НАДЁЖНОСТЬ ПО, АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ТЕХНОЛОГИЧЕСКИЕ ПРОЦЕССЫ

**Надёжность ПО** – свойство ПО своевременно выполнять заранее в указанных условиях к эксплуатации установленные функции. Надёжность устанавливается по результатам работы ПО, т.е. при динамической проверке всех программ на множестве входной информации.

Некорректное ПО заведомо не надёжно, однако и корректное ПО может быть не надёжно.

Определение надёжности ПО базируется на понятии отказа программ. Отказы ПО делятся на случайные и неслучайные.

* **Неслучайные отказы обусловлены действием так называемых компьютерных вирусов.**
* **Случайные отказы наблюдаются в случайные моменты времени работы ЭВМ.**

По свои последствиям эти отказы классифицируются на случайные сбои программ и устойчивые отказы ПО.

Сбой ПО, возникающий при некоторых возможно случайных состояниях ЭВМ и информации, наблюдаемые пользователем в случайные моменты времени и исчезающие без вмешательства программиста.

Устойчивый отказ ПО наблюдается в случайный момент процессорного времени в форме результата в рамках нормального функционирования ЭВМ. Причиной отказа ПО служат некоторая систематическая ошибка, после устранения которой программистом данный отказ исчезает, т.е. имеет место восстановления ПО.

Различают ошибки первичного и вторичного типа.

**Ошибки первичного** типа связаны с неточностями текста программ и возникают при подготовке носителей и документации ПО, при записях на алгоритмических языках и трансляции программ на машинный язык.

А также из-за неточностей алгоритмов и при неверных или некорректных постановках решаемых на ЭВМ вычислительных задач. Ошибки вторичного типа во многом являются следствием первичных ошибок программ. К ним относят ошибки: вычислительные (неверная индексация и подсчет временных параметров, расхождение результата ручного и машинного счета, появление неустойчивых операций и т.п.); логические (пропуск логических условий, неверные краевые условия и др.); сопряжения интерфейсов (межмодульных, программно-технических, информационных). Ошибки первичного и вторичного типов порождаются на этапах разработки спецификаций на ПО; проектирования ПО; реализации программ. **Отказы ПО при его эксплуатации имеют ряд отличий от отказов технических элементов: Отказ ПО не приводит к разрушению или поломке программного элемента. Отказы ПО не связаны с физическим износом элемента (в частности носителя программ). Отказ ПО не коррелирован с процессорным и, тем более, астрономическим временем (с процессорным временем или числом прогонов ПО программ пользователем).** При длительной эксплуатации ПО все его ошибки могут быть устранены и программы становятся абсолютно надежными. Если обозначить через N(t) число не выявленных ошибок ПО в произвольный момент 1=процессорного времени t, то формально имеет место соотношение lim N(t) =О, справедливое при условии, что в процессе восстановления программ в них не вносятся новые ошибки.

Опыт создания и эксплуатации ПО реального времени показывает, что при устранении одних ошибок вносятся другие. Поэтому при длительной эксплуатации ПО, общее число ошибок может оставаться постоянным или даже возрастать. Для описания надежности ПО используют такие же функциональные и числовые характеристики, как и при исследовании надежности технических элементов. Основные показатели надежности ПО:

1. **функция ненадежности или отказа ПО Q(t) = Вер {того, что отказ ПО появится до момента времени t};**

**2. функция надежности ПО P(t) = Вер {того, что отказ ПО появится после момента времени t);**

**3. интенсивность отказов ПО (лямбда)(t) = dQ/dt:**

**4. средняя наработка на отказ ПО: t = |t f(t)dt = |P(t)dt** 

Программное обеспечение АСУТП состоит из большого числа программ, подпрограмм и модулей, находящихся под управлением операционной системы реального времени или программы-диспетчера. Выполнение каждой из этих программ осуществляется последовательно во времени на одном и том же процессоре. Если эти программы имеют взаимные информационные связи или предназначены для получения одного результата у (вычисление одной функции), то в надежностном отношении такой программный комплекс представляет собой простую систему без избыточности и вероятность его безотказной работы равна произведению вероятностей безотказной работы каждой і-ой программы: pо-Про)

**где m - общее число программ.**

**Надежность такого ПО** определяется надежностью отказов самой «ненадежной» программы, имеющей наибольшее значение**, i=1.m.**

Для повышения надежности нерезервированного ПО следует в первую очередь улучшить характеристики самых «ненадежных» программ. (более жесткое динамическое тестирование «ненадежных» программ, расширяя при этом набор тестовых задач). Если тестирование не уменьшает интенсивность проявления ошибок, то переписывают «**ненадежную**» программу, стремясь усилить ее структурированность путем увеличения числа готовых и хорошо изученных программных модулей и стандартных подпрограмм и применения апробированных межмодульных интерфейсов. Понижению интенсивности способствует и переход на другой более высокий язык программирования.

Другой путь повышения надежности ПО связано с резервированием и введением в программную систему некоторой избыточности. Применительно к ПО АСУТП различают три вида резервирования: **1. временное; 2. информационное; 3. программное.**

Временное резервирование ПО заключается в многократном прогоне одних и тех же «ненадежных» программ и сравнении результатов расчета. Такое нагруженное резервирование позволяет устранять влияние случайных сбоев и выявлять случайные ошибки, требующие восстановления программ. Информационное резервирование ПО основано на дублированных исходных и промежуточных данных. Эти данные могут проходить дополни тельную обработку, например, усреднение, до ввода в ПО, где они обрабатываются один раз или обрабатываться одной и той же программой дважды, т.е. информационное резервирование подкрепляется временным. Программное резервирование предусматривает наличие в ПО двух или больше разных программ для получения одного и того же результата у или реализации одной функции. Здесь возможно нагруженное и ненагруженное резервирование. Резервирование программного обеспечения распределенных АСУТП часто сопровождается аппаратурным резервированием. При отказе ПО какой-либо локальной технологической станции или при выходе из строя используете в проекте предоставляемые с документацией һ- и с-файлы. В этом случае говорят об Custom-интерфейсе (не путать с COM-интерфейсами!). Либо вы используете для скриптовых запросов так называемую автоматизацию (OLE Automation). В этом случае для доступа к функциям объекта используется специальный CОМ-интерфейс IDispatch, который CОМ-объект в этом случае обязан поддерживать, предоставляя интерфейс Автоматизации (опять не путать с СОM- интерфейсамн!). При этом никакие компилируемые файлы не нужны. Но нужна так называемая библиотека типов.

**Реализация объектов**

Программирование COM - занятие не из лёгких. С помощью С-подобного языка MIDL (Microsoft Interface Definition Language - язык определения интерфейсов) описываются интерфейсы. С помощью компилятора MIDL.EXE они преобразовываются в описанные выше файлы, в том числе и в библиотеку типов. А далее используется библиотека ATL (Active Template Library - библиотека активных шаблонов), "умеющая" интерпретировать эти файлы и многое другое, связанное с СОМ

**OPС в свете СOM**

**Историческая справка**

Сравнительно недавно, в 1994 г., под эгидой Microsoft, была создана организация ОРС Foundation ( http://www.opcfoundation.org ). Как определяет сама OPС Foundation, её целью является разработка и поддержка открытых промышленных стандартов, регламентирующих методы обмена данными в реальном времени между клиентами на базе РС и ОС Microsoft. Сейчас эта организация насчитывает более 220 членов, включая почти всех ведущих поставщиков контрольно-измерительного и управляющего оборудования для АСУ ТП. Достаточно назвать такие фирмы, как Siemens, Schneider Automation, Rockwell Software, Wonderware, Intellution, Ci Technologies, не говоря уже о самой Microsoft.

**Технология**

Как уже отмечалось выше, технология ОРС реализована и продолжает реализовываться по типу- разработка стандартов. ОРС Foundation определяет направления, по которым ведутся разработки, и создаёт по этим направлениям комитеты. Комитеты делают следующее: разрабатывают спецификации СОМ-интерфейсов и СОМ-объектов; присваивают им GUID; оформляют всё в виде стандартов и опубликовывают; генерируют или создают вспомогательные файлы: idl-, h- и с-файлы для Custom- интерфейса; библиотеки типов для интерфейса автоматизации; заместители (proxy) и заглушки (stub) для поддержки межпроцессного взаимодействия; разрабатывают вспомогательные компоненты, например, утилиту орсепит, позволяющую ОРС-клиенту "увидеть" список всех ОРС-серверов локальной сети;

**Спецификации**

В настоящее время имеются следующие ОРС-стандарты.

ОPС Common Definitions and Interfaces - общие для всех ОPС-спецификаций интерфейсы.

Data Access Custom Interface Standard - спецификация COM-интерфейсов для обмена оперативными данными, программирование на С++.

Data Access Automation Interface Standard - спецификация COM-интерфейсов для обмена оперативными данными, программирование на языках типа Visual Basic. OPC Batch Custom Interface Specification - спецификация COM-интерфейсов конфигурирования оборудования, программирование на С++. OPC Batch Automation Interface Specification - Спецификация COM-интерфейсов для конфигурирования оборудования, программирование на языках типа Visual Basic. ОPС Alarms and Events Interface Specification - спецификация COM-интерфейсов для обслуживания событий (event) и нештатных ситуаций (alarm), программирование на C++. Historical Data Access Custom Interface Standard - спецификация COM-интерфейсов для С++ работы с хранилищами данными, программирование на С++.

OPC Security Custom Interface - спецификация COM-интерфейсов для обработки прав доступа к данным, программирование на С++.

Как видим, перечень достаточно большой. Консорциум ОРС Foundation пытается охватить все аспекты, связанные с взаимодействием с технологическим оборудованием. В

разработке самих спецификаций принимают участие ведущие производители оборудования и систем автоматизации, которые стараются максимально учесть свой опыт

и предоставить абсолютно "всё необходимое" тому, кто будет использовать ОРС. Далее мы проиллюстрируем это на примере спецификации Data Access (DA).

**ОРС-сервер (потребители "снизу")**

Кто же использует ОРС? Первая категория - производители оборудования автоматизации, или OEM (Original Equipment Manufacturer - поставщик комплексного оборудования). Предполагается, что тот, кто создаёт, например, плату сбора данных, снабжает её не только драйвером, но и реализует ОРС-сервер, работающий с этой с платой через драйвер или даже напрямую. Тем самым ОЕМ-производитель предоставляет стандартный доступ к своей плате.

Продукт Каждая переменная обладает свойствами. Различаются обязательные свойства, рекомендуемые и пользовательские. Обязательными свойствами, понятно, обязана обладать каждая переменная. Это, во-первых, текущее значение переменной, тип переменной и права доступа (чтение и/или запись). Во-вторых, очень важные свойства - качество переменной и метка времени. Технология ОРС ориентирована на работу с оборудованием, а оборудование может давать сбои, так что корректное значение переменной не всегда известно ОРС-серверу, о чём и уведомляется клиент через качество

(хорошее/плохое/неопределённое и дополнительная информация).

Метка времени сообщает о том, когда переменная получила данное значение и/или качество. Ещё одним обязательным свойством является частота опроса переменной ОРС-сервером. Не совсем понятно, почему это свойство объявлено обязательным, так как не все ОРС-серверы работают в режиме опроса оборудования. Поэтому существуют серверы, не реализующие это свойство. Последним из обязательных свойств является описание переменной. Это строковое значение, содержащее информацию для пользователя о том. зачем нужна эта переменная.

Дополнительные свойства являются необязательными для реализации в ОРС-сервере. Это, например, диапазон изменения (выход за границы диапазона должен специальным образом обрабатываться клиентом) и единица измерения. Есть перечень рекомендуемых свойств, но можно добавить и свои собственные.

**Получение данных**

Существует три основных способа получения ОРС-клиентом данных от ОРС-сервера: синхронное чтение, асинхронное чтение и подписка. При синхронном чтении клиент посылает серверу запрос со списком интересующих его переменных и ждёт, когда сервер его выполнит. При асинхронном чтении клиент посылает серверу запрос, а сам продолжает работать. Когда сервер выполнил запрос, клиент получает уведомление (через интерфейс соответствующего СОМ-объекта, реализованного в клиенте!). И, наконец, в случае подписки клиент передаёт серверу список интересующих его переменных, а сервер затем регулярно присылает клиенту информацию об изменившихся переменных из этого списка (онять же, через интерфейс соответствующего СОМ-объекта клиента!). Эти списки в терминологии ОРС называются группами. Каждый клиент может поддерживать одновременно много групп с разной скоростью обновления.

**Запись данных**

Ничем не отличается от чтения, за исключением того, что нет записи по подписке.

**Источники данных**

регламентирует только интерфейс между ОРС-клиентами и ОРС-серверами (как и положено в технологии клиент-сервер, допускается множественные подсоединения). И она абсолютно не регламентирует способ получения этих данных от оборудования! Разработчик сам определяет, где и как их брать.

**Организация данных**

Что должен сделать производитель, если он задался целью обеспечить свой продукт стандартным интерфейсом? Он должен получить нужную спецификацию и прилагаемые программные компоненты. Затем он должен изучить СОМ-интерфейсы тех СОМ- объектов этой спецификации, которые относятся в ней к модели ОРС-сервера. И, наконец, он должен посадить самого опытного программиста за Visual Studio, и тот с помощью ATL-библиотеки реализует требуемые интерфейсы, а значит и OPC-сервер

**ОРС-клиент (потребители "сверху")**

Правила игры заданы - ОРС-сервер поставляет данные, ОРС-клиент потребляет. Этим задаётся вторая категория пользователей спецификаций ОРС. И к ней относятся в первую очередь те, кто реализует программное обеспечение более высокого уровня. Например, поставщик SCADA-пакета. Или чего-то близкого по назначению.

Что же должен сделать производитель "верхнего" ПО, если он задался целью обеспечить свой продукт стандартным интерфейсом? Он должен получить нужную спецификацию и прилагаемые программные компоненты. Затем он должен изучить СОМ- интерфейсы тех СОМ-объектов этой спецификации, которые относятся в ней к модели ОРС-клиента. И, наконец, он должен посадить достаточно опытного программиста за Visual Studio, и тот с помощью ATL-библиотеки реализует требуемые интерфейсы, а значит и ОРС-клиент для Custom-интерфейса. Можно использовать Visual Basic или, скажем, Delphi, и тогда будет создан ОРС-клиент для интерфейса Автоматизации (если таковая реализована для данной спецификации).

**OPC Data Access**

Чтобы лучше почувствовать, что такое ОРС, рассмотрим подробнее главный, по

большому счёту, стандарт. Будем называть его сокращённо DA.

Стандарт DA предназначен для поставки оперативных данных от оборудования и/или к оборудованию. Для стандарта DA реализованы спецификации как Custom-интерфейса, так интерфейса Автоматизации. С точки зрения функциональных интерфейсов, последний ничем не отличается от Custom, кроме того, что не позволяет одновременно работать с несколькими ОРС-серверами и добавлен упоминавшийся выше COМ-интерфейс IDispatch, обязательный в OLE Automation. Это позволило OPС Foundation издать "обёртку" (wrapper) в виде dll, преобразующую один интерфейс в другой.

**Данные**

Основной единицей данных в ОРС является переменная (Item). Переменная может быть любого типа, допустимого в OLE: различные целые и вещественные типы, логический тип, строковый, дата, валюта, вариантный тип и так далее. Кроме того, переменная может быть массивом.

**Свойства**

Переменные в ОРС-сервере могут быть упорядочены либо в простой список, либо в дерево, напоминающее дерево файлов на диске (только вместо термина "папка" в ОРС говорят "ветвь"). И есть соответствующие интерфейсы для навигации по этому дереву. Можно, в частности, в любой момент запросить дерево переменных, поддерживаемых ОРС-сервером. Если оборудование допускает, дерево может изменяться динамически…

Есть механизм оповещения завершения работы ОРС-сервера. Есть возможность запросить информацию о самом сервере. Есть возможность запросить список зарегистрированных групп. В общем, есть много того, что старались предусмотреть разработчики ОРС- спецификаций, чтобы облегчить организацию взаимодействия поставщика данных (ОРС- сервера) и потребителя данных (ОРС-клиента).

**Инструментарий**

Как уже было сказано, чтобы написать ОРС-сервер или ОРС-клиент, нужно только взаимодействие с ОРС Foundation (OPС-спецификации) и Microsoft (Visual C++ и пр.).

**Toolkit**

Трудности программирования ОРС-сервер или ОРС-клиент можно уменьшить , если воспользоваться так называемыми Toolkit'ами. и предлагают средства, позволяющие более-менее безопасно и легко создавать ОРС-продукцию.

Типичный Toolkit представляет собой библиотеку, реализующую ОРС-объекты выбранной спецификации, что реализует все прихоти со стороны ОРС. Разработчику же, например, ОРС-сервера предлагается некий набор вызовов, достаточно простых (read. write, …), которые необходимо "подцепить" к своему оборудованию для доступа к его данным. Для знаюших объектное программирование заметим, что эти функции могут быть реализованы как виртуальные функции некоторого класса, которые нужно перегрузить в своём приложении. Так сделаны, например, Toolkit'ы фирмы FactorySoft ( http://www.factoryfoft.com ).

**ОРС и интеграция**

Теперь настало время взглянуть на ОРС с точки зрения главной темы. На рис. 1 представлена схема, иллюстрирующая возможные применения ОРС-серверов в АСУ предприятия. Мы различаем несколько уровней управления:

* нижний уровень - полевые шины (fieldbus) и отдельные контроллеры;
* средний уровень - цеховые сети;
* высший уровень АСУ ТП - уровень работы систем типа SCADA;
* уровень АСУ П - уровень приложений управления ресурсами предприятия.

Каждый из этих уровней может обслуживаться ОРС-сервером, поставляя данные ОРС- клиенту на более высоком уровне, или даже "соседу". Замена устройства не потребует изменения остальных приложений: драйвер изменился, по ОРС-интерфейс поверх пего остался прежний,

**ОPС через сеть**

Имеется устройство, управляемое через какой-пибуль сетевой протокол. В этом случае вполне типична реализация ОРС-сервера, получающего данные по этому протоколу. Единственная особенность в этом случае - предусмотреть механизмы восстановления связи в случае сбоев.

