МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
АГРАРНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

И. И. Гируцкий, А. Г. Сеньков

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА СИСТЕМ АВТОМАТИЗАЦИИ

Рекомендовано Учебно-методическим объединением по образованию в области автоматизации технологических процессов, производств и управления в качестве учебно-методического пособия для студентов учреждений высшего образования по направлению специальности 1-53 01 01-09 «Автоматизация технологических процессов и производств (сельское хозяйство)»

Минск
БГАТУ
2022

УДК 681.51(07)
ББК 32.965я7
Г51

Рецензенты:
кафедра «Робототехнические системы»
Белорусского национального технического университета
(кандидат технических наук, доцент,
заведующий кафедрой *А. Р. Околов*);
кандидат технических наук, доцент,
заведующий лабораторией научного обеспечения испытаний
и информационно-технических технологий
РУП «НПЦ НАН Беларуси по механизации
сельского хозяйства» *В. К. Клыбик*

Гируцкий, И. И.

Г51 Микропроцессорная техника систем автоматизации : учебно-методическое пособие / И. И. Гируцкий, А. Г. Сеньков. – Минск : БГАТУ, 2022. – 224 с.

ISBN 978-985-25-0152-1.

Приведена общая информация о принципах построения и работы микропроцессорных контроллеров, способах представления и хранения данных в цифровых микропроцессорных устройствах. Изложены основы программирования микропроцессорных контроллеров, дан краткий обзор существующих языков программирования международных стандартов. Базовый теоретический материал дополнен практическими примерами использования микропроцессорных контроллеров для решения задач автоматизации сельскохозяйственного производства.

Для студентов направления специальности 1-53 01 01-09 «Автоматизация технологических процессов и производств (сельское хозяйство)».

УДК 681.51(07)
ББК 32.965я7

ISBN 978-985-25-0152-1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Industrie 4.0, или четвертая промышленная революция, идет сегодня, набирает обороты и заключается в развитии робототехники, дальнейшей цифровизации экономики и автоматизации производства и сферы услуг, расширении применения технологий, не требующих обязательного присутствия человека, и искусственного интеллекта. Осуществить этот переход возможно при наличии высококвалифицированных инженерных кадров и программистов.

Автоматизация в сельскохозяйственном производстве часто более сложная, чем в промышленности. Это связано с взаимодействием с биологическими объектами, что вызывает необходимость одновременного контроля по нескольким параметрам, причем разным по своей физической величине, и требует повышения надежности системы. Производственные процессы распределены во времени и пространстве. Многие параметры зависят от случайных факторов. Основные требования, предъявляемые к современным системам управления – надежность, адаптивность (гибкость), возможность управления разнородными объектами и построения распределенных многоуровневых систем. Наиболее полно соответствуют перечисленным требованиям системы управления, выполненные на базе микропроцессорной техники. Инженеры агропромышленного комплекса должны быть готовы к использованию новых технологий и широкому внедрению автоматизированных систем управления (АСУТП) на базе электроники и микропроцессорной техники.

Чтобы обеспечить активное энергосбережение, необходимо не просто установить экономичные устройства, но и управлять ими, используя только необходимое количество энергии. Именно фактор управления имеет решающее значение для достижения максимальной эффективности. Особую роль приобретает интеллектуальное управление биотехническими системами сельскохозяйственного производства. Аккумулятором знаний о системе становится программное обеспечение, так называемый SOFT, выступая главной интеллектуальной и финансовой составляющей любого современного оборудования. Производство программного обеспечения, а не

металлоконструкций позволяет снизить энергоемкость ВВП и интеллектуальную зависимость.

Прародителем термина контроллер стало английское слово control – управление. Первые программируемые контроллеры появились в конце 60-х гг. прошлого столетия и могли выполнять простейшие логические функции. Изобретение микропроцессора позволило расширить функции контроллера, сделать его массовым универсальным устройством, используемым для автоматизации технологических процессов во всех отраслях промышленности и сельского хозяйства. Современный программируемый контроллер общепромышленного применения является специализированным компьютером, предназначенным для построения систем управления технологическими процессами. Главное отличие контроллера от компьютера – развитый интерфейс ввода/вывода электрических сигналов и работа в режиме реального времени. Наличие развитого интерфейса ввода/вывода электрических сигналов делает достаточно простым подключение разнообразных датчиков и исполнительных механизмов к контроллеру. А работа в режиме реального времени обеспечивает принятие управленческих решений в соответствии с динамикой объекта управления.

Применение промышленных контроллеров распространяется практически на все сферы человеческой жизнедеятельности: автоматизация технологических процессов, станки ЧПУ, системы жизнеобеспечения зданий, системы противоаварийной защиты и сигнализации, сбор и архивирование данных, управление дорожным движением, управление космическими кораблями, системы охраны, автоматизация проведения испытаний продукции, медицинское оборудование, системы связи, управление роботами и др.

Контроллеры используются не только как автономные средства локального управления технологическими установками, но и благодаря наличию сетевых интерфейсов являются основой интегрированных систем компьютеризированного управления целыми предприятиями.

Для разработки, внедрения и эксплуатации современных микропроцессорных средств автоматики нужны квалифицированные специалисты, обладающие умением изучения разнообразного технологического оборудования, знаниями в области схемотехники

и программирования. При этом именно управляющая технологическая программа содержит накопленные знания о построении эффективного управления конкретным технологическим объектом. Для разработки прикладного программного обеспечения используются аппаратно-ориентированные и универсальные системы проектирования. Стандарт МЭК 61131-3 предлагает пять языков программирования, как графические, так и текстовые.

Целью данного учебного пособия является формирование базовых знаний в области проектирования и использования для автоматизации технологических процессов современных компьютеризированных контроллеров.

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В СИСТЕМАХ УПРАВЛЕНИЯ

В современном мире все больше функций управления, сбора, передачи и хранения информации человек перекладывает на разнообразные программно-технические устройства. Для управления разнообразным технологическим оборудованием и процессами, которые должны осуществляться экономически выгодно и безопасно, применяется специализированный компьютер, называемый программируемым контроллером. В последние годы программируемые контроллеры получили широкое распространение в различных отраслях экономики благодаря относительной дешевизне, вычислительной мощности, адаптивности и надежности и кардинально изменили технологию управления .

История развития программируемых контроллеров

Сложность создания систем автоматизации (новое производство или модернизация устаревших систем управления действующими технологическими процессами и установками) связана с отсутствием четкой постановки задачи, появлением новых условий и требований в процессе разработки и внедрения. Эти особенности привели к необходимости создать управляющее устройство, алгоритм работы которого можно было бы менять, не переделывая монтажную схему и аппарат управления. В результате возникла логичная идея заменить системы управления с «жесткой» логикой работы (совокупность реле, регуляторов, таймеров, счетчиков и т. п.) на автоматы с программно заданной логикой работы. Так возникли программируемые логические контроллеры 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. Впервые ПЛК были применены в США для автоматизации конвейерного сборочного производства в автомобильной промышленности (1969 г.). В России первый программируемый контроллер был создан во Всесоюзном научно-исследовательском институте релестроения (ВНИИР,

Чебоксары) в 1978 г. Логический контроллер Б-9605 имел модульную конструкцию, что позволяло варьировать число входных/выходных сигналов. Процессор контроллера был выполнен на интегральных микросхемах 155 серии и имел пользовательскую память для размещения прикладных программ объемом 4 Кбайт.

Короткая история и многоплановость применений современной микропроцессорной техники привели к неоднозначности понимания самого термина «микропроцессорный программируемый контроллер». В качестве синонимов часто используются следующие определения: программируемый контроллер, промышленный контроллер, микропроцессорный контроллер или компьютеризированный контроллер.

Контроллер – это специализированный компьютер, предназначенный для построения систем управления технологическими процессами в жестких условиях реального производства. Отличительными особенностями этой специализации являются:

– простой интерфейс ввода с датчиков и вывода на исполнительные механизмы типовых электрических сигналов;

– циклический характер выполнения прикладных (управляющих технологических) программ;

– наличие операционной системы жесткого реального времени и специализированной системы программирования с набором графических и текстовых языков программирования, удовлетворяющих требованиям стандарта МЭК 61131-3;

– частичное или полное пылевлагозащищенное исполнение с возможностью размещения внутри и вне шкафа управления.

Высокие потребительские качества промышленных контроллеров, главными из которых являются высокая функциональность и надежность, а также универсальность применения при невысокой стоимости, привели к их массовому внедрению во все отрасли промышленности и сельского хозяйства. Поэтому сейчас является актуальной для специалистов различных профилей, выступающих в роли заказчика, разработчика или пользователя, ликвидация безграмотности в области современных микропроцессорных систем управления.

Современный специалист должен уметь структурировать проектные решения аппаратного и программного обеспечения, организации

человеко-машинного интерфейса и обмена информацией между распределенными подсистемами с использованием идеологии локальных вычислительных сетей (ЛВС). В информационно-управляющих системах с ПЛК сравнительно легко вводятся новые алгоритмы управления путем замены программы, без перемонтажа и замены аппаратуры. Поэтому современные информационно-управляющие системы представляют собой принципиально новую технологию управления, обладающую большой гибкостью и новыми возможностями в повышении эффективности производства.

Такие информационно-управляющие системы имеют следующие особенности:

• характеризуются многократным (в сотни, тысячи и более раз) увеличением объемов перерабатываемой информации о состоянии объекта управления при принятии управленческих решений;

• строятся на базе микропроцессорных контроллеров общепромышленного применения, промышленных и персональных компьютеров с встроенными вычислительными сетями, что позволяет создавать распределенные и многоуровневые системы управления;

• реализуют средствами программно-технического комплекса как информационно-вычислительные, так и управляющие функции (логическое и дисплейное управление, автоматическое регулирование, технологические защиты, блокировки и др.), то есть впервые интегрированный программно-технический комплекс заменяет ранее информационно не связанные локальные подсистемы (КИП, автоматическое регулирование, дистанционное управление, технологические защиты и др.);

• основные функциональные задачи реализуются в виде прикладного программного обеспечения, при этом избыточные программно-технические возможности универсальных устройств управления можно использовать для функциональной диагностики технологического оборудования, что придает черты «интеллектуальности» системе управления и значительно повышает надежность выполнения технологических процессов.

Многочисленные производители программируемых контроллеров, среди которых такие известные бренды, как Siemens, Omron, Allen Bradley, Bernecker & Rainer и другие предлагают широкий

продуктовый ряд изделий. Выбор определенного контроллера зависит от решаемой задачи, предыдущего опыта, качества фирменного сервиса и пр. Для понимания степени распространения программируемых контроллеров можно сослаться на данные фирмы GM, на предприятиях которой используются свыше 100 тыс. контроллеров в системах управления различным оборудованием.

Современный ПЛК представляет собой промышленный цифровой компьютер, предназначенный для выполнения функций управления.

ПЛК постоянно контролирует состояние устройств ввода информации о состоянии управляемого объекта и принимает решения на основе пользовательской программы для управления состоянием выходных устройств. Упрощенное представление состава и принципа действия ПЛК хорошо демонстрирует рис. 1. Из него видно, что ПЛК имеет три основные секции:

– входную;

– выходную;

– центральную.

Имеется еще источник питания. Возможно подключение к ПЛК внешнего ПК для программирования и отладки.

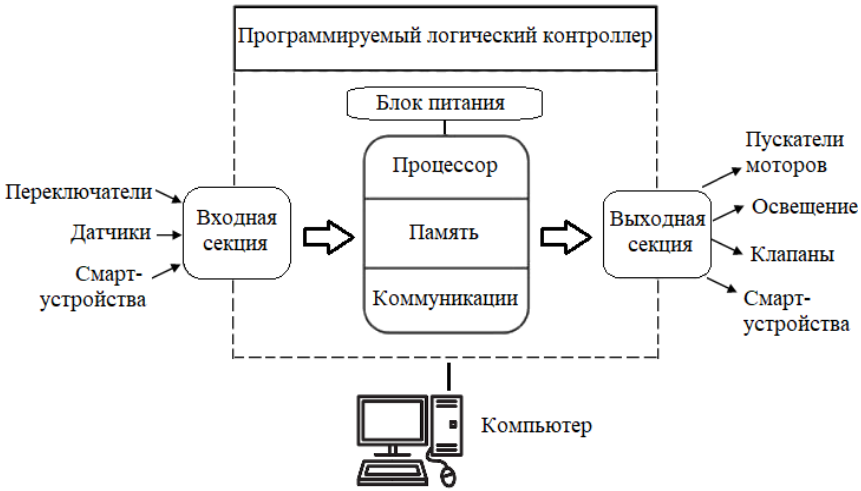


Рис. 1. Состав и принцип действия ПЛК

Центральная секция содержит центральный процессор (ЦП), память и систему коммуникаций. Она выполняет обработку данных, принимаемых от входной секции данных, и передает результаты обработки в выходную секцию. Входная секция ПЛК обеспечивает ввод в центральную секцию состояния переключателей, датчиков и смарт-устройств. Через выходную секцию ЦП управляет внешними исполнительными устройствами, среди которых могут быть электромагнитные пускатели моторов, источники света, клапаны и смарт-устройства.

Принцип работы ПЛК несколько отличается от других микропроцессорных устройств. Программное обеспечение универсальных ПЛК состоит из двух частей:

– системного программного обеспечения – операционной системы;

– прикладной программы пользователя.

Системное программное обеспечение (СПО) непосредственно контролирует аппаратные средства ПЛК. СПО отвечает за тестирование и индикацию работы памяти, источника питания, модулей ввода/вывода и интерфейсов, таймеров и часов реального времени. Система исполнения кода прикладной программы является составной частью СПО. Система исполнения включает драйверы модулей ввода/вывода, загрузчик кода программ пользователя, интерпретатор команд и отладочный монитор. Код СПО расположен в ПЗУ и может быть изменен только изготовителем ПЛК.

Код прикладной программы размещается в энергонезависимой памяти, чаще всего это электрически перепрограммируемые микросхемы. Изменение кода прикладной программы выполняется пользователем ПЛК при помощи системы программирования и может быть выполнено многократно.

Первая часть – операционная система ПЛК – управляет согласованной работой всех узлов и микросхем контроллера.

Вторая часть – прикладная программа пользователя – реализует заданный пользователем алгоритм управления объектом.

ПЛК работает по циклическому принципу (рис. 2).

В самом начале цикла ПЛК сканирует состояния входов, на которые поступают сигналы от датчиков и устройств. Считывание сигналов со входов контроллера выполняется операционной системой и результат записывается в память входов контроллера (см. рис. 1).

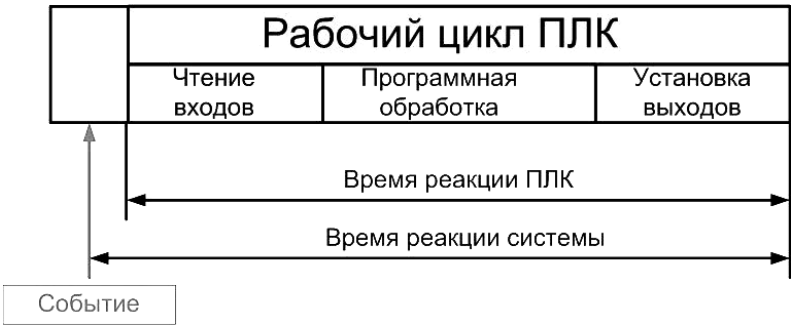


Рис. 2. Циклический режим работы ПЛК

После чего управление передается прикладной программе и в соответствии с алгоритмом управления происходит вычисление состояния выходов. Результат также записывается в память выходов.

В конце рабочего цикла контроллера происходит установка физических выходов.

Затем цикл выполнения программы повторяется. При изменении состояния входов (датчики) осуществляется изменение состояния выходов (исполнительные механизмы).

За счет этого обеспечивается максимальная простота построения прикладной программы – ее создатель не должен знать, как производится управление аппаратными ресурсами. Необходимо знать с какого входа приходит сигнал и как на него реагировать на выходах.

Таким образом, смысл работы ПЛК заключается в том, чтобы:

– собрать нужную информацию от объекта (объектов) управления через свои входы;

– обработать ее в соответствии с заданными алгоритмами;

– обработать коммуникационные интерфейсы;

– получить команды и сигналы с верхнего уровня управления;

– выдать управляющие команды на свои выходы;

– выдать необходимую информацию на верхний уровень управления (например, на сенсорную панель местного управления или в SCADA);

– продиагностировать себя самого.

А потом повторить сначала все перечисленные шаги.

Таким образом, ПЛК можно считать специализированным миникомпьютером. Причем ПЛК изначально ориентирован на эксплуатацию в цехах промышленных предприятий, где имеется множество источников электромагнитных помех, а температура может быть как положительной, так и отрицательной. Дополнительно к минимизации воздействия вышеуказанных факторов необходимо предусмотреть и защиту от агрессивной внешней среды, включающей пыль, брызги технологических жидкостей и паровоздушные взвеси. В таких случаях предусмотрена установка ПЛК в защитные шкафы или в удаленных помещениях. Отдельные модули могут размещаться на удалении до сотен метров от основного комплекта ПЛК и эксплуатироваться при экстремальных внешних температурах. Согласно МЭК 61131, для ПЛК с наружной установкой допустима температура 5 ºС–55 ºС. Для устанавливаемого в закрытых шкафах ПЛК необходимо обеспечить рабочий диапазон 5 ºС–40 ºС при относительной влажности 10 %–95 % (без образования конденсата).

ПЛК имеют ряд особенностей, отличающих их от прочих электронных приборов:

– в отличие от микроконтроллера (однокристального компьютера) – микросхемы, предназначенной для управления электронными устройствами – областью применения ПЛК обычно являются автоматизированные процессы промышленного производства в контексте производственного предприятия;

– в отличие от компьютеров ПЛК ориентированы на работу с агрегатами машин через развитый ввод сигналов датчиков и вывод сигналов на исполнительные механизмы, ориентированных на принятие решений и управление оператором;

– в отличие от встраиваемых систем ПЛК изготавливаются как самостоятельные изделия;

– наличие расширенного числа логических операций и возможность задания таймеров и счетчиков;

– все языки программирования ПЛК имеют легкий доступ к манипулированию битами в машинных словах, в отличие от большинства высокоуровневых языков программирования современных компьютеров.

Типы ПЛК

Современные ПЛК далеко ушли от первых упрощенных реализаций промышленного контроллера, но заложенные в систему управления универсальные принципы были стандартизированы и успешно развиваются уже на базе новейших технологий.

По конструктивному исполнению ПЛК бывают моноблочные и модульные.

В корпусе моноблочного ПЛК наряду с ЦП, памятью и блоком питания размещается фиксированный набор входов/выходов. Внешний вид некоторых моделей моноблочных ПЛК показан на рис. 3. Моноблочные функционально завершенные ПЛК могут включать в себя небольшой дисплей и кнопки управления. Дисплей предназначен для отображения текущих рабочих параметров и вводимых с помощью кнопок команд рабочих программ и технологических установок.

Рис. 3. Моноблочные ПЛК

В модульных ПЛК используют отдельно устанавливаемые модули входов/выходов. Согласно требованиям МЭК 61131, их тип и количество могут меняться в зависимости от поставленной задачи и обновляться с течением времени. Структура ПЛК подобной концепции представлены на рис. 4. Подобные ПЛК могут действовать в режиме «ведущего» и расширяться «ведомыми» ПЛК через интерфейс Ethernet. Модульные ПЛК комбинируются из отдельных функциональных модулей, совместно закрепляемых на стандартной монтажной рейке. В зависимости от количества обслуживаемых входов и выходов, устанавливается необходимое количество модулей ввода и вывода.

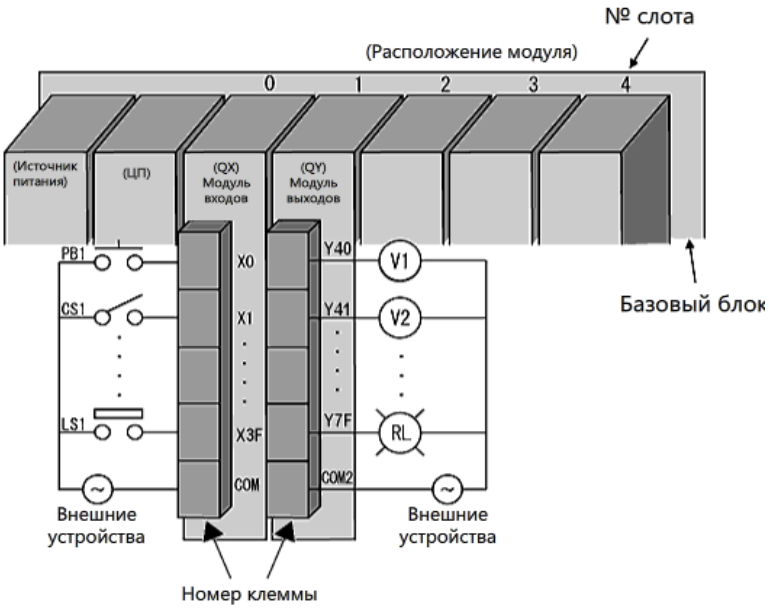


Рис. 4. Структура модульного ПЛК

Источник питания может быть встроенным в основной блок ПЛК, но чаще выполнен в виде отдельного блока питания (БП), закрепляемого рядом на стандартной рейке. Блок питания небольшой мощности представлен на рис. 5.

Рис. 5. Блок питания для ПЛК: внешний вид

Первичным источником для БП чаще всего служит промышленная сеть 24/48/110/220/400 В, 50 Гц. Другие модели БП могут использовать в качестве первичного источник постоянного напряжения на 24/48/125 В. Стандартными для промышленного оборудования и ПЛК являются выходные напряжения БП: 12, 24 и 48 В. В системах повышенной надежности возможна установка двух специальных резервированных БП для дублирования электропитания.

Для сохранения информации при аварийных отключениях сети электропитания в ПЛК используют дополнительную батарею.

Тип ПЛК выбирается при проектировании системы управления и зависит от поставленных задач и условий производства. В отдельных случаях это может быть моноблочный ПЛК с ограниченными функциями, имеющий достаточное количество входов и выходов. В других условиях потребуются ПЛК с расширенными возможностями, позволяющими использовать распределенную конфигурацию с удаленными модулями входа/выхода и с удаленными пультами управления технологическим процессом.

Связь между удаленными блоками и основным ядром ПЛК осуществляется через помехозащищенные полевые шины по медным кабелям и оптическим линиям связи. В отдельных случаях, например для связи с подвижными объектами, применяют беспроводные технологии, чаще всего это сети и каналы Wi-Fi. Для взаимодействия с другими ПЛК могут применяться как широко известные интерфейсы – RS-232 и RS-485, так и более помехозащищенные промышленные варианты типа Profibus и CAN.

Устройство программирования и человеко-машинный интерфейс

Вне самого ПЛК есть два очень важных компонента: устройство программирования и человеко-машинный интерфейс (human-machine interface – HMI). Устройство программирования может быть настольным компьютером, ноутбуком или ручным инструментом от того же производителя. Существуют также фиксированные ПЛК ввода/вывода со встроенными дисплеями и кнопками, которые позволяют программам записываться непосредственно в ПЛК.

Хотя устройство программирования позволяет пользователю просматривать и изменять код, выполняемый на ПЛК, HMI обеспечивает более высокий уровень абстракции, моделируя систему управления в целом. На рис. 6 показан встроенный сенсорный экран, который можно использовать в диспетчерской или в «поле» ближе к процессу. Эти типы интерактивных дисплеев очень распространены и часто устанавливаются непосредственно на корпусе ПЛК или поблизости для использования оператором. «Поле» – это площадь завода или предприятия, на котором выполняется фактическое управление технологическим процессом. Здесь находятся насосы, двигатели, клапаны, датчики температуры и давления, теплообменники, массовые расходомеры, роботизированные механизмы и прочее оборудование.

Рис. 6. Панель HMI оператора: внешний вид

В современных крупных, сложных отраслях HMI стал важной особенностью в реализации и развертывании системы управления. Как следует из его названия, человеко-машинный интерфейс представляет собой окно пользователя в схему или процесс управления. Он позволяет пользователю контролировать, взаимодействовать и, при необходимости, отключать систему управления. До современных HMI операторы завода полагались на стенды аналоговых датчиков и лампочек, чтобы понять состояние процессов.

Применение ПЛК

ПЛК широко используются в пищевой, перерабатывающей и упаковочной отраслях, а также в сельском хозяйстве. На базе ПЛК созданы автоматизированные системы управления автоклавами и дозирующими установками, насосами и печами, маслоотжимными прессами и пресс-грануляторами, экструдерами и другим оборудованием. ПЛК управляют процессами хлебопечения и копчения, производством кисломолочных продуктов и консервированием овощной продукции, процессом переработки сахара-сырца, заморозкой и охлаждением полуфабрикатов. Средства автоматизации поддерживают микроклимат в витринах и морозильных камерах, в теплицах и овощехранилищах. Они контролируют процесс производства мороженого и соков, молока и пива, крахмала, масла и многих других продуктов. На базе ПЛК автоматизированы хлебокомбинаты и кондитерские фабрики, молочные и мясокомбинаты, пивоварни и маслозаводы.

Создание на базе ПЛК автоматизированных систем управления в пищевой, перерабатывающей, упаковочной отраслях и в сельском хозяйстве позволяет производителям повысить качество выпускаемой продукции, обеспечить ее сохранность, избавиться в сложных технологических процессах от влияния «человеческого фактора», уменьшить количество обслуживающего персонала, что в целом способствует снижению себестоимости продукции и повышению ее конкурентоспособности.

АРХИТЕКТУРА ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА ОБЩЕПРОМЫШЛЕННОГО ПРИМЕНЕНИЯ

Архитектурой контроллера называют набор его основных компонентов и связей между ними.

Во многих отношениях, архитектура ПЛК напоминает архитектуру персонального компьютера общего назначения со специализированными модулями ввода/вывода (I/O). Однако некоторые важные характеристики отличают эти устройства. Во-первых, ПЛК намного более надежны и рассчитаны на безотказную работу в течение многих лет – и это их самая важная особенность. Во-вторых, ПЛК могут использоваться в условиях промышленного производства, где им приходится работать в условиях серьезного электромагнитного излучения, вибрации, экстремальных температур и влажности. В-третьих, ПЛК легко обслуживаются техническим персоналом на производстве.

Типовой состав ПЛК (рис. 7) включает центральный процессор, память, сетевые интерфейсы и устройства ввода/вывода. Иногда эта конфигурация дополняется устройством для программирования и пультом оператора, устройствами индикации.

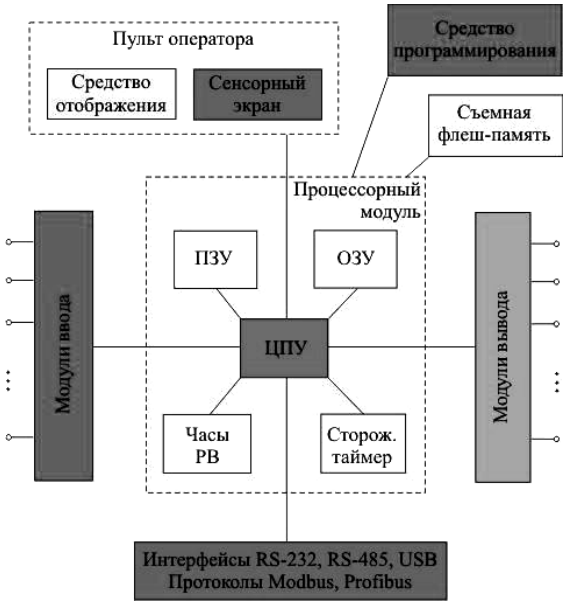


Рис. 7. Типовая архитектура ПЛК

Центральное процессорное устройство

Процессорный модуль включает в себя микропроцессор, или центральное процессорное устройство (ЦПУ), запоминающие устройства, часы реального времени и сторожевой таймер. Термины «микропроцессор» и «процессор» в настоящее время стали синонимами.

ЦПУ – электронный блок, исполняющий машинные инструкции (код программ), главная часть аппаратного обеспечения ПЛК. Иногда его называют микропроцессором или просто процессором.

Функции ЦПУ. Главная функция – управление всеми операциями ПЛК, а именно:

– выбор команды из памяти, их дешифровка и выполнение;

– получение данных из оперативной памяти, выполнение с ними арифметических и логических операций, передача их на внешние устройства;

– формирование сигналов, необходимых для работы внутренних узлов и внешних устройств;

– временное хранение результатов выполненных операций, переданных сигналов и других данных;

– прием запросов от внешних устройств и их обработка.

Таким образом, ЦПУ является «мозгом» ПЛК.

Основными блоками ЦПУ являются: устройство управления и арифметико-логическое устройство. Арифметико-логическое устройство (АЛУ) предназначено для выполнения предусмотренных в ЭВМ арифметических и логических операций. Участвующие в операциях данные выбираются из оперативной памяти (оперативное запоминающее устройство – ОЗУ), результаты операций отсылаются в ОЗУ. ОЗУ представляет собой единый массив памяти, непосредственно доступный процессору для записи и чтения данных, а также считывания программного кода.

Устройство управления (УУ) – координирует работу процессора, посылая в определенной временной последовательности управляющие сигналы в другие блоки ПЛК, обеспечивая их соответствующее функционирование и взаимодействие друг с другом.

Память ПЛК

Емкость памяти определяет количество переменных (тегов), которые могут быть обработаны в процессе функционирования ПЛК.

Память делят на несколько уровней иерархии, в зависимости от частоты использования хранящихся в ней данных и быстродействия. Основными типами памяти является постоянное запоминающее устройство (ПЗУ), ОЗУ и набор регистров.

Регистры являются самыми быстродействующими элементами памяти, поскольку они используются АЛУ для исполнения элементарных команд процессора.

ПЗУ используют для хранения редко изменяемой информации, такой как операционная система, драйверы устройств, загрузчик, исполняемый модуль программы пользователя.

ОЗУ используется для хранения данных, которые многократно изменяются в процессе работы контроллера, например, значения тегов, результаты промежуточных вычислений, диагностическая информация, массивы, выводимые на графики, данные для отображения на дисплее.

В качестве ПЗУ (или ROM – Read Only Memory) обычно используется электрически стираемая перепрограммируемая память (EEPROM – Electrically Erasable Programmable ROM). Разновидностью EEPROM является флэш-память, принцип действия которой основан на хранении заряда в конденсаторе, образованном плавающим затвором и подложкой МОП-транзистора. Особенностью флэш-памяти является ее энергонезависимость, то есть сохраняемость данных при выключенном питании. Стирание и перезапись во флэш-памяти выполняется не отдельными ячейками, а большими блоками, поэтому она получила название, происходящее от английского flash – «вспышка».

В качестве ОЗУ современные микропроцессоры используют статическую память (SRAM – Static Random Access Memory) и динамическую (DRAM – Dynamic Random Access Memory), SDRAM (Synchronous DRAM). SRAM выполняется на триггерах, информация в которых сохраняется неограниченно долго при наличии питания. В динамической памяти информация хранится на конденсаторах и поэтому DRAM требует периодической регенерации (перезарядки конденсаторов). К недостаткам SRAM памяти относится ее высокая стоимость. Достоинством является высокое быстродействие по сравнению с DRAM памятью. Оба типа памяти (DRAM и SRAM) не могут сохранять информацию при отключении питания ПЛК. Поэтому некоторые типы ПЛК используют батарейное

питание памяти для сохранения работоспособности системы автоматизации после кратковременного прерывания питания.

Сторожевой таймер (Watchdog Timer – WDT) – аппаратно реализованный механизм безопасности, который позволяет вернуть систему в рабочий режим в случае сбоя. Как правило, WTD состоит из счетчика и тактирующего устройства. Значение счетчика по сигналам тактового устройства постоянно уменьшается. Когда оно достигает нуля, генерируется короткий импульс, который сбрасывает и перезапускает систему.

Работающему процессору необходимо периодически, до срабатывания таймера, обновлять значение счетчика, иначе WTD вызовет перезагрузку системы. После обновления счетчика его значение вновь продолжит уменьшаться. Проще говоря, WDT постоянно «следит» за выполнением кода и перезагружает систему, если программное обеспечение зависает или больше не выполняет правильную последовательность кода.

Часы реального времени (РВ) представляют собой кварцевые часы, которые питаются от батарейки и поэтому продолжают идти при выключенном ПЛК. Часы РВ используются, например, для управления уличным освещением в зависимости от времени суток, в системах охраны объектов и других случаях, когда необходима привязка данных или событий к астрономическому времени.

Системная шина

Все данные между процессором, регистрами, памятью и I/O-устройствами (устройствами ввода/вывода) передаются по шинам. Системная шина (рис. 8) – совокупность линий передачи всех видов сигналов (в том числе данных, адресов и управления), предназначенных для передачи информации между микропроцессором и остальными электронными блоками ПЛК. В простейшем случае под понятием шина подразумевают параллельно проложенные провода, по которым передается двоичная информация. При этом по каждому проводу передается отдельный двоичный разряд. Информация может передаваться в одном направлении, как, например, для шины адреса или шины управления, или в различных направлениях (для шины данных). По шине данных информация передается к процессору или от процессора в зависимости

от операции записи или чтения, которую в данный момент осуществляет процессор.

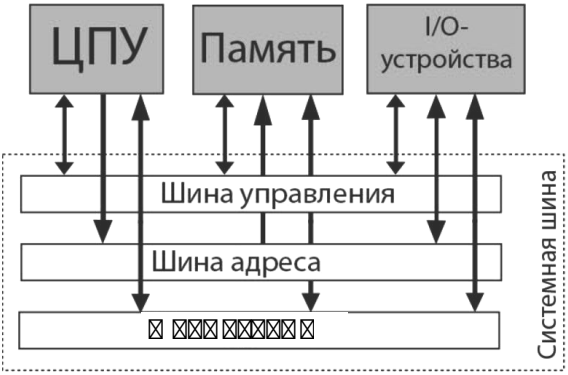


Рис. 8. Типовая архитектура ПЛК

Системная шина представлена совокупностью трех специализированных шин: шины данных, шины адреса и шины управления.

Шина данных предназначена для пересылки кодов обрабатываемых данных, а также машинных кодов команд между устройствами ЭВМ. По шине данных передается информация в микропроцессор и из него.

Шина адреса несет адрес (номер) той ячейки памяти или того порта ввода/вывода, который взаимодействует с микропроцессором. На шину адреса микропроцессор выводит информацию о номере (адресе) той ячейки памяти или устройства, с которым он собирается производить обмен информацией.

Шина управления несет сигналы управления, обеспечивающие правильное взаимодействие блоков ПЛК друг с другом и с внешней средой. В состав шины управления входят сигналы, управляющие процессом передачи информации, например:

– запись (write, WR);

– чтение (read, RD; запись или чтение определяется процессором);

– обмен с памятью (memory require, MREQ);

– обмен с устройством ввода/вывода (I/O require, ioreq или их комбинации).

Функционирование МПС сводится к следующей последовательности действий: получение данных от различных периферийных устройств (с клавиатуры терминала, от дисплеев, из каналов связи, от различного типа внешних запоминающих устройств), обработка данных и выдача результатов обработки на периферийные устройства (ПУ).

Микропроцессор выдает на шину адреса номер (адрес) ячейки ПЗУ, в которой хранится очередная команда, и из шины управления в ПЗУ поступают сигналы, обеспечивающие считывание содержимого указываемой шиной адреса ячейки памяти. ПЗУ выдает запрошенную команду на шину данных, откуда она принимается в микропроцессор. Здесь команда расшифровывается. Если данные, действия над которыми предусматривает команда, находятся в регистрах микропроцессора, то микропроцессор приступает к выполнению указанной в команде операции. Если при расшифровке команды выяснится, что участвующие в операции данные находятся в ОЗУ, то микропроцессор выставляет на шину адреса адрес ячейки, хранящей эти данные; после выдачи данных из ОЗУ микропроцессор принимает их через шину данных, затем выполняется операция над данными. После завершения текущей команды на шину адреса выдается адрес следующей команды, и описанный процесс повторяется.

Порты ввода/вывода

Порты ввода/вывода предназначены для общения ПЛК с внешними устройствами. С их помощью ПЛК передает информацию другим устройствам и принимает информацию от них. Входные и выходные данные ПЛК часто сокращаются как I/O – Inputs/Outputs (входы/выходы). Порты ввода/вывода – это часть ПЛК, которая соединяет «мозг» ПЛК (ЦПУ) с внешним миром (то есть с технологическим оборудованием). В системе ПЛК обычно есть отдельные модули для входных сигналов (порты ввода) и отдельные модули для выходов (порты вывода). Модуль входного сигнала обнаруживает состояние входных сигналов – кнопки, переключатели, датчики температуры и др. Модуль вывода контролирует приборы – реле, стартеры двигателей, индикаторы и др.

Дискретные (или двоичные) сигналы – это сигналы, которые могут быть только включены или выключены. Это самый простой и наиболее распространенный тип ввода/вывода. В системах автоматизации очень распространены двоичные сигналы, которые поступают от концевых выключателей, датчиков охранной или пожарной сигнализации, датчиков заполнения емкостей, датчиков сбегания ленты на конвейере, датчиков приближения и т. п.

Вывод дискретных сигналов используется для управления состоянием включено/выключено исполнительных устройств. Примерами дискретных выходов могут быть замыкающие или размыкающие автоматы защиты цепи, запуск или остановка генератора, открытие или закрытие клапана подачи воды или включение/отключение светового сигнала тревоги.

Другой распространенной формой ввода/вывода ПЛК является аналоговый ввод/вывод. Аналоговый ввод/вывод относится к сигналам, которые имеют диапазон значений намного больше, чем просто 1 или 0. Например, аналоговый сигнал может производить напряжение в диапазоне 0–10 В постоянного тока. Сигнал может быть 2 В; 3 В; 8,5 В и т. д. В мире ПЛК, модули ввода аналога обычно измеряют сигнал в одном из следующих диапазонов: от –10 до 10 В; от 0 до 10 В; от 1 до 5 В; от 0 до 1 мА или от 4 до 20 мА. В основном модуль аналогового ввода измеряет напряжение или ток от входного устройства. Существуют и другие типы аналоговых сигналов, но эти, безусловно, наиболее распространены. Таким же образом модуль аналогового выхода может подавать сигналы напряжения или тока в одном из упомянутых выше диапазонов.

Аналоговый сигнал – это диммер света. При регулировке ручки или ползунка диммера свет станет более тусклым или ярким в зависимости от направления регулировки. Аналогично аналоговый вход в ПЛК может увеличиваться или уменьшаться с очень небольшими приращениями. И ПЛК может производить аналоговый выход, который действует таким же образом.

Некоторыми реальными примерами аналоговых входов в промышленных условиях будут датчики температуры двигателя (термометры сопротивления, термопары и т. д.), датчики давления масла и весы. Датчик температуры может передать диапазон

температур от –50 ºС до +150 ºС, соответствующий 4–20 мА. Весовая шкала может сообщать диапазон от 0 до 1000 кг, соответствующий 0 до 10 В. И так далее и тому подобное. Аналоговые выходы могут использоваться для управления выходной мощностью генератора, положением иглы на аналоговом измерителе давления и др. Аналоговый выход 0–3 В может быть использован для того, чтобы управлять генератором от 0 до 2000 кВт. Или аналоговый выход 4–20 мА может быть использован для того чтобы управлять датчиком температуры от –20 ºС до +200 ºС.

СОПРЯЖЕНИЕ ДАТЧИКОВ И ИСПОЛНИТЕЛЬНЫХ МЕХАНИЗМОВ С КОНТРОЛЛЕРОМ

Подключение дискретных датчиков к ПЛК

В зависимости от модели ПЛК наиболее распространены дискретные входы (digital inputs – DI) на 5, 12, 24 В постоянного тока (VDC) и 110, 220 В переменного тока (VAC).

Чаще всего в ПЛК используются DI 24VDC (рис. 9). Если на такой дискретный вход подать напряжение 24 В постоянного тока, то ПЛК увидит на этом входе наличие сигнала – «логическую единицу», или True («Истина»). А если подать 0 В (разорвать электрическую цепь), то ПЛК определит на входе отсутствие сигнала – «логический ноль», или False («Ложь»).

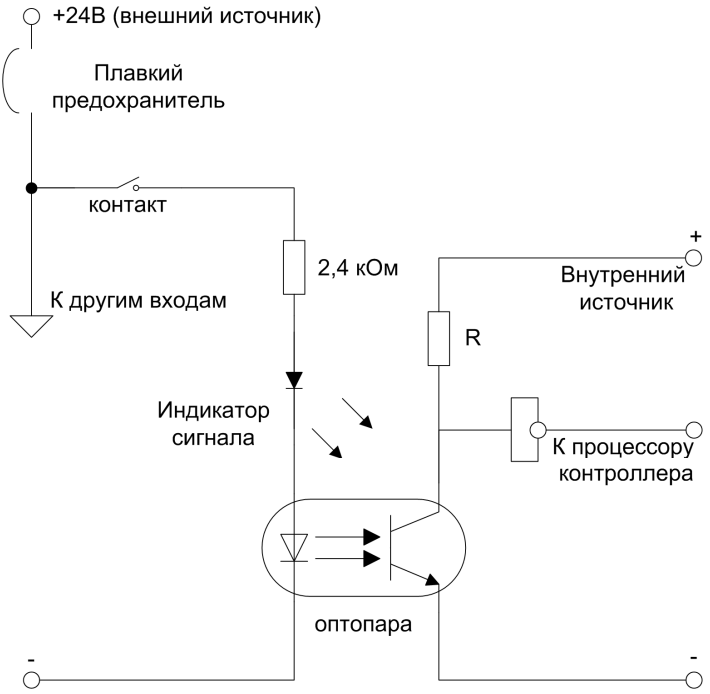


Рис. 9. Принцип работы дискретных входов

Дискретный вход ПЛК, как правило, включает в себя индикатор состояния (светодиод), гальваническую развязку и защиту от неверного подключения. Помимо этого, каждый дискретный вход оснащен аналоговым фильтром, подавляющим 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. Обобщенная структурная схема дискретного входа ПЛК приведена на рис. 10.

Рис. 10. Обобщенная структурная схема дискретного входа ПЛК

В цифровой электронике можно встретить понятие «подтягивающий резистор» обычно он применяется во входных цепях микроконтроллеров. Их смысл использования заложен в определении логического уровня. Подтягивающий резистор – резистор, включенный между проводником, по которому распространяется электрический сигнал, и питанием (pull-up resistor – подтягивающий вверх/подтягивающий к питанию резистор), либо между проводником и землей (pull-down resistor – стягивающий резистор). Подтягивающий резистор нужен, чтобы гарантировать на логическом входе, с которым соединен проводник, высокий (в первом случае) или низкий (во втором случае) уровень.

Логический уровень характеризуется некоторым диапазоном напряжений. В идеале, сигнал высокого логического уровня должен

быть равен ровно 5 В, а сигнал низкого уровня – ровно 0 В. Однако в реальных дискретных входах ПЛК не могут быть обеспечены подобные точные уровни напряжения, поэтому они могут принимать сигналы высокого и низкого уровней даже при значительном отклонении напряжения от идеальных величин. «Приемлемые» напряжения входного сигнала лежат в диапазоне от 0 до 0,8 В для низкого логического уровня и от 2 до 5 В для высокого логического уровня. Между логическими уровнями идет зона неопределенности.

Свободный вход микроконтроллера может под воздействием различных внешних и внутренних помех оказаться в непредсказуемом состоянии. Так, напряжение, вызванное помехой, может быть понято как «1» или «0» или попасть в зону неопределенности. В качестве примера рассмотрим схему подключения кнопки ко входу ПЛК, приведенную на рис. 11.

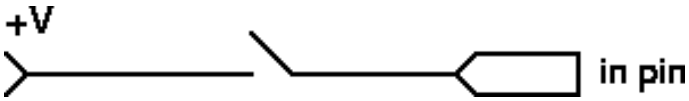


Рис. 11. Подключение кнопки к дискретному входу ПЛК

Необходимо, чтобы, когда кнопка не нажата (цепь разомкнута), вход фиксировал отсутствие напряжения. Но в данном случае вход находится в «никаком» состоянии. Он может срабатывать и не срабатывать хаотично, непредсказуемым образом. Причина этого – шумы, образующиеся вокруг: провода действуют как маленькие антенны и производят электричество из электромагнитных волн среды. Чтобы гарантировать отсутствие напряжения при разомкнутой цепи, рядом со входом ставится стягивающий резистор (рис. 12).

Теперь нежелательный ток будет уходить через резистор в землю. Для стягивания используются резисторы больших сопротивлений (10 кОм и более). В моменты, когда цепь замкнута, большое сопротивление резистора не дает большей части тока идти в землю: сигнал пойдет к входному контакту. Если бы сопротивление резистора было мало (единицы Ом), при замкнутой цепи произошло бы короткое замыкание.

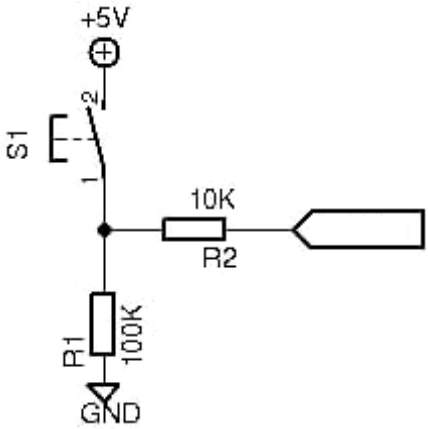


Рис. 12. Подключение кнопки к дискретному входу ПЛК с использованием стягивающего резистора

Аналогично подтягивающий резистор удерживает вход в состоянии логической единицы, пока внешняя цепь разомкнута (рис. 13).

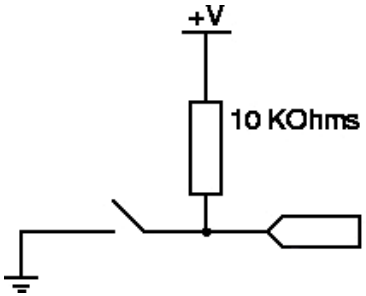


Рис. 13. Подключение кнопки к дискретному входу ПЛК с использованием подтягивающего резистора

Дребезг контактов и способы его подавления

Дребезг контактов – явление, происходящее в электромеханических коммутационных устройствах и аппаратах (кнопках, реле, герконах, переключателях, контакторах, магнитных пускателях и др.), длящееся некоторое время после замыкания электрических контактов. Обычные кнопки представляют собой механические устройства с пружинным контактом. При нажатии на кнопку

(см. схему подключения на рис. 12) сигнал не просто меняется от низкого до высокого, он на протяжении нескольких миллисекунд неоднократно меняет свое значение, прежде чем установится уровень False. Отличие ожидаемого процесса от реального иллюстрируют осциллограммы сигнала с кнопки, приведенные на рис. 14.

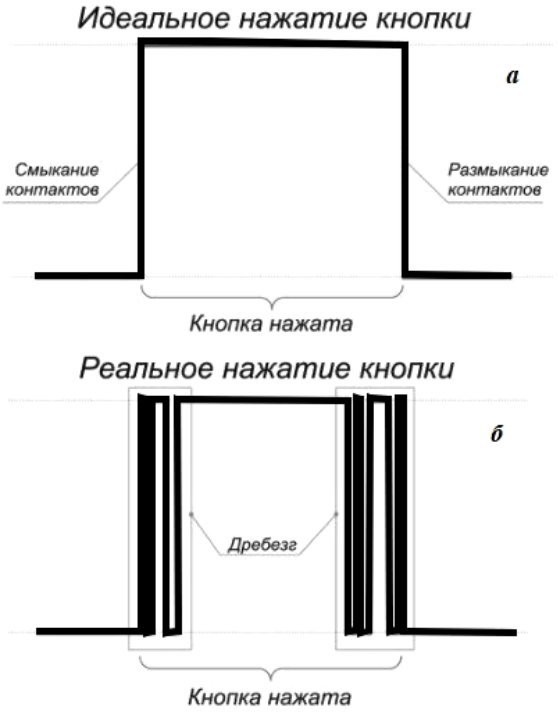


Рис. 14. Эффект дребезга контактов:
а – идеальное нажатие кнопки; *б* – реальное нажатие кнопки

Дребезг наблюдается и при размыкании электромеханических контактов. Практически все механические кнопки, контакторы и переключатели в той или иной степени подвержены дребезгу.

Аппаратный способ устранения дребезга основан на использовании сглаживающих фильтров. Сглаживающий фильтр, как следует из названия, занимается сглаживанием всплесков сигналов за счет добавления в схему элементов, имеющих своеобразную «инерцию» по отношению к таким электрическим параметрам как

ток или напряжение. Самым распространенным примером таких «инерционных» электронных компонентов является конденсатор, схема включения которого показана на рис. 15. Он может «поглощать» все резкие пики, медленно накапливая и отдавая энергию, точно так же как это делает пружина в амортизаторах.

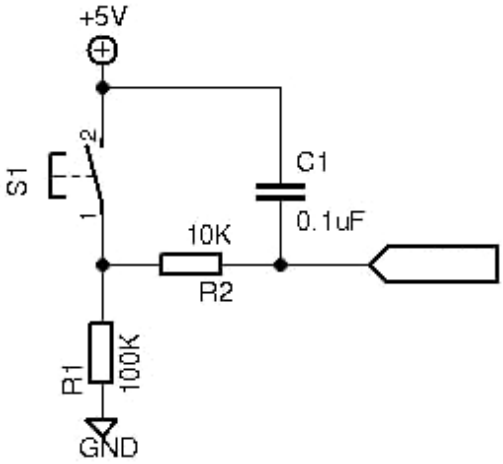


Рис. 15. Модифицированная схема подключение кнопки со стягивающим резистором для устранения дребезга

За счет инерции устройство как утюгом походит по «мятому» сигналу с большим количеством пиков и впадин, создавая пусть и не идеальную, но вполне гладкую кривую, у которой легче определить уровень срабатывания (рис. 16).

