запросу, по изменению состояния
Топология сети	Шина
Функции	Возможность подачи питания к сетевым устройствам;

Ethernet

Ethernet и TCP/IP протокол все чаще и чаще применяются в области промышленной автоматизации, чему особенно способствует появление новых технологий Ethernet, таких как Fast Ethernet со скоростью 100 Мбит/с, полнодуплексная связь, интеллектуальные коммутаторы. Для промышленного использования определено несколько различных прикладных протоколов: Ethernet/IP (ODVA), ProfiNet (PNO), IDA (IDA group), Foundation Fieldbus HSE (FF), Interbus on Ethernet (Interbus Club) b Modbus TCP/IP (ModConnect). Пример

участка Ethernet приведен на рисунке А.4, краткие технические характеристики в таблице А.6.

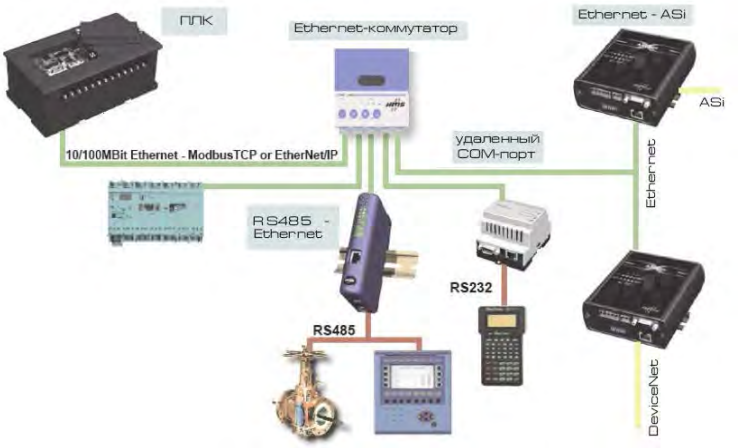


Рисунок А.4 – Пример участка сети Ethernet

Таблица А.6 – Технические характеристики сети Ethernet

Размер сети	Масштабируемая, размер практически не ограничен
Скорость передачи данных	10, 100, 1000 Мбит/с
Протяженность линии связи	Витая пара: 100 м; Оптоволокно: 35…2000 м, в зависимости от кабеля и скорости передачи данных
Топология сети	Звезда
Структура данных	Первые четыре уровня протокола являются стандартными и обеспечивают: передачу данных, доступ к шине, взаимодействие разных сетей (IP) и надежность доставки данных (TCP). Над этими уровнями могут располагаться пользовательские протоколы: ProfiNet, Modbus TCP, др. для автоматизации и HTTP, FTP, SMTP, Telnet для общецелевого пользования.

FIPIO

Полевая шина FIPIO разработана для обеспечения связи между различными компонентами распределенной сети управления. FIPIO соответствует стандартам WorldFip и основана на принципах взаимодействия «издатель/подписчик» (Producer/Consumer). Протокол FIPIO поддерживает как детерминированную передачу данных, так и произвольный обмен данными. Это означает, что по сети можно передавать произвольные сообщения (периодические отчеты, аварийные сигналы, данные конфигурации и т.п.) без нарушения передачи потока данных, критичных ко времени. На основе шины FIPIO можно строить распределенные контуры управления реального времени – и при этом без проблем передавать диагностические и другие данные. Технические характеристики сети FIPIO приведены в таблице А.7.

Таблица А.7 – Технические характеристики сети FIPIO

Размер сети	До 256 сетевых узлов
Протяженность линии связи	До 40 км
Скорость передачи данных	31.25 кбит/с, 1 Мбит/с, 2.5 Мбит/с, 6 Мбит/с (оптоволокно)
Обмен	Точка-точка (peer-to-peer), Remote I/O
Функции	Обеспечение возможности передачи сообщения (периодические отчеты, аварийные сигналы, данные конфигурации без нарушения потока данных, критичных ко времени; Возможность резервирования линии передачи с быстрым переключением на аппаратном уровне, что соответствует требованиям безопасности на ответственных производствах.

Interbus

Interbus изначально разрабатывалась как скоростная промышленная шина для связи с датчиками и исполнительными устройствами. Благодаря кольцевой топологии и оригинальной процедуре передачи, Interbus имеет отличные показатели по скорости и детерминированности передачи данных. Имеются функции встроенной диагностики, обеспечивается легкая установка и настройка plug-&-play, возможность применения оптоволокна. Interbus – кольцевая система,

т.е. все сетевые устройства объединены в замкнутое транспортное кольцо. Каждое устройство усиливает принятый сигнал и передает дальше, что обеспечивает качественную связь на больших расстояниях. Для передачи и приема данных для всех устройств используется один кабель.