**OPC для fieldbus**

Ещё одна разновидность ОРС-сервера - шлюз к сети полевой шины, такой как Profibus или

Lonworks. С точки зрения реализации это очень похоже на предыдущие случаи. Скорее всего, на компьютере с ОС Windows будет установлен адаптер fieldbus-сети, а ОРС-сервер будет работать с этой сетью через драйвер адаптера. В Internet можно найти немало таких примеров.

Сеть полевой шины работает в жёстком реальном времени, а ОРС предоставляет менее требовательный шлюз к этой сети из приложений более высокого уровня.

**И другие применения ОРС**

Можно придумать много других применений ОРС. Например, ОРС для работы с базами данных, вспомогательные ОРС-серверы, промежуточные и т.д. Возможности для фантазии неограниченны. Одну такую фантазию хотелось бы привести.

Технология DCOM, как уже говорилось, не работает в глобальных сетях. Поэтому для привлечения к ОРС-технологии Internet-технологий можно набросать такой путь.

Расширение web-сервера является ОРС-клиентом, собирающим данные от ОРС-серверов. А на стороне клиентов запускается динамическая html- или xml-страница, получающая данные от этого web-сервера. Её можно сделать даже ОРС-сервером для других приложений.

**Спецификации**

В настоящее время полностью завершённой являются только спецификация DA OPC.

Другие спецификации ещё дорабатываются, по крайней мере, с точки зрения интерфейса Автоматизации (для Batch ОРС доступна бета-версия интерфейса Автоматизации, для остальных спецификаций нет ещё даже бета-версий).

**Производители оборудования**

В настоящее время по-настоящему широкое распространение получил только стандарт DA ОРС. Можно сказать, что сейчас действительно очень многие производители снабжают свои продукты DA OPС-серверами, Чего нельзя сказать о других спецификациях. приложения Internet), а связанные с этим технологии объединяются под общим названием Component Services (Сервисы Компонентов).

**Объекты СОM**

Модель СОМ оперирует объектами, очень похожими на объекты в объектно- ориентированных языках программирования типа С++. Но сама технология СОМ не является языком программирования. Она только регламентирует поведение своих объектов. Нам нужно знать, что объект может быть создан, после чего он предоставляет свою функциональность вызвавшему процессу, а после использования - уничтожен.

**Интерфейсы объектов**

Объекты COM предоставляют свою функциональность через интерфейсы (Interface). Интерфейс в СОМ объединяет группу взаимосвязанных функций, предоставляемых объектом. Главная особенность интерфейсов СОМ заключается в их "публичности". Интерфейсы используются после того, как они "опубликованы", и после этого их нельзя изменять никогда . Если необходима новая версия интерфейса, издаётся новый интерфейс при сохранении старого. Этим обеспечивается совместимость при обновлении и модернизации объектов. И это первый шаг на пути к интеграции.

**Доступ к объектам**

Именно интерфейс, вернее указатель на него, является тем, с чем работает вызывающий процесс (читай программист). Объект может предоставлять несколько интерфейсов. Чтобы получить указатель на любой интерфейс, нужно воспользоваться функцией QueryInterface Указатель на этот интерфейс передаётся инициирующему процессу при создании объекта.

**Обмен в СОМ**

Объект СОМ является пассивной стороной. Он лишь предоставляет через интерфейсы свои функции. В этом смысле употребляется термин COМ-сервер. Запрашивающая программа, соответственно, называется СОМ-клиент. Но это не исключает того, что обе программы одновременно могут являться и СОМ-серверами, и CОМ-клиентами. ОРС-сервер может поставлять данные "по подписке", то есть сам инициализировать обмен с ОРС-клиентом при их обновлении,

**Регистрация**

Чтобы создать объект, нужно знать, где он находится, В Windows для этого используется регистрация объектов в системном реестре При этом каждый "СОМ-предмет" регистрации имеет уникальный в полном смысле этого слова идентификатор, называемый GUID (Globally Unique Identifier - глобально уникальный идентификатор). Присваивает идентификаторы своим "СОМ-детищам" их создатель, используя, например, программу GUIDGEN.EXE…

Регистрация делает доступной информацию о расположении объектов всем

приложениям…

**Обслуживание объектов**

Вопросы, затрагиваемые здесь, очень важны для понимания всего излагаемого.

**Объекты СОМ**

Объекты СОМ должны быть достаточно независимыми. Они зачастую, если не сказать в большинстве случаев, находятся вне программы CОМ-клиента, а могут быть запущены даже на другом компьютере. Это имеет принципиальные последствия. Даже на одном компьютере разные приложения Windows функционируют в своих собственных адресных пространствах. Это означает, что требуется кто-то, кто будет передавать вызовы из одного процесса в другой. Даже простое создание или уничтожение объекта в другом адресном пространстве вовсе не тривиальное дело. В СОМ эти и другие проблема решается с помощью специальных библиотек, таких как, OLE32.DLL. С одной стороны, эти библиотеки предоставляют функции для работы с объектами. Например, вызов CoCreateInstance создаёт объект. С другой стороны, активизируемые специальные компоненты выполняют диспетчерские функции, например, упаковку и передачу параметров вызываемым методам объектов (т.н. marshalling). В связи с этим упомянем два важных модуля: заместитель (proxy) и заглушка (stub). Они функционируют в адресном пространстве СОМ-клиента и СОМ-сервера соответственно и обеспечивают прозрачность вызовов. Механизм таков: COМ-клиент "непосредственно" вызывает функцию СОМ-интерфейса, которую ему "подсовывает" заместитель. Тот передаёт вызов заглушке через RPC (Remote Procedure Call - вызов удалённых процедур). А заглушка "непосредственно"

вызывает функцию COМ-сервера.

**Удалённые объекты**

Без сетевых решений разговора об интеграции в настоящее время можно даже и не начинать. В COM по этому поводу существует DCOM - расширение COM, позволяющее "добираться" до объектов на других компьютерах. Существенно то, что с точки зрения программирования, ничего не меняется: DCOM — это системный сервис, делающий СОМ прозрачным в локальных сетях. И это четвёртый шаг к интеграции. Но с тем же очевидным недостатком: DCOM должен присутствовать в операционной системе.

**Предоставление объектов**

Чтобы использовать объект, необходимо знать, как он устроен, вернее, как устроены его интерфейсы. Для этого они должны быть опубликованы. Например, в виде официальной документации. Или в виде стандарта.

**Использование объектов**

Использовать СОМ-объекты должны СОМ-клиенты. Но они могут быть разными, если мы говорим об интеграции. И могут использовать разные языки программирования, не исключая скриптовых типа Visual Basic. Технология COM здесь предусматривает две возможности. Либо вы программируете на С++ и тогда для описания интерфейсов используете в проекте предоставляемые с документацией һ- и с-файлы. В этом случае говорят об Custom-интерфейсе (не путать с COM-интерфейсами!). Либо вы используете для скриптовых запросов так называемую автоматизацию (OLE Automation). В этом случае для доступа к функциям объекта используется специальный CОМ-интерфейс IDispatch, который CОМ-объект в этом случае обязан поддерживать, предоставляя интерфейс Автоматизации (опять не путать с СОM- интерфейсамн!). При этом никакие компилируемые файлы не нужны. Но нужна так называемая библиотека типов.

**Реализация объектов**

Программирование COM - занятие не из лёгких. С помощью С-подобного языка MIDL (Microsoft Interface Definition Language - язык определения интерфейсов) описываются интерфейсы. С помощью компилятора MIDL.EXE они преобразовываются в описанные выше файлы, в том числе и в библиотеку типов. А далее используется библиотека ATL (Active Template Library - библиотека активных шаблонов), "умеющая" интерпретировать эти файлы и многое другое, связанное с СОМ

**OPС в свете СOM**

**Историческая справка**

Сравнительно недавно, в 1994 г., под эгидой Microsoft, была создана организация ОРС Foundation ( http://www.opcfoundation.org ). Как определяет сама OPС Foundation, её целью является разработка и поддержка открытых промышленных стандартов, регламентирующих методы обмена данными в реальном времени между клиентами на базе РС и ОС Microsoft. Сейчас эта организация насчитывает более 220 членов, включая почти всех ведущих поставщиков контрольно-измерительного и управляющего оборудования для АСУ ТП. Достаточно назвать такие фирмы, как Siemens, Schneider Automation, Rockwell Software, Wonderware, Intellution, Ci Technologies, не говоря уже о самой Microsoft.

**Технология**

Как уже отмечалось выше, технология ОРС реализована и продолжает реализовываться по типу- разработка стандартов. ОРС Foundation определяет направления, по которым ведутся разработки, и создаёт по этим направлениям комитеты. Комитеты делают следующее:

* разрабатывают спецификации СОМ-интерфейсов и СОМ-объектов;
* присваивают им GUID;
* оформляют всё в виде стандартов и опубликовывают;

Язык программирования Visual Basic 2010 входит в состав интегрированной среды разработки Microsoft Visual Studio 2010 , которая включает в себя, кроме Visual Basic 2010 следующие языки программирования высокого уровня: Microsoft Visual C++ 2010, Microsoft C# 2010, Microsoft F# 2010 и другой инструментарий. Язык программирования Visual Basic 2 010 позволяет создавать приложения с графическим интерфейсом, работающие в среде операционной системы Microsoft Windows, содержит средства для разработки приложений взаимодействующих с базами данных, приложений для работы с глобальной сетью Интернет. В пакет прикладных программ Microsoft Office встроен язык программирования VBA (Visual Basic for Applications – Visual Basic для приложений), представляющий усеченную версию языка Microsoft Visual Basic 2010. Таким образом, навыки программирования на Microsoft Visual Basic 2010 могут быть использованы при программировании на VBA в электронных таблицах Microsoft Excel, текстовом процессоре Microsoft Word и в системе управления базами данных Microsoft Access.

**VBA** — универсальный язык. Освоив его, вы не только получите ключ ко всем возможностям приложений Office и других, перечисленных ранее, но и будете готовы к тому, чтобы:

• создавать полноценные приложения на **Visual** **Basic** (поскольку эти языки — близкие родственники);

• использовать все возможности языка VBScript (это вообще "урезан-ный" **VBA**). ...

**VBA** изначально был ориентирован на пользователей, а не на профессио-нальных программистов (хотя профессионалы пользуются им очень ак-тивно), поэтому создавать программы на нем можно быстро и легко. Кро-ме того, в Office встроены мощные средства, облегчающие работу пользо-вателя: подсказки по объектам и по синтаксису, макрорекордер и т. п.

**Сервер среды** представляет базу данных производственного процесса, работающего в режиме реального времени.

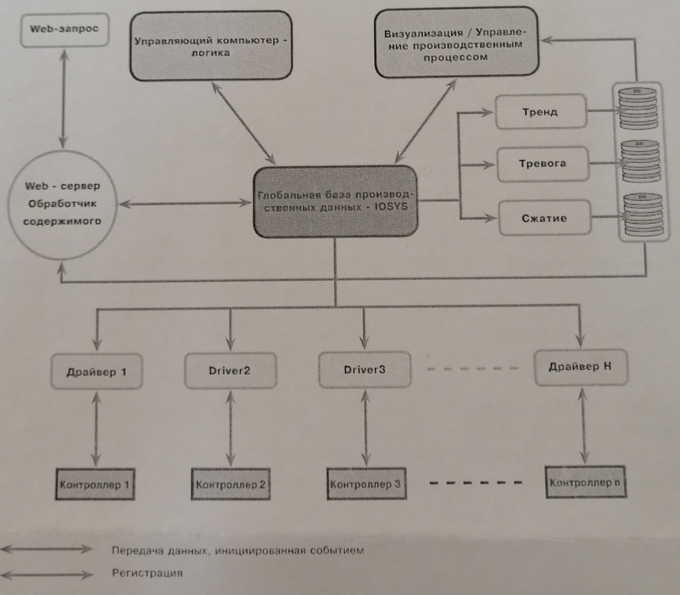
**Станция оператора** используется для контроля производственного процесса.

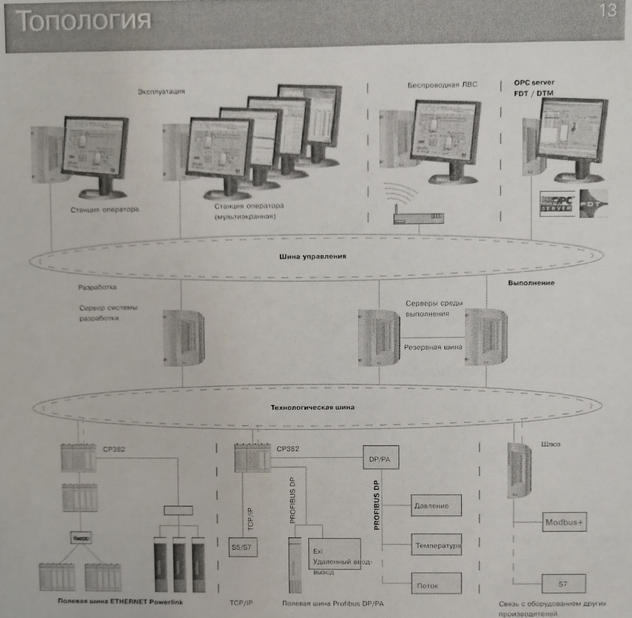
С точки зрения производственного процесса является надёжность и контроль.

Туда, где эти данные обрабатываются(сервер среды, станция оператора- ответственная за эксплуатацию и контроль), чтобы обеспечить наивысшую работоспособность сети.

**Глобальная база производственного этапа**

IO SYS. Программа IOSYS является глобальной онлайновой базой производствавенных данных и содержит все текущие технологические значения, IO SYS поддерживает хранение данных, распределение данных, и организацию данных для всей системы APROL.

Различные программы / клиенты (например, драйверы контроллера) снабжают IO SYS технологическими значениями. Эти технологические значения затем используются другими программами клиентами (например, Display Center на станции оператора). 



Архитектура Aprol

Система управления производственными процессами APROL R 3.0 включает три основных компонента:

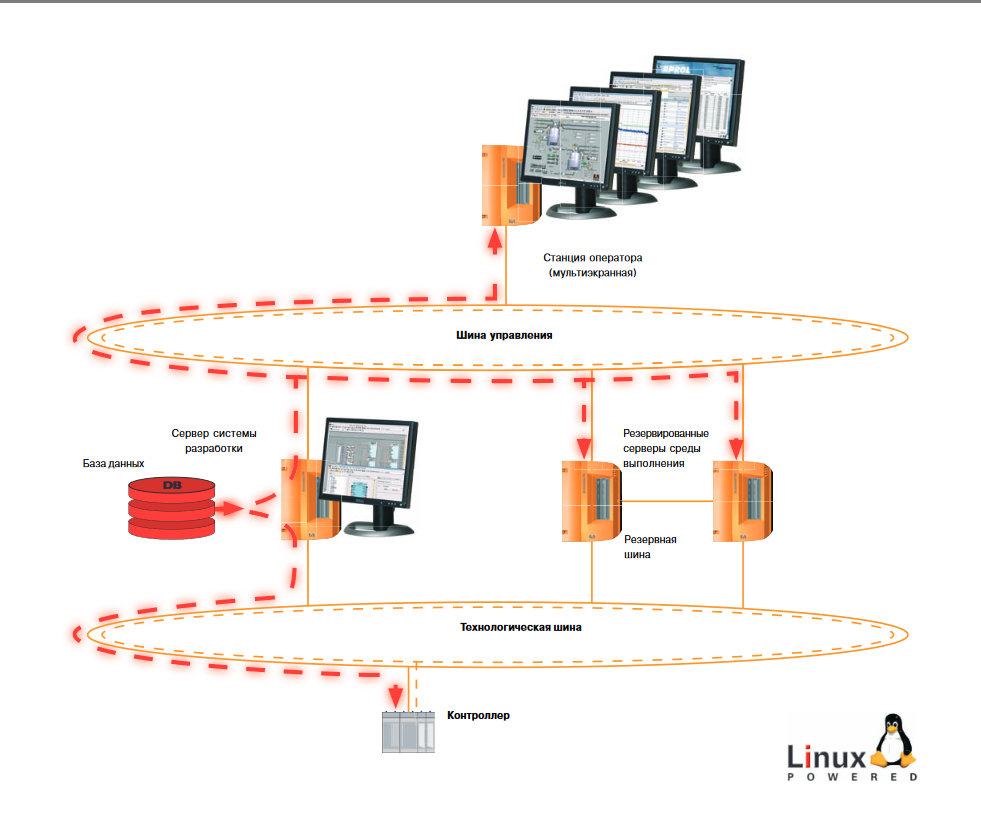
* Сервер системы разработки
* Сервер среды выполнения
* Станцию оператора

Сервер системы разработки содержит системную глобальную базу данных разработки. Другими словами, все объекты, которые созданы, конфигурировали, и которым присвоены параметры в инструменте автоматизированной разработки "CaeManager", подвергаются семантическому и синтаксическому разбору в централизованном каталоге объектов. Все объекты системы управления производственными процессами за загружаются из этой глобальной базы данных разработки в желательный целевой ресурс.

Сервер среды выполнения предоставляет базу данных системы управления производственным процессом, работающую в режиме реального времени, и содержит все технологические значения, системные значения, сообщения и тревоги.

Сервер среды выполнения также координирует архивирование технологических значений и событий в базу данных архивов. Станция оператора используется для управления и контроля производственного процесса на предприятии и отображает все архивированные данные для оператора.

Процесс загрузки  
 В ходе загрузки все объекты передаются из глобального каталога объектов (глобальной системной базы данных разработки) на контроллеры, серверы среды выполнения, и станции оператора. Если серверы среды выполнения выполнены с резервированием, то производится отдельная загрузка на оба сервера среды выполнения.



Самостоятельно: промышленный коммутаторы

**Промышленные коммутаторы**

Современные промышленные коммутаторы – это более 500 наименований разных устройств. Это могут быть и гигабайтные промышленные коммутаторы или модели монтажа в стойку с функциями PoE и SFP-портов.

## Особенности и примеры промышленных коммутаторов

Промышленные коммутаторы по принципу действия практически не отличаются от другого аналогичного оборудования.