Рис. 16. Сглаженный сигнал срабатывания кнопки

Гальваническая развязка цепей датчика и ПЛК

Одной из основных проблем построения дискретных входов является изоляция цепей датчика и контроллера. Модули ввода/вывода должны надежно отделять чувствительные блоки обработки информации от агрессивной промышленной среды электрических станций и подстанций, заполненной помехами, возмущениями, скачками и провалами токов и напряжений. Любая, даже самая агрессивная, атака не должна преодолевать барьеры модулей дискретных входов/выходов и повреждать именно эти модули, а не более сложные и дорогостоящие ЦАП, процессоры, модули памяти и прочее. Кроме того, вычислительные электронные компоненты микропроцессорных терминалов работают со своими уровнями напряжения, а напряжение модулей дискретных входов/выходов должно быть согласовано с параметрами тока, используемого на конкретном объекте.

Изоляция цепей датчика строится на основе гальванической развязки. Основной элемент большинства ячеек дискретных входов – оптрон. Оптрон создает гальваническую развязку и надежно отделяет вычислительную схему микропроцессорного реле от внешней среды (рис. 17). Для согласования номиналов оптрона с рабочими параметрами напряжения датчика используют резисторные делители напряжения (резисторы R1m, R2 на рис. 17). Номинальные сопротивления и мощности резисторов выбирают с учетом величины напряжения источника питания датчика.

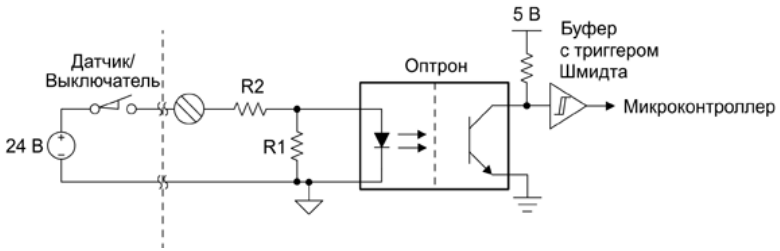


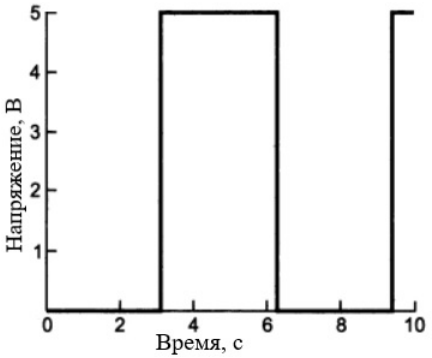
Рис. 17. Реализация дискретного входа на основе оптрона с ограничением тока резистивным делителем

Подключение аналоговых датчиков к ПЛК

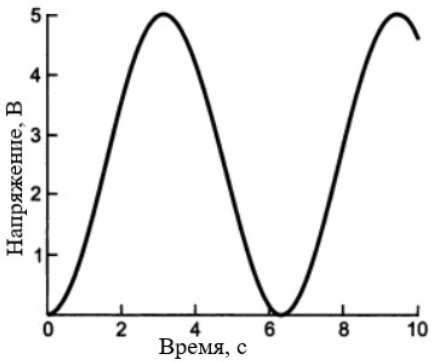
Аналоговый вход (AI, analog input) нужен для ввода в ПЛК значений температуры, давления и других физических величин, которые

измеряются соответствующими датчиками. Эти датчики передают на AI ПЛК значения непрерывно измеряемой физической величины в виде электрического сигнала, который называют аналоговым.

Графики на рис. 18 показывают, чем отличаются друг от друга аналоговые и цифровые сигналы. Слева – прямоугольные импульсы, амплитуда которых принимает только два значения: 0 и 5 В. Справа изображен фрагмент косинусоидального сигнала. Несмотря на то, что его амплитуда находится в тех же границах (0 и 5 В), аналоговый сигнал принимает бесконечное число значений между этими двумя.



а



б

Рис. 18. Цифровые (*а*) и аналоговые (*б*) сигналы

Аналоговые сигналы нельзя представить конечным числом состояний, теоретически они могут иметь бесконечное число значений в пределах некоторого диапазона. Допустим, солнечный свет – это аналоговый сигнал, который нужно измерить. Есть диапазон, в пределах которого меняется освещенность (измеряется в люксах – световом потоке на единицу площади). Можно обоснованно ожидать значение показаний между 0 лк (для совершенно черного) и 130 000 лк на прямом солнечном свете. Если бы измерительный прибор был абсолютно точен, то можно было бы получить бесконечное число значений в данном диапазоне.

Компьютерная система никогда не может оперировать с бесконечным числом десятичных разрядов для аналогового значения, потому что объем памяти и производительность компьютера ограничены. Для преобразования аналоговых значений в цифровые с заданной точностью предназначены аналого-цифровые преобразователи (АЦП).

Разрешение АЦП – минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП – связано с его разрядностью. В случае единичного измерения без учета шумов разрешение напрямую определяется *разрядностью* АЦП.

Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе. Например, двоичный 8-разрядный АЦП способен выдать 256 дискретных значений (0–255), поскольку $2^8 = 256$.

Разрешение по напряжению равно разности напряжений, соответствующих максимальному и минимальному выходному коду, деленной на количество выходных дискретных значений.

Пример 1:

– диапазон входных значений = от 0 до 10 В;

– разрядность двоичного АЦП 12 бит: $2^{12} = 4096$ уровней квантования;

– разрешение двоичного АЦП по напряжению: $10 / 4096 = 0{,}00244$ В = 2,44 мВ.

Пример 2:

–диапазон входных значений: от −10 до +10 В;

–разрядность двоичного АЦП 14 бит: $2^{14} = 16384$ уровня квантования;

–разрешение двоичного АЦП по напряжению: (10 – (–10)) / 16384 = = 20/16384 = 0,00122 В = 1,22 мВ.

Основные типы аналоговых сигналов:

– вольтовые – данные передаются уровнем напряжения: 0–1 В, 1–5 В, 0–10 В и т. д.;

– токовые – данные передаются уровнем тока: 0–20 мА, 4–20 мА и т. д.

AI ПЛК и датчик, который к нему подключен, должны иметь одинаковый тип аналогового сигнала (рис. 19). Датчики могут иметь разные схемы подключения к ПЛК: 2-проводное (2-wire), 3-проводное (3-wire) или 4-проводное (4-wire). По какой именно схеме нужно подключать конкретный датчик, указано в его инструкции.

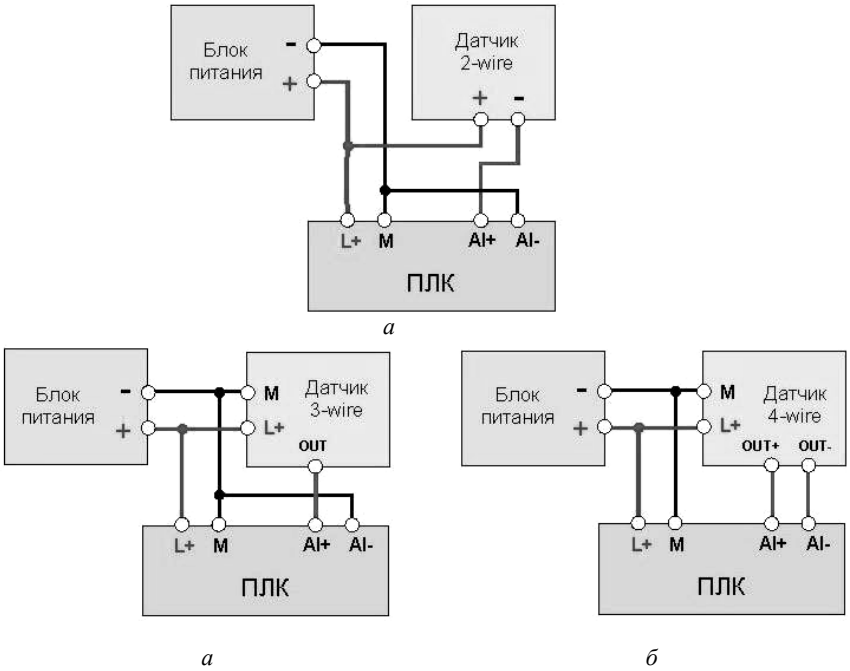


Рис. 19. Подключение датчиков к AI ПЛК Siemens:
а – двухпроводная схема; *б* – трехпроводная схема;
в – четырехпроводная схема

Попав в ПЛК, аналоговый сигнал с помощью аналого-цифрового преобразователя (АЦП, ADC) автоматически преобразуется в некое число. Обычно тип этого числа – целое беззнаковое (uint). Далее это число нужно в программе ПЛК перевести в реальные единицы измерения. Для этого во многих средах программирования ПЛК есть специальные функции. Например, функции NORM_X и SCALE_X в TIA PORTAL для ПЛК Siemens.

В тех ПЛК, где нет готовых функций преобразования, можно выполнить преобразование самому по формуле:

$$y = \frac{Val^{\mathrm{ADC}}\left(y_{\max} - y_{\min}\right)}{Val_{\max}^{\mathrm{ADC}}} + y_{\min},$$

где y – искомая величина;

$y_{\max}$ – верхняя граница измерения датчика;

$y_{\min}$ – нижняя граница измерения датчика;

Val^{ADC} – текущее значение AI;

$Val_{\max}^{\mathrm{ADC}}$ – максимальное количество значений, которое может выдать АЦП. В большинстве случаев зависит от его разрядности: 256 для 8-битных АЦП, 1024 для 10-битных, 4096 для 12-битных и т. д. Но может быть и другим, скажем, у Logo!8 при 10-битном АЦП максимум значений равен 1000.

Пример: датчик температуры с типом сигнала 0–20 мА и диапазоном измерения от –20 ºС до +220 ºС подключен к AI ПЛК с АЦП 12-бит. Чему равна температура, если текущее значение АЦП = 512?

$$t = \frac{512 \cdot \left(220\,^{\circ}\mathrm{C} - \left(-20\,^{\circ}\mathrm{C}\right)\right)}{4096} + \left(-20\,^{\circ}\mathrm{C}\right) = 10\,^{\circ}\mathrm{C}.$$

В общем случае измерительный тракт системы обработки аналоговых сигналов состоит из нескольких звеньев (рис. 20): входной сигнал, получаемый с датчика (или датчиков), поступает на усилитель через мультиплексор или напрямую. Главная задача усилителя в данной схеме – нормирование/усиление сигнала до оптимального для АЦП уровня. В свою очередь, АЦП производит оцифровку сигнала в соответствии с уровнем напряжения источника опорного

напряжения (ИОН), затем сигнал поступает на центральный процессор, где проходит цифровую обработку. Мультиплексоры в тракте служат для выборки одного из нескольких входных каналов.

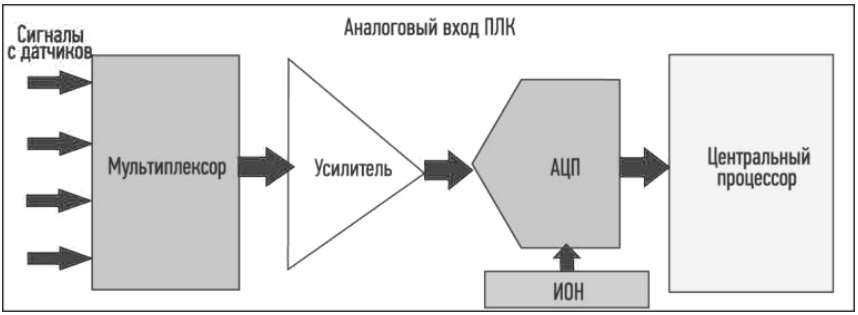


Рис. 20. Обобщенная структурная схема аналогового входа ПЛК

Уровень выходного сигнала с датчика может быть очень низким или очень высоким, что для максимизации динамического входного диапазона АЦП требует добавления усилителей или аттенюаторов соответственно. Эти предварительные каскады обычно реализуют на усилителях с программируемым коэффициентом усиления или на дискретных операционных усилителях и прецизионных резистивных делителях. АЦП и усилитель работают в тандеме, чтобы обеспечить наилучшее соотношение сигнал/шум (SNR) при заданных ограничениях по стоимости, размерам и потребляемой мощности.

Дискретные выходы. Дискретные выходы ПЛК необходимы для управления подключенными устройствами, например разного рода магнитными пускателями, лампочками, клапанами и прочим, посредством коммутации высокого или низкого сигналов. Дискретный выход представляет собой контакт, способный выдавать сигнал, являющийся с точки зрения программы логическим нулем или единицей. Такой сигнал способен замкнуть или разомкнуть управляющую или питающую цепь подключенного устройства, тем самым выполнив необходимый алгоритм работы.

Фактически дискретный выход – это переключатель, который замыкается по команде программируемого логического контроллера. Дискретный выход может иметь два состояния: разомкнутое (False)

и замкнутое (True). При замкнутом состоянии подсоединенное устройство будет включено, при разомкнутом – отключено (рис. 21).

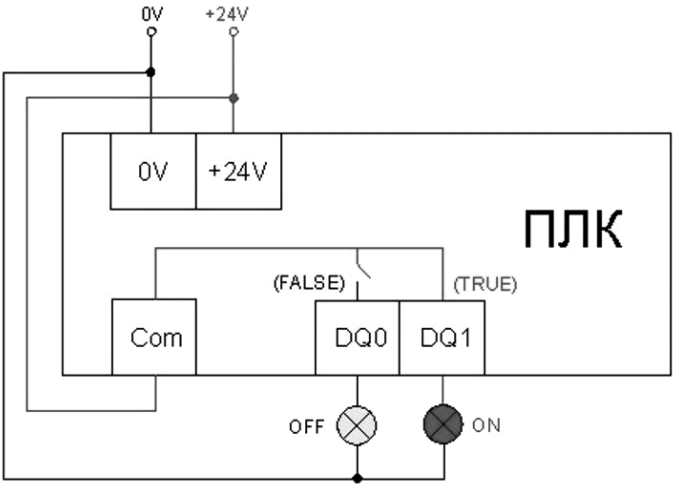


Рис. 21. Принцип работы дискретного выхода (DQ)

Дискретные выходы бывают транзисторные и релейные.

Дискретный выход типа «транзисторный ключ» – это электронный ключ, реализованный на полевом или биполярном транзисторе. Транзистор пропускает электрический ток, когда на его базу приходит управляющее напряжение.

Транзисторные дискретные выходы коммутируют только низковольтные сигналы до 24–30 В постоянного тока (VDC). С их помощью включают низковольтное оборудование, например лампы индикации. У транзисторных выходов большая скорость переключения, поэтому их также используют для широтно-импульсной модуляции (ШИМ, PWM), PID-регулирования и т. д. Недостаток транзисторных выходов – малая нагрузка, например, транзисторные выходы «ОВЕН ПЛК110» рассчитаны на напряжение до 24 В и ток до 0,4 А.

Если ПЛК имеет только транзисторные выходы и нужно коммутировать переменный ток, то устанавливается промежуточное реле (рис. 22).

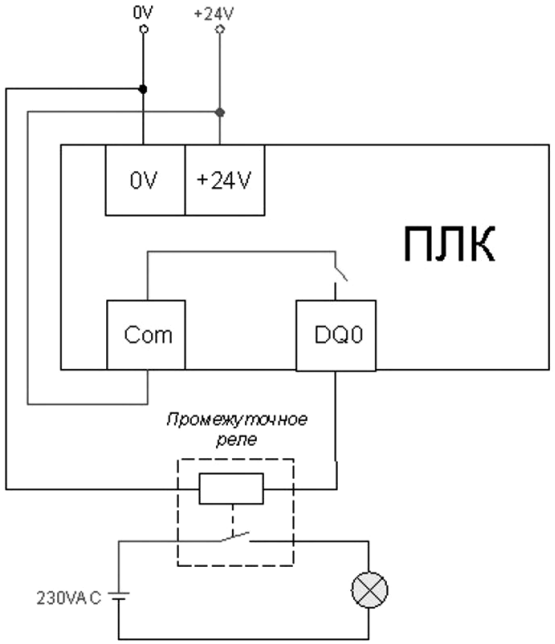


Рис. 22. Промежуточное реле к дискретным выходам транзисторного типа для коммутации переменного тока

Дискретный выход релейного типа представляет собой обычное электромагнитное реле, управляемое внутренней логикой контроллера (рис. 23). С помощью такого выхода можно управлять любой внешней силовой нагрузкой: электрической печью, клапаном, насосом, приводом и т. д. При этом необходимо учитывать мощность коммутируемого устройства (чтобы максимально возможный ток, протекающий в цепи, не превышал предельный ток, указанный для этого выхода в технических характеристиках).

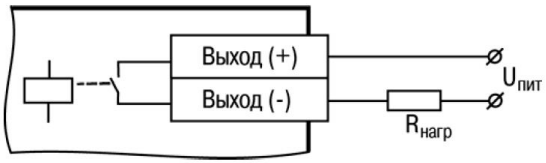


Рис. 23. Стандартная схема реализации дискретного выхода релейного типа

Преимущества релейных выходов (по сравнению с выходами транзисторного типа):

– выход уже готов к коммутации силовой (или слаботочной) нагрузки – нет необходимости в использовании внешних реле;

– не нужно устанавливать внешний источник питания выходов;

– релейные выходы независимы друг от друга и могут коммутировать разные по характеристикам цепи (например, один выход может включать лампу на 220 В, а другой – клапан на 12 В);

– не нагреваются в процессе работы.

Недостатки релейных выходов (по сравнению с выходами транзисторного типа):

– искрение контактов при коммутации индуктивной нагрузки;

– меньший ресурс по числу переключений (по времени работы);

– возможно залипание контактов реле при перегрузке;

– относительно большая задержка при срабатывании.

Аналоговые выходы ПЛК. Аналоговый выход (analog output, AO, AQ) используется в ПЛК для управления исполнительными устройствами и механизмами, имеющими аналоговый управляющий сигнал: направляющими аппаратами, регулирующими клапанами, частотными преобразователями и т. д.

Аналоговое управление применяется там, где объект может иметь более двух состояний. Отличие аналогового управления от дискретного рассмотрим на примере вентилятора (рис. 24). Чтобы просто включить или отключить вентилятор, достаточно дискретного управления. А если нужно во время работы вентилятора изменять скорость вращения его лопастей, то придется использовать аналоговое управление: чем выше уровень аналогового сигнала, тем больше будет скорость.

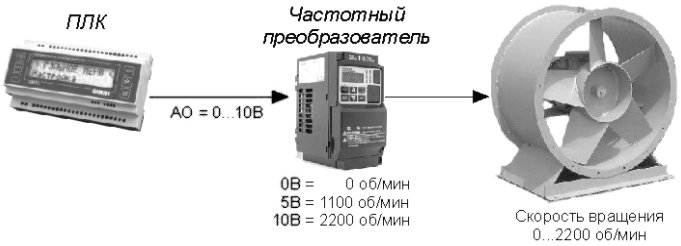
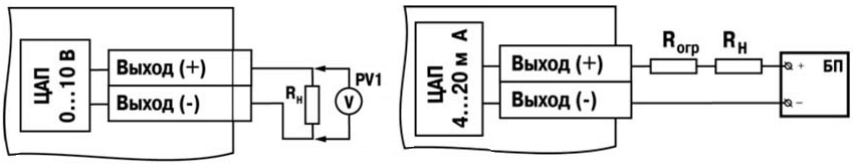


Рис. 24. Управление скоростью вращения вентилятора через АО ПЛК

Для использования АО нужно программно записать в регистр аналогового выхода число типа int, далее это число с помощью цифроаналогового преобразования (ЦАП, DAC) автоматически преобразуется в аналоговый электрический сигнал, который выводится из АО на объект управления (нагрузку).

Основные типы сигналов для аналогового управления: 0–10 В, 0–20 мА, 4–20 мА. Схемы подключения нагрузки к АО в разных моделях программируемых логических контроллеров могут отличаться (рис. 25).



а *б*

Рис. 25. Подключение аналоговой нагрузки к ПЛК ОВЕН:
а – подключение нагрузки к АО 0–10 В, $R_н > 2$ кОм ;
б – подключение нагрузки к АО 4–20 мА

АО ПЛК и объект управления должны быть настроены на один и тот же тип аналогового сигнала. В большинстве случаев у современных ПЛК аналоговые выходы универсальные и их можно программно настроить на различные типы сигналов.

ОСНОВЫ ОПЕРАЦИОННЫХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ И СИСТЕМ ПРОГРАММИРОВАНИЯ

Любая вычислительная система состоит, во-первых, из того, что в англоязычных странах принято называть словом hardware, или технического обеспечения: процессор, память, монитор, дисковые устройства и т. д., объединенные магистральным соединением, которое называется шиной.

Во-вторых, вычислительная система состоит из программного обеспечения – ПО. Все программное обеспечение принято делить на две части (рис. 26): прикладное и системное.

Рис. 26. Слои программного обеспечения компьютерной системы

К прикладному программному обеспечению, как правило, относятся разнообразные банковские и прочие бизнес-программы, игры, текстовые редакторы, вэб-браузеры для просмотра интернет-страниц и т. п. В случае с ПЛК к прикладному программному обеспечению относится программа пользователя, реализующая задуманный алгоритм управления объектом. То есть прикладное программное обеспечение относится к конкретной задаче управления

технологическим процессом и оборудованием на некотором определенном объекте.

Системное ПО ПЛК выполняет функции, во многом схожие с функциями операционной системы персонального компьютера. Операционная система (ОС) предназначена для управления всеми частями весьма сложной архитектуры компьютера.

Операционная система представляется пользователю виртуальной машиной, с которой проще иметь дело, чем непосредственно с оборудованием компьютера. Операционная система – это программа, постоянно работающая на компьютере и взаимодействующая со всеми прикладными программами.

Пользователь для решения своей конкретной задачи создает прикладную программу. Для этого сейчас используются только языки высокого уровня. Создание прикладной программы выполняется пользователем ПЛК при помощи специализированного пакета – системы программирования – на компьютере. Код программы размещается и сохраняется в энергонезависимой памяти ПЛК.

Такой архитектурный подход обеспечивает пользователю простоту создания программы. Пользователь не должен знать, как обращаться к конкретным физическим элементам входа-вывода, – это делают драйверы ОС. Компоненты ОС обеспечивают работу некоторых составных компонентов программы – таймеров, счетчиков. ОС организует сетевое взаимодействие с удаленными элементами системы, предоставляет пользователю информацию о текущем состоянии прикладной программы. Пользователю кажется, что все эти задачи и задачи прикладной программы решаются одновременно – ОС реализует многозадачный режим работы контроллера.

После включения питания ПЛК ОС выполняет самотестирование и настройку аппаратных ресурсов ПЛК, очистку оперативной памяти данных (ОЗУ), контроль целостности прикладной программы пользователя. Если прикладная программа сохранена в памяти (загружена) и нет запрета ее запуска, ПЛК переходит к выполнению ее действий.

Задачи управления любым объектом требуют непрерывного контроля его состояния. В любых цифровых микропроцессорных системах управления непрерывность контроля и управления достигается за счет применения дискретных алгоритмов, повторяющихся через некоторые достаточно малые промежутки времени. По этой

причине и в ПЛК действия прикладной программы выполняются циклически (см. рис. 2). Причем в каждом таком цикле – его называют рабочим циклом ПЛК – выполняются определение значений на входах, соответствующий расчет, выработка и выдача управляющих воздействий.

Такие ОС, использующиеся в ПЛК и компьютерах для решения задач автоматизации, получили название операционных систем реального времени – ОСРВ.

В начале цикла ПЛК производит чтение значений сигналов с физических входов. Считанные с портов ввода значения размещаются в ОЗУ, а именно в отдельном сегменте ОЗУ, называемом областью памяти входов – I-область (от англ. input – вход). Таким образом, в ОЗУ, в I-области, создается полная одномоментная копия значений входов. Выполнение первой фазы обеспечивается драйверами ОС.

Далее выполняется код прикладной программы пользователя, которая работает с копией значений входов, зафиксированных и размещенных в оперативной памяти, то есть со значениями, которые в процессе выполнения пользовательской программы в пределах одного рабочего цикла не изменяются. Это фундаментальный принцип функционирования абсолютного большинства промышленных ПЛК (ссылаясь на него, иногда подобные ПЛК выделяют в отдельный класс контроллеров сканирующего типа). Такой подход исключает неоднозначность алгоритма обработки данных в различных его ветвях и, в конечном счете, во многом определяет простоту создания программы. Кроме этого, пользователь не должен знать процедуру обращения к физическим входам, которых по типам может быть достаточно много, – за него чтение проводит ОС. Пользователь должен лишь знать, как в рамках прикладной программы обратиться в ОЗУ к переменным, соответствующим по величине сигналам на физических входах.

В процессе выполнения прикладной программы все рассчитанные значения, которые нужно вывести на физические выходы, фиксируются в оперативной памяти, а именно в отдельном сегменте ОЗУ, называемом областью памяти выходов – Q-область (от англ. quit – выход). Физические выходы ПЛК приводятся в соответствие с расчетными значениями после выполнения кода пользовательской программы. Эти действия выполняются ОС в отдельной фазе рабочего цикла.

Далее системное программное обеспечение переходит к фазе обслуживания аппаратных ресурсов – обеспечивается выполнение аппаратно-зависимых задач – ведение системных таймеров, часов реального времени, оперативное самотестирование, индикация состояний и пр.

Заканчивается цикл ПЛК фазой контроля времени цикла контроллера. Смысл выполнения этой фазы сводится к обеспечению постоянства данного цикла. Если не принимать специальных мер, то длительность цикла контроллера будет зависеть от времени выполнения прикладной программы и может все время меняться в зависимости от условий реализации алгоритма. Вместе с тем качественное решение некоторых задач управления, например реализация автоматического регулирования, будет зависеть от стабильности цикла получения входных значений и выдачи сигналов управления. Поэтому во многих контроллерах их создателями предусматривается для пользователя возможность фиксации длительности цикла в некоторых определенных пределах. Для обеспечения постоянства цикла контроллера в него вводится аппаратный таймер, задающий его длительность. Таймер следит, не превышает ли время выполнения прикладной программы заданного по длительности цикла. Если пользователь, исходя из знания того, как работает программа, правильно задал длительность цикла, а прикладная программа выполняется дольше, то это будет означать или зацикливание прикладной программы, или наличие аппаратных сбоев, приводящих к общему сбою («зависанию») контроллера. При этом таймер осуществляет сброс контроллера в исходное состояние, что позволяет или обеспечить дальнейшую работу контроллера вновь в нормальном режиме, или однозначно идентифицировать его устойчивое аварийное состояние. Кроме того, отсчеты этого аппаратного таймера обычно используются системным программным обеспечением для ведения часов реального времени и отсчета времени программными таймерами в прикладной программе.

Общая продолжительность рабочего цикла ПЛК (времени сканирования) в значительной степени определяется длительностью фазы выполнения кода пользовательской программы. Время, занимаемое прочими фазами рабочего цикла, как правило, существенно меньше и практически является величиной постоянной.

Обычно существует возможность устанавливать один из двух способов исполнения ПЛК рабочего цикла: циклический (новый цикл начинается сразу по окончании предыдущего цикла) или периодический (программист задает минимальную длительность цикла ПЛК). Во втором случае если предыдущий цикл завершился раньше заданного времени, то начало нового цикла задерживается до тех пор, пока не пройдет установленная длительность цикла.

Если пользователь задает длительность цикла контроллера, то делает это исходя из безусловного выполнения требования того, чтобы программа гарантированно (с некоторым запасом) выполнялась за время данного цикла. Пользователь при этом ориентируется на объем программы и быстродействие процессорного модуля ПЛК. Быстродействие процессорного модуля обычно оценивают по времени выполнения логических команд, поскольку они наиболее распространены при реализации алгоритмов управления. В технических характеристиках ПЛК может приводиться типовое время рабочего цикла.

Требования к длительности контроллерного цикла существенно зависят от области применения ПЛК. Если, исходя из условий решаемой задачи управления, пользователя не устроит время цикла выполнения созданной программы на имеющемся в его распоряжении контроллере, то это будет основанием для выбора контроллера с большей производительностью (быстродействием).

Время цикла не единственный фактор выбора контроллера определенной производительности. Работа контроллеров характеризуется еще временем реакции – временем с момента изменения состояния системы до момента выдачи соответствующей реакции. Очевидно, для ПЛК время реакции зависит от распределения моментов возникновения события и начала фазы чтения входов. Если изменение значений входов (событие) произошло непосредственно перед фазой чтения входов, то время реакции будет наименьшим и фактически равным времени цикла контроллера (рис. 27).

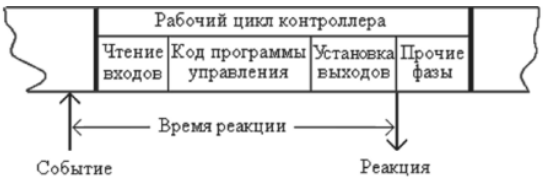


Рис. 27. Рабочий цикл ПЛК и определение времени реакции на событие

Худший случай будет наблюдаться, если изменение значений входов происходит сразу после фазы чтения входов. Тогда время реакции будет наибольшим, равным почти удвоенному времени цикла контроллера (цикла сканирования).

При оценке общего времени реакции проектируемой системы управления, помимо времени реакции ПЛК, существенное значение имеет время реакции датчиков и исполнительных механизмов, которое также необходимо учитывать.

Современные ПЛК имеют типовое значение времени рабочего цикла, измеряемое единицами миллисекунд и менее. Время реакции большинства исполнительных устройств значительно выше, постоянные времени управляемых объектов – секунды, и поэтому с реальными ограничениями возможности использования ПЛК по времени приходится сталкиваться редко. Качество регулирования в таких условиях устраивает абсолютное большинство пользователей. Для прикладного программиста все эти детали неважны. Для него при создании программы важно лишь то, что значения входов обновляются автоматически и исключительно в начале каждого рабочего цикла.

ГРАФИЧЕСКИЕ И ТЕКСТОВЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ ПРОМЫШЛЕННЫХ КОНТРОЛЛЕРОВ МЕЖДУНАРОДНЫХ СТАНДАРТОВ

С целью унификации и стандартизации процесса создания прикладных программ для ПЛК был разработан международный стандарт IEC 61131-3, определяющий требования к высокоуровневым языкам программирования, которые ориентированы в первую очередь на инженеров-технологов, не имеющих специальных навыков в области программирования на традиционных языках. Поддержка языков данного стандарта является необходимым компонентом любого современного ПЛК.

Стандарт IEC 61131-3 устанавливает пять языков программирования ПЛК (три графических и два текстовых) со следующими названиями:

– LD;
– FBD;
– SFC;
– IL;
– ST.

Из них языки IL и ST считают текстовыми языками, так как они используют текст, числа и пунктуацию как естественный код для создания программы. В то же время языки LD, FBD и SFC называют графическими, так как внешне они похожи на релейно-контактные схемы, схемы транзисторно-транзисторной логики и блок-схемы алгоритмов.

Стандарт обеспечивает совместное использование всех пяти языков, поэтому для каждого фрагмента задачи может быть выбран любой наиболее удобный язык. Программа, написанная для одного контроллера, может быть перенесена на любой контроллер, совместимый со стандартом IEC 61131-3.

Язык LD

Язык релейно-контактной символики, или диаграмм LD (Ladder Diagram), представляет собой графический язык разработки и был создан для инженеров, имеющих опыт разработки релейно-контактных схем автоматики.

Рассмотрим простой пример (рис. 28), в котором необходимо включить электродвигатель, при нажатии на выключатель Вход (X1). В результате получим устройство, которое можно условно разделить на три основные части:

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

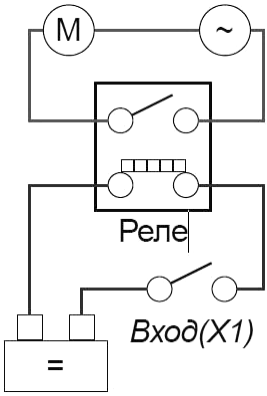


Рис. 28. Схема включения электродвигателя М через контакт реле

Всякий раз, когда выключатель Вход(X1) соединяет контакты (включается), мы закрываем цепь и ток, создавая электромагнитное поле в реле, переключает контакты реле, то есть запускает электродвигатель.

В данном примере использовано типичное индустриальное реле постоянного тока для управления включением/выключением устройств. Пока Вход(X1) открыт, ток не может течь сквозь катушку реле и электродвигатель не работает. Но как только Вход(X1) закрыт, ток создает электромагнитное поле, которое заставляет контакты реле соединяться. В результате ток, протекающий через контакт реле, заставляет вращаться электродвигатель.

Можно заменить реле контроллером (рис. 29). 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.

Сначала необходимо понять, каким образом можно объяснить ПЛК, какую задачу он должен выполнить.

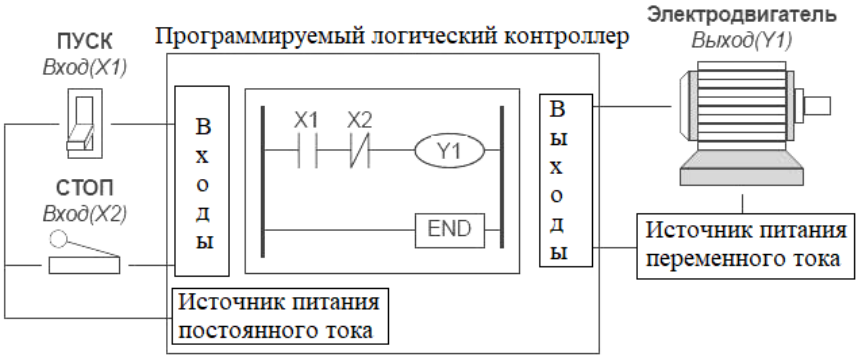


Рис. 29. Схема иллюстрации физической связи кнопок «ПУСК» и «СТОП» с входами контроллера, электродвигателя – с выходом контроллера; алгоритм управления определяется программой, записанной в память контроллера

Шаг первый: переопределить все составляющие оборудования, которые используются, 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 переменными «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 программистом.

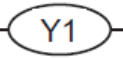
Шаг второй: заменить источник питания. Для языка релейно-контактных схем этим символом будут являться две параллельные прямые с левой и правой стороны диаграммы. Можно считать, что левая линия является «+», а правая линия «–».

Шаг третий: присвоить символы входам. В данном примере есть два входа:

• Вход(X1)  – нормально открытый контакт;

• Вход(X2)  – нормально закрытый контакт.

Шаг четвертый – последний: присваиваем символ выходам.

• Выход(Y1)  – символ катушки Y1.

В результате мы получили программу, которая может быть выполнена ПЛК. Инструкции языка релейно-контактной символики внешне похожи на условные графические изображения элементов релейно-контактной автоматики американских стандартов. Но нужно понимать, что в памяти контроллера нет никаких контактов и катушек, а программа работает с ячейками памяти, в которых записываются лог 1 и лог 0.

К недостаткам данного языка можно отнести то, что по мере увеличения количества «реле» в схеме она становится сложнее для интерпретации, анализа и откладки. Еще один недостаток языка LD заключается в следующем: язык, построенный по аналогии с релейными схемами, может быть эффективно использован только для описания процессов, имеющих дискретный (двоичный) характер; для обработки «непрерывных» процессов (с множеством аналоговых переменных) такой подход теряет смысл.

Язык FBD

Одним из популярных языков программирования ПЛК является графический язык функциональных блоковых диаграмм – Function Block Diagram (FBD).

Написанная на данном языке программа для контроллера состоит из некоторого списка цепей, которые одна за другой выполняются сверху – вниз. Кроме того, здесь имеется возможность присвоения отдельным цепям меток, в этом случае станет доступно использование инструкций перехода на метку, чтобы изменять последовательность исполнения цепей и создавать условия и циклы. Таким образом, программа, написанная на графическом языке FBD, представляет собой набор связанных друг с другом функциональных блоков, выходы и входы которых соединены линиями связи (рис. 30). Линии связи отражают определенные программные переменные, через которые происходит обмен данными от блока к блоку.

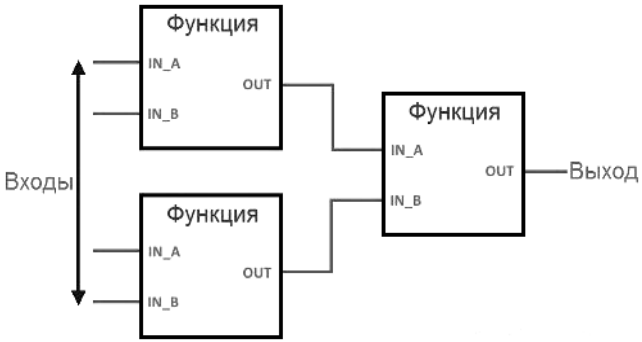


Рис. 30. Вид программы на языке FBD

Отдельный блок несет на себе конкретную функцию (логическое «И», «НЕ», счетчик и т. д.), при этом один блок может иметь несколько выходов и входов. Изначально значения переменных задаются константами или со специальных входов, а выходы их связываются дальше с другими переменными программы или с выходами ПЛК.

В процессе программирования на языке FBD применяются как стандартные блоки из библиотек, так и блоки, написанные на FBD или на иных языках стандарта МЭК 61131-3. Блок представляет собой элемент программы, своего рода подпрограмму, функциональный блок или функцию (логическое «НЕ», «ИЛИ», «И», таймер, счетчик, триггер, математическая операция, обработка аналогового сигнала и т. д.).

Из таких блоков графически составляются выражения, образующие цепи: к выходу одного блока присоединяется следующий блок, далее – еще блок, и так образуются цепи. По ходу цепи порядок выполнения блоков соответствует порядку их соединения, а результат выполнения цепи либо подается на выход ПЛК, либо записывается в какую-то внутреннюю переменную.

В качестве примера на рис. 31 представлен фрагмент программы управление насосом, подающим воду в бак.

Как LD, так и FBD, используют для некоторых блоковых команд понятие «поток сигнала» (EN и ENO). Некоторые команды (например, арифметические операции и команды пересылки) отображают параметры для EN и ENO. Эти параметры относятся к потоку сигнала и определяют, выполняется ли команда в этом цикле.

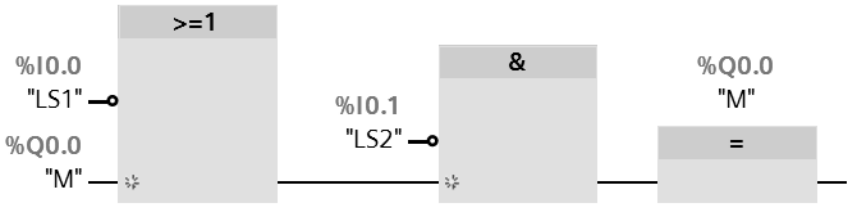


Рис. 31. Программа 2-хпозиционного регулирования уровня воды в баке, написанная на языке FBD

EN (Enable In = разблокировать вход) является булевым входом для блоков в LD и FBD. Поток сигнала (EN = 1) должен присутствовать на этом входе, чтобы блоковая команда выполнялась. Если вход EN блока LD присоединен непосредственно к левой шине электропитания, то блок всегда будет исполняться.

ENO (Enable Out = разблокировать выход) является булевым выходом для блоков в LD и FBD. Если у блока имеется поток сигнала на входе EN и блок выполняет свою функцию без ошибок, то выход ENO передает поток сигнала (ENO = 1) следующему элементу. Если в исполнении блоковой команды обнаружена ошибка, то поток сигнала прерывается (ENO = 0) у блоковой команды, которая вызвала ошибку.

Выполнение FBD-цепей идет слева направо, сверху вниз. Блоки, расположенные левее, выполняются раньше. Блок начинает вычисляться только после вычисления значений всех его входов. Дальнейшие вычисления не будут продолжены до вычисления значений на всех выходах. Другими словами, значения на всех выходах графического блока появляются одновременно. Вычисление цепи считается законченным только после вычисления значений на выходах всех входящих в нее элементов.

Язык последовательных функциональных схем SFC

Язык последовательных функциональных схем SFC (Sequential Function Chart), использующийся совместно с другими языками (обычно с ST и IL), является графическим языком, в котором программа описывается в виде схематической последовательности шагов, объединенных переходами. Пример программы на языке SFC приведен на рис. 32.

Использование компонентов языка SFC позволяет воссоздавать диаграмму процесса. Каждая SFC диаграмма обязательно начинается с Шага Инициализации (Start), который отображается в виде прямоугольника в прямоугольнике.

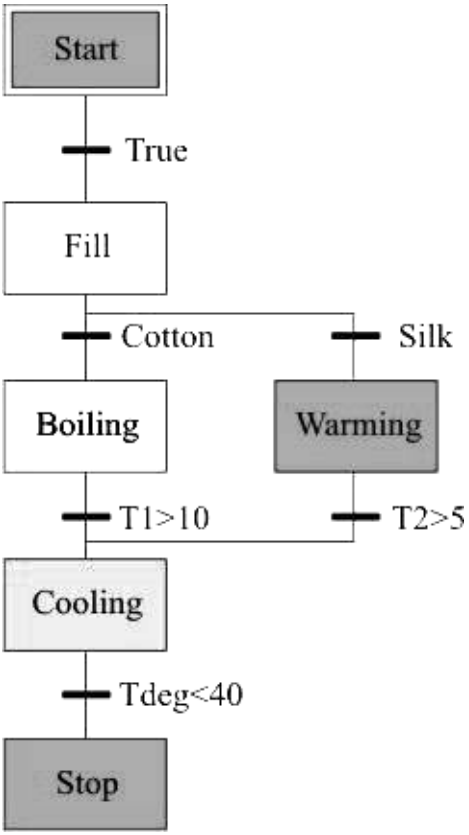


Рис. 32. Вид программы на языке SFC на примере программы управления работой стиральной машины

Шаги последовательности располагаются вертикально сверху вниз. Шаг описывает одну из стадий технологического процесса. На каждом шаге выполняется определенный перечень действий (операций). При этом для описания самой операции используются другие языки программирования, такие как IL или ST. В каждый

момент времени шаг является либо активным, либо неактивным. В реальных проектах шаги имеют осмысленные названия, например, Fill (наполнение), Boiling (кипячение) и т. д.

С каждым шагом может быть связано одно или несколько действий. Если шаг не содержит действий, то программа просто ожидает выполнения условия перехода к следующему шагу. Само по себе действие может быть программно закодировано на любом из остальных языков международного стандарта IEC 61131-3: LD, FBD, SCL или IL.

После того, как шаг выполнен, управление передается следующему за ним шагу. Между шагами находятся так называемые переходы. Переход описывается условием, при выполнении которого активным становится следующий шаг (шаги), а текущий шаг прекращает активность. Переходы графически соединяют нижнюю часть предыдущего шага и верхнюю часть шага, следующего за ним. Переход между шагами может быть условным и безусловным. Безусловный переход происходит всегда после полного выполнения всех операций на данном шаге. Так, на рис. 32 примерами безусловных переходов являются переход от шага Start («Старт») к шагу Fill («Наполнение»), а также следующий переход – от шага Fill к одному из шагов Boiling («Кипячение») или Warming («Нагрев»), который выполняется после завершения наполнения резервуара стиральной машины водой до требуемого уровня. Условный переход требует выполнение определенного логического условия для передачи управления на следующий шаг; пока это условие не выполнено программа будет оставаться внутри текущего шага, даже если все операции внутри шага уже выполнены. Так, на рис. 32 примером условного перехода является переход от одного из шагов Boiling («Кипячение») или Warming («Нагрев») к следующему шагу Cooling («Охлаждение»), условием выполнения которого является время, которое должно пройти от начала выполнения кипячения (10 минут) или нагрева (5 минут). Также условным переходом является переход от шага охлаждения Cooling к шагу окончания работы Stop: в данном случае требуемым условием является снижение температуры воды в резервуаре стиральной машины до 40 ºС.

С помощью переходов можно осуществлять разделение и слияние ветвей последовательности, организовать параллельную обработку

нескольких ветвей или заставить одну выполненную ветвь ждать завершения другой.

Расходимость (дивергенция) – это множественное соединение в направлении от одного шага к нескольким переходам. Примером дивергенции в программе на рис. 32 является разветвленный переход от шага Fill к одному из шагов Boiling («Кипячение») или Warming («Нагрев»): в зависимости от того, какой режим выставлен на соответствующем переключателе режимов стирки – Cotton («Хлопок») или Silk («Шелк»») – выполняется переход только к одному из шагов Boiling или Warming.

Сходимость (конвергенция) – это множественное соединение, направленное от нескольких переходов к одному и тому же шагу. Она обычно используется для группировки ветвей SFC – программы, которые берут начало из одинарной дивергенции. Примером конвергенции на SFC диаграмме на рис. 32 является переход от одного из шагов Boiling («Кипячение») или Warming («Нагрев») к следующему шагу Cooling («Охлаждение»).

Альтернативная расходимость представляет собой связь одного шага с несколькими переходами. При этом выполняется только один из переходов – Cotton (хлопок) или Silk (шелк) на рис. 32, так как их условия являются взаимоисключающими. Условия, связанные с различными переходами при расходимости, должны быть заданы пользователем так, чтобы гарантировать, что во время выполнения программы активируется одна конкретная ветвь. Сходимость для альтернативной расходимости представляет собой связь нескольких переходов с одним шагом. Такая сходимость обычно используется для SFC-ветвей, полученных при одиночной расходимости.

Альтернативная сходимость и расходимость изображаются на SFC-схеме с помощью горизонтальной линии (как на рис 32). Частным случаем альтернативной расходимости является ветвь с циклом, которая содержит переходы на предыдущие (расположенные выше по схеме) шаги.

Параллельная расходимость представляет собой связь одного перехода с несколькими шагами. После выполнения условия перехода первые шаги всех параллельных ветвей становятся активными. Такой подход используется для описания операций процесса управления, которые выполняются одновременно. Сходимость для

параллельной расходимости представляет собой связь нескольких шагов с одним переходом. Такая сходимость обычно используется для SFC-ветвей, полученных при двойной расходимости. Параллельная сходимость выполняется в том случае, если все предшествующие ей шаги являются активными, а условие перехода – истинным. После схождения ветвей все предшествующие шаги деактивируются, и активным становится шаг, расположенный после перехода.

Параллельная сходимость и расходимость изображаются на SFC-схеме с помощью двойной горизонтальной линии, как это показано на рис. 33.

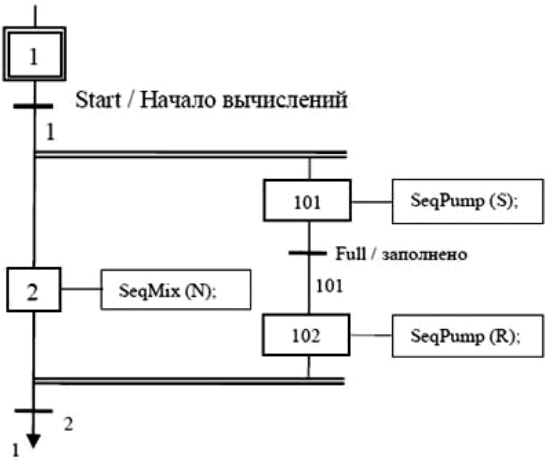


Рис. 33. Пример программы на языке SFC
с параллельной расходимостью и сходимостью

Язык IL (Instruction List, список команд)

IL – это текстовый язык программирования ПЛК, который по синтаксису напоминает ассемблер. IL был одним из первых языков программирования ПЛК. Язык IL считают языком низкого уровня – это означает, что IL приближен к машинному языку – двоичному коду, который центральный процессор компьютера выполняет непосредственно.

Программы, написанные на языке IL, – это последовательности команд. Команды состоят из операторов и операндов, которые

могут быть переменными, константами или метками. Кроме того, программы могут дополняться комментариями.

Инструкции языка IL выполняют операции с содержимым аккумулятора. Аккумулятор IL – универсальный регистр (регистр – ячейка памяти в ЦПУ), способная сохранять значения переменных любого типа. Программа на IL выполняется подряд, сверху вниз. Для изменения порядка выполнения операций применяется переход на метку. Переход выполняется как вверх, так и вниз.

Стандартные операторы IL:

LD – загрузить значение операнда в аккумулятор;

ST – присвоить значение аккумулятора операнду;

S – если аккумулятор True, установить логический операнд (True);

R – если аккумулятор True, сбросить логический операнд (False);

AND, OR, XOR, NOT – поразрядные логические операторы;

ADD, SUB, MUL, DIV, MOD – арифметические операторы;

GT, GE, EQ, NE, LE, LT – операторы сравнения;

JMP, CAL, RET – операторы перехода к метке, вызова функционального блока и выхода из функционального блока, соответственно.

Пример программы на языке IL приведен на рис. 34. Первая команда – LD x – загружает значение переменной x ОЗУ в регистр-аккумулятор. Следующая команда – MUL A – выполняет умножение содержимого регистра-аккумулятора на значение переменной A, при этом результат умножения сохраняется в регистре-аккумуляторе. Команда ADD B выполняет суммирование значения в регистре-аккумуляторе со значением переменной B, результат суммирования также сохраняется в регистре-аккумуляторе. Последняя команда – ST Y – копирует значение из регистра-аккумулятора в переменную Y в ОЗУ.

```
LD x
MUL A
ADD B
ST Y
```

Рис. 34. Пример программы на языке IL – код, реализующий линейное преобразование Y=A*x+B

Язык IL имеет недостатки, которые присущи другим низкоуровневым языкам программирования: сложность и высокую трудоемкость программирования, трудность модификации написанных на нем программ, малую степень «видимого» соответствия исходного текста программы и решаемой задачи.

Ввиду своей ненаглядности, IL практически не используется для программирования комплексных алгоритмов автоматизированного управления, но часто применяется для кодирования отдельных функциональных блоков, из которых впоследствии складываются схемы FBD или CFC. При этом IL позволяет достичь высокой оптимальности кода: программные блоки, написанные на IL, имеют высокую скорость исполнения и наименее требовательны к ресурсам контроллера.

Язык ST

ST (Structured Text) – это текстовый язык высокого уровня общего назначения, по синтаксису схожий с языком Pascal. Удобен для написания больших программ и работы с аналоговыми сигналами и числами с плавающей точкой. Преимуществом языка является возможность создания сложных математических и разветвленных алгоритмов.