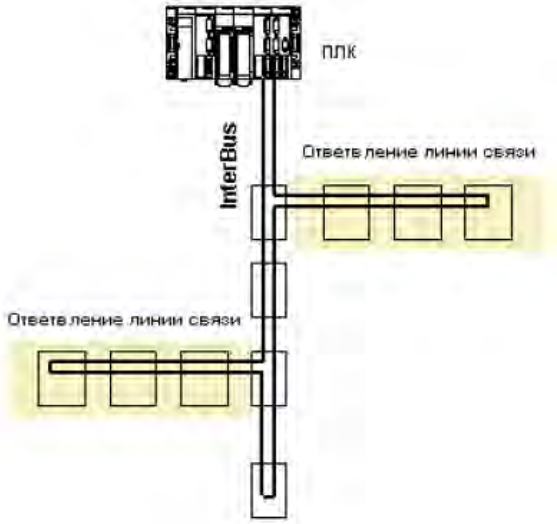


Рисунок А.5 – Пример участка сети Interbus

Основная линия связи отходит от главного устройства сети и далее может быть использована для формирования локальных ответвлений. Всего таким образом можно организовать до 16 уровней вложенных ответвлений, что позволяет очень гибко и быстро менять структуру сети под изменяющиеся требования. Кольцо автоматически замыкается на последнем в цепи устройстве, при этом согласующие резисторы-терминаторы на конце линии не требуются. Ответвления сети осуществляются через специальные соединители (coupling), которые позволяют подключать/отключать эти ответвления без нарушения работы всей сети. Пример участка сети Interbus показан на рисунке А.5, в таблице А.8 даны ее технические характеристики.

Таблица А.8 – Технические характеристики сети Interbus

Размер сети	До 256 сетевых узлов
Протяженность линии связи	До 13 км при использовании RS485; До 50 км при использовании оптоволокна; Минимальное расстояние между устройствами - 400 м;
Скорость передачи данных	500 кбит/с…2 Мбит/с
Топология	Активное кольцо
Обмен	Master/Slave, циклические и PCP данные;
Функции	Поддержка PMS сервисов: PCP v.2.0; Определение до 256 собственных PCP и VFD-объектов.

LonWorks

Технология LonWorks основана на открытом коммуникационном протоколе и позволяет создавать распределенные вычислительные системы. При этом в одну сеть можно объединить самые различные устройства: от дискретных датчиков до контроллеров и рабочих станций (PC). По сети LonWorks можно не только передавать данные, но и производить удаленную конфигурацию устройств.

Сети LonWorks применяются в самых разнообразных приложениях: «интеллектуальный дом», коммерческие сети, автоматизация производства, автоматизация отдельных устройств, транспортные системы. Технические характеристики сети LonWorks приведены в таблице А.9.

Таблица А.9 – Технические характеристики сети LonWorks

Размер сети	До 32000 сетевых узлов (доменов)
Протяженность линии связи	До 2000 м при скорости 78 кбит/с
Скорость передачи данных	1.25 Мбит/с (полный дуплекс)
Топология	Шина, звезда, кольцо
Обмен	Master/Slave, peer-to-peer («точка-точка»);
Функции	Использование объектов и профилей LonMark

Modbus Plus, RTU & TCP

Протокол MODBUS был разработан компанией Modicon в 1979 году для обеспечения связи между «интеллектуальными» устройствами по принципу Master-Slave/Client-Server. Это полностью открытый протокол, получивший очень широкое распространение и ставший де-факто стандартом в промышленной автоматизации. В настоящее время широко применяются три разновидности: Modbus Plus, Modbus RTU и Modbus TCP. Пример участка сети Modbus показан на рисунке А.6, а в таблице А.10 приведены ее технические характеристики.

Рисунок А.6 – Пример участка сети Modbus

Таблица А.10 – Технические характеристики сети Modbus

Размер сети	До 32 сетевых узлов (до 64 с повторителями)
Скорость передачи данных	До 2 Мбит/с
Протяженность линии связи	До 500 м без повторителей; До 2000 м с повторителями.
Обмен	Peer-to-peer («точка-точка»), Multi-Master;
Функции	Общая база транзакций; Сканирование сети; Чтение и запись отдельных регистров; Маршрутизация до 6 сетей.

Profibus

Profibus – одна из наиболее распространенных систем связи в области промышленной автоматизации. Наиболее широкое распространение Profibus получил в Европе, где очень активно продвигается компанией Siemens. Данный протокол был разработан более15 лет назад и первоначально был национальным немецким стандартом DIN19245. За прошедшее время в этот стандарт было внесено много исправлений и дополнений, в результате чего в настоящее время Profibus используется как на нижнем уровне автоматизации процессов , так и на верхнем уровне управления производством (Profibus-DP, Profibus-PV1/DPV2, Profibus-PA). Последняя редакция технологии Profibus закреплена в международном стандарте МЭК-61158 (IEC-61158) и допускает использование различных физических сред передачи данных. Пример участка сети Profibus приведен на рисунке А.7, в таблице А.11 даны ее технические характеристики.