Главной целью промышленных коммутаторов является выполнение тех же самых задач, то есть объединения отдельных узлов компьютерной сети на разных сегментах и передачи сведений от одного пользователя к другому.

**Но, несмотря на одинаковое предназначение, промышленные коммутаторы обладают определенными отличиями, а именно:**

* могут успешно функционировать в неблагоприятных условиях;
* в сравнении с другими видами коммутаторов, промышленные модели менее подвержены электромагнитному излучению и перепадам напряжения в сети;
* оборудование может успешно выполнять свои функции при низкой или высокой окружающей температуре.
* имеют небольшие габариты;
* могут устанавливаться на рейку DIN;
* прекрасно переносят многие механические воздействия и вибрацию. По степени защищенности относятся к IP30;
* промышленные изделия имеют запасной источник питания;
* на устройстве создается один или несколько тревожных конфигурируемых выходов.

Особое внимание следует уделить повышенной отказоустойчивости. С целью улучшения этого показателя в промышленные коммутаторы встраивается протокол STP. За счет этого можно объединить несколько отдельных коммутаторов в единую систему – кольцо.

При инициации этого протокола, коммутатор самостоятельно проводит структуру будущей локальной сети. Одновременно происходит отключение некоторых портов для исключения циклов пакетов.

Если же происходит разрыв, один из отключенных портов начинает работать, и система продолжает функционировать в обычном режиме.

В итоге, восстановление рабочей сети происходит без вмешательства инженера. Уже разработано несколько подобных протоколов, которые отличаются принципом действия и скоростью функционирования, однако работают по одной схеме.

В отдельных моделях коммутаторов имеется возможность агрегации портов. Эта процедура позволяет использовать несколько коммутаторов для единого канала. При агрегации портов происходит существенное улучшение связи, и пропускная способность повышается.

## Виды управляемых и неуправляемых промышленных коммутаторов

Раньше управляемые и неуправляемые коммутаторы имели принципиальные различия. Управляемый прибор был способен проводить рассылку по адресам, а неуправляемый не обладал функцией обработки каких-либо сигналов. Но в настоящее время в каждом из устройств появляется чип ASIC.

Это устройство дает возможность обработки фрейма, а также информации об адресе MAC. При этом работа с неуправляемыми коммутаторами становится более безопасной, так как количество вирусов значительно уменьшается.

Сегодня основным отличием управляемых коммутаторов является возможность получить доступ к настройкам, в то время как в неуправляемых коммутаторах, настройки изначально прописаны изготовителем.

## Способы соединения промышленных коммутаторов

Это сетевые устройства, обеспечивающие возможность соединения узлов компьютерной сети в рамках единичного или же нескольких секторов сети, применяемые на объектах промышленного назначения, в транспортной и электроэнергетических сферах. Работа данной техники осуществляется на OSI по канальному уровню.

**Промышленные коммутаторы** будут обширно представлены **на выставке Связь в ЦВК «Экспоцентр»**.

Основой работы устройств подобного типа служат мостовые технологии, коммутатор сохраняет в кэш таблицу коммутации, где имеется запись привязанности МАС-адрес узла к порту устройства.

***На сегодняшний день используют только три основных способа соединить узлы коммутатором:***

1. С промежуточным хранением (считка информации, сканирование ошибок, выбор порта назначения, отправка кадра).
2. Сквозной способ (считка адреса порта, коммутация).
3. Способ бесфрагментного соединения. Многие его также называют гибридным, и он является разновидностью сквозного способа.

Коммутаторы активно используют буфер собственной памяти для временного размещения там фреймов в том случае, когда уже занят порт, на который направляется сигнал.

Буферизация действует посредством двух методов: по портам или с общей памятью. Таким образом, процедура с использованием буфера памяти играет важную роль в работе коммутатора в целом.

Промышленные коммутаторы в силу того, что они эксплуатируются в основном в плохо отапливаемых зданиях и складах, а также просто на улице, должны обладать высоким уровнем защищенности, который допускает работу в столь суровой среде.

Необходимое условие – соответствие всем стандартам, а также поддержание работоспособности при высоких и крайне низких температурах. Пристальное внимание стоит уделять именно надежности создаваемой сети.

Выпускаются коммутаторы рядом компаний, занимающихся производством сетевого оборудования. Все модели позволяют распределить трафик и питание в сетях.

Производят управляемые и неуправляемые версии коммутаторов. Устройства без управления используются в сети на самых низких уровнях, там в данной функции просто нет необходимости. Неуправляемые коммутаторы просты и стоят намного меньше, чем управляемые.

**Ethernet Power in Модель ISO. Автоматизация**

Для связи систем различных изготовителей нужно согласовывать типы коммуникационной среды, процедуры доступа, протоколы.

Основа для … систем связи является общая стандартная модель передачи данных(соединение открытых систем. Разработано международной организацией по стандартам). 7 уровней модели

1. Физический
2. Канальный
3. Транспортный
4. Сетевой
5. Прикладной
6. Приложения

**Определение и характеристики**

ETHERNET Powerlink: Промышленная сеть Ethernet, работающая в режиме реального времени - это реальность Ethernet имеет гарантированное будущее.

Базовые концепции Ethernet доступны уже более 30 лет. Длительные циклы амортизации, характерные для индустрии автоматизации, требуют надежного основания.

Технология Ethernet хорошо известна Ethernet и соответствующие протоколы сегодня широко известны. Огромный объем имеющийся инструментов, программ и компонентов приведет к дальнейшему сокращению цен. Ethernet - прозрачная технология Стандарты Ethernet объединяют различные протоколы передачи данных на основе IP. Интеграция информационных технологий и автоматизации при использовании Ethernet дает вам реальную функциональную совместимость с гибкостью Internet. Ethernet является системой реального времени C ETHERNET Powerlink, Ethernet также включает уровни датчиков и исполнительных механизмов, с временами цикла до 200 мкс, и радикальное улучшение точности синхронизации, лучше одной микросекунды.

**Структура сети**

B ETHERNET Powerlink выделяются области реального времени и области, где режим реального времени не требуется. Это разделение соответствует типичной концепции машины и технологического процесса. Оно также удовлетворяет растущим требованиям к безопасности с целью предотвращения хакерских атак на уровне машин или ущерба от ошибочной передачи данных на верхних иерархических уровнях сети. Фактические требования режима реального времени удовлетворяются в области реального времени. Менее критические с временной точки зрения данные прозрачным образом передаются между областью реального времени и стандартной областью с использованием стандартных IP кадров. А четкое разграничение между сетью машины и сетью предприятия с самого начала предотвращает потенциальные угрозы безопасности, сохраняя полную прозрачность данных. Ethernet. Ethernet - roccapu

**100вавоTx**

Fast Ethernet со скоростью поредачидамнх с 10о моет/с, Moпольвует аедообринуо тонологио с раодко лжильнойвитой парой, максималаная дина согманта кабеля (между 2 станциями или станцией м концентраторомукоммутатором) - 100 м.

**100BaseFx**

Fast Ethernet со скоростью передачи данных с 100 мбит с, звездообразная топология с разводкой волоконно-оптическим кабелем, Максимальная длина сегмента кабеля (между 2 станциями или станцией и концентратором коммутатором) - 400 м.

**Концентратор**

Центральное соединяющее устройство для звездообразной топологии с разводкой витой парой. Коммутатор центральное интеллектуальное соединяющее устройство для звездообразной топологий с разводкой витой парой и сегментацией сетей.

**RJ-45**

8-выводной миниатюрный разыем (разъем типа Weser); используется, главным образом, для витой пары, Витая пара медный кабель с четырмя или восемью проводами, Два провода (пара проводов) свиты вместе. Кабели подразделятся на различные категории, категория кабелей 5 определяет стандартную витую пару 100 мбит с.

**UTP**

Неэкранированная вытая пара витая пара без экрана.

**s/UTP**

Экранированная неэкранированная витая парящем экраном вокруг всех проводов. витая пара с об SystP Экранированная витая пара витая пара с общим экраном вокруг всех проводов и плюс экранами вокруг каждой пары проводов.

**Патч-кабель**

Витая пара заводского изготовления; используется для соединения сетевой станции с концентратором или коммутатором Кросс-соединительный кабель. Витая пара заводского изготовления (кросс-оверный кабель); используется для прямого соединения сетевой станции со второй сетевой станцией, без концентратора или коммутатора

**Топология**

Структура физическото соецинения сети, Различактся линейная и шинная топологии, колыцевые структуры, звездообразная и древовидная топологии, B Fast Ethernet ислользуется звездообразная или древовидная топологии, Есть ограничения на каскадное соединение концентраторов или коммутаторов (последовательне соединения).

**уРоВНи Ethernet POWERLINK**

Уровень 2 - МАС, коммутатор…

Уровень 2 описывает механизмы, использованные для безопасной передачи данных. На этот уровень включены стандарты Ethernet для управления доступом к среде (МАС). На этом уровне работают мосты и коммутаторы, В отличие от концентраторов, эти устройства интеллектуальны и анализируют полученные кадры Ethernet. Коммутатор анализирует распределение станций на сети, используя адреса источника и адресата. Таким образом, он может посылать полученные на Уровень 3 - IP, маршрутизатор…

**Вторая часть ТСP/IP - это межсетевой протокол** IP. Он включен на уровень 3 и ответственен за реальную передачу данных по сети. На этом уровне работают маршрутизаторы. Они соединяют подсети, используя iP-адреса, которые уникальны в сети. Поэтому коммутаторы на уровне 3 - это маршрутизаторы.

**Уровень 4 - TCP, UDP…**

**Уровень 4 (транспортный уровень) включает пер**вую часть термина ТСP/IP, Протокол управленипередачей, TСР можно сравнить с UDP (Протоколом пользовательских датаграмм). Оба реализуют обмен данными между тремя нижними уровнями (системой транспортировки) и тремя верхними уровнями (прикладной системой). Единственное отличие -- метод, которым обрабатывается поток данных. ТСР предлагает больше механизмов защиты, чем UDP, но медленнее из-за обширной обработки данных.

**Уровень 5, Уровень 6, Уровень 7**

**Сеанс, презентация, приложение…**

Уровни 5, 6 и 7 формируют прикладную систему. Они ответственны за структурирование связи, представление данных и синхронизацию передаваемых данных. Все приложения, использующие TCP/IP, включены на эти уровни.

**Уровень протокола**

**Сетевой уровень - уровень 3**

На сетевом уровне устанавливастся логическое соединение между передатчиком и приемником, Работа сетевого уровня важна для маршрутизации сети, когда данные передаются более, чем между двумя точками, Кроме маршрутизации сети, выполняется управление пакетами.

Таким образом, транспортный протокол управляет работой всей сети.

**Транспортный уровень - уровень 4**

Транспортный уровень отделяет уровни, технологически зависимые от верхних уровней. На этом уровне логические адреса преобразуются в физические адреса, определяется маршрутизация сети и распределение пакетов

**Сеансовый уровень - уровень 5**

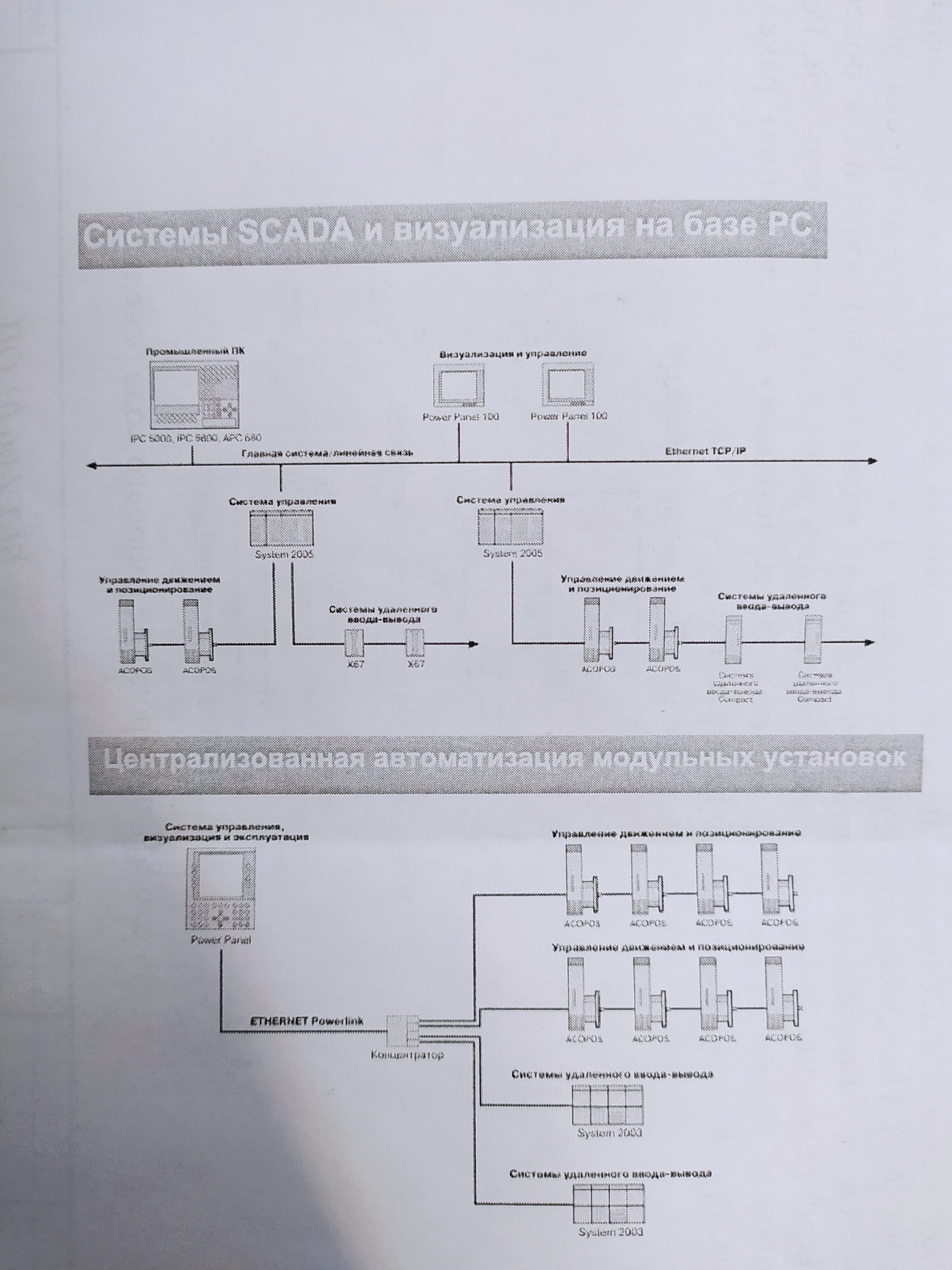
Этот уровень отвечает за синхронизацию задач на уровнях передачи и приема и содержит такие функции, как установка соединений, организация буферизации данных, проверка права доступа и контроль соединений.