Ниже приведен пример программы на языке ST – процедура вычисления максимума из массива. В начале программы следует блок VAR_CONSTANT … END_VAR, в котором определяются используемые в программе постоянные значения – константы. В данном примере такой константой является Array_Sz, значение которой равно 4, и которая определяет длину, то есть число элементов, в массиве arr. Сам массив arr, вместе с другими необходимыми переменными iter (номер итерации в цикле) и fnd_max (найденное максимальное значение массива), определен в следующем блоке VAR … END_VAR. Далее идет команда организации цикла FOR … END_FOR, выполняющая проход по всем элементам массива, каждое значение которого при этом сравнивается с максимальным, и если оно превосходит найденное до этого максимальное значение fnd_max, то в fnd_max сохраняется новое максимальное значение.

Пример:

```
VAR_CONSTANT
Array_Sz: BYTE := 4;
```

```
END_VAR
VAR
Iter: BYTE;
arr: ARRAY [1..Array_Sz] of real:=3.2, 4.2,1.4, 7.8;
fnd_max:REAL := -1.2E38;
END_VAR
FOR Iter := 1 TO Array_Sz DO
fnd_max := MAX(fnd_max, arr[Iter]);
END_FOR
```

Из всех языков программирования IEC61131 язык ST получает самое быстрое распространение.

Этот язык лучше всего подходит для сложного программирования ПЛК, такого как, например, управление процессами в производстве пластмасс или химической промышленности. Тригонометрические функции, математические вычисления и анализ данных на этом языке можно реализовать легче, чем на языке релейно-контактных схем или языке списка инструкций. Циклы выбора и указатели (переменные, используемые для косвенной адресации) позволяют реализацию более компактных программ, чем могут быть созданы на языке релейно-контактных схем. Для написания программы на языке ST используется удобный текстовый редактор, который облегчает ввод комментариев в программу, а также позволяет использовать знаки абзацев и пробелы для выделения связанных участков кода. Это облегчает задачу структурирования комплексных программ. Текстовый, неграфический характер языка ST, похожего на язык IL, позволяет создавать программы, которые работают гораздо быстрее, чем программы, созданные на языке LD. Окончательным преимуществом является то, что многие студенты инженерных специальностей лучше владеют компьютерными языками, чем основами электротехники, и поэтому лучше владеют языком ST, чем LD.

ЯЗЫК СТРУКТУРИРОВАННЫХ ТЕКСТОВ (ST). СТРУКТУРА ПРОГРАММЫ. СТАНДАРТНЫЕ БИБЛИОТЕКИ. ОСНОВНЫЕ ОПЕРАТОРЫ. ТИПЫ ДАННЫХ

Язык программирования контроллеров ST относится к классу текстовых языков высокого уровня и является в настоящее время самым распространенным языком программирования промышленных контроллеров. Этот язык уходит корнями в такие известные языки программирования, как Pascal и C. Язык ST удобен для написания больших программ и работы с аналоговыми сигналами и числами с плавающей точкой.

Структурированный язык управления SCL (Structured Control Language) также является текстовым языком программирования и представляет собой язык структурированного текста ST, адаптированный для программирования контроллеров линейки моделей Simatic производства фирмы Siemens.

Синтаксис языка ST

Для начала рассмотрим простой пример: запрограммируем функцию ограничителя Delimiter, которая должна ограничивать входные величины, адаптируя их к заданному диапазону значений (между верхним и нижним пределами); кроме того, в примере мы запрограммируем вызов этой функции в организационном блоке. Ниже приведен текст программы:

```
FUNCTION Delimiter : INT
VAR_INPUT
MAX : INT; //максимальное значение
IN : INT; //входное значение
MIN : INT; //минимальное значение
END_VAR
BEGIN
IF IN>MAX THEN Delimiter:=MAX; //адаптация к верхнему значению
ELSIF IN<MIN THEN Delimiter:=MIN;//адаптация к нижнему значению
ELSE Delimiter:=IN; //допустимое входное значение
```

```
    END_IF;
    END_FUNCTION
    ORGANIZATION_BLOCK Mainjprogram
    VAR_TEMP SINFO : ARRAY [1..20] OF BYTE;
    END_VAR
    BEGIN
    Result:=Delimiter (MAX:=Maximum, IN:=INPUT_VALUE,
MIN:=Minimum);
    END_ORGANIZATION_BLOCK
```

Пример программы начинается с объявления типа блока ограничителя Delimiter (функция FC) и типа для значения функции (INT). Далее следует описание параметров блока: параметры MAX (максимальное значение, верхняя граница), IN (величина входного сигнала) и MAX (максимальное значение, нижняя граница) объявляются входными параметрами (INPUT), относящимися к типу данных INT. Сама программа следует за разделом объявления переменных. В соответствии с программой, если входная величина IN больше, чем заданная максимальная величина, то функция принимает значение, равное максимальной величине. Если это не так, и входная величина IN меньше, чем заданная минимальная величина, то функция принимает значение, равное минимальной величине. Если оба рассмотренных случая не дали положительного результата, то функция принимает значение, равное входной величине IN.

Теперь мы запрограммируем вызов этой функции в организационном блоке Main Program. При вызове в организационном блоке Main Program функции ограничителя Delimiter ее значение присваивается глобальной переменной Result («Результат»); теперь эта переменная будет содержать в себе значение Input_value, ограниченное предельными значениями Maximum и Minimum.

Как видно из приведенного примера, ST-программа – это список ST-выражений. Выражения обеспечивают получение определенных значений. Выражение может включать в себя один адрес данных (одну переменную) или несколько адресов данных (несколько переменных), которые объединяются с помощью операторов. Пример:

```
a + b;
```

здесь a и b – это адреса данных (то есть, переменные), «+» – это оператор.

Имена, используемые в исходном коде (имена переменных, константы, ключевые слова), разделены неактивными разделителями (пробелами, символами окончания строки и табуляции) или активными разделителями, которые имеют определенное значение (например, разделитель «>» означает сравнение «больше чем»). В текст могут быть введены комментарии. Комментарий должен начинаться с символа «//» и заканчиваться в конце строки. Каждый оператор заканчивается точкой с запятой («;»). Основные операторы языка ST:

– оператор присваивания (x := 5);
– вызов подпрограммы или функции;
– вызов функционального блока;
– операторы выбора (IF, THEN, ELSE, CASE);
– итеративные операторы (FOR, WHILE, REPEAT);
– управляющие операторы (RETURN, EXIT);
– специальные операторы для связи с такими языками как SFC.

Скобки используются для того, чтобы отделить подчасти выражения и явно определить приоритетность операций.

Типы данных

Тип данных переменной определяет род информации, диапазон представления и множество допустимых операций. Все рассмотренные в прошлой главе стандартные языки программирования ПЛК – LD, FBD, SFC, IL и ST (SCL) – используют идеологию строгой проверки типов данных. Это означает, что любую переменную можно использовать только после ее объявления.

Типы данных МЭК разделяются на две категории – элементарные и составные. Элементарные, или базовые, типы являются основой для построения составных типов.

Элементарные типы данных:

– целые (целочисленные);
– с плавающей точкой;
– типы времени;
– строки;
– битовые.

Целочисленные переменные отличаются различным диапазоном сохраняемых данных и, естественно, различными требованиями к памяти. Подробно данные характеристики представлены в табл. 1.

Таблица 1

Целочисленные типы данных

Тип данных	Подробное название	Диапазон возможных значений
SINT	Short Integer	–128…127
INT	Integer	–32 768…32 767
DINT	Double Integer	$-2^{31}\ldots2^{31}-1$
LINT	Long Integer	$-2^{63}\ldots2^{63}-1$
USINT	Unsigned Short Integer	0…255
UINT	Unsigned Integer	$0\ldots2^{16}-1$
LDINT	Long Double Integer	$0\ldots2^{32}-1$
ULINT	Unsigned Long Integer	$0\ldots2^{64}-1$

Нижний предел диапазона целых чисел без знака равен 0, верхний предел определяется как $(2^n - 1)$, где n – число разрядов числа. Для чисел со знаком нижний предел составляет -2^n, верхний предел составляет $(2^n - 1)$.

Наименования целых типов данных образуются с применением префиксов, выражающих отношение разрядности представления данных к 16-разрядным словам: S – short, «короткое» (16×1/2), D – double, «двойное» (16×2), L – long, «длинное» (16×4). Префикс U (unsigned) указывает на представление целых без знака.

При начальной инициализации целочисленные переменные получают нулевые значения. Если необходимо задать другие начальные значения, это можно сделать при объявлении переменной.

Применение широкого спектра типов целочисленных переменных позволяет программисту оптимизировать код программы – добиться меньшего времени ее выполнения и (или) меньшего объема требуемого ОЗУ при соблюдении требований по точности вычислений. Скорость вычислений зависит от того, как микропроцессор оперирует переменными того или иного типа. Так, 16-разрядный процессор выполняет сложение двух 16-разрядных значений одной командой. Сложение же двух значений 32-разрядных переменных будет выполняться уже подпрограммой из нескольких команд. В общем случае меньшие по диапазону представляемых значений типы переменных требуют меньше памяти,

меньше кода и вычисления с их участием выполняются значительно быстрее.

Переменные типов BYTE, WORD, DWORD и LWORD определяются стандартом как битовые строки. Они представляют собой последовательности из 8, 16 и 32 бит соответственно. Помимо обращения с такими переменными как с единым целым, их можно использовать побитно.

Таблица 2

Битовые типы данных

Тип данных	Подробное название	Диапазон возможных значений
BOOL	Boolean	1 bit
BYTE	Byte	8 bits
WORD	Word	16 bits
DWORD	Double Word	32 bits
LWORD	Long Word	64 bits

Логические переменные объявляются ключевым словом BOOL. Они могут принимать только значение логического нуля или логической единицы. При начальной инициализации по умолчанию значения этих переменных принимаются равными логическому нулю.

Действительные типы – REAL и LREAL – представляют вещественные, то есть как целые, так и не целые переменные. Переменные действительных типов представляются как числа с плавающей запятой, то есть с помощью мантиссы и порядка. Переменные типа REAL представляют действительные числа в диапазоне $\pm 10^{\pm 38}$. Из 32 бит, занимаемых числом, мантисса занимает 23 бит. В результате точность представления будет приблизительно на уровне 6–7-десятичных разрядов после запятой. Длинный формат LREAL при представлении действительных чисел занимает 64 бита. Число содержит 52-битовую мантиссу. Точность представления чисел при этом будет на уровне 15–16-десятичных разрядов после запятой. Диапазон чисел длинного действительного $\pm 10^{\pm 307}$. Числа с плавающей запятой записываются в формате с точкой (14.0, –120.2, 0.33) или в экспоненциальной форме (–1.2E10 или 3.1e7).

Таблица 3

Типы данных с плавающей точкой (вещественные)

Тип данных	Подробное название	Диапазон возможных значений
REAL	Real Numbers	$\pm 10^{\pm 38}$
LREAL	Long Real Numbers	$\pm 10^{\pm 308}$

Интервал времени отражается переменными типа TIME. Числа, выражающие временной интервал, должны начинаться с ключевого слова TIME# или в сокращенной форме T#. В общем случае представление времени составляется из полей дней (d), часов (h), минут (m), секунд (s) и миллисекунд (ms). Порядок представления должен быть именно такой, хотя ненужные элементы можно опускать. Нужное значение интервала времени можно выразить через любые приведенные единицы.

Время суток и дата. Типы переменных, выражающие время дня или дату, указаны в табл. 4. Дата записывается в формате «год-месяц-число». Время записывается в формате «часы:минуты:секунды.сотые». Дата определяется ключевым словом DATE# (сокращенно D#), время дня – TIME OF DAY# (сокращенно TOD#), дата и время – DATE AND TIME# (сокращенно DT#).

Таблица 4

Типы данных для представления значений времени

Тип данных	Подробное название	Диапазон возможных значений
TIME	Duration of time after an event	T#10d4h38m57s12ms or TIME#10d4h38m
DATE	Calendar date	D#1989-05-22 or DATE#1989-05-22
TIME_OF_DAY	Time of day	TOD#14:32:07 or TIME_OF_DAY#14:32:07.77
DATE_AND_TIME	Date and time of day	DT#1989-06-15-13:56:14.77 or DATE_AND_TIME#1989-06-15-13:56:14.77

Переменные типа время суток (TIME OF DAY), в отличие от временного интервала, ограничены максимальным значением в 24 часа.

Время суток и дата – расширенный формат: DTL (date and time long). Переменная такого типа имеет длину 12 байт и хранит информацию о времени и дате в виде структуры из табл. 5.

Таблица 5

Структура переменной типа DTL

Номер байта	Содержимое байта	Тип данного	Диапазон значений
0	год	UINT	1970–2200
1			
2	месяц	USINT	1–12
3	день	USINT	1–31
4	день недели	USINT	1 (вс.)–7 (сб.) день недели не указывается явно при записи переменной в программе
5	час	USINT	0–23
6	минута	USINT	0–59
7	секунда	USINT	0–59
8	наносекунды	UDINT	0–999999999
9			
10			
11			

Диапазон возможных значений – от DTL#1970-01-01-00:00:00.0 до DTL#2200-12-31-23:59:59.999999999. Пример записи в программе: DTL#2008-12-31-20:15:45.250.

Строки (табл. 6). Этот тип данных объявляется ключевым словом STRING и определяет переменные, содержащие текстовую информацию. Размер строки задается при объявлении.

Таблица 6

Строки

Тип данных	Подробное название	Пример
STRING	Character String	‘My string’

Преобразование типов. Присваивать значение одной переменной другой можно, только если они обе одного типа. Если происходит присваивание значения переменной одного типа переменной другого типа, то происходит преобразование типов, меняющее физическое представление значения переменной в памяти данных, но не изменяющее само значение.

Допускается присваивание значения переменной совместимого типа, имеющей более широкое множество допустимых значений. В этом случае происходит неявное преобразование типа без потерь. Неявные преобразования типов данных с потерями запрещены. Так, например, логическую переменную, способную принимать только два значения (логические 0 и 1), можно присвоить переменной типа SINT (–128...+127), но не наоборот. При трансляции программы все подобные попытки отслеживаются и считаются грубыми ошибками. Если же это действительно необходимо, то выполнить присваивание с потерями возможно, но только при помощи специальных операторов.

Операторы преобразования выполняют также и более сложные операции, например преобразование числа или календарной даты в текстовую строку и наоборот.

Переменные

Сущность переменной может быть различной, но каждая переменная, прежде чем она будет применяться, должна быть определена, то есть должны быть объявлены ее имя и тип. На физическом уровне объявление переменной означает выделение ей определенного (постоянного или временного) места в памяти контроллера.

Имя переменной (ее идентификатор) выделяет каждую переменную среди других. Имя должно быть составлено из печатных символов и цифр. Существуют ограничения на выбор имен переменных.

Имя может быть однобуквенным (X, Z и т. д.), иметь цифровые индексы (XI, X2 и т. п.), вписываемые без пробела. Если есть необходимость в таком пробеле, например при написании двух или более слов, то в пробел ставится символ подчеркивания. Этот символ является значимым, то есть имена X_l, XI, _X1 и _X_1 воспринимаются программой как самостоятельные. Но цифру на первое место ставить нельзя: имена 1X, 2X недопустимы.

Нельзя применять в качестве имен операторы языков, например операторы языка IL: LD, ST, S, R, AND, MUL и др., а с индексами, символами подчеркивания или другими буквами – можно, например: SI, RU, AND_, MULTI и т. д. Даже не зная весь список операторов, легко установить ошибку – в процессе присваивания имени переменной при создании программы в том или ином пакете запрещенный идентификатор выделится (обычно цветом). Его необходимо удалить и вписать другое имя.

Регистр букв не влияет на работу ПЛК. Так, имена SET и Set воспринимаются одинаково.

В сложных программах трудно запомнить назначение того или иного элемента при упрощенной системе идентификации. Поэтому имя переменной (то есть идентификатор) можно записать в развернутом виде (к сожалению, не все системы программирования позволяют использовать буквы русского языка). Например, можно присвоить имена Dvigatel, pusk, BLOKIROVKA и т. д.

Области переменных и их адресация

Любая программа работает с определенным набором переменных, отражающих значения тех или иных величин. В программе контроллера, кроме «обычных» текущих переменных, которые рассчитываются или определяются по ходу программы, а также переменных, значения которых определяются сигналами на его физических входах, есть переменные, которые своими значениями определяют значения сигналов на выходах. Желательно иметь в своем распоряжении переменные, являющиеся общими и для системного, и для прикладного программного обеспечения для осуществления иногда необходимой их взаимозависимой работы.

Все это отразилось на том, что область памяти переменных контроллера всегда представляют в виде набора следующих областей:

− область входов ПЛК – I-область;
− область выходов ПЛК – Q-область;
− область прямо адресуемой памяти – M-область.
− оперативная память пользователя (ОЗУ).

Наименования разделов объявления переменных могут содержать дополнительные ключевые слова, уточняющие способ применения.

Таблица 7

Ключевые слова, уточняющие способ применения переменной в памяти ЦПУ

Ключевое слово	Применение переменной
RETAIN	Переменная должна быть размещена в энергонезависимой памяти. При выключении питания значение этой переменной сохраняется. (Такая память присутствует не во всех ПЛК)
CONSTANT	Константы, доступные только для чтения

Входы ПЛК – это переменные с адресами в области I. Они доступны в прикладных программах только для чтения. ЦПУ опрашивает физические входы контроллера в начале каждого рабочего цикла (см. рис. 2) непосредственно перед исполнением программы пользователя и записывает эти значения в образ процесса на входах, то есть в соответствующие биты и байты I-области. Обычно значения переменных из I-области памяти только считываются программой пользователя, так как физические входы получают свои значения непосредственно из подключенных к ним полевых устройств (датчиков), поэтому запись в эти входы запрещена.

Выходы ПЛК – это переменные с прямыми адресами в области Q, они доступны только для записи. ЦПУ копирует значения, хранящиеся в Q-области памяти, в физические выходы, то есть операционная система ПЛК преобразует данные из Q-области в соответствующие электрические сигналы, которые подаются на дискретные или аналоговые выходные порты (клеммы) ПЛК. Физические выходы непосредственно управляют полевыми устройствами, подключенными к этим выходам. К переменным Q-области разрешается доступ как на чтение, так и на запись.

В области M размещаются переменные, которые по каким-то причинам удобно разместить именно в области прямо адресуемых переменных. Переменные в области M могут быть доступны и для чтения, и для записи. В эти ячейки информацию заносит система исполнения (системное программное обеспечение), а прикладная программа может пользоваться этой информацией.

Например, можно назначить один байт в битовой M-области памяти в качестве системной памяти (рис. 35). Байт системной памяти предоставляет в распоряжение следующие четыре бита, на которые пользователь может ссылаться в своей пользовательской программе:

– бит Always 0 (low) [Всегда 0 (сброшен)] всегда установлен на 0;

– бит Always 1 (high) [Всегда 1 (установлен)] всегда установлен на 1;

– бит Diagnostic graph changed [Диагностическая диаграмма изменена] устанавливается на 1 на время одного цикла сканирования, после того как ЦПУ регистрирует диагностическое событие;

– бит First scan [Первый цикл] устанавливается на 1 на время первого цикла сканирования после завершения запуска. После исполнения первого цикла этот бит устанавливается в 0.

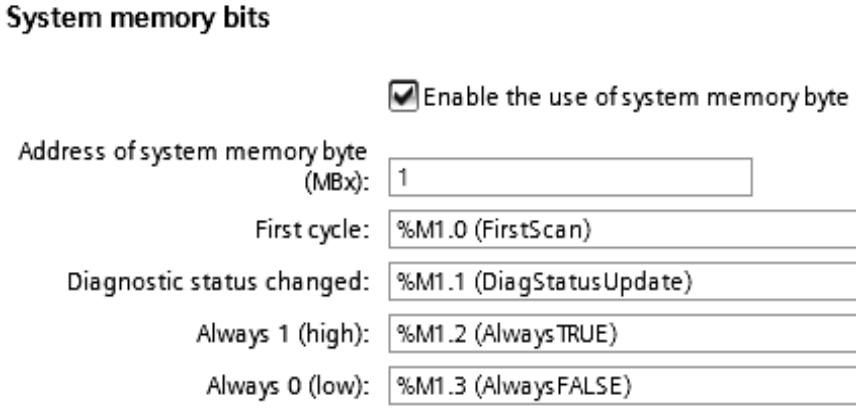


Рис. 35. Настройка байта системной памяти в M-области памяти:
System memory bits – биты системной памяти;
Enable the use of system memory byte – разблокировать использование байта системной памяти; Location of system memory byte – адрес байта системной памяти в M-области

Еще один полезный пример использования M-области памяти состоит в том, что пользователь может назначить один байт в битовой M-области памяти в качестве тактовых меркеров (рис. 36). Каждый бит этого байта, сконфигурированного в качестве тактового меркера, генерирует прямоугольный импульс. Байт тактовых меркеров предоставляет 8 различных частот, от 0,5 (медленно) до 10 Гц (быстро). Пользователь может использовать эти биты в качестве управляющих битов, особенно в соединении с командами обработки фронтов, для циклического запуска действий в программе пользователя.

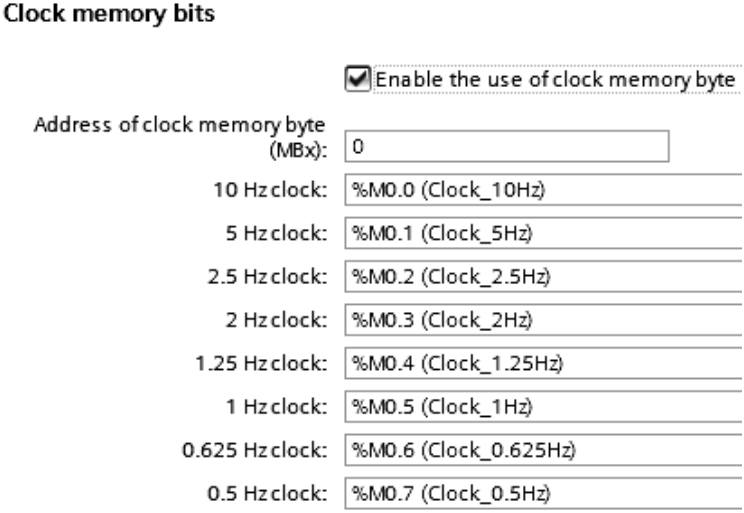


Рис. 36. Настройка байта тактовых битов памяти (меркеров) в M-области памяти: Clock memory bits – тактовые биты памяти; Enable the use of clock memory byte – разблокировать использование байта тактовых битов памяти; Location of clock memory byte – адрес байта тактовых битов; 10 Hz clock – тактовые импульсы с частотой 10 Гц

Для обращения к биту в области памяти вы указываете адрес, который включает в себя идентификатор области памяти, адрес байта и номер бита. Рис. 37 дает пример обращения к биту (это называется также адресацией байт.бит). В этом примере за областью памяти и адресом байта (I = вход (input), а 3 = байт 3) следует точка («.»), чтобы отделить адрес бита (бит 4).

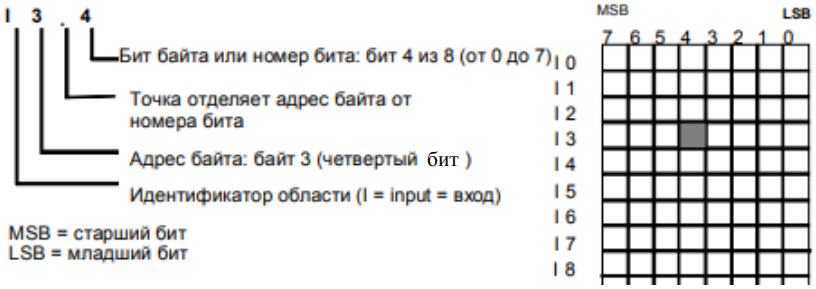


Рис. 37. Обращение к биту данных в памяти ЦПУ (адресация байт.бит)

Используя байтовый формат адреса, можно обращаться к данным в различных областях памяти ЦПУ (I, Q, M) как к байтам, словам или двойным словам. Для доступа к байту, слову или двойному слову в памяти ЦПУ нужно указать адрес, таким же способом, как и при указании адреса бита. Он включает идентификатор области (I-, Q-, M-область), обозначение размера данных и начальный байтовый адрес байта, слова или двойного слова, как это показано на рис. 38.

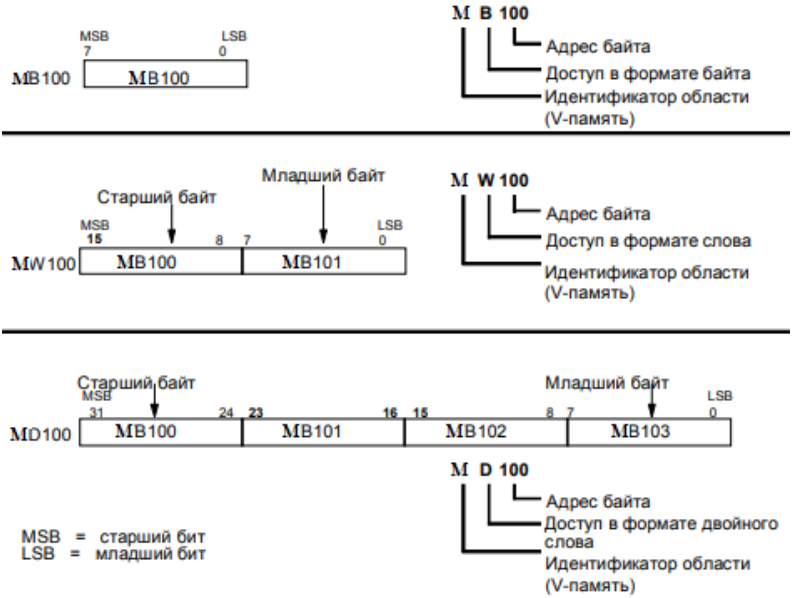


Рис. 38. Сравнение доступа к байту, слову и двойному слову по одному и тому же адресу

Для удобства работы с программой в пакетах программирования любой переменной можно дать символьное имя. Например, при создании программы для ПЛК компании Siemens это делается с помощью символьной таблицы переменных (тэгов). Такую адресацию называют тэговой.

Структура программы

В полную программу процесса входят операционная система и программа пользователя. Операционная система содержит общую часть всех инструкций и соглашений для реализации внутренних функций (например, сохранение данных при сбросе напряжения питания, управление реакцией пользователя при прерывании и т. д.). Она расположена в ПЗУ на так называемом EPROMe (Erasable Programmable Read Only Memory) и является фиксированной составной частью процессора. Пользователь не имеет возможности обращаться к операционной системе.

Программа пользователя содержит набор всех написанных пользователем инструкций и соглашений для обработки сигналов, с помощью которых производится управление установкой (процессом). В зависимости от размера и сложности программы пользователя можно выбрать для ее написания линейную или модульную структуру (рис. 39).

Линейная программа содержит все команды и выражения в одном программном блоке (организационный блок OB1 на рис. 39, *а*) и выполняет все команды последовательно друг за другом. Линейная структура организации программы пользователя подходит при простом алгоритме управления и короткой по размеру программе.

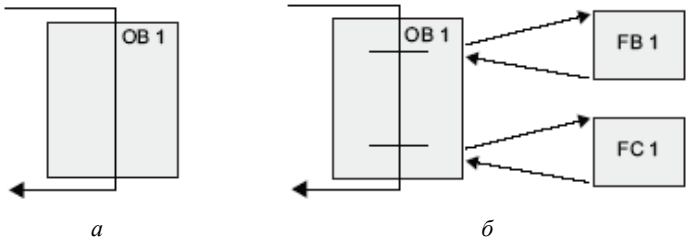


Рис. 39. Схематическое представление линейной (*а*) и модульной (*б*) структуры программы управления

При модульной структуре организации программы пользователя сложная задача автоматизации делится на небольшие подзадачи, соответствующие технологическим функциям процесса. Каждый кодовый блок содержит сегмент программы для соответствующей подзадачи. Если кодовый блок вызывается другим кодовым блоком, ЦПУ исполняет программный код в вызванном блоке. После того как вызванный блок обработан, ЦПУ возобновляет исполнение вызывающего блока (рис. 40).

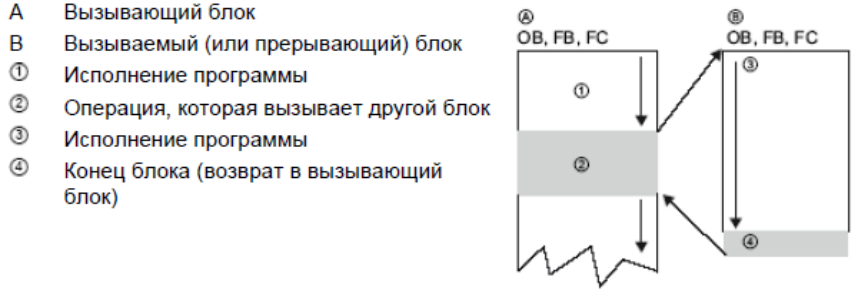


Рис. 40. Последовательность исполнения программы при вызове функционального (вызываемого) блока из главного (вызывающего) кодового блока

Компоненты организации программ (англ. Program Organization Unit – POU) являются базовыми элементами, из которых строится код проекта. Каждый компонент программы имеет собственное наименование, определенный интерфейс и создается на одном из языков стандарта. Один компонент может вызвать другие компоненты. Вызов самого себя (рекурсия) в стандарте МЭК не разрешен. Комбинировать различные языки в одном проекте можно при написании различных компонентов, но отдельный компонент целиком реализуется на одном языке МЭК. При вызове компонента язык его реализации значения не имеет.

К компонентам организации программ в стандарте относятся:

− функции;

− функциональные блоки;

− программы (или организационные блоки).

Все они во многом похожи, но имеют определенные особенности и различное назначение.

Компонент обладает свойством инкапсуляции – работает как «черный ящик», скрывая детали внутренней реализации. Для работы с компонентом достаточно знать его интерфейс, включающий описание входов и выходов. Внутреннее его устройство знать можно, но необязательно. Локальные (внутренние) переменные компонента извне недоступны.

Благодаря инкапсуляции компоненты успешно решают задачу структурной декомпозиции проекта. На верхнем уровне представления мы работаем с крупными компонентами. Каждый из них выполняет значительную для данного проекта задачу. Лишние подробности на этом уровне только мешают пониманию проблемы. Раскрывая вложенные компоненты один за другим, мы можем добраться до самого детального представления.

Еще одной задачей, решаемой компонентами, является локализация имен переменных. Переменные делятся на глобальные и локальные.

Глобальные переменные определяются на уровне проекта и доступны для всех его программных компонентов. Так, переменные, связанные со входами или выходами (переменные, размещенные в областях I и Q), всегда являются глобальными. Область видимости локальных переменных определяется рамками одного компонента.

FC (от англ. function – функция) – это программный компонент, отображающий множество значений входных параметров на выход. Функция всегда возвращает только одно значение. При объявлении функции указывается тип возвращаемого значения, имя функции и список входных параметров. Вызов функции производится по имени с указанием значений входных параметров. Функция может использоваться в математических выражениях, наряду с операторами и переменными. Функция не имеет внутренней памяти. Это означает, что функция с одними и теми же значениями входных параметров всегда возвращает одно и то же значение. Функция – это чистый код. Многократное использование функции не приводит к повторному включению кода функции при компоновке. Реализация функции присутствует в коде проекта только один раз. Всякий раз при вызове функции процессор исполняет один и тот же поименованный код. Функция может иметь локальные (временные) переменные, но при окончании своей работы она освобождает локальную память, которая может использоваться

в других функциях. Тип функций (тип возвращаемого значения) может быть любым из числа стандартных типов данных или типов, созданных пользователем. Тело функции может быть описано на языках IL, ST, LD или FBD. Использовать SFC нельзя. Из функции можно вызывать библиотечные функции и другие функции текущего проекта. Вызывать функциональные блоки и программы из функций нельзя.

FB (от англ. function block – функциональный блок) – программный компонент, отображающий множество значений входных параметров на множество выходных. После выполнения экземпляра функционального блока все его переменные сохраняются до следующего выполнения. Следовательно, функциональный блок, вызываемый с одними и теми же входными параметрами, может производить различные выходные значения. Сохраняются все переменные, включая входные и выходные. Так, если мы вызовем экземпляр функционального блока, не определяя значения некоторых входных параметров, он будет использовать ранее установленные значения. Возможность задания переменного числа входных значений заложена по определению и не требует каких-либо дополнительных усилий. Извне доступны только входы и выходы функционального блока, получить доступ к внутренним переменным блока нельзя. С позиций объектно-ориентированного программирования функциональные блоки – это объекты, реализующие инкапсуляцию, то есть сокрытие деталей реализации. Объединение кода и данных в единых рамках роднит функциональные блоки с классами объектно-ориентированных языков.

Прежде чем использовать функциональный блок, необходимо создать его экземпляр. Эта операция аналогична по смыслу объявлению переменной. Описав новый блок, мы фактически создали новый тип данных, подобный структуре. Каждый функциональный блок может иметь любое количество экземпляров. Так, различные экземпляры блока «таймер» совершенно независимы друг от друга. Каждый из них имеет собственные настройки и в программе «живет» собственной жизнью. Каждый экземпляр функционального блока имеет свой собственный уникальный идентификатор и свою область в статической памяти данных. Объявление еще одного экземпляра блока приводит к выделению еще одной области в памяти данных. Но код, очевидно, как и для функции, остается для всех экземпляров общим. Экземпляр функционального блока создается

в разделе объявлений переменных функционального блока, программы или в разделе глобальных переменных проекта.

ОВ (от англ. organization block – организационные блоки) – образуют интерфейс между операционной системой ЦПУ и программой пользователя. ОВ используются для исполнения определенных разделов программы:

− при запуске ЦПУ;

− при циклическом или зависящем от времени исполнении программы;

− при возникновении ошибок;

− при возникновении аппаратных прерываний.

Организационный блок циклического выполнения программы (ОВ1) исполняется операционной системой ЦПУ циклически, в каждом рабочем цикле в соответствии с рис. 2. Когда ОВ1 исполнен, операционная система отправляет глобальные данные. Перед повторным запуском ОВ1 операционная система записывает таблицу выходов образа процесса в модули вывода, обновляет таблицу входов образа процесса и получает глобальные данные для ЦПУ.

Операционная система ПЛК осуществляет контроль максимальной длительности цикла сканирования, чем гарантируется максимальное время реакции. Для ПЛК серии Simatic S7-1200, значение максимальной длительности цикла сканирования установлено по умолчанию на 150 мс. Если время выполнения программы превышает максимальное время цикла ОВ1, то операционная система вызывает ОВ 80 (ОВ обработки ошибок времени); если ОВ 80 не запрограммирован, то ЦПУ переходит в состояние STOP. Кроме контроля максимального времени цикла может быть гарантировано также минимальное время цикла сканирования. Операционная система задержит следующий запуск нового цикла (запись таблицы выходов образа процесса в модули вывода), пока не будет обеспечено минимальное время сканирования.

ПЛК серии SIMATIC S7 предоставляют также в распоряжение пользователя до девяти ОВ циклических прерываний (ОВ30–ОВ38), которые прерывают основную программу в блоке ОВ1 через строго фиксированные интервалы времени. Табл. 8 показывает установленные по умолчанию интервалы времени для ОВ циклических прерываний.

Таблица 8

Организационные блоки циклических прерываний по времени, доступные в ПЛК серии Simatic S7

Номер OB	Интервал времени по умолчанию
OB30	5 с
OB31	2 с
OB32	1 с
OB33	500 мс
OB34	200 мс
OB35	100 мс
OB36	50 мс
OB37	20 мс
OB38	10 мс

Прерывания по дате и времени. Существуют программы, которые должны выполниться один раз в определенный день и час или выполняться периодически, начиная с определенной даты и времени. Для этих целей в контроллерах серии Simatic S7 можно запрограммировать блоки прерываний по дате и времени. В распоряжение программиста имеется до восьми организационных блоков прерываний по времени (OB10–OB17), которые могут запускаться однократно или периодически со следующими интервалами: однократно, ежеминутно, ежечасно, ежедневно, еженедельно, ежемесячно, в конце каждого месяца.

Обработка включения питания. Часто при включении питания необходимо выполнить какие-либо однократные действия: первичную установку, инициализацию и т. д. Для этих целей предусмотрен организационный блок обработки включения питания OB100.

Организационные блоки аппаратных прерываний (OB40–OB47).

Контроллеры Simatic предоставляют в распоряжение программиста до восьми независимых друг от друга аппаратных прерываний со своими собственными OB.

Прерыванием называется временное прекращение выполнения микропроцессором текущей основной программы в режиме циклического сканирования (блок OB1) и переход к специальной подпрограмме-обработчику, то есть к одному из блоков OB40–OB47. Прерывание обычно происходит вне всякой связи с фоновой программой при поступлении сигнала от внешних выводов или от внутренних устройств микроконтроллера. Выполнение основной программы (рис. 41) останавливается, данные, необходимые для дальнейшего продолжения работы, сохраняются в отдельную область памяти (стек) и далее начинается выполнение подпрограммы обработки прерывания. После завершения исполнения процедуры обработки прерывания процессор контроллера возвращается к выполнению основной программы с того места, где оно было прервано.

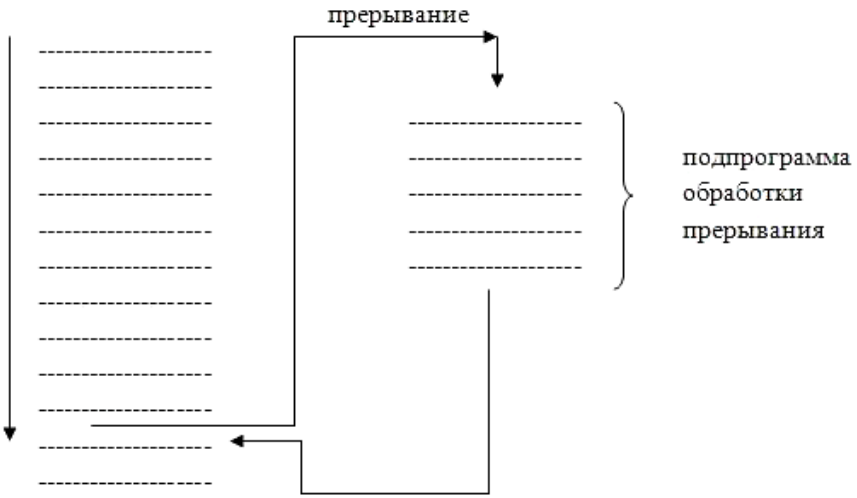


Рис. 41. Выполнение обработки аппаратного прерывания

Событиями, вызывающими аппаратные прерывания в ПЛК серии Simatic S7, являются, например, нарастающие и падающие фронты сигналов всех дискретных входов ЦПУ.

Нарастающий фронт возникает, когда цифровой вход переходит из состояния ВЫКЛ в состояние ВКЛ как реакция на изменение сигнала от полевого устройства (например, кнопки или конечного

выключателя), подключенного к этому входу. Падающий фронт возникает, когда цифровой вход переходит из состояния ВКЛ в состояние ВЫКЛ.

Таким образом, главное отличие организационного блока OB от функционального блока FB или функции FC состоит в том, что имеющиеся в программе *организационные блоки OB обязательно вызываются операционной системой ЦПУ и выполняются*:

− в каждом рабочем цикле (OB1);

− через определенный интервал времени (OB10–OB17, OB30–OB38);

− при запуске ПЛК из режима STOP в режим RUN (OB);

− при возникновении в системе события, вызывающего аппаратное прерывание (OB40–OB47).

Функциональные же блоки FB и функции FC операционной системой ПЛК не вызываются, а могут быть вызваны только соответствующей командой из программы пользователя, то есть из одного из организационных блоков.

Основные операторы. К основным операторам языка ST (SCL) относятся:

− арифметические операторы;

− логические (побитовые) операторы;

− операторы сравнения;

− оператор присвоения;

− операторы условного перехода: оператор IF – ELSEIF – ELSE, оператор CASE;

− операторы организации циклов: оператор FOR, оператор WHILE, оператор REPAET UNTIL.

Присвоение. Для обозначения присвоения используется парный знак «:=». В правой и левой части выражения должны быть операнды одного типа (автоматического приведения типов не предусмотрено). В левой части выражения (принимающая сторона) может быть использована только переменная. Правая часть может содержать выражение или константу.

Арифметические операторы:

+ – сложение;

- – вычитание;

* – умножение;

/ – деление;

mod – остаток от целочисленного деления.

Приоритет операций в выражениях указан в табл. 9 (чем выше приоритет, тем раньше исполняется операция). При записи арифметических выражений допустимо использование скобок для указания порядка вычислений.

Таблица 9

Приоритеты операций

Операция	Приоритет
Сравнения	1
Сложение, вычитание	2
Умножение, деление	3
OR	4
AND, XOR	5
NOT	6
Унарный минус	7
Вызов функции	8

Логические (побитовые) операторы:

OR – логическое (побитовое) сложение;

AND – логическое (побитовое) умножение;

XOR – логическое (побитовое) «исключающее ИЛИ»;

NOT – логическое (побитовое) отрицание.

В качестве примера использования логических операторов можно привести простейшую программу двухпозиционного регулирования уровня воды в баке гидроаккумулятора. На языке FBD данная программа имеет вид, представленный на рис. 31. Аналог данной программы на языке ST занимает всего одну строку кода и имеет следующий вид:

M := ((NOT LS1) OR M) AND (NOT LS2);

В операторах данной группы логическая операция выполняется для каждого бита из двух чисел. В результате новый номер – суммарный результат битовых операций.

Пример:

15 AND 8; // Результат: 8

Данная операция побитовая. Таким образом, чтобы понять, что здесь происходит, нужно преобразовать числа в двоичные значения:

$15_{10} = 1111_2$;

$8_{10} = 1000_2$.

Теперь каждый бит в числе 1111 (15) может быть использован в логической операции с другим числом 1000 (8): 1111 AND 1000.

Операторы сравнения:

= – сравнение на равенство;

<> – сравнение на неравенство;

> – сравнение на больше;

>= – сравнение на не меньше;

< – сравнение на меньше;

<= – сравнение на не больше.

В качестве результата сравнения всегда используется значение типа BOOL.

Пример:

TEMPERATURE := 93.9;

TEMPERATURE >= 100.0; // Результат: False

Операторы управления. С помощью операторов управления (control statements) пользователь может организовать ветвление программы, циклическое выполнение отдельных фрагментов программы и осуществлять переход в программе блока для выполнения другой ее части. Язык программирования ST поддерживает следующие операторы управления:

− IF (оператор для выполнения ветвления в программе по условию, проверяемому в отношении булевой переменной (или параметра типа BOOL));

− CASE (оператор для выполнения ветвления в программе по условию, проверяемому в отношении целой переменной (или параметра типа INT));

− FOR (оператор для организации в программе циклов с переменной – счетчиком циклов);

− WHILE (оператор для организации в программе циклов, инициируемых при выполнении определенного условия);

− REPEAT (оператор для организации в программе циклов с завершением по условию);

− CONTINUE (оператор для завершения текущего прохода цикла в программе);

− EXIT (оператор для выхода из цикла в программе);

− GOTO (оператор для продолжения выполнения программы, начиная с метки перехода);

− RETURN (оператор для выхода из программы блока).

Оператор IF. Оператор IF управляет выполнением той или иной части программы в зависимости от состояния булевой переменной. С помощью оператора IF пользователь может запрограммировать выполнение различных, определяемых условиями, ветвей программы.

```
IF condition THEN
statements;
…
…
END_IF;
```

Здесь condition – это адрес или выражение с типом BOOL. Если condition имеет значение True, то выполняются операторы после ключевого слова THEN. Если condition имеет значение False, то выполняются операторы после ключевого слова END_IF. Ключевое слово END_IF завершает оператор IF.

Пример:

```
IF condition THEN
statements1;
…
…
ELSE
statements0;
…
…
END_IF;
```

В данном примере, как и в предыдущем, condition имеет значение True или False. Если condition имеет значение True, то выполняются операторы после ключевого слова THEN. Если condition имеет значение False, то выполняются операторы после ключевого слова ELSE.

В качестве примера реализуем через оператор IF программу двухпозиционного регулирования уровня воды в баке гидроаккумулятора:

```
IF (LS1==FALSE) THEN // если уровень ниже нижней отметки – включить насос
M := TRUE;
END_IF;
IF (LS2==TRUE) THEN // если уровень выше верхней отметки – выключить насос
M := FALSE;
END_IF;
```

Оператор множественного выбора CASE. Данный оператор служит для организации выбора из диапазона значений. Формат записи оператора следующий:

```
CASE Expression OF
CASE_ELEMENT_1: statement_1;
CASE_ELEMENT_2: statement_2;
CASE_ELEMENT_3 .. CASE_ELEMENT_5: statement_2;
…
CASE_ELEMENT_n: statement_n;
[ELSE statement_0]
END_CASE;
```

CASE_ELEMENT – это список значений, перечисленных через запятую. Элементом списка может быть некоторое целое число (например, CASE_ELEMENT_1, CASE_ELEMENT_2) или диапазон целых чисел (например, CASE_ELEMENT_3 .. CASE_ ELEMENT_5).

Если текущее значение Expression не соответствует ни одному из приведенных в списке значений, то управление будет передано на предложение ELSE. Если предложение ELSE не указано, то никаких действий выполнено не будет.

Значение Expression может быть только целым. Например:

```
CASE k OF
1: k:=k*10;
2..5:  k:=k*5;
i:=0;
```

6, 9..20: k:=k–1;
ELSE k:=0;
i:=1;
END_CASE;

Если значение k равно 1, то будет выполнено умножение k на 10. Если значение k принадлежит числовому отрезку [2, 5], то будет выполнено умножение k на 5, а переменной i будет присвоено значение 0.

Если значение k будет равно 6 или будет принадлежать числовому отрезку [9, 20], то значение k будет уменьшено на 1.

Если значение k не соответствует ни одному из приведенных элементов списка, то в данном случае сработает предложение ELSE и переменной k будет присвоен 0, а переменной i присвоена 1.

При задании списка значений необходимо выполнять следующие условия:

− наборы значений внутри одного CASE не должны пересекаться;

− при указании диапазона значений начало диапазона должно быть меньше его конца.

Цикл FOR. Служит для задания цикла с фиксированным количеством итераций. Формат конструкции следующий:

FOR <Control Variable> := <expression1> TO <expression2>
[BY <expression3>] DO
<statement list>
END_FOR;

При задании условий цикла считается, что <Control Variable>, <expression1> … <expression3> имеют целые значения типа INT. Выход из цикла будет произведен в том случае, если значение переменной цикла превысит значение <expression2>. Например:

FOR i := 1 TO 10 BY 2 DO
k := k*2;
END_FOR;

Оператор BY задает приращение переменной цикла (в данном случае i будет увеличиваться на 2 при каждом проходе по циклу). Если оператор BY не указан, то приращение равно 1. Например:

FOR i := 1 TO (k/2) DO
var := var+k;

```
k := k–1;
END_FOR;
```

Внутри цикла могут использоваться другие циклы, операторы IF и CASE. Для выхода из цикла (любого типа) может использоваться оператор EXIT.

Пример:

```
FOR i := 1 TO 10 BY 2 DO
k := k*2;
IF k>20 THEN
EXIT;
END_IF;
END_FOR;
```

Цикл WHILE. Служит для определения цикла с предусловием. Цикл будет исполняться до тех пор, пока выражение в предложении WHILE возвращает TRUE. Формат конструкции следующий:

```
WHILE <Boolean-Expression> DO
<Statement List>
END_WHILE;
```

Значение <Boolean-Expression> проверяется на каждой итерации. Завершение цикла произойдет, если выражение <Boolean-Expression> вернет FALSE. Например:

```
k := 10;
WHILE k>0 DO
i := i+k;
k := k–1;
END_WHILE;
```

Внутри цикла могут использоваться другие циклы, операторы IF и CASE. Для досрочного завершения цикла используется оператор EXIT (см. пример в описание цикла FOR).

Цикл REPEAT UNTIL. Служит для определения цикла с постусловием. Завершение цикла произойдет тогда, когда выражение в предложении UNTIL вернет True. Другими словами: цикл будет выполняться, пока условие в предложении UNTIL не выполнится. Формат конструкции следующий:

```
REPEAT
<Statement List>
```

```
UNTIL <Boolean Expression>;
END_REPEAT;
Например:
k := 10;
REPEAT
i := i + k;
k := k – 1;
UNTIL k = 0;
END_REPEAT;
```

Внутри цикла могут использоваться другие циклы, операторы IF и CASE. Для досрочного завершения цикла используется оператор EXIT (см. пример в описании цикла FOR).

Распознавание нарастающего и падающего фронта

Операторы R_TRIG и F_TRIG используются для обнаружения, соответственно, положительного и отрицательного фронта изменения двоичного сигнала. Положительный фронт – это изменение двоичного сигнала с значения False на True, отрицательный фронт – изменение с True на False.

P_TRIG – поток сигнала или логическое состояние на выходе Q принимает значение True, когда нарастающий фронт (переход из ВЫКЛ во ВКЛ) обнаружен в состоянии на входе CLK.

F_TRIG – поток сигнала или логическое состояние на выходе Q принимает значение True, когда падающий фронт (переход из ВКЛ в ВЫКЛ) обнаружен в состоянии на входе CLK.

Все команды обнаружения фронта используют бит памяти (M_BIT) для хранения предыдущего состояния подлежащего контролю входного сигнала. Фронт обнаруживается путем сравнения состояния входа с состоянием этого бита памяти. Если эти состояния указывают на изменение сигнала на входе в интересующем нас направлении, то о появлении фронта сообщается установкой выхода в состояние True. Иначе выход устанавливается в состояние False.

Таймеры

С помощью таймерных команд вы можете создавать программируемые запаздывания:

TP – импульсный таймер генерирует импульс заданной длительности (рис. 42).

TON – выход Q таймера с запаздыванием включения устанавливается в состояние ВКЛ по истечении заранее заданного времени (рис. 43).

TOF – Выход Q таймера с запаздыванием выключения устанавливается в состояние ВЫКЛ по истечении заранее заданного времени (рис. 44).

TONR – выход запоминающего таймера с запаздыванием включения устанавливается в состояние ВКЛ по истечении заранее заданного времени. Истекшее время накапливается в течение нескольких интервалов выдержки таймера, пока вход R не будет использован для сброса истекшего времени (рис. 45).

