Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

М. Е. Антипин
Ю.О.Лобода

Программирование промышленных контроллеров

Учебное пособие
для студентов технических направлений и специальностей

Томск
2023

УДК 004.02
ББК 3стд2-02
А 72

Рецензенты:
Мальчуков А.Н., доцент кафедры КИБЭВС ТУСУР, канд. тех. наук
Беккерман Е.Н., доцент кафедры информационных технологий в исследовании дискретных структур НИ ТГУ, канд. физ.-мат. наук

Авторы
М. Е. Антипин, Ю. О. Лобода

Антипин, Михаил Евгеньевич
А 72 Программирование промышленных контроллеров: Учебное пособие/ М.Е. Антипин, Ю.О. Лобода. – Томск: Томск. гос. ун-т систем упр. и радиоэлектронники, 2023. – 80 с.

Учебное пособие содержит необходимый объем теоретического материала по дисциплине «Программирование промышленных контроллеров». В пособии рассмотрены не только языки программирования промышленных контроллеров, но и особенности их алгоритмизации и использования в проектах автоматизации.

Для студентов высших учебных заведений, обучающихся по техническим направлениям и специальностям.

Одобрено на заседании кафедры УИ, протокол № 6 от 02.02.2023.

УДК 004.02
ББК 3стд2-02

Оглавление

Введение

Основой любого производства являются технологические процессы, проходящие на уровне промышленного оборудования. Задачей автоматизации технологических процессов является предоставление диспетчеру исчерпывающей информации для принятия решений, передача на оборудование команд телеуправления и обеспечение их выполнения. При этом оборудование размещается в цехах, а то и под открытым небом, где может быть шумно или сыро, холодно или жарко, а часто и небезопасно. А для работы диспетчера выделяется комфортное помещение, оборудованное системой контроля микроклимата и средствами коммуникации и управления.

Поскольку оборудование может быть установлено достаточно далеко от рабочего места диспетчера, то подключение датчиков напрямую к компьютеру или промышленному серверу нецелесообразно, а часто и невозможно. Традиционно, датчиком называется преобразователь измеряемой величины в электрический сигнал. Речь идет о небольших напряжениях до 10В и малых токах до 20 мА. Применение длинных шлейфов приводит к появлению помех и ослаблению сигналов, а также требует повышения мощности питания датчиков. Оптимальным является размещение АЦП в непосредственной близости от объекта мониторинга и преобразование электрического сигнала в цифровой. В настоящее время существует достаточно много датчиков с цифровым выходом, позволяющих передавать измеренное значение набольшие расстояния. Однако, стоимость таких датчиков на порядок выше аналоговых, а установка большого их количества на один канал передачи данных заметно замедляет процесс опроса. Поэтому наиболее целесообразным решением является размещение в непосредственной близости от оборудования устройства сбора передачи данных, задача которого оцифровать сигналы со всех аналоговых датчиков, считать сигналы с цифровых датчиков и передать их одним пакетом по цифровому каналу в диспетчерский пункт.

Осуществлять видеонаблюдение за контролируемыми объектами неэффективно. Во-первых, человек не может анализировать одновременно много видеоизображений, что существенно ограничивает количество управляемых объектов. Во-вторых, визуальное наблюдение не позволяет контролировать многие важные параметры технологического процесса, которые и являются ключевой информацией для принятия решения. Поэтому для определения состояния на оборудование устанавливаются датчики (сенсоры), определяющие значения параметров технологического процесса. Исходя из полученной информации, система идентифицирует состояние отдельных единиц технологического оборудования, а также технологического процесса в целом. Состояния контролируемого оборудования и значения важных технологических параметров выводятся на пульт управления. На текущий момент в роли универсального пульта управления выступает персональный компьютер диспетчера или терминал промышленного сервера.

Телеуправление оборудованием также является не совсем тривиальной задачей. Ведь подача управляющего напряжения на исполнительный механизм (актуатор) вовсе не гарантирует, что оборудование поведет себя именно так, как желает диспетчер. Наиболее часто реализуется принцип управления по отклонению, когда управляющая подсистема отслеживает поведение объекта и вносит соответствующие корректировки в управляющее воздействие. Алгоритмы формирования управляющего воздействия довольно сложны, а требования к времени реагирования часто стоят достаточно остро. Ведь неправильное поведение объекта может привести к производству бракованной продукции, порче технологического оборудования, а в некоторых случаях представлять опасность для персонала предприятия или жителей прилегающих районов. Такой актуатор, как, например, электрический привод обладает очень динамическим характером механических перемещений и переходных процессов. Соответствующие информационные сигналы нужно обработать очень быстро и сформировать управляющее воздействие. Поэтому управляющее устройство должно быть установлено в

непосредственной близости от привода. Нет смысла передавать отдельные значения угла поворота или тока нагрузки по каналам связи. Поэтому в ходе телеуправления диспетчер только указывает состояние, в которое нужно привести оборудование или процесс в целом, или желаемое значение технологического параметра, т.н. «уставку», а процесс формирования управляющего воздействия на основании анализа поведения объекта осуществляет управляющее устройство, расположенное в непосредственной близости от технологического оборудования.

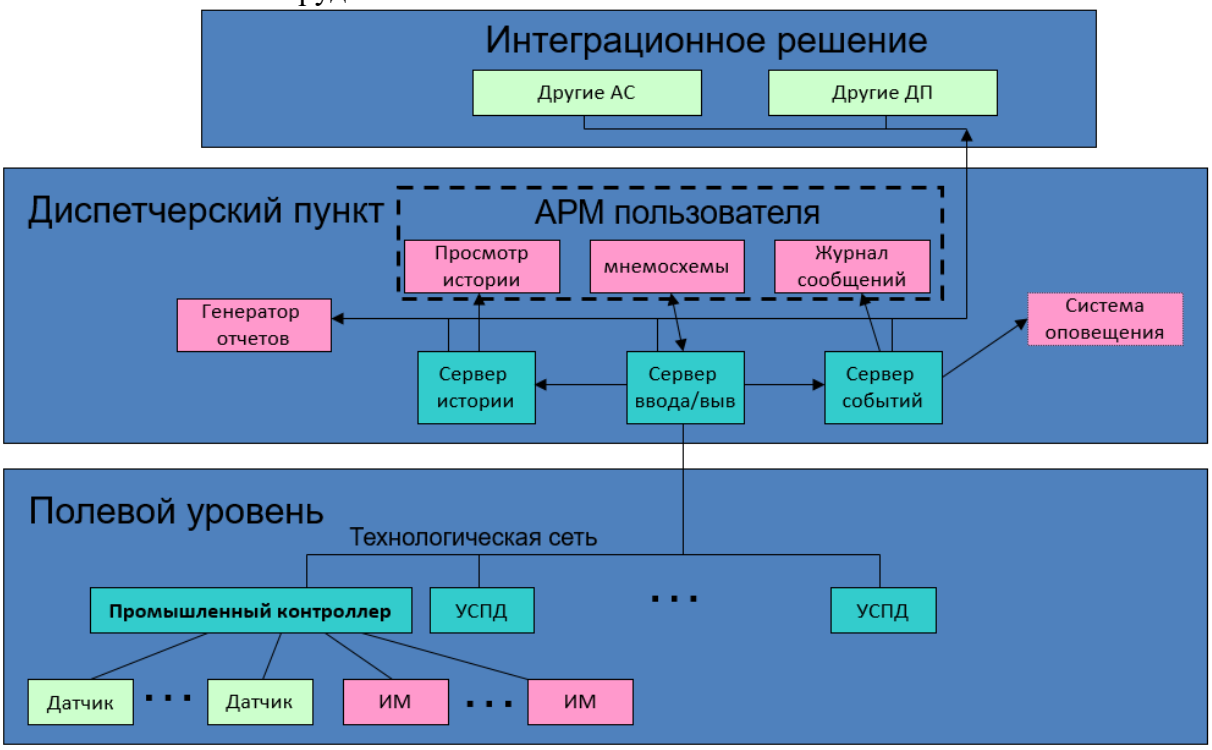


Рисунок В.1 – Общая схема проекта автоматизации технологических процессов

В большинстве проектов промышленной автоматизации функции опроса датчиков и выполнения алгоритмов управления реализуются в промышленных контроллерах. Исторически часто применяется название «программируемый логический контроллер» или PLC. Многие современные промышленные контроллеры обладают достаточной вычислительной мощностью и количеством внешних интерфейсов для подключения, чтобы управлять не одной единицей оборудования, а производственной линией, цехом или заводом целиком. При этом потребляемая мощность у промышленного контроллера заметно меньше, чем у классических компьютеров или промышленных серверов, что позволяет обеспечить непрерывность функций управления длительное время от источника бесперебойного питания.

Современные системы автоматизации на основе промышленных контроллеров работают по принципу управления данными. Т.е. обмен информацией и запуск алгоритмов управления осуществляются только тогда, когда имело место изменение данных, например, показаний датчиков. Это позволяет разгрузить каналы связи и перевести вычислительные мощности в режим экономии энергии, когда технологический процесс устойчив.

В ряде случаев система управления, построенная на промышленных контроллерах, способна управлять достаточно сложным промышленным объектом в автоматическом режиме, когда производственные задания поступают не от диспетчера, а от систем автоматизации более высокого уровня – MES, ERP. В этом случае роль персонала сводится к регламентному обслуживанию системы. К сожалению, приемлемо только для массового типа производства, не требующего частого переоснащения.

Коммуникационные возможности промышленных контроллеров позволяют строить сложные многоуровневые системы управления с применением эстафетной передачи данных и распределенными алгоритмами. Возможность концентрировать и агрегировать данные на каждом уровне позволяет меньше нагружать каналы коммуникаций и экономить кабельную продукцию. Модульная структура обеспечивает возможность использовать гетерогенные каналы коммуникации, легко переходить от проводных сетей к беспроводным.

Задачи автоматизации технологических процессов для разных предприятий отличаются масштабом, территориальным размещением, алгоритмами, доступными каналами передачи данных. Модульная структура и широкие возможности настройки промышленных контроллеров обеспечивают гибкость применяемых подходов к управлению. Для успешной реализации всех указанных задач требуются навыки конфигурирования и программирования промышленных контроллеров, формированию которых и посвящен этот курс.

Раздел 1. Архитектура промышленных контроллеров

Рассмотрение архитектуры промышленных контроллеров целесообразно начать от общей архитектуры устройств сбора-передачи данных (рис.1.1). Основой архитектуры является системная магистраль, состоящая из шины управления, шины адреса и шины данных. При рассмотрении промышленных контроллеров, имеющих преимущественно модульную структуру целесообразно включить в системную магистраль шину питания, состоящую из нескольких токопроводников и обеспечивающую подачу нескольких уровней питающего постоянного напряжения к элементам вычислительной системы – 3.3 В, 5 В, 12 В, 24 В.

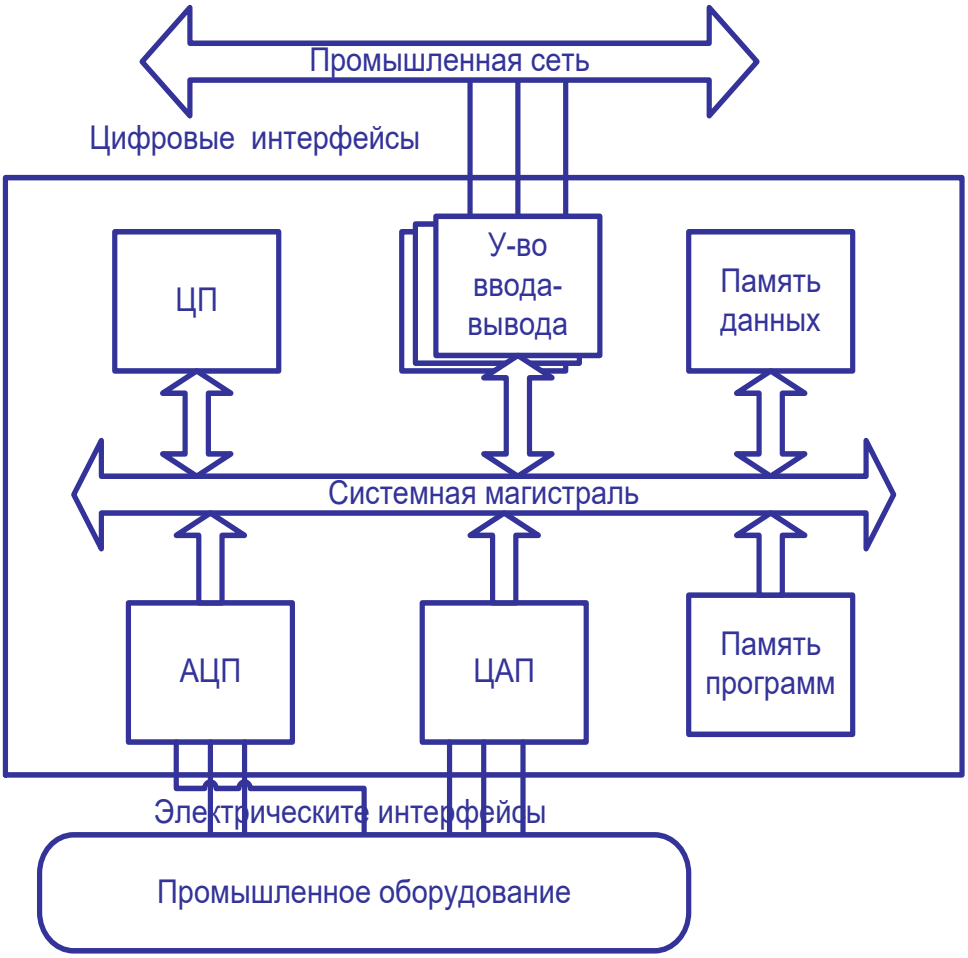


Рисунок 1.1 – Общая архитектура устройств сбора передачи данных

Остальные шины состоят из токопроводников, на которые одни устройства – элементы вычислительной системы подают уровень напряжения, соответствующий логическому 0 или 1, а другие считывают.

В состав вычислительной системы входят управляющие устройства: процессор, контроллер прерываний, контроллер прямого доступа к памяти. Управляющие устройства подают на шину управления команды для всех остальных элементов: устройств памяти и устройств ввода/вывода.

За каждым из устройств памяти и ввода-вывода закреплено адресное пространство. Адрес, к которому обращается процессор, устанавливается им на шине адреса. По команде с управляющей шины устройства считывают адрес и определяют, кому он принадлежит. Соответствующее устройство переходит в активное состояние и выполняет команду процессора.

Если процессор передает данные в устройство ввода-вывода или память (далее, устройство), то он выставляет соответствующие уровни на шине данных, а устройство их

считывает и помещает по заданному адресу. Если команда предполагает считывание данных из устройства, то устройство выставляет логические уровни на шине данных, а процессор их считывает. Количество токопроводников в шине данных определяет так называемую «разрядность» вычислительной системы. В значительной части вычислительных систем шина данных использует те же токопроводники, что и шина адреса с разделением по времени, которое называется временным мультиплексированием.

Устройства памяти делятся на постоянную и оперативную память. Постоянная, или, как еще говорят «энергонезависимая» память, предназначена для хранения программ (инструкций), а также сохранения результатов работы в виде файлов. Данные в энергонезависимой памяти сохраняются даже при отключении системы. Оперативная память используется для хранения значений, появляющихся в процессе работы системы. В нашем случае это могут быть сигналы с датчиков или поданные диспетчером команды (уставки) телеуправления. Также есть внутренняя память у процессора, в которой он хранит операнды и промежуточные результаты вычислений.

Устройства ввода-вывода являются средством общения вычислительной системы с внешним миром. Среди них можно выделить те, что наиболее востребованы в промышленных контроллерах. Это коммуникационные интерфейсы, такие как USB, UART и др, а также аналого-цифровые и цифро-аналоговые преобразователи (АЦП и ЦАП). Именно АЦП обеспечивает прием сигнала с аналоговых датчиков, а ЦАП – подачу электрических сигналов на исполнительные механизмы (актуаторы).

По приведенной выше архитектуре построены практически все вычислительные системы, применяемые в автоматизации. Но у промышленных контроллеров есть специфика, которую мы постараемся выделить, рассматривая сходные устройства сбора-передачи данных.

Наиболее часто промышленные контроллеры студенты путают с микроконтроллерами. В то же время различие между этими устройствами настолько существенны, что такая путаница недопустима. Микроконтроллер - это микросхема, в кристалле которой реализованы все элементы вычислительной системы – системная магистраль, процессор, память, АЦП, ЦАП, интерфейсы (рис.1.3).

В отличие от микропроцессора, который тоже представляет собой микросхему, «ножки» микроконтроллера способны принимать или передавать небольшие электрические сигналы. Это делает возможным прием сигналов с датчиков и передачу команд на актуаторы, в то время как процессор способен «общаться» только с другими элементами микропроцессорной системы посредством внешней системной магистрали. Микроконтроллер нельзя назвать самостоятельным устройством управления, так как для работы микросхемы, она должна быть смонтирована на какой-либо печатной плате, а ее информационные выводы (пины) должны быть на электрическом уровне согласованы с предполагаемой нагрузкой. Примером такой схемы может служить отладочная плата с микроконтроллером (рис. 1.4), которая используется для разработки и отладки устройств, собранных на базе микроконтроллера.

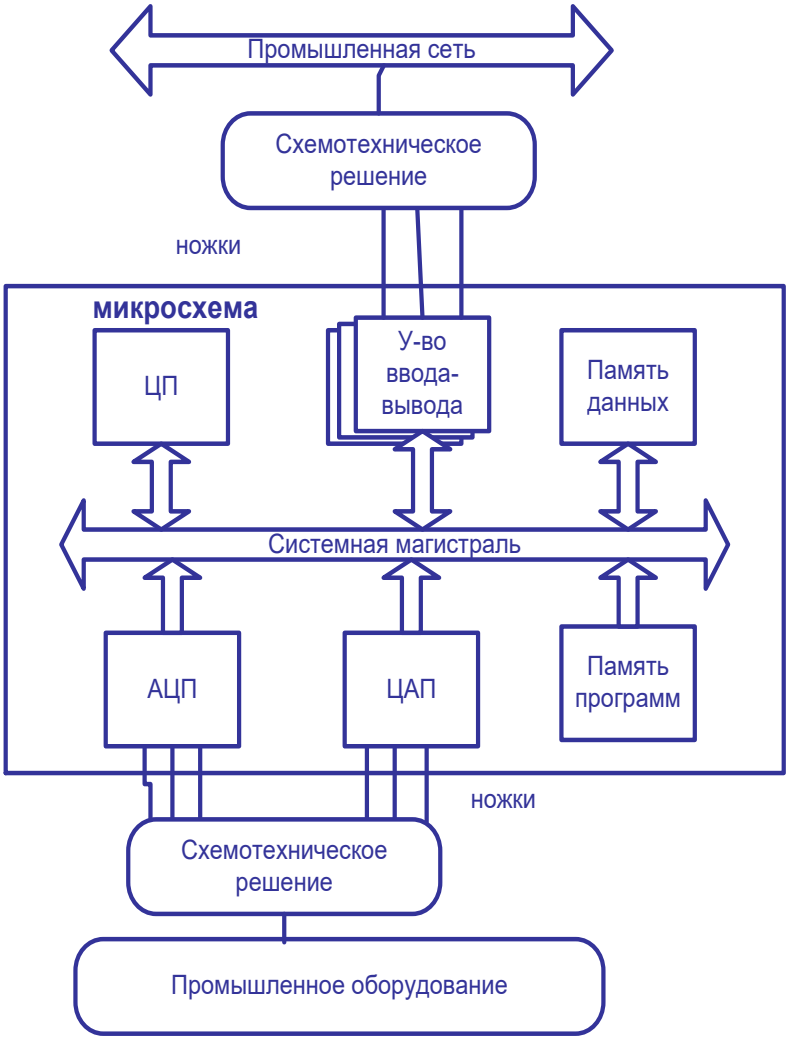


Рисунок 1.2 – Архитектура микроконтроллера

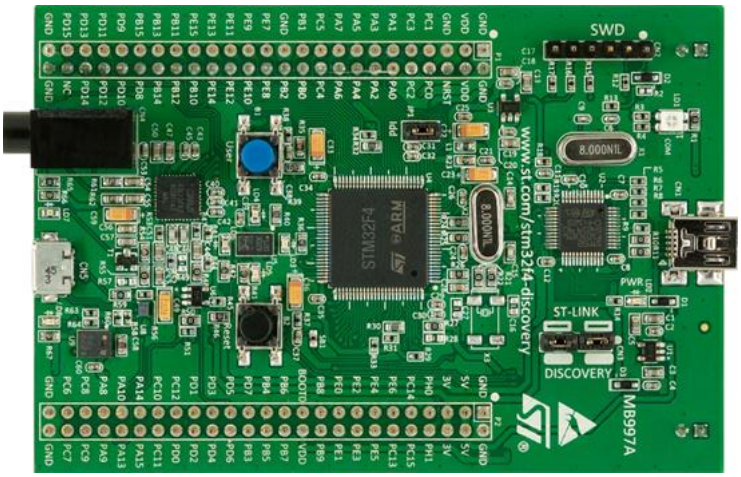


Рисунок 1.3 – Отладочная плата с микроконтроллером

Промышленный контроллер также является устройством на базе микроконтроллера, и даже, как правило, не одного. Но промышленный контроллер – это законченное устройство в корпусе, имеющее довольно мощные электрические входы и выходы. Микроконтроллер как правило оперирует напряжением не выше напряжения своего питания, т.е. 3,3 В или 5 В и токами до 5 мА. В большинстве промышленных контроллеров электрические интерфейсы рассчитаны на 12 В и 24 мА, что связано со стандартами в области промышленных датчиков. Но есть промышленные контроллеры,

способные измерять и выдавать куда большие напряжения и токи, которые используются в области электроэнергетики.

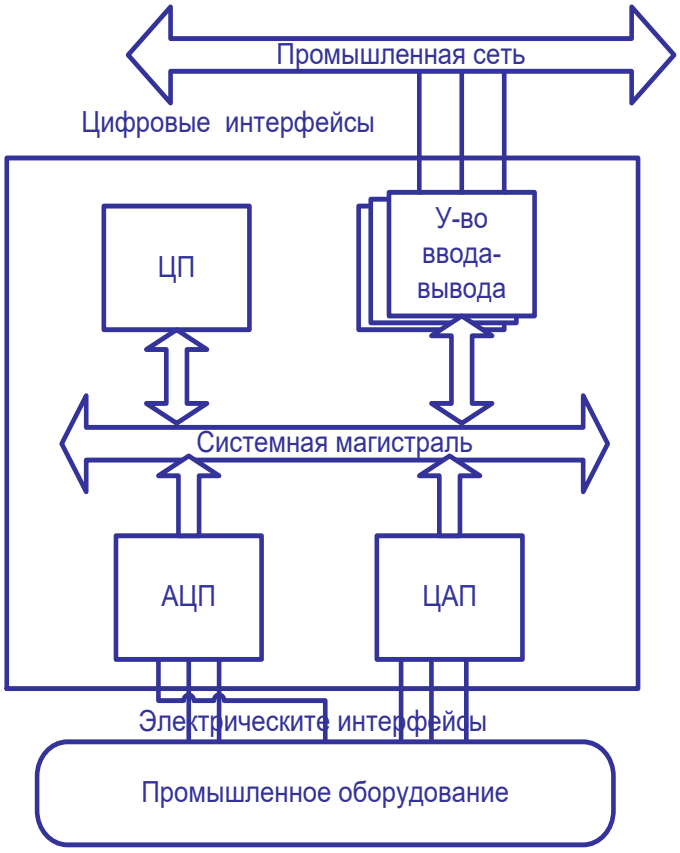


Рисунок 1.4 – Архитектура АПКП

Следующий тип микропроцессорных систем, довольно широко применяемых в автоматизации – адресуемые приемно-контрольные пункты (АПКП) архитектура которых приведена на рис. 1.5. Это устройства в корпусе с несколькими клеммными колодками для подключения датчиков и исполнительных механизмов, а также интерфейсными разъемами для подключения к промышленным сетям передачи данных. Эти устройства играют роль концентраторов. Сигналы с датчиков поступают на клеммы, оцифровываются и могут быть по запросу отправлены в центр диспетчерского управления. Адресное пространство таких устройств фиксированное. Логическая обработка не осуществляется.

Интеллекутальные реле очень похожи на АПКП, но назначение их другое. Они могут не подключаться к коммуникационным интерфейсам. Как правило, интеллектуальные реле замыкают или размыкают силовую цепь по условиям, зависящим от входных сигналов с датчиков, проходящих обработку по несложному встроенному алгоритму. В последнее время можно отметить рост популярности интеллектуальных реле с беспроводным WiFi интерфейсом применяемых для управления бытовыми приборами и другими токоприемниками по интернету вещей (IoT).

Рисунок 1.5 – Интеллектуальное реле

Особый класс контроллеров – телемеханизированные приборы учета (рис. 1.6). Собственно, данные приборы не являются управляющими устройствами, т.к. не передают команды управления на актуаторы. Все получаемые команды предназначены для управления самим устройством, например, для передачи показаний в центр управления. Зато данные устройства сертифицированы как средство измерения и имеют энергонезависимый буфер для хранения показаний учета.

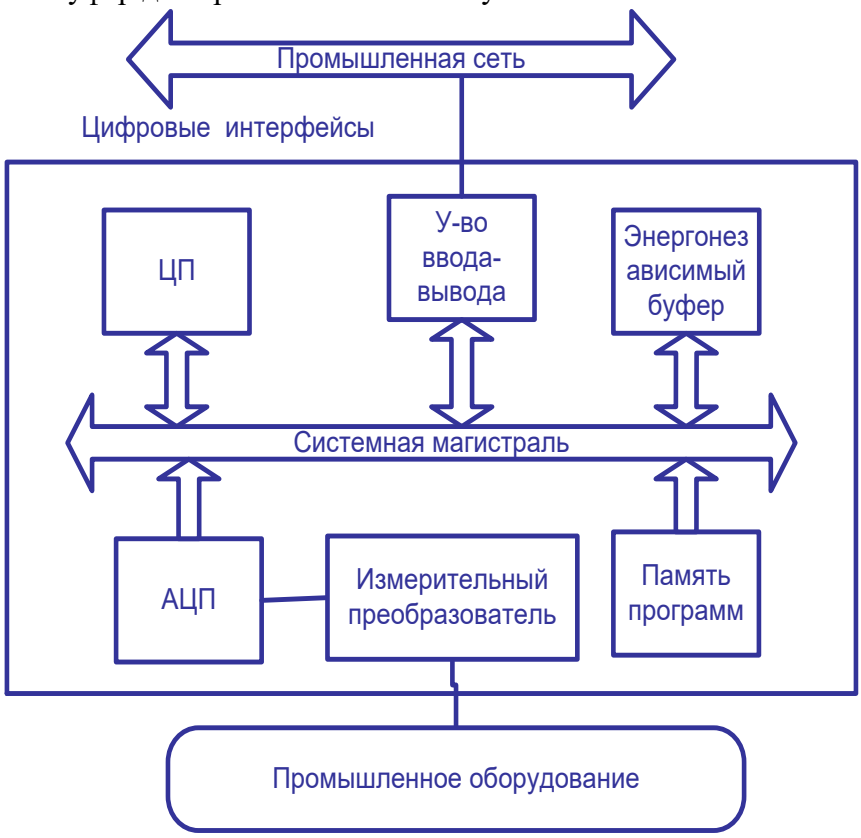


Рисунок 1.6 – Архитектура приборов учета

Отдельно стоит упомянуть регуляторы параметров (рис.1.7). Внешне напоминающие интеллектуальное реле или выносной модуль промышленного контроллера, эти самостоятельные устройства устанавливаются рядом с объектом, или непосредственно на объект и реализуют сложный и быстрый алгоритм управления, в основном ПИД-регулирования параметра. Содержать канал для измерения одного или

нескольких параметров, ЦАП для подачи управляющего напряжения на объект и интерфейс для задания уставки регулирования и коэффициентов регулятора. Некоторые интеллектуальные устройства такого класса способны к самонастройке параметров регулятора.