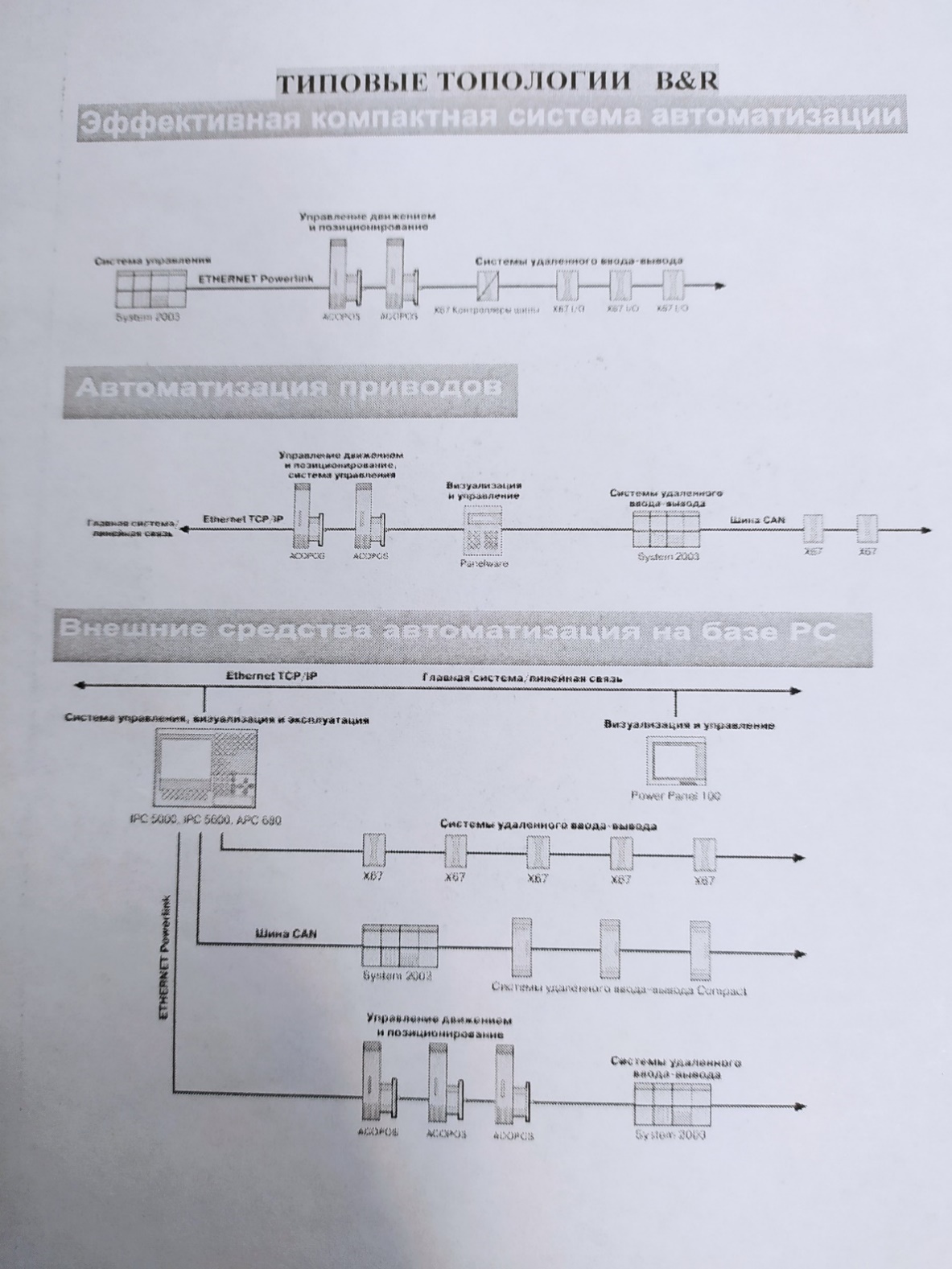
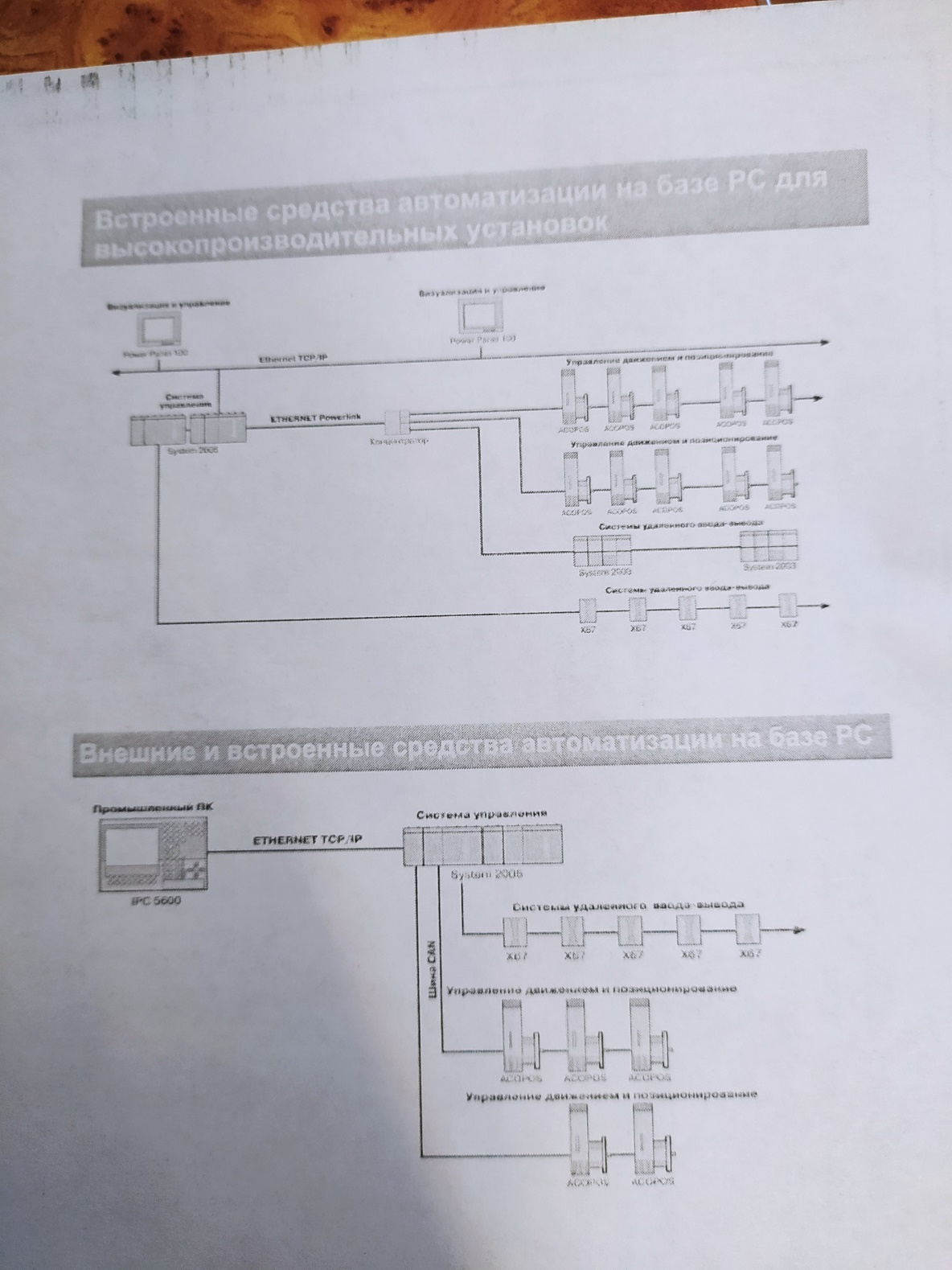
**Уровень представления - уровень 6**

На этом уровне формат данных узла преобразуется с учетом синтаксиса, используемого в сети. Дополнительные задачи включают, например, кодирование или сжатие данных.

**Прикладной уровень - уровень 7**

Этот уровень обеспечивает интерфейс между пользователем и сетью для работы с сетевыми приложенными без дополнительных специальных знаний по функционированию сетей.





**Режимы работы устройства**

Устройство Ethernet power … может работать в двух режимах:

1. Этот режим используется по умолчанию после включения устройства.
2. Предпусковой. В ходе запуска системы или после подсоединения устройства к существующей сети данные конфигурации загружаются по каналу(асинхронно).
3. Ethernet power in. После завершения процедуры загрузки устройство работает в реальном времени. Управляющий узел проверяет синхронизацию. Время цикла зависит от объёма данных, а также от количества узлов. Базовый цикл состоит из следующих фаз:
   1. Все синхронизирующие часы синхронизируются с …
   2. Управляющий узел присваивает каждому узлу фиксированное временное окно, для передачи критических данных. На этой фазе все другие узлы всегда могут принимать все данные(принцип издатель – подписчик).
   3. Асинхронная фаза. Управляющий узел представляет право передавать конкретные данные одному конкретному узлу. На этой фазе используются стандартные IP протоколы и адреса.
   4. Детарненированная синхронизация достигнута графиком циклической синхронизации для всех соединённых узлов, при обращении к физическому уровню.

На изонхронной фазе передаются критические данные.

Асинхронная фаза резервирует … Здесь освобождается доступ к физической среде, используя явные сообщения. В результате к сети всегда имеет доступ только один узел, что предотвращает столкновение.

В системе … для каждого узла используются уникальные мак-адреса. Кроме того в режиме реального времени присвоены идентификаторы …

**Назначение адресов в IP.**

Сервер имеет пул адресов и раздаёт их всем остальным узлам. Это означает, что каждый узел не имеет постоянного Ip-адреса. Каждый узел получает новый адрес, когда заканчивается его аренда. **OPC** ([аббр.](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0" \o "Аббревиатура) от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Open Platform Communications*[[1]](https://ru.wikipedia.org/wiki/OPC#cite_note-1), ранее [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [***O****LE*](https://ru.wikipedia.org/wiki/Object_Linking_and_Embedding) *for* ***P****rocess* ***C****ontrol*) — семейство [программных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) технологий, предоставляющих единый [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) для управления объектами [автоматизации](https://ru.wikipedia.org/wiki/%D0%90%D0%A1%D0%A3%D0%A2%D0%9F) и [технологическими процессами](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81). Многие из OPC протоколов базируются на [Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows)-технологиях: [OLE](https://ru.wikipedia.org/wiki/Object_Linking_and_Embedding), [ActiveX](https://ru.wikipedia.org/wiki/Microsoft_ActiveX), [COM](https://ru.wikipedia.org/wiki/Component_Object_Model)/[DCOM](https://ru.wikipedia.org/wiki/Distributed_Component_Object_Model). Такие OPC протоколы, как [OPC XML DA](https://ru.wikipedia.org/w/index.php?title=OPC_XML_DA&action=edit&redlink=1) и [OPC UA](https://ru.wikipedia.org/wiki/OPC_UA), являются [платформонезависимыми](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

Создание и поддержку спецификаций OPC координирует международная некоммерческая организация [OPC Foundation](https://ru.wikipedia.org/w/index.php?title=OPC_Foundation&action=edit&redlink=1), созданная в [1994 году](https://ru.wikipedia.org/wiki/1994_%D0%B3%D0%BE%D0%B4) ведущими производителями средств промышленной автоматизации. [Девиз](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B2%D0%B8%D0%B7) OPC Foundation — «Открытые коммуникации по открытым протоколам».

OPC – это программная технология, которая определяет правила взаимодействия оборудования и программного обеспечения верхнего уровня. В расшифровке названия OPC содержится еще одна аббревиатура: OLE – Object Linking and Embedding, это технология Microsoft, предназначенная для использования ресурсов одних программ внутри других. Например, проигрывание звукового файла, внедренного в текстовый документ Word реализуется как раз с помощью OLE. В настоящее время эта технология известна под называнием ActiveX и широко применяется, например, в web-программировании.

**OPC-client**

OPC представляет собой протокол передачи данных по сети, основанный на технологии OLE (OLE for Process Control) и предоставляющий единый интерфейс для управления объектами автоматизации и технологическими процессами.  OPC призван унифицировать передачу данных между различными устройствами, интерфейсы которых могут быть несовместимы.

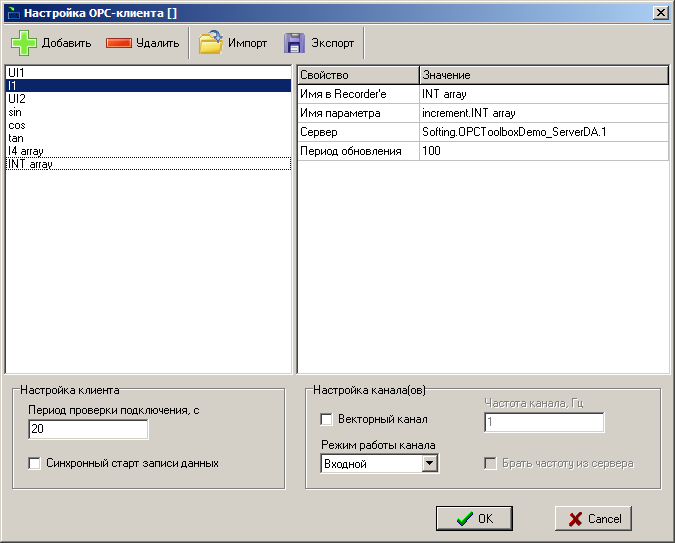
Основными составными частями протокола являются сервер, клиент и группа. Принимающая станция (клиент) может подключаться к различным передающим станциям (серверам). Для подключения к отдельному серверу нужно выделить отдельный объект клиента, то есть принимающая станция состоит из набора нескольких клиентов, каждый из которых может подключаться к  определённому серверу. Сервер, в общем случае, передает несколько параметров, список которых  можно у него запросить, выбрав те, которые желает получать клиент. Для  получения параметров их необходимо добавить в группу.

Группа – это список параметров с общей смысловой нагрузкой. Параметры могут группироваться по различным признакам без каких-либо ограничений. Клиент может произвольно группировать параметры. Группа существует как на стороне клиента, так и на стороне сервера. Клиент сначала должен сформировать группу, а затем уведомить об этом  сервер, который в свою очередь создаст ее у себя.

Обмен данными осуществляется с помощью группы. Можно использовать несколько способов обмена: синхронный обмен, асинхронный и по подписке. Синхронный обмен заключается в том, что клиент запрашивает данные и ожидает их получения. До тех пор, пока он не получит всех данных из соответствующей группы, выполнение программы не продолжается.  Асинхронный обмен, наоборот, предполагает, что клиент не будет дожидаться немедленного получения данных, а сервер специально сообщит ему об этом. Метод по подписке заключается в том, что клиент информирует сервер о желании получать данные с определенной частотой, после чего сервер передает ему данные с заданной частотой.

Специфика передачи параметров по OPC заключается в том, что они передаются, только в том случае, если изменили свое значение.

Плагин является клиентом OPC, то есть принимающей станцией, которая может получать данные с нескольких передающих станций (серверов). Для этого нужно указать перечень параметров от различных серверов, которые необходимо принимать. Каждый параметр принадлежит какому-то одному серверу, имеет свое имя и период обновления. Параметры, их свойства, а также общий вид окна настройки плагина представлены на рисунке 1.



**Рис. 1** Окно настройки принимаемых параметров

На рисунке 2 показано, как выбрать параметры для сервера, выбранного с помощью кнопки на панели управления. Описание сервера указывается прямо на этой кнопке.



**Рис. 2**  Добавление параметров

Список параметров можно сохранять и загружать из файла, для этого есть соответствующие кнопки. При начале работы будут загружаться настройки из того файла, с которым в последний раз работали. Название этого файла указывается в заголовке окна.

Действия, выполняемые программой, такие как, начало и конец приема, подключение  и переподключение к серверам, загрузка и сохранение конфигурации, протоколируются в файле.

**Общие сведения о периферийных устройствах и их связей с микроЭВМ**

1. общие сведения опериферийныхустройствахиихсвязисмикроэвм1.1. Назначение и классификацияДля эффективного использования ПЭВМ большое значение имеет оборудование их периферийными устройствами.

Функциональные возможности ПЭВМ в значительной мере зависят от применяемого набора и технических характеристик периферийных устройств.

Периферийные устройства ПЭВМ предназначены для ввода, вывода и хранения данных с последующим их вводом в ПЭВМ, являются средствами коммуникации ПЭВМ с внешними источниками и потребителями информации и обеспечивают согласование сигналов внешних объектов и сигналов, используемых в ПЭВМ. В процессе своей работы они лишь преобразуют данные из одной формы представления информации в другую, не меняя их содержания.

Периферийные устройства ПЭВМ различаются по назначению, виду обслуживаемых объектов, физической природе обрабатываемых сигналов и носителей информации, принципу действия, техническим и эксплуатационным характеристикам, конструктивному исполнению и другим признакам.

Все периферийные устройства ПЭВМ обычно подразделяют на три большие группы: 1) устройства связи ПЭВМ с человеком пользователем; 2) устройства связи с объектами контроля и управления; 3) устройства для длительного хранения информации большой емкости.

К первой группе относятся устройства ввода вывода и передачи информации, необходимые для общения человека с машиной, и электронные функциональные модули, их обслуживающие.

Это, прежде всего, клавиатуры, различные планшеты (графоповторители) для работы с чертежами, устройства управления курсором дисплея (манипуляторы типа «мышь», «джойстик» и т. п.), видеотерминалы (дисплеи), печатающие устройства, сканеры, графопостроители. В последние годы быстро развиваются средства ввода вывода человеческой речи.

Во вторую группу периферийных устройств включают устройства связи ПЭВМ с объектами и интерфейсами приборных систем: различные датчики и исполнительные органы, цифроаналоговые(ЦАП) и аналого-цифровые (АЦП) преобразователи устройства, необходимые для преобразования непрерывных сигналов с датчиков в цифровые сигналы и обратного преобразования при выдаче информации на исполнительные органы. С помощью периферийных устройств данной группы ПЭВМ приспосабливаются для управления технологическими процессами и оборудованием, автоматизации контроля и измерений, сбора данных и т.п. Все больше ПЭВМ оборудуется также средствами телеобработки, позволяющими подключать ПЭВМ к локальным и распределенным сетям ЭВМ.

Значительный интерес представляют модемы периферийные устройства, позволяющие передавать информацию от одной ПЭВМ к другой по телефонному каналу.

Третью группу периферийных устройств ПЭВМ составляют внешние запоминающие устройства (ВЗУ): накопители на гибких и жестких магнитных дисках, на магнитных лентах, на микросборках ЦМД,

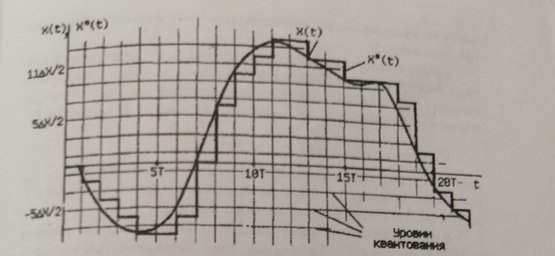
Серьезными конкурентами накопителям на магнитных носителях информации в ПЭВМ являются накопители на магнитооптических и оптических дисках. Характерна также классификация периферийных устройств в зависимости от выполняемых ими в ПЭВМ функций. В соответствии с ней периферийные устройства ПЭВМ можно разделить на две категории. К первой относятся периферийные устройства, без которых практически невозможно функционирование вычислительной системы: клавиатуры, накопители на магнитных дисках, дисплеи, печатающие устройства, т. е. периферийные устройства, входящие в базовый комплект ПЭВМ. Такие периферийные устройства часто называют системными (общего назначения). Ко второй категории относят устройства, которые предоставляют дополнительные возможности пользователю профессиональной ПЭВМ, позволяя создавать рациональную конфигурацию вычислительной системы в зависимости от ее конкретной профессиональной ориентации (накопители на магнитной ленте, манипуляторы, графические планшеты, графопостроители, сканеры, коммуникационные и интерфейсные адаптеры, средства мультимедиа и т. д.). Подобные периферийные устройства называют дополнительными. Подключение различных по функциональному назначению, принципу работы, интерфейсу и конструктивному исполнению периферийных устройств к центральному процессору ПЭВМ осуществляется через электронные функциональные модули. Эти модули выполняют обычно согласование интерфейсов периферийного устройства и системной шины ПЭВМ, буферизацию информации, а также некоторые локальные функции, часто сложные, освобождая от них центральный процессор. Электронные функциональные модули, обслуживающие работу периферийных устройств в составе ПЭВМ, аналогично самим периферийным устройствам могут быть разделены на две группы. Модули первой группы управляют работой системных периферийных устройств, их называют контроллерами (управляющими устройствами). Модули второй группы приспосабливают дополнительные периферийные устройства для работы с ПЭВМ. Поскольку модули служат для адаптирования внешних сигналов периферийных устройств к сигналам системной шины ПЭЭВМ, их часто называют адаптерами. Такое деление электронных функциональных модулей ПЭВМ довольно условно, поэтому в литературе понятия «контроллер» и «адаптер» в большинстве случаев считаются синонимами. Принцип подсоединения периферийных устройств к ПЭВМ (как системных, так и дополнительных) единый: через системную шину к центральному процессору подключается электронный функциональный модуль, управляющий работой данного периферийного устройства, этот же модуль электрически соединяется кабелем с самим периферийным устройством. Для подключения к ПЭВМ периферийных устройств, производимых различными предприятиями (фирмами), наборы сигналов, передаваемых по кабелю и разъемным соединителям, техническое исполнение, а также правила обмена информацией между периферийным устройством и адаптером образуют систему, называемую интерфейс периферийного устройства. Поэтому иногда электронные функциональные модули называют также интерфейсами, что, однако, не отражает сути последнего понятия. В профессиональной ПЭВМ обычно реализуются стандартные или унифицированные интерфейсы для подключения клавиатуры, дисплея, внешних запоминающих и других устройств. Электронные функциональные модули, адаптирующие к ПЭВМ периферийные устройства, осуществляют не только управление периферийным устройством, но и обмен данными через системную шину между центральным процессором и периферийным устройством. Они представляют собой сложные устройства и часто содержат микропроцессорные схемы. Для управления процессом обмена информацией предварительно составляются и записываются в память специальные управляющие программы драйверы. При необходимости выполнения операции ввода вывода работа текущей программы прерывается и управление передается драйверу (через системную шину необходимая команда поступает к адаптеру, который через соответствующий интерфейс взаимодействует с подключенным периферийным устройством).По окончании работы периферийного устройства управление снова передается основной программе.1.2. Критерии выбора периферийных устройств