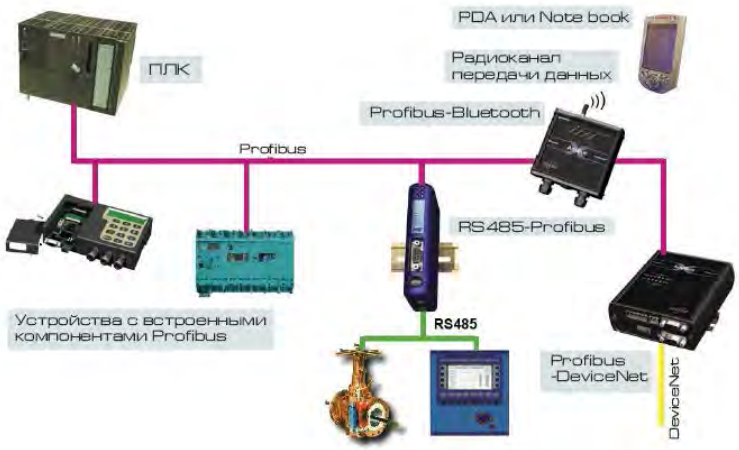


Рисунок А.7 – Пример участка сети Profibus

Таблица А.11 – Технические характеристики сети Profibus

Размер сети	До 126 сетевых узлов
Скорость передачи данных	9.6 кбит/с…12 Мбит/с
Протяженность линии связи	9.6…33.75кбит/с – 1200 м; До 185 кбит/с – 1000 м; До 500 кбит/с – 400 м; До 1.5 Мбит/с – 200 м; До 12 Мбит/с – 100 м;
Топология сети	Шина, сегментированная
Обмен	DP: Master/Slave, Cyclic (циклическая передача маркера), Polling (передача маркера по запросу). DPV1: Cyclic, Polling + ациклическая передача данных;

Profinet

Profinet - это новая коммуникационная технология, разработанная совместными усилиями нескольких компаний и организаций: Siemens, Phoenix Contact, the Profibus User Organisation и Interbus Club.

Profinet основан на технологиях Industrial Ethernet и Profibus DP и является стандартом, который может быть применен во всех областях автоматизации, от полевого уровня до уровня управления производством, распределенном управлении в реальном масштабе времени. Profinet может быть применен в промышленных системах обеспечения безопасности, имеет повышенную защищенность от несанкционированного доступа.

Стандарт Profinet имеет три класса:

- Profinet CBA: обмен данными между управляющими контроллерами;
- Profinet RT: программно реализованная система реального времени для связи контроллера с устройствами ввода/вывода;
- Profinet IRT: аппаратно реализованная система жесткого реального времени, в основном предназначенная для управления перемещением.

Технические характеристики совпадают с сетью Ethernet, с добавлением дополнительных возможностей реального времени.

ПРИЛОЖЕНИЕ Б. Краткий обзор SCADA-систем

Обычно системный интегратор или конечный пользователь, приступая к разработке прикладного программного обеспечения (ППО) для создания системы управления, выбирает один из следующих путей:

- программирование с использованием "традиционных" средств (традиционные языки программирования, стандартные средства отладки и пр.);
- использование существующих, готовых (COTS Commercial Off The Shelf) инструментальных проблемно-ориентированных средств.

Безусловно, нет ничего лучше качественного, хорошо отлаженного ППО, написанного высококвалифицированным программистом специально для некоторого проекта. Но следующую задачу этот программист вынужден решать опять практически с нуля. Процесс создания ППО для сложных распределенных систем становится недопустимо длительным, а затраты на его разработку очень высокими.

Таким образом, сама логика развития современного бизнеса в части разработки ППО для конечных систем управления требует использования всё более развитых инструментальных средств типа SCADA-систем (от Supervisory Control And Data Acquisition). Разработка современной SCADA-системы требует больших вложений и выполняется в длительные сроки. И именно поэтому в большинстве случаев разработчикам управляющего ППО, в частности ППО для АСУ ТП, представляется целесообразным идти по второму пути, приобретая, осваивая и адаптируя какой-либо готовый, уже испытанный универсальный инструментарий.

В связи с этим, возникает вопрос выбора SCADA-системы. Ниже перечислены **(таблица Б.1)** только некоторые из популярных на западном и российском рынках SCADA-систем, имеющих некоторую поддержку в нашей стране:

Таблица Б.1 – Популярные SCADA-системы имеющие поддержку в России

SCADA-система	**Фирма-изготовитель**	**Страна**
Factory Link	Unated States DATA-Co.	США
In Touch	Wonderware	США
Genesis	Iconics	США
RealFlex	BJ.Software.Systems	США
Sitex	Jade Software	Великобритания
FIX	Intellution	США
TraceMode	AdAstra	Россия
IGSS	Seven Technologies	Дания
Image	Технолинк	Россия
RSView	Rockwell Software.Inc	США

На примере указанных пакетов предлагается рассмотреть некоторые основные возможности и характерные особенности SCADA-систем. Публикаций по SCADA-системам в нашей прессе достаточно много. Просматривая их, хотелось бы достаточно схематично остановиться на уже ставшем традиционном наборе свойств и характеристик SCADA-систем и заострить внимание на новых, по-

явившихся недавно, связях SCADA-систем с окружающим миром. (OPC-серверы, расширения реального времени для Windows NT) и на моментах, которые нечасто находят отражение в публикациях о нише SCADA-систем в комплексе программных компонентов сквозной автоматизации производства. SCADA-системы закрывают цеховой уровень автоматизации, связанный, прежде всего, с получением и визуализацией информации от программируемых контроллеров, распределенных систем управления. Поставляемая на данный уровень информация недоступна на уровне управления производством. Поэтому важно отметить, что некоторые фирмы разрабатывают системы управления производством и обеспечивают обмен между этими уровнями.