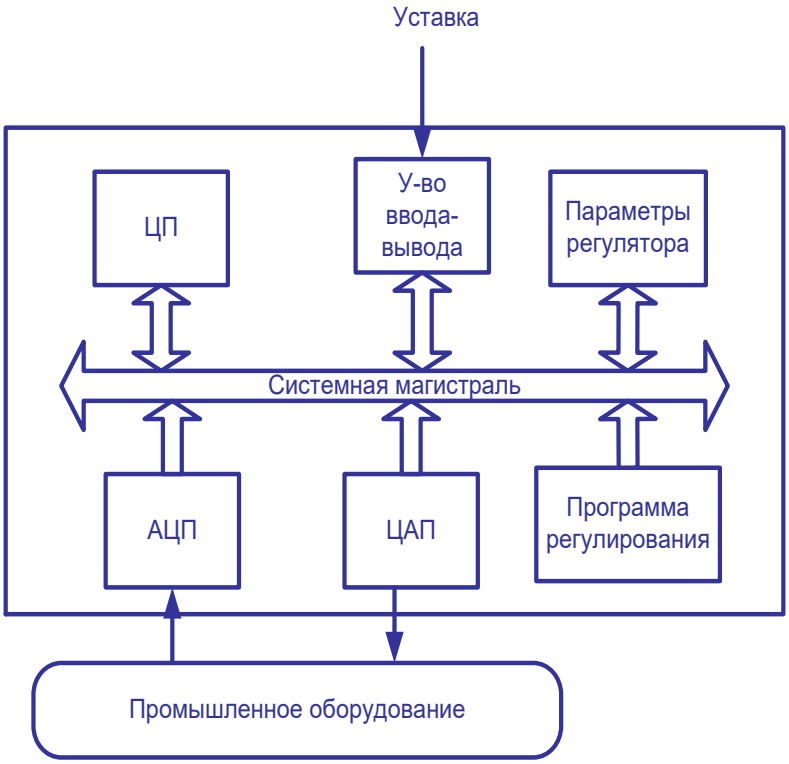


Рисунок 1.7 – Архитектура регулятора

В отличие от упомянутых вариантов управляющих устройств, промышленный контроллер помимо сбора и передачи информации от первичных источников, обеспечивает выполнение пользовательских программ. Поскольку промышленный контроллер расположен непосредственно рядом с объектом, то это позволяет ему осуществлять не только первичную обработку информации, но и быстродействующие алгоритмы автоматической защиты и регулирования, а также создание полностью автоматических систем управления.

Характерные признаки PLC:

- Модульная структура, позволяющая менять мощность питания, количество интерфейсов для связи с первичными источниками и технологической сетью;
- Операционная система реального времени, включающая среду исполнения пользовательских задач;
- Интерфейс для подключения мобильных инструментов тестирования и обслуживания (переносных пультов);

Также значимой частью промышленных контроллеров является система программирования пользовательских задач, которая устанавливается на персональный компьютер. В сам контроллер помещается исполняемый модуль.

В рамках модульной архитектуры микроконтроллер и процессор входят практически в каждый модуль. Организация опроса модулей осуществляется по шине, аналогичной системной магистрали вычислительной системы. Роль управляющего устройства играет модуль центрального процессора, в котором исполняется пользовательская задача. Каждый модуль контроллера имеет набор входных и/или выходных сигналов, конфигурационных параметров и сигналов состояния самого модуля. Модуль центрального процессора собирает выходные сигналы и сигналы состояния с каждого модуля, обрабатывает их в пользовательской задаче, формирует входные сигналы и параметры модулей, и отправляет их по системной магистрали.

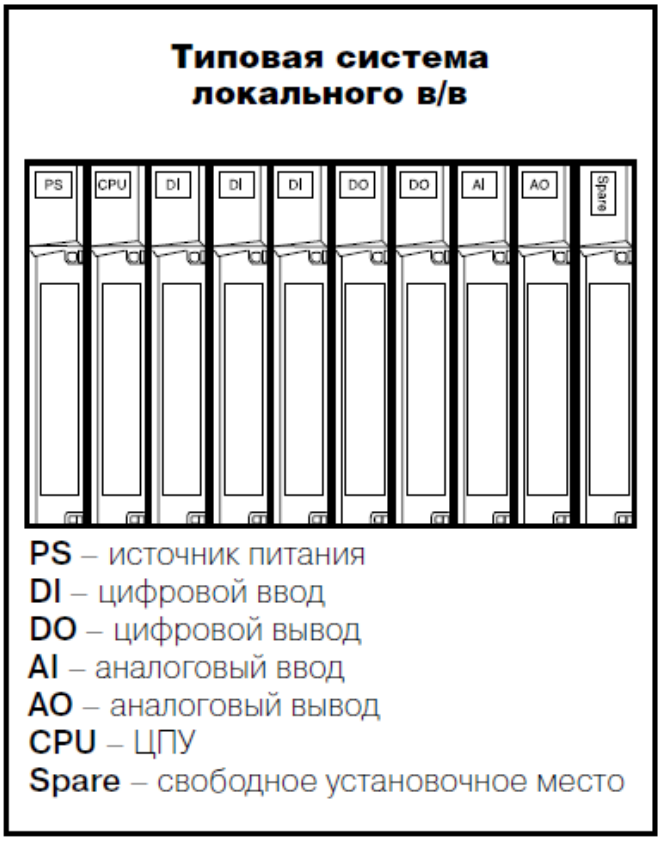


Рисунок 1.8 – Модульный промышленный контроллер

Как правило, модули контроллера устанавливаются локально на панель (крейт) или DIN-рейку. Однако, в ряде случаев промышленный контроллер может включать выносные модули, размещенные на значительном расстоянии от модуля центрального процессора. В этом случае опрос модулей осуществляется по коммуникационным интерфейсам. Например RS-485 или Industrial Ethernet.

Рисунок 1.9 – Промышленный контроллер с выносными модулями

Для выбора контроллера в конкретном проектном решении следует опираться на следующие характеристики:

количество и разнообразие уровней обрабатываемых сигналов (дискретных, аналоговых, значений токов и напряжений);

коммуникационные возможности – реализованные проводные и бепроводные интерфейсы передачи данных;

способы подключения устройств (разъемы, пружинные и винтовыеклеммники);

совместимость с другим оборудованием, применяемым в проекте;

характеристики процессора;

объем памяти;

класс защиты от внешних воздействий;

показатели надежности;

стоимость.

Промышленный контроллер – универсальный инструмент для автоматизации производства, позволяющий оптимизировать ресурсы, обеспечить гибкую настройку и масштабирование, управлять надежностью и скоростью реакции.

Раздел 2. Типы промышленных контроллеров и их модулей

Промышленные контроллеры различаются по задачам и характеристикам. Главные характеристики промышленного контроллера:

- Количество обслуживаемых точек ввода-вывода. Эта характеристика определяет масштаб объекта, которым может управлять контроллер.
- Длительность цикла обработки. Фактически определяет время реакции контроллера на входные значения. Данный параметр может сильно изменяться в зависимости от нагрузки. Поэтому для задач реального времени важно иметь возможность оценить его в конфигурации, обеспечивающей решение практической задачи.
- Доступные каналы коммуникации. Обеспечивают возможность включения контроллера в существующую или проектируемую систему, его эффективное взаимодействие с другими элементами.
- Число подключаемых модулей. Определяет возможность организации централизованного управления производственным участком или цехом, предприятием в целом.
- Возможности резервирования. Поскольку промышленный контроллер является ключевым звеном в цепочке автоматизации, то мерам обеспечения его надежности уделяется большое внимание. Могут резервироваться как контроллеры целиком (дублирование), так и отдельные части: питание, модуль центрального процессора.
- Возможность «горячей» замены модулей, вышедших из строя. Поскольку промышленные контроллеры на некоторых объектах работают без остановки годами, возможность ремонта без отключения контроллера целиком часто позиционируется производителями как важное преимущество.

По выполняемым задачам промышленные контроллеры подразделяются на следующие типы:

Специализированные контроллеры. Например, контроллер управления перемещением, температурный контроллер, контроллер микроклимата, контроллер пожарной сигнализации. Область применения: локальное управление технологической установкой или механизмом. Как правило, имеют маломощный процессор и встроенные специальные функции, позволяющие эффективно выполнять задачу.

Рисунок 2.1 – Специализированный контроллер для управления отоплением

Контроллеры противоаварийной защиты (ПАЗ). Очень быстродействующие контроллеры с простой логикой работы, обеспечивающие своевременное отключение оборудования в случае возникновения аварийной ситуации. Поскольку применяются на

пожаро- и взрывоопасных объектах химической, нефтехимической и нефтеперерабатывающей промышленности, как правило, имеют взрывозащищенное исполнение.

Рисунок 2.2 – Контроллер ПАЗ

Контроллеры общепромышленного назначения. Обеспечивают достаточное количество каналов для опроса сигналов с аналоговым и цифровым выходом, обладают широкими возможностями математической и логической обработки, поддерживают языки программирования. Применяются для автоматизации станков, производственных линий, машин, агрегатов, конвейеров.

Рисунок 2.3 – Контроллер общепромышленного назначения

Коммуникационные контроллеры. Обладают значительным количеством интерфейсов для подключения к каналам передачи данных. Осуществляют концентрацию данных из локальных источников для передачи по высокоскоростным интерфейсам. Обеспечивают создание систем управления территориально распределенным производством, таким как, линии электропередач, трубопроводный и железнодорожный транспорт.

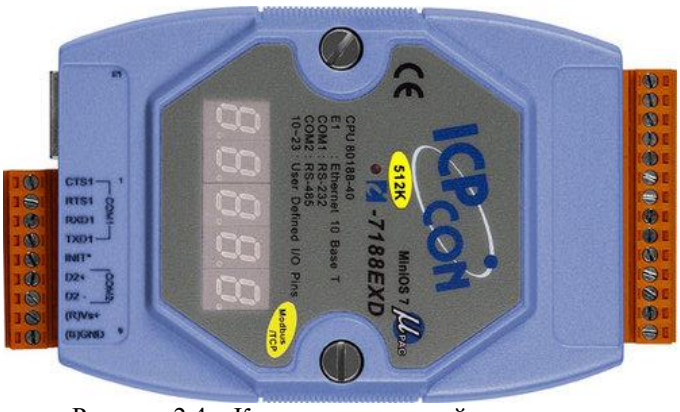


Рисунок 2.4 – Коммуникационный контроллер

По конструктивному исполнению контроллеры бывают:
Панельные – монтаж осуществляется на дверце шкафа или на панели;

Рисунок 2.5 – Панельный контроллер

DIN-реечные - монтаж осуществляется внутри шкафа на DIN-рейку;

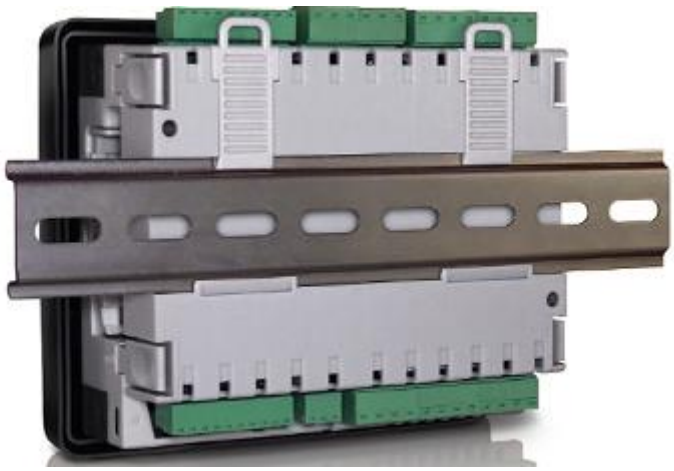


Рисунок 2.6 - DIN-реечный ПЛК

Стоечные – монтаж осуществляется в стойке;

Рисунок 2.7 – Стоечный контроллер

Бескорпусные – применяется производителями оборудования OEM (Original Equipment Manufacturer) в специализированных конструктивах.

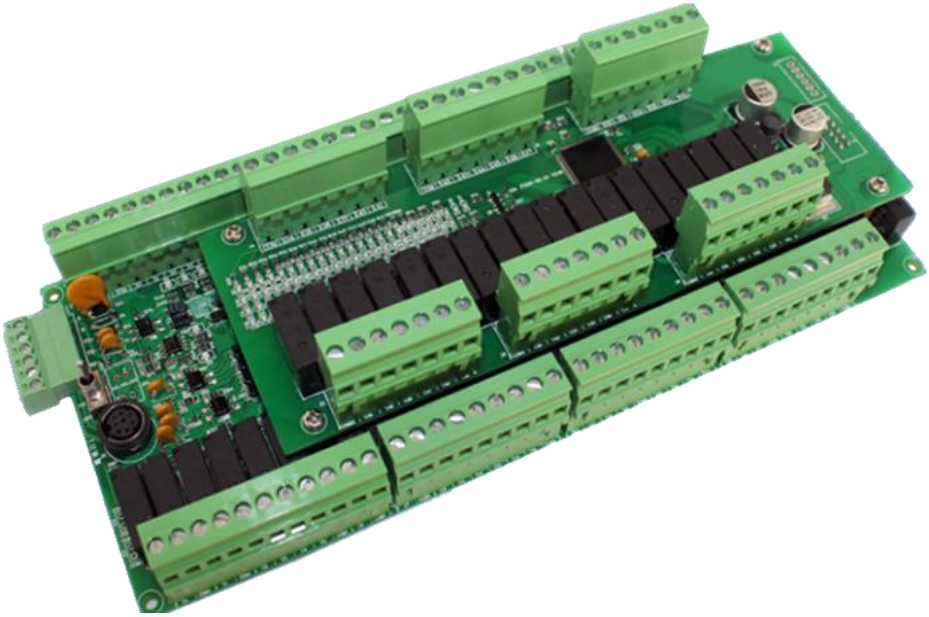


Рисунок 2.8 – Бескорпусной ПЛК

В зависимости расположения модулей ввода-вывода промышленных контроллеров различают:

Моноблочные ПЛК – осуществление удаления или замены модулей ввода-вывода невозможно. Конструкция промышленного контроллера представляет единый цельный корпус с устройствами ввода-вывода.

Рисунок 2.9 – Моноблочный ПЛК

Модульные ПЛК – смена модулей возможна. Конструкция представляет собой общую корзину с модулем центрального процессора и сменными модулями ввода-вывода. За выбор состава модулей, в зависимости от поставленных задач, отвечает проектировщик АСУ ТП.

Рисунок 2.10 – Модульный ПЛК

Распределенные ПЛК – модули ввода-вывода вынесены за пределы контроллера, выполняются в спецкорпусах и соединяются с контроллером при помощи промышленной сети с использованием интерфейсов, таких как например RS-485. Модули могут быть расположены на значительном удалении от самого промышленного контроллера.

Рисунок 2.11 – Распределенная конфигурация контроллера

Кроме уже описанного модуля центрального процессора в состав промышленных контроллеров могут входит еще ряд функциональных модулей: модуль питания, модули ввода-вывода дискретных и аналоговых сигналов, коммуникационные модули.

Модуль питания. Служит для преобразования напряжения питающей сети, как правило, переменного тока, в необходимые для работы постоянные напряжения 3,3 В, 5 В, 12 В, 24 В с током, достаточным для питания всех модулей контроллера. Зачем питание делать отдельным модулем, да еще и включать его в системную магистраль? Да ведь мы не знаем заранее, сколько модулей будет в том или ином проекте, а увеличение мощности при относительно низких напряжениях – дорогое удовольствие. Ведь протекающие токи достаточно велики и требуется запас как по диаметру проводников, так и по используемым элементам. Поэтому дешевле поставить параллельно несколько маломощных модулей питания, если это требуется, чем применять один мощный моноблок. Пожалуй, это единственный модуль контроллера, который может обойтись без процессора. Но и тут не все так просто. Ведь требуется поддерживать требуемые характеристики качества выходного напряжения, пр плавающем входном и переменной нагрузке. А кроме того на модуль питания часто ложатся функции обеспечения резервного питания от аккумулятора, который нужно заряжать при наличии основного питающего напряжения. Т.е. модуль может играть роль контроллера заряда.

Модули дискретного ввода. Дискретными входными сигналами в распределенных системах управления являются сигналы коммутации кнопок, концевых выключателей, контактов реле и пр. Модули дискретного ввода представляют собой устройства преобразования двоичных сигналов логических уровней 1 и 0 [11]. Этим уровням соответствует напряжение на замкнутом или разомкнутом ключах. Величина напряжения может быть различной, но чаще используется напряжение 12, 24, 36, 48, 125 или 250 В.

Основными характеристиками каналов дискретного ввода являются:

- число каналов дискретного ввода. Обычно кратно 8: 16, 32, 64, 128;
- напряжение канала дискретного ввода;

- индикация состояния канала. Обычно светодиодом, если логическая 1;
- гальваническое разделение отдельных каналов. Как правило используется групповое разделение по 8 сигналов в группе.

Гальваническая развязка сигналов осуществляется за счет передачи сигнала по оптопаре от светодиода к фотодиоду. Поскольку таким образом можно передать только цифровой сигнал, гальваническая развязка на оптопаре ставится после аналого-цифрового преобразователя. Стоимость модулей с гальванической развязкой выше, но увеличение точности за счет исключения взаимовлияния сигналов полностью оправдывает эти расходы.

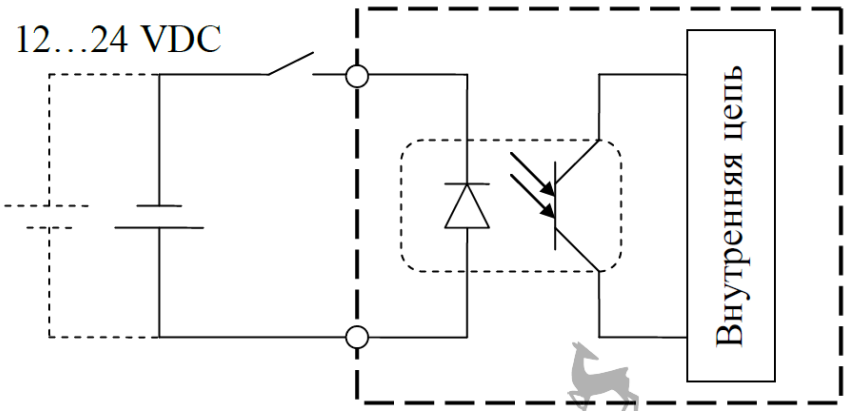


Рисунок 2.12 – Схема дискретного ввода

Режимы работы модуля:

нормальные дискретные входы (по умолчанию);

запирающиеся входы используются, чтобы обеспечить восприятие особенно коротких импульсов, длительность которых меньше, чем время рабочего цикла ПЛК. Функция «защелки» воспринимает импульс так, чтобы он мог быть обработан в течение следующего цикла без прерывания цикла работы ПЛК;

вызванные событием входы. Принцип управления по событиям позволяет событиям, сгенерированым различными источниками, быть учтенными и обработанными максимально быстро (например, превышение пороговых значений на счетных модулях и т. п.). Управляющие события являются внешними. Появление управляющего события вызывает прерывание выполняемого приложения и приводит к выполнению действий, связанных с определенным каналом;

входы прямого, обратного или реверсивного счетчика для подсчета импульсов;

вход Run/Stop. Функция Run/Stop используется для того, чтобы запустить или остановить выполнение прикладной программы при помощи специально сконфигурированного дискретного входа.

Модули дискретного вывода. в качестве дискретных выходных сигналов служат сигналы управления магнитными пускателями, реле, сигнальными лампами, исполнительными механизмами и др. Основными характеристиками модуля дискретного вывода являются:

- число каналов дискретного вывода. Обычно кратно 8: 16, 32, 64, 128;
- напряжение канала дискретного вывода;
- характеристика канала дискретного вывода – релейный («сухой контакт» – СК) или транзисторный вывод («открытый коллектор» – ОК);
- индикация состояния канала. Обычно светодиодом, если логическая 1;
- выходной ток канала дискретного вывода;
- защита модуля от короткого замыкания, перегрузок, скачков напряжения в управляемой цепи с последующей автоматической активацией вывода.

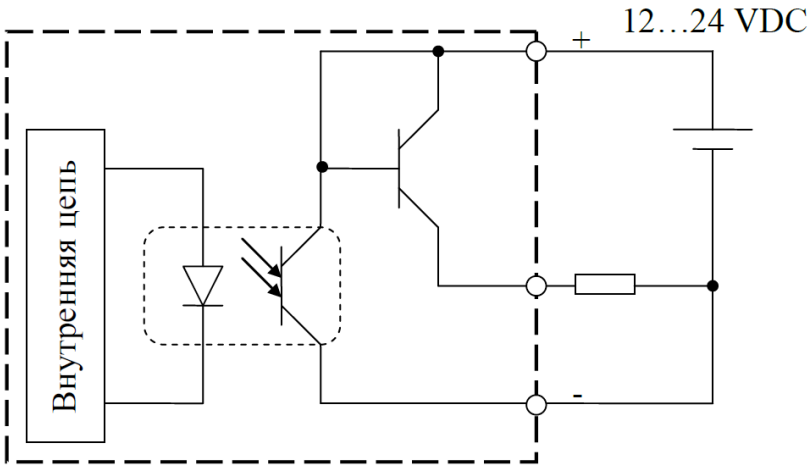


Рисунок 2.13 – схема дискретного вывода типа ОК

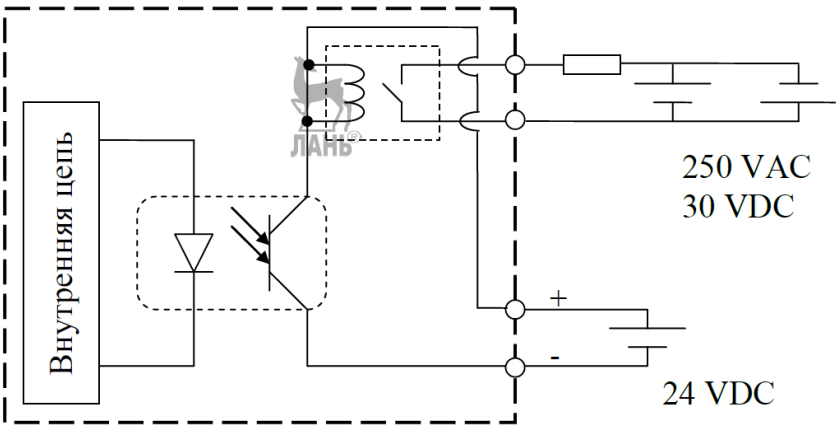


Рисунок 2.14 – схема дискретного вывода типа СК

Дискретные выходы, помимо своих стандартных функций, могут быть сконфигурированы для выполнения функции «Тревога» (Alarm), которая устанавливает выход в «0» при появлении блокирующей ошибки в программе или неисправности ПЛК.

Модули некоторых промышленных контроллеров могут совмещать функции дискретного ввода и вывода. При этом адресные пространства и разъемы для подключения сигналов ввода и вывода не смешиваются. Такие модули могут реализовывать несложное логическое управление без участия модуля центрального процессора. Например, отключить устройство при поступлении сигнала с датчика. Это приводит к заметному уменьшению времени реакции контроллера на такие сигналы.

Модули аналогового ввода. Осуществляют аналого-цифровое преобразование сигналов с аналоговых датчиков, предполагающих измерение параметров технологического процесса, таких как давление, вибрация, температура и др. Основными характеристиками модуля являются:

- число каналов ввода. Обычно 2, 4, 8;
- разрядность преобразования, отвечающая за точность преобразования и сказывающаяся на типе данных выходных сигналов модуля (однобайтные для 6-8 разрядного преобразования, 2 байтные для 12-16 разрядного, 4 байтные для 24-разрядных АЦП);
- диапазон входного напряжения и тока;
- быстродействие, зависящее не только от мощности установленного контроллера но и от типа АЦП, применяемого фильтра нижних частот, времени усреднения сигнала;
- гальваническая развязка каналов;

- подавление помех.

Физически входной аналоговый сигнал представляет собой сигнал напряжения или тока, который меняется во времени пропорционально значению измеряемого параметра. Наиболее распространенными стандартными сигналами являются следующие сигналы:

сигналы напряжения: 0...5 В, 0...10 В, 1...5 В, ±5 В,±10 В.

сигналы тока: 0...5 мА, 0...20 мА, 4...20 мА.

Сигнал 4...20 мА имеет преимущество по сравнению с остальными токовыми сигналами, так как позволяет определить разрыв линии при нулевом входном токе модуля, тогда как в других случаях ноль может соответствовать минимальному значению измеряемого параметра, и обрыв линии не очевиден.

Также можно уверенно говорить о предпочтительности датчиков с токовым выходом, поскольку часть напряжения может падать на шлейфе, которым датчик подключается к модулю. А ток во всей цепи остается постоянным. Таким образом, ток меньше подвержен влиянию помех.

Модули аналогового вывода. Применяется для преобразования числовой уставки в управляющее напряжение или ток при помощи цифро-аналогового преобразователя (ЦАП). Характеристики таких модулей сходны с модулями аналогового ввода.

Модули аналогового ввода-вывода совмещают в себе каналы ввода и вывода, а также реализуют встроенный ПИД-регулятор, т.е. формируют выходной сигнал по разнице между входным сигналом и уставкой, которую задает модуль центрального процессора. Коэффициенты ПИД-регулятора являются параметрами работы модуля, задаются в конфигурации и могут быть скорректированы в процессе работы.

Коммуникационные модули. Коммуникационный модуль предназначен для обмена данными между контроллером и внешними устройствами по вычислительным сетям передачи данных. В качестве внешних устройств выступают интеллектуальные терминалы, промышленные компьютеры, контроллеры, модули удаленного ввода-вывода и др. Характеристиками коммуникационных модулей являются допустимая скорость обмена (бит/с), число обслуживаемых устройств обмена данными, количество портов ввода-вывода, напряжение гальванической изоляции, напряжение питания, диапазон рабочих температур и ряд других.

Раздел 3. Типы данных в промышленных контроллерах

Промышленный контроллер оперирует различными сигналами, которые отличаются по назначению, способам обработки и источникам данных. Входящие сигналы поступают в котроллер с датчиков различных типов или из каналов промышленной коммуникации в соответствующие модули контроллера. Исходящие сигналы формируются по алгоритмам, заданным в пользовательской задаче или конфигурации контроллера, и передаются исполнительным механизмам или отправляются по каналам связи другим станциям.

Телесигнализация. Сокращенно «ТС». Тип данных Boolean. Входящий сигнал для модуля дискретного ввода и исходящий для коммуникационных модулей. Приставка «теле» означает удаленный, дистанционный. Т.е. это сигналы удаленной сигнализации. По сути ТС – это сигнал с дискретных датчиков, т.е. таких, которые имеют два состояния – 0 или 1. Такие датчики как правило отображают состояние объекта или регистрируют важные изменения в технологическом процессе. В качестве приемника сигналов ТС выступает модуль дискретного ввода.

Поскольку сигнализация несет важную информацию о технологическом процессе, то ее получение, обработка и отправка в диспетчерские пункт осуществляется с высоким приоритетом. Поскольку размер информации у сигналов ТС всего один байт, то сигналы ТС объединяются в группы, как правило, по 8, т.е. размером 1 байт. Стандарт IEC 60870-5 описывающий набор телемеханических протоколов, рекомендует для уменьшения объема передаваемой информации, отправлять данные I класса, т.е. только те, которые изменились. Группа ТС считается изменившейся, если в ней изменился хоть один сигнал.

Телесигнализация может быть также сформирована при логической обработке данных в контроллере. Например, при превышении значением с аналогового датчика заданного порога, или при обработке ответа исполнительного механизма на команду управления. Такие логические ТС конечно формируются медленнее, чем сигналы с дискретных датчиков, но обладают таким же высоким приоритетом при отправке по каналам связи.

Телеуправление. Сокращенно «ТУ». Тип данных Boolean. Входящий сигнал для коммуникационных модулей и исходящий для модуля дискретного вывода. Команды ТУ, полученные от коммуникационных модулей, как правило, формируются диспетчером. Иногда другими участниками процесса управления. Обычно входящие команды ТУ передаются модулю дискретного вывода. Но некоторые команды предназначены для управления контроллером в целом или отдельными его модулями. Например, для перезагрузки контроллера, прошивки нового программного обеспечения, сброса запирающих сигналов ТС, работающих по принципу «защелки».

По сути сигнал ТУ является командой включения или выключения чего-либо. Для обработки ТУ в сигнал должно поступить значение 1 (*true*). После отправки по каналам телемеханики сигнал ТУ сбрасывается в 0 (*false*). ТУ имеет наивысший приоритет в каналах передачи данных. Для отправки ТУ циклический опрос станций останавливается. Для передачи по коммуникационным каналам ТУ объединяют в группы, размером 1 байт. Отправка группы ТУ осуществляется при изменении хотя бы одного сигнала. Получающая сторона подтверждает получение ТУ путем отправки группы ТУ без изменений ответным кадром.

Модуль дискретного вывода преобразует сигнал в управляющее напряжение логической единицы и подает его на исполнительный механизм. Сброс сигнала нужно предусмотреть в пользовательской задаче контроллера.

Реакция исполнительного механизма на команду телеуправления может быть импульсной или непрерывной. Примером непрерывного телеуправления является классическое электромагнитное реле. Пока ТУ имеет значение 1, на обмотку подается управляющее напряжение и контакты замыкаются. Как только ТУ принимает состояние

0, напряжение снимается и контакты размыкаются. Примером импульсного телеуправления может служить магнитный пускатель. После подачи напряжения на пусковую кнопку, он переходит во включенное состояние и будет находится в нем, пока не будет подано напряжение на кнопку отключения, вне зависимости от дальнейшего поведения сигнала ТУ. В случае импульсного телеуправления, промышленный контроллер работает в более экономичном режиме, поскольку ему не нужно поддерживать управляющий ток в катушке реле. Но для управления таким объектом требуются не одно ТУ, а два – на включение и на выключение. Также импульсное управление устойчиво к появлению дребезга в контактах.