При выборе периферийных устройств и дополнительных электронных функциональных модулей пользователь ПЭВМ должен принимать во внимание их совместимость с используемой ПЭВМ, технические параметры (скорость обмена информацией, мощность, надежность работы и т. д.), которые должны соответствовать характеристикам ПЭВМ, а также наличие служб технического обслуживания и ремонта, запасных частей и других принадлежностей (например, НГМД, ленты и бумаги для печати).Важное значение для профессиональных ПЭВМ имеет возможность использования средств машинной графики аппаратуры (графических дисплеев, графопостроителей, устройств ввода и обработки графических изображений), методов и приемов для преобразования данных в их графическое представление и наоборот. Характеристики периферийных устройств (стоимость, масса и габаритные размеры, быстродействие, надежность и др.) в основном определяют характеристики всей вычислительной системы на базе ПЭВМ. Во многом это связано с тем, что традиционно периферийные устройства содержат большое количество громоздких и дорогостоящих, медленнодействующих и малонадежных электромеханических, а также механических узлов и деталей. Применение указанных узлов в периферийных устройствах сдерживает возможности комплексной микроминиатюризации ПЭВМ и обусловливает невысокие эксплуатационные характеристики периферийных устройств. В настоящее время в мире разрабатываются более совершенные малогабаритные периферийные устройства для ПЭВМ, в том числе основанные на нетрадиционных принципах преобразования и хранення информации.1.3. Кодированное информации

Так как большинство внешних устройств (ВУ) позволяет человекуобщаться с ЭВМ на привычном ему языке слов и десятичных чисел, а ЭВМ понимает лишь язык физических состояний, кодируемых двоичными числами, то в ВУ обычно производится кодирование (декодирование) информации, пересылаемой в (из) микроЭВМ. Для кодирования русских (33 буквы) и латинских (26 букв) прописных и строчных букв, а также различных графических (например, символа %) ислужебных (например, код для перевода строки в пишущей машинке)самволов необходимо иметь более чем 27=128 различных кодов. Поэтому кодирование обычно производится в 8битовом (1байтовом)коде обмена информацией.

Так как ВУ имеют достаточно много механических элементов, тоони менее надежны, чем электронные схемы процессора и памяти. Причиной неправильного восприятия кода считываемого символа могут быть пыль на считывающих головках накопителей на магнитных лентах или дисках, царапины на самих лентах и дисках и т. п. Искажения изза электрических помех возникают и при передаче данных между процессором (памятью) и ВУ (контроллером ВУ). Короче говоря, если надо ввести, вывести или передать символы, то могут появиться ошибки. Для обнаружения таких ошибок (а иногда и для их исправления) коды символов преобразуют в какую-либо стандартную форму(например, добавляют еще один бит, чтобы в новом коде было четное число единиц). При передаче такого стандартного кода производится определённого вида контроль (например, контроль четности), и по его результатам принимается решение об использовании полученногосимвола.1.4. Способы обмена данными между ВУ и микро ЭВМ

Сушествуют два способа передачи слов информации по линиям данных между ВУ и микро ЭВМ:\* параллельный, когда одновременно пересылаются все биты слова; последовательный, когда биты слова пересылаются поочередно, Упрощения путём ограничения их словаря (до ненки системы на конкретного «диктора». Перед началом приметы пользователь формирует определенный словарь. Затем он несколько раз произносит каждое слово, чтобы дать возможность машине запомнить варианты произношения, и после этого система оказывается готовой к работе. Она «слушает» слова, произносимые пользователем,и сопоставляет их с эталонами, хранящимися в словарной базе данных.

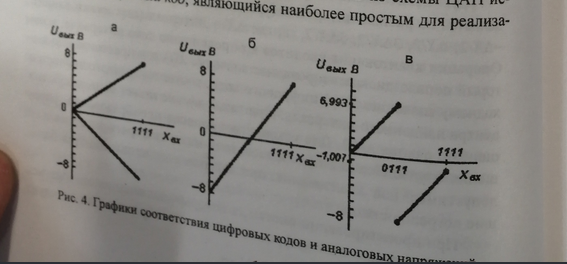
Если обнаруживается совпадение, идентифицируется соответствующее слово; в противном случае пользователь делает повторную попытку речевого ввода либо вводит информацию традиционным способом (например, с клавиатуры).Синтез речи, или процесс формирования речевых сигналов из фрагментов цифровых данных, находящихся в памяти ЭВМ, прощевсего реализуется путем составления ответа из слов, введенныхпользователем на этапе обучения ЭВМ. В этом случае машина выводитинформацию голосом пользователя.1.7. Устройства для связи микроЭВМ с объектамиуправленияМикроЭВМ управляют станками и игровыми автоматами, ходом научных экспериментов и доильными аппаратами, автомобилями и роботами. В этих (а также других) устройствах, приборах и системах используется много серийных электрических датчиков и исполнительных органов(например, электродвигателей), работающих с непрерывными (аналоговыми) электрическими сигналами. Для связи таких датчиков и исполнительных органов с микроЭВМ требуется преобразовать аналоговый сигнал в число, пропорциональное амплитуде этого сигнала, и наоборот.Процедура аналогоцифрового преобразования включает два этапа: дискретизацию по времени (выборки) и квантование по уровню.Процесс дискретизации состоит из измерения значений непрерывногосигнала X(t) только в дискретные моменты времени 0, Т, 2Т, …, отстоящие друг от друга на величину периода дискретизации Т (рис. 3).Такой измерительный процесс, если он реализуется без необходимых мер предосторожности, может привести к возникновению знаРис. 3. Дискретизация и квантование непрерынного сигнала X(t)чительных погрешностей. Важно иметь достаточное число отсчетовза единицу времени для правильного представления существенной информации, содержащейся в высокочастотной составляющей этогосигнала. В то же время частые измерения требуют увеличения быстродействия преобразования и обработки, а следовательно, сложности истоимости преобразователя. Поэтому темп дискретизации необходимо поддерживать на минимально допустимом уровне, при котором составляющая общей погрешности, обусловленная процессом дискретизации, не превышает установленных пределов.

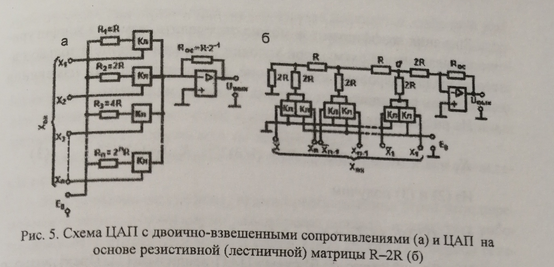
Для квантования диапазон изменения входного сигнала подразделяется на равные интервалы (уровни квантования): …, 5ДX/2, ЗДХ/2,AX/2, AX/2, 3AX/2, 5AX/2, …, где AX интервал (шаг квантования).Операция квантования сводится к определению того интервала, в который попало дискретизированное значение X(t), и к присваиванию выходному значению X\*(t) цифрового кода, соответствующего значению центра найденного интервала. При такой замене может быть допущена ошибка, равная ДX/2. Для ускорения процесса преобразования, упрочения и удешевления преобразователя надо выбирать максимально допустимый шаг квантования, при котором еще не появляются большие погрешности.

При проектировании систем, в которых должны использоваться аналогоцифровые или цифроаналоговые преобразователи, следует иметь хотя бы минимальное представление о принципах работы этих преобразователей.

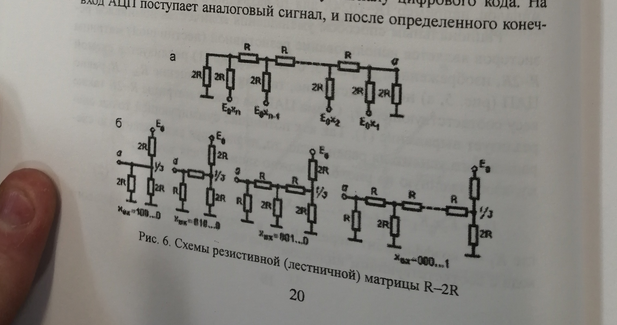
Цифроаналоговый преобразователь (ЦАП) функциональный узел, однозначно преобразующий кодовые комбинации цифрового сигнала в значения аналогового сигнала. Основой для нахождения однозначного соответствия может служить напряжение на выходе ЦАП:Unnx = Eo(x 21 +x222 + … +x,2 "),(1)где Бо опорное напряжение; X{x,, х,…, х) цифровой код; X1(1, 2, …,принимают значения 0 или 1.

При определенном Её каждому х; на выходе устройства соответствует напряжение UвыхіВ цифроаналоговых преобразователях используют три основныхдвончных кода: прямой, смещенный, дополнительный. Графики соответствия цифровых кодов Хах и аналогового напряжения Uвых при прямом (а), смещенном (б) и дополнительном (в) кодах показаны на рис. 4.

Наиболее просто определяется соответствие цифровых и аналоговых величин при прямом коде. Этот код удобен при преобразовании сигналов следящих систем, так как при переходе через нуль не меняются старшие разряды кода, что позволяет реализовать линейный переход от малых положительных к малым отрицательным выходным напряжениям. Для преобразования как положительных, так и отрицательных кодов используют знаковый разряд, который управляет переключением выходного напряжения ЦАП.Для исключения коммутирующих элементов из схемы ЦАП используют смещенный код, являющийся наиболее простым для реализа

Рис. 5. Схема ЦАП с двоично взвешенными сопротивлениями (а) и ЦАП на основе резистивной (лестничной) матрицы R2R (б)ции в схеме преобразователя.

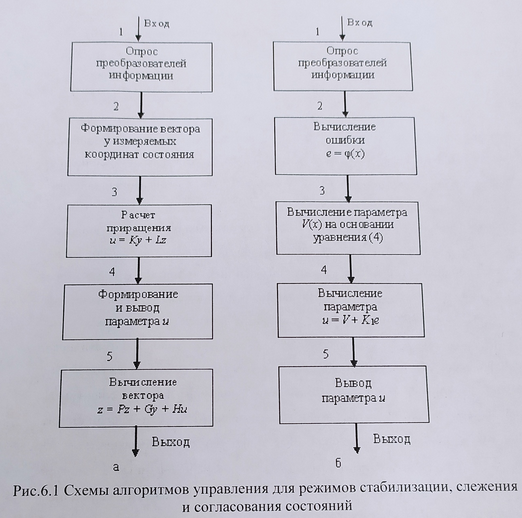
При применении дополнительного кода положительные числа преобразуются так же, как для прямого кода, а отрицательные двоичным дополнением соответствующего положительного числа (инверсия всех разрядов с последующим добавлением единицы в младший разряд).Базовая схема ЦАП, реализующего выражение (1), показанная нарис. 5, а, содержит источник опорного напряжения Е0, матрицы двоично весовых резисторов, набор ключей и дифференциальный операционный усилитель. Основные недостатки этой схемы определяются необходимостью применения резисторов с большим диапазоном номиналов, например 1R1024R для 10разрядного ЦАП. Рациональным способом уменьшения количества номиналов резисторов является использование резистивной (лестничной) матрицыR2R, изображенной на рис. 5, б. Выражение (1) реализуется схемойЦАП (рис. 5, а) непосредственно, так как соотношение Rac | R; равновесу соответствующего х;. Схема ЦАП на основе матрицы R2R такжереализует выражение (1). Так как потенциал суммирующей точки операционного усилителя равен нулю, то, анализируя эквивалентную схему, изображенную на рис. 6, а, можно записать для точки аU, E,x,K + E,x,K, +…+E,x,K,,a(2)где К коэффициент передачи Его в точку а от разряда цифрового кода с соответствующим индексом. Значения коэффициентов можно определить, изменяя конфигурацию эквивалентной схемы, при условии равенства единице только одного разряда Цифрового кода. На рис. 6, б приведены эти изменения для четырех разрядов: трех старших и одного младшего. Из рисунка следует, что, =1/3; K, (1/3)2\*; K, = (1/3)2 2; K, (1/3)2 ".Из (2) и (3) получим(3)U (1/3)E (x,21 + x,2 2 + x, 2 +…+ x,2 ").При Ra = 3R (рис. 5, б) схема ЦАП реализует (1). Практическую реализацию многоразрядных схем ЦАП рационально проводить на основе микросхем, содержащих основные блоки ЦАП в одном корпусе. Источники опорного напряжения Е0, как правило, строятся на основе использования полупроводниковых стабилитронов. На рис. 7 приведены схемы источников опорного напряжения на основе операционного усилителя с параллельной обратной связью (а) и с последовательной обратной связью (б).Отношение сопротивлений R2/R1определяется требуемыми значениями Ед и U. Аналого-цифровое преобразователи (АЦП) решают задачу поиска однозначного соответствия аналоговому сигналу цифрового кода. На вход АЦП поступает аналоговый сигнал, и после определенного конечного сигнала.



**Проектирование программных систем полиграфических машин и систем обработки информации**

Новая информационная база микро ЭВМ контроллеры, позволяет реализовать систему управления, оптимальные законы управления и этим обеспечивает их более высокие статические и динамические показатели.

Типовыми режимами работы электропривода являются разгон, торможение, реверс. Стабилизация определённых параметров, при сильно изменяющихся нижних воздействиях, слежение, в том числе программное, при наличии внешних воздействий, согласование режимов стабилизации или слежения. Если в электроприводе последовательно реализуется два из указанных режима, то параметры стационарного регулятора с жёсткой структурой следует выбирать с учётом обеспечения приемлимого качества к процессам этих режимов. Наивысшее качество процессов может быть достигнуто в том случае, когда для каждого режима работы электропривода используется регулятор соответствующей структурой, настройка которого зависит от изменяющихся параметров объекта управления и внешних воздействий. При использовании управляющих эвм структура регулятора изменяется простым переходом от одной программы расчёта управляющего воздействия к другой.



Рассмотрим возможные структуры и методы проектирования цифровых регуляторов для каждого режима работы электропривода. Для режимов стабилизации и слежения при наличии возмущений может быть использован один и тот же типовой алгоритм управления, схема которого изображена на рис. 6.1, а. В блоке / этого алгоритма опрашиваются преобразователи информации, используемые в системе управления и измеряющие состояние электропривода и внешних воздействий. Формирование вектора ув блоке 2 заключается в переходе от абсолютных значений выходных величин преобразователей информации к отклонениям относительно номинальных (или программных) значений; здесь же в вектор у включается значение управляющего возвектора

z = Pz + Gy + Hu

Вывод параметраи

В блоке 3 рассчитывается приращение и управляющего воздействия относительно номинального управления, обеспечивающего номинальный (или программный) режим стабилизации. Исходными

данными для этого являются вектор у измеряемых координат и вектор z, являющийся оценкой неизмеряемых координат состояния. Матричные коэффициенты К и L являются матрицами-строками, если электродвигатель управляется по одному входу (по цепи якоря или по цепи возбуждения), и имеют две строки, если электродвигатель управляется по двум входам. Выбором матриц К и L обеспечиваются заданные свойства собственного движения системы управления электроприводом, а также компенсируются доминирующие возмущающие воздействия. Для расчета матриц К и L используются методы комбинированного, модального и оптимального управления. В блоке 4 формируется полное управляющее воздействие как сумма номинального и рассчитанного отклонений управления; это полное управляющее воздействие выводится на цифроаналоговый

Преобразователь. В блоке 5 вычисляется вектор оценки неизмеряемых координат состояния. Эта оценка используется для расчета управляющего воздействия в следующем такте. Выбором матриц Р, G и Н можно обеспечить требуемую скорость сходимости оценки, фильтрацию ошибок изме-

рения, экстраполяцию отсчетов, снимаемых с преобразователей информации, а также требуемый порядок астатизма системы. Одновременно требуется обеспечить выполнение некоторых со-отношений координат вектора состояния (фазовых координат) х вида ф(х) = 0, определяющих, например, условия постоянства частоты вращения двигателя, мощности т. п. В пространстве состояний рассматриваемое уравнение задает некоторое многообразие, в частности поверхность, отклонение от которого характеризуется ошибкой e = q(x).

Алгоритм управления, обеспечивающий минимизацию этой ошибки, имеет вид:

и = V(x) +Ke.