Характеристики SCADA-систем

Функциональные возможности

В силу тех требований, которые предъявляются к системам SCADA, спектр их функциональных возможностей определен и реализован практически во всех пакетах. Перечислим основные возможности и средства, присущие всем системам и различающиеся только техническими особенностями реализации:

- автоматизированная разработка, дающая возможность создания программного обеспечения (ПО) системы автоматизации без реального программирования;
- средства сбора первичной информации от устройств нижнего уровня;
- средства управления и регистрации сигналов об аварийных ситуациях;
- средства хранения информации с возможностью ее постобработки (как правило, реализуется через интерфейсы к наиболее популярным базам данных);
- средства обработки первичной информации;
- средства визуализации представления информации в виде графиков, гистограмм и т.п.;
- возможность работы прикладной системы с наборами параметров, рассматриваемых как единое целое (recipie , или установки).

Основу большинства SCADA-пакетов составляют несколько программных компонентов (база данных реального времени, ввода-

вывода, предыстории, аварийных ситуаций) и администраторов (доступа, управления, сообщений).

Следует отметить, что технология проектирования систем автоматизации на основе различных SCADA-систем во многом схожа и включает следующие этапы.

- Разработка архитектуры системы автоматизации в целом. На этом этапе определяется функциональное назначение каждого узла системы автоматизации.
- Решение вопросов, связанных с возможной поддержкой распределенной архитектуры, необходимостью введения узлов с горячим резервированием и т.п.
- Создание прикладной системы управления для каждого узла. На этом этапе специалист в области автоматизируемых процессов наполняет узлы архитектуры алгоритмами, совокупность которых позволяет решать задачи автоматизации.
- Приведение параметров прикладной системы в соответствие с информацией, которой обмениваются устройства нижнего уровня (например, программируемые логические контроллеры ПЛК) с внешним миром (датчики температуры, давления и др.). Отладка созданной прикладной программы в режиме эмуляции (в некоторых системах, например IGSS, режим отладки практически отсутствует) и в реальном режиме.

Перечисленные выше возможности систем SCADA в значительной мере определяют стоимость и сроки создания ПО, а также сроки ее окупаемости.

Технические характеристики

Перечислим характеристики, важные для оценки функциональности SCADA-систем, с кратким их анализом.

- **Программно-аппаратные платформы, на которых реализована SCADA-система**

Анализ перечня таких платформ необходим **(таблица Б.2),** поскольку от него зависит ответ на вопросы распространения SCADA-системы на имеющиеся вычислительные средства, а также оценка стоимости эксплуатации системы (прикладная программа, разработанная в одной операционной среде, может выполняться в любой другой, которую поддерживает выбранный SCADA-пакет). В различных SCADA-системах этот вопрос решен по разному. Так,

FactoryLink имеет весьма широкий список поддерживаемых программно-аппаратных платформ.

Таблица Б.2. Поддержка программно-аппаратных платформ

Операционная система	Компьютерная платформа
DOS / MS Windows	IBM-PC
OS/2	PS/2
SCO-UNIX	IBM-PC
VMS	VAX
AIX	RS6000
HP-UX	HP-9000
MS Windows NT	Системы с реализованным Windows NT

В то же время в таких SCADA-системах, как RealFlex и Sitex основу программной платформы принципиально составляет единственная, хотя и удовлетворяющая многим требованиям, операционная система реального времени QNX.

Подавляющее большинство SCADA-систем реализовано на платформах MS Windows. Именно такие системы предлагают наиболее полные и легко наращиваемые человеко-машинные интерфейсные (Man Machine Interface MMI) средства. Учитывая продолжающееся усиление позиций Microsoft на рынке операционных систем (ОС), следует отметить, что даже разработчики многоплатформных SCADA-систем, такие как United States DATA Co, приоритетным считают дальнейшее развитие своих SCADA-систем на платформе Windows NT. Некоторые фирмы, до сих пор поддерживавшие SCADA-системы на базе ОС реального времени (РВ), начали менять ориентацию, выбирая системы на платформе Windows NT. Все более очевидным становится применение ОС реального времени, в основном, во встраиваемых системах. Таким образом, основным полем, где сегодня разворачиваются главные события глобального рынка SCADA-систем, стала ОС MS Windows NT на фоне всё ускоряющегося сворачивания активности в области MS DOS, MS Windows 3.xx/95.

▪ **Имеющиеся средства сетевой поддержки.**

Одна из основных особенностей современного мира систем автоматизации высокая степень интеграции этих систем. В любой из них могут быть задействованы объекты управления, исполнительные

механизмы, аппаратура, регистрирующая и обрабатывающая информацию, рабочие места операторов, серверы баз данных и т.д. Очевидно, что для эффективного функционирования в этой разнородной среде SCADA-система должна обеспечивать высокий уровень сетевого сервиса. Желательно, чтобы она поддерживала работу в стандартных сетевых средах (ARCNET, ETHERNET и т.д.) с использованием стандартных протоколов (NETBIOS, TCP/IP и др.), а также обеспечивала поддержку наиболее популярных сетевых стандартов из класса промышленных интерфейсов (PROFIBUS, CANBUS, LON, MODBUS и т.д.) Обобщенная схема подобной системы приведена на **рисунке Б.1.**