RT – сбрасывает таймер, стирая данные о времени, хранящиеся в заданном экземплярном блоке данных таймера.

Каждый таймер использует структуру, хранящуюся в блоке данных, для сохранения данных о времени (табл. 10). Вы назначаете блок данных, когда вы вставляете таймерную команду в редакторе.

Таблица 10

Параметры программного блока таймера

Параметр	Тип данных	Описание
IN	Bool	Разблокирующий вход таймера
R	Bool	Сброс на ноль истекшего времени таймера TONR
PT	Bool	Вход предустановленного значения времени
Q	Bool	Выход таймера
ET	Time	Выход истекшего времени
Блок данных таймера	DB	Указывает, какой таймер должен быть сброшен командой RT

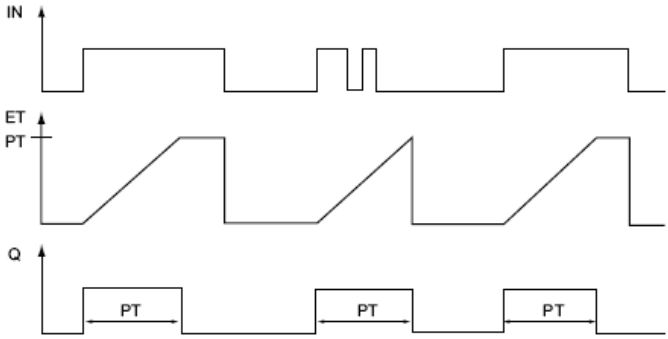


Рис. 42. Временная диаграмма таймера TP

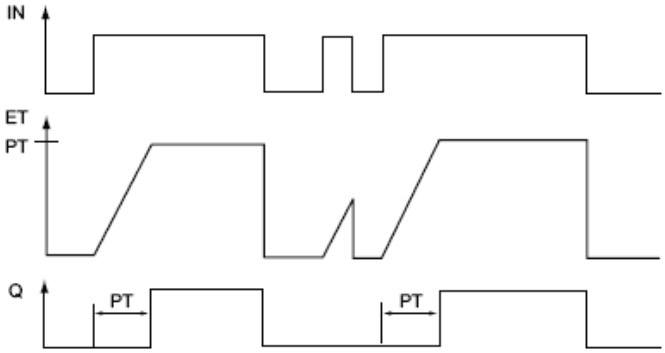


Рис. 43. Временная диаграмма таймера TON

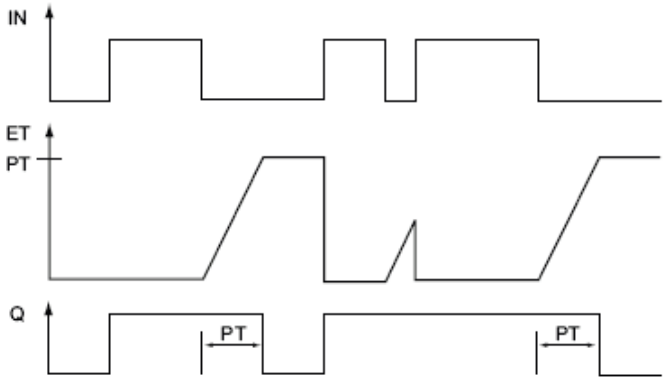


Рис. 44. Временная диаграмма таймера TOF

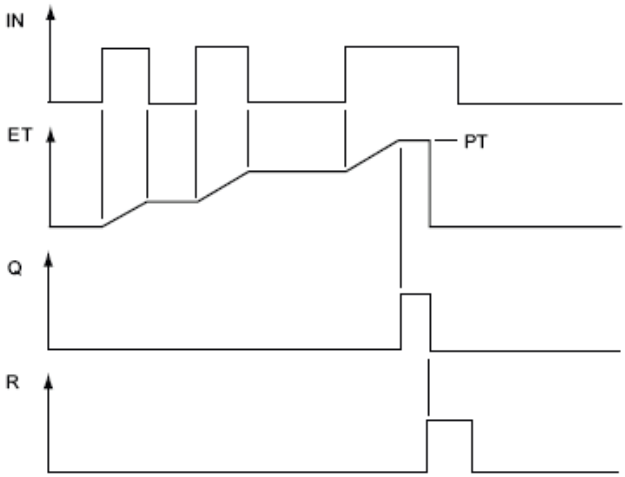


Рис. 45. Временная диаграмма таймера TONR

Параметр IN запускает и останавливает таймеры:

- переход с 0 на 1 параметра IN запускает таймеры TP, TON и TONR;
- переход с 1 на 0 параметра IN запускает таймер TOF.

В табл. 11 показано влияние изменений значения в параметрах PT и IN.

Таблица 11

Параметры программного блока таймера

Таймер	Изменения в параметрах PT и IN
TP	Изменение PT не оказывает влияния во время работы таймера. Изменение IN не оказывает влияния во время работы таймера
TON	Изменение PT не оказывает влияния во время работы таймера. Изменение IN на ЛОЖЬ, когда таймер работает, сбрасывает и останавливает таймер

Окончание таблицы 11

Таймер	Изменения в параметрах PT и IN
TOF	Изменение PT не оказывает влияния во время работы таймера. Изменение IN на значение ИСТИНА, когда таймер работает, сбрасывает и останавливает таймер
TONR	Изменение PT не оказывает влияния во время работы таймера, но оказывает влияние, когда таймер возобновляет работу. Изменение IN на ЛОЖЬ, когда таймер работает, останавливает таймер, но не сбрасывает его. Изменение IN обратно на значение ИСТИНА заставляет таймер работать, начиная с накопленного значения времени

Значения PT (preset time [предустановленное время]) и ET (elapsed time [истекшее время]) хранятся в памяти как двойные целые со знаком, которые представляют миллисекунды. Тип данных TIME использует идентификатор T# и может быть введен как простая единица времени "T#200ms" или в виде комбинированных единиц времени "T#2s_200ms" (табл. 12).

Таблица 12

Диапазон возможных значений переменной типа Time

Тип данных	Размер	Допустимый диапазон значений
Time	32 бита	от T#-24d_20h_31m_23s_648ms до T#24d_20h_31m_23s_647ms от -2 147 483 648 мс до +2 147 483 647 мс

Счетчики

С помощью команд счета вы можете подсчитывать события внутри программы и внешние события в процессе:

CTU – суммирующий счетчик.

CTD – вычитающий счетчик.

CTUD – реверсивный счетчик.

Описание функций счетчиков представлено в табл. 13.

Таблица 13

Параметры программного блока счетчика

Параметр	Тип данных	Описание
CU, CD	Bool	Счет вверх или вниз, каждый раз на одну единицу
R (CTU, CTUD)	Bool	Сброс значения счетчика в ноль
LOAD (CTD, CTUD)	Bool	Управление загрузкой для предустановленного значения
PV	SInt, Int, DInt, USInt, UInt, UDInt	Предустановленное значение
Q, QU	Bool	Истина, если CV >= PV
QD	Bool	Истина, если CV <= 0
CV	SInt, Int, DInt, USInt, UInt, UDInt	Текущее значение счетчика

Числовой диапазон значений счетчика зависит от выбранного вами типа данных. Если значение счетчика целое без знака, то вы можете считать в обратном направлении до нуля, а в прямом направлении до границы диапазона. Если значение счетчика целое со знаком, вы можете считать в обратном направлении до нижней границы, а в прямом направлении до верхней границы.

CTU увеличивает значение на 1, когда значение параметра CU изменяется с 0 на 1. Если значение параметра CV (текущее значение счетчика) больше или равно значению параметра PV (предустановленное значение счетчика), то выходной параметр счетчика Q = 1. Если значение параметра сброса R изменяется с 0 на 1, то текущее значение счетчика сбрасывается в 0. На рис. 46 показана временная диаграмма CTU.

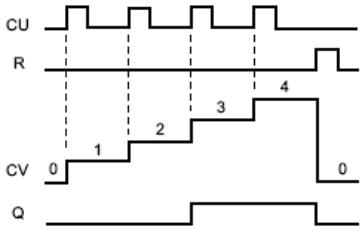


Рис. 46. Временная диаграмма работы счетчика CTU

CTD уменьшает значение на 1, когда значение параметра CD изменяется с 0 на 1. Если значение параметра CV (текущее значение счетчика) меньше или равно 0, то выходной параметр счетчика Q = 1. Если значение параметра LOAD изменяется с 0 на 1, то значение параметра PV (предустановленное значение) загружается в счетчик как новое CV (текущее значение счетчика). На рис. 47 показана временная диаграмма CTD со значением счетчика типа целое без знака (где PV = 3).

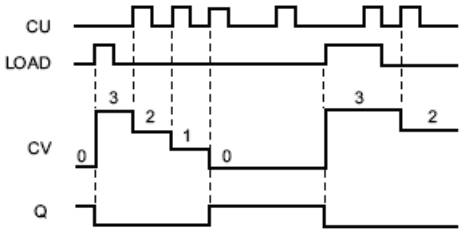


Рис. 47. Временная диаграмма работы счетчика CTD

CTUD увеличивает или уменьшает значение на 1, когда происходит переход с 0 на 1 соответственно на входе прямого (CU) или обратного (CD) счета. Если значение параметра CV (текущее значение счетчика) больше или равно значению параметра PV (предустановленное значение), то выходной параметр счетчика QU = 1. Если значение параметра CV меньше или равно нулю, то выходной параметр счетчика QD = 1. Если значение параметра LOAD изменяется с 0 на 1, то значение параметра PV (предустановленное значение) загружается в счетчик как новое CV (текущее значение счетчика). Если значение параметра сброса R изменяется с 0 на 1, то текущее значение счетчика сбрасывается в 0. На рис. 48 показана

временная диаграмма CTUD со значением счетчика типа целое без знака (где PV = 4).

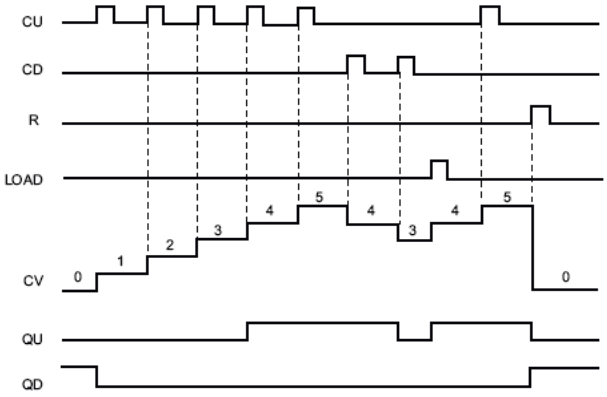
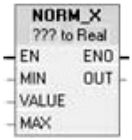


Рис. 48. Временная диаграмма работы счетчика CTUD

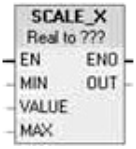
Команды нормализации NORM_X и масштабирования SCALE_X



Команда нормализации NORM_X нормализует входное значение VALUE внутри диапазона значений, указанного в параметрах MIN и MAX, в соответствие со следующей формулой:

$$\mathrm{OUT} = \frac{\mathrm{VALUE} - \mathrm{MIN}}{\mathrm{MAX} - \mathrm{MIN}}.$$

Таким образом, по сути, выполняется линейное сжатие значения VALUE от некоторого исходного диапазона $[\mathrm{MIN};\mathrm{MAX}]$ возможных значений к диапазону $[0;1]$.



Команда масштабирования SCALE_X масштабирует нормализованный вещественный параметр VALUE, где (0,0 <=

VALUE <= 1,0), в тип данных и диапазон значений, указанные в параметрах MIN и MAX:

$$\mathrm{OUT} = \mathrm{MIN} + \mathrm{VALUE} \cdot (\mathrm{MAX} - \mathrm{MIN}).$$

Таким образом, по сути, выполняется линейное растяжение значения VALUE от исходного диапазона $[0;1]$ к некоторому конечному диапазону $[\mathrm{MIN};\mathrm{MAX}]$ значений.

Команды NORM_X и SCALE_X обычно используются для обработки аналоговых сигналов с аналоговых входов и выходов ПЛК. Аналоговые сигналы применяются для измерения значений различных изменяющихся физических величин, таких как температура, давление, расход и т. д. Но так как различных типов сигналов очень много, для удобства используются нормирующие преобразователи, встроенные в датчик, либо выносные, которые преобразуют выходной электрический сигнал с датчика в унифицированный токовый сигнал 4–20 мА или сигнал напряжения 0–10 В. И далее этот унифицированный сигнал приходит на аналоговый вход ПЛК.

Далее аналоговый сигнал с помощью АЦП (аналого-цифрового преобразователя) преобразуется в некое целочисленное значение, обычно в формате Integer (целые числа в диапазоне от –32768 до 32767). Так, в ПЛК S7-1200 для сигнала 4–20 мА представление аналогового сигнала изменяется от 0 до 27648 счетов АЦП. Затем это число в программе с помощью специальных инструкций необходимо перевести в реальные единицы измерения.

Типовой метод преобразования значения аналогового входа использует инструкции NORM_X – нормализации значения и SCALE_X – масштабирования.

NORM_X нормализует параметр VALUE к диапазону значений, определенному параметрами MIN и MAX (рис. 49). На вход параметра VALUE приходит значение с датчика. MIN – минимальное значение в диапазоне, в данном случае 0, MAX – максимальное значение, то есть 27648. На выходе мы получаем нормализованное значение, которое заносим в переменную temp_value.

Затем масштабируем полученное значение согласно диапазону измерения датчика, например, для датчика давления, от 0 до 160 бар (рис. 50).

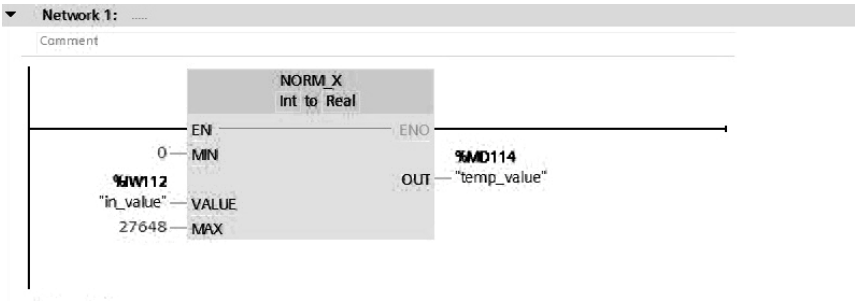


Рис. 49. Использование команды NORM_X при обработке сигнала аналогового датчика

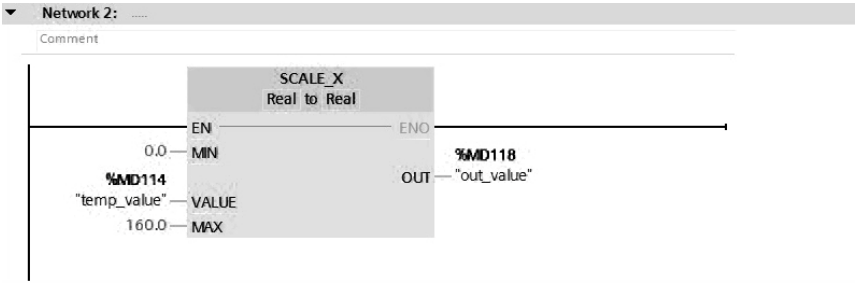


Рис. 50. Использование команды SCALE_X при обработке сигнала аналогового датчика

Результирующим значением выходной переменной out_value будет фактическое значение давления в bar. Таким образом мы из электрического сигнала с датчика давления получаем целочисленное значение этой переменной.

НАСТРОЙКА И ПРОГРАММИРОВАНИЕ КОНТРОЛЛЕРОВ В СРЕДЕ ПРОГРАММИРОВАНИЯ TIA PORTAL

Для разработки программ для ПЛК в настоящее время используются интегрированные среды разработки (ИСР), или англ. Integrated development environment (IDE), содержащие в своем составе текстовые редакторы, компиляторы, редакторы связей, загрузчики и симуляторы. ИСР обычно представляет собой единственную программу, в которой проводится вся разработка. Она, как правило, содержит много функций для создания, изменения, компилирования, развертывания и отладки программы ПЛК.

Система программирования контроллеров фирмы Сименс

TIA Portal (Totally Integrated Automation Portal) – интегрированная среда разработки программного обеспечения систем автоматизации технологических процессов на основе оборудования производства фирмы Siemens. В TIA Portal объединены три основных программных пакета (рис. 51):

− Simatic Step 7 v.11 для программирования контроллеров S7-1200, S7-300, S7-400 и WinAC;

− Simatic WinCC v.11 для разработки человеко-машинного интерфейса (программирование сенсорных панелей и SCADA-систем);

− Sinamics StartDrive v.11 для программирования преобразователей частоты Sinamics;

− PLCSIM – симулятор работы ПЛК.

В данной программе, как в единой программной платформе, объединено все необходимое для работы с компонентами автоматизации Siemens на всех этапах работы с проектом. Разработка проектов для контроллеров и устройств распределенного ввода/вывода, конфигурирование систем человеко-машинного интерфейса и SCADA-систем, параметрирование сетевых компонентов и модулей связи, отладка программных алгоритмов управления, а также ввод в эксплуатацию приводов – все это объединено в общую структуру программного обеспечения и имеет унифицированный пользовательский интерфейс.

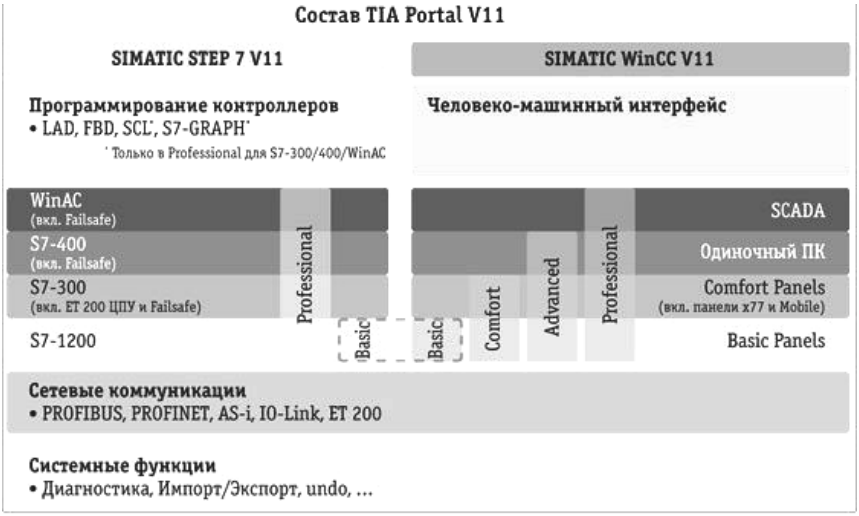


Рис. 51. Состав TIA Portal

После окончания процесса установки TIA Portal на компьютер на рабочем столе компьютера появится соответствующий ярлык для запуска TIA Portal:  . После запуска TIA Portal появится начальное окно программы. В среде TIA Portal предусмотрено два способа отображения структуры проекта автоматизации: портальное представление (portal view) и проектно-ориентированное представление (project view). Портальное представление отображает структуру проекта с точки зрения задач и функций, которые могут быть выполнены в проекте, например, создание нового проекта – Create new project, открытие уже созданного ранее и сохраненного на жестком диске проекта – Open existing project, отображение используемых в проекте устройств (контроллеров, панелей оператора, модулей ввода/вывода и др.) и настройку сетевых соединений между устройствами – Devices & networks, мониторинг и диагностика доступных для программирования в данном проекте устройств – Online & Diagnostics и др.

Проектно-ориентированное представление (рис. 52) отображает все компоненты внутри проекта и позволяет получить быстрый доступ к любому из них. В процессе работы над проектом при необходимости

в любой момент можно переключиться от портального к проектно-ориентированному представлению структуры проекта и обратно.

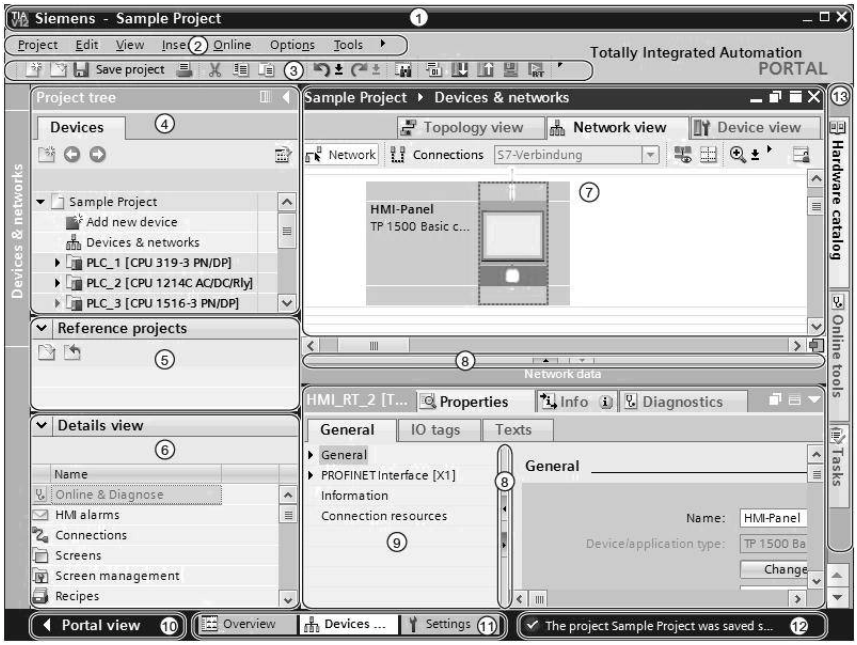


Рис. 52. Проектно-ориентированное представление структуры проекта (project view):
1 – панель заголовка (название проекта); 2 – главное меню;
3 – панель кнопок управления; 4 – дерево проекта;
5 – отображение других проектов, связанных с данным проектом;
6 – подробные данные об объекте, выбранном в дереве проекта;
7 – рабочая область окна; 8 – разделители; 9 – окно инспектора свойств объектов;
10 – переход к портальному представлению;
11 – панель переключения между задачами; 12 – строка состояния;
13 – панель вкладок библиотек компонентов

Конфигурация устройств

Конфигурация устройств для конкретного ПЛК создается добавлением ЦПУ и других модулей в создаваемый проект.

Для создания своей конфигурации устройств вставьте в свой проект ЦПУ. Выбор ЦПУ в диалоговом окне Add a new device [Добавить новое устройство] создает стойку и ЦПУ.

У ЦПУ нет заранее сконфигурированного IP-адреса. Программисту нужно вручную назначить IP-адрес для ЦПУ при создании конфигурации устройств. Если имеющийся ЦПУ подключен к маршрутизатору в сети, то нужно также ввести IP-адрес для маршрутизатора.

Если вы подключены к ЦПУ, то вы можете загрузить конфигурацию этого ЦПУ, включая возможно имеющиеся модули, в свой проект. Создайте для этого новый проект и выберите вместо определенного ЦПУ неопределенный ЦПУ (Unspecified CPU). В программном редакторе в меню Online выберите команду Hardware detection [Распознавание аппаратуры]. В редакторе конфигурации устройств выберите опцию для распознавания конфигурации подключенного устройства. После того как вы выбрали ЦПУ в диалоговом окне Online, STEP 7 Basic загружает конфигурацию аппаратуры из ЦПУ, включая возможные модули (SM, SB или CM). Затем вы можете конфигурировать параметры для ЦПУ и модулей.

Конфигурирование работы ЦПУ

Для конфигурирования рабочих параметров ЦПУ (CPU) выберите ЦПУ в отображении набора устройств (синяя рамка вокруг всего ЦПУ) и откройте вкладку Properties [Свойства] в окне просмотра параметров (рис. 53).

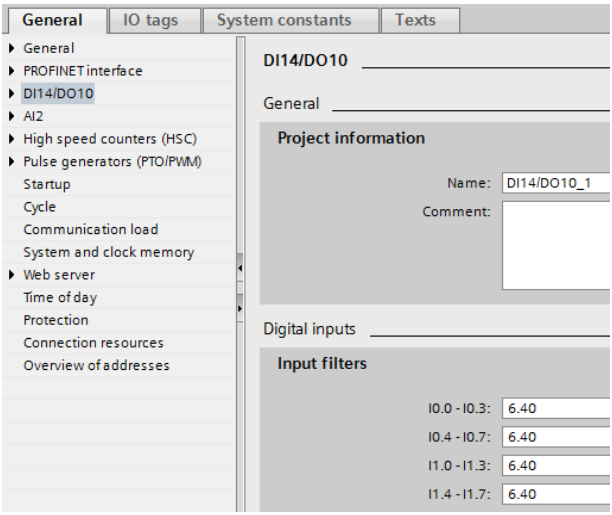


Рис. 53. Рабочая вкладка Properties [Свойства] для аппаратной настройки ПЛК в среде TIA Portal

В окне свойств вы можете установить следующие параметры:

– интерфейс PROFINET: установка IP-адреса для ЦПУ и синхронизации времени;

– DI, DO, и AI: настройка поведения локальных (встроенных) цифровых и аналоговых входов и выходов;

– скоростные счетчики и генераторы импульсов: активизация и настройка быстрых счетчиков (HSC) и генераторов импульсов, используемых для операций с последовательностями импульсов (pulse-train operations, PTO) и широтно-импульсной модуляции (pulse-width modulation, PWM).

Когда вы конфигурируете выходы ЦПУ или сигнальной платы в качестве генераторов импульсов (для использования с PWM или основными командами управления перемещениями), соответствующие адреса выходов (Q0.0, Q0.1, Q4.0 и Q4.1) удаляются из памяти выходов (Q) и не могут быть использованы для других целей в вашей пользовательской программе. Если ваша пользовательская программа запишет какое-либо значение в выход, используемый в качестве генератора импульсов, то ЦПУ не запишет это значение в физический выход;

– запуск: настройка поведения ЦПУ после выключения и последующего включения, например, для запуска в состоянии STOP или перехода в режим RUN после теплого пуска;

– время суток: установка времени, часового пояса и переключения между летним и зимним временем;

– защита: установка защиты от чтения/записи и установка пароля для доступа к ЦПУ;

– системная и тактовая битовая память (тактовые меркеры): Установка байта для функций «системной памяти» (для битов «первый цикл», «всегда включен» и «всегда выключен») и установка байта для функций «тактовой памяти» (где каждый бит включается и выключается с заранее заданной частотой);

– время цикла: установка максимального времени цикла или фиксированного минимального времени цикла;

– коммуникационная нагрузка: назначение процентной доли времени ЦПУ для коммуникационных задач.

Интерфейс связи контроллера и компьютера

Онлайновое соединение между устройством программирования и целевой системой необходимо для загрузки программ и данных

проекта в целевую систему, а также, например, для следующих действий:

– тестирование программ пользователя;
– отображение и изменение режима работы ЦПУ;
– отображение и установка даты и времени на ЦПУ;
– отображение информации о модуле;
– сравнение онлайновых и оффлайновых блоков;
– диагностика аппаратуры.

Затем можно будет обратиться к данным в целевой системе в онлайновом или диагностическом представлении через панель задач Online tools [Онлайновые инструментальные средства].

Текущее онлайновое состояние устройства отображается пиктограммой справа рядом с устройством в отображении проекта. Оранжевый цвет указывает на онлайновое соединение. Выберите Accessible Nodes [Доступные узлы], чтобы найти ЦПУ, находящиеся в сети.

Запись прикладной программы в память контроллера

Обычно программа ПЛК загружается в контроллер Simatic S7-1200 с помощью TIA Portal (рис. 54), в результате она записывается на карту памяти SIMATIC Memory Card, которая вставлена в ЦПУ.

Вы можете загрузить свой проект из устройства программирования в ЦПУ из любого из следующих мест:

– «Дерево проекта»: щелкните правой клавишей мыши на элементе программы, а затем выберите в контекстном меню пункт Download [Загрузить];

– меню Online: щелкните на опции Download to device [Загрузить в устройство];

– панель инструментов: щелкните на символе Download to device [Загрузить в устройство].

Возможна также ситуация, когда у заказчика, который находится в другом городе, нет TIA Portal. При этом у разработчика проекта нет связи с контроллером заказчика.

Необходимо загрузить программу в ПЛК.

Решение данной задачи включает в себя следующие шаги:

1. Создаем папку SIMATIC_MC для загрузочных данных: C:\Virtual_SIMATIC_Memory_Card\EcoLe_Line1\SIMATIC_MC.

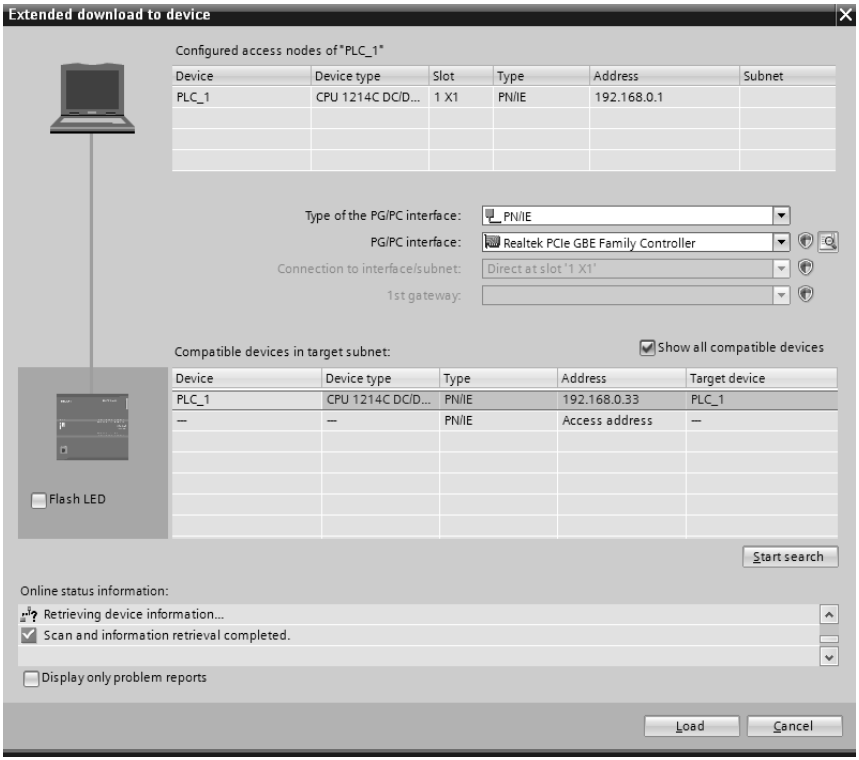


Рис. 54. Диалоговое окно загрузки пользовательской программы из среды TIA Portal в память ПЛК

2. Открываем проект в TIA Portal и добавляем пользовательский Card Reader (рис. 55). Указываем путь к созданной папке и нажимаем кнопку OK. В TIA Portal создается виртуальная карта памяти.

3. Записываем загрузочные данные на виртуальную карту памяти (рис. 56). Выбираем созданную виртуальную карту памяти и нажимаем кнопку с зеленой галочкой.

4. Выбираем целевое устройство – ЦПУ и нажимаем кнопку Load.

5. Убеждаемся, что в созданную папку SIMATIC_MC записались загрузочные данные.

6. Архивируем эту папку и отправляем архивный файл заказчику по электронной почте.

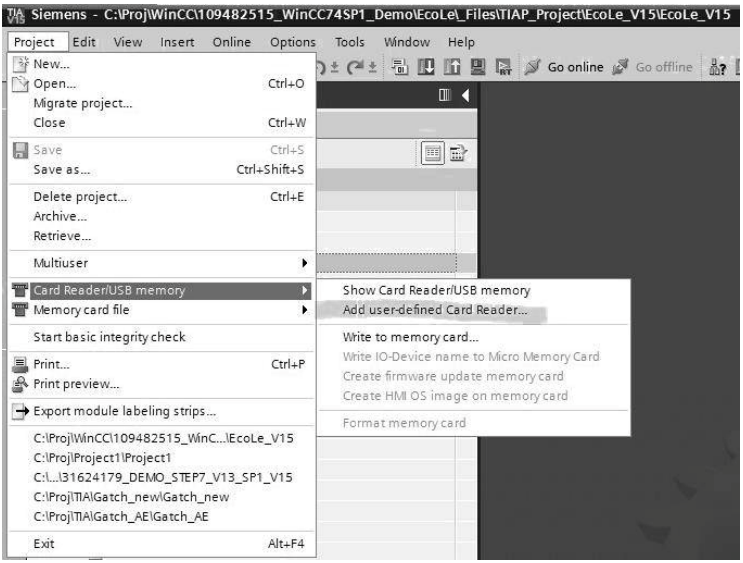


Рис. 55. Действия по добавлению пользовательского Card Reader в проект TIA Portal

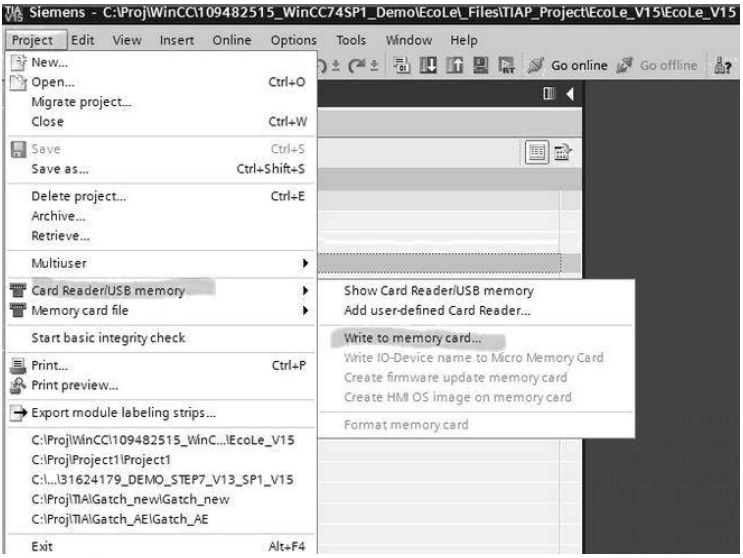


*Рис. 56. З*апись загрузочных данных в виртуальную карту памяти в проекте TIA Portal

7. Заказчик получает этот архивный файл, распаковывает его, выключает ЦПУ, вытаскивает из него карту памяти SIMATIC Memory Card, вставляет карту памяти в кардридер на своем ноутбуке и копирует на нее загрузочные данные.

8. Заказчик вставляет карту памяти в ЦПУ и включает ПЛК.

Создание резервной копии S7-1500: выгрузка проекта из ПЛК, архивирование проекта.

Предположим, что кто-то другой разработал проект в среде TIA Portal, написал программу управления некоторым объектом, загрузил программу в контроллер Simatic, и в настоящий момент эта программа выполняется на контроллере.

У вас нет исходного кода проекта, но вы хотите выгрузить его из контроллера в TIA Portal, не остановив при этом производство, и сделать резервную копию прикладной программы.

Решение данной задачи включает в себя следующие шаги.

1. Подключаемся к ПЛК, открываем TIA Portal и через сервис Online access находим доступные в сети устройства (рис. 57).

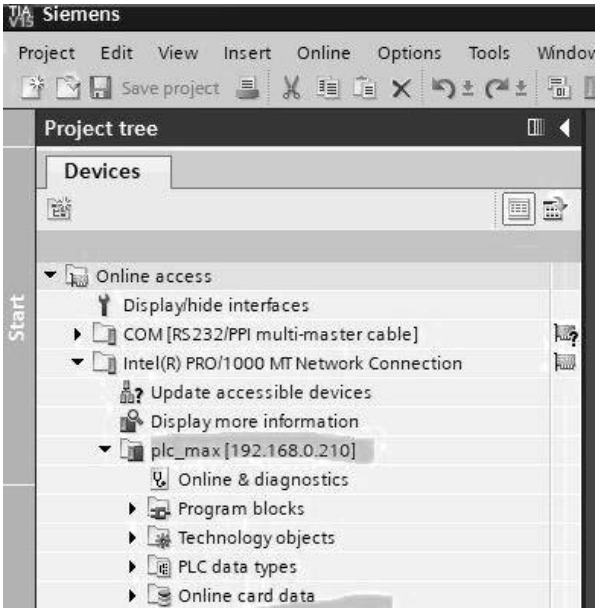


Рис. 57. Настройка online-доступа к подключенному к сети контроллеру в среде TIA Portal

2. Открываем меню Online и запускаем процедуру Upload device as new station (рис. 58). Убеждаемся, что проект скачался из ПЛК в TIA Portal.

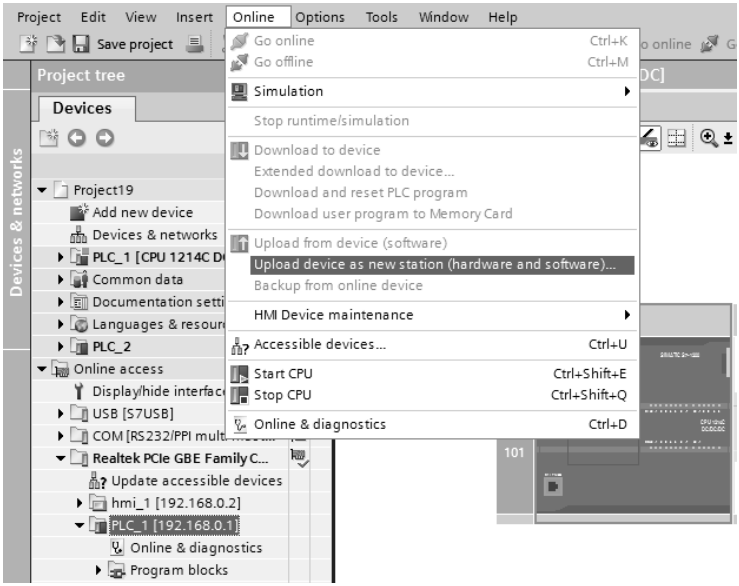


Рис. 58. Скачивание проекта из ПЛК в TIA Portal

3. Сохраняем проект в архив (Archive) (рис. 59). Для извлечения проекта из архива достаточно выполнить обратную операцию Retrieve.

Если данные в ПЛК защищены с помощью пароля, то без пароля скачать проект не получится.

4. Для создания полной резервной онлайн копии ЦПУ как точки восстановления, можно использовать процедуру Backup from online device, которая выполняется при остановленном ЦПУ (рис. 60).

В онлайн копии сохраняются актуальные значения сохраняемых тегов (retentive tags). Онлайн копию нельзя открыть и изменить. Точка восстановления (online backup) сохраняется в специальном разделе проекта.

В случае необходимости онлайн копия может быть загружена в ПЛК.

Сохраняемые теги ПЛК (Retentive PLC tags) сохраняют свои значения при пропадании и восстановлении питания.

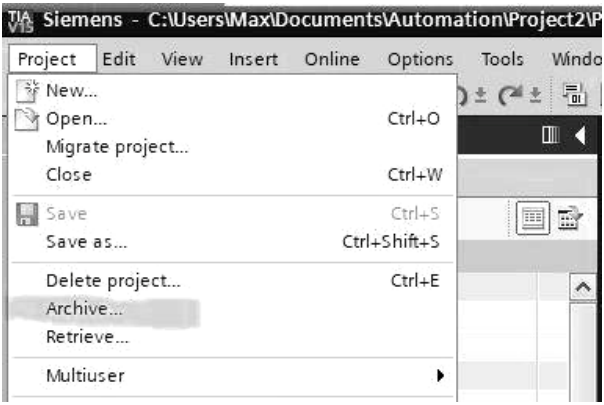


Рис. 59. Архивирование проекта в TIA Portal

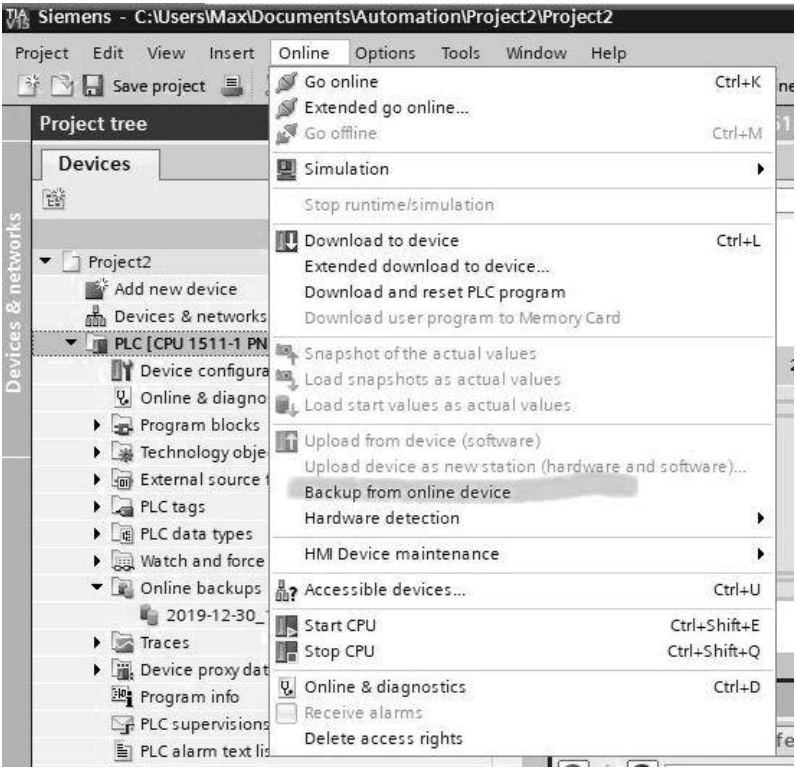


Рис. 60. Создание полной резервной онлайн копии ЦПУ в TIA Portal

Мониторинг выполнения программы

Для использования функции Monitoring On/Off необходимо в окне главной программы Main OB1 нажать кнопку Monitoring On/Off  . После чего схема примет вид, показанный на рис. 61.

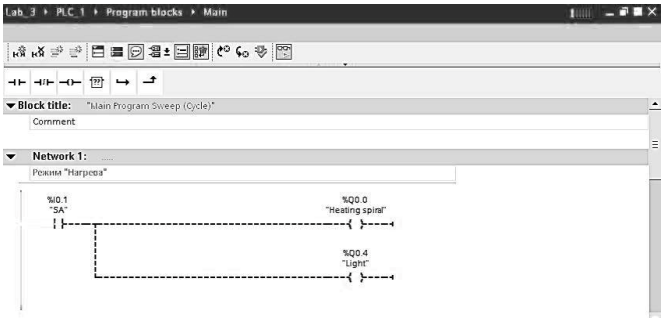


Рис. 61. Функция Monitoring On/Off активна

Аналогично для использования функции Monitor All в окне редакторе таблицы PLCtags необходимо нажать кнопку Monitor All  (рис. 62).

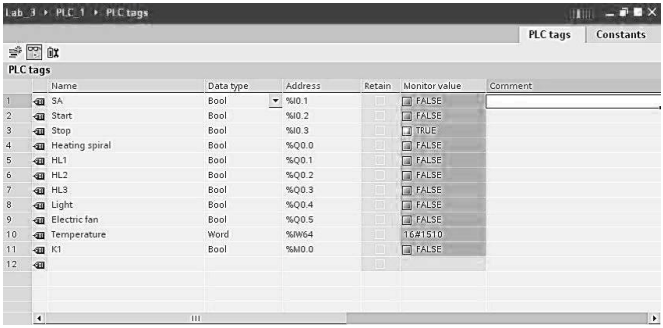


Рис. 62. Функция Monitoring All активна

Значения всех переменных отображаются в столбце Monitorvalue. Если дискретный сигнал неактивен (значение на I/O соответствует 0), то сигнал принимает значение False, если активен – True. Для аналоговых сигналов указывается формат данных и значение величины сигнала.

ОТЛАДКА И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Таблицы наблюдения для контроля программы пользователя

Таблица наблюдений позволяет осуществлять функции контроля и управления в информационных точках, когда ЦПУ выполняет вашу программу. Этими информационными точками могут быть элементы образа процесса (I или Q), физические входы или выходы (I_:P или Q_:P), M или DB в зависимости от функции контроля и управления.

Функция контроля не изменяет процесс исполнения вашей программы. Она снабжает вас информацией об исполнении программы и о данных программы в ЦПУ.

Функции управления позволяют пользователю управлять последовательностью исполнения и данными программы. При использовании функций управления следует соблюдать осторожность. Эти функции могут существенно влиять на исполнение пользовательской или системной программы. Этими тремя функциями являются изменение, принудительное задание и разблокирование выходов в состоянии STOP.

С помощью таблицы наблюдения вы можете выполнять следующие онлайновые функции:

– контроль состояния переменных;

– изменение значений отдельных переменных;

– принудительное присваивание переменной определенного значения.

Вы можете выбрать, когда переменная должна наблюдаться или изменяться:

– начало цикла: значение считывается или записывается в начале цикла сканирования;

– конец цикла: значение считывается или записывается в конце цикла сканирования;

– переключение в STOP.

Для создания таблицы наблюдения:

1. Дважды щелкните на Add new watch table [Добавить новую таблицу наблюдения], чтобы открыть новую таблицу наблюдения (рис. 63).

2. Введите имя переменной, чтобы добавить переменную в таблицу наблюдения.

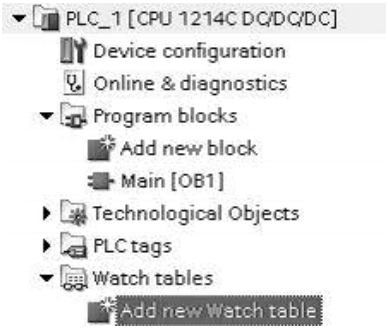


Рис. 63. Создание таблицы наблюдения за состоянием переменных ПЛК в среде TIA Portal

Для контроля переменных имеются следующие возможности:

– Monitor all [Контролировать все]: эта команда запускает контроль видимых переменных в активной таблице наблюдения.

– Monitor now [Контролировать теперь]: эта команда запускает контроль видимых переменных в активной таблице наблюдения. Таблица наблюдения выполняет контроль переменных немедленно и только один раз.

Для изменения переменных имеются в распоряжении следующие возможности:

– Modify to 0 [Изменить на 0] устанавливает значение выбранного адреса на 0;

– Modify to 1 [Изменить на 1] устанавливает значение выбранного адреса на 1;

– Modify now [Изменить сейчас] немедленно изменяет значение выбранных адресов на время одного цикла;

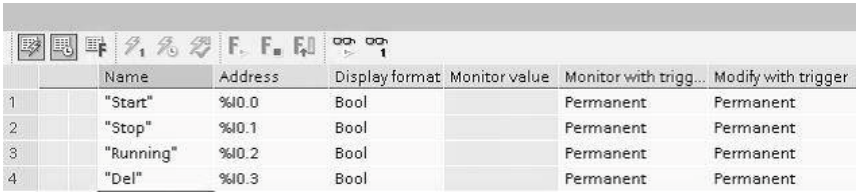
– Modify with trigger [Инициирование изменений] изменяет значение выбранных адресов. Эта функция не обеспечивает обратной связи, чтобы показать, что выбранные адреса были действительно изменены. Если требуется ответная реакция на изменения, используйте функцию Modify now [Изменить сейчас];

– Enable peripheral outputs [Разблокировать периферийные выходы] деактивизирует команду на блокировку выходов и имеется

в распоряжении только тогда, когда ЦПУ находится в состоянии STOP.

Для контроля переменных вы должны находиться в онлайновом соединении с ЦПУ.

Различные функции могут быть выбраны с помощью кнопок в верхней части таблицы наблюдения (рис. 64).



	Name	Address	Display format	Monitor value	Monitor with trigg...	Modify with trigger
1	"Start"	%I0.0	Bool		Permanent	Permanent
2	"Stop"	%I0.1	Bool		Permanent	Permanent
3	"Running"	%I0.2	Bool		Permanent	Permanent
4	"Del"	%I0.3	Bool		Permanent	Permanent

Рис. 64. Вид таблицы наблюдения за состоянием переменных ПЛК в среде TIA Portal

Введите имя переменной для контроля и выберите формат отображения из ниспадающего списка. При наличии онлайнового соединения с ЦПУ щелчок на кнопке Monitor [Контролировать] отобразит текущее значение информационной точки в поле Monitor value [Контролируемое значение].

Использование инициирования при контроле и изменении переменных ПЛК

Инициирование определяет, в какой точке цикла будут контролироваться или изменяться выбранные адреса (табл. 14).

Таблица 14

Тип инициирования значения переменных ПЛК

Тип инициирования	Описание
Постоянное	Непрерывно регистрирует данные
В начале цикла сканирования	Постоянно: постоянно регистрирует данные в начале цикла сканирования, после того как ЦПУ прочитает входы
	Однократно: регистрирует данные однократно в начале цикла сканирования, после того как ЦПУ прочитает входы

Окончание таблицы 14

Тип инициирования	Описание
В конце цикла сканирования	Постоянно: постоянно регистрирует данные в конце цикла сканирования, перед тем как ЦПУ запишет выходы
	Однократно: регистрирует данные однократно в конце цикла сканирования, перед тем как ЦПУ запишет выходы
При переходе в STOP	Постоянно: постоянно регистрирует данные при переходах ЦПУ в STOP
	Однократно: регистрирует данные однократно после перехода ЦПУ в STOP

Для изменения переменной ПЛК при заданном способе инициирования выберите начало или конец цикла.

Изменение выхода: лучшим инициирующим событием для изменения выхода является конец цикла, непосредственно перед которым ЦПУ записывает выходы.

Контролируйте значения выходов в начале цикла, чтобы определить, какое значение записано в физические выходы. Контролируйте также выходы перед тем, как ЦПУ записывает значения в физические выходы, чтобы проверить логику программы и сравнить с фактическим поведением входов и выходов.

Изменение входа: лучшим инициирующим событием для изменения входа является начало цикла, непосредственно после того, как ЦПУ считывает входы, и до того, как программа пользователя использует входные значения.

Если изменять входы в начале цикла, то нужно также контролировать значение входов в конце цикла, чтобы убедиться, что значение входа в конце цикла не изменилось с начала цикла сканирования. Если имеется разница в значениях, то пользовательская программа, возможно, записывает вход ошибочно.

Чтобы узнать, почему ЦПУ перешел в STOP, используйте способ инициирования Transition to STOP [Переход в STOP], чтобы зарегистрировать последние значения процесса.