Аналитическое выражение для составляющей V(x) может быть получено из условия е = 0. Матрица K1 обеспечивает желаемое качество отработки ошибки и выбирается с помощью математического

обеспечения, используемого при синтезе регуляторов стабилизации или слежения. На основании сказанного блок-схема алгоритма управления со гласованием конечного и начального состояний режимов разгона и стабилизации имеет вид, изображенный на рисунке 6.1, б. В блоке 1 опрашиваются преобразователи информации и в ЭВМ вводятся координаты вектора состояния х электропривода. Если число преобразователей информации меньше числа координат вектора состояния и если выход преобразователей сильно зашумлен, то после блока 1 необходима. Затем рассчитывается отклоненис е изображающей точки от соответствующего многообразия и выбирается коэффициент усиления стабилизирующей составляющей управляющего воздействия (блок 5). В блоке б вычисляется основная составляющая V управляющего воздействия, обеспечивающая движение по выбранному многообразию. В блоке 7 формируется полное управляющее воздействие, которо затем выводится на объект управления (блок 8). Составляющая управления Ке обеспечивает ликвидацию скользящих режимов в том случае, когда основная составляющая V имест релейный характер. Все рассмотренные программы выполняются на одной и той же управляющей ЭВМ. Тип программы, т. е. структура регулятора выбирается с помощью специальной планирующей программы (рис. 6.3), исходными данными для которой являются команды с пульта управления, а также показания преобразователей информации о состоянии объекта управления и внешних воздействий. Планирующая программа начинает выполняться после пуска ЭВМ и прекращает - после ее останова. Сразу после пуска ЭВМ выполняется специальная программа установки начального состояния регистров и областей ОЗУ (блок 2). При этом задаются начальные условия для всех структур регулятора электропривода. Остальные блоки программы выполняются периодически в процессе работы управляющей ЭВМ.

В блоке 3 опрашиваются уставки с пульта управления, а также показания преобразователей информации. Введенные данные в блоке 4 сравниваются с данными, полученными на предыдущем цикле. Изменяющиеся условия работы электропривода индицируются присвоением логической переменной L единичного значения (блок б). При этом в блоке 5 запоминаются новые условия работы. Если же условия работы не изменяются, то в блоке 8 логической переменной L присва-

ивается нулевое значение. В блоке 9 в результате проверки логических условий, определяющих последовательность переключения структуры регулятора, находится номер і программы расчета управляющего воздействия, которая выполняется в данное время. При изменившихся условиях работы (L = 1 в блоке 10) в блоке 11 рассчитываются параметры программы П, (элементы матричных коэффициентов алгоритма стабилизации, коэффициенты уравнения многообразия алгоритма согласования и т. п.). Затем выполняется одна из описанных программ П, расчета управляющего

Воздействия.

Схема типового алгоритма такого оптимального управления приведена на рис. 6.2.

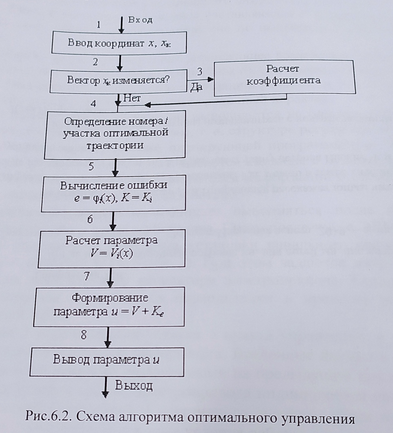


Рис.6.2. Схема алгоритма оптимального управления В блоке 1 осуществляется ввод координат вектора состояния х с преобразователей информации, а также ввод с пульта управления координат вектора конечного состояния хк, которое должно быть достигнуто в результате разгона, торможения или реверса. Если число преобразователей информации меньше числа координат вектора х,

то после блока 1 вводится блок оценки координат вектора х. Если с пульта управления поступает новая уставка хк (блок 2), то в блоке 3 рассчитываются новые значения коэффициентов аппроксимирующих функций pi(x) = 0. В блоке 4 путем проверки логических условий, полученных на этапе синтеза, определяется номер і участка оптимальной траектории.

Затем рассчитывается отклоненис е изображающей точки от соответствующего многообразия и выбирается коэффициент усиления стабилизирующей составляющей управляющего воздействия (блок 5). В блоке б вычисляется основная составляющая V управляющего воздействия, обеспечивающая движение по выбранному многообразию.

В блоке 7 формируется полное управляющее воздействие, которое затем выводится на объект управления (блок 8). Составляющая управления Ке обеспечивает ликвидацию скользящих режимов в том случае, когда основная составляющая V имест релейный характер.

Все рассмотренные программы выполняются на одной и той же управляющей ЭВМ. Тип программы, т. е. структура регулятора выбирается с помощью специальной планирующей программы (рис. 6.3), исходными данными для которой являются команды с пульта управления, а также показания преобразователей информации о состоянии объекта управления и внешних воздействий. Планирующая программа начинает выполняться после пуск ЭВМ и прекращает - после ее останова. Сразу после пуска ЭВМ выполняется специальная программа установки начального состояния регистров и областей ОЗУ (блок 2). При этом задаются начальные условия для всех структур регулятора электропривода. Остальные блоки программы выполняются периодически в процессе работы управляющей ЭВМ.

В блоке 3 опрашиваются уставки с пульта управления, а также показания преобразователей информации. Введенные данные в блоке 4 сравниваются с данными, полученными на предыдущем цикле. Изменяющиеся условия работы электропривода индицируются присвоением логической переменной L единичного значения (блок б). При этом в блоке 5 запоминаются новые условия работы. Если же условия работы не изменяются, то в блоке 8 логической переменной L присвавается нулевое значение.

В блоке 9 в результате проверки логических условий, определяющих последовательность переключения структуры регулятора, находится номер і программы расчета управляющего воздействия, которая выполняется в данное время. При изменившихся условиях работы (L = 1 в блоке 10) в блоке 11 рассчитываются параметры программы П, (элементы матричных коэффициентов алгоритма стабилизации, коэффициенты уравнения многообразия алгоритма согласования и т. п.). Затем выполняется одна из описанных программ П, расчета управляющего воздействия.

**Схема** — это абстракция какого-либо процесса или системы, наглядно отображающая наиболее значимые части. Схемы широко применяются с древних времен до настоящего времени — чертежи древних пирамид, карты земель, принципиальные электрические схемы. Очевидно, древние мореплаватели хотели обмениваться картами и поэтому выработали единую систему обозначений и правил их выполнения. Аналогичные соглашения выработаны для изображения схем-алгоритмов и закреплены ГОСТ и международными стандартами.

На территории Российской Федерации действует единая система программной документации (ЕСПД), частью которой является Государственный стандарт — **ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем»** [1]. Не смотря на то, что описанные в стандарте обозначения могут использоваться для изображения схем ресурсов системы, схем взаимодействия программ и т.п., в настоящей статье описана лишь разработка схем алгоритмов программ.

Рассматриваемый ГОСТ практически полностью соответствует международному стандарту **ISO 5807:1985**.

## Элементы блок-схем алгоритмов

Блок-схема представляет собой совокупность символов, соответствующих этапам работы алгоритма и соединяющих их линий. Пунктирная линия используется для соединения символа с комментарием. Сплошная линия отражает зависимости по управлению между символами и может снабжаться стрелкой. Стрелку можно не указывать при направлении дуги слева направо и сверху вниз. Согласно п. 4.2.4, линии должны подходить к символу слева, либо сверху, а исходить снизу, либо справа.

Есть и другие типы линий, используемые, например, для изображения блок-схем параллельных алгоритмов, но в текущей статье они, как и ряд специфических символов, не рассматриваются. Рассмотрены лишь основные символы, которых всегда достаточно студентам.

|  |  |
| --- | --- |
| flowcharts_terminatorТерминатор начала и конца работы функции | Терминатором начинается и заканчивается любая функция. Тип возвращаемого значения и аргументов функции обычно указывается в комментариях к блоку терминатора. |
| flowcharts_dataОперации ввода и вывода данных | В ГОСТ определено множество символов ввода/вывода, например вывод на магнитные ленты, дисплеи и т.п. Если источник данных не принципиален, обычно используется символ параллелограмма. Подробности ввода/вывода могут быть указаны в комментариях. |
| flowcharts_processВыполнение операций над данными | В блоке операций обычно размещают одно или несколько (ГОСТ не запрещает) операций присваивания, не требующих вызова внешних функций. |
| flowcharts_solutionБлок, иллюстрирующий ветвление алгоритма | Блок в виде ромба имеет один вход и несколько подписанных выходов. В случае, если блок имеет 2 выхода (соответствует оператору ветвления), на них подписывается результат сравнения — «да/нет». Если из блока выходит большее число линий (оператор выбора), внутри него записывается имя переменной, а на выходящих дугах — значения этой переменной. |
| flowcharts_procedureВызов внешней процедуры | Вызов внешних процедур и функций помещается в прямоугольник с дополнительными вертикальными линиями. |
| flowcharts_loopНачало и конец цикла | Символы начала и конца цикла содержат имя и условие. Условие может отсутствовать в одном из символов пары. Расположение условия, определяет тип оператора, соответствующего символам на языке высокого уровня — оператор с предусловием (while) или постусловием (do … while). |
| flowcharts_preprocessПодготовка данных | Символ «подготовка данных» в произвольной форме (в ГОСТ нет ни пояснений, ни примеров), задает входные значения. Используется обычно для задания циклов со счетчиком. |
| flowcharts_connectorСоединитель | В случае, если блок-схема не умещается на лист, используется символ соединителя, отражающий переход потока управления между листами. Символ может использоваться и на одном листе, если по каким-либо причинам тянуть линию не удобно. |
| flowcharts_commentКомментарий | Комментарий может быть соединен как с одним блоком, так и группой. Группа блоков выделяется на схеме пунктирной линией. |

**Числовое программное управление** *CNC* — *компьютерное число с*труктурный в состав ЧПУ входят:

* **пульт оператора (или консоль ввода-вывода)**, позволяющий вводить управляющую программу, задавать режимы работы; выполнить операцию вручную. Как правило, внутри шкафа пульта современной компактной ЧПУ размещаются её остальные части;
* **дисплей (или операторская панель)** — для визуального контроля режимов работы и редактируемой управляющей программы/данных; может быть реализован в виде отдельного устройства для дистанционного управления оборудованием;

См. также: [Операторская панель](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BD%D0%B5%D0%BB%D1%8C)

* **контроллер** — компьютеризированное устройство, решающее задачи формирования траектории движения режущего инструмента, технологических команд управления устройствами автоматики станка, общим управлением, редактирования управляющих программ, диагностики и вспомогательных расчетов (траектории движения режущего инструмента, режимов резания);
* [**ПЗУ**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%BF%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) — память, предназначенная для долговременного хранения (годы и десятки лет) системных программ и констант; информация из ПЗУ может только считываться;
* [**ОЗУ**](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) — память, предназначенная для временного хранения управляющих программ и системных программ, используемых в данный момент.

В роли **контроллера** выступает [промышленный контроллер](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80), как то: [микропроцессор](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), на котором построена [встраиваемая система](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0); [программируемый логический контроллер](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80) либо более сложное устройство управления — [промышленный компьютер](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80).

Важной характеристикой CNC-контроллера является количество осей (каналов), которые он способен синхронизировать (управлять) — для этого требуется высокая производительность и соответствующее программное обеспечение.

В качестве исполнительных механизмов используются [сервоприводы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4), [шаговые двигатели](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%B3%D0%B0%D1%82%D0%B5%D0%BB%D1%8C).

Для передачи данных между исполнительным механизмом и системой управления станком обычно используется промышленная сеть (например, [CAN](https://ru.wikipedia.org/wiki/Controller_Area_Network), [Profibus](https://ru.wikipedia.org/wiki/Profibus), [Industrial Ethernet](https://ru.wikipedia.org/wiki/Industrial_Ethernet)).

Крупнейшие производители систем ЧПУ (по данным на 2009 год)[[4]](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5#cite_note-4):

* [Fagor Automation](https://ru.wikipedia.org/w/index.php?title=Fagor_Automation&action=edit&redlink=1) — 8037, 8055, 8060, 8065, 8070;
* [FANUC](https://ru.wikipedia.org/wiki/FANUC) — 0i-MD, 0i-TD, 0i-PD, 0i Mate-MD, 0i Mate-TD, 30i-MODEL B;
* [Fidia](https://ru.wikipedia.org/w/index.php?title=Fidia&action=edit&redlink=1) — nC 12R, nC 15, C10, C20, C40;
* [HAAS](https://ru.wikipedia.org/wiki/HAAS);
* [Heidenhain](https://ru.wikipedia.org/wiki/Heidenhain) — TNC 128, TNC 320, TNC 620, TNC 640, MANUALpus 620, CNC PILOT 640;
* [Mitsubishi Electric](https://ru.wikipedia.org/wiki/Mitsubishi_Electric) — C70, M70V, M700V;
* [Rexroth Bosch Group](https://ru.wikipedia.org/w/index.php?title=Rexroth_Bosch_Group&action=edit&redlink=1) — IndraMotion MTX micro, IndraMotion MTX standard, IndraMotion MTX performance, IndraMotion MTX advanced;
* [Siemens](https://ru.wikipedia.org/wiki/Siemens) — [Sinumerik](https://ru.wikipedia.org/w/index.php?title=Sinumerik&action=edit&redlink=1) 802D, 808D, 810D, 828D, 840D.

*ое управление*) — область техники, связанная с применением цифровых вычислительных устройств для управления производственными процессами[[1]](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5#cite_note-1).

Оборудование с ЧПУ может быть представлено:

* **станочным парком**, например, станками (станки, оборудованные числовым программным управлением, называются **станками с ЧПУ**) для обработки металлов (например, [фрезерные](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA) или [токарные](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%B0_%D1%81%D1%82%D0%B0%D0%BD%D0%BA%D0%BE%D0%B2)), дерева, пластмасс;
* [приводами](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%B8%D0%B2%D0%BE%D0%B4) асинхронных электродвигателей, использующих [векторное управление](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5);
* характерной системой управления современными [промышленными роботами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%80%D0%BE%D0%B1%D0%BE%D1%82);
* [Периферийные устройства](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B8%D1%84%D0%B5%D1%80%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), например: [3D-принтер](https://ru.wikipedia.org/wiki/3D-%D0%BF%D1%80%D0%B8%D0%BD%D1%82%D0%B5%D1%80), [3D-сканер](https://ru.wikipedia.org/wiki/3D-%D1%81%D0%BA%D0%B0%D0%BD%D0%B5%D1%80).

После того как составлена управляющая программа, оператор при помощи [программатора](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%BE%D1%80) вводит её в контроллер. Команды управляющей программы размещаются в ОЗУ. В процессе создания или после ввода управляющей программы оператор (в данном аспекте выполняющий функцию программиста) может отредактировать её, включив в работу системную программу редактора и выводя на дисплей всю или нужные части управляющей программы и внося в них требуемые изменения. При работе в режиме изготовления детали управляющая программа *кадр за кадром* поступает на выполнение. В соответствии с командами управляющей программы контроллер вызывает из ПЗУ соответствующие системные подпрограммы, которые заставляют работать подключенное к ЧПУ оборудование в требуемом режиме — результаты работы контроллера в виде электрических сигналов поступают на [исполнительное устройство](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) — приводы подач, либо на устройства управления автоматикой станка.

Управляющая система считывает инструкции [специализированного языка программирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (например, [G-кода](https://ru.wikipedia.org/wiki/G-code)) программы, который затем [интерпретатором](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80) системы ЧПУ переводится из входного языка в команды управления главным приводом, приводами подач, контроллерами управления узлов станка (например, включить/выключить подачу охлаждающей эмульсии).

Разработка управляющих программ в настоящее время выполняется с использованием специальных модулей для [систем автоматизированного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (САПР) или отдельных систем автоматизированного программирования ([CAM](https://ru.wikipedia.org/wiki/CAM)), которые по электронной модели генерируют программу обработки.

Для определения необходимой траектории движения рабочего органа в целом (инструмента/заготовки) в соответствии с управляющей программой используется [интерполятор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), рассчитывающий положение промежуточных точек траектории по заданным в программе конечным.

В системе управления, кроме самой программы, присутствуют данные других форматов и назначения. Как минимум, это *машинные данные* и *данные пользователя*, специфически привязанные к конкретной системе управления либо к определённой серии (линейке) однотипных моделей систем управления.

Программа для станка (оборудования) с ЧПУ может быть загружена с внешних носителей, например, магнитной ленты, [перфорированной бумажной ленты (перфоленты)](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%84%D0%BE%D0%BB%D0%B5%D0%BD%D1%82%D0%B0_(%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8)), [дискеты](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%B5%D1%82%D0%B0) или [флеш-накопителей](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B5%D1%88-%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) в собственную память либо временно, до выключения питания — в оперативную память, либо постоянно — в [ПЗУ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%BF%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), [карту памяти](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D0%B5%D1%88-%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) или другой накопитель: [жёсткий диск](https://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) или [твердотельный накопитель](https://ru.wikipedia.org/wiki/%D0%A2%D0%B2%D0%B5%D1%80%D0%B4%D0%BE%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BD%D0%B0%D0%BA%D0%BE%D0%BF%D0%B8%D1%82%D0%B5%D0%BB%D1%8C). Помимо этого, современное оборудование подключается к централизованным системам управления посредством [заводских (цеховых) сетей связи](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C).

Наиболее распространенный язык программирования ЧПУ для металлорежущего оборудования описан документом [ISO 6983](https://ru.wikipedia.org/w/index.php?title=ISO_6983&action=edit&redlink=1) Международного комитета по стандартам и называется «[G-код](https://ru.wikipedia.org/wiki/G-code)». В отдельных случаях — например, системы управления гравировальными станками — язык управления принципиально отличается от стандарта. Для простых задач, например, раскроя плоских заготовок, система ЧПУ в качестве входной информации может использовать текстовый файл в формате обмена данными — например, [DXF](https://ru.wikipedia.org/wiki/DXF) или [HPGL](https://ru.wikipedia.org/wiki/HPGL).

Лекция 4. проектирование микропроцессорных систем Управления полиграфическим оборудованием

4.1. Технология разработки и отладкимикропроцессорной системы (МПС) Уровни представления микропроцессорной системы. Микропроцессорная система может быть описана на различных уровняхабстрактного представления. Существующую микропроцессорную системуможно описать на любом известном уровне представления, но в начальнойстадии проектирования ее можно описать только на концептуальном уровне. В процессе разработки системы происходит переход от одного уровня еепредставления к другому, более детальному. Каждая абстракция несет в себе только информацию, которая соответствует данному уровню, и не содержит относительно более низких уровней. Какихлибо сведений Микропроцессорная система может быть описана, например, на одном изследующих уровней абстрактного представления:

1) «черный ящик»;

2) структурный;

3) программный;

4) логический;

5) съемный.