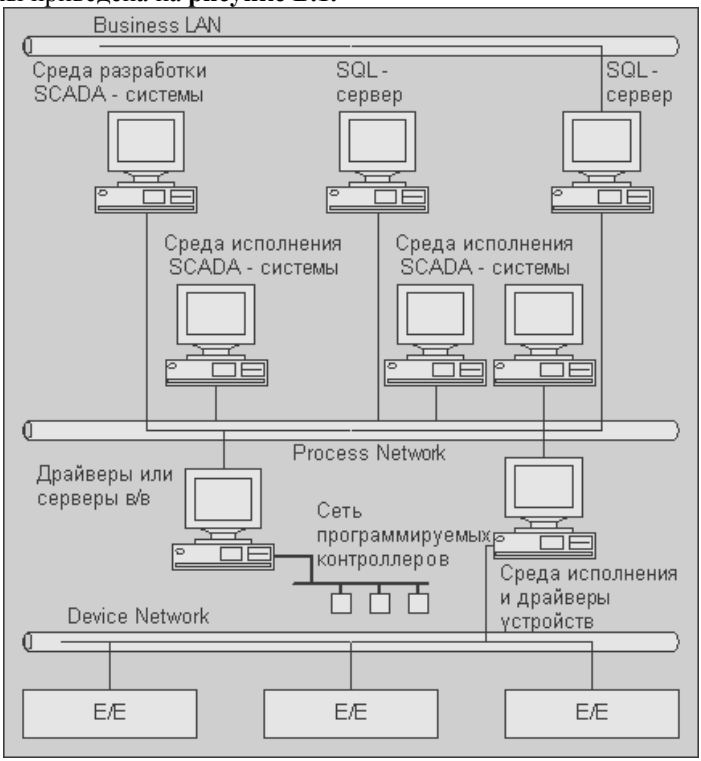


Рисунок Б.1 – Схема интеграции SCADA-приложений в комплексные системы управленияЭтим требованиям в той или иной степени удовлетворяют практически все рассматриваемые SCADA-системы, с тем только различием, что набор поддерживаемых сетевых интерфейсов, конечно же, разный.

▪ **Встроенные командные языки.**

Большинство SCADA-систем имеют встроенные языки высокого уровня, VBasic-подобные языки, позволяющие сгенерировать адекватную реакцию на события, связанные с изменением значения переменной, с выполнением некоторого логического условия, с нажатием комбинации клавиш, а также с выполнением некоторого фрагмента с заданной частотой относительно всего приложения или отдельного окна.

▪ **Поддерживаемые базы данных.**

Практически все SCADA-системы, в частности, Genesis, InTouch используют синтаксис ANSI SQL, который не зависит от типа базы данных. Таким образом, приложения виртуально изолированы, что позволяет менять базу данных без серьезного изменения самой прикладной задачи, создавать независимые программы для анализа информации, использовать уже наработанное программное обеспечение, ориентированное на обработку данных.

▪ **Графические возможности.**

Для специалиста-разработчика системы автоматизации, также как и для специалиста- технолога , чье рабочее место создается, очень важен графический пользовательский интерфейс (Graphic Users Interface HMI). Функционально графические интерфейсы SCADA-систем очень похожи. В каждой из них существует графический объектно-ориентированный редактор с определенным набором анимационных функций. Используемая векторная графика дает возможность осуществлять широкий набор операций над выбранным объектом, а также быстро обновлять изображение на экране, используя средства анимации.

Крайне важен также вопрос о поддержке в рассматриваемых системах стандартных функций GUI. Поскольку большинство рассматриваемых SCADA-систем работают под управлением Windows, это и определяет тип используемого GUI.

▪ **Открытость систем**

Программная система является открытой, если для нее определены и описаны используемые форматы данных и процедурный интерфейс, что позволяет подключить к ней внешние, независимо разработанные компоненты.

Разработка собственных программных модулей. Перед фирмами-разработчиками систем автоматизации часто встает вопрос о создании собственных (не предусмотренных в рамках систем SCADA) программных модулей и включение их в создаваемую систему автоматизации. Поэтому вопрос об открытости системы является важной характеристикой SCADA-систем. Фактически открытость системы означает доступность спецификаций системных (в смысле SCADA) вызовов, реализующих тот или иной системный сервис. Это может быть и доступ к графическим функциям, функциям работы с базами данных и т.д.

Драйверы ввода-вывода. Современные SCADA-системы не ограничивают выбора аппаратуры нижнего уровня, так как предоставляют большой набор драйверов или серверов ввода-вывода и имеют хорошо развитые средства создания собственных программных модулей или драйверов новых устройств нижнего уровня. Сами драйверы разрабатываются с использованием стандартных языков программирования. Вопрос, однако, в том, достаточно ли только спецификаций доступа к ядру системы, поставляемых фирмой-разработчиком в штатном комплекте (система Trace Mode), или для создания драйверов необходимы специальные пакеты (системы FactoryLink, InTouch), или же, вообще, разработку драйвера нужно заказывать у фирмы-разработчика.