Разблокирование выходов в состоянии STOP

Таблица наблюдения дает вам возможность записывать значения в выходы, когда ЦПУ находится в состоянии STOP. Эта функция позволяет вам проверять подключение выходов и гарантировать, что провод, подключенный к выходному контакту, инициирует сигнал высокого или низкого уровня на клемме устройства, к которой он подключен.

Вы можете изменять состояние выходов в режиме STOP, если выходы разблокированы. Если выходы заблокированы, то вы не можете их изменять в режиме STOP.

Для разблокирования изменения выходов в состоянии STOP выберите опцию Enable peripheral outputs [Разблокировать периферийные выходы] команды Modify [Изменять] в меню Online или щелкните правой клавишей мыши на соответствующей строке таблицы наблюдения.

Переход ЦПУ в режим RUN блокирует опцию Enable peripheral outputs.

Если входам или выходам присвоены принудительные значения, то ЦПУ не может разблокировать выходы в состоянии STOP. Функция присваивания принудительных значений сначала должна быть отменена.

Принудительное присваивание значений в ЦПУ

ЦПУ позволяет вам принудительно присваивать значения входам и выходам (форсировать входы и выходы), задавая адрес физического входа или выхода (I_:P или Q_:P) в таблице наблюдения и запуская форсирование.

В программе считывание физических входов перекрывается принудительно заданными значениями. Во время обработки программа использует принудительно заданные значения. Когда программа записывает физический выход, то значение выхода перекрывается принудительно заданным значением. Принудительно заданное значение появляется на физическом выходе и используется процессом.

Если входу или выходу принудительное значение присваивается в таблице наблюдения, то эти действия становятся частью программы пользователя. И теперь, если даже программирующая

программа будет закрыта, то принудительно установленные значения остаются активными в программе, исполняемой в ЦПУ, пока вы их не отмените, перейдя в режим онлайн с помощью программирующей программы и остановив функцию форсирования. Программы с принудительно установленными входами и выходами, будучи загружены в другой ЦПУ из карты памяти, будут продолжать форсировать входы и выходы, выбранные в программе.

Если ЦПУ исполняет программу пользователя из защищенной от записи карты памяти, то вы не сможете инициировать или изменить форсирование входов/выходов из таблицы наблюдения, так как вы не можете заменить значения в защищенной от записи программе пользователя. Любая попытка принудительно изменить значения, защищенные от записи, генерирует ошибку. Если вы используете карту памяти для переноса программы пользователя, то все элементы на карте памяти будут переданы в ЦПУ вместе с их принудительно установленными значениями.

Цифровые входы и выходы, используемые такими устройствами, как скоростной счетчик (HSC), широтно-импульсная модуляция (PWM) и вывод последовательности импульсов (PTO), назначаются во время конфигурирования устройства. Когда адреса цифровых входов и выходов назначаются этим устройствам, значения этих адресов не могут быть изменены функцией принудительного присваивания значений таблицы наблюдения.

ОРГАНИЗАЦИЯ СВЯЗИ КОНТРОЛЛЕРОВ С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ. ЛОКАЛЬНЫЕ И СЕТЕВЫЕ ИНТЕРФЕЙСЫ

Для компьютеров и связанных с ним устройств наиболее распространенной является задача передачи дискретных данных, как правило, в значительных количествах (не один бит). Самый распространенный способ представления данных сигналами – двоичный: например, условно высокому (выше порога) уровню напряжения соответствует логическая единица, низкому – логический ноль (возможно и обратное представление).

Интерфейс (interface) – совокупность средств и методов взаимодействия между элементами системы. Стандарт интерфейса определяет:

– механические характеристики интерфейса (разъемы и соединители);

– электрические характеристики сигналов (логические уровни);

– функциональные описания интерфейсных схем (протоколы передачи).

Протокол передачи – стандарт, определяющий поведение функциональных блоков при передаче данных. Большинство протоколов передачи работают по принципу ведущий – ведомые (Master – Slave), то есть используют одно ведущее устройство и одно или несколько ведомых устройств.

Рис. 65. Организация сетевого взаимодействия устройств по принципу «ведущий – ведомые»

Передатчик – устройство, передающее информацию в данный момент времени.

Приемник – устройство, принимающее информацию в данный момент времени.

Информационное сообщение – минимальный объем передаваемых данных, достаточный для интерпретации устройством-приемником в системе связи.

Линия связи – совокупность технических устройств и физической среды, обеспечивающая распространение сигналов от передатчика к приемнику.

Ведущее устройство (Master) – главное устройство в сети, которое может самостоятельно запрашивать данные у ведомых устройств.

Ведомое устройство (Slave) – устройство, которое не может самостоятельно инициировать передачу своих данных, а передает или принимает их только по запросу ведущего устройства сети.

Скорость передачи информации – скорость передачи данных, выраженная в количестве бит, символов или блоков, передаваемых за единицу времени. Обычно единицей измерения скорости передачи является 1 бод = 1 бит/с.

Для того чтобы избежать одновременного появления на линии связи информационных сообщений от нескольких ведомых устройств используется адресация. Каждому ведомому устройству присваивается индивидуальный адрес. Информационное сообщение, направленное от ведущего устройства, содержит адрес того ведомого устройства, которому оно предназначено. Ведомое устройство сравнивает принятый адрес со своим внутренним адресом и только в случае их совпадения реагирует на полученное сообщение и имеет право выставить ответ в линию связи. Ведущее устройство также может отправить сообщение, адресованное всем ведомым устройствам. Такое сообщение называется широковещательным, и ответ ведомых устройств на широковещательное сообщение, как правило, не предусмотрен. Широковещательное сообщение содержит широковещательный адрес, который не может быть присвоен ни одному из ведомых устройств.

Различают дуплексный, полудуплексный и симплексный режимы работы приемо-передающих устройств. В режиме дуплекс устройства могут передавать и принимать информацию одновременно.

В режиме полудуплекс устройства могут передавать или принимать информацию в каждый момент времени. В режиме симплекс передача информации всегда ведется только в одном направлении.

Виды интерфейсов связи

Различают последовательные и параллельные интерфейсы связи.

Параллельный интерфейс – для каждого бита передаваемой группы используется своя сигнальная линия (обычно с двоичным представлением), и все биты группы передаются одновременно за один квант времени.

Последовательный интерфейс использует одну сигнальную линию, и биты группы передаются друг за другом по очереди; на каждый из них отводится свой квант времени (битовый интервал).

В составе микроконтроллеров реализуют различные виды интерфейсов. Наиболее распространенные среди них:

– последовательный интерфейс SPI;

– универсальный синхронно-асинхронный последовательный интерфейс USART;

– двухпроводный последовательный интерфейс I2C;

– последовательный интерфейс CAN;

– последовательный интерфейс USB.

Интерфейс RS-232. RS-232 – это интерфейс (порт) последовательной передачи данных. В программируемых логических контроллерах используется для загрузки программ, связи с панелями оператора HMI, SCADA на ПК оператора, модулями ввода/вывода и другими ПЛК. В домашних компьютерах еще недавно RS-232 в виде COM-порта активно использовался для подключения мыши и интернет-модема. Со временем COM-порт на компьютере вытеснили более скоростные интерфейсы, например, USB. Но для большинства задач промышленной автоматизации скорости RS-232 хватает с головой, поэтому в ПЛК он еще долго будет популярен из-за простоты и надежности.

Для связи по интерфейсу RS-232 (рис. 66) используются только три провода: прием данных (Rx), передача данных (Tx) и земля (GND). Скорость передачи данных – до 115 200 бит/с. Передача данных происходит последовательно: главное устройство (Master) посылает запрос, подчиненное устройство (Slave) отвечает.

Для организации связи между ПЛК и другим устройством по RS-232 необходимо:

– соединить порты RS-232 обоих устройств кабелем типа «витая пара» длиной не более 15 м. Желательно, чтобы кабель был экранирован;

– установить на обоих устройствах одинаковые параметры RS-232: скорость, количество бит данных, количество стоповых бит, четность;

– установить на обоих устройствах одинаковый протокол передачи данных. Например, Modbus RTU;

– настроить протокол: одно из устройств сделать мастером, второе слейвом. Назначить слейву сетевой адрес;

– в мастере настроить опрос регистров (ячеек памяти) слейва и дальнейшую их программную обработку.

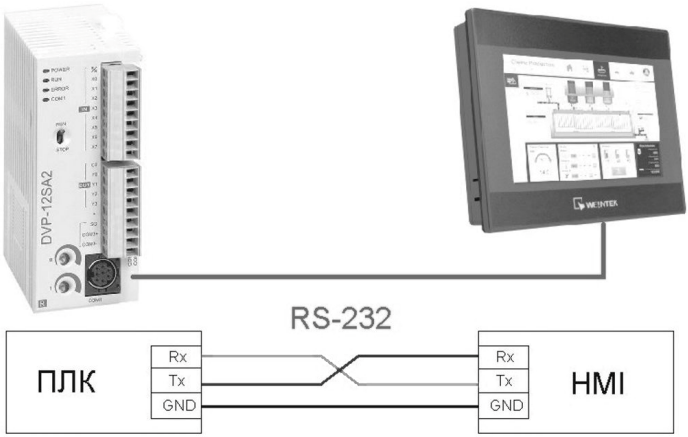


Рис. 66. Подключение устройств по интерфейсу RS-232

Подключить компьютер или ноутбук к RS-232 ПЛК можно несколькими способами (рис. 67):

– по USB, через преобразователь RS232/USB;

– через платы RS-232, которые вставляются в слоты PCI или PCI-e материнской платы ПК;

– непосредственно через COM-порт, контакты которого до сих пор размещают на всех материнских платах ПК.

Интерфейс RS-232 (или EIA-232) предназначен для организации приема-передачи данных между передатчиком или терминалом (англ. Data Terminal Equipment, DTE) и приемником или коммуникационным оборудованием (англ. Data Communications Equipment, DCE) по схеме «точка–точка».

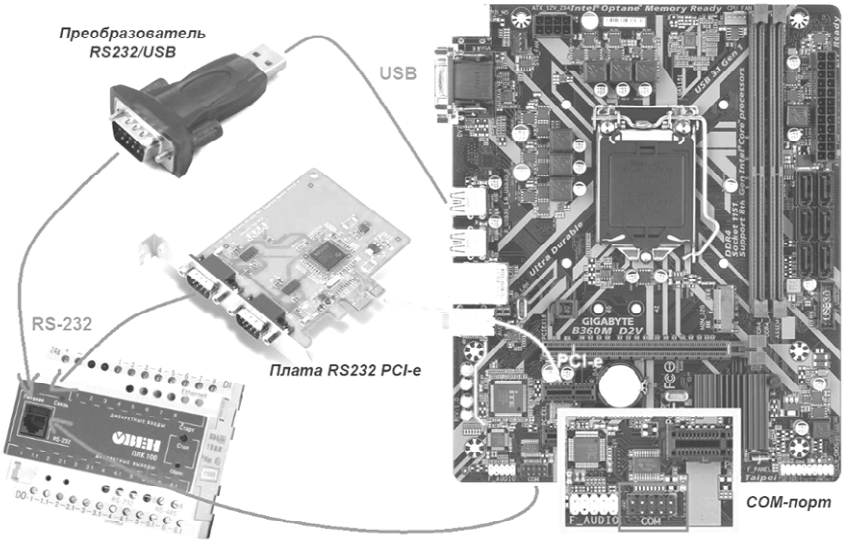


Рис. 67. Варианты подключения ПК к порту RS-232 ПЛК

Для электрических кабельных соединений используют разъемы DB9 (9-контактные) или, реже, DB25 (25-контактные) (рис. 68).

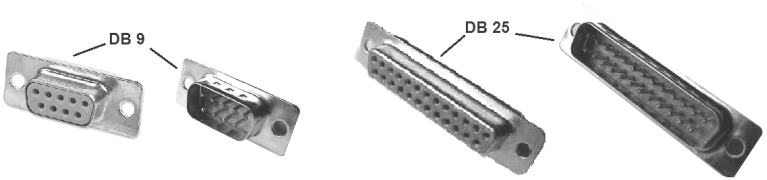


Рис. 68. Внешний вид электрических кабельных разъемов интерфейса RS-232

Назначение контактов COM-порта приведено в табл. 15.

Для успешного обмена данными ряд переменных параметров протокола должны быть заданы одинаково на стороне приемника и передатчика:

– скорость обмена данными в битах в секунду (300, 1200, 2400, 4800, 9600, 19 200, 38 400, 57 600 или другая, если она поддерживается обеими сторонами);

– количество бит данных – от 4 до 8;

– контроль четности может быть четным, нечетным или вообще отсутствовать;

– длина стоп бита может достигать одну, полторы или две длительности бита данных.

Таблица 15

Пример схемы распайки кабеля RS-232

Номер контакта	Назначение	Обозначение
1	Активная несущая	DCD
2	Прием компьютером	RXD
3	Передача компьютером	TXD
4	Готовность к обмену со стороны приемника	DTR
5	Земля	GND
6	Готовность к обмену со стороны источника	DSR
7	Запрос на передачу	RTS
8	Готовность к передаче	CTS
9	Сигнал вызова	Ri

К основным электрическим характеристикам относят (рис. 69):

– логические уровни передатчика: «0» – от +5 до +15 В, «1» – от –5 до –15 В;

– логические уровни приемника: «0» – от +3 В и выше, «1» – от –3 В и ниже;

– максимальная нагрузка передатчика: входное сопротивление приемника не менее 3 кОм.

Длина кабеля влияет на максимальную скорость передачи информации. Более длинный кабель имеет большую емкость и соответственно для обеспечения надежной передачи более низкую скорость. Большая емкость приводит к тому, что изменение напряжения одного сигнального провода может передаться на другой смежный сигнальный провод. Максимальным расстоянием обычно считается 15 м, но это не установлено в стандарте. Мы рекомендуем

использовать на расстояниях до 50 м вне зависимости от типа используемого оборудования и характеристик кабеля.

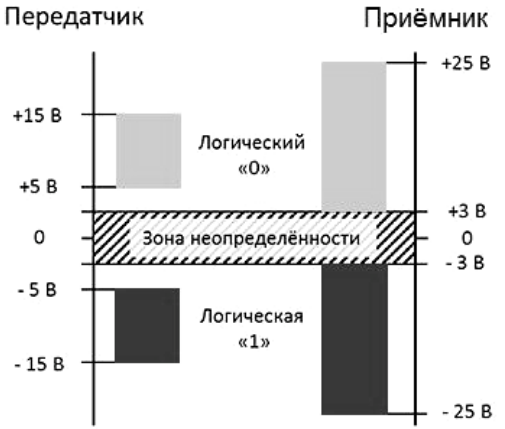


Рис. 69. Логические уровни сигналов передатчика и приемника RS-232

Назначение сигналов следующее:

1. FG – защитное заземление (экран).

2. TxD – данные, передаваемые компьютером в последовательном коде (логика отрицательная).

3. RxD – данные, принимаемые компьютером в последовательном коде (логика отрицательная).

4. RTS – сигнал запроса передачи. Активен во все время передачи.

5. CTS – сигнал сброса (очистки) для передачи. Активен во все время передачи. Говорит о готовности приемника.

6. DSR – готовность данных. Используется для задания режима модема.

7. SG – сигнальное заземление, нулевой провод.

8. DCD – обнаружение несущей данных (детектирование принимаемого сигнала).

9. DTR – готовность выходных данных.

10. RI – индикатор вызова. Говорит о приеме модемом сигнала вызова по телефонной сети.

Для двухпроводной линии связи в случае только передачи из компьютера во внешнее устройство используются сигналы SG

и TxD. Все 10 сигналов интерфейса задействуются только при соединении компьютера с модемом.

Формат передаваемых данных показан на рис. 70. Собственно, данные (5, 6, 7 или 8 бит) сопровождаются стартовым битом, битом четности и одним или двумя стоповыми битами. Получив стартовый бит, приемник выбирает из линии биты данных через определенные интервалы времени. Очень важно, чтобы тактовые частоты приемника и передатчика были одинаковыми, допустимое расхождение – не более 10 %. Скорость передачи по RS-232 может выбираться из ряда: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200, 38 400, 57 600, 115 200 бит/с.

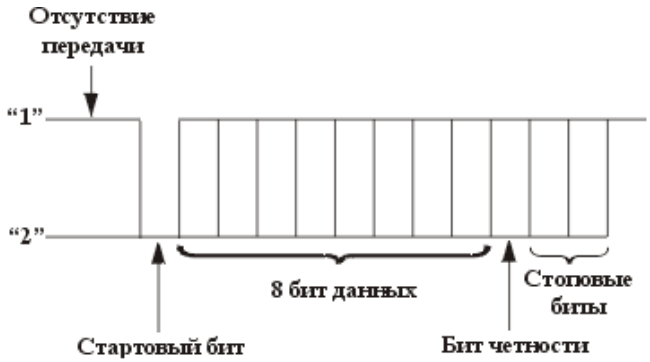


Рис. 70. Формат данных RS-232

Интерфейс «токовая петля». Интерфейс «токовая петля» используется для передачи информации с 1950-х гг. Первоначально в нем использовался ток 60 мА; позже, с 1962 г., получил распространение интерфейс с током 20 мА. В 1980-х гг. начала широко применяться «токовая петля» 4–20 мА в разнообразном технологическом оборудовании, датчиках и исполнительных устройствах средств автоматики. Популярность «токовой петли» начала падать после появления стандарта на интерфейс RS-485 (1983 г.), и в настоящее время в новом оборудовании она практически не применяется.

В передатчике «токовой петли» используется не источник напряжения, как в интерфейсе RS-485, а источник тока. По определению, ток, вытекающий из источника тока, не зависит от параметров

нагрузки. Поэтому в «токовой петле» протекает ток, не зависящий от сопротивления кабеля $R_{кабеля}$, сопротивления нагрузки $R_н$ и ЭДС индуктивной помехи $R_{инд}$ (рис. 71), а также от напряжения питания источника тока $E_п$ (рис. 72, 73). Ток в петле может измениться только вследствие утечек кабеля, которые очень малы.

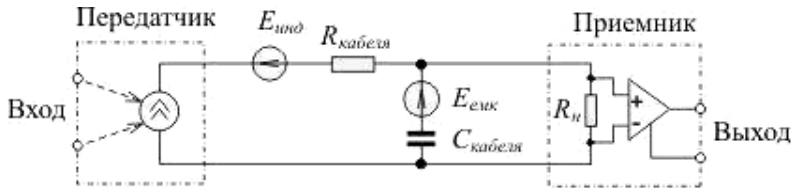


Рис. 71. Принцип действия «токовой петли»

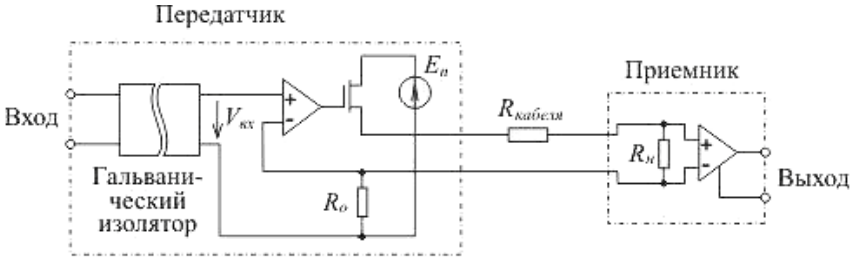


Рис. 72. Вариант построения аналоговой «токовой петли» со встроенным в передатчик источником питания

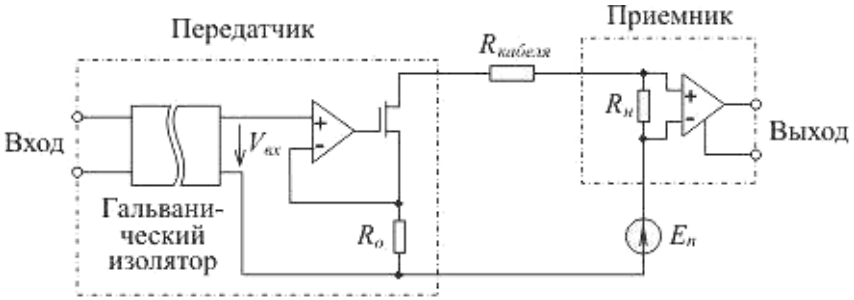


Рис. 73. Вариант построения аналоговой «токовой петли» с выносным источником питания

Емкостная наводка $E_{емк}$, ЭДС которой приложена не последовательно с источником тока, а параллельно ему, не может быть

ослаблена в «токовой петле» и для ее подавления следует использовать экранирование. В качестве линии передачи обычно используется экранированная витая пара, которая совместно с дифференциальным приемником позволяет ослабить индуктивную и синфазную помеху.

На приемном конце ток петли преобразуется в напряжение с помощью калиброванного сопротивления $R_н$. При токе 20 мА для получения стандартного напряжения 2,5 В, 5 В или 10 В используют резистор сопротивлением 125 Ом, 250 Ом или 500 Ом соответственно.

Основным недостатком «токовой петли» является ее принципиально низкое быстродействие, которое ограничивается скоростью заряда емкости кабеля $C_{кабеля}$ от источника тока. Например, при типовой погонной емкости кабеля 75 пФ/м и длине 1 км емкость кабеля составит 75 нФ. Для заряда такой емкости от источника тока 20 мА до напряжения 5 В необходимо время 19 мкс, что соответствует скорости передачи около 9 кбит/с.

Интерфейс «токовая петля» распространен в двух версиях: цифровой и аналоговой.

Аналоговая «токовая петля». Аналоговая версия «токовой петли» используется, как правило, для передачи сигналов от разнообразных датчиков к контроллеру или от контроллера к исполнительным устройствам. Применение «токовой петли» в данном случае дает два преимущества. Во-первых, приведение диапазона изменения измеряемой величины к стандартному диапазону обеспечивает взаимозаменяемость компонентов. Во-вторых, становится возможным передать сигнал на большое расстояние с высокой точностью (погрешность «токовой петли» может быть снижена до ±0,05 %). Кроме того, стандарт «токовая петля» поддерживается подавляющим большинством производителей средств промышленной автоматизации.

В варианте 4–20 мА в качестве начала отсчета принят ток 4 мА. Это позволяет производить диагностику целостности кабеля (кабель имеет разрыв, если ток равен нулю) в отличие от варианта 0–20 мА, где величина 0 мА может означать не только нулевую величину сигнала, но и обрыв кабеля. Вторым преимуществом уровня отсчета 4 мА является возможность подачи энергии датчику для его питания.

На рис. 72, 73 показаны два варианта построения аналоговой «токовой петли». На рис. 72 используется встроенный незаземленный источник питания $E_{\text{п}}$, на рис. 73 источник питания – внешний. Встроенный источник удобен при монтаже системы, а внешний удобен тем, что его можно выбрать с любыми параметрами в зависимости от поставленной задачи.

Принцип действия обоих вариантов состоит в том, что при бесконечно большом коэффициенте усиления операционного усилителя (ОУ) напряжение между его входами равно нулю и поэтому ток через резистор $R_{\text{о}}$ равен $V_{\text{вх}}/R_{\text{о}}$, а поскольку у идеального ОУ ток входов равен нулю, то ток через резистор строго равен току в петле $I = \dfrac{V_{\text{вх}}}{R_{\text{о}}}$ и, как следует из этой формулы, не зависит от сопротивления нагрузки. Поэтому напряжение на выходе приемника определяется как $IR_{\text{н}} = \dfrac{R_{\text{н}}}{R_{\text{О}}} V_{\text{вх}}$.

Достоинством схемы с операционным усилителем является возможность калибровки передатчика без подключенного к нему кабеля и приемника, поскольку вносимая ими погрешность пренебрежимо мала.

Напряжение источника $E_{\text{п}}$ выбирается таким, чтобы обеспечить работу транзистора передатчика в активном (ненасыщенном) режиме и скомпенсировать падение напряжения на проводах кабеля и сопротивлениях $R_{\text{О}}$, $R_{\text{н}}$. Для этого выбирают $E > I\left(R_{\text{О}} + R_{\text{кабеля}} + R_{\text{н}}\right) + V_{\text{нас}}$, где $V_{\text{нас}}$ – напряжение насыщения транзистора (1–2 В). Например, при типовых значениях $R_{\text{О}} = R_{\text{н}} = 500$ Ом и сопротивлении кабеля 100 Ом (при длине 1 км) получим напряжение источника питания петли 22 В; ближайшее стандартное значение равно 24 В. Отметим, что мощность, связанная с избыточным напряжением источника питания по сравнению с рассчитанным значением, будет рассеиваться на транзисторе, что особенно существенно для интегральных передатчиков, не имеющих теплоотвода.

Цифровая «токовая тепля». Цифровая «токовая петля» используется обычно в версии «0–20 мА», так как она реализуется гораздо проще, чем «4–20 мА» (рис. 74). Поскольку при цифровой передаче данных точность передачи логических уровней роли не играет,

можно использовать источник тока с не очень большим внутренним сопротивлением и низкой точностью. Так, на рис. 74 при стандартном значении напряжения питания $E_п = 24$ В и падении напряжения на входе приемника 0,8 В для получения тока 20 мА сопротивление $R_О$ должно быть равно примерно 1,2 кОм. Сопротивление кабеля сечением 0,35 мм^2 и длиной 1 км равно 97 Ом, что составит всего 10 % от общего сопротивления петли и им можно пренебречь. Падение напряжения на диоде оптрона составляет 3,3 % от напряжения источника питания, и его влиянием на ток в петле также можно пренебречь. Поэтому с достаточной для практики точностью можно считать, что передатчик в этой схеме является источником тока.

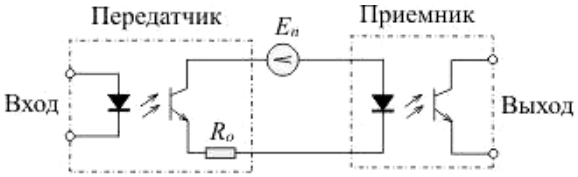


Рис. 74. Принцип реализации цифровой «токовой петли»

Как аналоговая, так и цифровая «токовая петля» может использоваться для передачи информации нескольким приемникам одновременно (рис. 75).

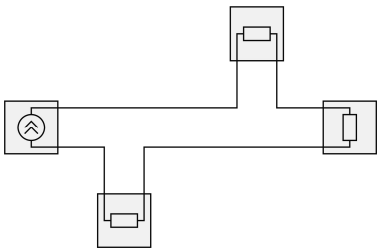


Рис. 75. Токовая петля может быть использована для передачи информации нескольким приемникам

Вследствие низкой скорости передачи информации по «токовой петле» согласование длинной линии с передатчиком и приемником не требуется.

Интерфейс USB

Интерфейс USB был разработан лидерами компьютерной и телекоммуникационной промышленности (Compaq, DEC, IBM, Intel, Microsoft, NEC и Northern Telecom) для подключения компьютерной периферии вне корпуса ПК с автоматическим автоконфигурированием (Plug&Play). Первая версия стандарта появилась в 1996 г.

Интерфейс USB представляет собой последовательную, полудуплексную, двунаправленную шину со скоростью обмена:

– USB 1.1 – 1,5 или 12 Мбит/с;

– USB 2.0 – 480 Мбит/с.

Шина позволяет подключить к ПК до 127 физических устройств. Каждое физическое устройство может, в свою очередь, состоять из нескольких логических (например, клавиатура со встроенным манипулятором-трекболом).

Кабельная разводка USB начинается с узла (host) (рис. 76). Хост обладает интегрированным корневым концентратором (root hub), который предоставляет несколько разъемов USB для подключения внешних устройств. Затем кабели идут к другим устройствам USB, которые также могут быть концентраторами, и функциональным компонентам (например, модем или акустическая система). Концентраторы часто встраиваются в мониторы и клавиатуры (которые являются типичными составными устройствами). Концентраторы могут содержать до семи «исходящих» портов.

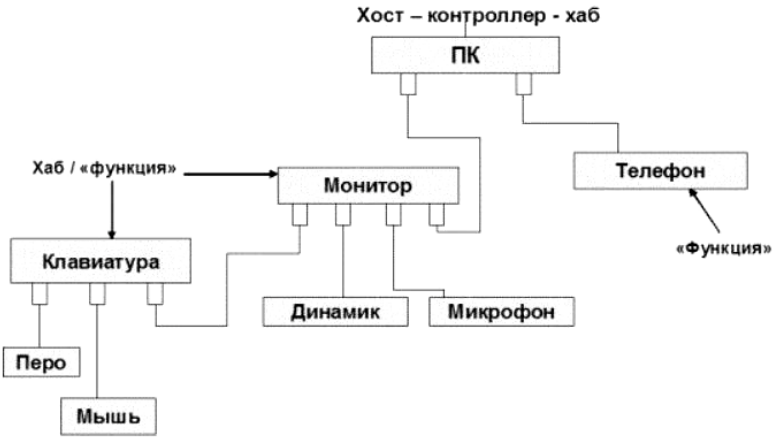


Рис. 76. Топология подключения устройств к USB

Хабы – это устройства, которые позволяют физически подключить устройства USB к шине. Они предоставляют порты для подключения, ретранслируют трафик от хост-контроллера к конечным устройствам и обратно, отслеживают состояние и физически управляют электропитанием портов. У хабов есть один восходящий (upstream) порт, – это тот порт, который подключен по направлению к хост-контроллеру, и несколько нисходящих (downstream) портов, – это порты, к которым подключаются конечные устройства. Хабы можно каскадировать, подключая к нисходящему порту хаба еще один хаб. Самый главный хаб, интегрированный с хост-контроллером, называется корневым хабом (он же – корневой концентратор или root hub).

Другими словами, можно сказать, что у хаба есть две основных задачи:

1) создать хост-контроллеру иллюзию, что он непосредственно разговаривает с подключенным к хабу устройством;

2) наблюдать за своим сегментом шины (за девайсами, подключенными к нисходящим портам), сообщать «наверх» обо всех изменениях и, если надо, – подключать и отключать питание портов.

Конечные устройства – это все те полезные устройства, которые мы подключаем к шине USB (флэшки, принтеры, мышки и т. д.)

Для передачи сигналов шина USB использует четырехпроводной интерфейс. Одна пара проводников («+5В» и «общий») предназначена для питания периферийных устройств с нагрузкой до 500 мА. Данные передаются по другой паре проводов. Для передачи данных используются дифференциальные напряжения до 3 В (с целью снижения влияния шума) и схема кодирования NRZI (что избавляет от необходимости выделять дополнительную пару проводников под тактовый сигнал).

Интерфейс USB 1.1 декларирует два режима:

– низкоскоростной подканал (пропускная способность – 1,5 Мбит/с), предназначенный для таких устройств, как мыши и клавиатуры;

– высокопроизводительный канал, обеспечивающий максимальную пропускную способность 12 Мбит/с, что может использоваться для подключения внешних накопителей или устройств обработки и передачи аудио- и видеоинформации.

Все концентраторы должны поддерживать на своих исходящих портах устройства обоих типов, не позволяя высокоскоростному

трафику достигать низкоскоростных устройств. Высокопроизводительные устройства подключаются с помощью экранированного кабеля, длина которого не должна превышать 3 м. Если же устройство не формулирует особых требований к полосе пропускания, его можно подключить и неэкранированным кабелем (который может быть более тонким и гибким). Максимальная длина кабеля для низкоскоростных устройств – 5 м. Требования устройства к питанию (диаметр проводников, потребляемая мощность) могут обусловить необходимость использования кабеля меньшей длины. Из-за особенностей распространения сигнала по кабелю число последовательно соединенных концентраторов ограничено шестью (или семью) пятиметровыми отрезками кабеля.

Хост узнает о подключении или отключении устройства из сообщения от концентратора (эта процедура называется опросом шины – bus enumeration). Затем хост присваивает устройству уникальный адрес USB (1:127). После отключения устройства от шины USB его адрес становится доступным для других устройств.

Хост опрашивает все устройства и выдает им разрешения на передачу данных (рассылая для этого пакет-маркер – Token Packet). Таким образом, устройства лишены возможности непосредственного обмена данными – все данные проходят через хост. Это условие сильно мешало внедрению интерфейса USB на рынок портативных устройств. В результате в конце 2001 г. было принято дополнение к стандарту USB 2.0 – спецификация USB OTG (On-The-Go), предназначенная для соединения периферийных USB-устройств друг с другом без необходимости подключения к хосту (например, цифровая камера и фотопринтер). Устройство, поддерживающее USB OTG, способно частично выполнять функции хоста и распознавать, когда оно подключено к полноценному хосту (на основе ПК), а когда – к другому периферийному устройству. Спецификация описывает также протокол согласования выбора роли хоста при соединении двух USB OTG-устройств.

Архитектура USB допускает четыре базовых типа передач данных между хостом и периферийными устройствами:

1. Изохронные передачи (isochronous transfers) – потоковые передачи в реальном времени, занимающие предварительно согласованную часть пропускной способности шины с гарантированным временем задержки доставки. На полной скорости (FS) можно

организовать один канал с полосой до 1,023 Мбайт/с (или два по 0,5 Мбайт/с), заняв 70 % доступной полосы (остаток можно занять и менее емкими каналами). На высокой скорости (HS) можно получить канал до 24 Мбайт/с (192 Мбит/с). Надежность доставки не гарантируется – в случае обнаружения ошибки изохронные данные не повторяются, недействительные пакеты игнорируются. Шина USB позволяет с помощью изохронных передач организовывать синхронные соединения между устройствами и прикладными программами. Изохронные передачи нужны для потоковых устройств: видеокамер, цифровых аудиоустройств (колонки USB, микрофон), устройств воспроизведения и записи аудио- и видеоданных (CD и DVD). Видеопоток (без компрессии) шина USB способна передавать только на высокой скорости.

2. Прерывания (interrupts) – передачи спонтанных сообщений, которые должны выполняться с задержкой не большей, чем требует устройство. Предел времени обслуживания устанавливается в диапазоне 10–255 мс для низкой и 1–255 мс для полной скорости. На высокой скорости можно заказать и 125 мкс. Доставка гарантирована, при случайных ошибках обмена выполняется повтор (правда, при этом время обслуживания увеличивается). Прерывания используются, например, при вводе символов с клавиатуры или передаче сообщений о перемещениях мыши. Прерываниями можно передавать данные и к устройству (как только устройство сигнализирует о потребности в данных, хост своевременно их передает). Размер сообщения может составлять 0–8 байт для низкой скорости, 0–64 байт – для полной и 0–1024 байт – для высокой скорости передачи.

3. Передачи массивов данных (bulk data transfers) – это передачи без каких-либо обязательств по своевременности доставки и по скорости. Передачи массивов могут занимать всю полосу пропускания шины, свободную от передач других типов. Приоритет этих передач самый низкий, они могут приостанавливаться при большой загрузке шины. Доставка гарантированная – при случайной ошибке выполняется повтор. Передачи массивов уместны для обмена данными с принтерами, сканерами, устройствами хранения и т. п.

4. Управляющие передачи (control transfers) используются для конфигурирования устройств во время их подключения и для управления устройствами в процессе работы. Протокол обеспечивает

гарантированную доставку данных и подтверждение устройством успешности выполнения управляющей команды. Управляющая передача позволяет подать устройству команду (запрос, возможно, с дополнительными данными) и получить на него ответ (подтверждение или отказ от выполнения запроса и, возможно, данные). Только управляющие передачи на USB обеспечивают синхронизацию запросов и ответов; в остальных типах передач явной синхронизации потока ввода с потоком вывода нет.

Высокая помехозащищенность USB обусловлена следующими факторами:

1) высокое качество сигналов, достигаемое благодаря дифференциальным приемникам/передатчикам и экранированным кабелям;

2) защита полей управления и данных CRC-кодами;

3) независимость функций от неудачных обменов с другими функциями;

4) обнаружение подключения и отключения устройств и конфигурирование ресурсов на системном уровне;

5) управление потоком для обеспечения изохронности и управления аппаратными буферами;

6) самовосстановление протокола с тайм-аутом при потере пакетов.

ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ. ПОНЯТИЕ О СЕТЯХ ПРОМЫШЛЕННОЙ АВТОМАТИЗАЦИИ. ПОМЕХОЗАЩИЩЕННОСТЬ И НАДЕЖНОСТЬ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Промышленная сеть – сеть передачи данных, связывающая различные датчики, исполнительные механизмы, промышленные контроллеры и используемая в промышленной автоматизации. Термин употребляется преимущественно в автоматизированной системе управления технологическими процессами (АСУ ТП).

Устройства используют сеть:

– для передачи данных, между датчиками, контроллерами и исполнительными механизмами;

– диагностики и удаленного конфигурирования датчиков и исполнительных механизмов;

– калибрования датчиков;

– питания датчиков и исполнительных механизмов;

– передачи данных между датчиками и исполнительными механизмами минуя центральный контроллер;

– связи между датчиками, исполнительными механизмами, ПЛК и АСУ ТП верхнего уровня;

– связи между контроллерами и системами человеко-машинного интерфейса (операторскими системами).

В промышленных сетях для передачи данных применяют:

– кабели;

– оптоволоконные линии;

– беспроводную связь (радиомодемы и Wi-Fi).

Промышленные сети могут взаимодействовать с обычными компьютерными сетями, в частности использовать глобальную сеть Internet.

Термин «полевая шина» является дословным переводом английского термина fieldbus. Термин «промышленная сеть» является более точным переводом и в настоящее время именно он используется в профессиональной технической литературе.

Промышленные сети отличаются от офисных следующими свойствами:

– специальным конструктивным исполнением, обеспечивающим защиту от пыли, влаги, вибрации, ударов;

– широким температурным диапазоном (обычно от –40 ºС до +70 ºС);

– повышенной прочностью кабеля, изоляции, разъемов, элементов крепления;

– повышенной устойчивостью к воздействию электромагнитных помех;

– возможностью резервирования для повышения надежности;

– повышенной надежностью передачи данных;

– возможностью самовосстановления после сбоя;

– детерминированностью (определенностью) времени доставки сообщений;

– возможностью работы в реальном времени (с малой, постоянной и известной величиной задержки);

– работой с длинными линиями связи (от сотен метров до нескольких километров).

Достоинства. В сравнении с подключением периферийного оборудования к контроллеру отдельными проводами промышленная сеть имеет следующие достоинства:

1. В несколько раз снижается расход на кабель и его прокладку.

2. Увеличивается допустимое расстояние до подключаемых датчиков и исполнительных устройств.

3. Упрощается управление сетью датчиков и исполнительных механизмов.

4. Упрощается модификация системы при изменении типа датчиков, используемого протокола взаимодействия, добавлении устройств ввода/вывода.

5. Позволяют дистанционно настраивать датчики и проводить их диагностику.

Недостатки:

1. При обрыве кабеля теряется возможность получать данные и управлять не одним, а несколькими устройствами (в зависимости от места обрыва и топологии сети остается возможность автономного функционирования сегмента сети и схемы управления).

2. Для повышения надежности приходится резервировать каналы связи или использовать кольцевую топологию сети.

Классификация промышленных сетей

Соединение промышленной сети с ее компонентами (устройствами, узлами сети) выполняется с помощью интерфейсов. Сетевым интерфейсом называют логическую и (или) физическую границу между устройством и средой передачи информации. Обычно этой границей является набор электронных компонентов и связанного с ними программного обеспечения. При существенных модификациях внутренней структуры устройства или программного обеспечения интерфейс остается без изменений, что является одним из признаков, позволяющих выделить интерфейс в составе оборудования.

Наиболее важными параметрами интерфейса являются пропускная способность и максимальная длина подключаемого кабеля. Промышленные интерфейсы обычно обеспечивают гальваническую развязку между соединяемыми устройствами. Наиболее распространены в промышленной автоматизации последовательные интерфейсы RS-485, RS-232, RS-422, Ethernet, CAN, HART, AS-интерфейс.

Для обмена информацией взаимодействующие устройства должны иметь одинаковый протокол обмена. В простейшей форме протокол – это набор правил, которые управляют обменом информацией. Он определяет синтаксис и семантику сообщений, операции управления, синхронизацию и состояния при коммуникации. Протокол может быть реализован аппаратно, программно или программно-аппаратно. Название сети обычно совпадает с названием протокола, что объясняется его определяющей ролью при создании сети.

Взаимодействие устройств в промышленных сетях выполняется в соответствии с моделями: клиент–сервер или издатель–подписчик (производитель–потребитель).

В модели клиент–сервер взаимодействуют два объекта. Сервером является объект, который предоставляет сервис, то есть который выполняет некоторые действия по запросу клиента. Сеть может содержать несколько серверов и несколько клиентов. Каждый клиент может посылать запросы нескольким серверам, а каждый сервер может отвечать на запросы нескольких клиентов. Эта модель удобна для передачи данных, которые появляются периодически или в заранее известное время, как, например, значения

температуры в периодическом технологическом процессе. Однако эта модель неудобна для передачи случайно возникающий событий, например, события, состоящего в случайном срабатывании датчика уровня, поскольку для получения этого события клиент должен периодически, с высокой частотой, запрашивать состояние датчика и анализировать его, перегружая сеть бесполезным трафиком.

В модели взаимодействия издатель–подписчик имеется один издатель и множество подписчиков. Подписчики сообщают издателю список тегов, значения которых они хотят получать по определенному расписанию или по мере появления новых данных. Каждый клиент может подписаться на свой набор тегов. В соответствии с установленным расписанием издатель рассылает подписчикам запрошенную информацию.

В любой модели взаимодействия можно выделить устройство, которое управляет другим (подчиненным) устройством. Устройство, проявившее инициативу в обмене, называют ведущим, главным или мастером (Master). Устройство, которое отвечает на запросы мастера, называют ведомым, подчиненным или слейвом (Slave). Ведомое устройство никогда не начинает коммуникацию первым. Оно ждет запроса от ведущего и только отвечает на запросы. Например, в модели клиент–сервер клиент является мастером, сервер – подчиненным. В модели издатель–подписчик на этапе подписки мастером является клиент, а на этапе рассылки публикаций – сервер.

Сети могут иметь топологию «звезды», «кольца», шины, смешанную.

«Звезда» в промышленной автоматизации используется редко. «Кольцо» используется в основном для передачи маркера в многомастерных сетях. Шинная топология является общепринятой, что является одной из причин применения термина «промышленная шина» вместо «промышленная сеть». К общей шине в разных местах может быть подключено произвольное количество устройств.

Промышленные протоколы CAN, PROFIBUS, Foundation fieldbus

Промышленная сеть реального времени CAN (Controller Area Network – сеть контроллеров) представляет собой сеть с общей

средой передачи данных. Это означает, что все узлы сети одновременно принимают сигналы, передаваемые по шине. Невозможно послать сообщение какому-либо конкретному узлу. Все узлы сети принимают весь трафик, передаваемый по шине. Однако CAN-контроллеры предоставляют аппаратную возможность фильтрации CAN-сообщений.

Каждый узел состоит из двух составляющих. Это собственно CAN-контроллер, который обеспечивает взаимодействие с сетью и реализует протокол, и микропроцессор (ЦПУ) (рис. 76).

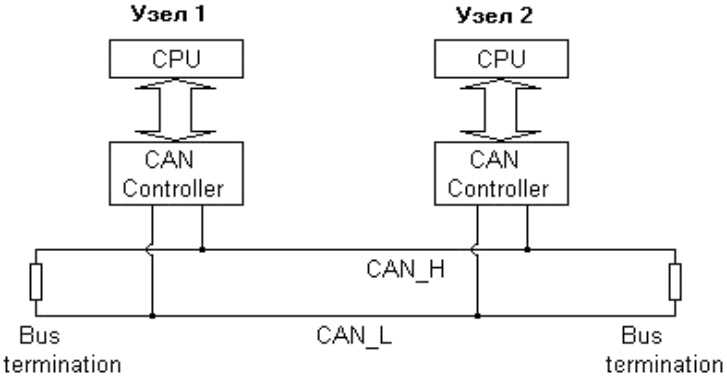


Рис. 77. Топология сети CAN

CAN-контроллеры соединяются с помощью дифференциальной шины, которая имеет две линии – CAN_H (can-high) и CAN_L (can-low), по которым передаются сигналы. Логический ноль регистрируется, когда на линии CAN_H сигнал выше, чем на линии CAN_L, логическая единица – в случае, когда сигналы CAN_H и CAN_L одинаковы (отличаются менее чем на 0,5 В). Использование такой дифференциальной схемы передачи делает возможным работу CAN сети в очень сложных внешних условиях. Логический ноль называется доминантным битом, а логическая единица – рецессивным. Эти названия отражают приоритет логической единицы и нуля на шине CAN. При одновременной передаче в шину логического нуля и единицы, на шине будет зарегистрирован только логический ноль (доминантный сигнал), а логическая единица будет подавлена (рецессивный сигнал).

Данные в CAN передаются короткими сообщениями-кадрами стандартного формата (рис. 77).

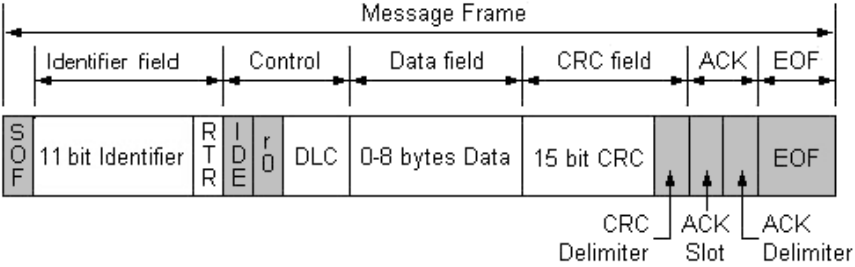


Рис. 78. Кадр данных стандарта CAN

В CAN существуют четыре типа сообщений:

– Data Frame;

– Remote Frame;

– Error Frame;

– Overload Frame.

Data Frame (кадр данных) – это наиболее часто используемый тип сообщения. Он состоит из следующих основных частей (рис. 78):

– поле арбитража (arbitration field) содержит 11-битное число-идентификатор, который определяет приоритет сообщения в случае, когда два или более узлов одновременно пытаются передать данные в сеть;

– поле данных (data field) содержит от 0 до 8 байт данных;

– поле CRC (CRC field) содержит 15-битную контрольную сумму сообщения, которая используется для обнаружения ошибок;

– слот подтверждения (Acknowledgement Slot) (1 бит), каждый CAN-контроллер, который правильно принял сообщение, посылает бит подтверждения в сеть. Узел, который послал сообщение, слушает этот бит, и в случае если подтверждение не пришло, повторяет передачу. В случае приема слота подтверждения передающий узел может быть уверен лишь в том, что хотя бы один из узлов в сети правильно принял его сообщение.

Remote Frame (кадр удаленного запроса) – это Data Frame без поля данных и с выставленным битом RTR (1 – рецессивные бит).

Основное предназначение Remote кадра – это инициация одним из узлов сети передачи в сеть данных другим узлом.

Error Frame (сообщение об ошибке) – это сообщение об ошибке. Передача такого сообщения приводит к тому, что все узлы сети регистрируют ошибку формата CAN-кадра, и в свою очередь автоматически передают в сеть Error Frame. Результатом этого процесса является автоматическая повторная передача данных в сеть передающим узлом.

Overload Frame (сообщение о перегрузке) – используется перегруженным узлом, который в данный момент не может обработать поступающее сообщение, и поэтому просит при помощи Overload-кадра о повторной передаче данных.

Физический уровень протокола CAN

Физический уровень (Physical Layer) протокола CAN определяет сопротивление кабеля, уровень электрических сигналов в сети и т. п. Существует несколько физических уровней протокола CAN (ISO 11898, ISO 11519, SAE J2411).

В подавляющем большинстве случаев используется физический уровень CAN определенный в стандарте ISO 11898. ISO 11898 в качестве среды передачи определяет двухпроводную дифференциальную линию с импедансом (терминаторы) 120 Ом (допускается колебание импеданса в пределах от 108 до 132 Ом). Физический уровень CAN реализован в специальных чипах – CAN приемопередатчиках (transceivers), которые преобразуют обычные TTL уровни сигналов используемых CAN-контроллерами в уровни сигналов на шине CAN. Наиболее распространенный CAN приемопередатчик – Phillips 82C250, который полностью соответствует стандарту ISO 11898.

Максимальная скорость сети CAN в соответствие с протоколом равна 1 Мбит/с. При скорости в 1 Мбит/с максимальная длина кабеля равна примерно 40 метрам (табл. 16). Ограничение на длину кабеля связано с конечной скоростью света и механизмом побитового арбитража (во время арбитража все узлы сети должны получать текущий бит передачи одновременно, то есть сигнал должен успеть распространится по всему кабелю за единичный отсчет времени в сети).

Таблица 16

Соотношение между скоростью передачи и максимальной длиной кабеля

Скорость передачи	Максимальная длина сети
1000 Кбит/с	40 м
500 Кбит/с	100 м
250 Кбит/с	200 м
125 Кбит/с	500 м
10 Кбит/с	6 км

Контроль ошибок – важнейший аспект работы протокола CAN. Стандарт предусматривает несколько механизмов контроля ошибок.

Во-первых, это ***контроль передачи битов*** – уровень сигнала в сети сравнивается с передаваемым для каждого бита.

Второй механизм заключается в ***использовании дополнительных битов*** (*stuffing bit*). После передачи любых пяти одинаковых битов автоматически добавляется передача бита противоположного значения. При передаче шести одинаковых битов диагностируется ошибка stuffing'a. Этот механизм используется для кодирования всех полей фреймов данных и запроса. Исключением являются только поля промежутка подтверждения, разграничителя контрольной суммы и EOF.

Стандартная процедура *проверки контрольной суммы*. Передатчик вычисляет контрольную сумму для текущего кадра и передает ее в линию. В свою очередь, приемник также вычисляет контрольную сумму для принимаемых данных и сравнивает ее с тем значением, которое было отправлено передатчиком. В случае не совпадения значений диагностируется ошибка CRC.

Также выполняется *контроль битов фрейма, которые должны иметь заранее определенное значение*. В случае если реальное значение не совпадает с тем, которое ожидается, возникает ошибка.

Благодаря всем этим механизмам, вероятность необнаружения ошибки является очень низкой.