Некоторыми телемеханическими протоколами предусмотрен ответ на телеуправление (ОТУ). По сути это сигнал ТС, формируемый успешным или неуспешным прохождением ТУ по каналам телемеханики. Но в отличие от классического ТС сигнал ОТУ имеет наивысший приоритет, как и ТУ.

Телеизмерения. Сокращенное название «ТИ». Тип данных – числовой. Представляют с собой оцифрованные сигналы с аналоговых датчиков. Датчик является преобразователем измеряемой физической величины в электрический сигнал. Сигнал на выходе АЦП представляет собой целое неотрицательное число. В зависимости от разрядности АЦП результат оцифровки может быть записан в формат Byte (6-8 разрядные), Word (12-16 разрядные), DWord (24 разрядные). Для обозначения соответствующих типов сигналов используются сокращения ТИ1, ТИ2 и ТИ4 соответственно, где цифра означает размер данных в байтах. При программировании контроллеров следует учитывать, что на выходе АЦП тип беззнаковое целое. Попытка преобразовать его в знаковый тип того же размера, например, преобразование DWord в Integer может привести к переполнению и искажению значения. Предпочтительно преобразовать это значение, называемое часто «физическим», в инженерные единицы измерения исходной величины.

Особняком стоит тип телеизмерений ТИ4Float, где используется значение с плавающей точкой, которое является результатом преобразования измеренных значений в инженерные единицы. Такой сигнал может приходить, например, с интеллектуальных датчиков или с другого контроллера.

Телеизмерения являются входящими сигналами через модуль аналогового ввода и исходящими для коммуникационных модулей. Данные телеизмерений в телемеханических протоколах имеют низкий приоритет, но при этом имеют большой объем данных. В отличие от ТС сигналы ТИ могут меняться часто. Если отправлять все изменения, то это приведет к формированию большого количества данных I класса, что обязательно скажется на удлинении цикла опроса. А между тем, далеко не все изменения стоит передавать. Например, при измерении температуры окружающей среды нет смысла передавать значения чаще чем раз в минуту, а также регистрировать изменения меньше половины градуса. В этом случае в пользовательской задаче или в настройках модуля аналогового ввода задается минимальное значение изменения и/или минимальный интервал времени, прошедшего с момента последнего изменения.

Также в отношении сигналов ТИ активно применяется интегрирование значения по времени с целью фильтрации шумов. Интегрирование может осуществляться как на уровне модуля аналогового ввода, так и в пользовательской задаче. В пользовательской задаче также может быть сформирован цифровой фильтр нижних частот (ФНЧ), в том числе рекурсивный.

Телерегулирование. Сокращенное название «ТР». Тип данных числовой. Значения телерегулирования передаются на аналоговые исполнительные механизмы в качестве уставок, например, на драйвер электропривода или регулятор параметра. Уставки, как правило, формируются диспетчером и передаются в контроллер по промышленной сети. Размерность сигналов ТР зависит от ЦАП, применяемого в модуле

аналогового вывода контроллера. По аналогии с сигналами ТИ выделяют ТР1, ТР2, ТР4 и ТР4Float.

Следует отметить, что значения ТР4Float не могут быть напрямую переданы на исполнительные механизмы. Их нужно сначала пересчитать для ЦАП с учетом разрядности. Например, если в контроллере используется 10-разрядный ЦАП, формирующий напряжение на выходе от 0 до 12В для формирования управляющего воздействия ТР4Float равного 8.723 следует отправить на модуль аналогового вывода значение ТР4:

(8.723*2^10)div12–1=743.

Сигналы ТР имеют высокий приоритет в каналах передачи данных и передаются немедленно, с прерыванием общего цикла опроса. Как правило, данных ТР относительно немного, поскольку уставки процессов меняются нечасто, поэтому сигналы ТР передаются каждый отдельным кадром с обязательным подтверждением получения. Значение уставки сохраняется в контроллере до получения следующего значения.

Телесчет. Сокращенное название «ТИИ». Тип данных – целые неотрицательные числа. 4 байта (Double Word). Данный тип предназначен для счетчика срабатываний. В некоторых случаях он является основным. Например, при опросе приборов учета, которые передают свои показания именно в виде последовательности импульсов. Сигнал может формироваться в пользовательской задаче контроллера, в коммуникационных модулях (например, счетчик кадров) или в специальном режиме работы модуля дискретного ввода. При переполнении значение автоматически сбрасывается в 0 и счет начинается сначала.

В каналах передачи данных значения ТИИ имеют низкий приоритет и передаются также, как и сигналы ТИ.

Метка времени. Этот тип данных не самостоятельный. Метка времени закрепляется за значениями ТИ, ТС, ТУ или ТР для того, чтобы зафиксировать время последнего изменения соответствующего сигнала. Кроме того, метка времени обязательно присутствует в каждой записи сервера истории. В системах промышленной автоматизации используются два основных формата метки времени, имеющие типы Date и FileTime. Тип Date представляет собой число с плавающей точкой с двойной точностью, т.е. имеет размер 8 байт. Число обозначает количество времени в сутках, прошедшее с 00:00:00 01.01.1900 г. Тип FileTime представляет собой целое число 100-наносекундных тиков, прошедших с 00:00:00 01.01.1601 г. размером 8 байт. Преимущественно, используется тип FileTime, поскольку точность метки времени в этом случае является фиксированной (100 нс).

Передача метки времени сильно увеличивает объем данных, передаваемых по промышленной сети. Так, при передаче значения ТИ2, размером 2 байта добавление метки времени размером 8 байтов приводит к увеличению объема передаваемой информации в 5 раз. Поэтому метка времени прикрепляется только к значениям высокой важности. Для остальных значение метки времени выставляется на принимающей стороне при поступлении данных, что нужно учитывать в алгоритмах, использующих обработку времени.

По этой же причине отдельная метка времени не присваивается каждому ТС. Метка времени присваивается группе, и при получении каждое значение считается обновившимся. Как говорилось ранее, группа ТС передается по каналу, если в ней изменилось хотя бы одно значение. Т.е. изменение одного ТС влечет изменение метки времени для всех сигналов группы. Если необходимо этого избежать, то следует хранить предыдущее значение группы и присваивать новую метку времени только изменившимся сигналам.

Некоторые протоколы передачи данных (например, ModBus) в промышленных сетях вообще не предусматривают возможности передачи метки времени. В этом случае можно на передающей стороне записать метку времени для важных сигналов в значения

ТИ или СИ (статистическая информация, описание будет далее) с последующей алгоритмической распаковкой на принимающей стороне.

В некоторых телемеханических протоколах (например IEC 6870-5) предусмотрен сокращенный формат метки времени, имеющий объем 4 байта. В этом случае отсчет тиков в формате FileTime начинается с первого января текущего года. Это нужно учесть при формировании итоговой метки времени, например, для правильной записи исторических значений.

Статус сигнала. Также не является самостоятельным элементом, а закрепляется за сигналами ТИ, ТР, группами ТС и ТУ. Отвечает главным образом за достоверность значения. Как правило, имеет размер 1 байт, который интерпретируется как битовые флаги. В зависимости от версии контроллера и протокола флаги могут обозначать:

- достоверность значения;
- сигнал изменился более одного раза с момента последнего опроса;
- значение достигло верхнего допустимого предела (измерения);
- значение достигло нижнего допустимого предела;
- обрыв связи с датчиком;
- датчик неисправен и др.

Статистическая информация. Сокращенное наименование СИ. Данные сигналы несут информацию о работе самого промышленного контроллера. Например, температуру процессора, версию прошивки, количество зафиксированных сбоев в работе, диагностическую информацию по каналам связи и состоянию датчиков, исправность модулей и т.п. Может иметь размер данных 1, 2 или 4 байта – СИ1, СИ2 и СИ4. Поскольку данные этого типа не несут информации об управляемом процессе, то СИ передаются как правило по специальному запросу персонала, обслуживающего систему. В качестве такого запроса может выступать команда ТУ, адресованная самому контроллеру. Эта команда не передается на актуатор. По ней сигналы СИ становятся 1 класса.

Все указанные типы данных широко используются для управления технологическими процессами. Но некоторые протоколы и производители контроллеров предусматривают дополнительные типы данных. Например, текстовые. Для обработки таких типов следует изучить эксплуатационную документацию контроллеров.

Программируемый логический контроллер способен работать с различными типами данных, которые определяют род информации, диапазон представления и множество допустимых операций. Согласно стандарту IEC 61131 типы данных разделяются на элементарные и пользовательские.

Элементарные типы данных

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

Таблица 3.1 – целочисленные типы данных

Тип	Нижний предел	Верхний предел	Размер, байт
BYTE	0	255	1
WORD	0	65535	2
DWORD	0	2^{32}-1	4
LWORD	0	2^{64}-1	8
SINT	-128	127	1
INT	-2^{15}	2^{15}-1	2
DINT	-2^{31}	2^{31}-1	4
LINT	-2^{63}	2^{63}-1	8
USINT	0	255	1
UINT	0	65535	2
UDINT	0	2^{32}-1	4

2. **Логические переменные** объявляются ключевым словом **BOOL**. Они могут принимать только значение логического нуля («0») *FALSE* (ЛОЖЬ) или логической единицы («1») *TRUE* (ИСТИНА). При начальной инициализации логическое значение по умолчанию — ЛОЖЬ. Занимает 8 бит памяти, если не задан прямой битовый адрес.

3. **Переменные действительного типа** (**REAL** и **LREAL**) представляют действительные числа в формате с плавающей точкой. Для типа REAL необходимо 32 бита памяти и 64 – для LREAL. Диапазон значений REAL от: 1.175494351e-38F до 3.402823466e+38F. Диапазон значений LREAL от: 2.2250738585072014e-308 до 1.7976931348623158e+308.

4. **Время суток и дата -** типы переменных, выражающие время дня или дату, представляются в соответствии с ISO 8601.

Таблица 3.2 – Переменные времени

Тип	Обозначение	Нижний предел	Верхний предел
DATE	D	1 января 1970 г.	6 февраля 2106 г.
TIME_OF_DATE	TOD	00:00:00	23:59:59,999
DATE_AND_TIME	DT	00:00:00 1 января 1970 г.	06:28:15 6 февраля 2106 г.

5. **Интервал времени** – переменные типа **TIME**. В отличие от времени суток (TIME_OF_DAY) временной интервал не ограничен максимальным значением в 24 часа. Числа, выражающие временной интервал, должны начинаться с ключевого слова **TIME#** (в сокращенной форме **T#**). Максимальное значение для типа TIME: 49d17h2m47s295ms (4194967295 ms).

6. **Тип строковых переменных** (**STRING**) определяет переменные, содержащие текстовую информацию. Размер строки задается при объявлении. Если размер не указан, принимается размер по умолчанию – 80 символов. Размер задается в круглых или квадратных скобках.

Пример объявления строки размером до 35 символов: str:STRING(35) := ‘Просто строка’;

Пользовательские типы данных

Массивы

Массивы представляют собой множество однотипных элементов с произвольным доступом. Они могут быть одномерными или многомерными. Размерность массива и диапазоны индексов задаются при объявлении.

Синтаксис:

<Имя массива>:**ARRAY** [<li1>..<hi1>,<li2>..<hi2>,<li3>..<hi3> **OF** <тип элемента>;

где li1, li2, li3 указывают нижние пределы индексов; hi1, hi2 и hi3 – верхние пределы. Индексы должны быть целого типа и только положительные. Отрицательные индексы использовать нельзя.

Элементарные типы данных могут образовывать одно-, двух-, и трехмерные массивы. Путем вложения массивов можно получить многомерные массивы, но не более 9-мерных (“ARRAY[0..2] OF ARRAY[0..3] OF …”).

Пример:

Card_game: ARRAY [1..13, 1..4] OF INT;

Пример инициализации простых массивов:

arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;

arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (*сокращение для 3 по 7: 1,7,7,7 *)

arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (*сокращение для 0,0,4,4,4,4,2,3 *)

Для доступа к элементам двухмерного массива используется следующий синтаксис:

<Имя_массива>[Индекс1,Индекс2]

Пример: Card_game [9,2]

Структуры

Структуры предназначены для создания новых типов данных на основе элементов разных базовых типов. С переменной типа структура можно обращаться как с единым элементом, передавать в качестве параметра, создавать указатели, копировать и т. д.

Объявление структуры должно начинаться с ключевого слова STRUCT и заканчиваться END_STRUCT.

Синтаксис:

TYPE <имя_структуры>
STRUCT
<переменная_0> ,< переменная _1>, …< переменная _n>
END_STRUCT

END_TYPE

Пример объявления:

TYPE STRUCT1
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT

Перечисления

Перечисление позволяет определить несколько последовательных значений переменной и присвоить им наименования. Перечисление доступно в любой части проекта, даже при локальном его объявлении внутри POU. Поэтому наиболее разумно создавать все перечисления на вкладке типы данных (Data types) «Организатора Объектов» (Object Organizer). Объявление должно начинаться с ключевого слова TYPE и заканчиваться строкой END_TYPE.

Синтаксис:

TYPE <*Имя_перечисления*>:(<*Элемент_0*> ,< *Элемент_1*>, …< *Элемент_n*>); END_TYPE

Переменная типа <*Имя_перечисления*> может принимать только перечисленные значения. При инициализации переменная получает первое из списка значение. Если числовые значения элементов перечисления не указаны явно, им присваиваются последовательно возрастающие числа, начиная с 0. Фактически элемент перечисления – это число типа INT и работать с ними можно точно также. Можно напрямую присвоить число переменной типа перечисление.

Элемент, уже включенный в перечисление, нельзя повторно включать в другое перечисление.

Ограничение диапазона значений

Ограничение диапазона позволяет объявить переменную, значения которой ограничены в определенных пределах. Существует возможность создать в проекте новые типы данных с ограниченным диапазоном значений либо задать диапазон непосредственно при объявлении переменной.

Создание нового типа выглядит так:

TYPE < *Имя* > : < *Целый тип* > (<*от*>..<*до*>) END_TYPE;

< *Имя*> любой допустимый МЭК идентификатор;

<*Целый тип*> один из типов SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD);

<*от*> константа, определяющая начало диапазона значений включительно;

<*до*> константа, определяющая конец диапазона значений включительно.

Системные флаги

Системные флаги – это неявно объявленные переменные, различные для конкретных моделей PLC. Для получения списка доступных системных флагов используйте команду “Insert” “Operand”. В диалоге ассистента ввода (Input Assistant) флаги собраны в разделе System Variable.

Каждая переменная обязательно имеет наименование и тип. Сущность переменной может быть различной: представлять вход или выход ПЛК, данные в оперативной или энергонезависимой памяти.

Оценочные средства для разделов 1-3

Вопросы для самопроверки:

1. Чем отличается микроконтроллер от промышленного контроллера?
2. Какую сертификацию должны проходить приборы учета?
3. В какой модуле промышленного контроллера может отсутствовать процессор?
4. Назовите характеристики промышленного контроллера, важные для проектирования.
5. Для чего используются бескорпусные ПЛК?
6. У каких контроллеров присутствует терминал?
7. Как и зачем осуществляется гальваническая развязка измерительных каналов?
8. Какой тип сигналов контроллера имеет наивысший приоритет?
9. От чего зависит размер данных телеизмерений?
10. Какие типы сигналов несут в себе диагностическую информацию о состоянии контроллера?

Тестовые вопросы:

1. Выберите функции, характерные для ПЛК:

а) отображение информации;

б) сбор показаний с датчиков;

в) выполнение алгоритмов управления;

г) подача управляющих команд на актуаторы

2. Какой модуль ПЛК регистрирует показания дискретных датчиков:

а) аналогового ввода;

б) дискретного ввода;

в) коммуникационный;

г) дискретного вывода

3. От чего зависит размер данных телеизмерений?

а) от протокола передачи данных;

б) от точности датчика;

в) от разрядности АЦП модуля аналогового ввода;

г) размер ТИ всегда 2 байта, ни от чего не зависит

4. Что такое датчик?

а) преобразователь измеряемой величины в электрический сигнал;

б) средство измерения физической величины; в) прибор для визуальных измерений;

г) резистор, изменяющий свое сопротивление при изменении внешних условий.

5. Какой датчик называется дискретным?

а) у которого два состояния; который выдает сигнал в заданные моменты времени;

б) который выдает сигнал квантованный по значению;

в) который выдает цифровой сигнал

6. Какой модуль ПЛК может подавать команды ТУ на актуаторы?

а) аналогового ввода; аналогового вывода; дискретного ввода; б) дискретного вывода;

в) никакой из перечисленных

7.Функции ПЛК в проекте автоматизации:

а) главный пункт сбора данных;

б) предоставление пользовательского интерфейса;

в) сбор данных с первичных преобразователей;

г) передача данных по промышленной сети

8. Может ли ПЛК применяться без модулей ввода вывода?

а) нет;

б) может для выполнения алгоритмов управления;

в) может для коммуникационных целей;

г) может, но без выполнения пользовательской задачи

9. Зачем нужна метка времени?

Открытый вопрос – дайте развернутый письменный ответ.

10. Какой интерфейс наиболее востребован у интеллектуальных реле и почему? Открытый вопрос – дайте развернутый письменный ответ

Индивидуальные задания:

Задание 1.1. Выбор средств автоматизации.

Уровень сложности – легкий. Выбрать и обосновать интеллектуальное реле для автоматизации инженерных систем в квартире. Индивидуальные параметры: система освещения, система отопления, пожарная сигнализация, охранная сигнализация, управление жалюзи, электромагнитный замок.

Уровень сложности – средний. Выбрать и обосновать датчики, исполнительные механизмы, контроллер для автоматизации инженерных систем. Индивидуальные параметры: система освещения, система отопления, пожарная сигнализация, охранная сигнализация, управление жалюзи, электромагнитный замок.

Уровень сложности – сложный. Спроектировать систему управления инженерными системами: выбрать датчики, исполнительные механизмы, контроллер, составить принципиальную электрическую схему и блок-схему алгоритма управления. Индивидуальные параметры: система освещения, система отопления, пожарная сигнализация, охранная сигнализация, управление жалюзи, электромагнитный замок.

Задание 1.2. Формирование модульного состава.

Уровень сложности – легкий. Сформировать модульный состав контроллера для автоматизации малого промышленного объекта. Индивидуальные параметры: насосная станция, система приточно-вытяжной вентиляции, котельная, АЗС, лифт.

Уровень сложности – средний. Сформировать модульный состав и определить параметры работы каждого модуля контроллера для автоматизации малого промышленного объекта. Индивидуальные параметры: насосная станция, система приточно-вытяжной вентиляции, котельная, АЗС, лифт.

Уровень сложности – сложный. Сформировать конфигурацию модулей и сигналов контроллера для автоматизации малого промышленного объекта. Индивидуальные параметры: насосная станция, система приточно-вытяжной вентиляции, котельная, АЗС, лифт.

Раздел 4. Конфигурирование промышленных контроллеров

Понятие конфигурации промышленного контроллера. Промышленный контроллер является универсальным инструментом для автоматизации различных технологических процессов. Но универсальность всегда приводит к избыточности. Широкие возможности, не востребованные в каждом конкретном случае, приводят к необоснованному удорожанию, излишнему энергопотреблению, увеличенным габаритам средств автоматизации. Решению этой проблемы во многом способствует модульная структура контроллера. Задачей проектировщика является выбор модели контроллера, и его модульного состава, подходящего к условиям конкретного проекта. Также модульная структура дает возможность заложить некоторый запас вычислительных мощностей и интерфейсов для подключения дополнительных точек ввода-вывода и каналов коммуникации. Модульный состав контроллера в конкретном проекте, текущие настройки каждого модуля, подключенные каналы коммуникации, источники данных и исполнительные механизмы являются конфигурацией промышленного контроллера.

Конфигурация контроллера является частью конфигурации проекта автоматизации в целом и должна быть согласована с конфигурацией всех остальных средств автоматизации: промышленных сетей, серверов ввода-вывода, других контроллеров, средств отображения информации. Достигнуть этого можно разными способами. Например, использовать интегрированную среду конфигурирования проекта, из которой могут быть сконфигурированы все элементы, или созданием специального документа – информационного обеспечения проекта. Интегрированная среда конфигурирования, как правило, входит в состав SCADA-пакетов, развертываемых на верхнем уровне автоматизации, т.е. в диспетчерском пункте. Поскольку контроллер может работать в полностью автоматическом режиме, и быть главным управляющим устройством в проекте, то к каждой модели контроллера предусмотрены автономные средства конфигурирования. Т.е. конфигурацию контроллера можно рассматривать, создавать и применять отдельно от всех остальных средств автоматизации, как мы и будем делать в рамках данной дисциплины.

Конфигурация контроллера состоит из модульного состава, размещения модулей, настроек резервирования и конфигураций отдельных модулей. Размещение модулей может быть локальным или распределенным.

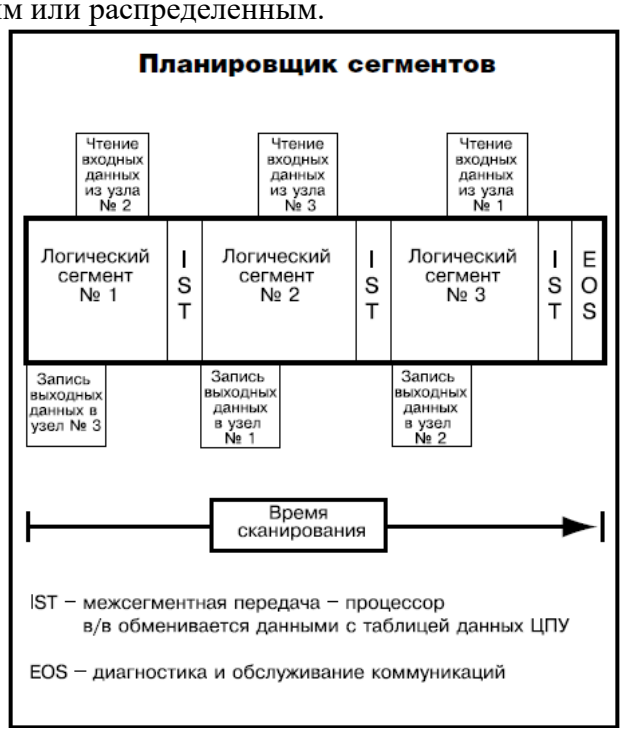


Рисунок 4.1 – Конфигурирование распределенных модулей

При локальном размещении важна только позиция каждого модуля. Обычно позиция задается номером подключения к системной магистрали. При распределенной архитектуре выносные модули распределяются по логическим сегментам. И номер модуля задается в логическом сегменте.

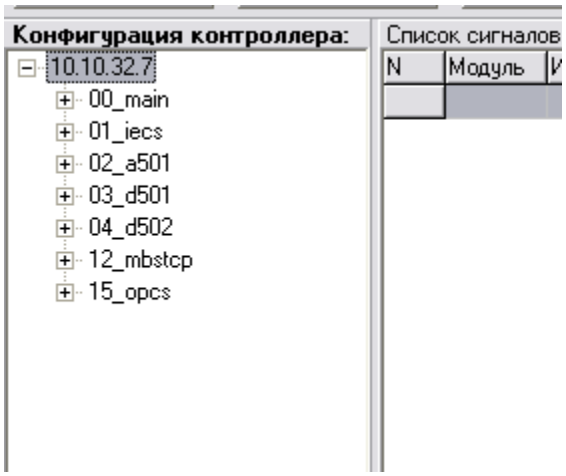


Рисунок 4.2 – Модульный состав локального промышленного контроллера

Многие современные контроллеры поддерживают технологию Plug and Play, которая позволяет при подключении к магистрали нового модуля автоматически его распознать, включить в текущую конфигурацию и немедленно запустить в работу, т.е. начать опрос модулем ЦП. На первый взгляд это удобно, но на самом деле не имеет важного практического значения. Ведь при подключении нового устройства алгоритмы обработки его сигналов в пользовательской задаче еще не реализованы. Да и модульный состав контроллера в реальном проекте прописан в проектной документации, и в процессе эксплуатации объекта изменяется нечасто. Другое дело возможность «горячей» замены неисправного модуля на новый аналогичный без остановки контроллера. В этом случае должны сохраняться и применяться все настройки, сделанные в конфигурации контроллера ранее.

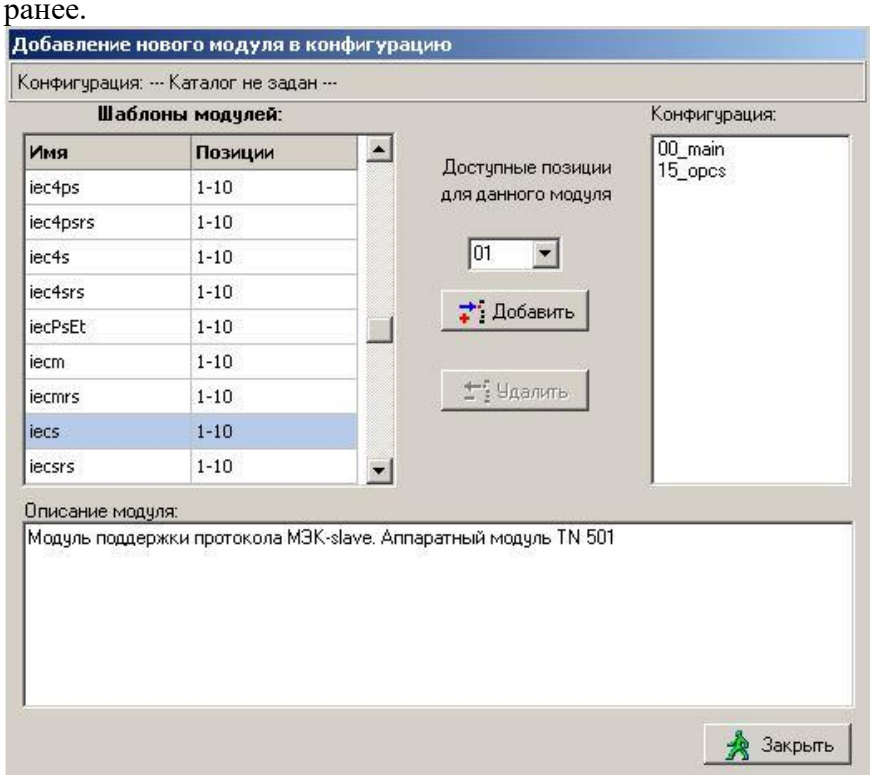


Рисунок 4.3 – Формирование модульного состава контроллера

В распределенной конфигурации контроллера для связи между модулем ЦП и выносными модулями используется не системная магистраль, а канал передачи данных, как правило с высокой пропускной способностью. В этом случае опрос выносных модулей осуществляется через соответствующий коммуникационный модуль, непосредственно связанный с контроллером.

Конфигурация отдельного модуля представлена на рисунке.

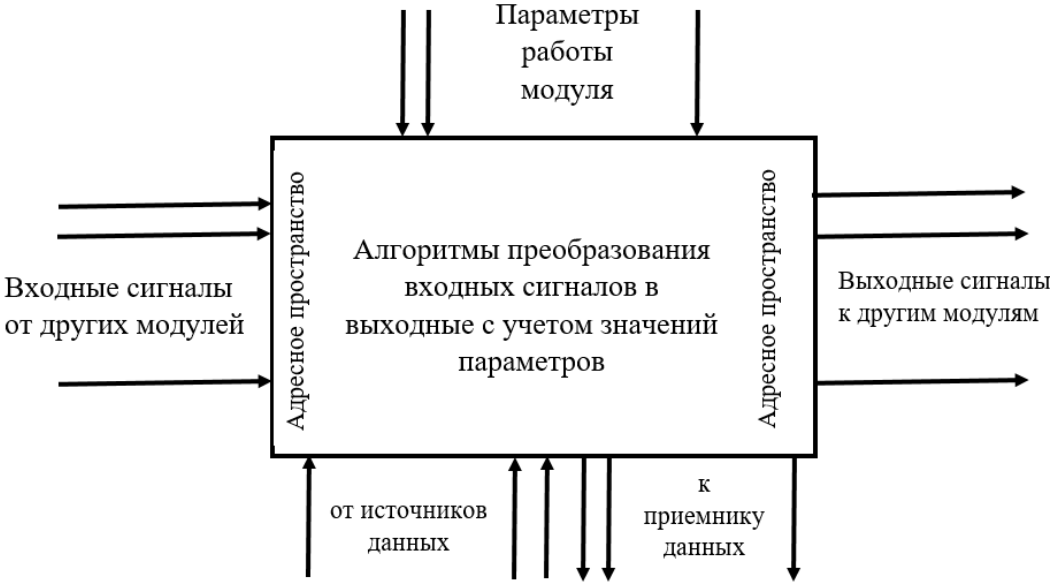


Рисунок 4.4 – представление конфигурации модуля

В целом каждый модуль контроллера, кроме модуля ЦП, работает по алгоритму, заданному встроенным программным обеспечением, так называемой «прошивкой». Но поскольку в проекте часто требуется настроить нюансы работы, то у модуля есть ряд параметров, значения которых могут влиять на алгоритмы обработки данных. Так, для коммуникационных модулей может быть задан адрес станции в сети, скорость передачи данных, объем передаваемого кадра и т.п. Пример конфигурирования параметров модуля контроллера приведен на рисунке. Как правило, при добавлении модуля в конфигурацию контроллера к нему сразу применяются значения параметров по умолчанию. В некоторых случаях можно воспользоваться этими значениями. Но если, например, нужно задать IP-адрес, то значением по умолчанию никак не обойтись, адрес нужно указать явно. Особенно осторожно следует поступать с модулями Plug and Play. Ведь применение параметров по умолчанию для немедленной работы модуля может привести к серьезным коллизиям в промышленных сетях.