Для подсоединения драйверов ввода-вывода к SCADA используются два механизма: стандартный динамический обмен данными (Dynamic Data Exchange DDE) и обмен по внутреннему (известному только фирме разработчику) протоколу. В SCADA-системах основным механизмом, используемым для связи с внешним миром, до сих пор остается механизм DDE. Но из-за своих ограничений по производительности и надежности он не совсем пригоден для обмена информацией в реальном масштабе времени. Взамен DDE компания Microsoft предложила более эффективное и надежное средство передачи данных между процессами OLE (Object Linking and Embedding включение и встраивание объектов). Механизм OLE поддерживается в RSView, Fix, InTouch, Factory Link и др. На базе OLE появляется новый стандарт OPC (OLE for Process Control OLE

для АСУТП), ориентированный на рынок промышленной автоматизации. Новый стандарт, во-первых, позволяет объединять на уровне объектов различные системы управления и контроля, функционирующие в распределенной гетерогенной среде, во-вторых, устраняет необходимость использования различного нестандартного оборудования и соответствующих коммуникационных программных драйверов. С точки зрения SCADA-систем, появление OPC-серверов означает разработку программных стандартов обмена с технологическими устройствами. Поскольку производители полностью разбираются в своих устройствах, то эти спецификации являются для них руководством к разработке соответствующих драйверов. Так как эти программные драйверы уже появляются на рынке, разработчики SCADA-систем предлагают свои механизмы связи с OPC-драйверами. OPC-интерфейс допускает различные варианты обмена: получение сырых данных с физических устройств, из распределенной системы управления или из любого приложения (рисунок Б.2). На рынке появились инструментальные пакеты для написания OPC-компонентов, например, OPC-Toolkits фирмы FactorySoft Inc., включающий OPC Server Toolkit, OPC Client Toolkit, примеры OPC-программ.

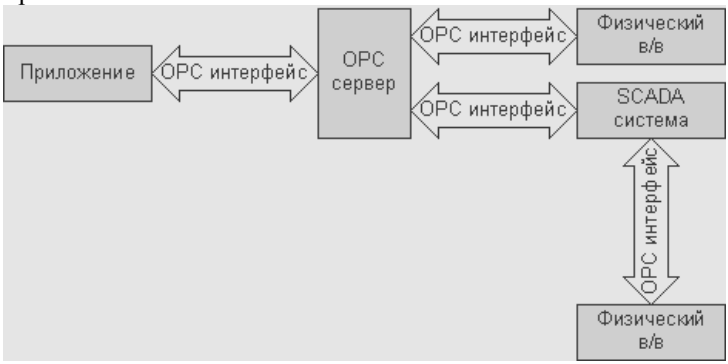


Рисунок Б.2 – Варианты обмена SCADA-систем с приложениями и физическими устройствами через OPC-интерфейс

Встраиваемые объекты ActiveX. Объекты ActiveX это объекты, в основе которых лежит модель составных объектов Microsoft COM (Component Object Model). Технология COM определяет общую схему взаимодействия компонентов программного обеспечения в среде Windows и предоставляет стандартную инфраструктуру, поз-

воляющую объектам обмениваться данными и функциями между прикладными программами. Большинство SCADA-систем являются контейнерами, которые уведомляются ActiveX о происшедших событиях. Любые ActiveX-объекты могут загружаться в систему разработки большинства SCADA-систем и использоваться при создании прикладных программ. Управление ActiveX-объектами осуществляется с помощью данных, методов и событийных функций, свойственных выбранному объекту.

Разработки третьих фирм. Многие компании занимаются разработкой драйверов, ActiveX-объектов и другого программного обеспечения для SCADA-систем. Этот факт очень важно оценивать при выборе SCADA-пакета, поскольку это расширяет область применения системы непрофессиональными программистами (нет необходимости разрабатывать программы с использованием языков C или Basic).

Для реализации вышеуказанных технологий разработаны специальные библиотеки и инструментальные системы для ОС Windows. Использование же только спецификаций стандартов для этого не только достаточно трудоемко, но и требует высокого профессионализма программистов и, следовательно, затруднительно для не-Windows платформ.

▪ **Жесткое реальное время для Windows NT**Один из существенных недостатков SCADA-систем на платформах Windows 3.xx/95 по сравнению со SCADA-системами на платформах ОСРВ отсутствие поддержки жесткого реального времени. Ситуация стала изменяться с появлением Windows NT. Выход в свет этой ОС стимулировал разработку новых подходов в поддержке жесткого реального времени. Прежде всего, сама по себе Windows NT делает весьма успешные попытки потеснить ОСРВ. Тем не менее, Windows NT имеет ряд ограничений. Такие ее особенности, как предпочтение аппаратного прерывания программному (даже если это простое движение мыши), выполнение в подпрограмме обработки аппаратных прерываний лишь необходимых действий с выполнением последующей обработки через очередь отложенных процедур, отсутствие приоритетной обработки процессов в очереди отложенных процедур, не позволяют отнести Windows NT к категории классических ОС реального времени.

Ряд фирм (LP Elektronik, Imagination Systems, RadSys, Spectron Microsystems, VenturCom) предприняли более радикальные попытки превратить Windows NT в ОС жесткого реального времени. Рассмот-

рим некоторые ключевые особенности реализации такой идеи на подсистеме реального времени RTX (Real Time Extension), предложенной фирмой Ventur Com. Именно эта реализация получает в настоящее время наиболее широкое распространение. Фирмы-разработчики SCADA-систем незамедлительно начали предлагать применение новых решений. Так, набор прикладных интерфейсов программирования RTX 4.1 (Ventur Com) в FIX позволяет:

- осуществлять полный контроль над задачами реального времени;
- использовать фиксированную систему из 128 приоритетов для контроля RTX-задач;
- применять стандартные средства обмена данными между задачами;
- обращаться к стандартным функциям из Win32 API.