Таким образом, в качестве основных характеристик протокола CAN можно отметить следующие:

– очень высокая надежность и защищенность;

– каждое сообщение имеет свой собственный приоритет;

– реализован механизм обнаружения ошибок;

– автоматическая повторная отправка сообщений, которые были доставлены с ошибкой;

– уже упомянутый широковещательный характер передачи данных;

– возможность присутствия нескольких ведущих (master) устройств в одной сети;

– широкий диапазон скоростей работы;

– высокая устойчивость интерфейса к помехам;

– кроме того, есть механизм обнаружения «сбойных» узлов с последующим удалением таких узлов из сети.

Промышленная сеть стандарта PROFIBUS

PROFIBUS (PROcess FIeld BUS) (читается «профибас») – открытая промышленная сеть, прототип которой был разработан компанией Siemens AG для своих промышленных контроллеров SIMATIC. На основе этого прототипа организация пользователей Profibus разработала международные стандарты, принятые затем некоторыми национальными комитетами по стандартизации. Очень широко распространена в Европе, особенно в машиностроении и управлении промышленным оборудованием.

PROFIBUS – это наиболее успешно развивающаяся открытая шина промышленного применения (Fieldbus), обладающая широким диапазоном приложений. Открытость PROFIBUS гарантирует, что устройства, приобретенные от различных поставщиков, могут осуществлять связь друг с другом без необходимости применения интерфейсов-адаптеров.

Протоколы сети PROFIBUS. Одни и те же каналы связи сети PROFIBUS допускают одновременное использование нескольких протоколов передачи данных:

– PROFIBUS DP (Decentralized Peripheral – Распределенная периферия) – протокол, ориентированный на обеспечение скоростного обмена данными между системами автоматизации (ведущими DP-устройствами) и устройствами распределенного ввода/вывода (ведомыми DP-устройствами). Протокол характеризуется минимальным временем реакции и высокой стойкостью к воздействию внешних электромагнитных полей. Оптимизирован для высокоскоростных и недорогих систем. Эта версия сети была спроектирована

специально для связи между автоматизированными системами управления и распределенной периферией. Электрически близка к RS-485, но сетевые карты используют 2-хпортовую рефлективную память, что позволяет устройствам обмениваться данными без загрузки процессора контроллера.

– PROFIBUS PA (Process automation – автоматизация процесса) – протокол обмена данными с оборудованием полевого уровня, расположенным в обычных или Ex-зонах (взрывоопасных зонах). Протокол отвечает требованиям международного стандарта IEC 61158-2. Позволяет подключать датчики и приводы на одну линейную шину или кольцевую шину.

– PROFIBUS FMS (Fieldbus message specification – спецификация сообщений полевого уровня) – универсальный протокол для решения задач по обмену данными между интеллектуальными сетевыми устройствами (контроллерами, компьютерами/программматорами, системами человеко-машинного интерфейса) на полевом уровне. Некоторый аналог промышленного Ethernet, обычно используется для высокоскоростной связи между контроллерами и компьютерами верхнего уровня и используемыми диспетчерами. Скорость до 12 Мбит/с.

Все протоколы используют одинаковые технологии передачи данных и общий метод доступа к шине, поэтому они могут функционировать на одной шине.

Сеть PROFIBUS построена в соответствии с многоуровневой сетевой моделью ISO 7498. Profibus определяет следующие уровни:

– физический уровень – отвечает за характеристики физической передачи;

– канальный уровень – определяет протокол доступа к шине;

– уровень приложений – отвечает за прикладные функции.

Физически PROFIBUS может представлять собой:

– электрическую сеть с шинной топологией, использующую экранированную витую пару, соответствующую стандарту RS-485;

– оптическую сеть на основе волоконно-оптического кабеля;

– инфракрасную сеть.

Скорость передачи по ней может варьироваться от 9,6 Кбит/с до 12 Мбит/с.

Шина PROFIBUS использует специальный двужильный кабель с разъемами DB-9 (рис. 79). Для связи нескольких узлов разъем

может соединять два кабеля. Также в нем есть переключатель для терминального резистора. Терминальный резистор должен быть включен на концевых устройствах сети – таким образом сообщается, что это первое или последнее устройство. Если на промежуточном разъеме включить резистор, то следующий за ним участок будет отключен.

Рис. 79. Кабель PROFIBUS с соединительными разъемами

Для всех версий Profibus существует единый ***протокол доступа к шине***. Данный протокол реализует процедуру доступа с помощью маркера (англ. token). Сеть Profibus состоит из ведущих (англ. master) и ведомых (англ. slave) станций. Ведущая станция может контролировать шину, то есть может передавать сообщения (без удаленных запросов), когда она имеет право на это (то есть когда у нее есть маркер). Ведомая станция может лишь распознавать полученные сообщения или передавать данные после соответствующего запроса. Маркер циркулирует в логическом кольце, состоящем из ведущих устройств. Если сеть состоит только из одного ведущего, то маркер не передается (в таком случае в чистом виде реализуется система master-slave).

На рис. 80 показана гибридная технология доступа с участием активных и пассивных узлов.

Все активные узлы (ведущие) формируют логическое маркерное кольцо, имеющее фиксированный порядок, при этом каждый

активный узел «знает» другие активные узлы и их порядок в логическом кольце (порядок не зависит от топологии расположения активных узлов на шине).

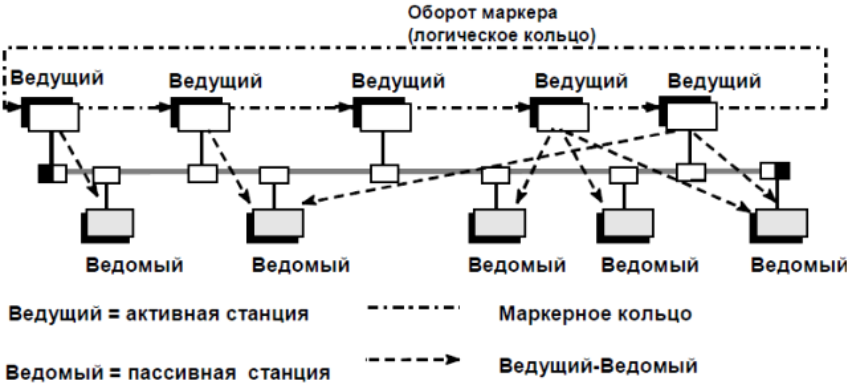


Рис. 80. Принципы технологии доступа к среде передачи информации в сетях PROFIBUS

Право доступа к каналу передачи данных, так называемый «маркер», передается от активного узла к активному узлу в порядке, определяемом логическим кольцом.

Если узел получил маркер (адресованный именно ему), он может передавать пакеты. Время, отпущенное ему на передачу пакетов, определяется временем удержания маркера. Как только это время истекает, узлу разрешается передать только одно сообщение высокого приоритета. Если такое сообщение у узла отсутствует, он передает маркер следующему узлу в логическом кольце. Маркерные таймеры, по которым рассчитывается максимальное время удержания маркера, конфигурируются для всех активных узлов.

Если активный узел обладает маркером и если для него сконфигурированы соединения с пассивными узлами (соединения «ведущее устройство–ведомое устройство»), производится опрос пассивных узлов (например, считывание значений) или передача данных на эти устройства (например, передача уставок).

Пассивные узлы никогда не принимают маркер.

HART – стандарт передачи данных через токовую петлю 4–20 мА

Один из способов автоматического контроля промышленного оборудования – это использование токовой петли 4–20 мА. Первичная переменная (PV) передается как значение тока в диапазоне 4–20 мА в двухпроводной линии с питанием датчика по тем же двум проводам. Недостаток этого метода заключается в том, что вы можете контролировать только одну переменную. Протокол Скоростного адресного доступа к удаленному преобразователю (Highway Addressable Remote Transducer, HART) дает возможность передавать больше информации по той же двухпроводной системе. Протокол HART является распространенным методом связи в промышленной автоматизации на протяжении уже многих лет.

При создании HART-протокола в 1980 г. преследовалась цель сделать его совместимым с широко распространенным в то время стандартом «токовая петля», но добавить возможности, необходимые для управления интеллектуальными устройствами. Поэтому аналоговая «токовая петля» 4–20 мА была модернизирована таким образом, что получила возможность полудуплексного цифрового обмена данными. Для этого аналоговый сигнал $A(t)$ суммируется с цифровым сигналом $D(t)$ (рис. 81) и полученная таким образом сумма передается с помощью источника тока 4–20 мА по линии связи.

Частотно-модулированный сигнал – это переменный ток амплитудой +/–0,5 мА с частотой 1200 Гц для цифровой единицы и 2200 Гц для цифрового нуля. Среднее значение этого синусоидального сигнала равно нулю, так что он не оказывает влияния на сигнал 4–20 мА и не искажает показания датчика.

Благодаря сильному различию диапазонов частот аналогового (0–10 Гц) и цифрового (1200 и 2200 Гц) сигналов они легко могут быть разделены фильтрами низких и высоких частот в приемном устройстве. При передаче цифрового двоичного сигнала логическая единица кодируется синусоидальным сигналом с частотой 1200 Гц, ноль – 2200 Гц. При смене частоты фаза колебаний остается непрерывной. Такой способ формирования сигнала называется частотной манипуляцией с непрерывной фазой. Выбор частот соответствует американскому стандарту BELL 202 на телефонные каналы связи.

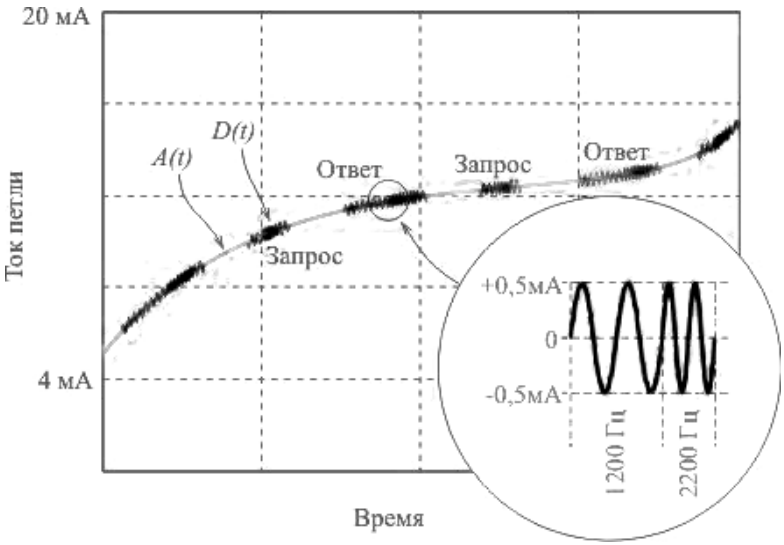


Рис. 81. Принцип формирования информационного сигнала по протоколу HART

Принцип взаимодействия устройств на физическом уровне показан на рис. 82. Сопротивление $R_н$ выбирается так же, как и в токовой петле (стандартом предусмотрена величина 230–1100 Ом) и служит для преобразования тока 4–20 мА в напряжение. Акт взаимодействия устройств инициирует контроллер. Цифровой сигнал от источника напряжения $E^{\sim}$ через конденсатор $C_{вч}$ подается в линию передачи и принимается на стороне датчика в форме напряжения в диапазоне от 400 до 800 мВ. Приемник датчика воспринимает HART-сигналы в диапазоне от 120 мВ до 2 В. Получив запрос, датчик формирует ответ, который в общем случае может содержать как аналоговый сигнал $A(t)_2$, так и цифровой ($D(t)_2$). Аналоговый сигнал обычно содержит информацию об измеренной величине, а цифровой – информацию о единицах и диапазоне измерения, о выходе величины за границы динамического диапазона, о типе датчика, имени изготовителя и т. п. Аналоговый и цифровой сигнал суммируются и подаются в линию связи в форме тока (рис. 81, 82). На стороне контроллера ток преобразуется в напряжение резистором $R_н$. Полученный сигнал подается на фильтр нижних частот с частотой среза 10 Гц и на фильтр верхних частот с частотой среза 400–800 Гц. На выходе фильтров выделяются цифровой

сигнал $D(t)_2$ и аналоговый $A(t)_2$. При использовании фильтров второго порядка погрешность, вносимая цифровым сигналом в аналоговый, составляет всего 0,01 % от 20 мА.

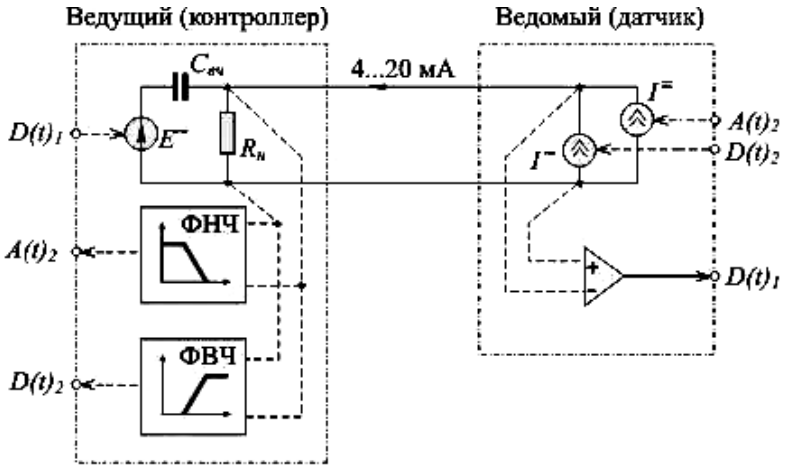


Рис. 82. Принцип работы HART-протокола на физическом уровне

Протокол HART имеет два основных режима работы: «точка-точка» и режим множественного доступа. В режиме «точка–точка» имеется одно ведущее устройство и одно ведомое. Преимущество этого режима заключается в том, что цифровые данные легко передаются по существующей линии 4–20 мА, что обеспечивает более детальный мониторинг устройства по существующей инфраструктуре сетей связи. В режиме множественного доступа к одной линии присоединяется несколько ведущих и ведомых устройств, поэтому могут передаваться только цифровые данные протокола HART в виде частотно-модулированного сигнала, а постоянный ток в линии фиксируется на уровне 4 мА. Режим множественного доступа может быть полезен, если много вынесенных устройств обменивается данными с единой системой управления, но в этом случае постоянный ток интерфейса «токовая петля» не может быть использован для непрерывного отслеживания основного измеряемого значения.

Протокол связи HART имеет следующие достоинства:

– простая настройка, сервис и техническое обслуживание устройств с поддержкой HART;

– совместимость с обычными аналоговыми полевыми устройствами и датчиками;

– открытый стандарт, доступный каждому изготовителю оборудования КИП;

– достаточно высокая помехоустойчивость.

Главным недостатком HART является то обстоятельство, что усовершенствования протокола могут производиться только в области программного обеспечения, а не в аппаратной части протокола (в связи с необходимостью поддерживать совместимость со «стареющей» технологией аналоговой аппаратуры). Следовательно, сегодня HART является медленной технологией по сравнению с другими протоколами и системами связи.

SCADA-СИСТЕМЫ. ИНТЕРФЕЙСЫ СВЯЗИ. ПРОТОКОЛ OPC

Современные интегрированные системы управления производством (рис. 83) строятся по принципу пирамиды и охватывают весь цикл работы предприятия от систем управления нижнего уровня до систем управления предприятием в целом.

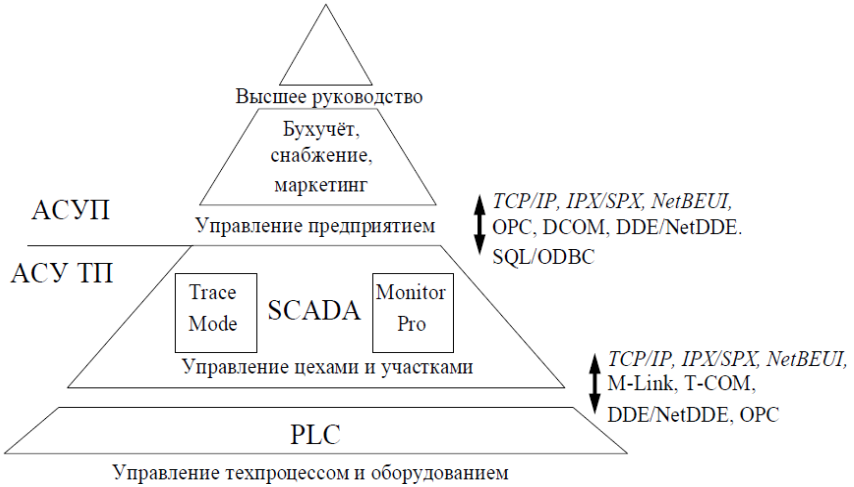


Рис. 83. Интегрированные системы управления производством

Архитектура АСУ ТП включает два уровня. Первый, самый нижний – уровень объекта управления (контроллерный) – включает различные датчики для сбора информации о ходе технологического процесса, электроприводы и исполнительные механизмы для реализации регулирующих и управляющих воздействий. Датчики поставляют информацию локальным ПЛК, которые могут выполнять следующие функции:

– сбор и обработку информации о параметрах технологического процесса;

– управление электроприводами и другими исполнительными механизмами;

– решение задач автоматического логического управления и др.

ПЛК должны гарантированно откликаться на внешние события, поступающие от объекта, за некоторое гарантированное максимально допустимое время отклика, то есть ПЛК должны работать в режиме реального времени.

На первом уровне (уровне датчиков и исполнительных механизмов) аналоговый интерфейс связи 4–20 мА в настоящее время заменяется коммуникационной технологией, объединяющей датчики, исполнительные механизмы и контроллеры в единую цифровую сеть – Fieldbus (полевая шина или промышленная сеть). Это позволяет большое число 2-, 3-, 4-проводных линий связи, идущих от множества датчиков и исполнительных механизмов к каналам ввода/вывода контроллеров, заменить на один кабель. К приборам первого уровня по этому кабелю может передаваться также и электропитание.

Второй уровень – диспетчерский пункт (ДП) – включает одну или несколько станций управления, представляющих собой автоматизированное рабочее место (АРМ) диспетчера/оператора. Станции управления предназначены для отображения хода технологического процесса и оперативного управления.

Системы оперативного диспетчерского управления и сбора данных – системы SCADA – являются неотъемлемой частью современных АСУ ТП и обеспечивают интерфейс между человеком и системой управления. SCADA-системы реализуют все основные функции визуализации измеряемой и контролируемой информации, передачи данных и команд системе контроля и управления.

Для процесса управления в современных диспетчерских системах характерны следующие особенности:

– процесс SCADA применяется в системах, в которых обязательно участие человека (оператора, диспетчера);

– процесс SCADA разработан для систем, в которых любое неправильное воздействие может привести к отказу (потере) объекта управления или даже катастрофическим последствиям;

– оператор несет, как правило, общую ответственность за управление системой, которая при нормальных условиях только изредка требует подстройки параметров для достижения оптимальной производительности;

– активное участие оператора в процессе управления происходит нечасто и в непредсказуемые моменты времени, обычно в случае наступления критических событий (отказы, нештатные ситуации и пр.);

– действия оператора в критических ситуациях могут быть жестко ограничены по времени (несколькими минутами или даже секундами).

Современная АСУ ТП обязательно должна предусматривать также информационную связь с корпоративными системами управления предприятием – уровнем АСУП (см. рис. 83, уровень АСУП). В современной англоязычной и международной терминологии принято понятие ERP-системы (Enterprise Resource Planning) – планирование ресурсов предприятия или MRP-системы (Manufacturing Resource Planning) – планирование ресурсов производства. Системы ERP (уровень «высшего руководства» на рис. 83) ориентированы на предприятие в целом, а MRP – на его технологические подразделения.

В интегрированных системах управления предприятием между системами SCADA и ERP присутствует промежуточная группа систем, называемая MES (Manufacturing Execution Systems, уровень управления предприятием на рис. 83). Эта группа возникла вследствие обособления задач, не относящихся к ранее определенным группам SCADA и ERP. К системам MES принято относить приложения, отвечающие:

– за управление производственными и людскими ресурсами в рамках технологического процесса;

– планирование и контроль последовательности операций технологического процесса;

– управление качеством продукции;

– хранение исходных материалов и произведенной продукции по технологическим подразделениям;

– техническое обслуживание производственного оборудования;

– связь систем SCADA и ERP.

Определение CDADA-системы. Основные решаемые задачи

SCADA (от англ. Supervisory Control And Data Acquisition – диспетчерское управление и сбор данных) – программный пакет,

предназначенный для разработки или обеспечения работы в реальном времени систем сбора, обработки, отображения и архивирования информации об объекте мониторинга или управления.

SCADA-системы решают следующие задачи:

1) обмен данными с «устройствами связи с объектом» (то есть с промышленными контроллерами и платами ввода/вывода) в реальном времени через драйверы;

2) обработка информации в реальном времени;

3) логическое управление;

4) отображение информации на экране монитора в удобной и понятной для человека форме (пример на рис. 84);

5) ведение базы данных реального времени с технологической информацией;

6) аварийная сигнализация и управление тревожными сообщениями;

7) подготовка и генерирование отчетов о ходе технологического процесса;

8) осуществление сетевого взаимодействия между SCADA ПК;

9) обеспечение связи с внешними приложениями (СУБД, электронные таблицы, текстовые процессоры и т. д.).

SCADA-системы предназначены:

– для более точного ведения технологического процесса, стабилизации качества продукции и уменьшения процента брака;

– уменьшения количества действий оператора, с целью концентрации его внимания на выработке более эффективных решений по управлению процессом;

– программного контроля правильности выработки команд дистанционного управления и, следовательно, минимизации количества ошибок, допускаемых операторами;

– автоматического выявления и оповещения об аварийных и предаварийных ситуациях;

– предоставления полной необходимой информации персоналу в виде различных отчетов;

– анализа факторов, влияющих на качество готовой продукции.

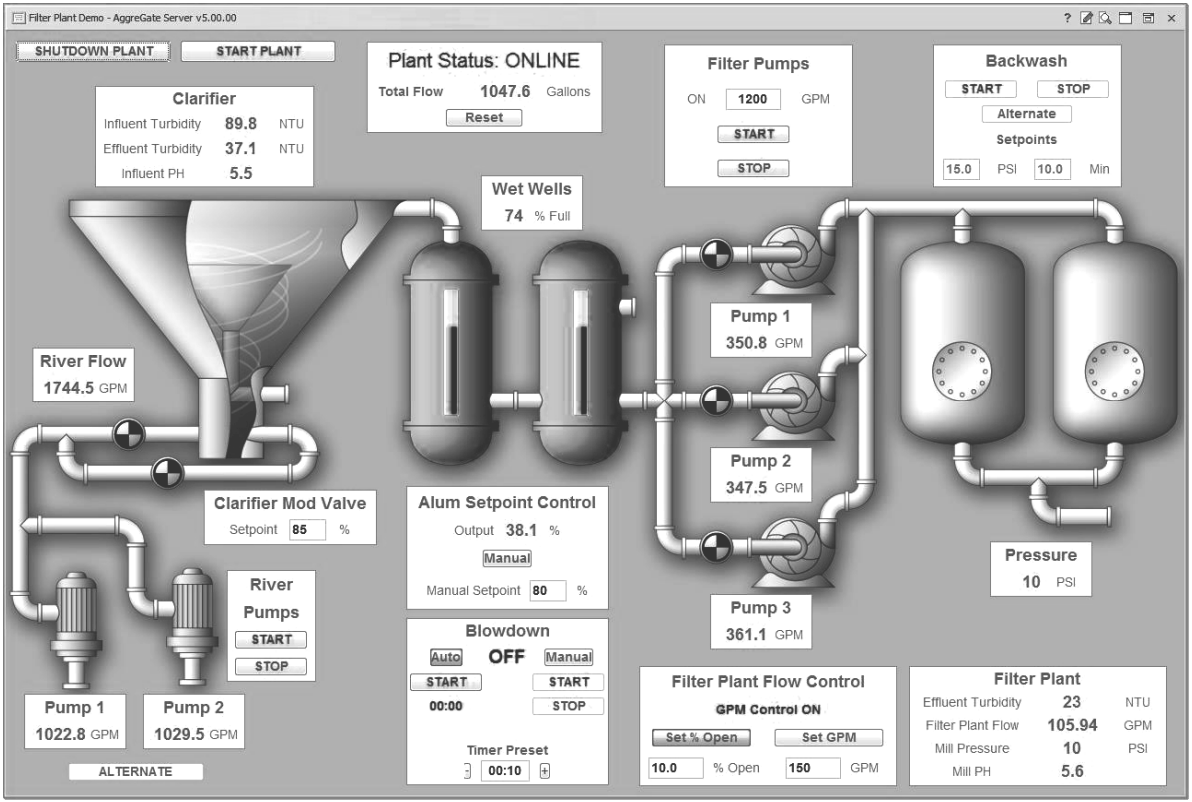


Рис. 84. Пример человеко-машинного интерфейса SCADA-системы

Термин SCADA обычно относится к централизованным системам контроля и управления всей системой, или комплексами систем, осуществляемого с участием человека. Большинство управляющих воздействий выполняется автоматически с помощью ПЛК. Непосредственное управление процессом обычно обеспечивается ПЛК, а SCADA управляет режимами работы (рис. 85). Например, ПЛК может управлять потоком охлаждающей воды внутри части производственного процесса, а SCADA-система может позволить операторам изменять уставки для потока, менять маршруты движения жидкости, заполнять те или иные емкости, а также следить за тревожными сообщениями (алармами), такими как потеря потока и высокая температура, которые должны быть отображены, записаны и на которые оператор должен своевременно реагировать. Цикл управления с обратной связью проходит через ПЛК, в то время как SCADA-система контролирует полное выполнение цикла.

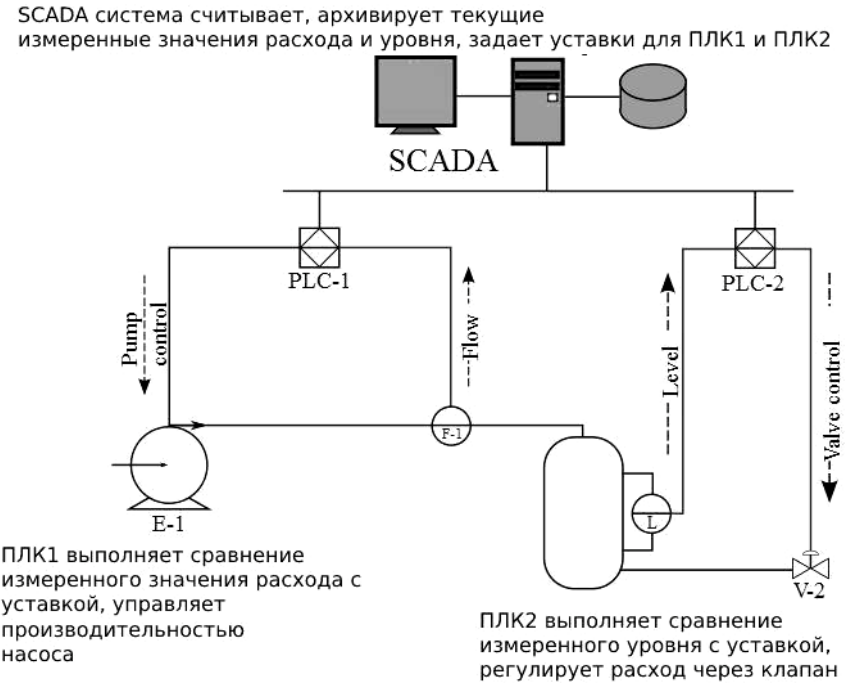


Рис. 85. Пример человеко-машинного интерфейса SCADA-системы

Сбор данных начинается на уровне ПЛК и включает показания измерительного прибора. Далее данные собираются и форматируются таким способом, чтобы оператор диспетчерской, используя человеко-машинный интерфейс (см. рис. 84), мог принять контролирующие решения – корректировать или прервать стандартное управление средствами ПЛК. Данные также могут быть записаны в архив для построения трендов и другой аналитической обработки накопленных данных.

Кроме того, определяют пять функций человека-оператора в системе диспетчерского управления:

– обучение (программирование) компьютерной системы на последующие действия;

– отслеживание результатов полуавтоматической работы системы;

– вмешательство в процесс:

- в случае критических событий, когда автоматика не может справиться с задачей управления;
- при необходимости подстройки (регулировки) параметров процесса;

– самообучение в процессе работы (получение опыта).

Основные компоненты SCADA. Архитектура SCADA-систем

Все современные SCADA-системы включают три основных структурных компонента (рис. 86):

1. Remote Terminal Unit (RTU) – удаленный терминал, подключающийся непосредственно к контролируемому объекту и осуществляющий управление в режиме реального времени. К RTU можно отнести датчики, исполнительные механизмы, устройства связи с объектом, микроконтроллеры, осуществляющие обработку информации и управление в режиме жесткого реального времени.

2. Master Terminal Unit (MTU), Master Station (MS) – диспетчерский пункт управления (терминал), который осуществляет обработку данных и управление высокого уровня и обеспечивает человеко-машинный интерфейс между человеком-оператором и системой. Реализуется в виде автоматизированного рабочего места (АРМ) оператора (технолога и т. п.).

3. Communication System (CS) – коммуникационная система (каналы связи) между RTU и MTU. Она необходима для передачи

между удаленными точками (RTU) и диспетчерским пунктом управления (MTU). В качестве коммуникационной системы можно использовать различные проводные и беспроводные каналы связи.

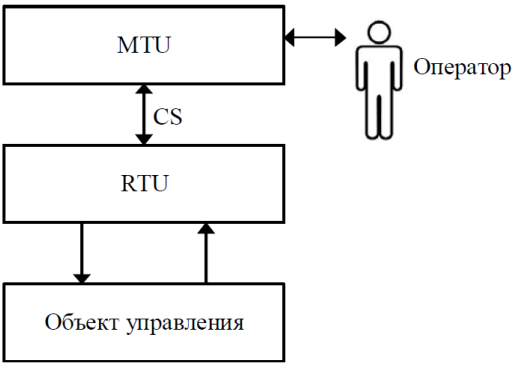


Рис. 86. Основные структурные компоненты SCADA-системы

На начальном этапе развития (80-е гг.) каждый производитель микропроцессорных систем управления разрабатывал свою собственную SCADA-программу. Такие программы могли взаимодействовать только с узким кругом контроллеров и по всем параметрам были закрытыми (отсутствие набора драйверов для работы с устройствами различных производителей и средств их создания, отсутствие стандартных механизмов взаимодействия с другими программными продуктами и т. д.).

С появлением концепции открытых систем (начало 90-х гг.) программные средства для операторских станций становятся самостоятельным продуктом.

Концепция открытых систем предполагает свободное взаимодействие программных средств SCADA с программно-техническими средствами разных производителей. Это актуально, так как для современных систем автоматизации характерна высокая степень интеграции большого количества компонент. В системе автоматизации кроме объекта управления задействован целый комплекс программно-аппаратных средств: датчики и исполнительные устройства, контроллеры, серверы баз данных, рабочие места операторов, автоматизированные рабочие места (АРМ) специалистов и руководителей и т. д. (рис. 87). При этом в одной

системе могут быть применены технические средства разных производителей.

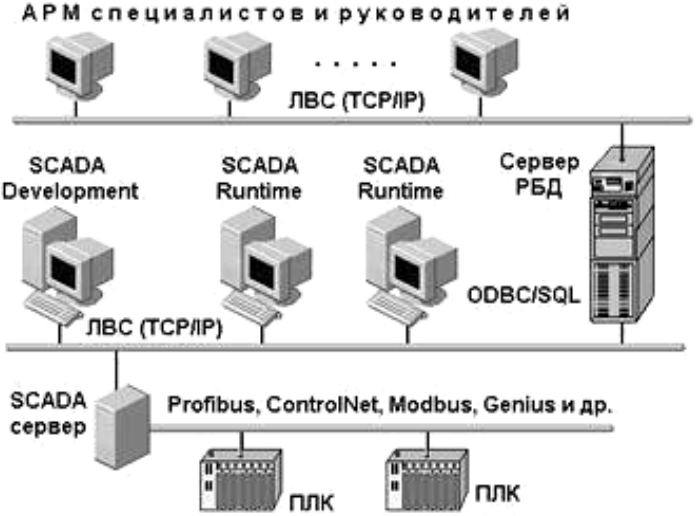


Рис. 87. Интеграция SCADA в систему управления

Одной из первых задач, поставленных перед разработчиками SCADA, стала задача организации многопользовательских систем управления, то есть систем, способных поддерживать достаточно большое количество АРМ-пользователей (клиентов). В результате появилась клиент-серверная технология или архитектура. Клиент-серверная архитектура характеризуется наличием двух взаимодействующих самостоятельных процессов – клиента и сервера, которые, в общем случае, могут выполняться на разных компьютерах, обмениваясь данными по сети.

Клиент-серверная архитектура (рис. 88) предполагает, что вся информация о технологическом процессе от контроллеров собирается и обрабатывается на сервере ввода/вывода (сервер базы данных), к которому по сети подключаются АРМ-клиенты.

Под станцией-сервером в этой архитектуре следует понимать компьютер со специальным программным обеспечением для сбора и хранения данных и последующей их передачи по каналам связи оперативному персоналу для контроля и управления технологическим

процессом, а также всем заинтересованным специалистам и руководителям. По определению сервер является поставщиком информации, а клиент – ее потребителем. Таким образом, рабочие станции операторов/диспетчеров, специалистов, руководителей являются станциями-клиентами.

Рис. 88. Клиент-серверная архитектура SCADA-систем

Сервер разделен на пять серверов, в соответствии с пятью задачами, которые он может выполнять:

– сервер ввода/вывода;

– сервер трендов;

– сервер тревог;

– сервер отчетов;

– сервер синхронизации времени.

Для небольших проектов все модули могут исполняться на одном компьютере. В проектах с большим количеством переменных модули можно распределить на несколько компьютеров в разных сочетаниях.

SCADA-программы имеют в своем составе два взаимозависимых модуля: Development (среда разработки проекта) и Runtime (среда исполнения). В целях снижения стоимости проекта эти модули могут устанавливаться на разные компьютеры. Например, станции оператора, как правило, являются узлами Runtime (или

View) с полным набором функций человеко-машинного интерфейса. При этом хотя бы один компьютер в сети должен быть типа Development. На таких узлах проект разрабатывается, корректируется, а также может и исполняться.

Графический интерфейс

Качество отображения информации на мнемосхемах определяется характеристиками графических возможностей пакетов. К ним можно отнести графический редактор, возможность создания объемных изображений, наличие библиотек и разнообразие графических заготовок и готовых объектов, богатство инструментария, многообразие динамических свойств элементов мнемосхем, форматы импортируемых изображений, наличие инструментария для создания растровых рисунков, наличие и возможности многооконных режимов и т. п.

При создании компонентов операторских интерфейсов (например, мнемосхем, см. пример на рис. 84) разработчику приходится использовать графические объекты, представляющие собой технологические аппараты (колонны, емкости, теплообменники и т. д.), участки трубопровода и такие устройства, как клапаны, насосы, электродвигатели, контроллеры, компьютеры и т. д. Как правило, это сложные объекты, полученные объединением множества простых объектов или рисунки типа Bitmap.

Создание каждого из этих объектов требует большого времени и может значительно затянуть разработку проекта. Для ускорения работы над проектом практически все SCADA-пакеты предлагают разработчику библиотеки готовых объектов, включающие сотни и тысячи графических компонентов. Теперь нет необходимости рисовать объект, если подобный объект есть в библиотеке. Достаточно открыть библиотеку объектов щелчком по соответствующей иконке инструментария, выбрать раздел, затем – объект и вставлять его в любые окна разрабатываемого интерфейса. Операция вставки готового объекта занимает всего несколько секунд.

Разработчику надо лишь выбрать требуемый объект из библиотеки, вставить его в графическую страницу и в появившийся на экране диалог ввести имя/имена переменной/переменных.

SCADA-система Simatic WinCC

SCADA-система WinCC разработана компанией Siemens.

В настоящее время этот программный продукт занимает первое место в Европе среди SCADA-систем и третье место в мире. В 1999 г. появилась пятая версия этой системы. Она базируется на операционных системах Windows 95/98/NT, является открытой и масштабируемой.

В основной комплект поставки WinCC входят следующие опции:

– Alarm Logging – для подготовки, отображения, квитирования и архивирования сообщений;

– User Administrator – для управления доступом к ресурсам WinCC;

– Text Library – позволяет создавать библиотеку соответствий между словами для переключения языков;

– Report Designer – встроенный генератор отчетов;

– Global Scripts – редактор, с помощью которого можно писать С-функции для обработки событий;

– Tag Logging – система архивирования данных. Совместно с редактором предоставляются средства для табличного и графического отображения значений в базе данных;

– Graphics Designer – редактор для рисования мнемосхем.

Система сообщений. Сообщения предоставляют оператору информацию о рабочих и аварийных состояниях технологического процесса. Сообщения информируют оператора о критических ситуациях на самых ранних стадиях их возникновения, поэтому позволяют сократить время простоя.

В процессе проектирования определяются события процесса, которые будут инициировать появление соответствующих сообщений. Например, событием может быть установка определенного бита в контроллере системы автоматизации или превышение предельных значений параметров (выход за уставку значения параметра).

Система сообщений состоит из компонента проектирования и компонента исполнения:

1. Компонентом проектирования системы сообщений является редактор Alarm Logging [Регистрация аварийных сообщений]. В редакторе Alarm Logging [Регистрация аварийных сообщений] определяются события, при которых появляются соответствующие

сообщения, а также содержание сообщений. В графическом редакторе Graphic Designer [Графический дизайнер] есть специальный графический объект WinCC Alarm Control [Окно отображения сообщений WinCC], в котором могут отображаться сообщения, сконфигурированные в редакторе Alarm Logging [Регистрация аварийных сообщений].

2. Компонентом исполнения системы сообщений является Alarm Logging Runtime [Система исполнения регистрации аварийных сообщений]. При исполнении проекта Alarm Logging Runtime [Система исполнения регистрации аварийных сообщений] отвечает за выполнение определенных задач контроля. Система исполнения также осуществляет управление операциями вывода сообщений и квитирования отображаемых сообщений.

Сообщения отображаются в табличном виде в WinCC Alarm Control [Окне отображения сообщений WinCC]. Примерный вид такого сообщения, отображаемого на экране сенсорной панели оператора, показан в следующей табл. 17. Таким образом, использование диагностических сообщений способствует более быстрому определению места возникновения возможных сбоев и ошибок в системе управления, а, следовательно, скорейшему их устранению.

Таблица 17

Примерный вид сообщения на экране панели оператора

№	Время	Дата	Сообщение
4	12:50:24:590	24.08.2018	Уровень воды в котле превысил максимально допустимый

В общем, система диагностических сообщений среды WinCC включает в себя два типа сообщений:

– системные сообщения (system-defined alarms);

– пользовательские сообщения (user-defined alarms).

Системные сообщения отображают информацию о состоянии ПЛК и о возможных ошибках сетевого соединения ПЛК и ЧМИ панели оператора.

Пользовательские сообщения (пример в табл. 17) предназначены для отображения информации о состоянии объекта управления. Настройка отображения пользовательских сообщений выполняется разработчиком в процессе разработки и создания микропроцессорной программы управления. Пользовательские сообщения подразделяются на дискретные и аналоговые.

Дискретные сообщения могут отображать информацию о событиях дискретного характера в управляющей системе, например в случае рассматриваемой системы управления водогрейным котлом, о включении/выключении клапанов наполнения и сливного клапана, включении/выключении электронагревательного элемента, старте и остановке процесса автоматического управления и т. д.

Аналоговые сообщения обычно отображают информацию о выходе за допустимые пределы аналоговых величин в объекте управления, о выходе уровня воды в котле или температуры воды за допустимые верхнее или нижнее значения.

В зависимости от степени важности отображаемой информации диагностические сообщения могут требовать подтверждения реакции оператора (alarms with acknowledgment) и не требовать такого подтверждения (alarms without acknowledgment). Подтверждение оператора означает в данном случае, что оператор увидел на экране панели управления данное сообщение и определенным образом на него среагировал.

Каждое диагностическое сообщение, возникшее в микропроцессорной системе управления в процессе работы, характеризуется своим состоянием, или, другими словами, имеет свой статус. Возможные состояния сообщений перечислены в следующих табл. 18, 19.

Таблица 18

Статус сообщений, не требующих подтверждения оператора

Статус сообщения	Описание
Входящее (incoming)	Условие для возникновения данного сообщения выполнено
Исходящее (outgoing)	Условие для возникновения данного сообщения более не выполняется

Таблица 19

Статус сообщений, требующих подтверждения оператора

Статус сообщения	Описание
Входящее Incoming	Условие для возникновения данного сообщения выполнено
Исходящее, без подтверждения Outgoing, not acknowledged	Условие для возникновения данного сообщения более не выполняется, при этом оператор не выполнил подтверждения
Исходящее, подтвержденное впоследствии Outgoing, subsequently acknowledged	Условие для возникновения данного сообщения более не выполняется, оператор выполнил подтверждение после того, как условие уже перестало выполняться
Входящее, подтвержденное оператором Incoming, acknowledged	Условие для возникновения данного сообщения выполнено, оператор выполнил подтверждение
Исходящее, подтвержденное своевременно Outgoing, acknowledged first	Условие для возникновения данного сообщения более не выполняется, при этом оператор выполнил подтверждение, когда данное условие выполнялось

Также, в зависимости от степени важности, все возможные сообщения подразделяются на классы сообщений. Класс сообщения определяет то, каким образом данное сообщение будет отображаться на панели оператора. Например, сообщение «Уровень воды в котле превысил верхнюю отметку» может быть отнесено к классу «Предупреждения» [Warning], отображение данного сообщения будет происходить на белом фоне и не требовать подтверждения оператора. С другой стороны, сообщение «Уровень воды превысил максимально допустимую границу» следует отнести к классу «Ошибка» [Error], соответствующему сбоям и аварийным ситуациям в работе системы управления. Отображение сообщения «Ошибка» на панели оператора может происходить на красном фоне в режиме мерцания и обязательно будет требовать подтверждения реакции оператора. В системе диагностических сообщений

среды WinCC существуют классы сообщений: предопределенные [Predefined alarm classes] и пользовательские [Custom alarm classes].

К предопределенным классам сообщений относятся:

– «Предупреждение» [Warning]. Сообщения данного класса отображают информацию о штатных состояниях и операциях контролируемого технологического процесса. Не требуется подтверждение оператора;

– «Ошибка» [Error]. Отображается информация о критических и опасных, аварийных состояниях контролируемого процесса. Требуется обязательное подтверждение оператора;

– «Системное сообщение» [System]. Предопределенные системные сообщения о системных событиях ПЛК и панели оператора.

Для создания и настройки диагностических сообщений в проекте следует открыть панель настройки сообщений ЧМИ [HMI alarms], расположенную в дереве проекта, в ветви, относящейся к соответствующей панели оператора. Панель настройки сообщений ЧМИ, внешний вид которой показан на рис. 89, содержит вкладки [Discrete alarms], [Analog alarms] для создания и настройки дискретных и аналоговых пользовательских сообщений соответственно, вкладку [Alarm classes] – для создания и настройки способов отображения различных классов сообщений, а также вкладку [Alarm groups] – для группировки сообщений по группам.

Как видно из рис. 89, для каждого из предопределенных классов сообщений можно настроить способ отображения сообщения на экране панели оператора, а именно: цвет фона сообщения в статусе «Входящее», цвет фона этого же сообщения в статусе «Входящее подтвержденное» и т. д.

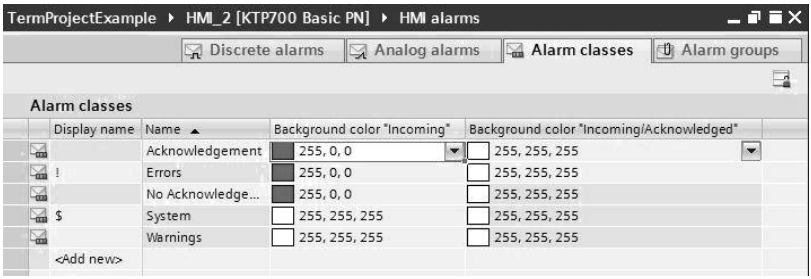


Рис. 89. Внешний вид панели настройки сообщений [HMI alarms] в среде разработки TIA Portal

Последовательность создания и настройки дискретного пользовательского сообщения включает следующие шаги:

1. В памяти ПЛК в таблице тегов создать переменную типа Int, в которой будут содержаться биты-триггеры всех дискретных сообщений. При этом для каждого бита-триггера создать отдельно взятую переменную типа Bool. Пример создания такой триггерной переменной показан в табл. 20. В данном случае переменная AlarmTagPLC типа Word имеет размер 16 бит, два бита из которых – нулевой и первый – заняты переменными типа Bool, являющимися триггерными битами дискретных событий.

2. В памяти ЧМИ-панели (в дереве проекта это пункт [HMI→HMI tags→Show all tags]) создать переменную AcknowledgeTad типа Int, в которой будут содержаться биты подтверждения реакции оператора для соответствующих событий.

3. Открыть вкладку редактора дискретных сообщений [Discrete alarms] на панели сообщений ЧМИ.

4. Для создания нового пользовательского сообщения выполнить двойной щелчок левой клавишей мыши на ячейке с надписью <Add new> (<Добавить новое>).

5. Для настройки созданного таким образом нового сообщения следуют открыть в нижней части экрана панель инспектора свойств сообщения [Properties]. Для этого следует нажать правой клавишей мыши на созданном сообщении и во всплывающем меню выбрать пункт [Properties]. Далее в открывшемся инспекторе свойств для данного сообщения следует определить отображаемый на экране текст сообщения – в поле [Alarm text], числовой идентификатор сообщения – в поле [ID], класс сообщения – в поле [Alarm class]. При необходимости, в поле [Tooltip→Text] следует ввести текст дополнительной подсказки, который будет отображаться в процессе выполнения программы при выводе данного сообщения на панель оператора.

6. В полях [Trigger→Tag] и [Trigger→Bit] нужно указать переменную и соответствующий номер бита – триггера данного сообщения.

7. В случае если для данного события выбран класс события «Ошибка», [Error], то в поле Acknowledgement нужно указать переменную и номер бита подтверждения реакции на данное событие. Эта переменная может располагаться как в памяти панели

ЧМИ, так и в памяти ПЛК. В нашем случае это переменная AcknowledgeTad типа Int в памяти панели ЧМИ.

Таблица 20

Расположение триггерной переменной и триггерных битов дискретных сообщений в памяти ПЛК

Name	Data Type	Logical Address	Comment
AlarmTagPLC	Int	%MW28	Триггерная переменная для дискретных сообщений
trigBitNoWater	Bool	%M28.0	Триггерный бит сообщения – нет воды в котле
trigBitMaxWaterLevel	Bool	%M28.1	Триггерный бит сообщения – уровень воды выше максимального

Создание и настройка аналоговых сообщений выполняется на вкладке редактора аналоговых сообщений [Analog alarms] на панели сообщений ЧМИ. Последовательность создания и настройки аналогового пользовательского сообщения включает следующие шаги:

1. Открыть вкладку редактора аналоговых сообщений [Analog alarms] на панели сообщений ЧМИ.

2. Для создания нового пользовательского сообщения выполнить двойной щелчок левой клавишей мыши на ячейке с надписью <Add new> («Добавить новое»).

3. Для настройки нового сообщения следуют открыть в нижней части экрана панель инспектора свойств сообщения [Properties]. Для этого следует нажать правой клавишей мыши на созданном сообщении и во всплывающем меню выбрать пункт [Properties]. Далее в открывшемся инспекторе свойств для данного сообщения следует определить отображаемый на экране текст сообщения – в поле [Alarm text], числовой идентификатор сообщения – в поле [ID], класс сообщения – в поле [Alarm class]. При необходимости, в поле [Tooltip→Text] следует ввести текст дополнительной

подсказки, который будет отображаться в процессе выполнения программы при выводе данного сообщения на панель оператора.

4. В поле [Trigger→Tag] нужно указать переменную целочисленного (Int) или вещественного (Real) типа, изменение значения которой будет определять возникновение данного события. Кроме того, в поле [Limit] следует указать опорное значение [Value], относительно которого система будет отслеживать изменения триггерной переменной, а также условие возникновения сообщения [Mode]. В качестве условия [Mode] может быть выбран один из двух возможных вариантов: [High limit violation] (выше максимального значения) либо [Low limit violation] (ниже минимального значения). Так, например, если требуется сообщать пользователю о возрастании контролируемой температуры свыше определенного максимального значения T_{max}, то в поле [Trigger→Tag] нужно указать переменную «*T*» в памяти ПЛК, в которой хранится значение температуры в градусах Цельсия, а в поле [Limit] указать переменную либо константу T_{max}, содержащую максимальное допустимое значение температуры, и условие [High limit violation]. Кроме того, для исключения влияния шумов и случайных колебаний, характерных для аналоговых величин, можно также ввести для данного сообщения зону нечувствительности [Deadband], задав для этого соответствующий режим нечувствительности в поле [Deadband→Mode] и ширину зоны нечувствительности [Deadband→Value] в процентах опорного значения [Limit→Value].

Для отображения сообщений на экране ЧМИ панели в процессе выполнения программы следует использовать компонент [Alarm window] (окно отображения сообщений). Компонент [Alarm window] не отображается постоянно и не привязан к какой-либо отображаемой пользовательской странице (экрану). Отображение окна сообщений [Alarm window] происходит в момент возникновения того или иного сообщения и продолжается до тех пор, пока выполняется условие для данного сообщения (в случае сообщения класса [Warning]) либо пока сообщение не было подтверждено оператором (в случае сообщения класса [Error]). Для настройки элемента [Alarm window] следует открыть так называемый глобальный экран [Global screen], находящийся в дереве проекта в ветви [Project→HMI→Screen management→Global screen]. В случае если по умолчанию там отсутствуют компоненты [Alarm window]

для отображения различных классов пользовательских и системных сообщений, нужно перетащить компонент [Alarm window] на глобальный экран со вкладки [Toolbox→Controls], где он отображен в виде пиктограммы  . Перетащив компонент [Alarm window] и расположив его на глобальном экране, далее, нажав на нем правой клавишей мыши и выбрав во всплывающем меню пункт [Properties], следует открыть редактор его свойств и на многочисленных вкладках, задавая соответствующие значения полей, выполнить настройки поведения и отображения компонента [Alarm window], а именно: указать классы сообщений, который будут отображаться на данном компоненте, задать его размеры и положение на экране ЧМИ панели, цветовую гамму, размер шрифта, наличие либо отсутствие кнопок подтверждения сообщений либо отображения подсказок и т. д.