Имя	Тип	Значение	Мин.	Макс.	Рек.	Комментарий
Driver	String	FIFO_New				Имя драйвера
CopyProc	Bool	0	0	1	0	Признак разрешенных копий процесса
ConfTA	U2	2000	500	10000	2000	Тайм-аут ожидания подтверждения данных от ЦП (мс)
Reserv1	I2	0	-32768	32767	0	Резерв 1
Reserv2	I2	0	-32768	32767	0	Резерв 2
Reserv3	I1	0	-128	127	0	Резерв 3
Reserv4	I1	0	-128	127	0	Резерв 4
IntegrTime	I2	100	20	5100	1000	Время интегрирования (мс). Должно задаваться кратно 20 мс
CalibrPer	U1	10	1	100	10	Периодичность калибровки
EnblCycle...	U1	0	0	1	0	Признак циклической передачи данных (0-передавать цик...
DsblDiag	U1	1	0	1	1	Признак выдачи диагностики (0-выдается, 1-не выдается)

Рисунок 4.5 – Пример конфигурирования параметров модуля аналогового ввода

Каждый модуль осуществляет обмен информацией:

1. С модулем центрального процессора, и в частности, с исполняемым модулем пользовательской задачи, разработка программы для которой является предметом этой дисциплины.
2. С другими модулями. Например, модуль аналогового ввода передает измеренные значения ТИ в коммуникационный модуль для отправки в диспетчерский пункт. Или специализированная команда ТУ, не требующая алгоритмического сопровождения в пользовательской задаче передается непосредственно на модуль дискретного вывода.
3. С внешними источниками и приемниками данных. Например, источником и приемником данных может быть промышленная сеть, обмен по которой осуществляется с использованием этого модуля. Или источником данных может быть датчик. Приемником может быть исполнительный механизм.

Настройка обмена данными с другими модулями контроллера осуществляется путем маршрутизации выходных сигналов модуля. Т.е. задается модуль и элемент адресного пространства в который следует отправлять изменения выходного сигнала. Пример маршрутизации приведен на рисунке. Дополнительная маршрутизация входных сигналов от других модулей при этом не требуется, поскольку входной сигнал одного модуля является выходным сигналом другого, и уже сконфигурирован.

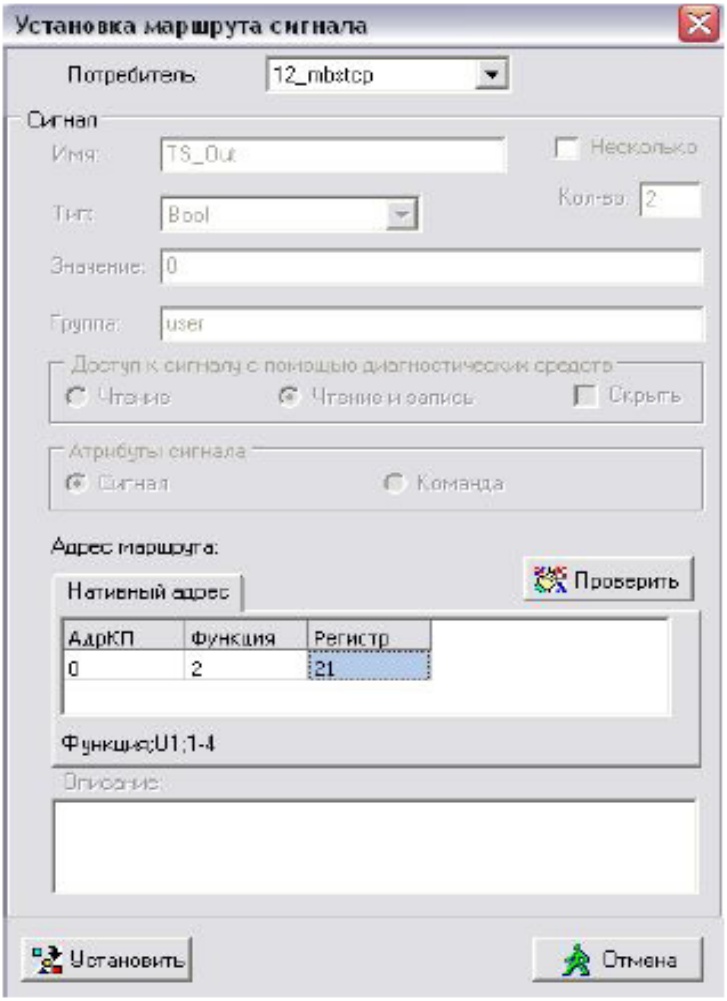


Рисунок 4.6 – Пример маршрутизации выходного сигнала

В ряде случаев производитель промышленных контроллеров предлагает несколько модулей со схожим функционалом. Разница может быть в количестве внешних

интерфейсов (например, модуль дискретного вывода на 16 и на 32 сигнала), в применяемом процессоре, вариантах внешнего исполнения, и, конечно же в стоимости. Если возникает необходимость замены одного модуля контроллера на другой, то нужно не только скоировать параметры замещаемого модуля но и изменить маршрутизацию сигналов в других модулях. Для этого полезно знать информацию обо всех маршрутизированных сигналах. Такую информацию можно получить в виде таблицы в настройках исходного модуля.

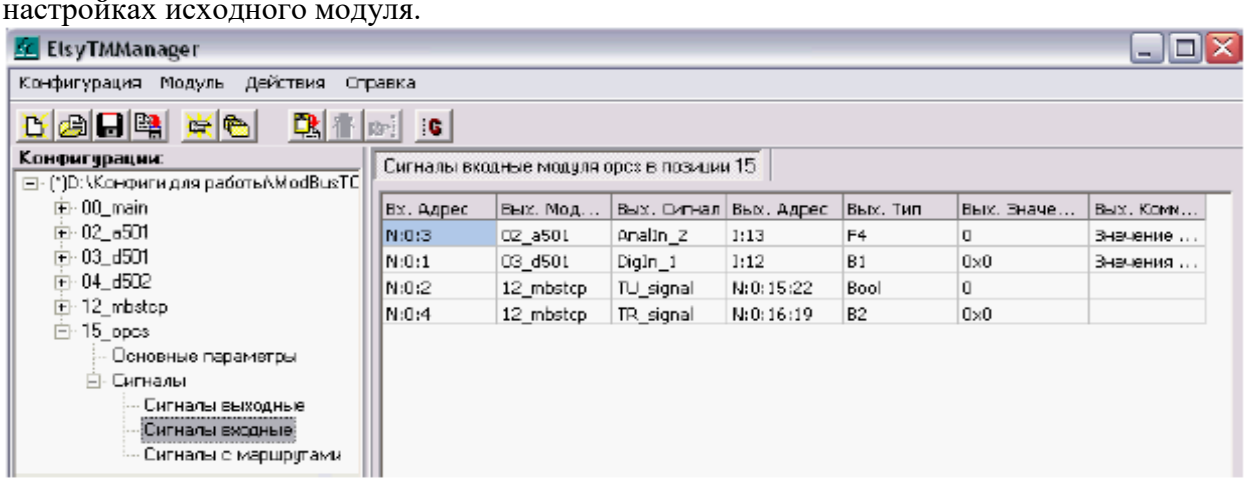


Рисунок 4.7 – Пример таблицы маршрутизированных сигналов

Следует обратить внимание, что некоторые модули кардинально могут изменять логику своей работы в зависимости от встроенного программного обеспечения (прошивки). Речь идет фактически о разных модулях, реализованных на одной и той же аппаратной платформе. Отличить такие модули при внешнем осмотре или даже при вскрытии корпуса невозможно. Так, например, коммуникационный модуль с разъемом RJ45 (Ethernet) может использоваться как опросчик или подчиненная станция в протоколах ModBusTCP, Ethernet/IP, FBUS, PowerLink. В этом случае конкретная прошивка модуля также является частью конфигурации контроллера, поскольку аппаратный модуль нельзя однозначно идентифицировать и включить в работу без этой информации.

В ряде случаев некоторые модули, особенно модуль центрального процессора или коммуникационные, могут одновременно выполнять логические функции нескольких модулей. Например, модуль может выступать в качестве опросчика по одному протоколу и в качестве подчиненной станции по другому протоколу. Для единообразия в конфигурации контроллера каждая такая логическая функция реализуется как модуль, который, конечно, является программным, в некоторой степени, виртуальным. Т.е. мы можем не обнаружить этого модуля, добавленного в системную магистраль или на крейт контроллера, но при этом все функции данного модуля будут выполняться вполне реально! Т.е. один аппаратный модуль может быть представлен несколькими модулями в конфигурации контроллера. В частности, в качестве отдельного модуля, реализуемого на аппаратном модуле ЦП, выступает исполняемый модуль пользовательской задачи. У него также есть параметры, входные и выходные сигналы для обмена с другими модулями. Но он не подключен к внешним источникам данных. Их роль играют внутренние переменные и функциональные блоки программы.

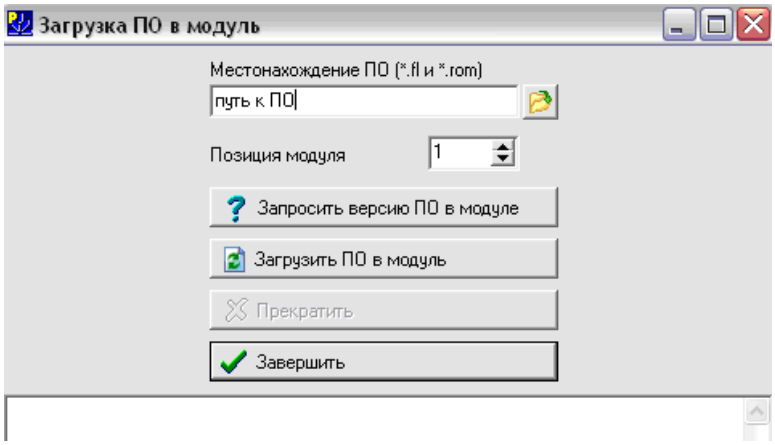


Рисунок 4.8 – Окно загрузки ПО в модуль

Поскольку контроллер является ключевым звеном в проекте автоматизации, важную роль играют мероприятия по повышению его надежности. Одним из самых эффективных является резервирование контроллера или его отдельных модулей. В первую очередь речь идет о модулях питания и ЦП. Для резервирования устанавливаются два или более одинаковых модуля с одинаковой конфигурацией и подключенные к одним и тем же внешним источникам и приемникам данных. Если с модулями питания как правило все просто, их устанавливают несколько больше, чем нужно, и даже если один из них выйдет из строя, то остальные обеспечат необходимую мощность до замены неисправного, то с другими модулями требуется алгоритм резервирования. Дело в том, что работать в полнофункционально оба модуля не могут. Т.е. пока основной исправно работает, резервный должен молчать, но быть наготове. Закладывать функцию резервирования в прошивку каждого модуля не всегда разумно. Поэтому алгоритм реализован в отдельном модуле резервирования. Этот модуль может быть как аппаратным, так и виртуальным, реализованным в модуле ЦП или в одном из коммуникационных. Модуль мониторит работоспособность основного модуля и в случае обнаружения неисправности запускает в работу резервный. Параметры работы модуля настраиваются в конфигураторе аналогично другим модулям.

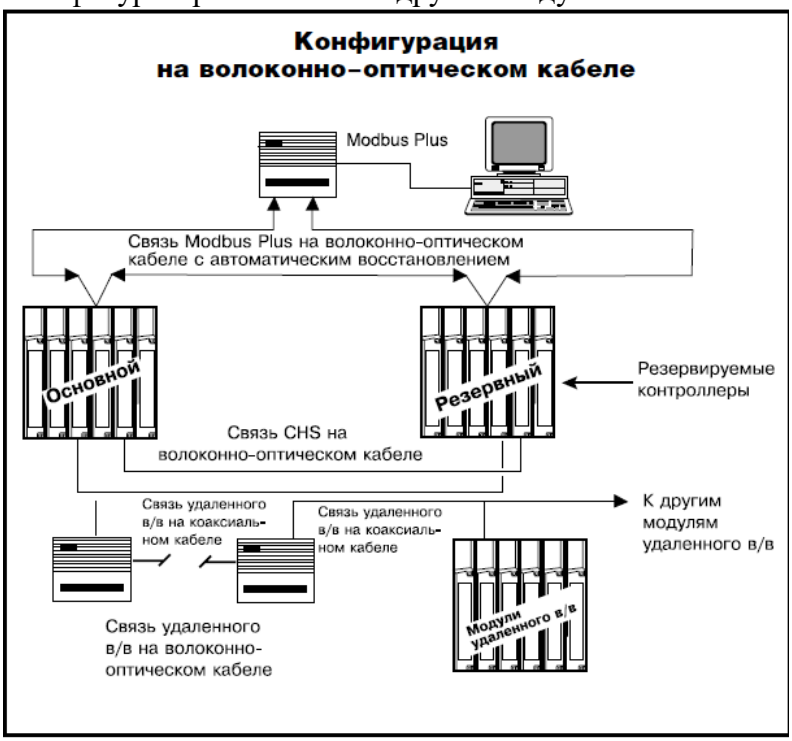


Рисунок 4.9 – Пример резервирования промышленного контроллера

Как правило, редактирование конфигурации осуществляется в офлайне, не на промышленном контроллере, а на персональном компьютере разработчика. После настройки конфигурации требуется ее проверка (верификация). Удобно, если производитель контроллера предоставляет инструменты для верификации конфигурации.

Загрузка конфигурации в промышленный контроллер осуществляется как правило по сети, уж конечно, без демонтажа контроллера. Перед загрузкой новой конфигурации настоятельно рекомендуется сохранить старую, чтобы обеспечить возможность возврата. Непосредственно после загрузки необходимо воспользоваться инструментами диагностики состояния контроллера, о которых поговорим позже, для мониторинга работы с новой конфигурацией.

Отдельно нужно рассмотреть конфигурацию пользовательской задачи контроллера, формированию которой посвящена данная дисциплина. Стандарт IEC 61131-3 предлагает следующую модель конфигурации программного обеспечения промышленного контроллера:

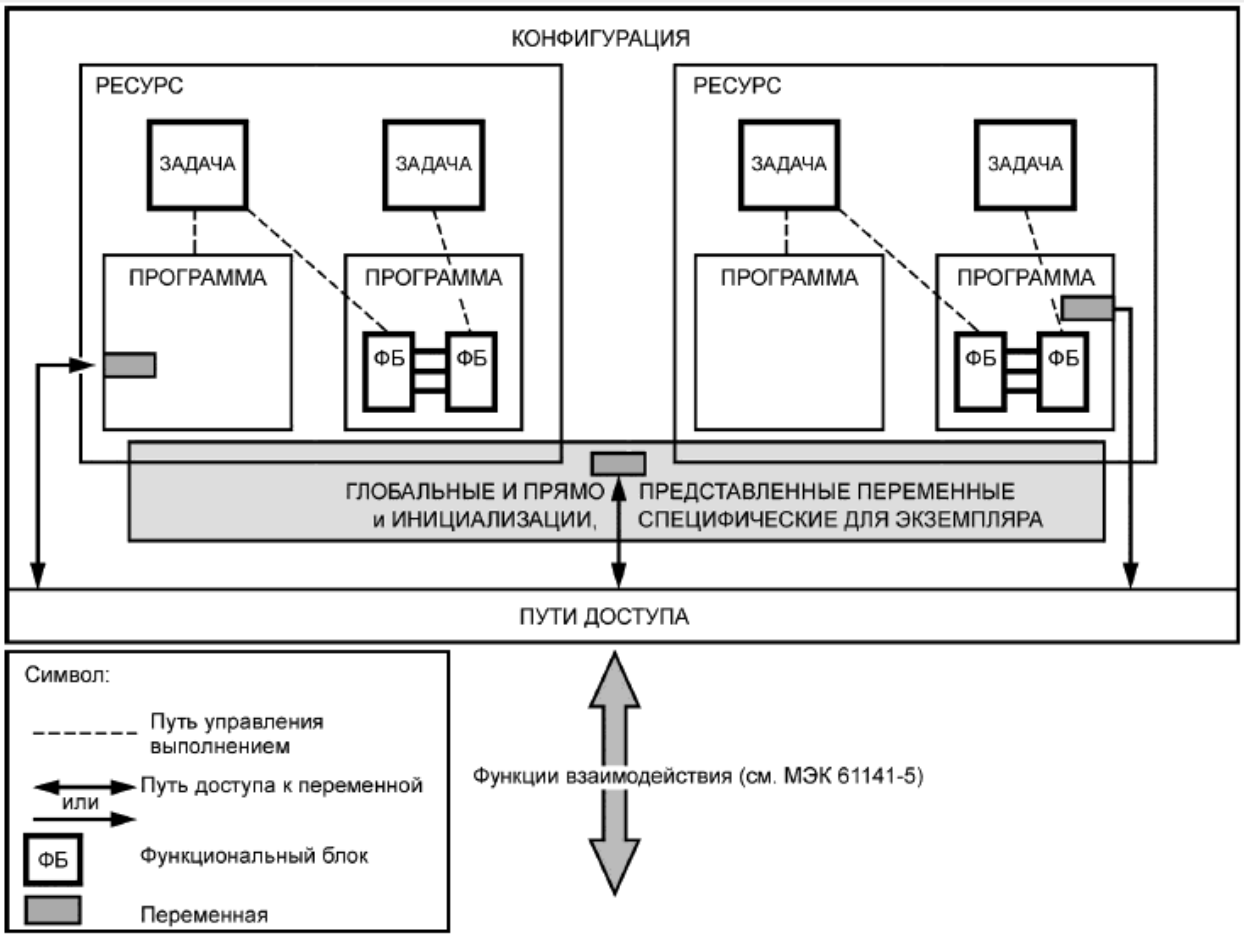


Рисунок 4.10 – Конфигурация программного обеспечения контроллера

Конфигурация является элементом языка, который соответствует системе программируемого контроллера, как определено в МЭК 61131-1. Ресурс соответствует "функции обработки сигналов" и ее "человеко-машинному интерфейсу" и "функциям интерфейса с датчиками и исполнительными механизмами" (при наличии таковых), как определено в МЭК 61131-1.

Конфигурация содержит один или более ресурсов, каждый из которых содержит одну или более программ, выполняемых под контролем нуля или более задач. Программа может содержать нуль или более экземпляров функциональных блоков или других элементов языка, как определено в настоящем стандарте. Задача способна вызывать

(например, на периодической основе) выполнение набора программ и экземпляров функциональных блоков.

Конфигурации и ресурсы могут запускаться и останавливаться через функции "интерфейс оператора", "программирование, тестирование и мониторинг" или "операционная система", определенные в МЭК 61131-1. Запуск конфигурации будет вызывать инициализацию ее глобальных переменных с последующим запуском всех ресурсов конфигурации. Запуск ресурса будет вызывать инициализацию всех переменных в ресурсе с последующей активацией всех задач в ресурсе. Останов ресурса будет вызывать прекращение всех его задач, в то время как останов конфигурации будет вызывать останов всех ее ресурсов.

Механизмы управления задачами определены в 6.8.2, а механизмы запуска и останова конфигураций и ресурсов через функции взаимодействия определены в МЭК 61131-5. Программы, ресурсы, глобальные переменные, пути доступа (и соответствующие привилегии доступа) и конфигурации могут быть загружены или удалены "функцией взаимодействия", определенной в МЭК 61131-1. Загрузка или удаление конфигурации или ресурса будет эквивалентно загрузке или удалению всех элементов, которые там содержатся.

Если конфигурация создавалась для отдельного контроллера, то необходимо также позаботиться о ее включении в общую конфигурацию проекта автоматизации. Хорошо сконфигурированные и правильно запрограммированные промышленные контроллеры способны работать без остановок и сбоев годами.

Раздел 5. Языки программирования промышленных контроллеров

Если для программирования микроконтроллеров используются традиционные языки программирования, такие как Assembler, C++, Java, Python, то для промышленных контроллеров используются специальные языки. Так сложилось исторически, что промышленные контроллеры применялись не IT-специалистами, а электрониками, осуществлявшими автоматизацию производства, в том числе в докомпьютерную эпоху. Поэтому языки программирования промышленных контроллеров в большей степени ориентированы на их знания и навыки. Языки программирования промышленных контроллеров устанавливаются стандартом IEC 61131-3. На сегодняшний день в стандарт входит 5 языков: IL, ST, LD, FBD, SFC. Далее рассмотрим последовательно характеристики каждого из языков.

Язык IL (сокращение от Instruction List) - текстовый язык низкого уровня. Выглядит как типичный язык Ассемблера, что объясняется его происхождением: для некоторых моделей ПЛК фирмы Siemens является языком Ассемблера. В рамках стандарта IEC 6 1131-3 к архитектуре конкретного процессора не привязан. Происхождение - STEP 5 (Siemens). Пример программы на языке IL приведен на рисунке.

Рисунок 5.1 –Пример программы на языке IL

Программа, написанная на IL работает с одной ячейкой памяти, которая называется Result (результат). Каждая инструкция (мнемокод) языка записывается в отдельную строку. В строке содержится четыре поля, порядок которых определен стандартом и не должен меняться: метка, оператор, операнд и комментарий. Некоторые поля могут быть не заполнены. Обязательным является только оператор. Операнды являются обязательными, если они необходимы оператору. Метки используются для переходов.

В языке определено всего порядка 20 базовых операторов:

LD – поместить значение операнда в Result;
ST – присвоить операнду значение Result;
S – присвоить операнду значение *True* если Result имеет значение *True*;
R – присвоить операнду значение *False* если Result имеет значение *True*;
AND – логическое И;
OR – логическое ИЛИ;
XOR – исключающее ИЛИ;
NOT – побитовая инверсия Result;
ADD – сложение;
SUB – вычитание;
MUL – умножение;
DIV – деление;
MOD – остаток от деления;
GT – операция сравнения «больше»;
GE – операция сравнения «больше или равно»;
QE – логическое равенство;
NE – не равно;

LT – операция сравнения «меньше»;
LE – операция сравнения «меньше или равно»;
JMP – переход на метку;
CAL – вызов функционального блока;
RET – возврат в вызывающую программу.

Также базовый оператор может быть дополнен модификатором, изменяющим выполнение команды. Модификатор *N* (negation) производит инверсию операнда. Модификатор *C* (condition) предполагает выполнение команды только в том случае, если Result имеет значение *True*. Особое значение имеет модифицированный оператор JMPC, осуществляющий переход по условию. Модификаторы *N* и *C* могут использоваться совместно, и в этом случае записываются слитно - *CN*. Также важным является оператор *CAL*, позволяющий использовать функции, программы и функциональные блоки, написанные в других модулях, в том числе и на других языках программирования. Стандарт также допускает многоуровневые вложения операндов, которые оформляются в круглых скобках. При этом каждый вложенный оператор записывается с новой строки. Каждая закрывающая скобка также идет отдельной строкой.

Еще до принятия стандартов IEC серии 1131 IL получил распространение в области автоматизации, как самый простой в реализации язык, поскольку его поддерживает любой процессор. Преимущество данного языка заключается в максимальной близости языковых конструкций к машинным кодам, что обеспечивает интерпретатору возможность формирования исполняемого модуля с минимальным количеством инструкций. Набор команд ассемблера отличается, в зависимости от архитектуры процессора, а соблюдение требований IEC 61131-3 обеспечивает аппаратную переносимость разработанных алгоритмов. Однако, сложности написания и модификации программ на этом языке по сравнению с другими языками, приводят к тому, что на нем реализуются только простые алгоритмы, не требующие сложной логики для принятия решений.

Язык ST (сокращение от Structured Text) - текстовый высокоуровневый язык общего назначения, по синтаксису ориентированный на Паскаль. Происхождение: Grafcet (Telemechanique-Groupe Schneider). Пример программы на языке ST приведен на рисунке

```
PROGRAM position
VAR
   initial:        AT %I0.0   :   BOOL;  (*исходное состояние*)
   key_start:      AT %I0.1   :   BOOL;  (*кнопочный старт*)
   key_halt:       AT %I0.2   :   BOOL;
   position_A:     AT %I0.3   :   BOOL;  (*Позиция А достигнута*)
   motor:          AT %Q0.0   :   BOOL;
END_VAR

if (initial AND key_start) then
   motor := TRUE;
end_if;
if (key_halt OR position_A) then
   motor := FALSE;
end_if;
END_PROGRAM
```

Рисунок 5.2 –Пример программы на языке ST

Текстовые программы выполняются по правилам чтения: сверху вниз и слева направо. На практике язык ST удобен для создания сложных логических процедур, трудно реализуемых на языке IL, и слишком громоздких на графических языках. Кроме того, элементы языка активно используются в составе сложных проектов совместно с другими языками (например , SFC, FBD или LD).

Основу языка представляют операторы и команды, запись каждой из которых оканчивается «;». Значительная часть инструкций выполняют арифметические, логические или битовые операции. Также имеются функции для преобразования одного типа данных в другой.

Основные функции языка ST:

OR, XOR, AND – булевы функции;

>, >=, <, <=, =, <> - операции сравнения;

–, +, *, / - арифметические операции;

NOT – дополнение;

EXPT – возведение в степень;

:= - присваивание.

Для определения порядка выполнения операций используются приоритеты и круглые скобки. Самым низким приоритетом обладают булевы функции и операции сравнения. Самый высокие – фрагменты кода, заключенные в круглые скобки.

Операторы языка ST используются для изменения порядка выполнения инструкций:

IF - оператор выбора (ветвления). Выполняет одно из действий в зависимости от результата проверки одного или нескольких логических условий. Полные синтаксис оператора:

```
IF <Boolean_expression 1> THEN
        <instructions 1> ;
ELSIF <Boolean_expression 2> THEN
        <instructions 2> ;

ELSIF <Boolean_expression n> THEN
        <instructions n> ;
ELSE
        <instructions> ;
END_IF;
```

Часто используются сокращенные конструкции типа «IF – THEN – END_IF» или «IF – THEN - ELSE- END_IF»

FOR - оператор организации цикла с заранее известным количеством повторов. Синтаксис:

```
FOR  <INT_Var> := <INIT_VALUE> TO <END_VALUE>
                  {BY <Step size>} DO
        ...
      <instructions> ;
        ...
END_FOR;
```

Цикл работает с переменной – счетчиком INT_Var, которой на первом шаге присваивается начальное значение INT_VALUE. Далее следует проверка – сравнение INT_Var > END_VALUE?, которая в случае соответствия логическому «нет» допускает выполнение следующих за ключевым словом DO инструкций (это так называемое тело цикла). После этого переменная – счетчик INT_Var увеличивается на величину Step size, которая выступает необязательным параметром конструкции и по умолчанию равна единице. Далее следует проверка условия INT_Var > END_VALUE?, и цикл повторяется. Если очередная проверка INT_Var > END_VALUE? даст логическое «да», то это приведет к завершению цикла.

WHILE - оператор организации неопределенного количества циклов, необходимость повтора каждого цикла проверяется непосредственно перед его выполнением. Синтаксис оператора:

```
WHILE <Boolean expression> DO
    ...
    <Instructions>;
    ...
END_WHILE;
```

Набор инструкций между ключевым словом DO и операторной скобкой END_WHILE выполняется до тех пор, пока Boolean_expression имеет значение *True*. Как только логическое выражение примет значение *False*, происходит выход из цикла.

REPEAT - оператор организации неопределенного количества циклов, необходимость повтора каждого цикла проверяется непосредственно после его выполнения. Синтаксис:

```
REPEAT
   ...
   <Instructions>;
   ...
  UNTIL <Boolean expression>
END_REPEAT
```

Логика выполнения аналогична цикла WHILE с той лишь разницей, что значение логического выражения проверяется после выполнения операторной скобки, что приводит к тому, что цикл выполнится не менее одного раза.

CASE - оператор выбора определенного действия в зависимости от конкретного значения одной переменной. Синтаксис:

```
CASE  <Var1> OF
    <Value1>: <Instruction 1>;
    <Value2>: <Instruction 2>;
    <Value3, Value4, Value5>: <Instruction 3>;
    <Value6  ..    Value10>: <Instruction 4>;
     ELSE <Alternative_instruction>;
END_CASE;
```

Значение переменной Var1 определяет выбор действия из заранее определенного списка. Для этого после ключевого слова OF следуют предполагаемые значения Value переменной Var1 и предусмотренные для каждого значения действия – Instruction i. Значения Value могут указываться как одиночные, так и в виде списка чисел, разделенных запятыми, или даже целого диапазона значений, границы которого отделены многоточием. Если переменная Var1 не примет ни одно из предусмотренных значений, то в этом случае имеется альтернативное действие (или целый список), обособленное ключевым словом ELSE. Завершает данную управляющую конструкцию ключевое слово END_CASE.