**Этапы проектирования микропроцессорных систем**

* 1. Формализация требований к системе
  2. Разработка структуры и архитектуры системы
  3. Разработка и изготовление аппаратных средств и программного обеспечения

**Отладка и приемисдаточные испытания**

Этап 1. На этом этапе составляются внешние спецификации,перечисляются функции системы, формализуется техническое задание (ТЗ)на систему, формально излагаются замыслы разработчика в официальнойдокументации.

Этап 2. На данном этапе определяются функции отдельных устройств ипрограммных средств, выбираются микропроцессорные наборы, на базекоторых будет реализована система.

Этап 3. После определения функций, реализуемых аппаратурой, ифункций, реализуемых программами, схемотехники и программистыодновременно приступают к разработке и изготовлению соответственноопытного образца и программных средств. Разработка и изготовлениеаппаратуры состоят из разработки структурных и принципиальных схем,изготовления образца, автономной отладки. Разработка программ состоит из разработки алгоритмов; написания текста исходных программ; трансляции исходных программ в объектные программы; автономной отладки.

Этап 4. Комплексная отладка предусматривает совместные работу, аппаратного и программного обеспечения МПС.

**Источники ошибок проектирования микропроцессорной системы.**

Рассмотрим источники ошибок на первых трех этапах проектирования.

Этап 1. На этом этапе источниками ошибок могут быть: логическая несогласованность требований, упущения, неточности алгоритма.

Этап 2. На данном этапе источниками ошибок могут быть: упущения функций, несогласованность протокола взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

Этап 3. На этом этапе источниками ошибок могут быть: при разработке аппаратуры - упущения некоторых функций, неверная интерпретация технических требований, недоработка в схемах синхронизации, нарушение правил проектирования; при изготовлении прототипа – неисправности комплектующих изделий, неисправности монтажа и сборки; при разработке программных средств - упущения некоторых функций технического задания, неточности в алгоритмах, неточности кодирования.

**Источники ошибок проектирования микропроцессорной системы.**

Рассмотрим источники ошибок на первых трех этапах проектирования.

**Этап 1**. На этом этапе источниками ошибок могут быть: логическая несогласованность требований, упущения, неточности алгоритма.

**Этап 2.** На данном этапе источниками ошибок могут быть: упущения функций, несогласованность протокола взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

**Этап 3.** На этом этапе источниками ошибок могут быть: при разработке аппаратуры - упущения некоторых функций, неверная интерпретация технических требований, недоработка в схемах синхронизации, нарушение правил проектирования; при изготовлении прототипа - неисправности комплектующих изделий, неисправности монтажа и сборки; при разработке программных средств - упущения некоторых функций технического задания, неточности в алгоритмах, неточности кодирования. Каждый из перечисленных источников ошибки может породить большое число субъективных или физических неисправностей, которые необходимо локализовать и устранить. следует из перечня источников ошибок, субъективные неисправности могут быть внесены на этапе разработки спецификации системы, а это означает, что даже после самых тщательных испытаний системы на соответствие ее внешним спецификациям в системе могут находиться субъективные

неисправности.

**Проверка правильности проекта. Основные методы контроля**

правильности проектирования следующие: верификация - формальные методы доказательства корректности проекта; моделирование; тестирование Существует много работ по верификации программного обеспечения, микропрограмм, аппаратуры. Однако эти работы носят теоретический характер. На практике пока используют моделирование поведения объекта и тестирование.

**Отладка микропроцессорных систем.** O правильности функционирования микропроцессорной системы на уровне «черного ящика» с полностью неизвестной внутренней структурой можно говорить лишь тогда, когда произведены ее испытания, в ходе которых реализованы все возможные комбинации входных воздействий, и в каждом случае проверена корректность ответных реакций. Однако исчерпывающее тестирование имеет практический смысл лишь для простейших элементов систем. Следствием этого является тот факт, что ошибки проектирования встречаются при эксплуатации, и для достаточно сложных систем нельзя утверждать об их отсутствии на любой стадии жизни системы. В основе почти всех методов испытаний лежит та или иная гипотетическая модель неисправностей, первоисточником которой служат неисправности, встречающиеся в практике. В соответствии с моделью в рамках каждого метода предпринимаются попытки создания тестовых наборов, которые могли бы обеспечить удовлетворительное выявление моделируемых неисправностей. Любой метод тестирования хорош ровно настолько, насколько правильна лежащая в его основе модель неисправности. Важным моментом является правильный выбор соотношения между степенью общности модели, стоимостью и степенью сложности формирования и прогона тестов, ориентированных на моделируемые неисправности. Чем конкретнее модель, тем легче создать для нее систему тестов, но тем выше вероятность того, что неисправность останется незамеченной. Если же модель неисправностей излишне общая, то из-за комбинаторного возрастания числа необходимых тестовых наборов и/или времени вычислений, требуемого для работы алгоритмов формирования тестов, она станет непрактичной и пригодной только для несложных систем.

**Обнаружение ошибки и диагностика неисправности**. Дефект не может быть обнаружен до тех пор, пока не будут созданы условия для возникновения из-за него неисправности, результат которой должен быть, в свою очередь, передан на выход испытуемого объекта, для того чтобы сделать неисправность наблюдаемой. Метод испытаний должен позволить генерировать тесты, ставящие испытуемый объект в условия, при которых моделируемые неисправности проявляли бы себя в виде обнаруживаемых ошибок. Если испытуемый объект предназначен для эксплуатации, то при обнаружении ошибки необходимо произвести локализацию неисправности с целью ее устранения путем ремонта или усовершенствования испытуемого объекта. Диагностика неисправности - процесс определения причины появления ошибки по результатам тестирования. Отладка - процесс обнаружения ошибок и определение источников их появления по результатам тестирования при проектировании микропроцессорных систем. Средствами отладки являются приборы, комплексы и программы. Точность, с которой тот или иной тест локализует неисправности, называется его разрешающей способностью. Требуемая разрешающая способность определяется конкретными целями испытаний. Например, при испытаниях аппаратуры в процессе эксплуатации для ее ремонта часто необходимо установить, в каком сменном блоке изделия имеется неисправность. В заводских условиях желательно осуществлять диагностику неисправности вплоть до уровня наименьшего заменяемого элемента, чтобы минимизировать стоимость ремонта

**Функции средств отладки**. Сроки и качество отладки системы зависят от средств отладки. Чем совершеннее приборы, имеющиеся в распоряжении инженера-разработчика, тем скорее можно начать отладку аппаратуры и программ и тем быстрее обнаружить ошибки, локализовать источники, устранение которых обойдется дороже на более позднем этапе проектирования.

Средства отладки должны:

1) управлять поведением системы или/и ее модели на различных уровнях абстрактного представления;

2) собирать информацию о поведении системы или/и ее модели, обрабатывать и представлять на различных уровнях абстракции;

3) преобразовывать системы, придавать им свойства контролепригодности;

4) моделировать поведение внешней среды системы, которая проектируется. В общем случае нельзя локализовать источник ошибки проектируемой системы, имея информацию о поведении системы только на ее внешних выводах, поэтому проектируемую систему преобразовывают. Например, прежде чем изготовлять однокристальную микроЭВМ с теми или иными «зашивками» ПЗУ, программы отлаживают на эмуляционном кристалле, у которого магистраль выведена на внешние контакты и вместо ПЗУ установлено Озу.

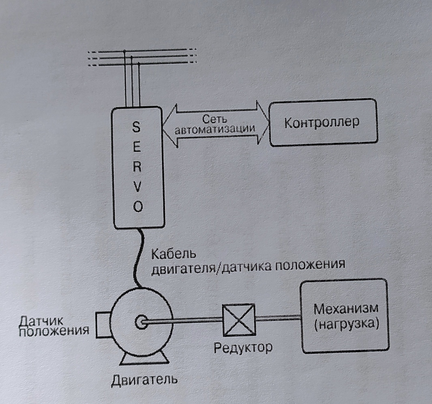
**Автономная отладка**. Процесс отладки прототипа проектируемой системы должен начинаться с отладки аппаратуры и отладки программ. Отладка аппаратуры предполагает тестирование отдельных устройств микропроцессорной системы - процессора, ОЗУ, контроллеров, блока питания, генератора тактовых импульсов путем подачи тестовых входных воздействий и приема ответных реакций. Тестовые входные воздействия иответные реакции определяются, исходя из спецификаций на устройства, а также структурных схем устройств. При этом проверяются реальная аппаратура прототипа, спецификации, структурные схемы и отлаживаются тесты. После отладки отдельных устройств проверяется их взаимодействие. Процессор системы работает с шинами адресов, данных и управления.

**Отладка программного обеспечения микропроцессорной системы.**

Отладка программ микропроцессорной системы проводится, как правило, на тех же ЭВМ, на которых велась разработка программ, и на том же языке программирования, на котором написаны отлаживаемые программы, и может быть начата на ЭВМ даже при отсутствии аппаратуры МПС. При этом в системном программном обеспечении ЭВМ должны находиться программы (интерпретаторы или эмуляторы), моделирующие функции отсутствующих аппаратных средств. Так, разработка и автономная отладка программных средств может вестись на больших ЭВМ, миниЭВМ, микроЭВМ, система команд которых не совпадает c системой команд используемого микропроцессора. Кроме того, при отладке программ может отсутствовать внешняя среда микропроцессорной системы, ее также необходимо моделировать.

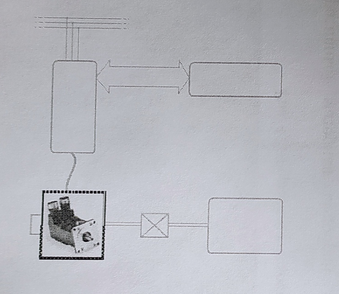
Проверка корректности программ, т. е. проверка соответствия их я внешним спецификациям, осуществляется тестированием. Программы проверяются на функционирование с различными исходными данными. Результаты функционирования программ сравниваются с эталонными значениями.

В пошаговом режиме программа выполняется по одной команде за один раз, а пользователь анализирует содержимое памяти, регистров и т. д., чтобы проверить, соответствуют ли результаты ожидаемым. Пошаговый режим ем может быть трудоемким, если средства отладки будут требовать отдельных команд после каждого шага для того, чтобы показать необходимую информацию в понятном для пользователя виде. Имеются средства отладки, автоматически показывающие после каждого шага содержимое регистров процессора и ячеек памяти, используемых в последней команде, и несколько следующих команд. Пошаговый режим является весьма мощным средством предварительного тестирования Кроме того, неоднократно проходя отдельными шагами через один и тот же участок объектной программы, программист может легко изменять содержимое регистров и ячеек памяти, особенно если средства отладки имеют динамически обновляемый дисплей, и тем самым проверить работу программы в разных условиях.



1. **Двигатель**
   1. **Общий обзор**

Двигатели преобразуют электрическую энергию в кинетическую или вращательную энергию. Они являются важной частью системы привода.



С начала исследований электромеханических энергетических установок в начале 19-ого столетия были разработаны 3 различны типа двигателя:

- двигатель постоянного тока. электрическая [машина постоянного тока](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%BF%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D1%82%D0%BE%D0%BA%D0%B0), преобразующая электрическую энергию [постоянного тока](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D1%8B%D0%B9_%D1%82%D0%BE%D0%BA) в [механическую энергию](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%8D%D0%BD%D0%B5%D1%80%D0%B3%D0%B8%D1%8F).

- синхронный двигатель

- асинхронный двигатель

Главное отличие синхронного от асинхронного двигателя заключается в устройстве ротора.

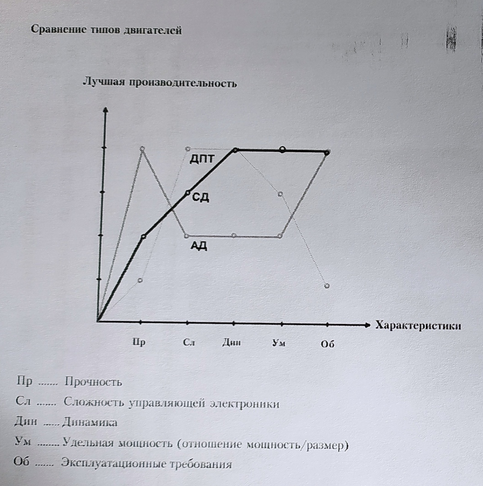
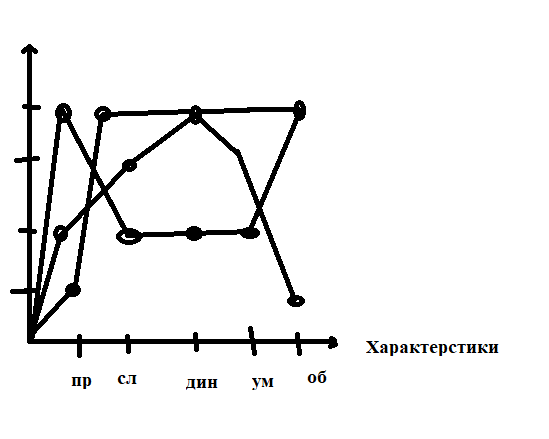
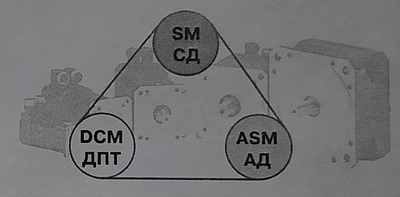
Роторы синхронных двигателей представляют собой постоянные или электрические магниты. Постоянное магнитное поле, создаваемое ими, взаимодействует с вращающимся магнитным полем статора.

Двигатели синхронной разновидности сложнее в использовании, поскольку они:

* в отличие от асинхронных моделей нуждаются в дополнительном источнике постоянного тока;
* подвержены более быстрому износу деталей (по причине использования контактных колец со щетками);
* требуют применения вспомогательных механизмов для запуска (индукционный двигатель имеет собственный пусковой момент).

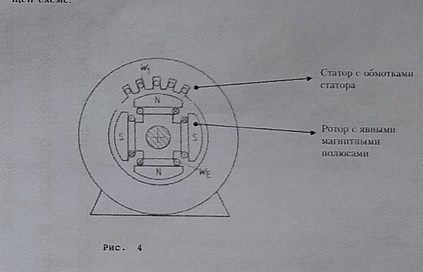
Для асинхронных моделей характерны:

* простота конструкции;
* надежность в эксплуатации.

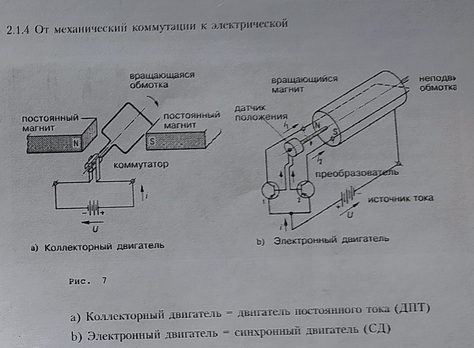
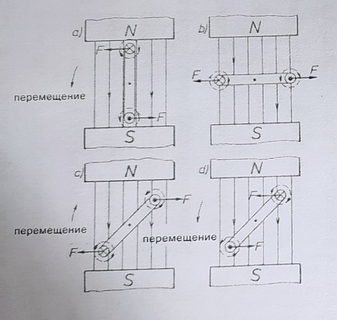
При этом синхронные двигатели обладают более широкими возможностями с точки зрения коэффициента мощности, а также менее чувствительны к перепадам напряжения, но стоимость таких агрегатов выше, что делает их использование менее выгодным

2.1.2 Конструкция синхронного двигателя

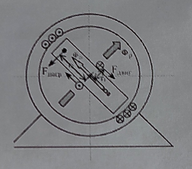
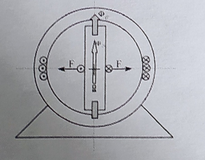
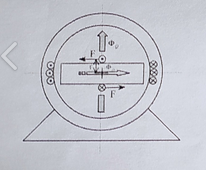
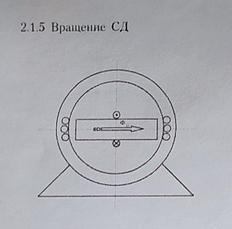
Основные элементы этого двигателя с вращающимся полем, в форме двигателя с внутренним полюсом (статор + ротор), показаны на следующей схеме:

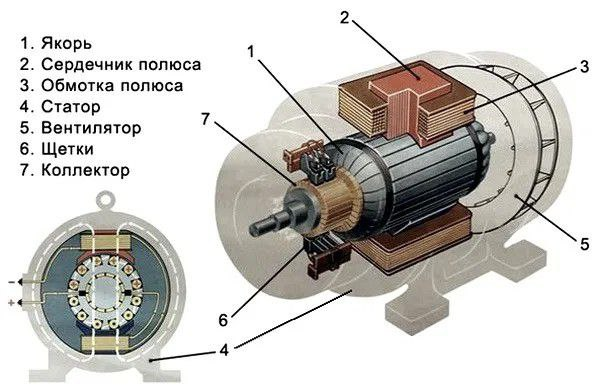


Обмотки статора соединены трехфазной звездой (L1, L2, L3). При подаче трехфазного электропитания обмотка статора вырабатывает вращающееся поле. Для двигателей средней производительности в роторе используются, главным образом, высококачественные постоянные магниты. Они производят поле ротора, соответствующее его позиции. Используя следующую формулу, можно показать, что сила, действующая щая на проводник на расстоянии г от геометрического центра (вала гателя) создает вращающий момент М между вектором силы и рот зависящий от угла альфа: M - 2 • F • r • sin a Для получения максимального момента угол должен равняться 90". иллюстрируется на следующей схеме:



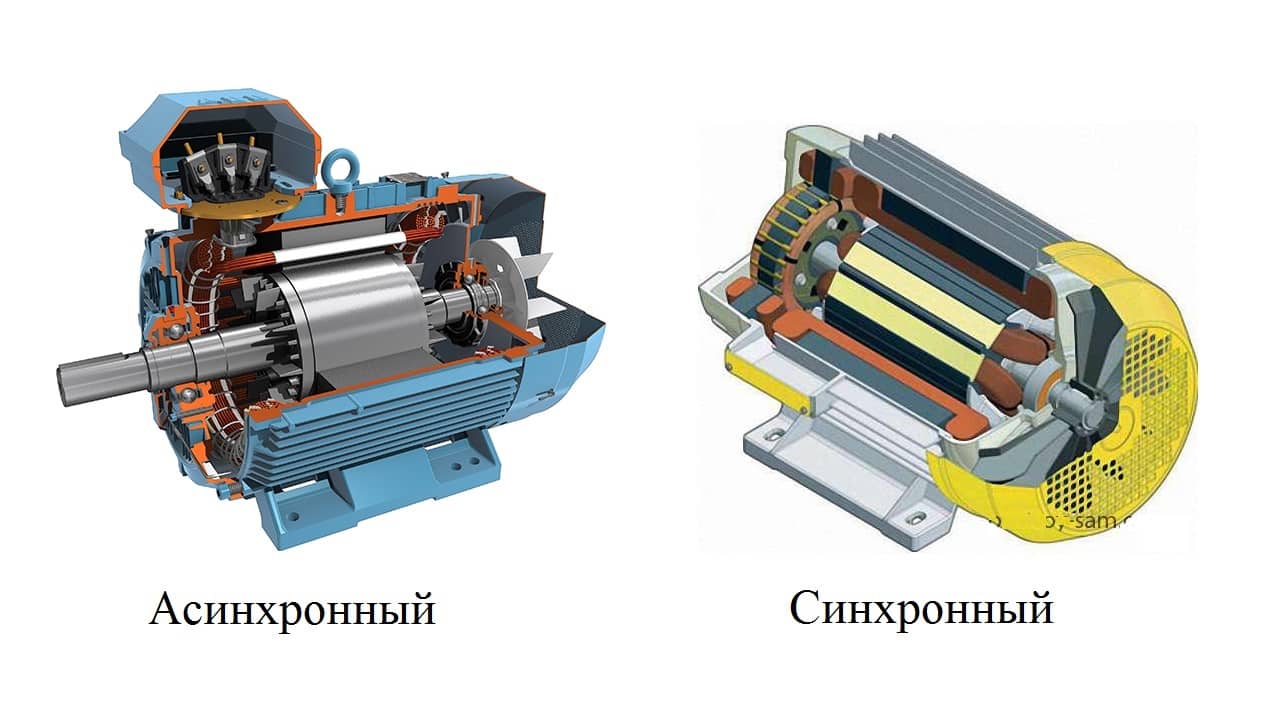
Чтобы показать последовательность вращения, предположим, что поле возбудителя ф, создается одной катушкой ротора, через которую протекает постоянным ток. При этом поле возбудителя постоянно, а его направление зависит от позиции ротора. Если ток протекает через некоторые иа обмоток статора, то создается поле статора Ф ф, проходит через ротор, и, согласно правилу левой руки, на проводник с током действует сила. Ротор переменщается возникающим вращающим моментом, Это перемещение заканчивается, когда магнитное поле ротора направлено так же, как магнитное поле статора. В этом случае результирующий врачщающий момент = 0, потому что силы взаимно компенсируются. Если к ротору приложена нагрузка, то он отклоняется из установленного положения, пока двигатель не создаст достаточный вращающий момент. Вращающий момент двигателя зависит от угла между фиф. Однако, мы хотим устранить отклонения между установленной и реальной позицииями. Для этого можно запитать обмотки статора так, чтобы поле ф, было перпендикулярно к полю ф используя ток I создающий вращающий момент двигателяля, который работает против нагрузки. Эта процедура (создание угла 90" между фиф ), позволяющая плучить максимальный вращающий момент, называется управлением по ориентации поля или векторным управлением.





Двигатель постоянного тока

ротор



статор

**1 вопрос: цифровые и цифроаналоговые преобразователи**

**Аналого-цифровой преобразователь**[[1]](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C#cite_note-1)[[2]](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C#cite_note-2)[[3]](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C#cite_note-3) (*АЦП*, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Analog-to-digital converter, ADC*) — устройство, преобразующее входной [аналоговый сигнал](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB) в дискретный код ([цифровой сигнал](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB)).

Обратное преобразование осуществляется при помощи [цифро-аналогового преобразователя](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE-%D0%B0%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) (ЦАП, DAC).

Как правило, АЦП — [электронное](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) устройство, преобразующее [напряжение](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BD%D0%B0%D0%BF%D1%80%D1%8F%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) в двоичный цифровой код. Тем не менее, некоторые неэлектронные устройства с цифровым выходом следует также относить к АЦП, например, некоторые типы [преобразователей угол-код](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D1%87%D0%B8%D0%BA_%D1%83%D0%B3%D0%BB%D0%B0_%D0%BF%D0%BE%D0%B2%D0%BE%D1%80%D0%BE%D1%82%D0%B0). Простейшим одноразрядным двоичным АЦП является [компаратор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BE%D1%80).

По способу применяемых алгоритмов АЦП делят на:

* [Последовательные прямого преобразования](https://ru.wikipedia.org/wiki/%D0%90%D0%A6%D0%9F_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)
* Последовательного приближения
* Последовательные с сигма-дельта-модуляцией
* Параллельные одноступенчатые
* Параллельные двух- и более ступенчатые (конвейерные)

АЦП первых двух типов подразумевают обязательное применение в своем составе [устройства выборки и хранения](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8_%D0%B8_%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) (УВХ). Это устройство служит для запоминания аналогового значения сигнала на время, необходимое для выполнения преобразования. Без него результат преобразования АЦП последовательного типа будет недостоверным. Выпускаются интегральные АЦП последовательного приближения, как содержащие в своем составе УВХ, так и требующие внешнее УВХ[[*источник не указан 2355 дней*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)].

### Линейные АЦП

Большинство АЦП считается [линейным](https://ru.wikipedia.org/w/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_(%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1), хотя аналого-цифровое преобразование, по сути, является нелинейным процессом (поскольку операция преобразования непрерывного пространства в дискретное — операция нелинейная).

Термин *линейный* применительно к АЦП означает, что диапазон входных значений, отображаемый на выходное цифровое значение, связан по линейному закону с этим выходным значением, то есть выходное значение *k* достигается при диапазоне входных значений от *m*(*k* + *b*) до *m*(*k* + 1 + *b*),

где *m* и *b* — некоторые константы. Константа *b*, как правило, имеет значение 0 или −0,5. Если *b* = 0, АЦП называют *квантователь с ненулевой ступенью* (*mid-rise*), если же *b* = −0,5, то АЦП называют *квантователь с нулём в центре шага квантования* (*mid-tread*).

### Нелинейные АЦП

Если бы [плотность вероятности](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8) амплитуды входного сигнала имела [равномерное распределение](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D0%B2%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), то отношение сигнал/шум (применительно к шуму квантования) было бы максимально возможным. По этой причине обычно перед квантованием по амплитуде сигнал пропускают через безынерционный преобразователь, передаточная функция которого повторяет [функцию распределения](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) самого сигнала. Это улучшает достоверность передачи сигнала, так как наиболее важные области амплитуды сигнала квантуются с лучшим разрешением. Соответственно, при цифро-аналоговом преобразовании потребуется обработать сигнал функцией, обратной функции распределения исходного сигнала.

Это тот же принцип, что и используемый в [компандерах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B4%D0%B5%D1%80), применяемых в магнитофонах и различных коммуникационных системах, он направлен на максимизацию [энтропии](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%8D%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%8F). (Не путать [компандер](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B4%D0%B5%D1%80) с [компрессором](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BF%D1%80%D0%B5%D1%81%D1%81%D0%BE%D1%80_%D0%B0%D0%BC%D0%BF%D0%BB%D0%B8%D1%82%D1%83%D0%B4%D1%8B&action=edit&redlink=1)!)

Например, голосовой сигнал имеет [лапласово распределение](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%9B%D0%B0%D0%BF%D0%BB%D0%B0%D1%81%D0%B0) амплитуды. Это означает, что окрестность нуля по амплитуде несёт больше информации, чем области с большей амплитудой. По этой причине логарифмические АЦП часто применяются в [системах передачи голоса](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B3%D0%BE%D0%BB%D0%BE%D1%81%D0%B0&action=edit&redlink=1) для увеличения динамического диапазона передаваемых значений без изменения качества передачи сигнала в области малых амплитуд.

8-битные логарифмические АЦП с [a-законом](https://ru.wikipedia.org/wiki/A-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD) или [μ-законом](https://ru.wikipedia.org/wiki/%D0%9C%D1%8E-%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD) обеспечивают широкий [динамический диапазон](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D0%B0%D0%BF%D0%B0%D0%B7%D0%BE%D0%BD_(%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) и имеют высокое разрешение в наиболее критичном диапазоне малых амплитуд; линейный АЦП с подобным качеством передачи должен был бы иметь разрядность около 12 бит.

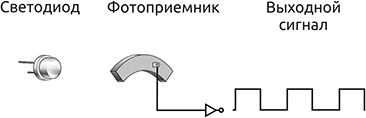
2 вопрос: вторичные системы управления, угловые повороты угловая скорость, световые сигналы

**Инкрементальный энкодер** — это устройство, преобразующее угловое перемещение вала в нормированный дискретный сигнал. Конструктивно энкодер состоит из измерительного преобразователя, защищенного корпусом и вала, передающего вращательное движение. Для защиты от внешних воздействий измерительной части предусмотрены уплотнения корпуса и вала, обеспечивающие защиту от внешних воздействий IP54.Со стороны вала расположен фланец с резьбовыми отверстиями (М3) для крепления скобы.

Внешний вид инкрементального энкодера OPKON PRI H

Внутри корпуса располагается фотоприемник (датчик) и взаимодействующий с ним специальный кодирующий диск. В качестве кодирующего диска используется лимб с нанесенными оптическими метками. Во время работы кодирующий диск приводится в движение валом и прерывает световой сигнал.

Таким образом, принцип действия энкодеров OPKON PRI основан на фотоэлектрическом эффекте: при попадании светового потока на фотоприемник, формируется сигнал, соответствующий логической единице, а при его отсутствии — логическому нулю.

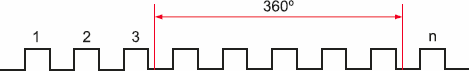


Прерывание светового потока осуществляется вращающимся кодирующим диском. Количество прерываний строго постоянно и соответствует количеству меток на кодирующем диске. В характеристиках энкодеров эта величина обозначается как разрешение — количество импульсов, поступающих за один оборот (имп/об, PPR, pulse per revolution). С помощью него можно определить дискретность энкодера — перемещение, соответствующее одному импульсу энкодера.

*Например, если использовать энкодеры с разрешением 100, 360 и 5000 имп/об, то один импульс будет соответствовать перемещению на 3.6°, 1°, 0.072° соответственно*.

Зная разрешение энкодера и считая количество импульсов (контроллером или счетчиком) можно определить:

1. Угол поворота (φ).  
   Подсчет количества импульсов (n), поступивших от энкодера, позволяет определить угол поворота (φ), совершенный валом.

Разрешениеимпоб

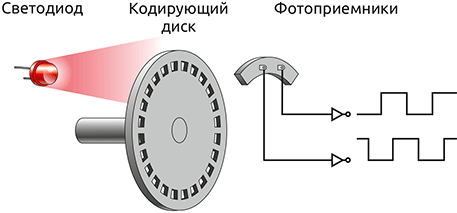
 *Пример: если энкодер выдал 1250 импульсов, а сам он имеет разрешение 5000 имп/об, то это означает, что вал совершил поворот на 90° (1250 \* 360 / 5000 = 90).*

 Угловую скорость (ω).  
Для определения скорости вращения (об/мин) необходимо подсчитывать количество импульсов, поступающих за единицу времени (например, за 1 секунду).

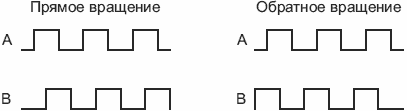
секРазрешениеимпоб

1. *Пример: если за 1 секунду поступило 1250 импульсов и энкодер также имеет разрешение 5000 имп/об, то скорость вращения вала — 15 оборотов в минуту (1250 \* 60 / 5000 = 15).*

Представленные примеры справедливы для механизмов одностороннего действия. Измерение вращения в двух направлениях невозможно, так как один импульсный сигнал не может дать информацию о том, в какой момент произошла смена направления. Для решения этой проблемы в энкодерах используется два фотоприемника со специальным фильтром (маской), формирующих два сигнала (сигнал А и сигнал В). Данная конструкция позволяет энкодеру выдавать два импульсных сигнала, сдвинутых друг относительно друга на ¼ периода (90°).



Смещение сигналов обеспечивает возможность определения направления в пределах одного импульса. При вращении в прямом направление сигнал А всегда будет опережать сигнал В, а при вращении в обратном — наоборот, первым будет сигнал В.

Последовательность импульсов при прямом и обратном вращении

Помимо сигналов А и В в энкодерах дополнительно используется сигнал Z. Он выдает единичный импульс, при прохождении кодирующим диском полного оборота (360°). Сигнал Z позволяет подсчитывать количество оборотов. Таким образом, каждый энкодер оборудован тремя каналами (ABZ) для передачи сигнала.

3 вопрос: програмное обеспечение для инентификациии объектов

# Инструменты OSINT: изображения и видео

[Блог компании Timeweb Cloud](https://habr.com/ru/company/timeweb/blog/) [Информационная безопасность \*](https://habr.com/ru/hub/infosecurity/)[Open source \*](https://habr.com/ru/hub/open_source/)[Обработка изображений \*](https://habr.com/ru/hub/image_processing/)[Научно-популярное](https://habr.com/ru/hub/popular_science/)

  
  
OSINT, Open-source intelligence — разведка по открытым источникам включает в себя поиск, выбор и сбор разведывательной информации из общедоступных источников, а также её анализ.  
Предлагаем вашему вниманию подборку инструментов для анализа изображений и видео. Теперь вы будете 100 раз задумываться, прежде чем выкладывать фотки с видом из окна или видео.

### InVID-WeVerify Расширение для Chrome и Firefox для проверки изображений и видео. С помощью расширения можно получать контекстную информацию о видео Facebook и YouTube, выполнять обратный поиск изображений в поисковых системах Google, Baidu или Яндекса, посмотреть метаданные, поискать по ключевым кадрам, узнать, кто первый загрузил видео или изображение в сеть, чтобы установить авторство. Этот плагин опенсорсный: [GitHub](https://github.com/AFP-Medialab/invid-verification-plugin%20/), по лицензии MIT.

Компьютерный анализ изображений является одним из самых мощных и широко распространенных методов экспериментальных наук. Решение сложных и разнообразных проблем, стоящих перед исследователями, зачастую требует достоверных опытных данных, для получения которых необходимы быстро работающие программно-аппаратные системы для автоматической локализации и идентификации объектов в изображениях.

Одна из важных задач, которая возникает в том или ином виде во многих областях науки - это быстрый поиск фрагментов в изображениях. Яркими примерами являются задачи подсчета количества бактерий на фотоснимке, поиска заданного участка местности на аэрофотоснимке, сравнение выпускаемой продукции с эталоном и т.д.

На данный момент в мире существует всего несколько программных продуктов, осуществляющих быстрый поиск образца на изображении для случая, когда образец внутри данного изображения может быть повернут. Еще меньше программ, позволяющих выполнить поиск образца, искаженного аффинным преобразованием.

Один из лидеров по времени поиска среди таких систем является продукт американской фирмы Datacube - MaxVision Toolkit [http://www.datacube.com], который обладает рядом неоспоримых достоинств: поиск образца, искаженного аффинным преобразованием, нелинейной коррекции освещенности. Но основным недостатком метода нормализованной корреляции оттенков серого, на котором основан продукт, является большая вычислительная сложность. Это заставляет исследователей искать новые подходы к решению данной задачи.

Нельзя не отметить вклад фирмы ABBY Software House в создание систем автоматического распознавания текста. Продукт FineReader является одним из лучших в мире для этой задачи. Одним из недостатком этого продукта является чувствительность к большим (больше 15 градусов) углам поворота сканируемого изображения. Таким же недостатком обладают различные системы локализации и распознавания этикеток и штрих кодов. На сегодняшний день требуются быстро работающие решения в контексте данной задачи.

Научная новизна.

В результате проведенных исследований реализован ряд конкурентно-способных программных систем для локализации и идентификации объектов в изображениях.

Предложен высокоэффективный алгоритм для решения задачи о локализации фрагмента в изображении, базирующийся на использовании множества опорных точек.

Программная система, созданная на основе этого метода, успешно локализует фрагменты, искаженные преобразованием масштаба и поворота. По времени локализации она не уступает лидерам в дайной области, а в некоторых случаях и превосходит их.

В задаче о локализации объектов в видео потоке применен метод «адаптивного фильтра» и метод фильтрации одномерных шумов, с помощью которых удалось построить систему устойчивую к небольшим резким изменениям угла зрения видеокамеры. Созданная система работает в режиме реального времени и по времени локализации не уступает своим зарубежным аналогам.

В рамках решения задачи о локализации маркера и распознавания содержащегося в нем текста, реализована система, которая успешно локализует маркеры специального вида, повернутые на произвольный угол, и имеющие произвольный размер. По времени локализации маркеров, искаженных преобразованием поворота, превосходит широко известную систему распознавания текста FineReader.

В задаче о восстановлении положения тела по информации, получаемой с множества одномерных видеокамер, предложен новый подход, основанный на использовании многомерного метода Ньютона.

В первой главе представлены алгоритмы и дано краткое описание программной системы для быстрого поиска фрагмента изображения в другом изображении. При этом фрагмент может быть повернут и/или его масштаб изменен.

Поиск фрагментов в изображениях — важная задача, которая в том или ином виде возникает во многих отраслях. В качестве примера можно привести подсчет количества бактерий на фотоснимке, поиск заданного участка местности на аэрофотоснимке и т. д.

Нередко задача осложняется тем, что искомый фрагмент может отличаться от образца расположением, цветовыми характеристиками, такими как яркость, контрастность, насыщенность цвета и др. Его геометрия может быть искажена, например, вытянута по какому-то направлению.