Система архивирования и регистрации

Назначением системы архивирования значений процесса является сбор, обработка и архивирование данных, поступающих от промышленной установки. Полученные таким образом данные могут использоваться для выявления ключевых управленческих и технологических критериев, которые бы позволяли оценить работу установки.

В режиме исполнения значения процесса, которые необходимо архивировать, собираются, обрабатываются, а затем сохраняются в архивной базе данных. В режиме исполнения можно вывести и посмотреть текущие или архивные значения процесса в виде тренда или в виде таблицы. Кроме того, архивные значения процесса можно распечатать в виде журнала.

Конфигурирование параметров архивирования осуществляется в редакторе Tag Logging [Регистрация тегов]. В этом редакторе конфигурируются архивы значений процесса и вторичные архивы, определяются циклы сбора и архивирования данных, а также выбираются те значения процесса, которые необходимо архивировать.

В Graphics Designer [Графическом дизайнере] можно сконфигурировать элементы управления ActiveX для отображения данных процесса в режиме исполнения. Выводить данные на экран можно в виде трендов и таблиц.

При конфигурировании в Report Designer [Дизайнере отчетов] определяется, в каком виде архивные данные процесса выводятся в журнал регистрации данных. Значения процесса в журнале регистрации данных могут быть представлены в табличной форме или в виде тренда.

В архивировании значений процесса участвуют следующие подсистемы WinCC:

– Automation system [Система автоматизации] (AS): сохраняет значения процесса, которые передаются WinCC с помощью коммуникационных драйверов;

– Data manager [Менеджер данных] (DM): обрабатывает значения процесса и возвращает их в систему архивирования через теги процесса;

– Archive system [Система архивирования]: обрабатывает полученные значения процесса (например, формирует среднее значение). Метод обработки зависит от того, каким образом сконфигурирован архив;

– Runtime database [База данных системы исполнения] (DB): хранит архивируемые значения процесса.

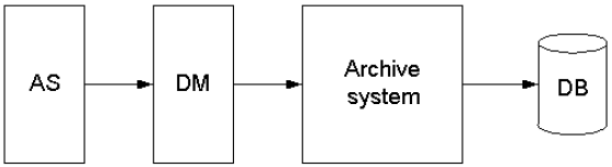


Рис. 90. Структура программного модуля архивирования значений процесса

Администрирование и идентификация пользователей микропроцессорной системы в панели оператора Simatic

При проектировании человеко-машинного интерфейса микропроцессорной системы управляющие элементы панели оператора могут быть защищены от несанкционированного использования с помощью паролей. Например, в случае рассматриваемой системы управления работой водогрейного котла, такими элементами на панели оператора могут быть управляющие кнопки запуска («Старт») и остановки («Стоп») автоматического режима управления, цифровые поля ввода значений верхнего и нижнего уровня

воды, максимальной и минимальной температуры воды в котле (рис. 94). После этого важные параметры и настройки могут быть изменены только уполномоченным персоналом.

Принцип действия модуля «Администратор пользователя». Модуль User Administrator («Администратор пользователя») выполняет назначение и управление авторизациями доступа пользователя. Если оператору присвоена соответствующая авторизация, он получает доступ к системе WinCC. Авторизации присваиваются каждому отдельному пользователю. При этом может быть присвоено до 999 различных уровней доступа. Авторизации пользователя могут назначаться в процессе работы модуля Run-Time.

Когда пользователь входит в систему, редактор User Administrator (Администратор пользователя) проверяет, зарегистрирован ли пользователь. Это применимо как к системе конфигурирования, так и к системам Run-Time всех редакторов. Если пользователь не зарегистрирован, он не имеет права на вход в систему. В самых крайних случаях это может означать, что он не может получить разрешение даже на вызов или просмотр различных данных.

Если зарегистрированный пользователь производит вызов функций, Администратор пользователя выполняет проверку, есть ли у пользователя авторизация на доступ к этим функциям. Если авторизации доступа нет, Администратор пользователя отказывает пользователю в доступе к выбранным функциям.

Система безопасности основана на полномочиях, группах пользователей и пользователях. Если в процессе работы системы требуется выполнить определенное действие с каким-либо элементом управления, защищенным паролем, например, нажать на экране панели кнопку «Старт» или ввести новое значение для максимальной температуры воды в котле, то ЧМИ-панель сначала потребует авторизации пользователя. Для этого отображается диалоговое окно регистрации, в котором вводится имя пользователя и пароль (рис. 91). После этого оператор может работать с управляющими элементами, для которых у него есть необходимые полномочия.

Для назначения пользовательских авторизаций Администратор пользователя подразделяется на две компоненты:

– *система конфигурирования модуля Администратор пользователя*: система конфигурирования модуля User Administrator (User Administrator CS) работает с пользователями. При входе в систему

нового пользователя система назначает пароли и записывает авторизации пользователя в таблицу;

– *система Run-Time Администратора пользователя*: основной задачей системы Run-Time Администратора пользователя (User Administrator RT) является контроль над входами в систему и авторизациями доступа. Это относится как к уровню конфигурирования, так и к уровню Run-Time.

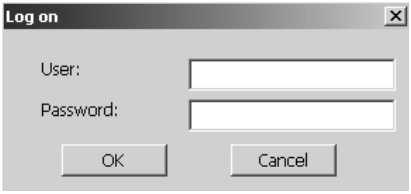


Рис. 91. Внешний вид диалогового окна регистрации пользователя

Администратор пользователя (User Administrator) выполняет следующие задачи:

– загрузка и изменение пользователей и групп;

– назначение и управление авторизациями пользователя;

– структурирование защиты от доступа;

– выборочная защита от неавторизованного доступа к отдельным системным функциям;

– отмена регистрации пользователя после перерыва в управлении (длину перерыва можно устанавливать).

На стадии разработки визуального интерфейса панели оператора и создания программы разработчик программы формирует группы пользователей в таблице Groups и соответствующие им полномочия – в таблице Authorizations на панели User groups [группы пользователей], вид которой показан на рис. 92. По умолчанию в любом проекте обычно включены минимум две группы пользователей: Administrator group [Администраторы] и Users [Пользователь ПЛК]. Кроме того, по умолчанию также следующие полномочия:

– User administration [Администрирование пользователей];

– Monitor [Просмотр] – отображение и просмотр значений параметров процесса на панели;

– Operate [Выполнение операций] – выполнение операций, например, нажатие управляющей кнопки «Старт», ввод новых значений максимальной и минимальной температуры и т. д.

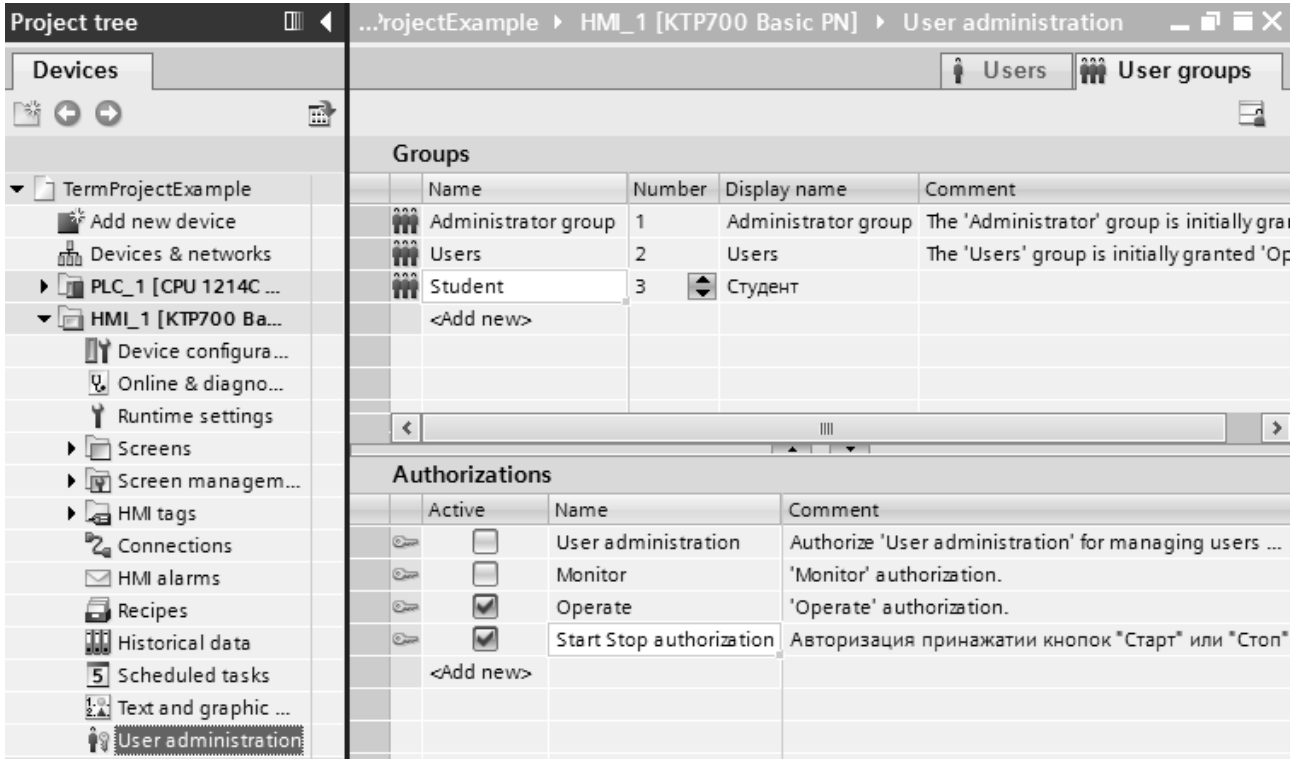


Рис. 92. Панель настройки групп пользователей [Groups] и полномочий [Authorizations]

При необходимости, разработчик системы может добавить к указанным группам пользователей и полномочиям нужное число дополнительных групп пользователей.

Настройка системы администрирования пользователей выполняется в несколько этапов. Основные этапы настройки:

1. Добавить требуемые группы.
2. Выбрать необходимые права доступа для групп.
3. Добавить пользователей и определить для них соответствующие регистрационные имена и пароли. Свойства группы (права доступа) копируются для каждого добавляемого в эту группу нового пользователя. Поэтому рекомендуется помещать пользователя, для которого необходимы определенные права доступа в ту группу, для которой установлены соответствующие права.
4. Выберите для разных пользователей различные права доступа. Здесь так же можно установить время, после которого выполняется автоматический выход пользователя из системы для защиты системы от несанкционированного доступа.

Так, например, как видно на рис. 92, в проект добавлена группа пользователей Student [Студент], для которой, помимо предлагаемых по умолчанию системных прав доступа, добавлено дополнительно право доступа Start Stop authorization [Авторизация при нажатии кнопок Старт, Стоп]. Пользователь, относящийся к группе Student [Студент], таким образом, будет наделен правами доступа Operate и Start Stop authorization.

Далее, как видно из рис. 93, на панели Users [Пользователи] в таблице Users [Пользователи] добавлены пользователи с именами Ivanov, Petrov, Federer, Nadal, Djokovich, относящиеся к группе Student. Для каждого из них введен свой пароль. Каждый из этих пользователей будет наделен правами доступа Operate и Start Stop authorization. Кроме того, в системе всегда присутствует пользователь Administrator, наделенный по умолчанию всеми системными правами доступа: User administration, Monitor, Operate.

Каждый пользователь включается ровно в одну группу. Вводить пользователей и назначать им пароли могут следующие лица:

– проектировщик на стадии разработки системы и написания программы;

– администратор на панели оператора в процессе выполнения программы;

– пользователь с полномочиями управления пользователями на панели оператора в процессе выполнения программы.

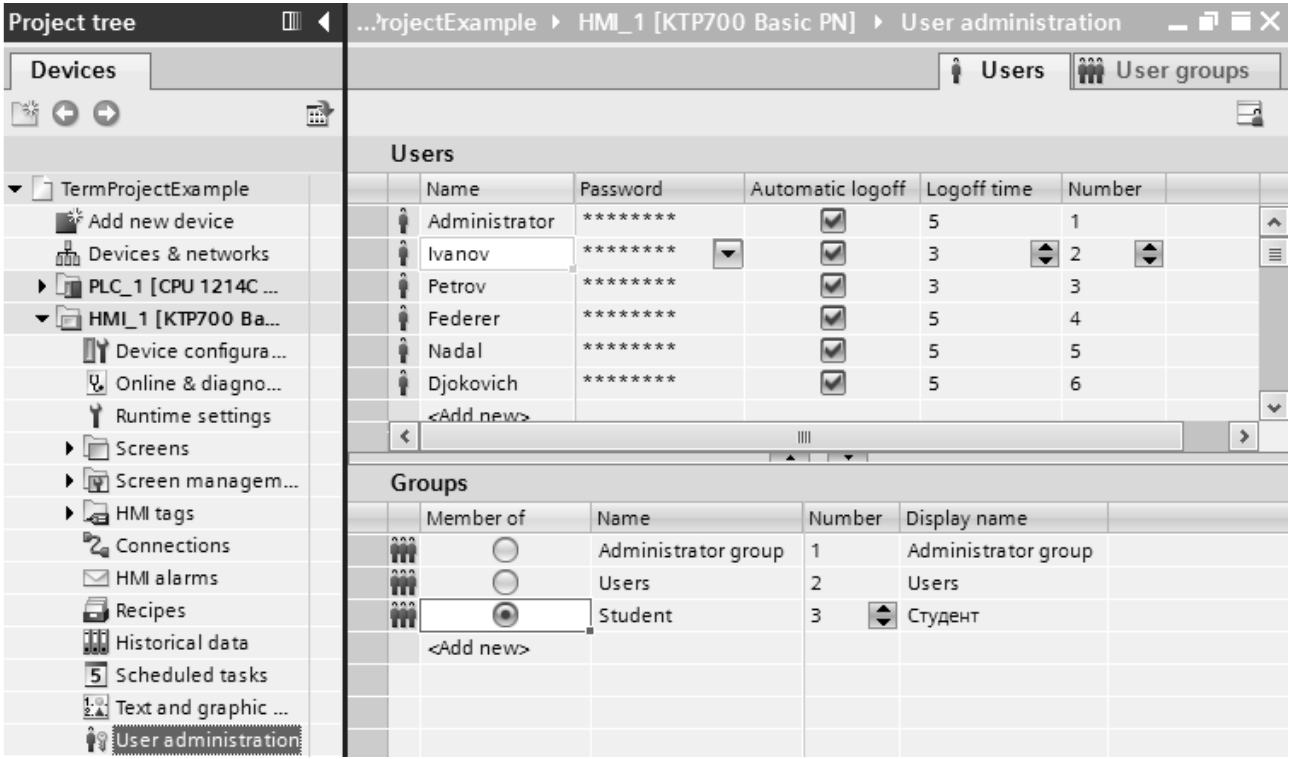


Рис. 93. Панель настройки пользователей [Users]

В процессе работы системы на реальном объекте, если, например, оператор по фамилии Петров уволится с предприятия, администратор системы (например, главный инженер предприятия) сможет удалить его из числа пользователей прямо с помощью панели оператора, не задействуя для этого компьютер и не перепрограммируя панель оператора.

Для каждого пользователя в системе может быть запроектировано время окончания сеанса (поля Automatic logoff и Logoff time в таблице Users на рис. 93). Если время между двумя любыми действиями пользователя превышает указанное число минут, то пользователь автоматически снимается с регистрации. Чтобы продолжить управлять элементами, которым назначен пароль, пользователь должен снова зарегистрироваться.

Далее, для каждого управляющего элемента на панели оператора следует также задать соответствующее право доступа. На рис. 94 показано: чтобы задать право доступа для кнопки «Пуск» на панели оператора, следует в инспекторе свойств данной кнопки в нижней части экрана на вкладке Properties→General→Security [Свойства→Основные→Безопасность] выбрать следует в поле Runtime→Security [Безопасность в режиме исполнения программы] выбрать для данной кнопки одно из ранее заданных в системе прав доступа, например, Start Stop authorization. Таким образом, нажать на кнопку «Пуск» сможет только тот пользователь, который относится к группе пользователей, наделенных данным правом доступа, то есть к группе Student [Студент]. При нажатии на кнопку появится диалоговое окно регистрации пользователя (рис. 91) и только после правильного ввода имени пользователя и пароля произойдет нажатие данной кнопки. Для других пользователей, не наделенных правом доступа Start Stop authorization, данная кнопка будет заблокирована.

Компонент User view [Обзор пользователей] используется для отображения пользователей на панели оператора. Данный компонент отображает следующую информацию о пользователях:

– User [Пользователь];
– Password [Пароль];
– Group [Группа];
– Logoff time [Время окончания сеанса].

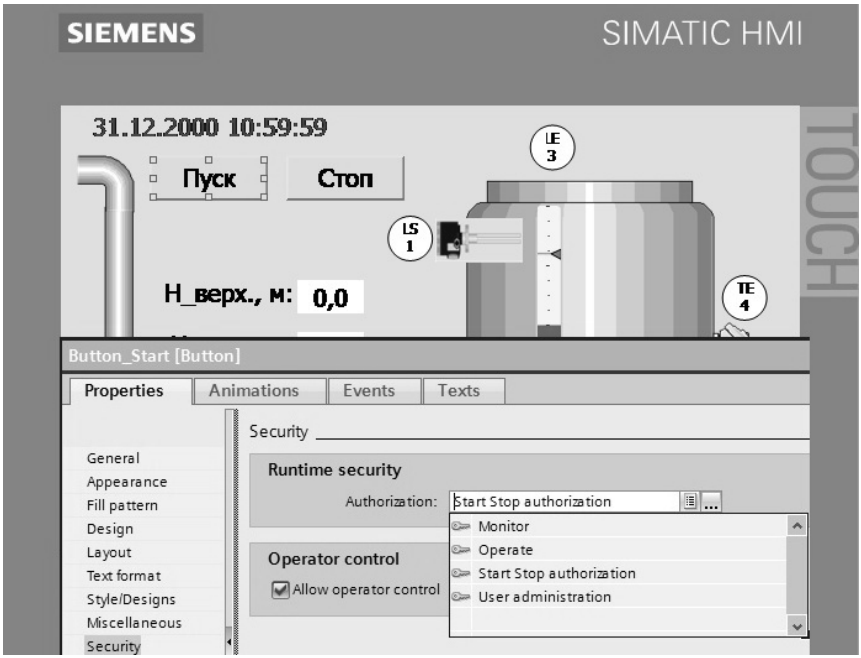


Рис. 94. Настройка авторизации для кнопки «Старт» на панели оператора

Подсистема рецептов в SCADA-системе Simatic WinCC

Рецепты – это комбинации программных переменных (тегов) для определенной цели. Цель использования рецептов заключается в совместной передаче данных в ПЛК. Это включает синхронизацию между операторской панелью (ОП) и ПЛК.

Используя пример картотеки (рис. 95), определим термины «рецепт» и «запись», так как они важны для дальнейшего понимания процесса.

Рецепт соответствует отдельному ящику каталога (то есть «грейпфрут» или «лимон»). Для каждого ящика определены ссылочные поля (теги) на конкретные рецепты. Рецепт задает структуру данных. Эта структура не может быть изменена впоследствии с операторского терминала.

Записи соответствуют карточкам в каждом ящике каталога («напиток», «сок» и «нектар»). Запись содержит конкретные значения для рецепта. Записи создаются, изменяются и удаляются в операторском

терминале. Кроме того, они хранятся в операторском терминале, что экономит память PLC.

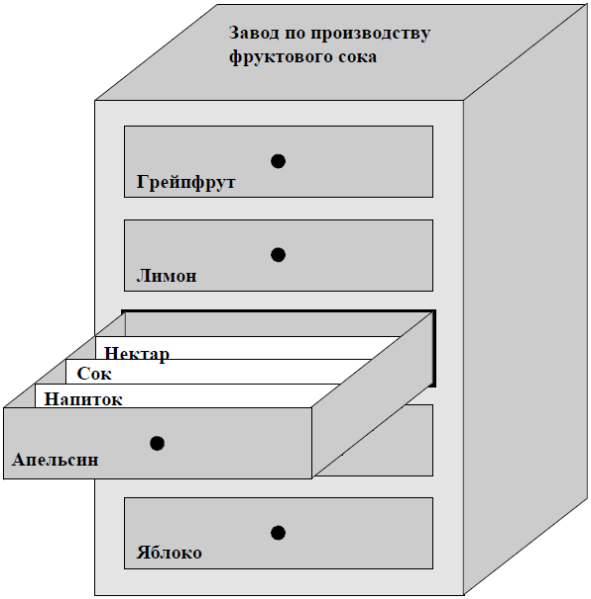


Рис. 95. Пример рецепта и записи на аналогии картотеки

Пример использования рецепта – его применение на разливочной станции комплекса по производству фруктового сока (рис. 96). Одна и та же разливочная станция используется для производства апельсинового напитка, апельсинового сока и апельсинового нектара. Соотношение смесей в каждом случае разное, но ингредиенты одни и те же.

Предположим, что есть ***рецепт «Смесь»***, представляющий собой структуру данных как в табл. 21.

Обозначения тегов orange, flavoring и т. д. – это так называемые *имена полей*. Имена полей отображаются также в операторском терминале. Таким образом, например, тег orange может быть идентифицирован как тег, обозначающий компонент смеси апельсин. Пропорции смешивания разные для каждого напитка, но ингредиенты всегда идентичны.

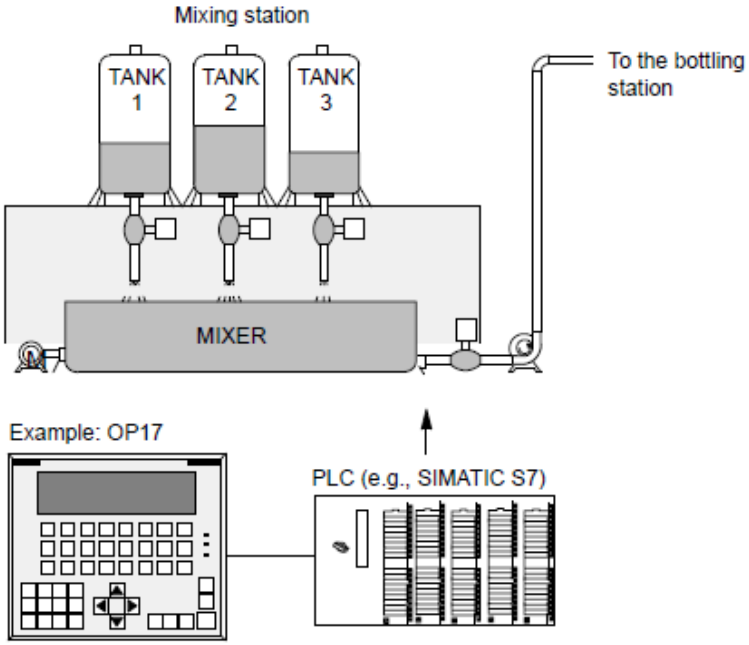


Рис. 96. Схематическое изображение оборудования линии приготовления фруктового сока

Таблица 21

Структура данных рецепта «Смесь»

Переменная	Тип данных	Комментарий
orange	Real	Требуемый объем концентрата апельсинового, л
water	Real	Требуемый объем воды, л
sugar	Int	Требуемая масса сахара, кг
flavoring	Int	Требуемая масса ароматизатора, кг

Рецепт состоит из серии записей рецепта. Записи содержат значения ингредиентов для различных напитков (табл. 22).

Таблица 22

Структура данных рецепта «Смесь»

Ингредиенты	Название напитка		
	Апельсиновый напиток	Апельсиновый сок	Апельсиновый нектар
orange	90	95	70
water	10	5	30
sugar	1.5	0.5	1.5
flavoring	0.2	0.1	0.4

На операторской панели тегам, назначенным полям ввода, присваиваются значения и хранятся в памяти операторской панели. Вместе эти значения образуют одну запись данных рецепта. Все записи данных хранятся в памяти операторской панели. Только запись данных, активная в текущий момент времени, сохраняется в памяти ПЛК. Это экономит место в памяти ПЛК.

Ниже перечислены основные шаги для конфигурирования рецепта:

1. Определение структуры рецепта: назначьте теги в структуре рецепта. Теги связываются с полями записей. Определите имя рецепта. Это имя используется для выбора рецепта в проекте и в операторском терминале.

2. Настройка тегов рецепта: в Simatic HMI можно задать следующие опции:

– Synchronize Tags (Синхронизировать теги) – эта опция указывает, что данные записи читаются из PLC или носителя и записываются в тег или читаются из тегов, сконфигурированных для рецепта. Это устанавливает связь между тегами, сконфигурированными в рецепте и тегами экранных форм. При загрузке записи значения записываются в теги, использующиеся на экранных формах;

– Tags Offline (Теги отключены) Если данная опция также включена, введенные значения записываются только в теги и в PLC не передаются. В противном случае, введенные значения передаются прямо на PLC.

3. Определите носитель записей операторского терминала. Область хранения записей подлежит конфигурированию. В зависимости от целевого устройства существуют следующие возможности:

– любой путь любого диска;

– встроенная Falsh-память;

– плата памяти, (PC-плата).

4. Задайте синхронизацию загрузки. Можно задать режимы загрузки записей в PLC с синхронизацией или без.

5. Создание экранной формы рецепта. Сконфигурируйте одну или более экранную форму для создания, хранения и загрузки записей в операторский терминал.

Редактирование записей в операторской панели.

Записи в операторском терминале могут редактироваться в таблицах или экранных формах.

Редактирование в форме таблицы. Для редактирования записей предусмотрено окно редактирования рецепта. Оно предоставляет простой и быстрый способ работы с записями и в основном используется для редактирования записей в небольших рецептах. Значения, вводимые на операторском терминале, не передаются непосредственно в ПЛК.

Редактирование в экранных формах рецепта. Разработчик проекта может использовать экранные формы рецепта для настройки интерфейса пользователя, для редактирования записей и визуальной отладки системы с использованием графики экранных форм и специальных шаблонов ввода записей.

Этот метод обычно используется при обработке средних и больших записей в автономном режиме, в окне просмотра рецепта. Значения, введенные с операторского терминала, записываются в теги, но непосредственно в ПЛК не передаются.

Обзор стандарта OPC

Технология связывания и внедрения объектов для систем промышленной автоматизации OPC (Open Platform Communications) предназначена для обеспечения универсального механизма обмена данными между датчиками, исполнительными механизмами, контроллерами, устройствами связи с объектом и системами представления технологической информации, оперативного диспетчерского управления, а также системами управления базами данных.

Стандарт OPC разработан международной организацией OPC Foundation, членами которой являются более 400 фирм, работающих в области средств автоматизации и измерительной техники. Главной целью стандарта OPC явилось обеспечение возможности совместной работы (интероперабельности) средств автоматизации, *функционирующих на разных аппаратных платформах*, *в разных промышленных сетях* и *производимых разными фирмами*.

Главной целью стандарта OPC явилось обеспечение возможности совместной работы (интероперабельности) средств автоматизации, функционирующих на разных аппаратных платформах, в разных промышленных сетях и производимых разными фирмами. До разработки OPC стандарта SCADA-пакет нужно было адаптировать к каждому новому оборудованию индивидуально. Существовали длинные списки «поддерживаемого оборудования», очень сложной была техническая поддержка. При модификации оборудования нужно было вносить изменения во все драйверы, каждый из которых поддерживал протокол обмена только с одной клиентской программой. Число таких драйверов доходило до сотен.

Реализовав поддержку OPC-клиента, разработчики SCADA-систем избавились от необходимости поддерживать сотни драйверов для различных устройств, а производители оборудования, добавив OPC-сервер, обрели уверенность в том, что их продукт может применяться пользователями любых SCADA-систем. Производители аппаратных средств, пользуясь спецификацией OPC, имеют возможность разрабатывать OPC-сервер для обеспечения единственного и наиболее общего способа организации доступа к данным и передачи в адрес приложений-клиентов различных производителей программного обеспечения для промышленной автоматизации.

Обычно технологию OPC применяют для обмена данными между контроллерами и SCADA-системой. OPC состоит из двух частей: OPC клиента и OPC сервера. Программное обеспечение OPC сервера через драйверы устройств по полевым шинам опрашивает различные устройства. Программное обеспечение OPC клиента обычно встроено в SCADA-систему и предназначено для получения данных с OPC-сервера.

OPC DA сервер обеспечивает обмен данными (запись и чтение) между клиентской программой (обычно SCADA-системой) и конечными устройствами (рис. 97). Данные в OPC представляют

собой переменную задачи с некоторыми свойствами. Переменная может быть любого типа, допустимого в OLE: различные целые и вещественные типы, логический тип, строковый, дата, массив и т. д. Свойства могут быть обязательными, рекомендуемыми и пользовательскими.

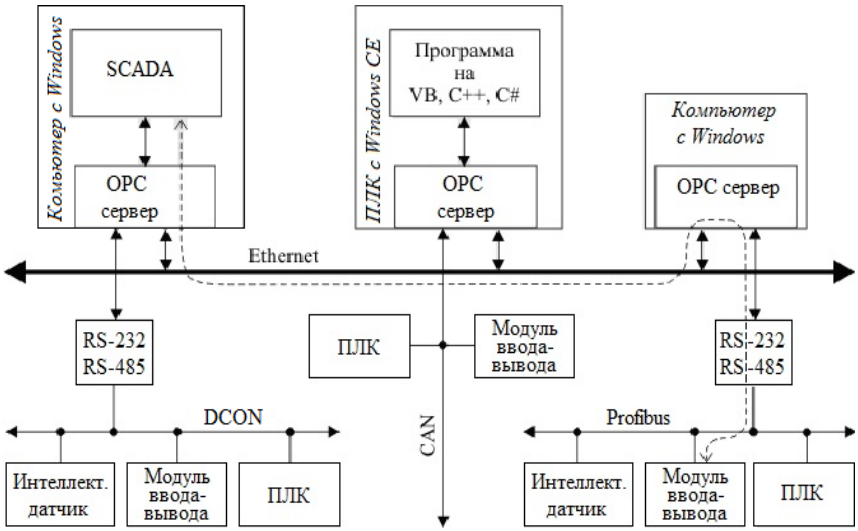


Рис. 97. Пример применения OPC технологии для сетевого доступа к данным в системах автоматизации

Обязательные свойства:

− текущее значение переменной, ее тип и права доступа (чтение и/или запись);

− качество переменной зависит от выхода измеряемой величины за границы динамического диапазона, отсутствии данных, ошибки связи и других параметров. Обычно принимает значения: хорошее/плохое/неопределенное и дополнительная информация;

− метка времени сообщает о времени, когда переменная получила данное значение;

− частота опроса переменной OPC-сервером задает время обновления значения переменной;

− описание переменной, которое содержит информацию для пользователя о том, что представляет собой эта переменная;

Дополнительно могут быть указаны необязательные свойства: диапазон изменения значения, единица измерения и другие пользовательские параметры.

Для чтения данных из OPC сервера можно использовать различные режимы:

– синхронный – клиент посылает запрос серверу и ждет от него ответ;

– асинхронный – клиент отправляет запрос и сразу же переходит к выполнению других задач. Сервер после обработки запроса посылает клиенту уведомление и тот забирает предоставленные данные;

– режим подписки – сервер отсылает клиенту только те теги, которые изменились. Для того чтобы шум данных не был принят за их изменение, вводится понятие «мертвой зоны», которая слегка превышает максимально возможный размах помехи.

– режим обновления данных – клиент вызывает одновременное чтение всех активных тегов. Активными называются все теги, кроме обозначенных как «пассивные». Такое деление тегов уменьшает загрузку процессора обновлением данных, принимаемых из физического устройства.

АЛГОРИТМИЗАЦИЯ ЗАДАЧ УПРАВЛЕНИЯ. ОСОБЕННОСТИ ЦИФРОВОГО УПРАВЛЕНИЯ. СХЕМА ВЗАИМОДЕЙСТВИЯ КОНТРОЛЛЕРА И ОБЪЕКТА УПРАВЛЕНИЯ

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 рис. 98.

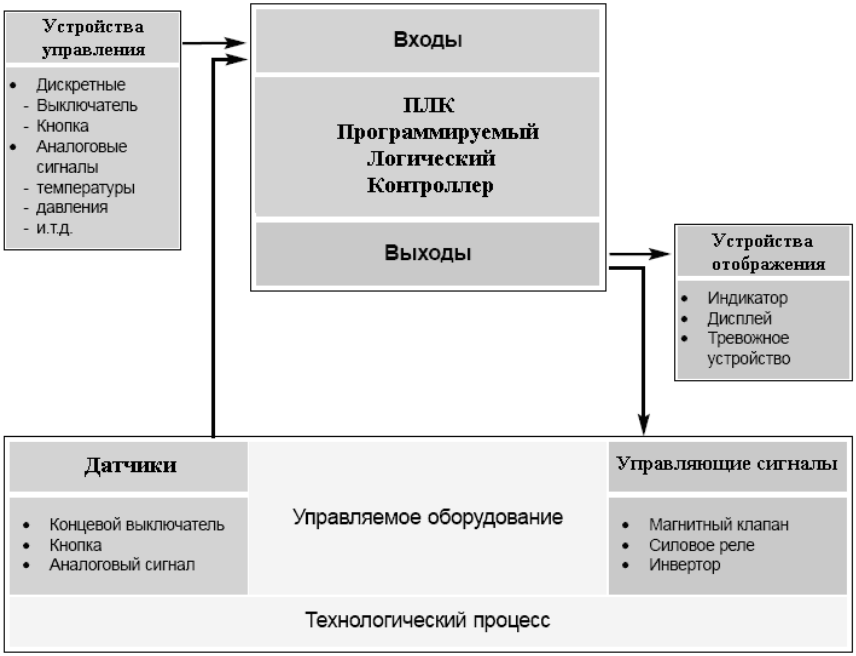


Рис. 98. Функциональная схема системы управления на базе ПЛК

Из рисунка видно, что ПЛК имеет три основные секции:
– входную;
– выходную;
– центральную.

Имеется еще источник питания. Возможно подключение к ПЛК внешнего ПК для программирования и отладки. Центральная секция содержит центральный процессор (ЦП), память и систему коммуникаций. Она выполняет обработку данных, принимаемых от входной секции данных, и передает результаты обработки в выходную секцию. Входная секция ПЛК обеспечивает ввод в центральную секцию состояния переключателей, датчиков и смарт-устройств. Через выходную секцию ЦП управляет внешними исполнительными устройствами, среди которых могут быть электромагнитные пускатели моторов, источники света, клапаны и смарт-устройства.

Для понимания работы контроллера на рис. 99 приведен алгоритм его работы.

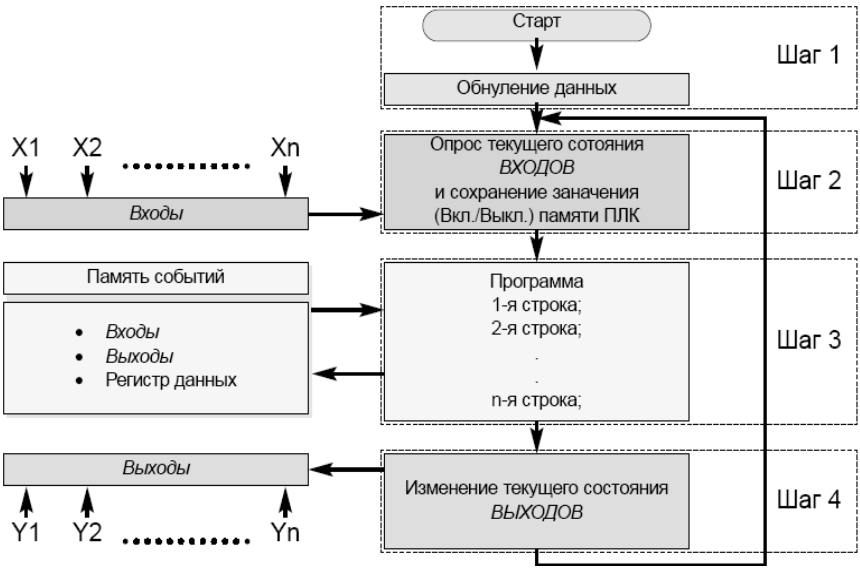


Рис. 99. Схема работы контроллера

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 состояние 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.

Управление процессом в реальном времени

Управляющий компьютер должен работать со скоростью, соответствующей скорости процесса. Само понятие «реальное время» указывает на то, что в реакции компьютерной системы на внешние события не должно быть заметно запаздывания. Различают две системы реального времени – жесткие и мягкие. В системе жесткого реального времени существует строго ограниченный временной порог, при превышении которого произойдут необратимые последствия. Данные последствия могут носить катастрофический характер.

В системах мягкого реального времени с увеличением времени реакции на воздействие ухудшатся характеристики системы, но никаких катастрофических последствий данное ухудшение иметь не будет. Главная специфика применения ПЛК – решение одновременно нескольких логических задач. Прикладная программа может иметь реализацию в виде множества логических задач, которые должны работать одновременно.

В качестве примера типичной задачи компьютерного управления технологическим процессом рассмотрим задачу управления прессом для пластика (рис. 100).

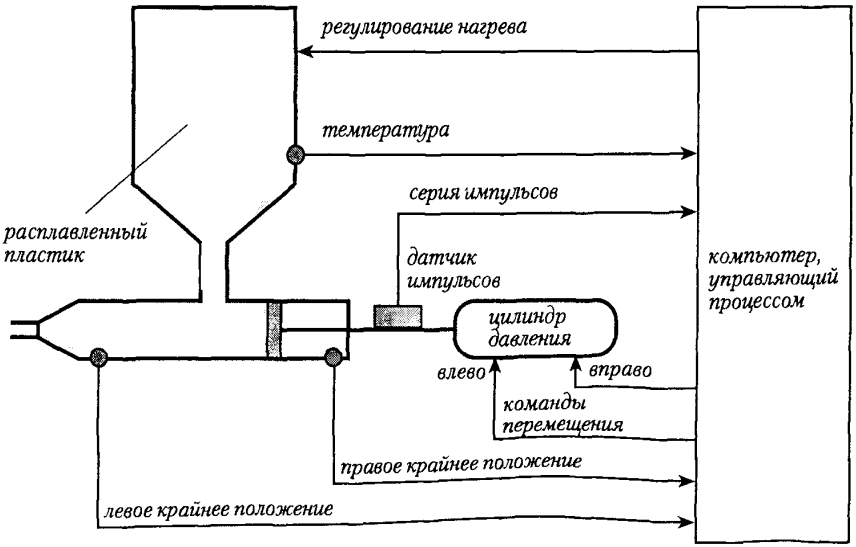


Рис. 100. Пресс для пластика

Компьютер должен одновременно регулировать температуру (поддерживать ее постоянной) и координировать последовательность технологических операций. Контейнер содержит расплавленный пластический материал; температура пластика должна поддерживаться в пределах узкого диапазона. Управляющий компьютер периодически считывает текущую температуру и рассчитывает тепло, необходимое для ее поддержания на требуемом уровне. Тепло поступает от нагревательного элемента, управляемого компьютером. Время

его работы согласовано с количеством тепла, которое необходимо подвести.

Нижняя часть пресса состоит из поршня, выталкивающего определенное количество расплавленного пластика через насадку. Когда поршень находится в крайнем правом положении, цилиндр заполняется пластиком. Затем поршень быстро перемещается влево, выдавливая требуемое количество пластика. Положение поршня контролирует импульсный датчик, генерирующий определенное число импульсов на каждый миллиметр перемещения, а объем выдавливаемого пластического материала определяется числом импульсов за время перемещения. Движение поршня прекращается при достижении заданного числа импульсов.

Чтобы обеспечить приемлемую производительность, температура пластика должна иметь заданное значение к тому моменту, когда поршень при движении вправо минует выходное отверстие контейнера. Компьютерная система должна регулировать температуру и движение поршня одновременно. Значение температуры поступает в виде непрерывного сигнала от датчика. Положение поршня рассчитывается исходя из числа импульсов. Кроме того, еще два датчика генерируют двоичные сигналы при достижении поршнем крайнего положения.

Управление на основе последовательного программирования

Попытаемся проанализировать следующую проблему: могут ли задачи управления в реальном времени решаться с помощью последовательного программирования. Блок-схема регулирования температуры представлена на рис. 101. Программа считывает температуру пластика каждые 10 секунд, определяет необходимое время нагрева (переменная heat_time), включает нагреватель и затем переходит в цикл занятого ожидания (busy loop) обновления счетчика времени (переменная C), во время которого компьютер не может выполнять никакие другие операции. Очевидно, что это не самое эффективное использование компьютера.

Алгоритм управления перемещением поршня показан на рис. 102. Компьютер выдает команду начать движение вправо, затем непрерывно контролирует информацию от датчика конечного

положения до тех пор, пока не получит сигнал о том, что оно достигнуто. Затем начинается обратное движение поршня влево, при этом компьютер должен в цикле занятого ожидания ждать очередного импульса и суммировать их (счетчик импульсов обозначен *n*). Цикл считается завершенным при достижении заданного числа импульсов (переменная pulse_ref). Затем весь цикл повторяется сначала. Так же как и при регулировании температуры, компьютер не может выполнять других операций, пока он находится в цикле ожидания очередного импульса.

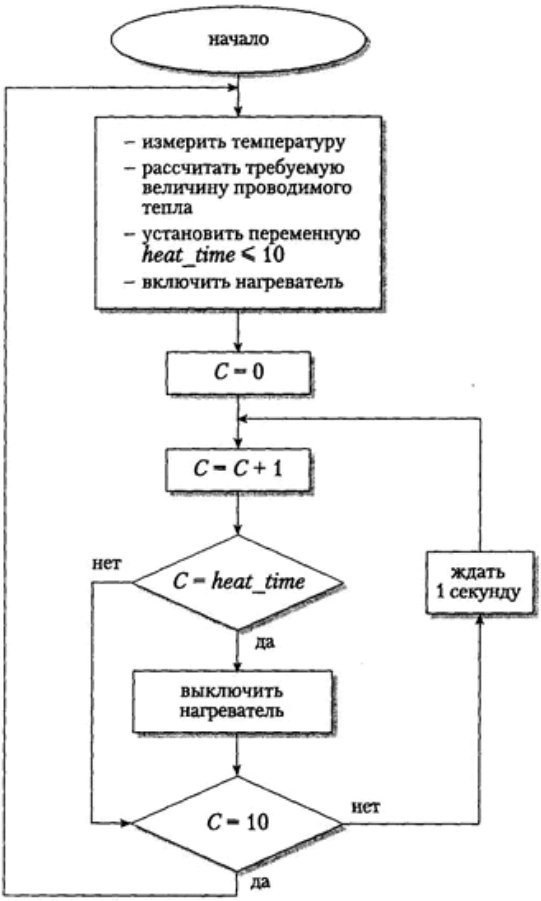


Рис. 101. Блок-схема алгоритма регулирования температуры пластика

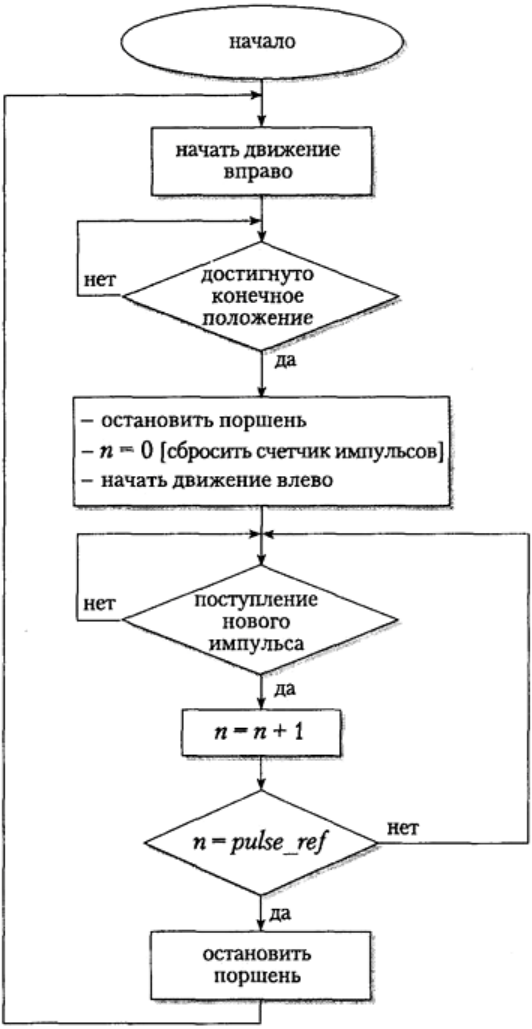


Рис. 102. Блок-схема управления движением поршня пресса

Каждую из двух задач можно решить непосредственно с помощью последовательного кода. Но объединить их в одной программе непросто. Циклы взаимного ожидания использовать нельзя, так как управляемый процесс не позволяет длительных задержек, а исполнение одной задачи не должно быть связано с другой. Можно

обойтись и без циклов ожидания, однако в этом случае программа будет все время переключаться между двумя задачами и проверять, какая задача должна исполняться следующей, станет громоздкой и сложной для анализа. Попытка последовательного расположения блоков инструкций, исполнение которых фактически должно быть параллельным, порождает взаимосвязи между практически независимыми функциями.

В любых цифровых устройствах непрерывность достигается за счет применения дискретных алгоритмов, повторяющихся через достаточно малые промежутки времени. Таким образом, вычисления в ПЛК всегда повторяются циклически (рис. 103). Одна итерация, включающая замер, обсчет и выработку воздействия, называется рабочим циклом ПЛК. Выполняемые действия зависят от значения входов контроллера, предыдущего состояния и определяются пользовательской программой.

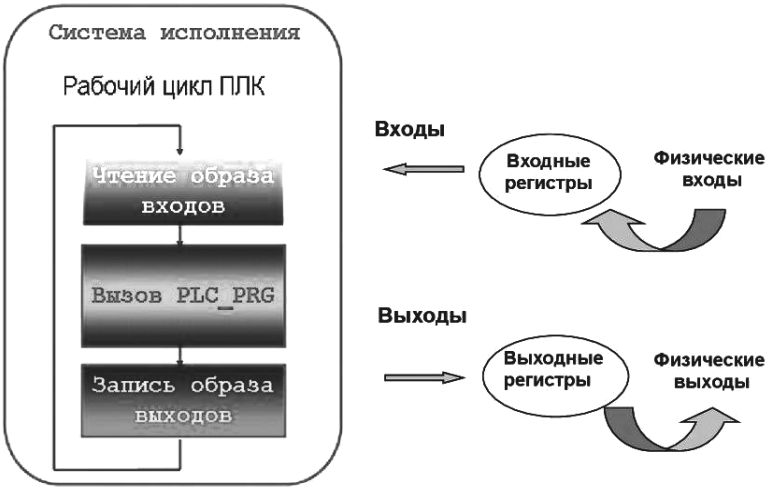


Рис. 103. Цикл работы ПЛК

В самом начале цикла ПЛК производит физическое чтение входов. Считанные значения размещаются в области памяти входов. Таким образом создается полная одномоментная зеркальная копия значений входов. Далее выполняется код пользовательской программы. Пользовательская программа работает с копией значений

входов и выходов, размещенной в оперативной памяти. Если прикладная программа не загружена или остановлена, то данная фаза рабочего цикла не выполняется. Отладчик системы программирования имеет доступ к образу входов/выходов, что позволяет управлять выходами вручную и проводить исследования работы датчиков. После выполнения пользовательского кода физические выходы ПЛК приводятся в соответствие с расчетными значениями.

Пользовательская программа работает только с мгновенной копией входов. Таким образом, значения входов в процессе выполнения пользовательской программы не изменяются в пределах одного рабочего цикла. Это фундаментальный принцип построения ПЛК сканирующего типа. Такой подход исключает неоднозначность алгоритма обработки данных в различных его ветвях.

Общая продолжительность рабочего цикла ПЛК называется временем сканирования. Время сканирования в значительной степени определяется длительностью фазы кода пользовательской программы. Время, занимаемое прочими фазами рабочего цикла, практически является величиной постоянной. Для задачи среднего объема в ПЛК с системой исполнения CoDeSys время распределится примерно так: 98 % – пользовательская программа, 2 % – все остальное.

Время реакции – это время с момента изменения состояния системы до момента выработки соответствующей реакции. Очевидно, для ПЛК время реакции зависит от распределения моментов возникновения события и начала фазы чтения входов. Если изменение значений входов произошло непосредственно перед фазой чтения входов, то время реакции будет наименьшим и равным времени сканирования. Худший случай, когда изменение значений входов происходит сразу после фазы чтения входов. Тогда время реакции будет наибольшим, равным удвоенному времени сканирования минус время одного чтения входов. Иными словами, время реакции ПЛК не превышает удвоенного времени сканирования.

Управление на основе прерываний

Для обработки событий, происходящих асинхронно по отношению к выполнению программы, лучше всего подходит механизм прерываний. Прерывание можно рассматривать как некоторое особое событие в системе, требующее моментальной реакции.

Возникновение подобных сигналов обусловлено такими событиями, как:

– завершение операций ввода/вывода;

– истечение заранее заданного интервала времени;

– попытка деления на нуль;

– сбой в работе аппаратного устройства и др.

Прерывание (англ. interrupt) – сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается и управление передается обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

Основная цель введения прерываний – реализация асинхронного режима работы и распараллеливания работы отдельных устройств вычислительного комплекса. Механизм прерываний реализуется аппаратно-программными средствами.

Структуры систем прерываний могут быть самыми различными, но все они имеют общую особенность – прерывание непременно ведет за собой изменение порядка выполнения команд процессором.

Обработка прерывания в ПЛК (рис. 104) производится в три этапа:

1. Прекращение выполнения текущей программы. Должно произойти так, чтобы потом вернуться и продолжить работу. Для этого необходимо сохранить содержимое регистров, так как они являются ресурсами, разделяемыми между программами. Эти регистры сохраняются микропроцессором автоматически. Наиболее удобным местом хранения регистров является стек.