EXIT - управляющая инструкция для немедленного завершения циклов FOR, WHILE, REPEAT.

RETURN - Оператор для безусловного завершения работы текущего модуля (выход из работающей подпрограммы на языке ST).

Язык LD (сокращение от Ladder Diagram) - графический язык программирования, являющийся стандартизованным вариантом класса языков релейно-контактных схем. Логические выражения на этом языке описываются в виде реле, которые широко применялись в области автоматизации в 60-х годах. Ввиду своих ограниченных возможностей язык дополнен привнесенными средствами: таймерами, счетчиками и т.п. Происхождение: различные варианты языка релейно-контактных схем (Allen-Bradley,

AEG Schneider Automation, GE-Fanuc, Siemens). Пример программы на языке LD приведен на рисунке.

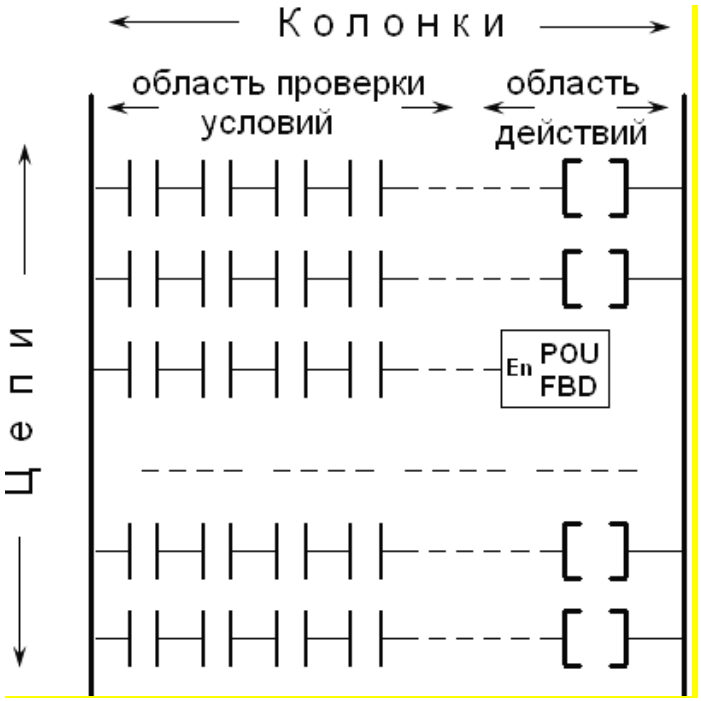


Рисунок 5.3 – Организация программы на языке LD

Программа представляет собой прототип релейно-контактных цепей заключенных между двумя вертикальными шинами питания «+» и «–», которые в некоторых средах программирования не показываются. Основными элементами схем являются:

Контакты - ┤ ├ прямой и ┤ / ├ инверсный.

Реле –( )– прямое и –(/)– инверсное.

Реле с фиксацией –(S)–.

Реле отключения (сброса) с фиксацией –(R)–.

Переход к цепи, указанной меткой –►.

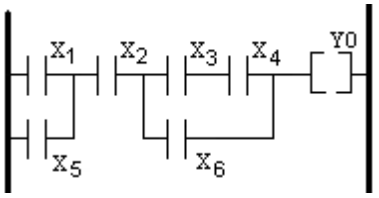


Рисунок 5.4 – Пример логического выражения на языке LD

Язык FBD (сокращение от Functional Block Diagram) - графический язык по своей сути похожий на LD. Вместо реле в этом языке используются функциональные блоки, по внешнему виду - микросхемы. Алгоритм работы некоторого устройства на этом языке выглядит как функциональна схема электронного устройства: элементы типа "логическое И", "логическое ИЛИ" и т.п., соединенные линиями. Корни языка выяснить сложно, однако большинство специалистов сходятся во мнении, что это не что иное, как перенос идей языка релейно-контактных схем на другую элементную базу. Пример программы на языке FBD приведен на рисунке

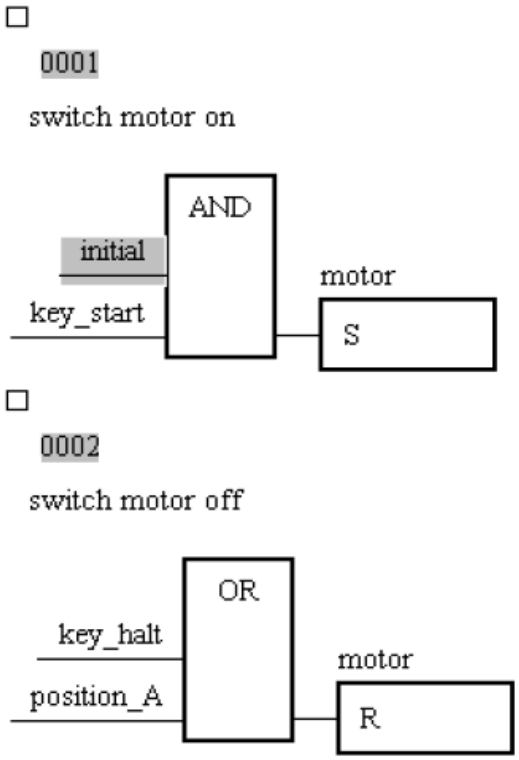


Рисунок 5.5 –Пример программы на языке FBD

Программа, написанная на графическом языке FBD, представляет собой набор связанных друг с другом функциональных блоков, выходы и входы которых соединены линиями связи. Линии связи отражают определенные программные переменные, через которые происходит обмен данными от блока — к блоку. 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 промышленного контроллера.

В процессе программирования на языке FBD применяются как стандартные блоки из библиотек, так и блоки, сами 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 контроллера, либо записывается в какую-то внутреннюю переменную.

Функции и функциональные блоки языка FBD в основном графически дублируют языковые конструкции языка ST. Логика работы программы на графических схемах более наглядна. Но графические программы занимают больше места на экране. Удобно применять язык FBD для реализации главного исполняемого модуля программы контроллера, объединяющего в себе функциональные блоки, разработанные на других языках.

Язык SFC (сокращение от Sequential Function Chart) - графический язык, используемый для описания алгоритма в виде набора связанных пар: шаг (step) и переход (transition). Шаг представляет собой набор операций над переменными. Переход - набор условных логических выражений, определяющий передачу управления к следующей паре шаг-переход. По внешнему виду описание на языке SFC напоминает хорошо известные

логические блок-схемы алгоритмов. SFC имеет возможность распараллеливания алгоритма. Однако SFC не имеет средств для описания шагов и переходов, которые могут быть выражены только средствами других языков стандарта. Происхождение: Grafcet (Telemechanique-Groupe Schneider). Пример программы на языке SFC приведен на рисунке

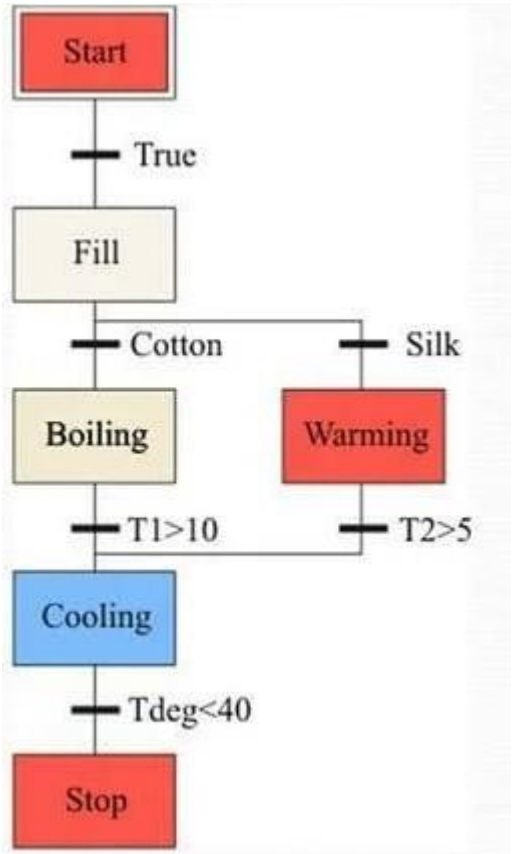


Рисунок 5.6 –Пример программы на языке SFC

Язык SFC использует следующие структурные элементы для создания программы: шаг (в том числе, начальный шаг), переход, блок действий, прыжок и связи типа дивергенция и конвергенция. Реализация программ на языке SFC больше всего похожа на сети Петри.

Шаг описывает одну операцию. У каждого шага три контакта: сверху и снизу для соединения с переходами, и сбоку для соединения с блоком действия. Два шага не могут быть соединены непосредственно между собой, только через элемент перехода.

Переход определяет условие перехода от одного шага к другому. т.е. активации следующего шага в процессе выполнения программы. Условием перехода может быть логическая переменная или константа, логический адрес или логическое выражение, описанное на любом языке. Условие может включать серию инструкций, образующих логический результат, в виде ST выражения или на любом другом языке.

Блок действий определяет операции, которые должны выполняться при активации (выполнении) шага. Шаги без связанного блока действий идентифицируются как ждущий шаг. Блок действий может состоять из предопределённых действий. Каждому предопределённому действию присваивается имя. Одно действие может использоваться сразу в нескольких шагах. Действие может выполняться непрерывно, пока активен шаг, либо единожды. Это определяется специальными квалификаторами. Квалификаторы также могут ограничивать время выполнения каждого действия в шаге.

Прыжок означает переход на заданный шаг. В программе прыжок устанавливается вместо очередного шага.

Дивергенция – это множественное соединение в направлении от одного шага к нескольким переходам. Активируется только одна из ветвей. Условия, связанные с различными переходами в начале дивергенции, не являются взаимоисключающими по умолчанию. Взаимоисключение должно быть явно задано в условиях переходов, чтобы гарантировать, что во время выполнения программы активируется одна конкретная ветвь.

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.

Параллельная дивергенция – это множественное соединение, направленное от одного перехода к нескольким шагам. Она соответствует параллельному выполнению операций процесса.

Параллельная конвергенция – это соединение нескольких шагов к одному и тому же переходу. Обычно она используется для группирования ветвей, взявших начало дивергенции.

Перечисленные языки используются ведущими фирмами изготовителями промышленных контроллеров, имеют длительную историю применения, достаточно распространены и известны пользователям по тем или иным модификациям. Несмотря на то, что во многих случаях такие модификации несущественны, это влечет определенные неудобства при работе с контроллерами различных фирм-изготовителей. С этой точки зрения, стандарт IEC 61131-3, несомненно, прогрессивен, поскольку позволяет привести бесчисленное число различных вариантов и интерпретаций языков программирования промышленных контроллеров к единому знаменателю.

Среды программирования состоят из двух частей: набора средств разработки для установки на персональный компьютер и исполняемого на целевом контроллере ядра-интерпретатора. Набор средств разработки состоит из редактора, отладчика и препроцессора, который подготавливает описанный проектировщиком алгоритм к формату, "понятному" ядру-интерпретатору. Этот набор имеет современный пользовательский интерфейс, позволяет тестировать алгоритм в режиме эмуляции контроллера и получать листинг алгоритма на языках его описания.

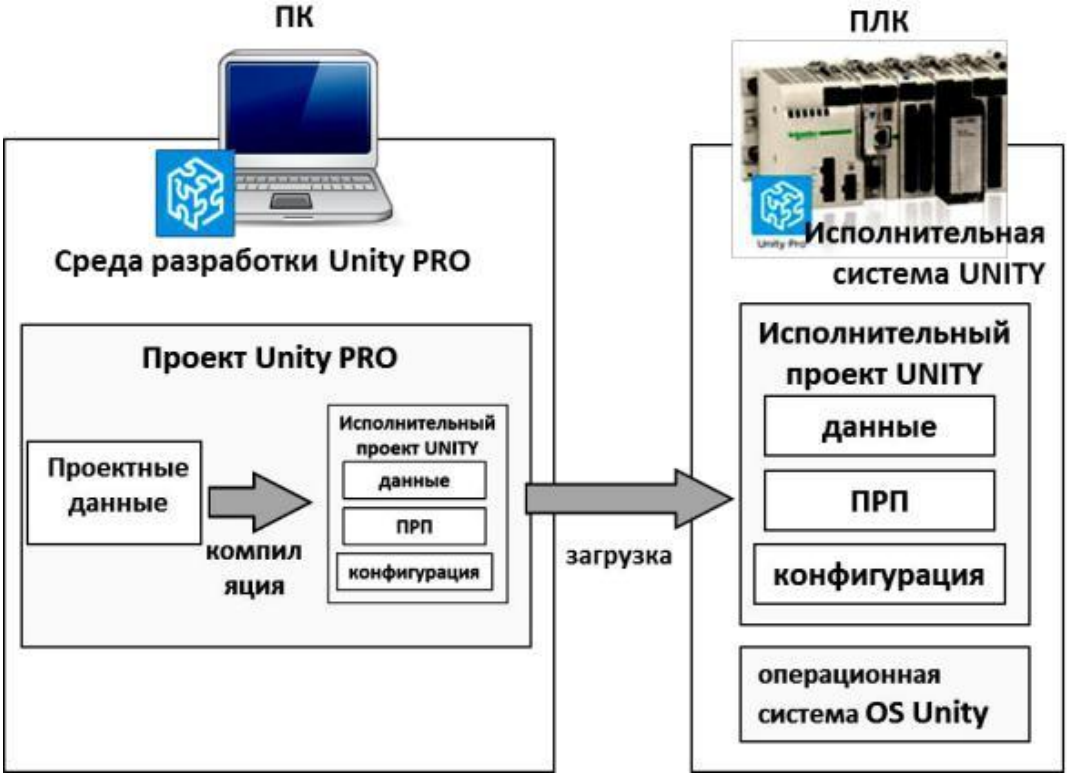


Рисунок 5.7 – Среда разработки и исполнения пользовательской задачи в ПЛК

После создания пользовательская программа совместно с ядром-интерпретатором загружается в целевой промышленный контроллер для исполнения. Ядро-интерпретатор, как следует уже из его названия, транслирует пользовательский алгоритм во время исполнения. Это позволяет сконцентрировать машинно-зависимый код и таким образом снизить накладные расходы при переходе на другой контроллер. Однако, сразу необходимо отметить, что интерпретационная модель имеет недостаток - она всегда снижает показатели эффективности исполнения программы.

Раздел 6. Особенности исполнения программ в промышленных контроллерах и связанные с этим языковые конструкции

Для исполняющей системы контроллер с загруженной программой представляется приведенным на рисунке образом.

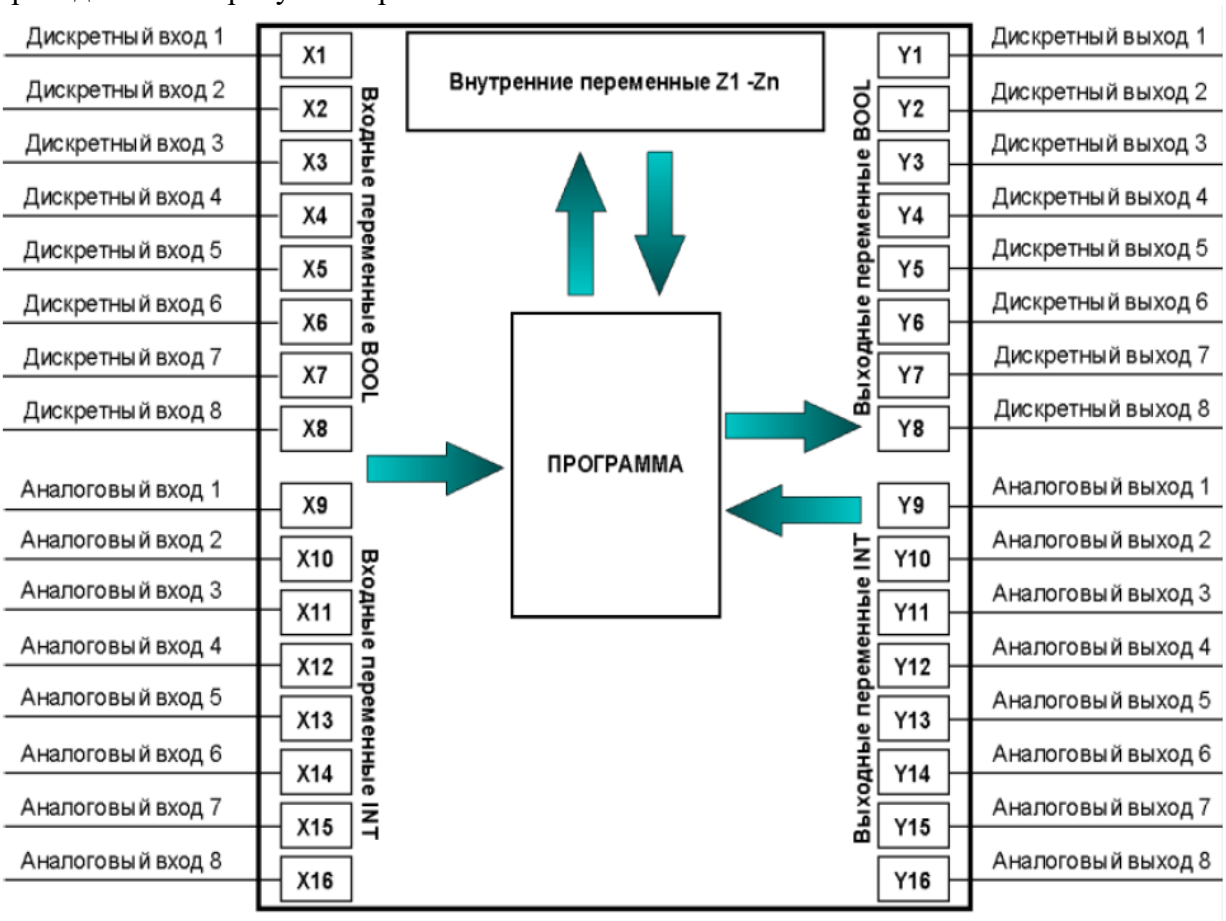


Рисунок 6.1 – логическая структура пользовательской задачи

Программа – это набор инструкций, производимых с данными – операндами. В случае промышленного контроллера операндами являются входные и выходные сигналы, а также внутренние переменные. Входные сигналы поступают в пользовательскую задачу из модулей контроллера – модулей аналогового и дискретного ввода, коммуникационных модулей. В основном используются два типа операндов – числовые и дискретные (булевы). Числовые операнды могут иметь формат с плавающей точкой – REAL или целочисленный. Целочисленные значения могут быть знаковыми и беззнаковыми и иметь разный размер от 1 до 8 байт.

В качестве операндов могут использоваться как переменные, объявленные в разделе объявлений, так и литеральные константы. Имя переменной должно содержать только буквы латинского алфавита, цифры и символ подчеркивания. Имя переменной не может начинаться с цифры. Синтакисис языков IEC-61131.3 является нерегистрочувствительным, т.е. можно писать имя переменной как строчными, так и заглавными буквами. Имя переменной должно быть содержательным. Если имя содержит несколько слов, то каждое слово начинается с прописной буквы. Нельзя использовать для имени слова, зарезервированные языком для других синтаксических конструкций. Некоторые кириллические символы совпадают по написанию с латинскими буквами, но для компилятора это другие символы. При копировании программы из электронных

текстов не стоит удивляться возникновению синтаксических ошибок, ведь авторы не всегда переключают раскладку клавиатуры.

Тип переменной должен быть объявлен в разделе объявлений и не может изменяться в процессе выполнения программы. Также в разделе объявлений может быть задано начальное значение переменной. Но это имеет смысл в основном для внутренних переменных программы. Производить арифметические и логические операции можно только с данными одного формата. Если нужно провести операцию с данными разного формата, то их приводят к одному формату, как правило, к формату результата выполнения операции. Для преобразования форматов переменных используются встроенные функции. Например, функция INT_TO_UINT переводит знаковое целое в беззнаковое:

(A : INT; B : UINT; - в разделе объявлений)

B := INT_TO_UINT(A);

Для перевода вещественных значений в целочисленные можно использовать не только функцию преобразования типа, но и функцию усечения дробной части TRUNC:

(A : REAL; B : WORD; - в разделе объявлений)

B := Real_TRUNC_WORD (A);

Переменные могут быть:

Внутренние – объявляются в разделе Var. Используются только внутри программного модуля, в котором они объявлены. Имеют свойство сохранять значения.

Внутренние без сохранения значений. Объявляются в разделе VAR_TEMP. Значения обновляются при каждом вызове перограммного модуля.

Глобальные - доступны во всех модулях программы. Объявляются в разделе VAR_GLOBAL. Сохраняют свое значение

Внешние. Используются для связи с другими модулями промышленного контроллера. Объявляются в разделе VAR_EXTERNAL. Обязательно содержат физический адрес данных. Для обозначения физического адреса используется специальный символ % (знак процента). Входные переменные от модулей (каналов) контроллера обозначаются %I, выходные - %Q. После этих знаков идет тип данных, определенный в конфигурации контроллера: X или без символа (BOOL), B(BYTE), W(WORD), DW(DWORD), LW(LWORD). А затем номер, в иерархической привязке. Например, %QB7.5 – отправить результат в 5 байт 7 модуля в конфигурации контроллера. %IW2.5.7.1 получить значение переменной из первого "канала" (слова) седьмого "модуля" на пятом "стеллаже" второй "шины ввода/вывода" программируемого контроллера.

Входящие – переменные, значения которых модуль получает при вызове из программы. Объявляются в разделе VAR_INPUT.

Исходящие – переменные, значения которых модуль передает в вызывающую программу. Объявляются в разделе VAR_OUTPUT.

Литеральные константы – это значения, написанные непосредственно в коде программы. В процессе исполнения программы эти значения никак изменить нельзя. Тип литеральной константы задается специальным символом «#» (решетка), унарный минус, установленный перед цифрами, символизирует отрицательное значение. По умолчанию запись, состоящая из цифр, и не содержащая десятичной точки, является целым числом, например, «-123». Запись с десятичной точкой считается вещественным числом, например, «123.0». Для действительных чисел нередко применяется экспоненциальная запись с символом «E»: «-1.23E2». Двоичный литерал: 2#1111_1111, восьмеричный литерал: 8#377, шестнадцатеричный литерал: 16#FF. Типизированный литерал: INT#-123, BOOL#0. Сложная типизация UINT#16#89AF. Булевы значения могут быть заданы специальными словами FALSE и TRUE.

Символьно-строковый литерал однобайтовых символов является последовательностью нуля или более символов, с предшествующим и завершающим

символом одиночной кавычки: ‘a2F'. В строках однобайтовых символов, трехсимвольная комбинация символа доллара ($) с двумя следующими шестнадцатиричными символами интерпретируется как шестнадцатиричное представление восьмибитового кода символа (Юникод): '$0A’. Символьно-строковый литерал двухбайтовых символов является последовательностью 0 или более символов из набора символов ИСО/МЭК 10646, с предшествующим и завершающим символом двойной кавычки: “A”. В строках однобайтовых символов, трехсимвольная комбинация символа доллара ($) с двумя следующими шестнадцатиричными символами интерпретируется как шестнадцатиричное представление восьмибитового кода символа: “$00C4”. Специальные двухсимвольные комбинации приведены в таблице 6.1.

Таблица 6.1 – Специальные строковые комбинации

Номер	Описание	Комбинации
1	Знак доллара	$$
2	Единичная кавычка	$'
3	Перевод строки	$L или $l
4	Новая строка	$N или $n
5	Прогон (перевод) страницы	$P или $p
6	Возврат каретки	$R или $r
7	Табуляция	$T или $t
8	Двойная кавычка	$»

Данные продолжительности времени ограничиваются слева ключевым словом T#, TIME# или LTIME#. Поддерживается представление данных о продолжительности времени в терминах дней, часов, минут, секунд и долей секунды, или любой их комбинации. Обозначения единиц измерения времени: d(сутки), h(часы), m(минуты), s(секунды), ms (миллисекунды), us (микросекунды), ns (наносекунды). Примеры: T#14ms, t#25h15m, LT#14.7s.

Для процедур, запускаемых по расписанию, удобнее использовать литералы даты и времени. Ключевые слова даты DATE# или D#, времени суток TIME_OF_DAY# или TOD#, даты и времени: DATE_AND_TIME# или DT#. Примеры: D#1984-06-25, TOD#15:36:55.36, DT#1984-06-25-15:36:55.360_227_400.

Функция - элемент языка, который во время выполнения обычно вырабатывает результат в виде одного элемента данных и, возможно, дополнительные выходные переменные. Функции бывают стандартные (т.е. определенные стандартом IEC 61131-3) и пользовательские. Значение функции зависит только от значений входящих переменных.

Стандартом определен довольно значительный ряд арифметических и алгебраических функций над числовыми операндами, такие как, тригонометрические функции (SIN, COS, TAN, ASIN, ACOS, ATAN), модуль (ABS), квадратный корень (SQRT), логарифмические (LN, LOG, EXP). Также определены битовые и поразрядные операции: сдвига вправо (SHR, ROR) и влево (SHL, ROL). Операции над булевыми значениями: AND, NOT, OR, XOR. Операции выбора и сравнения, операции над символьными и строковыми данными.

Функциональный блок - это программный компонент, который представляет хорошо определенную часть программы для обеспечения модульности и структуризации. В отличие от функции выходные значения функционального блока зависят не только от значений входных переменных, но и от внутреннего состояния. Поэтому в отличие от функции каждому экземпляру функционального блока присваивается оригинальное имя, как переменной. Функциональные блок также могут быть стандартными и пользовательскими.

Тип пользовательского функционального блока определяется в отдельном модуле программы. Раздел объявлений функционального блока содержит входные, выходные и внутренние переменные. А может содержать глобальные и внешние переменные. Созданный пользователем тип функционального блока может использоваться в программе промышленного контроллера многократно или однократно, или вообще не использоваться. Использование входных и выходных параметров, определенных правилами на рисунке 6.2.

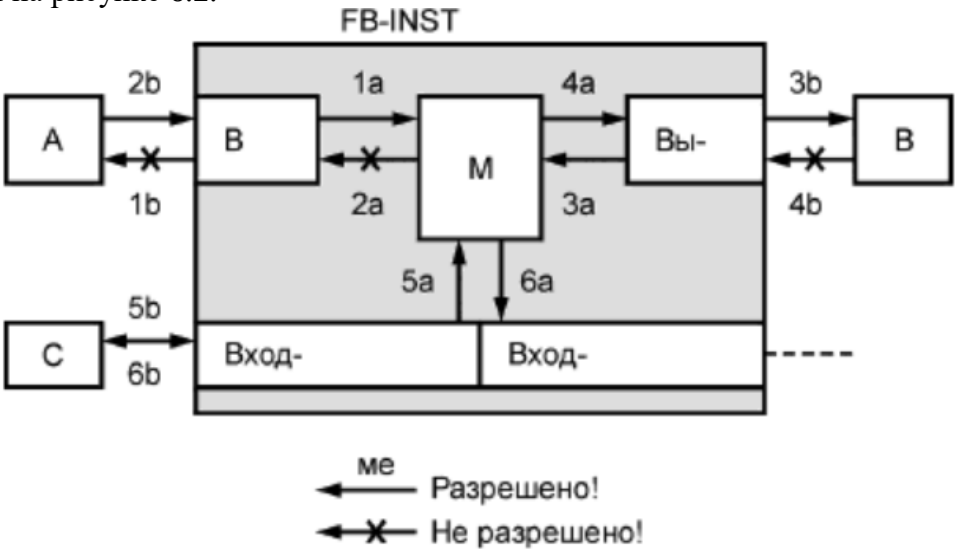


Рисунок 6.2 – Правила использования входных и выходных переменных

Когда объявляется экземпляр функционального блока, начальные значения входных, выходных и общих переменных могут объявляться в перечне, заключенном в скобки, с последующим оператором присваивания, который следует за идентификатором типа функционального блока. Например: empLoop: PID:=(PropBand:= 2.5, Integral:= T#5s); . Элементы, для которых начальные значения не перечислены в описанном выше перечне инициализации, получают неявное начальное значение, объявленное для этих элементов в объявлении типа функционального блока.

Вызов экземпляра функционального блока может быть представлен в текстовой или графической форме. Текстуальный вызов функционального блока состоит из имени экземпляра с последующим перечнем параметров. В графическом представлении имя экземпляра функционального блока располагается над блоком.

Стандартные функциональные блоки взаимодействия для программируемых контроллеров определены в МЭК 61131-5. Данные функциональные блоки предоставляют функциональность взаимодействия, такую как средства проверки устройств, сбор данных опроса, запрограммированный сбор данных, управление параметрами, управление с взаимоблокировкой, запрограммированное аварийное оповещение, управление и защита соединений.

Триггеры. Одним из важнейших стандартных функциональных блоков является RS и SR триггеры. Отличаются они приоритетом включения (SR) или отключения (RS). Эти триггеры являются типичными бистабильными элементами, которые могут использоваться для формирования команд телеуправления. После подачи команды на вход Set выход триггера Q1 принимает значение TRUE и находится в нем до тех пор, пока не будет подана команда на вход Reset.