Появление подобных решений, во-первых, наносит очередной удар по SCADA-системам на базе ОСРВ, поскольку отнимает (хотя и достаточно искусственно) у них очень важный козырь преимущества жесткого реального времени, заложенные в ОСРВ, и, во-вторых, теснит применение ОСРВ во встраиваемых системах.

- **Эксплуатационные характеристики**Эксплуатационные характеристики SCADA-системы имеют большое значение, поскольку от них зависит скорость освоения продукта и разработки прикладных систем. Они в конечном итоге отражаются на стоимости реализации проектов.

Удобство использования. Следует отметить, что сервис, предоставляемый SCADA-системами на этапе разработки прикладного ПО, обычно очень высок это вытекает из основных требований к таким системам. Почти все они имеют Windows-подобный пользовательский интерфейс, что во многом повышает удобство их использования, как в процессе разработки, так и в период эксплуатации прикладной задачи.

Наличие и качество поддержки. Необходимо обращать внимание не только на наличие технической поддержки SCADA-систем, как таковой, но и на ее качество. Для зарубежных систем в Беларуси возможны следующие уровни поддержки: услуги фирмы-разработчика; обслуживание региональными представителями фирмы-разработчика; взаимодействие с системными интеграторами. Судя по большому количеству установок зарубежных систем, можно предположить, что поддержка этих систем достаточно эффективна.

Русскоязычные системы, несмотря на сравнительно малые количества установок по сравнению с системами ведущих зарубежных фирм (имеется в виду глобальный рынок), создавались и поддерживаются фирмами-разработчиками, содержащими штаты высокопрофессиональных программистов, которые имеют все предпосылки для качественного технического обслуживания своих продуктов. Так, для освоения Trace Mode фирма AdAstra предоставляет полную документацию на русском языке, организует периодические курсы обучения, реализует горячую линию, готова по заказу внести в систему функциональные изменения или разработать необходимые драйверы.

Русификация. Любая система управления, имеющая интерфейс с оператором, должна допускать возможность общения с человеком на его родном языке. Поэтому крайне важна возможность использования в системе различных шрифтов кириллицы, ввод/вывод системных сообщений на русском языке, перевод документации, различных информационных материалов. Для некоторых систем (Image, Trace Mode) эта проблема вообще отсутствует, так как они разрабатывались русскоязычными фирмами. Для многих зарубежных продуктов проблема русификации в значительной мере снимается, во всяком случае, для подсистем исполнения или подсистем исполнения (RunTime), если они используют наборы шрифтов Windows. Часть зарубежных систем имеют переводы документации на русский язык (InTouch). Нужна ли русифицированная среда разработки? Положительный ответ не очевиден. Но если да, то среда, обязательно протестированная и рекомендованная фирмой-разработчиком. Так как с технической точки зрения проблем с русификацией нет (использование редакторов ресурсов из любой среды разработки Borland C++, Visual C++), то проблема лишь в легитимности этой процедуры.

▪ **Интеграция многоуровневых систем автоматизации**

Схематично уровни управления предприятием показаны на рисунке Б3. SCADA-системы ответственны за получение информации с уровня Управления, снизу, т. е. от различных датчиков через устройства сопряжения, от программируемых контроллеров, поставляющих информацию для непосредственного управления производственным процессом. Далее информация с уровня Управления поступает на вход SCADA-систем. На SCADA-уровне возможно оперативное управление процессом, принятие тактических решений на

основе информации, полученной на уровне Управления. Сам процесс поступления информации на производстве происходит и сверху, и снизу. Сверху формируется информация, отвечающая за работу предприятия в целом, осуществляется планирование производства.

Рисунок Б 3 – Уровни управления предприятием

На рисунке Б.4 дана информационная модель предприятия.

Точная, своевременная, достоверная информация на каждом уровне производства позволяет оценить уровень издержек, качество и конкурентоспособность продукции. Для организации связи между информацией сверху и снизу необходим класс инструментальных средств управления производством, ответственный за доставку данных в реальном времени с уровня Управления наверх и в обратном направлении, с возможной обработкой этих данных. Поэтому достаточно важным критерием сравнения инструментальных средств, поддерживающих разработку АСУ ТП, является наличие средств доставки информации со SCADA-уровня наверх, на уровень планирования производства. Ряд фирм (Intellution, Wonderware) предлагает продукты (Fix BOS, InTrack, InBatch), представляющие собой системы управления производством. Основное их назначение заключается в создании прикладных программ, моделирующих и прослеживающих каждую стадию производственных процессов от загрузки сырья до выпуска готовой продукции.