2. Переход к выполнению и выполнение программы обработки прерывания. Здесь определяется источник прерывания и вызывается соответствующий обработчик прерывания.

Возврат управления прерванной программе. Необходимо привести стек в состояние, в котором он был сразу после передачи управления данной процедуре.

При обработке каждого прерывания должна выполняться следующая последовательность действий:

1) восприятие запроса на прерывание: прием сигнала и идентификация прерывания;

2) запоминание состояния прерванного процесса: определяется значением счетчика команд (адресом следующей команды) и содержимым регистров процессора;

3) передача управления прерывающей программе (в счетчик команд заносится начальный адрес подпрограммы обработки прерываний, а в соответствующие регистры – информация из слова состояния процессора);

4) обработка прерывания;

5) восстановление прерванного процесса и возврат в прерванную программу.

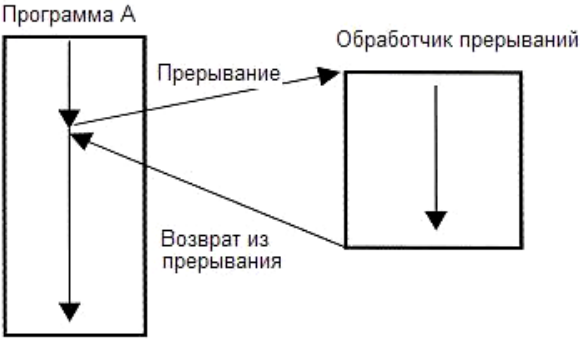


Рис. 104. Блок-схема обработки прерывания в ПЛК

Главные функции механизма прерывания:

– распознавание или классификация прерываний;

– передача управления соответственно обработчику прерываний;

– корректное возвращение к прерванной программе (перед передачей управления обработчику прерываний содержимое регистров процессора запоминается либо в памяти с прямым доступом, либо в системном стеке).

Типы прерываний. Прерывания, возникающие при работе вычислительной системы, можно разделить на 4 группы, представленные на схеме на рис. 105.

Аппаратные прерывания вызываются физическими устройствами и возникают по отношению к программе асинхронно, то есть в общем случае невозможно предсказать, когда и по какой причине программа будет прервана.

Рис. 105. Типы прерываний ПЛК

Аппаратные прерывания не координируются с работой программного обеспечения. Когда вызывается прерывание, то процессор оставляет свою работу, выполняет прерывание, а затем возвращается на прежнее место.

Внешние прерывания возникают по сигналу какого-либо внешнего устройства, например:

– прерывание, которое информирует систему о том, что требуемый сектор диска уже прочитан, его содержимое доступно программе;

– прерывание, которое информирует систему о том, что завершилась печать символа на принтере и необходимо выдать следующий символ;

– прерывания по нарушению питания;

– нормальное завершение некоторой операции ввода/вывода (нажатие клавиши на клавиатуре);

– прерывание по таймеру.

Прерывание по таймеру вызывается интервальным таймером. Этот таймер содержит регистр, которому может быть присвоено определенное начальное значение посредством специальной привилегированной команды. Значение этого регистра автоматически уменьшается на 1 по истечении каждой миллисекунды времени. Когда это значение становятся равным нулю, происходит прерывание по таймеру. Подобный интервальный таймер используется операционной системой для определения времени, в течение которого программа пользователя может оставаться под управлением машины.

Внутренние прерывания вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями, а именно прерывание происходит:

– при нарушении адресации (в адресной части выполняемой команды указан запрещенный или несуществующий адрес, обращение к отсутствующему сегменту или странице при организации механизмов виртуальной памяти);

– наличии в поле кода не задействованной двоичной комбинации;

– делении на нуль;

– переполнении или исчезновении порядка;

– обнаружении ошибок четности, ошибок в работе различных устройств аппаратуры средствами контроля.

ОСНОВНЫЕ ОПЕРАЦИИ: ВВОД, ПЕРЕРАБОТКА ИНФОРМАЦИИ, ВЫВОД СИГНАЛОВ УПРАВЛЕНИЯ

Обработка аналоговых сигналов в процессе ввода в контроллер

Для ввода аналогового сигнала в контроллер и его последующей обработки, он должен быть оцифрован, то есть преобразован в цифровой код. Процесс обработки сигнала от аналогового датчика до использования в контроллере схематически показан на рис. 106.

Рис. 106. Схема обработки аналогового сигнала при вводе в контроллер

Сигналы от датчиков доводятся до нормированного уровня (4–20 мА, 0–10 В) нормирующими преобразователями (НП)

и проходят этап аналоговой фильтрации. Аналоговые фильтры позволяют устранить высокочастотные шумы, которые могут быть вызваны, например, электромагнитными помехами при передаче сигнала по кабелю.

Необходимо отметить, что сигнал должен быть отфильтрован от высокочастотных шумов до цифровой обработки в контроллере. Это является необходимым условием правильного выбора периода дискретизации при вводе сигнала. Дело в том, что измеряемый аналоговый сигнал, например, сигнал аналогового датчика температуры – термосопротивления или термопары, представляет собой смесь полезного сигнала и шумов. Значение полезного сигнала однозначно обусловлено значением измеряемой величины: так, в случае, когда аналоговым датчиком является металлический термометр сопротивления, зависимость сопротивления его чувствительного элемента от температуры выражается известной формулой

$$R_{\text{Т}} \approx R_0\left(1 + AT + BT^2\right),\ \left(0\,^{\circ}\text{C} \le T \le 850\,^{\circ}\text{C}\right),$$

где $R_{\text{Т}}$ – сопротивление чувствительного элемента при температуре T °С, Ом;

R_0 – сопротивление при 0 °С, Ом;

$A = 3{,}9083 \cdot 10^{-3}\ ^{\circ}\text{C}^{-1}$, $B = -5{,}775 \cdot 10^{-7}\ ^{\circ}\text{C}^{-2}$ – коэффициенты, значения которых определены опытным путем и стандартизированы.

Шумы в датчиках, называемые также стохастическими погрешностями, являются случайными процессами, то есть колебания значений сигнала шума имеют случайный характер, являются в достаточной степени непредсказуемыми и могут меняться очень быстро.

В итоге при изменении аналогового сигнала, существует общее правило, определяющее частоту, с которой нужно выполнять измерения – частоту опроса датчика, частоту выборки: частота выборки сигнала определяется из условия $f_{\text{изм}} \ge 2 f_{\max}$, где $f_{\max}$ – это максимальная частота изменения сигнала.

В качестве, пожалуй, самого наглядного бытового примера можно привести пример, когда аналоговые звуковые сигналы – музыка и другие аудиозаписи – преобразуются и хранятся в компьютерах в виде цифровых музыкальных файлов. Так как человеческое ухо способно различать на слух звуковые колебания частотой от 20 Гц до 20 кГц, то для качественного преобразования звука в цифровой формат используются частоты выборки 44,1 кГц (соответствует формату Audio CD), 48 кГц, 96 кГц (цифровой формат DVD-audio) и более.

При более низкой частоте выборки ухудшается качество оцифровки сигнала. Так, например, в современной цифровой телефонии используется частота выборки 8 кГц, так как ее достаточно для более-менее удовлетворительного качества передачи человеческого голоса. В то же время чрезмерно большая частота выборки хотя и улучшит качество оцифровки аналогового сигнала, но при этом будет, зачастую, неоправданно загружать контроллер.

Отфильтрованные сигналы от датчиков поступают на аналоговый мультиплексор, основное назначение которого – последовательное подключение сигналов от N датчиков к устройству выборки-хранения (УВХ) и аналого-цифровому преобразователю (АЦП) для дальнейшей обработки. Такая схема позволяет существенно снизить общую стоимость системы ввода за счет применения только одного УВХ и АЦП на все каналы аналогового ввода. УВХ запоминает мгновенное значение сигнала в момент подключения датчика и удерживает его постоянным на своем выходе на время преобразования в АЦП.

В контроллере введенный цифровой сигнал проверяется на физическую достоверность и, при необходимости, проходит этап цифровой (программной) фильтрации.

Происхождение входного сигнала можно представить так, как показано на рис. 107. Первичный сигнал от датчика на месте преобразуется электронным устройством в определенный стандартный сигнал, а совокупность датчика и этого устройства называется информационным преобразователем. После этого стандартизированный сигнал, несущий информацию об измеряемой переменной объекта управления, может быть подан на обычную аналоговую входную плату.

В мире принято использовать два вида стандартных электрических аналоговых сигналов:

− постоянное напряжение от 0 до 10 В;

− постоянный ток от 4 до 20 мА или от 0 до 20 мА.

Рис. 107. Организация ввода аналоговых сигналов в ПЛК

При этом минимально возможному измеряемому значению физической величины с датчика соответствует низшее значение стандартного сигнала (0 В, 0 или 4 мА), а максимальному измеряемому значению – максимальное значение стандартного сигнала (10 В или 20 мА). Например, пусть датчик температуры имеет допустимый диапазон измерения от –50 ºС до +100 ºС. Тогда, при стандартном выходном сигнале в виде постоянного напряжения температуре – 50 ºС будет соответствовать напряжение на выходе датчика 0 В (отсутствие напряжения), а при температуре +100 ºС на выходе получим напряжение 10 В.

Однако стандартные аналоговые сигналы являются сигналами низкого уровня и могут значительно искажаться в результате наводок от различных источников электромагнитного поля, дополнительных сопротивлений в измерительных цепях и т. д. Кроме того, если выбран стандартный сигнал в виде постоянного напряжения, то обрыв провода от датчика не рассматривается как аварийная ситуация, а может быть понят управляющим устройством как минимальное значение измеряемой величины.

Сигнал, представленный электрическим током, менее подвержен влиянию шумов, чем сигнал, представленный напряжением, поэтому обычно выбирается подключение датчика в виде токового контура (рис. 108), причем токовый сигнал на приемной стороне

преобразуется в напряжение при помощи балластного резистора. Токовый контур можно использовать с несколькими приемными устройствами (это могут быть, например, измерительный прибор, регистратор или вход ПЛК), соединенными последовательно.

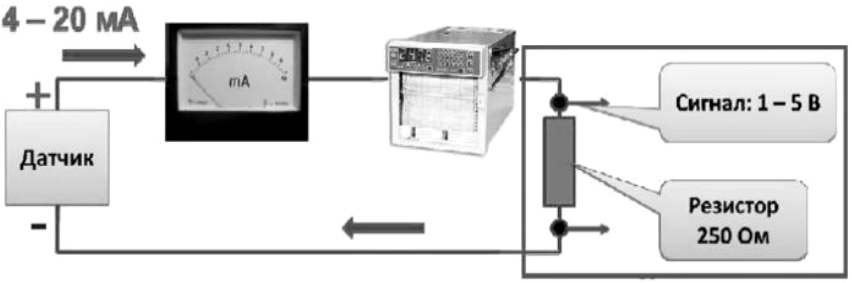


Рис. 108. Подключение датчика в виде токового контура 4–20 мА

Самый распространенный стандарт представляет аналоговый сигнал в виде тока с диапазоном изменения 4–20 мА, где 4 мА соответствует минимальному уровню сигнала, а 20 мА – максимальному. Сигнал 4–20 мА часто с помощью балластного резистора величиной 250 Ом преобразуется в сигнал 1–5 В.

«Нулевой» сигнал 4 мА (называемый смешением) предназначен для двух целей. Во-первых, он используется, как зашита от повреждений преобразователя или кабельного шнура. Если происходит отказ преобразователя или обрыв шнура или же в линии связи возникает короткое замыкание, то ток через балластный резистор будет равен нулю, что соответствует «отрицательному» сигналу 0 В на приемной стороне. Это может быть очень легко обнаружено и использовано как аварийный сигнал «неисправность преобразователя».

Контроль граничных значений аналоговых переменных

Часто в процессе управления измеряемые значения аналоговых сигналов датчиков нужно сравнить с некоторыми номинальными значениями и результат сравнения можно использовать для управления исполнительными механизмами. Так, в алгоритме управления работой стиральной машины (см. рис. 32) окончание стрики

наступает после процесса охлаждения белья в стиральном барабане и определяется снижением температуры в стиральном барабане ниже 40°C. На текстовом языке программирования контроллеров это условие может быть представлено в виде следующего фрагмента программного кода:

```
IF Tdeg<40 THEN   //если температура в барабане менее 40 ºC
Cooling:=FALSE; // прекратить охлаждение
Stop:=TRUE; // перейти к этапу завершения стрики
END_IF;
```

Аналоговые входные сигналы с 20 % компенсацией

Аналоговые измерительные преобразователи часто используют компенсацию в 20 %. Это создает так называемое изображение live zero для аналогового входного модуля. Пример аналогового выходного сигнала с 20 % компенсацией показан на рис. 109.

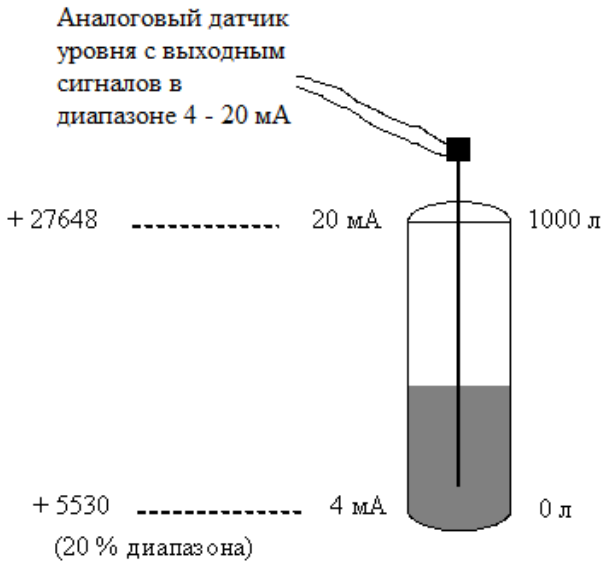


Рис. 109. Пример аналогового выходного сигнала с 20 % компенсацией

Согласно рисунку вверху значение, передаваемое с датчика на аналоговый входной модуль, составляет от 4 до 20 мА. Вместо того

чтобы подавать 0 мА при 0 л, датчик посылает 4 мА. Из-за этой компенсации контроллер определяет, заявлен ли нулевой уровень передачи данных. Если бы датчик при нулевом уровне (0 л) посылал 0 мА, то ЦПУ не смог бы определить, вышел ли датчик из строя или нет. Поэтому для нулевого уровня (0 л) посылается 4 мА. При выходе датчика из строя ток падает ниже ожидаемых 4 мА, и ЦПУ определяет, что заявленный уровень недействителен.

Компенсация в 20 % для аналогового входа рассматривается согласно следующему уравнению:

$$y = \left(N_{\text{ai}} - 5530 \right) \frac{125}{100},$$

где N_{ai} – измеренное оцифрованное значение сигнала датчика;

y – значение сигнала с учетом компенсации в 20 % полезного диапазона.

СТРУКТУРА УПРАВЛЕНИЯ С ЦИФРОВЫМИ РЕГУЛЯТОРАМИ НА БАЗЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ; ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕГУЛЯТОРОВ

Аналоговые и дискретные регуляторы

Регуляторы можно строить на основе как аналоговой, так и цифровой техники. Соответственно, для анализа и проектирования аналогового и цифрового регулятора требуются разные математические методы. Хотя цифровая технология позволяет хорошо моделировать работу аналоговой системы управления, то есть реализовать аналоговые понятия цифровыми средствами, ее возможности гораздо шире. Например, можно построить нелинейные и самонастраивающиеся регуляторы, которые нельзя создать на основе только аналоговых средств. Главная проблема цифрового управления – найти соответствующую структуру регулятора и его параметры. После определения этих параметров реализация алгоритмов управления обычно представляет собой простую задачу. Помимо этого, каждый регулятор должен включать средства защиты, предотвращающие опасное развитие процесса под действием регулятора в нештатных ситуациях. Многие производственные процессы характеризуются несколькими входными и выходными параметрами. В большинстве случаев внутренние связи и взаимодействие соответствующих сигналов не имеют принципиального значения, и процессом можно управлять с помощью набора простых регуляторов, при этом каждый контур управления обрабатывает одну пару вход/выход. Такой подход используется в системах прямого цифрового управления.

Релейное управление

Релейное управление – самый простой алгоритм из возможных, ведь у нас есть только два состояния – «вкл.» и «выкл.».

Рассмотрим алгоритмы, которые сделают релейное управление более надежным и повысят точность регулирования. Начнем с самого простого и очевидного:

```
if (temp < 50.0) digitalWrite(relayPin, 1);
else digitalWrite(relayPin, 0);
```

Данный код не нуждается в комментариях, он просто включает реле, когда условная температура ниже 50 ºС, и выключает, когда она выше. Если вызывать данный код без задержки или таймера – мы получим жуткий дребезг в момент включения и выключения реле, так как шумы измерений будут постоянно менять результат условия.

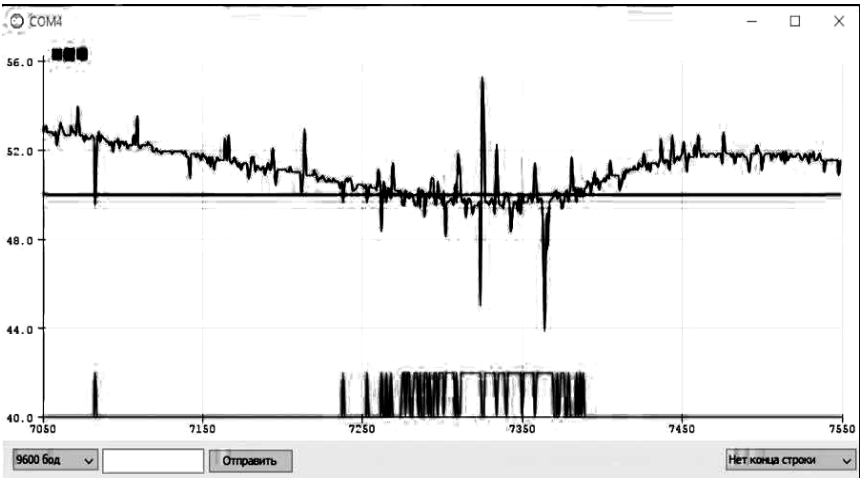


Рис. 110. Двухпозиционное регулирование
с регулятором без зоны неоднозначности

Зеленый график – как раз состояние реле. Из этого графика также следует неутешительный вывод: значения надо фильтровать, это сильно увеличит стабильность системы.

Первым шагом к созданию нормального релейного регулятора является период работы регулятора, его можно реализовать как задержкой:

```
if (temp < 50.0) digitalWrite(relayPin, 1);
else digitalWrite(relayPin, 0);
delay(1000);
```

так и таймером:

```
static uint32_t tmr;
if (millis() - tmr >= 1000) {
tmr = millis();
```

if (temp < 50.0) digitalWrite(relayPin, 1);
else digitalWrite(relayPin, 0);
}

Ситуация изменится к лучшему, ведь даже при всем желании реле не сможет переключаться чаще, чем раз в секунду.

Второй способ – ***гистерезис*** – позволяет еще сильнее уменьшить количество переключений реле и даже избавиться от опроса по таймеру, что повышает реакцию системы на изменения, сохранив при этом хорошую устойчивость к помехам. Гистерезис разделяет установку на две, чуть меньше и чуть больше, на размер окна гистерезиса (рис. 111).

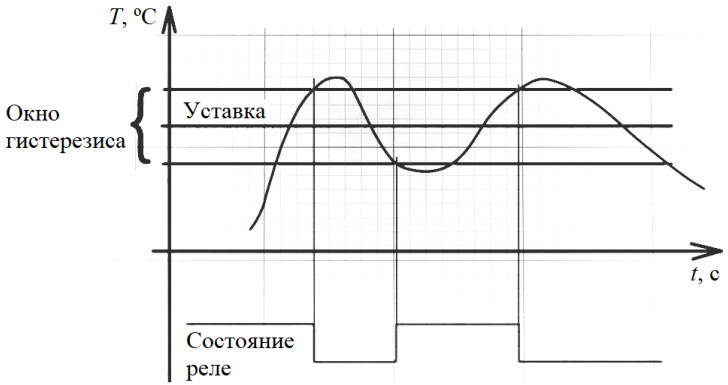


Рис. 111. Принцип двухпозиционного регулирования с зоной неоднозначности

Логика работы такова, что мы включаем реле на нагрев ниже нижней линии, и выключаем только выше верхней. То есть образуется область, внутри которой система, грубо говоря, движется по инерции от последнего переключения и переходит в новое состояние только при выходе из этой области. Очевидно, что добавление гистерезиса сильно уменьшает не только количество переключений реле, но и точность, потому что мы сами задаем область, точность внутри которой нам фактически безразлична, как и шумы измерения.

В коде гистерезис можно реализовать следующим образом:

#define RELAY_PIN 2
float setpoint = 50.0; // установка

float hyster = 2; // ширина окна гистерезиса

//.....

static bool relayState = false; // обязательно глобальная или статическая!

if (temp < (setpoint - hyster)) relayState = true;

else if (temp > (setpoint + hyster)) relayState = false;

digitalWrite(RELAY_PIN, relayState);

Пример использования гистерезиса в регулировании уровня воды в баке приведены на рис. 112 и 113.

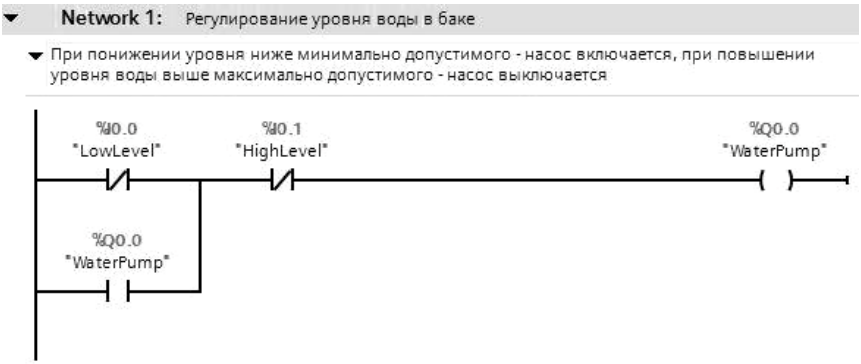


Рис. 112. Двухпозиционное регулирование уровня воды в баке с зоной неоднозначности, программа на языке релейно-контактных лестничных диаграмм LD

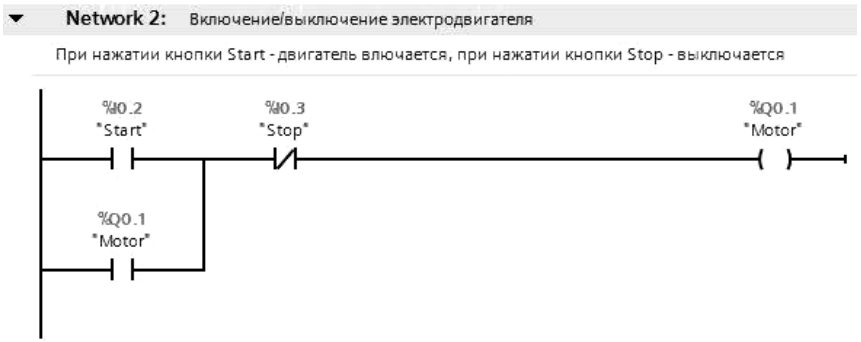


Рис. 113. Программа управления работой электродвигателя (пуск/останов кнопочными постами) на языке релейно-контактных лестничных диаграмм LD

Алгоритм ПИД-регулятора

ПИД-регулятор – прибор, встроенный в управляющий контур, с обязательной обратной связью. Он предназначен для поддержания установленных уровней задаваемых величин, например, температуры воздуха. Устройство подает управляющий или выходной сигнал на устройство регулирования, на основании полученных данных от датчиков или сенсоров. Контроллеры обладают высокими показателями точности переходных процессов и качеством выполнения поставленной задачи.

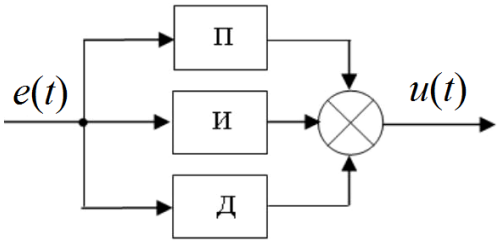


Рис. 114. Структурная схема (математическое описание) ПИД-регулятора

Работа ПИД-регулятора заключается в подаче выходного сигнала о силе мощности, необходимой для поддержания регулируемого параметра на заданном уровне. Для вычисления показателя используют сложную математическую формулу, в составе которой есть 3 коэффициента – пропорциональный, интегральный, дифференциальный.

Возьмем в качестве объекта регулирования емкость с водой, в которой необходимо поддерживать температуру на заданном уровне с помощью регулирования степени открытия клапана с паром. Пропорциональная составляющая появляется в момент рассогласования с вводными данными. Простыми словами это звучит так – берется разница между фактической температурой и желаемой, умножается на настраиваемый коэффициент и получается выходной сигнал, который должен подаваться на клапан. То есть как только градусы упали, запускается процесс нагрева, поднялись выше желаемой отметки – происходит выключение или даже охлаждение.

Дальше вступает интегральная составляющая, которая предназначена для того, чтобы компенсировать воздействие окружающей среды или других возмущающих воздействий на поддержание нашей температуры на заданном уровне. Поскольку всегда присутствуют дополнительные факторы, влияющие на управляемые приборы, в момент поступления данных для вычисления пропорциональной составляющей, цифра уже меняется. И чем больше внешнее воздействие, тем сильнее происходят колебания показателя. Происходят скачки подаваемой мощности.

Интегральная составляющая пытается на основе прошлых значений температуры, вернуть ее значение, если оно поменялось.

А дальше выходной сигнал регулятора, согласно коэффициенту, подается для повышения или понижения температуры. Со временем подбирается та величина, которая компенсирует внешние факторы, и скачки исчезают.

Интеграл используется для исключения ошибок путем расчета статической погрешности. Главное в этом процессе – подобрать правильный коэффициент, иначе ошибка (рассогласование) будет влиять и на интегральную составляющую.

Дискретная модель ПИД-регулятора. Для того чтобы аналоговый регулятор реализовать программно в ПЛК, необходима его дискретная модель.

В цифровых системах управления входная величина регулятора $e(t)$ – ошибка регулирования – квантуется в аналого-цифровом преобразователе (АЦП) по времени с интервалом дискретизации T, и на его вход поступает дискретная последовательность $e(n)$. Выходная последовательность дискретного ПИД-регулятора формируется в виде суммы

$$u(n) = u_{\text{п}}(n) + u_{\text{и}}(n) + u_{\text{д}}(n),$$

где $u_{\text{п}}(n)$, $u_{\text{и}}(n)$ и $u_{\text{д}}(n)$ – соответственно пропорциональная, интегральная и дифференциальная составляющие. В зависимости от выбранного метода перехода от непрерывных операторов к их дискретным аналогам возникают различные варианты уравнений, описывающих дискретные ПИД-регуляторы.

Пропорциональная составляющая определяется формулой:

$$u_{\text{п}}(n) = k_{\text{п}} \cdot e(n).$$

Интегральную составляющую можно аппроксимировать конечной суммой:

$$u_{\text{и}}(n) = k_{\text{и}} T \sum_{m=0}^{n-1} e(m).$$

Производная функции $e(t)$ в момент времени $t = Tn$ аппроксимируется обратной разностью. При этом:

$$u_{\text{д}}(n) = k_{\text{д}} \cdot \left.\frac{de(t)}{dt}\right|_{t=Tn} \approx k_{\text{д}} \frac{e(n) - e(n-1)}{T}.$$

Таким образом, алгоритм работы дискретного ПИД-регулятора описывается выражением:

$$u(n) = k_{\text{п}} \cdot e(n) + k_{\text{и}} T \sum_{m=0}^{n-1} e(m) + k_{\text{д}} \frac{e(n) - e(n-1)}{T}.$$

В этом алгоритме для формирования интегральной составляющей необходимо запоминать все предыдущие значения сигнала ошибки $e(t)$ и суммировать их на каждом интервале дискретности. Это существенно увеличивает время вычисления текущего значения $u(n)$ управляющего воздействия. Поэтому, как правило, используют рекуррентные алгоритмы, в которых для вычисления текущего значения используется предыдущее значение $u(n-1)$:

$$u(n) = u(n-1) + q_0 \cdot e(n) + q_1 \cdot e(n-1) + q_2 \cdot e(n-2),$$

где $q_0 = k_{\text{п}} + \frac{k_{\text{д}}}{T}$; $q_1 = -k_{\text{п}} + k_{\text{и}} T - 2\frac{k_{\text{д}}}{T}$; $q_2 = \frac{k_{\text{д}}}{T}$.

Таким образом осуществляется переход от аналогового к цифровому управлению.

Выбор периода дискретизации и параметров регулятора

В общем случае выбор периода дискретизации в цифровых системах управления является сложной и неоднозначной задачей, поскольку приходится учитывать противоречивые требования. Так, если выбрать период дискретизации слишком малым, то возрастает время вычисления управляющего воздействия. При большом периоде дискретизации ухудшается качество регулирования. Поэтому выбор периода дискретизации – далеко не тривиальная задача и требует учета многих факторов.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ СИСТЕМ АВТОМАТИЗАЦИИ

Автоматическая система для отбрасывания бутылок

Принцип работы: данная система позволяет отбрасывать упавшие бутылки, которые затрудняют процесс производства (рис. 115).

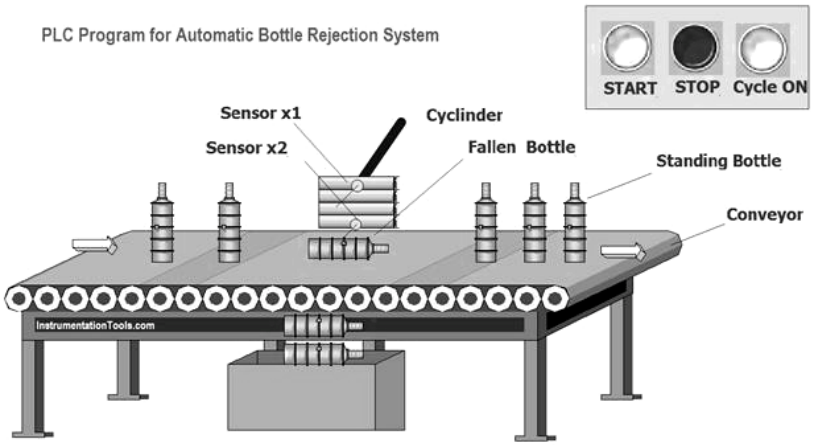


Рис. 115. Схема технологического процесса

Ленточный конвейер используется для перемещения бутылок с одной станции на другую. Но прежде чем бутылки попадут на заправочную станцию, необходимо сделать все бутылки стоящими для дальнейшего их заполнения. Упавшая бутылка на конвейере может создать проблему в следующем процессе, поэтому здесь показана простая программа для ПЛК, которая обрабатывает с конвейера упавшую бутылку.

Этот процесс осуществляется с помощью датчиков и исполнительных механизмов. Когда конвейер работает, все бутылки перемещаются с одной станции на другую для последующего процесса. Для обнаружения стоящих и упавших бутылок используются два датчика и один пневматический цилиндр для выталкивания упавшей бутылки с конвейера.

Программа управления приведена на рис. 116, 117.

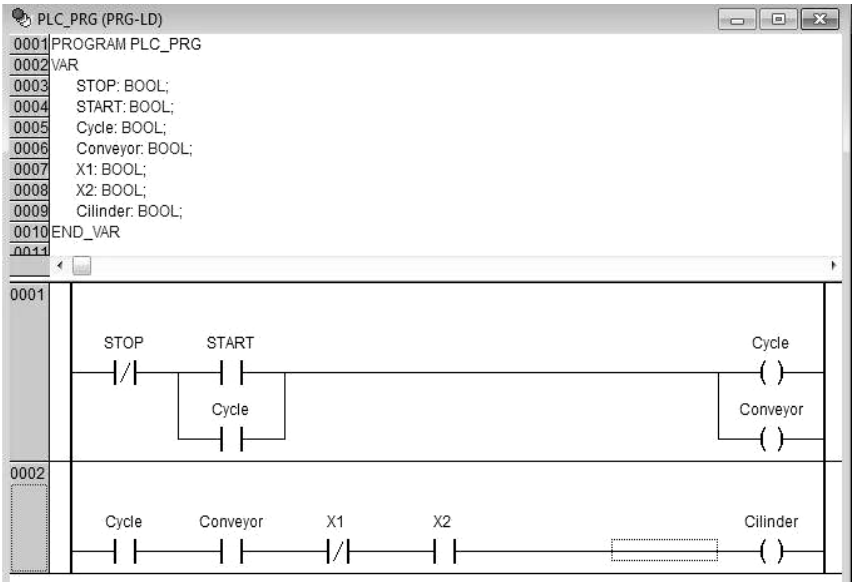


Рис. 116. Программа на языке релейных диаграмм для ПЛК в CodeSys автоматической системы отбраковки бутылок

Описание работы программы (рис. 117). Нажимаем кнопку START, тем самым питая схему. Запускается цикл и конвейер. Во второй части цепи находится два индукционных датчика X1 и X2, с помощью которых и определяется положение бутылки на конвейере. Когда бутылка упала, срабатывает датчик X2 и его контакт разрывает цепь, тем самым не пропуская упавшую бутылку.

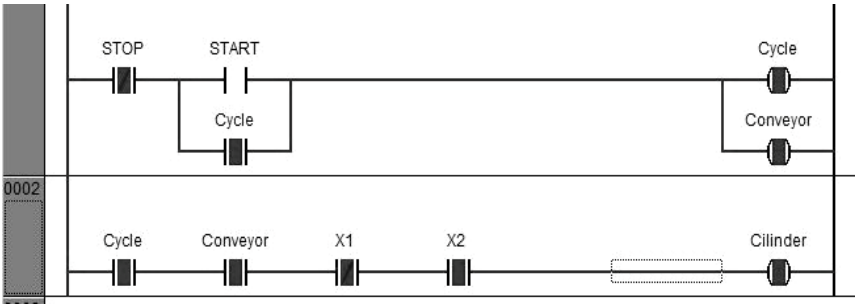


Рис. 117. Процесс сброса бутылки

Когда бутылки перемещаются по конвейеру, эти датчики определяют положение бутылок независимо от того, стоят они или упали. Датчик X2 определяет нижнее положение бутылки, а датчик X1 верхнее положение бутылки. Если датчик X2 определяет бутылку, а датчик X1 не определяет, то включается пневматический привод (Cilinder), и он отбросит бутылку с конвейера. После этого остальные бутылки попадут на станцию розлива воды и весь цикл будет завершен.

Автоматизация вентиляционной системы

Рассмотрим пример работы программы ПЛК для системы управления вентиляторами для промышленности. Схема вентиляционной системы представлена на рис. 118.

Во время работы системы должны работать любые два вентилятора из трех. Для запуска любых двух вентиляторов, например вентилятора *2* и вентилятора *3*, для каждого предусмотрены отдельные кнопки запуска и остановки.

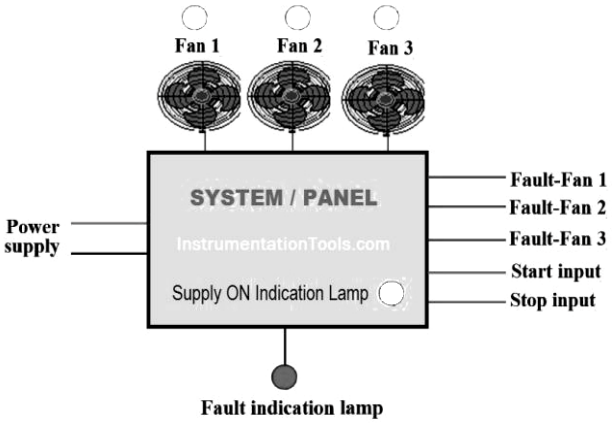


Рис. 118. Схема вентиляционной системы

Предположим, что вентилятор *2* и вентилятор *3* работают, и один из них выходит из строя, тогда вентилятор *1* должен включаться автоматически, то есть в любой момент времени должны работать два вентилятора. В случае неисправности любых двух вентиляторов входное питание системы должно автоматически отключаться.

Состояние «вкл.» вентиляторов, а также состояние основного питания должно указываться соответствующим светодиодом. Если есть неисправность с более чем одним вентилятором, то это состояние должно указываться мигающим светодиодом с частотой 5 Гц. Неисправность с одним вентилятором или отсутствие неисправности с вентилятором должны указываться постоянным светом на индикаторе состояния неисправности.

Это простой пример блока управления вентиляторами, используемый в промышленности.

Описание программы. В программе используется схема запуска цикла – катушка CYCLEK. Его можно запустить, нажав START, и остановить, нажав STOP. Главный выключатель (MS) должен быть включен (рис. 119).

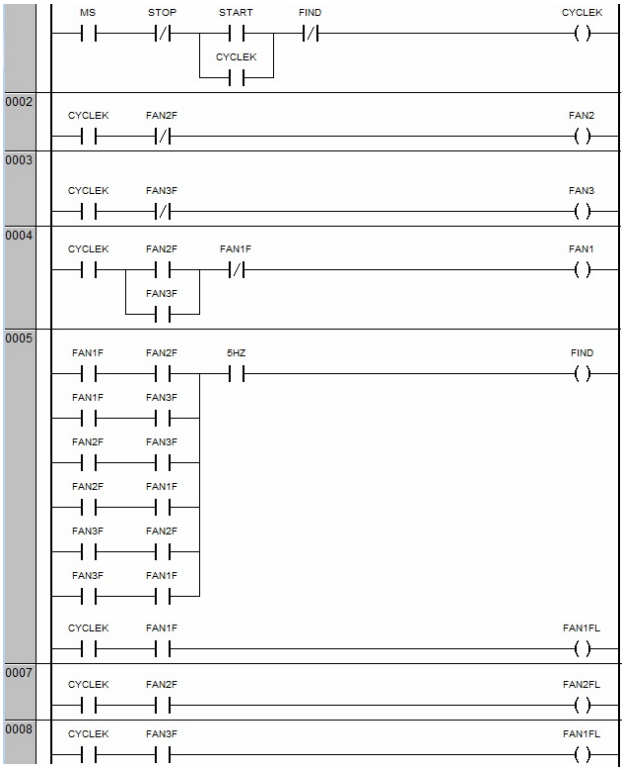


Рис. 119. Программа на языке релейных диаграмм для ПЛК в CodeSys

Когда цикл включен (CYCLEK) и неисправностей вентилятора *2* и вентилятора *3* нет, то вентилятор *2* (FAN2) и вентилятор *3* (FAN3) будут включены. Если вентилятор *2* или вентилятор *3* неисправен, то будет запущен вентилятор *1* (FAN1).

В системе, если какие-либо два вентилятора из трех неисправны, то лампа индикации неисправности (FIND) начнет мигать с частотой 5 Гц. Для этого нужно использовать специальный таймер, но чтобы не усложнять программу ограничимся пока в ней отдельным контактом. Индикаторные лампы для вентилятора *1*, вентилятора *2* и вентилятора *3* включаются в соответствии с сигналом неисправности.

Автоматизация водоотливной установки

На рис. 120 представлена схема водоотливной установки, как пример автоматизируемого технологического процесса.

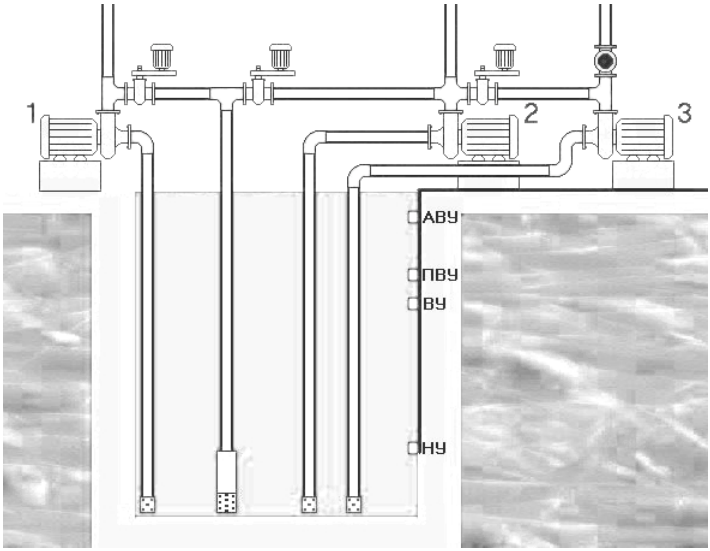


Рис. 120. Схема водоотливной установки

На рисунке есть следующие обозначения:
1, 2, 3 – водооткачивающие насосы;
НУ – нижний уровень;

ВУ – верхний уровень;

ПВУ – повышенный верхний уровень;

АВУ – аварийный верхний уровень.

Задача водоотливной установки сводится к откачке воды из сливной ямы шахтной выработки. Не допускается превышение уровня воды выше аварийного уровня.

На каждом уровне установлены дискретные датчики уровня жидкости с двумя выходными логическими состояниями: истина и ложь. Также в системе имеется датчик давления, предназначенный для контроля наличия воды в насосной системе.

Датчики подключаются к цифровым входам ПЛК.

В ПЛК записывается управляющая программа, формирующая управляющие сигналы по данным от датчиков. Эти сигналы поступают на выходы релейного типа, далее, через промежуточные согласующие устройства, на исполнительные устройства – различные насосы.

Поэтому важно понимать, что при программировании ПЛК имеет значение лишь типы и число входов выходов, а также логика управления процессом.

Поэтому давайте опишем входные/выходные сигналы ПЛК, управляющего водоотливной установкой.

Входные сигналы:

1) датчик нижнего уровня: тип – логический;

2) датчик давления: тип – логический;

3) датчик верхнего уровня: тип – логический;

4) датчик повышенного верхнего уровня: тип – логический;

5) датчик аварийного верхнего уровня: тип – логический;

Выходные сигналы:

1) коммутация питания заливочного насоса: тип – релейный;

2) коммутация питания первого насоса: тип – релейный;

3) коммутация питания второго насоса: тип – релейный;

4) коммутация питания третьего насоса: тип – релейный.

Словесное описание процесса работы водоотливной установки:

1. Вода начинает заполнять емкость.

2. При достижении нижнего уровня (датчик НУ) включается заливочный насос.

3. Насос работает до тех и только в том случае если давление в системе недостаточно. То есть значение датчика давления равно логическому нулю.

4. При достижении водой верхнего уровня срабатывает датчик верхнего уровня и включается первый насос.

5. При включении первого насоса начинается отсчет времени работы первого насоса. За заранее определенное значение времени первый насос должен откачать воду до нижнего уровня.

6. Если вода продолжает повышаться, то есть скорость притока воды выше, чем производительность первого насоса, то вода достигает повышенный верхний уровень, срабатывает датчик ПВУ и включается второй насос. Кроме того, второй насос включается также и в том случае если первый насос не успевает откачать воду до нижнего уровня.

7. При включении второго насоса начинается отсчет времени работы второго насоса. За заранее определенное значение времени первый и второй насосы должны откачать воду до нижнего уровня.

8. Если они не справляются с этой задачей, или вода продолжает прибывать и достигает верхнего аварийного уровня, включается третий насос. Расчет системы осуществляется таким образом, чтобы в наихудшем варианте хватило производительности трех насосов. Поэтому уровень воды постепенно опускается до нижнего уровня.

9. Когда вода опускается до нижнего уровня все включенные насосы отключаются и вода вновь начинает прибывать.

По приведенному словесному описанию составим блок-схему алгоритма ТП (рис. 121).

Ниже приводится листинг программы АСУ водоотливом на языке ST:

```
if NY  then
        ZAL:=not(DAWL);
        if WY and not(NS1) then
                NS1:=true;
        END_IF;
        if (PWY or TNS1) and not(NS2) then
                NS2:=true;
        END_IF;
        if (AWY or TNS2) and not(NS3) then
                NS3:=true;
        END_IF;
end_if;
if not(NY) then
```

```
        NS1:=false;
        NS2:=false;
        NS3:=false;
        ZAL:=false;
END_IF;
ton1 (NS1, qt);
TNS1:=ton1.Q;
ton2 (NS2, qt);
TNS2:=ton2.Q;
```

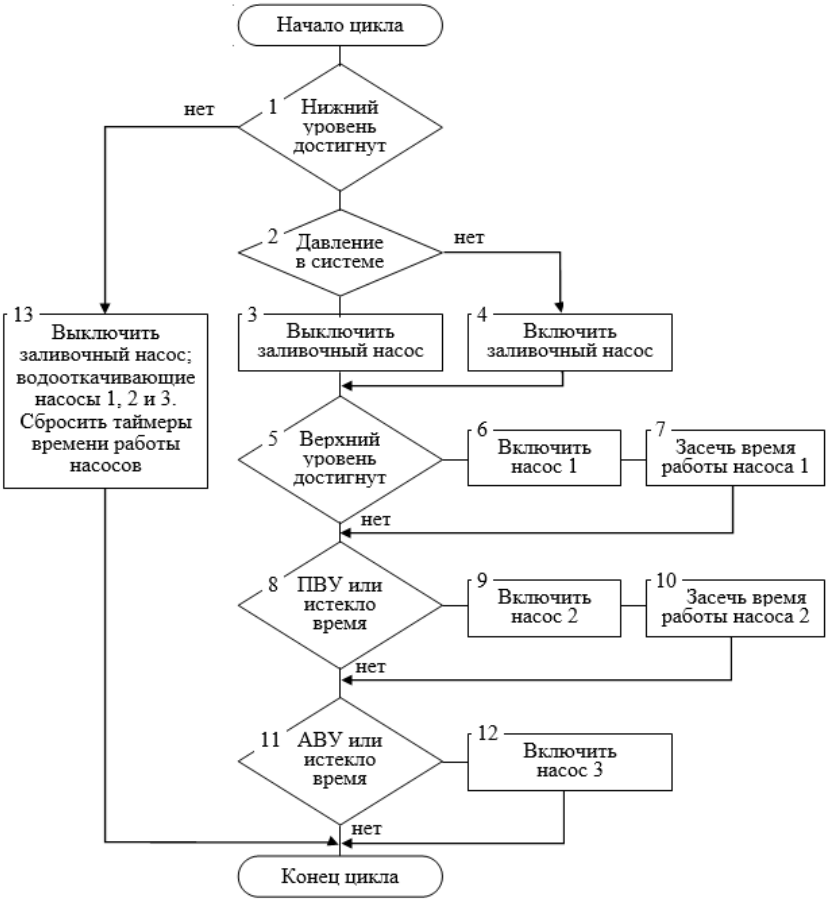


Рис. 121. Блок-схема АСУ водоотливом

Первое и главное преимущество ПЛК, обусловившее их широкое распространение, заключается в том, что одно компактное электронное устройство может заменить десятки и сотни электромеханических реле. Второе преимущество в том, что функции логических контроллеров реализуются не аппаратно, а программно, что позволяет постоянно адаптировать их к работе в новых условиях с минимальными усилиями и затратами.

Применение ПЛК обеспечивает высокую надежность, простое тиражирование и обслуживание систем управления, ускоряет монтаж и наладку оборудования, обеспечивает возможность быстрого обновления алгоритмов управления (в том числе и на работающем оборудовании).

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Парр, Э. Программируемые контроллеры : руководство для инженера / Э. Парр ; пер. с англ. – М. : БИНОМ. Лаборатория знаний, 2007. – 516 с.
2. Парк, Дж. Сбор данных в системах контроля и управления. Практическое руководство / Дж. Парк, С. Маккей. – М. : Группа ИДТ, 2006. – 504 с.
3. Парк, Дж. Передача данных в системах контроля и управления: практическое руководство / Дж. Парк, С. Маккей, Э. Райт ; [перевод с англ. В. В. Савельева]. – М. : Группа ИДТ, 2007. – 480 с.
4. Петров, И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / И. В. Петров // М. : СОЛОН-Пресс, 2010. – 256 с.
5. Олссон, Г. Цифровые системы автоматизации и управления / Г. Олссон, Д. Пиани. – СПб. : Невский Диалект, 2001. – 557 с.
6. Гируцкий, И. И. Компьютеризированные системы управления в сельском хозяйстве / И. И. Гируцкий, А. Г. Сеньков. – Минск : БГАТУ, 2014. – 221 с.
7. Микропроцессорная техника систем автоматизации. Лабораторный практикум : учебно-методическое пособие / сост.: И. И. Гируцкий, А. Г. Сеньков. – Минск : БГАТУ, 2017. – 136 с.
8. Технические средства автоматизации. курсовое проектирование : учебно-методическое пособие / сост. И. И. Гируцкий. – Минск : БГАТУ, 2020. – 64 с.
9. Иванов, В. Н. Программирование логических контроллеров : учебное пособие / В. Н. Иванов. – М. : Солон-пресс, 2020. – 356 с.
10. Минаев, И. Г. Программируемые логические контроллеры в автоматизированных системах управления / И. Г. Минаев и др. – 2-е изд., перераб. и доп. – Ставрополь : АГРУС, 2010. – 128 с.
11. Нестеров, К. Е. Программирование промышленных контроллеров : учебно-методическое пособие / К. Е. Нестеров, А. М. Зюзев. – Екатеринбург : Изд-во Уральского университета, 2019. – 96 с.: ил.

ДЛЯ ЗАМЕТОК

Учебное издание

Гируцкий Иван Иванович,
Сеньков Андрей Григорьевич

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА СИСТЕМ АВТОМАТИЗАЦИИ

Учебно-методическое пособие

Ответственный за выпуск *Н. М. Матвейчук*
Редактор *Д. О. Бабакова*
Корректор *Д. О. Бабакова*
Компьютерная верстка *Д. О. Бабаковой*
Дизайн обложки *Д. О. Бабаковой*

Подписано в печать 26.04.2022. Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Ризография.
Усл. печ. л. 13,02. Уч.-изд. л. 10,18. Тираж 99 экз. Заказ 24.

Издатель и полиграфическое исполнение:
учреждение образования
«Белорусский государственный аграрный технический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
№ 1/359 от 09.06.2014.
№ 2/151 от 11.06.2014.
Пр-т Независимости, 99–1, 220023, Минск.