Импульсные триггеры. Предназначены для определения переднего R_Trig или заднего F_Trig фронта сигнала. Дело в том, что команда, поданная на контроллер, например, при помощи нажатия кнопки, замыкающей датчик типа сухой контакт, будет обрабатываться на каждом цикле контроллера, пока кнопка не будет отпущена. Если при этом ведется счетчик срабатываний, то он увеличиться многократно, в том числе с

возможностью переполнения типа. Чтобы этого избежать, используют импульсные триггеры, которые срабатывают однократно, переходя в значение TRUE лишь в одном цикле контроллера, при прохождении фронта.

Блоки таймеры. При программировании контроллера очень часто требуется внести задержку между получением сигнала и выполнением команды. В классических языках программирования используется функция Sleep или Delay. Но применение подобных команд в программах, исполняемых в контроллерах, недопустимо. Ведь в этом случае контроллер останавливает выполнение инструкций до истечения заданного времени. За это время может произойти все что угодно – авария, пропущенные сигналы телесигнализации и необработанные команды телеуправления. В микроконтроллерах задержка реализуется обращением к системному таймеру, которому задают время, по истечению которого таймер прерывает обработку главной программы для выполнения соответствующей процедуры. Программирование промышленных контроллеров не привязано к архитектуре применяемых процессоров, поэтому прямое обращение к системному таймеру в них не предусмотрено. Задержка исполнения алгоритма реализуется через специальные функциональные блоки: TON - задержка включения на заданное время, TOF – задержка отключения на заданное время, TP – формирование импульса через заданное время. Пример текстового использования блока TON:

В разделе объявлений:

```
tm: TON;
ts1, tu1: BOOL;
```

В тексте программы

```
tm(IN := ts1, PT := T#5s | tu1:= Q);
```

или

```
tu1:= tm.Q(IN := ts1, PT := T#5s);
```

Пример графического использования того же блока:

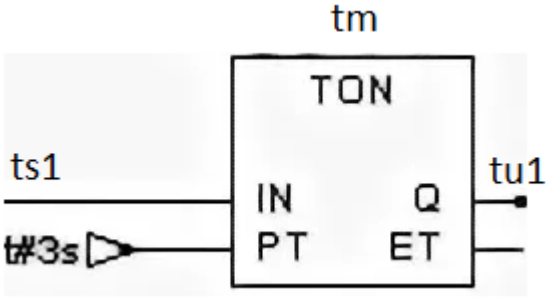


Рисунок 6.3 – Графическое представление функционального блока

Операторы являются одной из составляющих частей любого языка программирования, в том числе и IEC-61131.3. Программа в основном состоит из инструкций – операций и вызовов функций или функциональных блоков – выполняемых одна за другой. Если в выполняемом алгоритме есть точки ветвления, то при выполнении определенных условий нужно изменить порядок следования инструкций. Именно это и осуществляют операторы.

В языках программирования выделяют операторы условий (IF THEN ELSE) и выбора (CASE), циклов (FOR, REPEAT, WHILE), выхода из подпрограммы (RETURN).

При программировании контроллера всегда нужно учитывать, что основная программа выполняется циклически. На каждом цикле значения входных переменных преобразуются в значения выходных. Чем сложнее алгоритм пользовательской задачи, тем медленнее выполняется цикл. Если за время цикла пользовательской задачи какая-то переменная успеет измениться более одного раза, то по крайней мере одно из ее значений не будет обработано. Если в технологическом процессе требуется строго заданное время реакции на событие, то длительность цикла должна быть меньше этого времени. Если

производитель контроллера предоставляет инструменты для измерения реального времени цикла, то следует ориентироваться именно на этот показатель. Если же такого инструментария нет, то время цикла может быть измерено внутри пользовательской задачи. Для этого создают специальную внутреннюю переменную счетчик типа DWORD, значение которой увеличивают на единицу на каждом цикле при подаче булевого значения TRUE в специальную команду ТУ. На входе к ТУ устанавливают блок TOF с задержкой в 1 мс. После загрузки программы в контроллер подача указанного ТУ дает на выходе переменной счетчика количество циклов, выполненных за одну миллисекунду. Если время цикла превышает допустимый предел, то следует уменьшить вычислительную нагрузку на контроллер. Например, перенести часть вычислений на другой элемент системы (промышленный сервер) или установить дополнительный контроллер для обслуживания части сигналов.

Языковые конструкции, определенные в стандарте IEC 61131-3 достаточно разнообразны и могут использоваться для достаточно гибкой настройки алгоритмов пользовательской задачи.

Оценочные средства для разделов 4-6

Вопросы для самопроверки:

1. Почему при конфигурировании контроллеров нежелательна технология Plug and play?
2. Что подразумевает возможность горячей замены модулей контроллера?
3. Значения каких конфигурационных параметров модулей контроллера нельзя оставить по умолчанию?
4. Каким образом осуществляется резервирование контроллеров?
5. На каких специалистов ориентирован язык Ladder Diagram?
6. Чем отличаются операторы WHILE и REPEAT?
7. Почему на языке FBD редко разрабатывают сложные алгоритмы?
8. Чем отличается функциональный блок от функции?
9. Для чего нужны в программе операторы?
10. В какой ситуации полезно применение импульсного триггера F_Trig?

Тестовые вопросы:

1. Какие из указанных языков программирования ПЛК определены стандартом IEC-61131-3?

а) C++, FBD,
б) Java,
в) ST,
г) Python,
д) LD.

2.Чем отличается функциональный блок от функции?

а) зависит от многих переменных;
б) сохраняет свое состояние;
в) используется только в языке FBD;
г) ничем

3. Как называются переменные, значения которых передаются в исполнительные модули ПЛК?

а) глобальные;
б) внешние;
в) входящие;
г) исходящие.

4. Что такое "литеральная константа"?

а) значение, заданное в разделе объявлений;
б) значение, заданное в тексте программы;
в) общеизвестное значение;
г) нет такого понятия

5. Для чего используются внутренние переменные пользовательской задачи?

а) для записи значений с датчиков;
б) для подачи команд управления;
в) для сохранения промежуточных результатов вычислений;
г) для хранения уставок телерегулирования

6. Что делает в программе оператор?

а) изменяет значения переменных;
б) удаляет лишние объекты;
в) изменяет порядок следования инструкций;
г) сохраняет данные в файл

7. Язык релейных схем (LD):

а) используется для создания систем на реле;
б) применяется специалистами по релейным схемам для программирования ПЛК;

в) является языком самого низкого уровня;

г) не используется для программирования ПЛК

8. Какие стандартные функции можно использовать для преобразования вещественного значения в целое?

а) оператор присваивания;

б) _TO_;

в) ABS ;

г) _TRUNC_.

9. Опишите назначение оператора CASE.

Открытый вопрос – дайте развернутый письменный ответ.

10. Запишите физический адрес внешней переменной типа Byte, входящей в пользовательскую задачу из 7 байта 3 модуля контроллера.

Открытый вопрос – дайте развернутый письменный ответ.

Индивидуальное задание – разработка программного кода.

Уровень сложности – легкий. Разработать программу перекладки значений. Индивидуальные параметры: язык программирования, физические адреса и типы сигналов.

Уровень сложности – средний. Разработать программу подачи сигнала ТУ и формирования ответа. Индивидуальные параметры: язык программирования физические адреса сигналов, характеристика объекта управления.

Уровень сложности – сложный. Разработать программу ПИД-регулирования параметра. Индивидуальные параметры: язык программирования, физические адреса сигналов, характеристика объекта управления.

Раздел 7. Типовые алгоритмы обработки данных и управления

В данной лекции будут рассмотрены алгоритмы обработки данных в промышленных контроллерах, встречающиеся наиболее часто.

В промышленных контроллерах редко встречается высшая математика. Решение дифференциальных уравнений, взятие многомерных интегралов, разложение в ряды и вариационное исчисление останутся за кадром нашего рассмотрения. Такие алгоритмы чаще реализуются на компьютерах исследователей, разработчиков и проектировщиков. Когда дело доходит до реальных алгоритмов управления, более востребованными являются простые вычислительные схемы. Так, пожалуй, самым распространенным в телемеханике является **алгоритм перекладки значений**. Т.е. значение, полученное от одного модуля, без изменений перекладывается во входящий сигнал другого модуля. По сути это можно реализовать операцией присваивания, или соединением контактов в релейной схеме. Иногда требуется преобразовать тип значения, т.к. тип исходного сигнала может не совпадать с типом входного сигнала модуля. А в ряде случаев желательно проверить еще несколько условий:

Не произойдет ли переполнение принимающего типа? В этом случае значение будет усечено до максимального значения типа, что приведет к искажению значения.

Достоверно ли перекладываемое значение? Для установления этого факта обычно требуется побитная интерпретация статуса сигнала. Недостоверные значения не должны обрабатываться или отправляться по каналам передачи данных. Зато сообщение о полученном недостоверном значении должно попасть в статистическую (диагностическую) информацию.

Находится ли значение в заданном диапазоне? Диапазон изменения значения может быть задан, например, исходя из характеристик датчика. При выходе значения за диапазон переполнения значения наблюдаться не будет, но характеристика технологического процесса будет искажена, например, нелинейностью характеристики датчика. Выход значения за заданный диапазон можно контролировать по флагу в статусе сигнала или путем логического сравнения с границами диапазона внутри пользовательской задачи. Указанные выше проверки стоит проводить не только в случае перекладки, но и вообще, для любого полученного значения входящих внешних сигналов.

Обработано ли принимающей стороной предыдущее значение сигнала? Эта проверка касается готовности приемника сигнала к обработке следующего значения. Для выяснения этого обстоятельства. Как правило, следует знать логику работы принимающего модуля и осуществлять контроль по приходящим от него диагностическим сигналам.

На связи ли приемник сигнала? Может статься, что модуль-получатель неисправен, или недоступен в настоящий момент. Стоит предусмотреть специальную процедуру обработки такого значения. Например, поместить ее в буферный массив для последующей отправки при появлении модуля на связи.

Алгоритмы противоаварийной защиты (ПАЗ) оборудования. Предназначены для предотвращения аварийных ситуаций. Как правило, такие алгоритмы автоматически подают команду отключения оборудования при выполнении заданного условия, соответствующего аварийной ситуации. Условия задаются относительно значений сигналов ТС и ТИ. Проверка должна осуществляться на каждом цикле работы контроллера, во всех режимах работы. Рекомендуется формулировать условие относительно ТС, т.к. они имеют более высокий приоритет, чем ТИ, их сбор (опрос) производится быстрее, т.к. не требует работы коммутаторов, АЦП. Это позволит обеспечить ускоренное время реакции на события. Выполнение алгоритма ПАЗ должно сопровождаться обязательной передачей сообщения в управляющий центр – технологический сервер, диспетчерский пункт или оператору системы. Деблокировка аварийной защиты, т.е. сигнал обратного включения оборудования, как правило, подается

оператором системы после устранения причин возникновения аварийной ситуации, т.к. автоматическая деблокировка может привести к повтору аварийной ситуации.

Перечень распространенных аварийных ситуаций:

1) Переполнение (опорожнение) емкости (резервуара). Контролируется по достижению максимально-допустимого (минимально-допустимого) уровня. Рекомендуется данное условие контролировать не по показаниям аналогового уровнемера, а по срабатыванию дискретного датчика, установленного на этом уровне. Возможная обработка: отключение систем, подающих поток жидкости в резервуар, отключение насоса или перекрытие входящей трубы секущей задвижкой.

2) Достижение максимального значения давления. Рекомендуется контролировать выполнение данного условия не по аналоговому датчику давления, а по срабатыванию порогового дискретного датчика или предохранительного клапана. Возможная обработка: отключение насоса или перекрытие напорной трубы секущей задвижкой.

3) Достижение максимальной температуры. Рекомендуется контролировать данное значение не по аналоговому датчику температуры, а по срабатыванию порогового дискретного датчика. Возможная обработка: выключение нагревательного элемента, включение охлаждающей системы.

4) Короткое замыкание в цепи питания. Превышение током заданного порогового значения при снижении характеристик производительности агрегата (например, оборотов двигателя или давления насоса). Возможная обработка: отключение питания

Алгоритм пересчета оцифрованных значений в инженерные величины также является весьма распространенным. Датчик является первичным преобразователем измеряемой величины в электрический сигнал. Электрический сигнал поступает в АЦП модуля аналогового ввода контроллера. Результатом работы АЦП является целое беззнаковое число с количеством ненулевых разрядов, соответствующим разрядности применяемого АПЦ. Так, в случае 8 разрядной АЦП, значение измеряемой температуры 82,5°С сначала преобразуется в значение тока от 4 до 20 мА, а затем ток переводится в число от 0 до 255. Предположим, получилось число 141. В каких единицах оно выражено? Нам удобнее работать с температурой, выраженной в градусах Цельсия или Кельвина, как она присутствует в формулах для вычислений других параметров процесса.

Для перевода следует знать, что все промышленные датчики работают в линеаризованном диапазоне. Т.е. пересчет в инженерные величины осуществляется по линейному закону. Только вместо коэффициентов a и b задаются так называемые ранги сигнала:

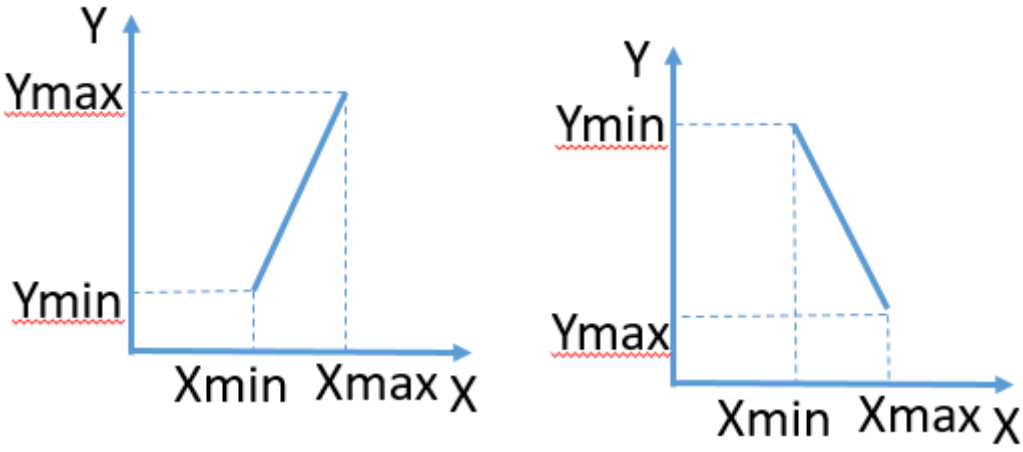
Минимальное инженерное значение - Ymin.

Минимальное физическое - Xmin, т.е. на выходе АЦП, соответствующее минимальному инженерному.

Максимальное инженерное значение – Ymax.

Максимальное физическое – Xmax, соответствующее максимальному инженерному. При этом Xmax>Xmin

Максимальное инженерное значение может оказаться меньше минимального инженерного, если в первичном преобразователе – датчике, используется физическое явление с обратно пропорциональным законом.



а) прямая пропорциональность б) обратная пропорциональность

Рисунок 7.1 – Пересчет значений с датчика

Вычисление значений технологических параметров в инженерных единицах имеет тип числа с плавающей точкой и осуществляется по формуле:

$$Y = Y_{min} + \frac{Y_{max} - Y_{min}}{X_{max} - X_{min}}(X - X_{min})$$

Конечно мы понимаем, что преобразование по линейному закону – это приближение, ведь большинство физических эффектов, применяемых при разработке датчиков носят нелинейный характер. Для такого преобразования используется малый отрезок характеристики датчика, на котором нелинейные искажения невелики. Но если характер зависимости сильно нелинейный и не позволяет использовать линейный пересчет в необходимом диапазоне, то используются, так называемые, датчики с двумя диапазонами оцифровки. Т.е. в пределах измерений задают две линеаризованных области. Для этого на график добавляют третью точку Xcp,Ycp.

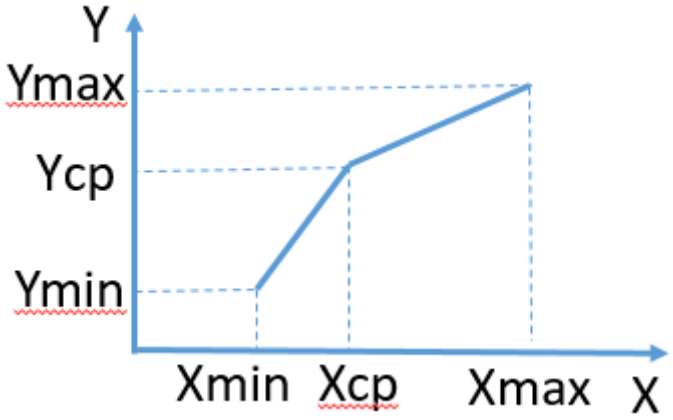


Рисунок 7.2 – Пересчет значения для датчика с двумя диапазонами оцифровки

В этом случае формула для пересчета имеет вид:

$$Y = \begin{cases} Y_{min} + \frac{Y_{cp} - Y_{min}}{X_{cp} - X_{min}}(X - X_{min}), X > X_{cp} \\ Y_{cp} + \frac{Y_{max} - Y_{cp}}{X_{max} - X_{cp}}(X - X_{cp}), X < X_{CP} \end{cases}$$

Алгоритм устранения дребезга датчиков. Что такое дребезг контактов? Когда срабатывание дискретного датчика, два металлических контакта замыкаются. Может

показаться, что контакт наступил мгновенно. Это не совсем правильно. Внутри коммутатора есть движущиеся части. Вначале датчик создает контакт между металлическими частями, но только в кратком разрезе микросекунды. Затем он делает контакт немного дольше, а затем еще немного дольше. В конце коммутатор полностью замыкается. Коммутатор скачет (дребезжит) между состояниями наличия и отсутствия контакта. «Когда коммутатор замыкается, два контакта фактически разъединяются и снова соединяются обычно от 10 до 100 раз за время, примерно равное 1 мс» («Искусство схемотехники», Хоровиц и Хилл, второе издание). Часто промышленный контроллер работает быстрее, чем дребезг, что приводит к тому, что оборудование думает, что датчик сработал несколько раз.

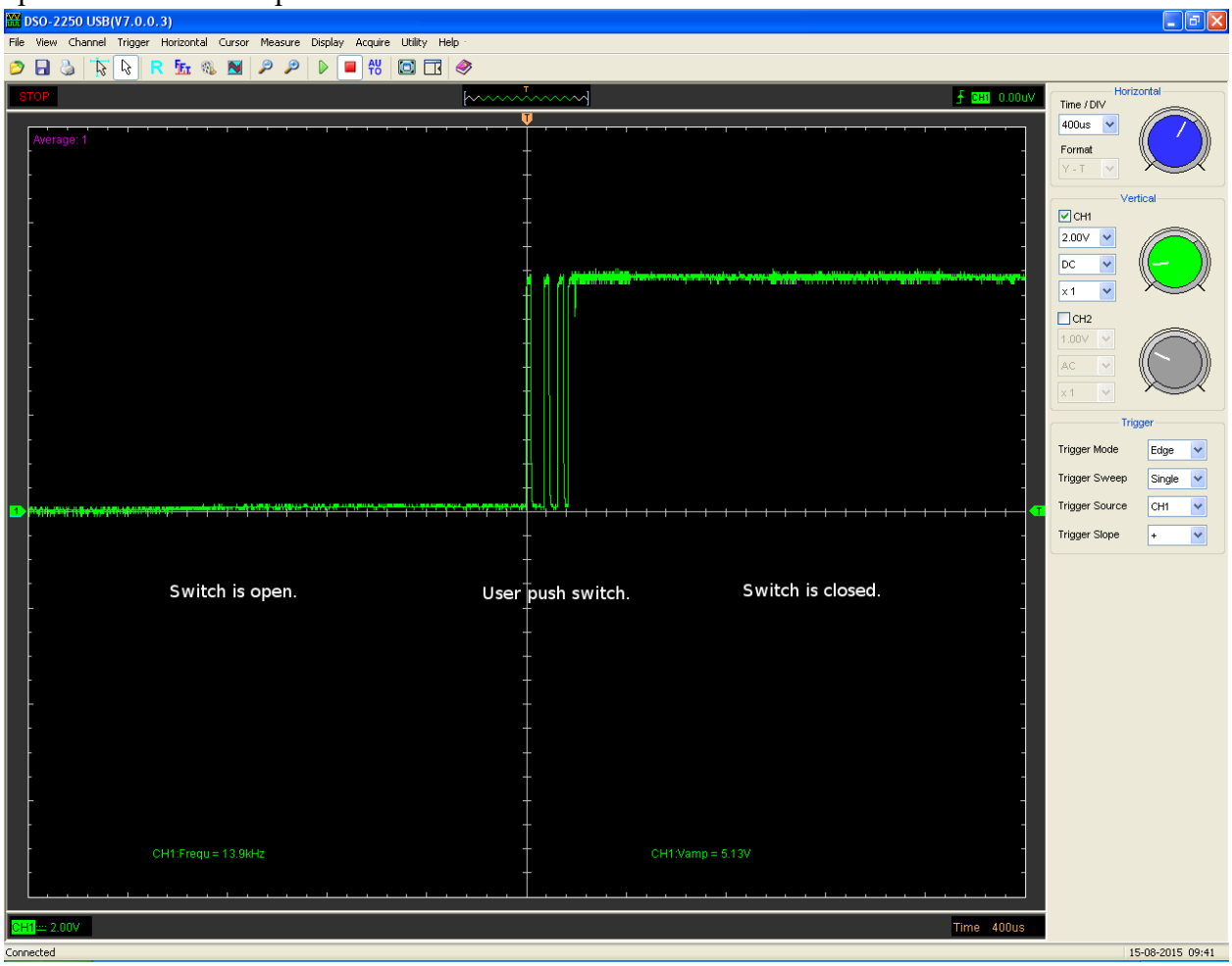


Рисунок 7.3 – Осциллограмма дребезга контактов

В цепях модуля дискретного ввода, как правило, приняты меры для устранения кратковременного дребезга контактов. Но один из самых эффективных способов – это устранение дребезга на программном уровне. Для этого можно использовать триггеры задержки по времени TON и TOF. Если сигнал держится достаточно долго, то значение стоит переключить. Интервал времени задержки выбирается опытным путем. Можно рекомендовать начать с 10 мс. Если этого недостаточно для подавления дребезга – увеличить вдвое. Если требования реального времени по данному сигналу оказались меньше полученного значения, то рекомендуется использовать датчик другого типа или схемотехническое решение с фильтром нижних частот.

Алгоритм подачи команды телеуправления. Команды телеуправления изменяют состояние технологического оборудования и процесса в целом, поэтому требуют особого внимания. Первичный контроль права авторизованной подачи команды осуществляется на диспетчерском пунктом и остается за кадром пользовательской задачи

промышленного контроллера. Команды ТУ поступают из диспетчерского пункта по промышленным сетям. При этом на уровне сетей обеспечивается правильность передачи данных. Поэтому все поступившие в пользовательскую задачу контроллера команды обязательны к исполнению.

Элементарное выполнение команды ТУ предполагает перекладку команды во входящий сигнал модуля дискретного вывода. Но в большинстве случаев используется более сложный, многоэтапный алгоритм. На первом этапе осуществляется подготовка оборудования к выполнению команды ТУ. Например, специальной командой оборудование переводится из местного (ручного) режима управления и в дистанционный. Выполнение подготовительного этапа контролируется по состоянию соответствующих сигналов ТС. Если подготовительный этап не проходит, то может быть сде5лана повторная попытка его провести, или формируется сообщение для отправки в диспетчерский пункт о невозможности выполнения команды.

На втором этапе подается сама команда ТУ. Выполнение команды контролируется по состоянию датчиков и может занимать значительное время. Например, закрытие задвижки на магистральном нефтепроводе осуществляется 6 минут, в течение которых задвижка плавно перекрывает сечение трубы и входит в уплотнение с корпусом. После завершение команды управления может осуществляться завершающий этап, связанный с переводом исполнительного устройства обратно, в режим местного управления.

Временное усреднение значений аналоговых датчиков. Аналоговые датчики не дребезжат, в отличие от дискретных контактов, но их оцифрованные значения тоже подвержены зашумлению. Для подавления высокочастотных шумов можно использовать на входе фильтр нижних частот. Возможно, хорошим решением стал бы аппаратный фильтр, например, RC цепочка. Но, во-первых, условия измерений меняются, и хотелось бы иметь возможность подстраиваться, во-вторых, диапазон номиналов электронных компонент не всегда позволяет задать схемотехническому фильтру оптимальные характеристики. Поэтому хорошим выходом может стать цифровой ФНЧ, реализованный в пользовательской задаче.

Для реализаций фильтра необходимо хранить в массиве памяти n последних отсчетов сигнала и сдвигать их при получении очередного значения. Расчет фильтра требует фиксированного времени частоты дискретизации сигнала. Поэтому выполнять алгоритм нужно не по циклу контроллера, а через равные промежутки времени, активируемые, например, функциональным блоком ТР. Классическая формула ФНЧ:

$$y(n) = \sum_{k=0}^{N-1} (h_k(n) \times x(n-k))$$

Коэффициенты hk идеального ФНЧ рассчитываются по формуле:

$$h_D(n) = 2f_c \times \frac{\sin(nw_c)}{nw_c} \text{ для } n \neq 0$$

$$h_D(n) = 2f_c \text{ для } n = 0$$

Где fc и wc – линейная и круговая частота среза.

Для практических применений добавляют весовую функцию w(n):

$$h(n) = h_D(n) \times w(n)$$

Где w(n) рассчитывается по формуле:

$$w(n) = 0{,}42 - 0{,}5 \times cos\left(\frac{2\pi n}{N-1}\right) + 0{,}08 \times \cos\left(\frac{4\pi n}{N-1}\right)$$

Алгоритмы блокировки неверных действий персонала. В ряде случаев следует предусмотреть возможности блокировки некорректных команд. Т.е. после получения команды телеуправления следует проверить возможность ее исполнения. Например,

подача потока жидкости в наполненный резервуар, или увеличение нагрева при достижении максимальной температуры. Команда, приводящая к аварийному состоянию технологического процесса не должна передаваться на исполнительный механизм. Таким образом, для каждого ТУ должно быть сформировано условие его выполнения.

Алгоритм регулирования. Регулирование параметров производится по принципу управления по отклонению. Т.е. для правильного формирования управляющего воздействия осуществляется мониторинг текущего значения параметра и его отклонения от желаемого.

Рассмотрим на примере регулирования температуры технологического процесса. Наиболее простой закон регулирования температуры - позиционный. При этом методе, на нагреватель подается полная мощность до достижения заданного значения температуры, после чего подача мощности прекращается. Несмотря на это, разогретый нагреватель продолжает отдавать тепло, и температура объекта какое-то время продолжает нарастать, что приводит к перегреву, иногда значительному. При последующем остывании объекта, по достижении заданного значения температуры, на нагреватель вновь подается полная мощность. Нагреватель сначала разогревает себя, затем окружающие области объекта, и, таким образом, охлаждение будет продолжаться до тех пор, пока волна тепла не достигнет датчика температуры. Следовательно, реальная температура может оказаться значительно ниже заданного значения. Таким образом, при позиционном законе регулирования возможны значительные колебания температуры около заданного значения.

Этот недостаток можно уменьшить или даже вовсе устранить, применяя пропорционально-интегрально-дифференциальный закон регулирования (ПИД закон). ПИД предполагает уменьшение мощности, подаваемой на нагреватель, по мере приближения температуры объекта к заданной температуре. Кроме того, в установившемся режиме регулирования по ПИД закону находится величина тепловой мощности, необходимой для компенсации тепловых потерь и поддержания заданной температуры.