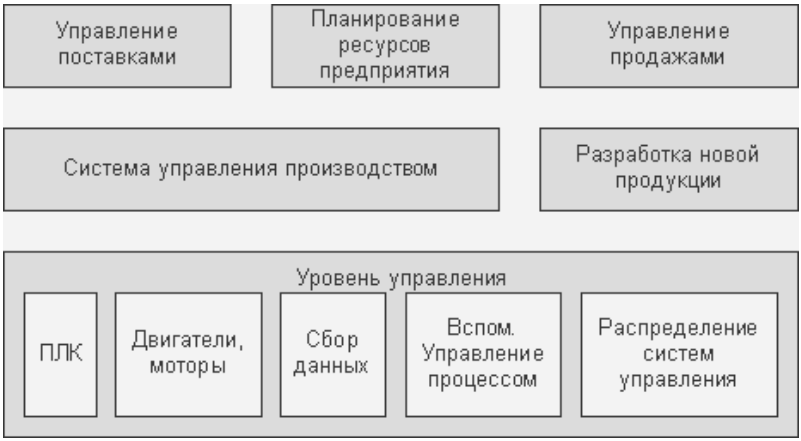


Рисунок Б.4 – Информационная модель предприятия

Огромное стратегическое значение имеет то, насколько инструментальные системы АСУ ТП связаны с Microsoft BackOffice Suite, поскольку последние стали распространенными офисными программными продуктами. Поэтому, например, все продукты FactorySuite компании Wonderware легко интегрируются с такими продуктами, как Microsoft SQL Server, Windows NT Server, System Management Server, SNA Server и Mail-Server. Фирма Wonderware предлагает IndustrialSQL Server, позволяющий регистрировать данные в реальном времени. Источником данных могут быть InTouch-серверы ввода-вывода. Построен же IndustrialSQL Server на базе Microsoft SQL Server. Это существенно расширяет возможности всего производственного персонала в смысле возможности доступа к полной информации о любом этапе производства.

Все более актуальным становится требование передачи как статической (в определенные моменты времени), так и динамической (постоянно) информации на web-узлы. Появившиеся в некоторых web-браузерах объекты ActiveX (в четвертой версии Microsoft Explorer, например) позволяют передавать данные из SCADA-системы на web-страницы. Но существуют и более многофункциональные компоненты типа Scout фирмы Wonderware, обеспечивающие возможность доступа к системам автоматизации на базе InTouch через Internet/Intranet и позволяющие удаленному пользователю вза-

имодействовать с прикладной задачей автоматизации, как с простой WEB-страницей.

Подведем некоторый итог

По функциональным возможностям все рассмотренные системы в целом сравнимы. Технология программирования близка к интуитивному восприятию автоматизируемого процесса. Плюс мощное объектно-ориентированное программирование, используемое в большинстве этих пакетов, делает эти продукты легкими в освоении и доступным для широкого круга пользователей.

Все системы можно считать открытыми, обеспечивающими возможность дополнения функциями собственной разработки, имеющими открытый протокол для разработки собственных драйверов, развитую сетевую поддержку, возможность включения ActiveX-объектов и доступность к стандартным базам данных.

Важной особенностью всех SCADA-систем является количество поддерживаемых разнообразных ПЛК. Системы InTouch, Factory Link, GENESIS, RealFlex поддерживают десятки и сотни драйверов, что делает их безусловными лидерами по этому показателю.

Построение прикладной системы на основе любой из рассмотренных SCADA-систем резко сокращает набор необходимых знаний в области классического программирования, позволяя концентрировать усилия по освоению знаний в самой прикладной области.

У разработчиков SCADA-систем на платформе Windows NT появилась возможность использовать расширение реального времени (RTX), чтобы преодолеть недостатки WindowsNT в задачах реального времени.

Следует отметить тенденции включения SCADA-систем в системы комплексной автоматизации предприятия. Это обеспечивает точную, своевременную информацию на каждом уровне производства.

Применение в SCADA-системах новых технологий, разработка инструментальных средств комплексной автоматизации предприятия свидетельствуют о стремлении и возможности фирм-разработчиков постоянно совершенствовать свои продукты, что является немаловажным фактором при выборе инструментального средства, даже если не все его технологические решения в ближайшее время будут использованы Вами.

Так как общее поле деятельности ведущих компаний-производителей описываемых инструментальных систем сегодня концентрируется в области MS Windows NT/2000/XP, и общие технические возможности систем достаточно близки, то главный упор делается на качество технической поддержки, обучение пользователей, на оказание дополнительных комплексных услуг по освоению и внедрению конечной системы управления. Другими словами, на сокращение издержек системных интеграторов и конечных пользователей на инжиниринг и менеджмент своих проектов, а также на уменьшение стоимости сопровождения конечной системы. Именно эти показатели сегодня, в основном, влияют на рейтинг и рыночный успех той или иной SCADA-системы. Пожалуй, эти показатели даже более важны, чем абсолютные стоимостные характеристики SCADA.

Учебное издание

ЛИВШИЦ Юрий Евгеньевич
ЛАКИН Владимир Иванович
МОНИЧ Юлия Игоревна

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ ДЛЯ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ

Учебно-методическое пособие и лабораторные работы
для студентов всех форм обучения специальностей

1-53 01 01 «Автоматизация технологических процессов и производств»,
1-53 01 06 «Промышленные роботы и робототехнические комплексы»,
1-40 01 01 «Программное обеспечение информационных технологий»,
1-40 01 02 «Информационные системы и технологии»

В 2 частях

Ч а с т ь 1

Компьютерная верстка *Т. М.Саковец*

Подписано в печать 10.08.2012. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 11,97. Уч.-изд. л. 9,36. Тираж 100. Заказ 889.
Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.