Пропорционально - интегрально-дифференциальный закон регулирования обеспечивает значительно более высокую точность поддержания температуры, чем позиционный. Мощность N, которая должна выделяться нагревателем, выраженная в процентах от его максимальной мощности, рассчитывается по формуле:

$$N = \frac{100}{Kp}\left(\Delta T + \frac{1}{Ki}\int_0^1 \Delta T dt - Kd\frac{dT}{dt}\right)$$

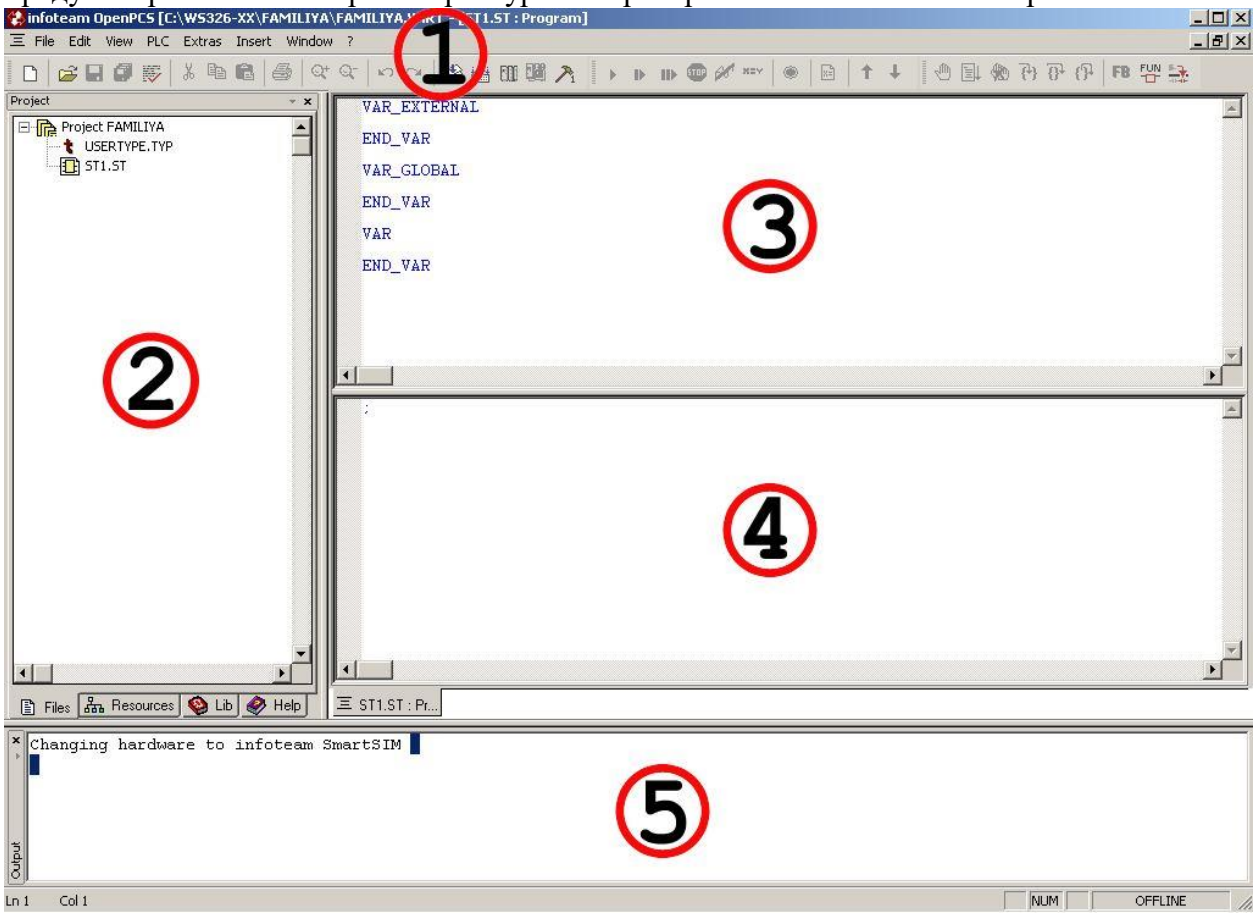
где Kp, Ki, Kd - пропорциональный, интегральный и дифференциальный коэффициенты регулирования соответственно (ПИД коэффициенты). Алгоритмов экспериментального и теоретического определения коэффициентов достаточно много. В частности, для нестационарных систем есть алгоритмы самонастройки коэффициентов.

Гистерезис регистрации нового состояния. Одной из важных задач автоматизации является идентификация состояния каждого технологического объекта и процесса в целом. Идентификация осуществляется на основе анализа взаимного состояния датчиков. В реальном технологическом процессе могут наблюдаться пограничные состояния, приводящие к частому изменению состояния условий. В этом случае, для диспетчера наблюдается картина, сходная с упомянутым выше дребезгом дискретных датчиков. Для того, чтобы устранить этот неприятный эффект, переход между режимами осуществляется не по одному граничному значению параметра, а по двум. Например, если задано максимальное значение температуры 70°C, то переход в режим перегрева осуществляется при достижении 70°C, а обратный переход в нормальное состояние фиксируется при падении температуры до 69,5°C. Конкретное значение величины гистерезиса задается из технологических ограничений, накладываемых на точность измерения контролируемого параметра.

Раздел 8. Отладка программ в промышленных контроллерах

Прежде чем программа заработает на объекте промышленной автоматизации она проходит многоступенчатую проверку и отладку. Ведь цена ошибки очень велика. Так сбой в работе контроллера, вызванный некорректным программным обеспечением, может привести к потере функций управления, аварийному состоянию оборудования, финансовым потерям и опасности для жизни и здоровья сотрудников.

Первым этапом отладки написанной программы является проверка и исправление синтаксиса. Проверка синтаксиса обязательно осуществляется в средах программирования промышленных контроллеров. Обычно в нижней части предусмотрена область просмотра журнала проверки и ошибок компилятора.



1 – Меню и панель инструментов
2 – Менеджер конфигураций и проектов
3 – Редактор раздела объявления переменных
4 – Редактор кода программы
5 – Окно диагностики и тестирования

Рисунок 8.1 – Окно программы редактирования среды программирования OpenPCS

Но нужно отметить, что успешное прохождение проверки говорит только об отсутствии синтаксических ошибок, и вовсе не гарантирует правильную работу алгоритмов, заложенных в пользовательскую задачу контроллера. Проверка синтаксиса легко обнаруживает необъявленные переменные, неопределенные значения, неверное написание имен функций и функциональных блоков. Но она не способна гарантировать правильную логику работы программы. Поэтому отладку нужно совмещать с тестированием проекта, и обязательно проверить реализацию всех функций, предусмотренных техническим заданием.

Одним из доступных средств отладки является симулятор контроллера, предоставляемый каждой средой программирования. В симуляторе реализованы модули

дискретного ввода и вывода, средства онлайн симуляции значений сигналов, что дает возможность полноценно проверить логику работы.

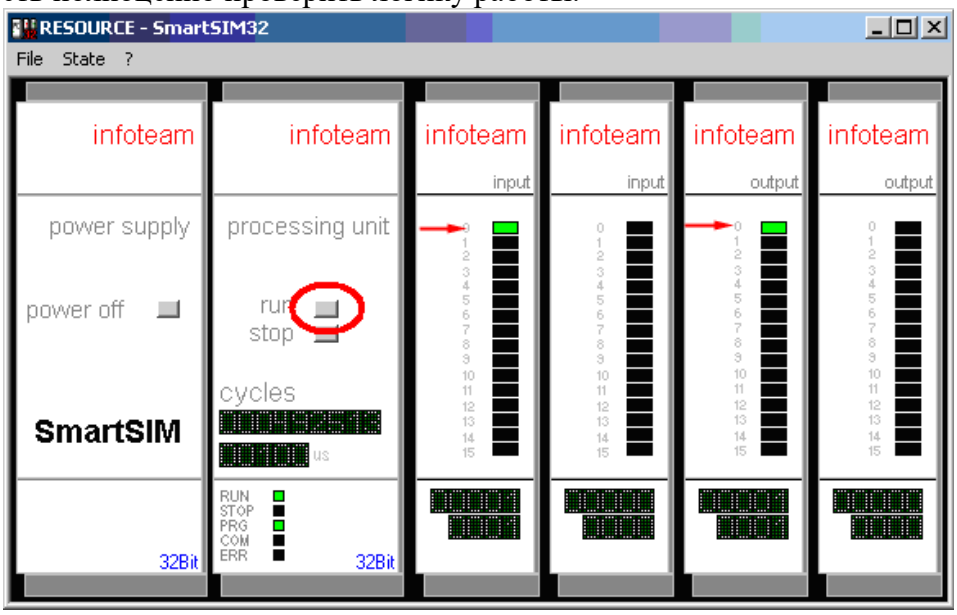


Рисунок 8.2 – Пример симулятора контроллера

На этапе отладки программы в реальном контроллере среда программирования дает возможность подключиться к среде исполнения пользовательской задачи в онлайн режиме. При этом доступны не только значения текущих сигналов контроллера, но и состояния внутренних переменных пользовательской задачи, результаты промежуточных вычислений. Имеется также возможность записать значения в некоторые сигналы, если они не являются входящими из других модулей контроллера.

Одним из камней преткновения в диагностике контроллера является отсутствие у него пользовательского терминала. На передней панели контроллера иногда расположены индикаторы его работы.

Рисунок 8.3 – Индикация работы промышленного контроллера

Чтобы посмотреть значения сигналов необходимо подключиться к контроллеру пультом, роль которого могут играть ноутбук, планшет или стационарный компьютер при удаленном подключении по сети. Специальные диагностические инструменты,

установленные на устройство, представляют собой монитор, в котором доступен список модулей, их текущих параметров, значения входящих и исходящих сигналов:

Имя	Тип	Значение	Мин.	Макс.	Рек.	Комментарий
Driver	String	FIFO_New				Имя драйвера
CopyProc	Bool	0	0	1	0	Признак разрешенных копий процесса
ConfTA	U2	2000	500	10000	2000	Тайм-аут ожидания подтверждения данных от ЦП (мс)
Reserv1	I2	0	-32768	32767	0	Резерв 1
Reserv2	I2	0	-32768	32767	0	Резерв 2
Reserv3	I1	0	-128	127	0	Резерв 3
Reserv4	I1	0	-128	127	0	Резерв 4
IntegrTime	I2	100	20	5100	1000	Время интегрирования (мс). Должно задаваться кратно 20 мс
CalibrPer	U1	10	1	100	10	Периодичность калибровки
EnblCycle...	U1	0	0	1	0	Признак циклической передачи данных (0-передавать цик...
DsblDiag	U1	1	0	1	1	Признак выдачи диагностики (0-выдается. 1-не выдается)

Рисунок 8.4 – Вкладка с текущими значениями параметров модуля контроллера

Также возможности отладочного просмотра предлагает среда программирования, подключенная к контроллеру в режиме отладки. При этом в окне диагностики и тестирования можно сформировать список сигналов для мониторинга. Некоторым сигналам можно установить тестовые значения. Но на работающем проекте делать это категорически не рекомендуется, поскольку может привести к подаче управляющей команды. Только на диагностическом стенде.

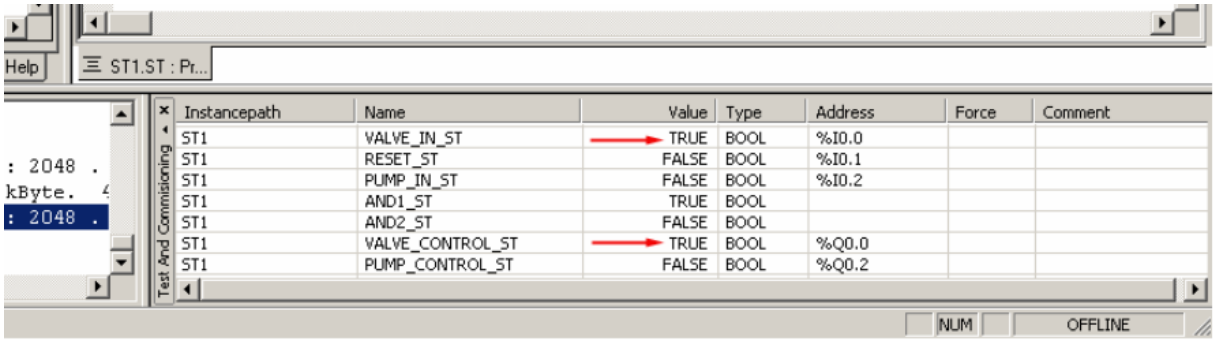


Рисунок 8.5 – Мониторинг онлайн значений сигналов в среде программирования

Большинство систем программирования ПЛК предоставляют возможности пошаговой отладки. При этом, конечно, нет необходимости пошагово проходить всю программу. Нужно лишь определить точку останова, с которой начнется пошаговое выполнение. Точку останова нужно установить в той процедуре, где разработчик подозревает наличие логической ошибки обработки данных.

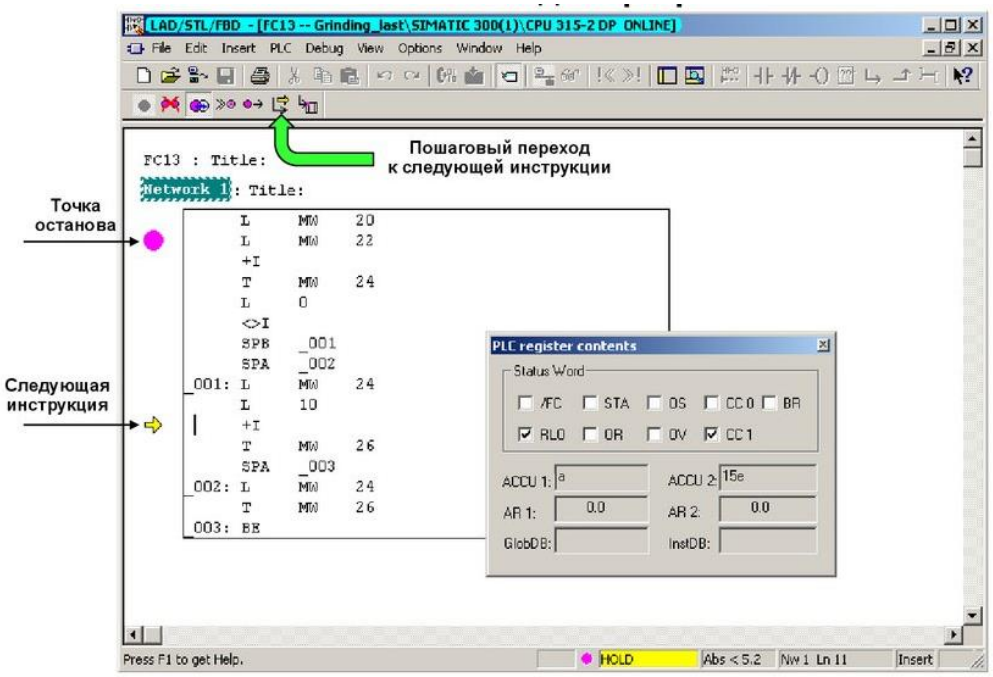


Рисунок 8.6 – Пошаговая отладка программы в контроллере

Хорошим вспомогательным инструментом является журнал работы контроллера, в котором фиксируются все операции, изменяющие состояния контроллера. Журнал хранится в памяти контроллера и доступен при помощи программы просмотра, предоставляемой производителем контроллера. Параметры логирования могут быть настроены в конфигурации контроллера. Не рекомендуется сохранять в журнал всю доступную информацию, поскольку в этом случае сильно увеличивается объем журнала, и поиск необходимых сообщений сильно затрудняется. После успешной отладки пользовательской задачи рекомендуется установить минимальные параметры логирования.

Если контроллер устанавливается в новый проект, то весьма желательно собрать испытательный стенд, в котором есть контроллер той же модульной структуры, что и на реальном объекте, возможности физического моделирования значений сигналов, а также развернуты необходимые каналы промышленной коммуникации. Такой стенд дает возможность проверить и конфигурацию контроллера, и разработанную пользовательскую задачу без риска повреждения объекта, и нанесения вреда заказчику.

Если контроллер уже установлен на объекте, его конфигурация и программное обеспечение прошли опытную эксплуатацию, но требуется внесение изменений, то рекомендуется действовать по следующему алгоритму:

1. Сохранить текущую конфигурацию контроллера на энергонезависимом носителе информации, чтобы обеспечить возможность восстановления.

2. Перед прошивкой измененной конфигурации в контроллер отключить ВСЕ исполнительные механизмы.

3. Прошить новую конфигурацию и проверить прохождение всех команд телеуправления. Только в случае успешной проверки подключить исполнительные механизмы.

4. В течении трех суток обеспечить круглосуточное дежурство инженера АСУ.

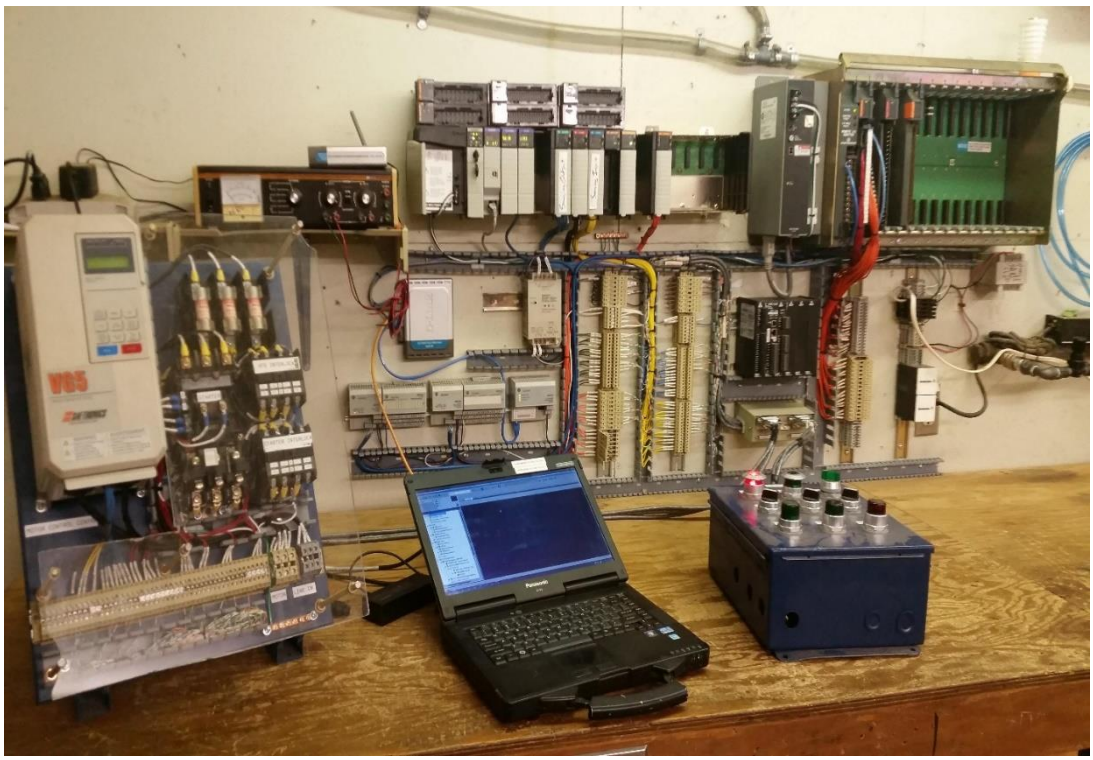


Рисунок 8.9 – Испытательный стенд

Документирование проекта является важной частью отладки и технической поддержки системы. Обслуживающий персоналд должен вести журнал изменения конфигурации контроллера. После успешного внесения изменений, копия работающей конфигурации должна сохраняться на энергонезависимом носителей с регистрацией даты, и номера изменений. Также по проекту автоматизации рекомендуется вести документ «Информационное обеспечение», в котором указаны все сигналы, источники их данных, номер обрабатывающего контроллера и номер модуля, ответственного за получение значения, адрес сигнала в адресном пространстве модуля, алгоритмы обработки, адреса передачи по каналам связи. Ведение такого документа в электронном или бумажном виде сильно поможет поддерживать проект в работоспособном состоянии, восстанавливать систему в случае сбоев.

Раздел 9. Коммуникационные возможности промышленных контроллеров

Отличие промышленных информационных сетей от сетей офисных, главным образом, состоит в предъявляемых требованиях:

Отказоустойчивость. Последствия потери на несколько минут соединения между компьютерами в офисе может привести разве что к недовольству руководства и лишению премии системного администратора. На производственных предприятиях кратковременный разрыв связи может стать причиной огромных убытков или возникновения опасности для жизни людей. Усугубляет ситуацию и то, что промышленные сети зачастую прокладываются и работают в сложных технологических условиях, где оборудование подвергается воздействию повышенных температур, высокой влажности, регулярным вибрационным и значительным механическим нагрузкам

Гарантия доставки данных. обеспечивается подтверждением получения сообщений и повторная отправка сообщений для которых не пришло подтверждение. Также передающая станция должна обеспечивать контроль наличия получающей в сети, и осуществлять сохранение неотправленных данных в буфере. Гарантия сохранения порядка следования сообщений реализуется с помощью нумерации байтов в сообщениях.

Жесткое реальное время. Система должна обеспечивать обмен данными в рамках требуемых временных ограничений. Процессы (задачи) реального времени могут иметь следующие характеристики и связанные с ними ограничения[3]: дедлайн (англ. deadline) — критический срок обслуживания, предельный срок завершения какой-либо работы; латентность (англ. latency) — время отклика (время задержки) системы на внешние события; джиттер (англ. jitter) — разброс значений времени отклика. Можно различить джиттер запуска (англ. release jitter) — период времени от готовности к исполнению до начала собственно исполнения задачи и джиттер вывода (англ. output jitter) — задержка по окончании выполнения задачи. Джиттер может возникать под влиянием других одновременно исполняемых задач.

Гибкая настройка для подключения устройств различных производителей, которые могут отличаться даже по скорости передачи данных, не говоря уже о разнице в заполнении адресного пространства.

Предъявляются также дополнительные требования:

- Экономия затрат при монтаже кабельных соединений.
- Минимизация обслуживания оборудования.
- Эффективная диагностика неисправностей.
- Дистанционная диагностика состояний станций и исполнительных устройств.
- Конфигурирование датчиков/исполнительных элементов как станций сети со своими адресами.
- Избыточность, обеспечивающая модификацию системы, подключение новых устройств, передачу дополнительной информации.

Общепринятая концепция промышленной коммуникации представлена на схеме. Вся промышленная сеть делится на 4 уровня (рис. 9.1). Нижний уровень сенсоров и актуаторов характерен малыми объемами информации у каждой станции, но высокими временными требованиями. По мере подъема к другим уровням системы в следствие концентрации данных наблюдается увеличение трафика, отчасти компенсируемое повышением вычислительных мощностей станций сети и снижением допустимой времени реакции на событие. Очевидно, промышленные контроллеры участвуют на втором и третьем уровне иерархии системы.

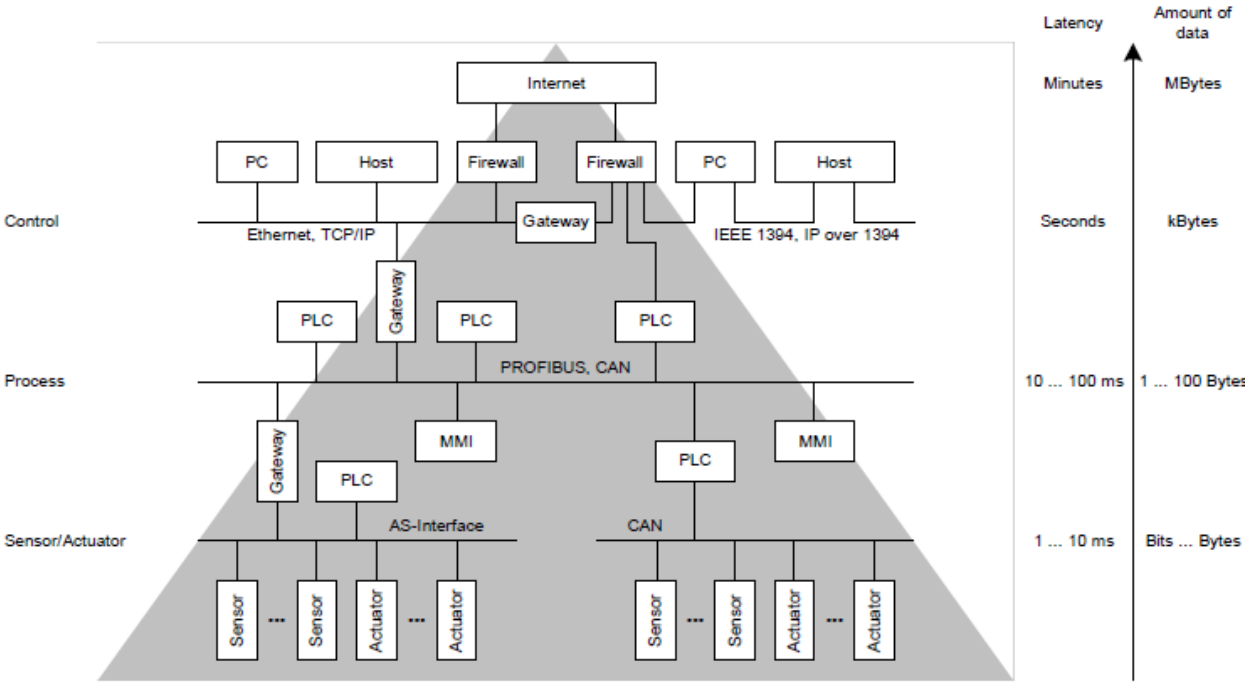


Рисунок 9.1 – Концепция промышленной коммуникации

Сети, обеспечивающие информационные обмен между контроллерами, датчиками и исполнительными устройствами, можно разделить на два уровня:

-управляющие промышленные сети, решающие задачи сбора и обработки данных на уровне промышленных контроллеров, управления технологическим процессом;

-полевые сети или шины, задачи которых сводятся к опросу датчиков и управлению работой разнообразных исполнительных устройств.

Для обеспечения безошибочности и максимального удобства передачи информации сетевые операции регулируются набором правил и соглашений, называемых сетевым протоколом. Сетевой протокол определяет типы разъемов, кабелей, сигналы, форматы данных и способы проверки ошибок, а также алгоритмы для сетевых интерфейсов и узлов, предполагая стандартными в пределах сети принципы подготовки сообщений и их передачи.

Реализация требований гарантии доставки данных и реального времени в промышленных сетях достигается во многом в результате применения суровой дисциплины, жесткого управления доступом каждой станции к каналам передачи данных. Возможные варианты разграничения доступа представлены на схеме. В современных промышленных сетях используется в основном временное мультиплексирование с контролируемым доступом. Наиболее распространен метод доступа, регулируемый главной станцией – мастером. Таковы популярные телемеханические протоколы ModBus и IEC 60870-5. Также встречается вариант передачи функций управления от станции к станции согласно логической или физической организации сети, например ModBus+, Profibus. Иногда используются более экзотичные варианты, например, как в CAN очередность устанавливается по номеру пакета.

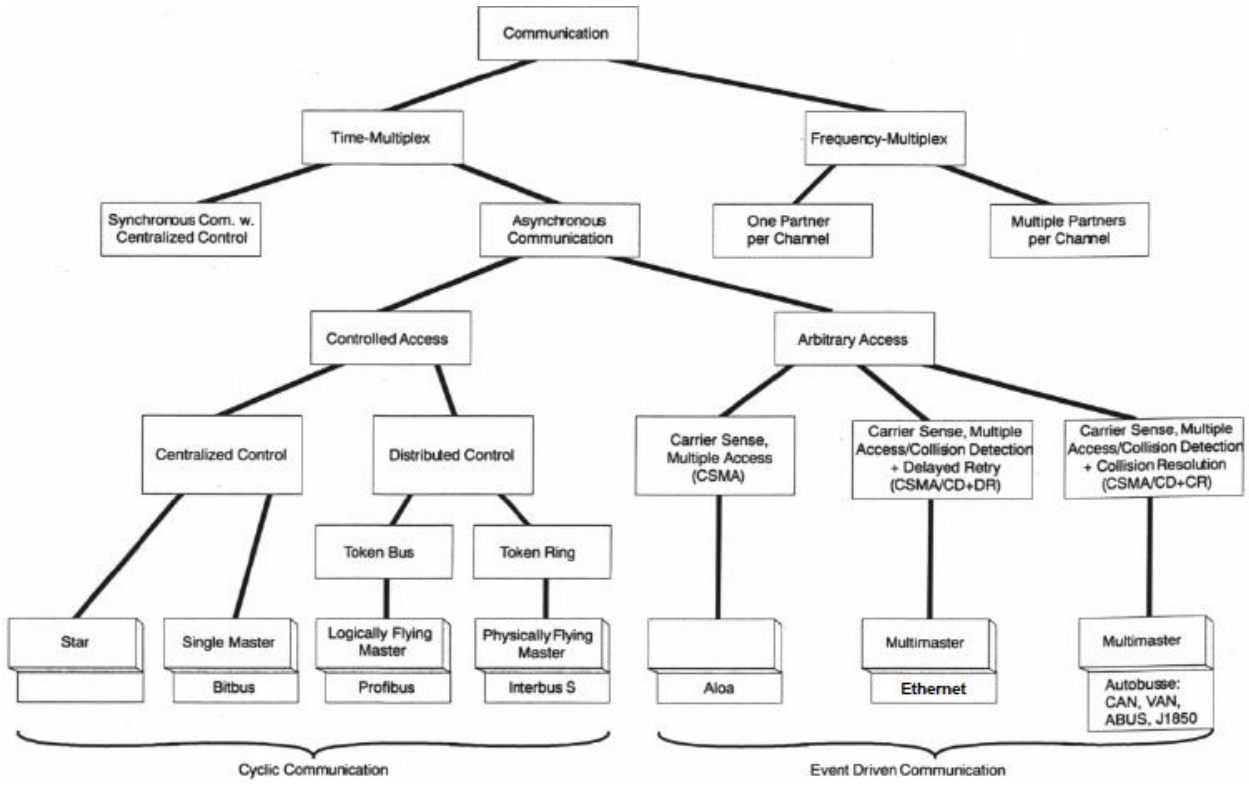


Рисунок 9.2 – Метод доступа станций к промышленным сетям

Модель OSI. Концептуальная модель, которая обобщает и стандартизирует представление средств сетевого взаимодействия в телекоммуникационных и компьютерных системах, независимо от их внутреннего устройства и используемых технологий. Модель OSI была разработана в 1984 году Международной организацией стандартизации (ISO). Основной целью ее создания был поиск решения проблемы несовместимости устройств, использующих различные коммуникационные протоколы, путем перехода на единый, общий для всех систем стек протоколов. Данная модель подразделяет коммуникационную систему на уровни абстракции как представлено на рисунке. Протоколы связи решают две задачи: они обеспечивают взаимодействие между сущностями, находящимися на одном и том же уровне абстракции, но на разных станциях и абстрактно описывают функционал, который (N-1)-ый уровень предоставляет (N)-ому, где N - один из 7 уровней модели OSI. В рамках модели, любой протокол может взаимодействовать либо с протоколами своего уровня (горизонтальные взаимодействия), либо с протоколами уровня на единицу выше/ниже своего уровня (вертикальные взаимодействия). Согласно модели, проблемы взаимодействия следует решать именно на том уровне, на котором они возникают. Инкапсуляция – метод проектирования протоколов в которой логически независимые функции сети не зависят от реализации нижележащих механизмов с помощью включения этих механизмов в более высокоуровневые объекты.

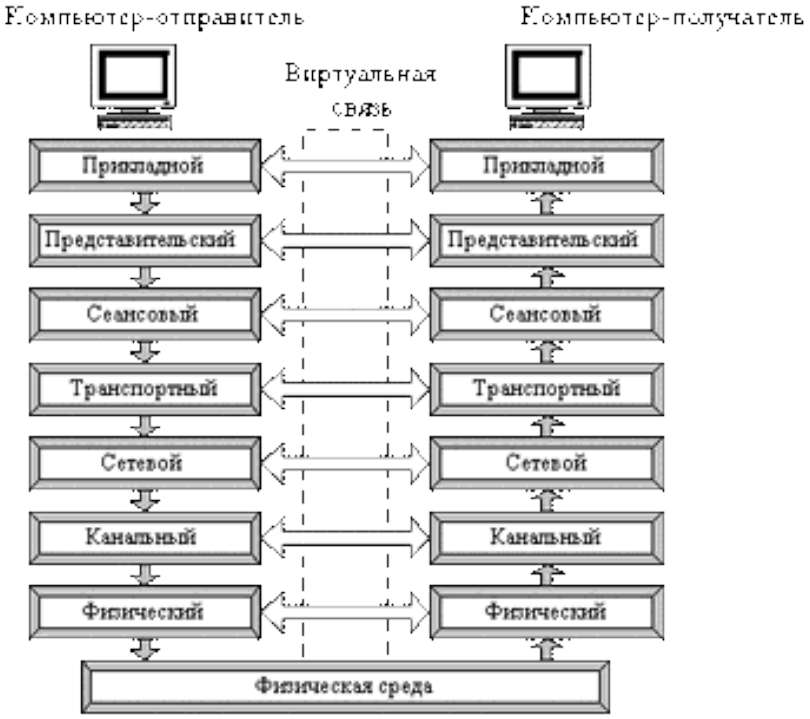


Рисунок 9.3 – Модель OSI

Протокол ModBus – самый распространенный промышленный протокол. Поддерживается почти всеми производителями промышленных контроллеров. Был представлен в 1979 году компанией Modicon (ныне Schneider Electric). Изначально это был открытый стандарт, работающий по интерфейсу RS-232. Позже появилась реализации протокола для интерфейсов RS-485 и Modbus TCP. Сейчас наибольшее применение имеет Modbus TCP.

Протокол не предусматривает аутентификацию и шифрование передаваемых данных. Поэтому, при использовании Modbus TCP необходимо использовать дополнительные VPN-тоннели. Используется доступ к каналу, управляемый мастером. Slave-устройство не может инициировать передачу данных, поэтому master должен постоянно опрашивать ведомые устройства. Максимальное количество станций в сети – 247. Адрес 0 используется мастером для широковещательных пакетов. Каждая подчиненная станция имеет в адресном пространстве четыре сегмента данных:

Output Coils — дискретные входы устройства, доступны только для чтения. Имеют функцию «02» — чтение группы ячеек.

Holding Coils — дискретные выходы устройства, или внутренние значения. Доступны для чтения и записи. Име.т функции: «01» — чтения группы ячеек, «05» — запись одной ячейки, «15» — запись группы ячеек.

Output Registers — аналоговые 16-битные входы устройства. Доступны только для чтения. Имеют функцию: «04» — чтение группы регистров

Holding Registers — аналоговые16-битные выходы устройства, либо внутренние значения. Доступны для чтения и записи.

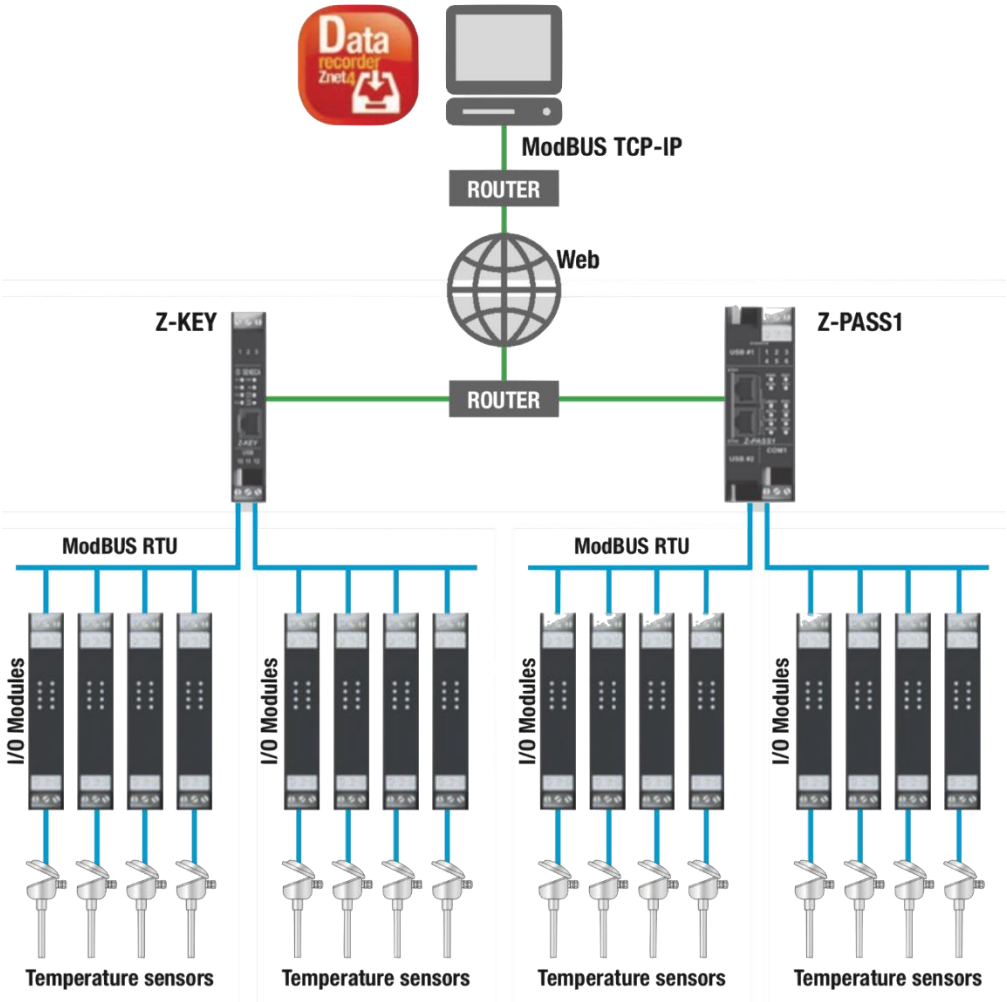


Рисунок 9.4 – Возможная конфигурация промышленной сети Modbus

В каждом сегменте 65536 элементов, т.е. используется двухбайтовая адресация. Протокол обладает хорошей информационной емкостью и простой логикой работы.

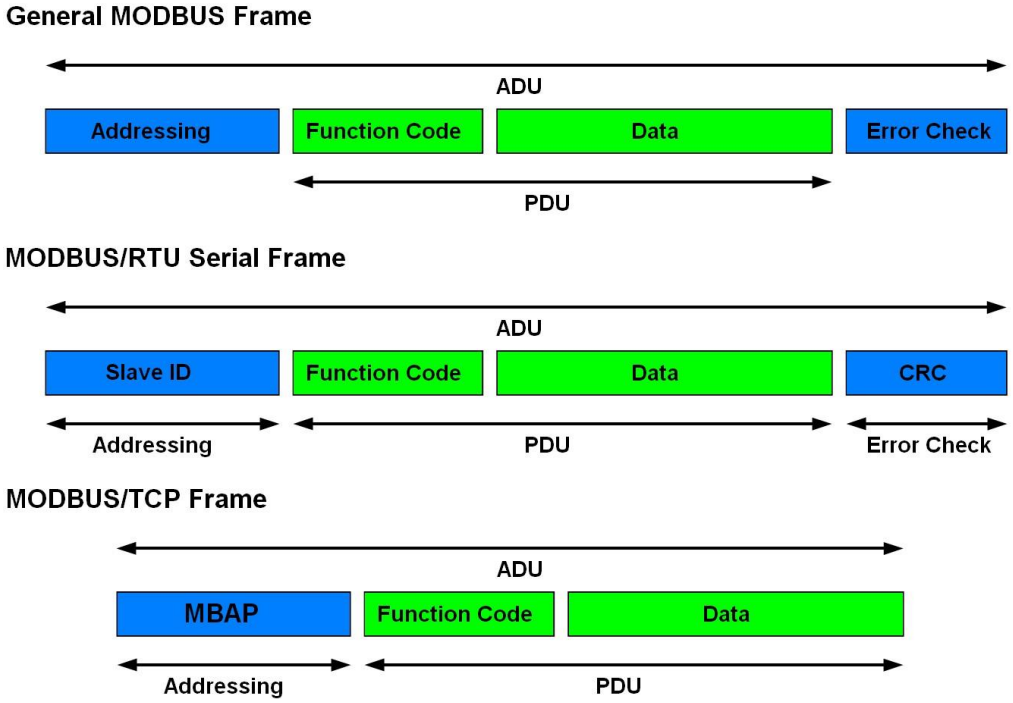


Рисунок 9.5 – Структура кадров сети Modbus

Протокол CAN – это промышленный стандарт, позволяющий осуществить объединение в единую сеть различных узлов, механизмов, датчиков и т. п. Протокол является широковещательным, это значит, что все устройства в CAN-сети принимают все передаваемые по шине сигналы. Режим передачи данных – последовательный, при этом байты сообщений формируют кадры определенного вида.

Физический уровень передачи данных представляет собой соединение посредством двухпроводной дифференциальной линии (витой пары). Важным условием работоспособности шины является наличие на концах витой пары согласующих резисторов, которые также называют терминаторами, с сопротивлением 120 Ом. В отличие от многих других протоколов в CAN не рекомендуется описание битов данных как “логического нуля” и “логической единицы”. Здесь используются понятия **доминантный** и **рецессивный** бит. Важнейшим свойством является то, что если один из узлов сети хочет выставить на линии рецессивный бит, а другой доминантный, то в итоге на линии окажется доминантный бит.

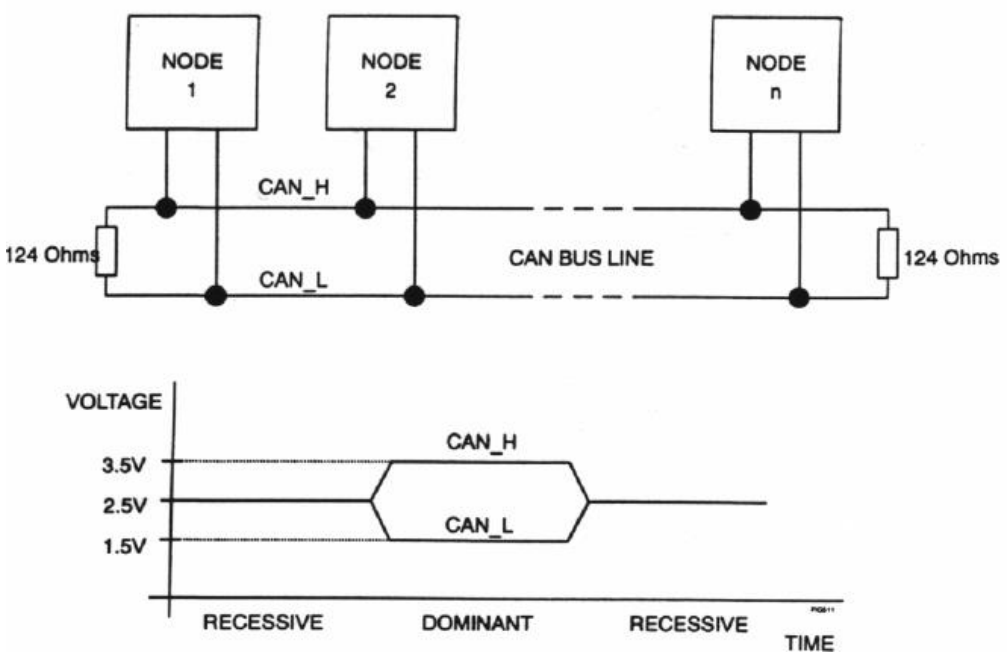


Рисунок 9.6 – физический уровень протокола CAN

Для управления очередностью передачи используется понятие арбитража. Пусть несколько устройств хотят передать данные. Каждый из этих передатчиков сравнивает значение, которое он передает, со значением, фактически присутствующим на линии. В том случае, если передаваемое значение совпадает со считанным, устройство продолжает высылать свои данные. Если значения совпали у нескольких устройств, то все они продолжают передачу как ни в чем не бывало. Продолжается это до того момента, когда значения станут различными. Если несколько устройств хотят передать рецессивный бит, а одно – доминантный, то в соответствии с правилом, которое мы обсудили выше, на линии окажется доминантный бит. В таком случае отправленные и считанные значения для устройств, пытающихся выдать на линию рецессивное состояние, не совпадут. В этом случае они должны прекратить передачу. А тот узел, который в этот момент передавал доминантный бит, продолжит свою работу.

Структура кадра представлена на схеме. 2 варианта: базовый и расширенный формат.

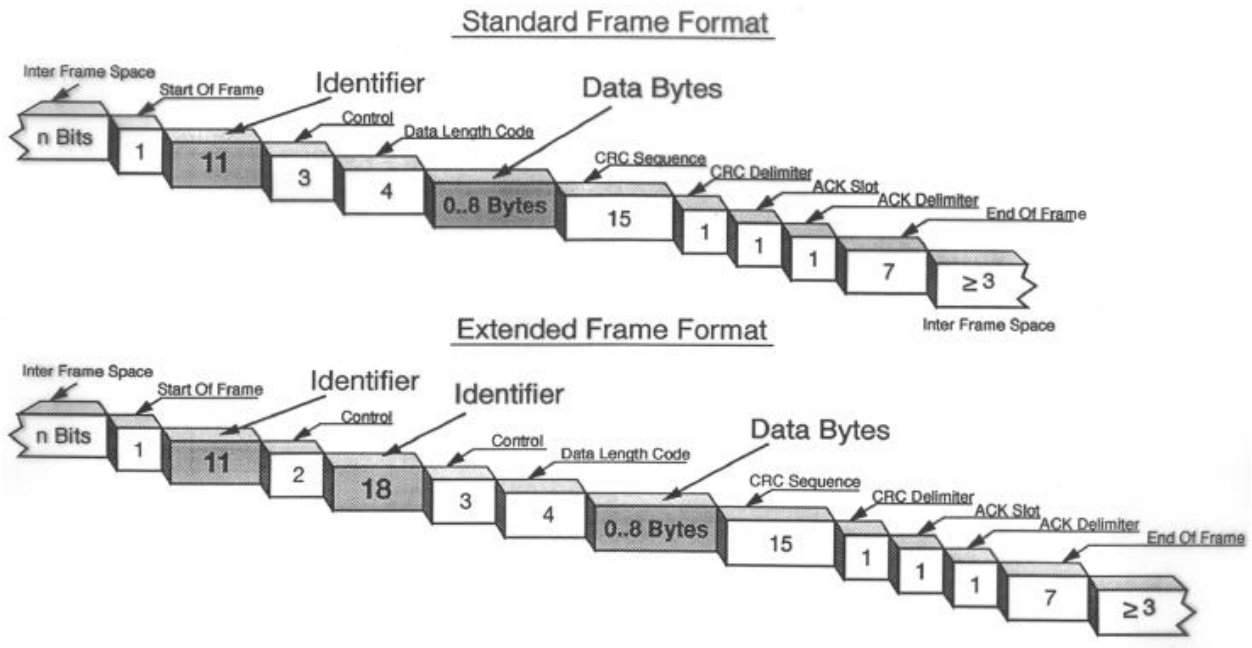


Рисунок 9.7 – Структура кадра протокола CAN

Телемеханический протокол IEC 60870-5 является событийным протоколом. В отличие от циклического обмена по рассмотренному выше протоколу ModBus запрос от мастера идет только к тем участникам сети, у которых есть данные I класса. Таким образом, происходит сильная экономия сетевого трафика. Сетевые устройства не накапливают ошибки при некачественном соединении. С учетом того, что доставка события происходит с меткой времени, даже если есть некоторая задержка, мастер шины получает информацию о произошедших событиях на удаленных объектах.

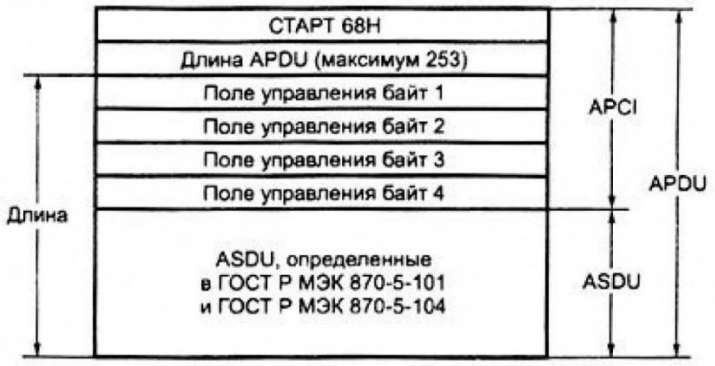


Рисунок 9.8 – Структура пакета данных

- APCI - Управляющая Информация Прикладного Уровня;
- ASDU - Блок Данных. Обслуживаемый Прикладным Уровнем (Блок данных Прикладного Уровня);
- APDU - Протокольный Блок Данных Прикладного Уровня.
- СТАРТ 68 Н определяет точку начала внутри потока данных.

Интерфейс транспортного уровня (интерфейс между пользователем и TCP) — это ориентированный на поток интерфейс, в котором не определяются какие-либо старт-стопные механизмы для ASDU (IEC 60870-5-101). Чтобы определить начало и конец ASDU, каждый заголовок APCI включает следующие маркировочные элементы: стартовый символ, указание длины ASDU вместе с полем управления. Может быть передан либо полный APDU, либо (для целей управления) только поля APCI.

К параметрам контроллеров, характеризующим их способность взаимодействовать с другими устройствами системы управления, относятся:

- количество и разнообразие портов в процессорных модулях;

- широта набора интерфейсных модулей и интерфейсных процессоров;
- поддерживаемые протоколы;
- скорость обмена данными и протяженность каналов связи.

Понятие **коммуникационного контроллера**. При построении многоуровневых систем автоматизации часто возникают задачи организации информационного обмена между уровнями. В одном случае необходим обмен комплексными сообщениями на средних скоростях. В другом - быстрый обмен короткими сообщениями с использованием упрощенного протокола обмена.

В последние годы проявилась тенденция применения в системах управления технологий сквозного сетевого доступа: от мощных промышленных серверов и многофункциональных контроллеров до интеллектуальных полевых устройств (датчики, исполнительные устройства и т. п.). При этом такая связь должна удовлетворять всем современным требованиям по функциональности, надежности и открытости. Рассмотренные выше протоколы не предназначены для непосредственного взаимодействия с устройствами полевого уровня.

Полевые шины (шины уровня датчиков и исполнительных устройств, Actuator Sensor Interface (ASI), Smart Distributed System (SDS)) должны удовлетворять двум требованиям. Во-первых, необходимо передавать данные в соответствии с жестким временным регламентом. Во-вторых, объем данных должен быть минимальным, чтобы обеспечить работоспособность сети в критические по нагрузкам моменты. Сеть уровня датчиков обеспечивает непосредственный интерфейс между реальным технологическим процессом и промышленными контроллерами.

Передаваемую в такой сети информацию можно разделить на два основных типа: данные о процессе и параметрические данные. Оба типа данных принципиально различны и предъявляют к коммуникационной системе разные требования.

Данные о процессе (изменение состояния кранов, переключателей, управляющих сигналов и т. п.) не являются сложными и, как правило, определяются несколькими информационными битами. Объем такой информации имеет четкую тенденцию к сокращению. Совсем недавно эти данные для одного простого устройства занимали 8-16 бит. Но уже сейчас развитие технологии привело к тому, что с простейших датчиков (дискретного типа) приходит всего 1-2 бита информации.

Данные о процессе имеют явно выраженный циклический характер. Более того, для реализации задач автоматического управления необходимо, чтобы опрос каналов и выдача команд на управление проводились через регламентируемые интервалы времени. Это так называемое требование детерминированности коммуникационной системы. Благодаря небольшому объему передаваемых данных системы промышленной связи способны действительно удовлетворять временным требованиям со стороны реальных процессов.

Параметрические данные необходимы как для отображения текущего состояния сетевых устройств (интеллектуальных), так и их перепрограммирования. В противоположность данным о процессе параметрическая информация не имеет циклического характера. Доступ к ней реализуется по запросу, в ациклическом режиме. Передача параметрических данных требует и реализует методы специальной защиты, а также механизмов подтверждений. Комплексный параметрический блок для интеллектуальных устройств занимает от нескольких десятков байт до нескольких сотен килобайт. В сравнении с быстро меняющимися данными временные требования для передачи параметров можно считать некритичными. В зависимости от типа устройств и протяженности сети требования по времени простираются от нескольких сотен миллисекунд до нескольких минут.

Эстафетная передача данных – это возможность перехода из одного канала к другому без полной распаковки данных. Т.е. коммуникационный контроллер в этом случае играет роль роутера из одной подсети в другую. Причины эстафетной передачи –

ограничения по дальности связи по выбранному каналу, ограничения по числу станций в одной сети, концентрация и интеграция данных при передаче от полевого уровня на диспетчерский пункт.

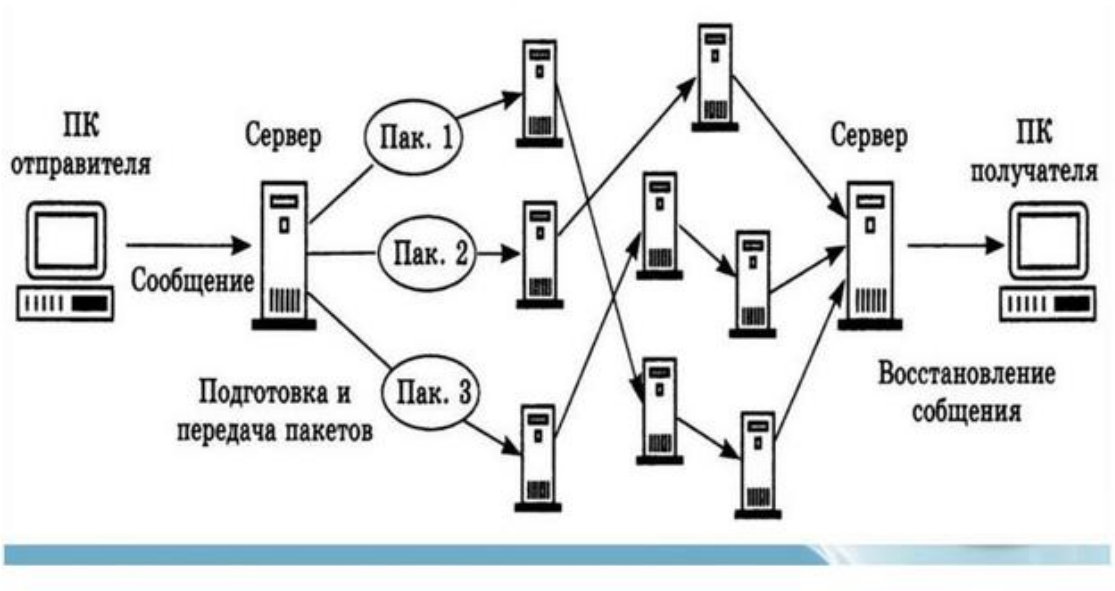


Рисунок 9.9 – Эстафетная передача данных

Коммуникационные возможности современных промышленных контроллеров позволяют использовать их для выполнения всего спектра посреднических операций в промышленных сетях – прием и отправка данных, шифрование и дешифрование, согласование6 электрических интерфейсов и преобразование протоколов. Настройки контроллеров позволяют скорректировать и организовать работу промышленной сети на любом уровне модели OSI.

Оценочные средства по разделам 7-9

Вопросы для самопроверки:

1. Какие проверки следует провести перед выполнением элементарной операции перекладки значений?
2. Как обеспечить приоритетное выполнение алгоритмов ПАЗ?
3. Какой стандартный функциональный блок позволяет эффективно решать проблему дребезга датчиков?
4. Каким способом можно усреднить значение аналогового сигнала по времени?
5. Как устранить множественное изменение состояния объекта при девиации значения ключевого параметра вблизи граничного уровня?
6. Какими способами можно контролировать состояния сигналов ТС в контроллере?
7. Как провести обновление алгоритмов действующей пользовательской задачи?
8. Перечислите параметры промышленных сетей, важные при проектировании?
9. Как устанавливается очередность передачи пакетов в сети CAN?
10. Какой адресный сегмент станции сети ModBus лучше подходит для обмена сигналами ТИ?

Тестовые вопросы:

1. Какие данные могут быть помещены в ячейку только для чтения протокола ModBus?

а) ТИ;
б) ТС;
в) ТУ;
г) ТР.

2. Проверка синтаксиса:

а) гарантирует работоспособность программы;
б) определяет соответствие кода программы правилам языка программирования;
в) указывает на логические ошибки в программе;
г) определяет соответствие программы конфигурации ПЛК.

3. Симулятор ПЛК используется для:

а) обучения программированию;
б) отладки программ и алгоритмов;
в) замещения ПЛК компьютером;
г) проверки синтаксиса программы

4. Сколько раз будет обработано нажатие пользователем кнопки, подключенной к модулю дискретного ввода, в течении одной секунды, если цикл контроллера составляет 250 мс?

а) 1 раз;
б) 4 раза;
в) 40 раз;
г) 25 раз.

5. Как осуществляется диагностика промышленного контроллера, не имеющего пользовательского интерфейса?

а) мобильными инструментами диагностики;
б) дистанционно;
в) в симуляторе;
г) не осуществляется.

6. Сколько уровней в модели интерфейса OSI?

а) 4;
б) 5;
в) 6;

г) 7.

7. Какой обмен данными называется асинхронным?

а) в котором транзакция завершается по строб-сигналу;

б) в котором транзакция завершается по времени;

в) в котором транзакция завершается "рукопожатием";

г) в котором не задана скорость передачи информации.

8. Какими преимуществами обладают беспроводные сети перед проводными?

а) выше степень защиты от НСД;

б) меньше затрат на монтаж;

в) выше скорость передачи данных;

г) возможность установки на мобильные объекты.

9. Как обеспечить приоритетное выполнение алгоритмов ПАЗ?

Открытый вопрос – дайте развернутый письменный ответ.

10. Что такое «дребезг датчиков» и как бороться с этим явлением?

Открытый вопрос – дайте развернутый письменный ответ.

Индивидуальное задание – разработка пользовательского функционального блока:

Уровень сложности – легкий. Разработать функциональный блок, реализующий подсчет импульсов при съеме показаний со счетчика. Индивидуальные параметры: язык программирования, модель счетчика.

Уровень сложности – средний. Разработать функциональный блок, реализующий учет наработки оборудования. Использовать функции работы со временем. Индивидуальные параметры: язык программирования, физические адреса сигналов для индикации работы оборудования.

Уровень сложности – сложный. Разработать функциональный блок, осуществляющий Запуск программы регулирования по расписанию. Индивидуальные параметры: язык программирования, расписание, характеристика объекта управления.

Список рекомендованной литературы

1. Гофман, П. М. Инструменты программирования промышленных контроллеров. CoDeSys : учебное пособие / П. М. Гофман, П. А. Кузнецов. — Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2019. — 94 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/147515 (дата обращения: 22.09.2022).
2. Мятеж, С. В. Промышленные контроллеры : учебное пособие / С. В. Мятеж. — Новосибирск : НГТУ, 2016. — 160 с. — ISBN 978-5-7782-3097-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/118135 (дата обращения: 22.09.2022).
3. Смирнов, Ю. А. Технические средства автоматизации и управления : учебное пособие / Ю. А. Смирнов. — 3-е изд., стер. — Санкт-Петербург : Лань, 2020. — 456 с. — ISBN 978-5-8114-5413-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/140779 (дата обращения: 22.09.2022).