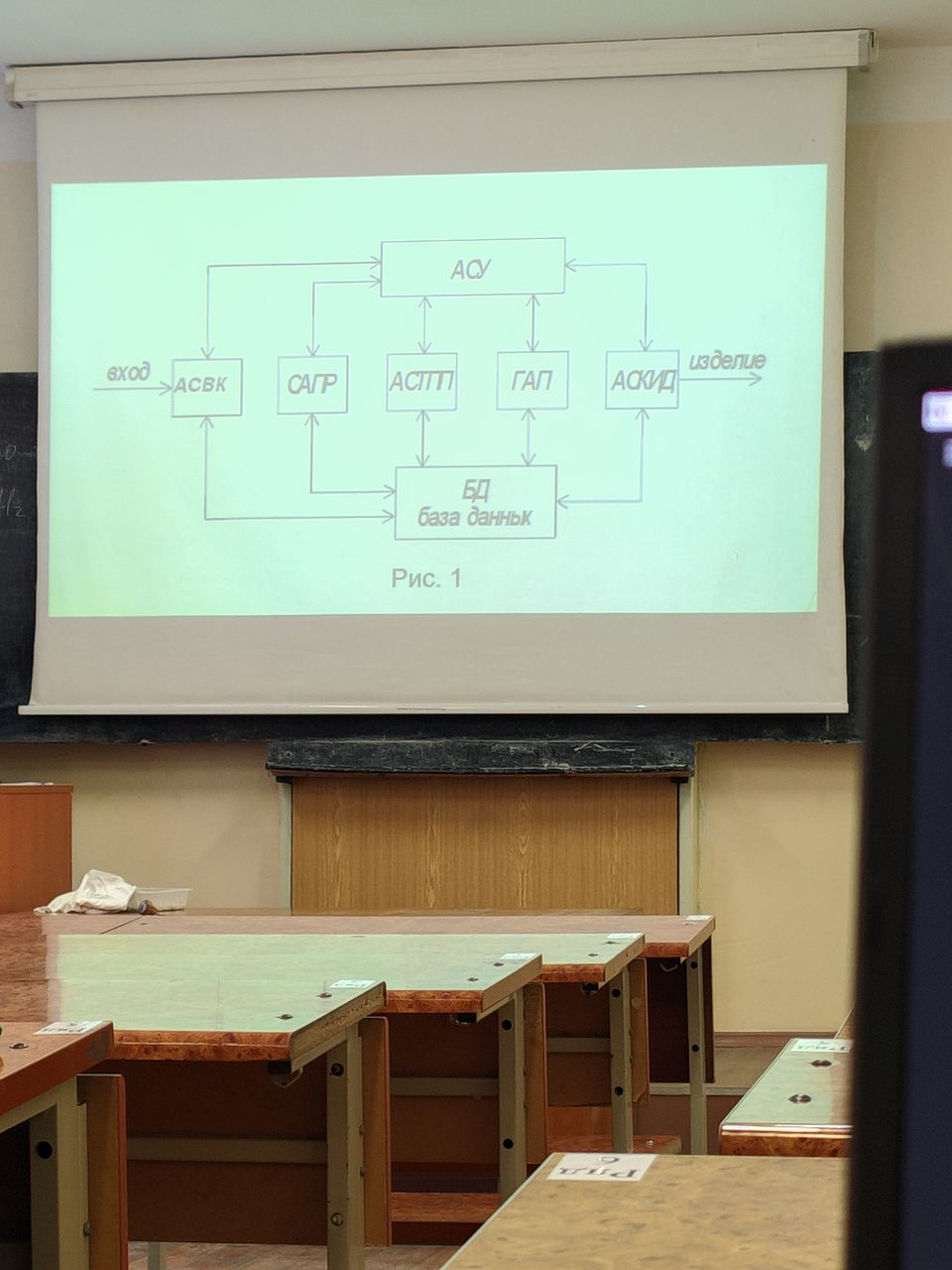
**Лк 1 Автоматизация. Определение, терминология**

* Автоматическая эксплутационная диагностика
* Надежность оборудования

PP41 - микроконтроллер

ГАП - гибкие автоматизированные системы - предназначены для автоматизации тех процессов со сменным объектом производства, в том числе для единичного и мелкосерийнгого производства. Включает в себя основное и вспомагательное оборудование работающее от одной СУ (системы управления)



Начальным этапом ялв-ся разработка подсистемы АСВК(Входной контроль), САПР, АСТПП (технология), ГАП, АСКИД (контроль и измерение детали). Основной элемент при создании такой подсистемы - унификация банка данныз для информативного обеспечения функционирования подсистемы. Унификация банка данных предполагает прежде всего применение групповой технологии, технологических процессов обр-ки деталей объединенных рядом технологических и конструктивных признаков

В состав ГАП включены подсистемы: обработки, манипулирования, контроля и управления. Проектирование, разработка и внедрение осуществляется в установленном нормами и правилами порядке

**Классификация СУ**

* по принципу синзронизации
* по степени централизации
* по методу воздействия
* по числу управляемых координат
* по виду программоносителя
* по наличию или отсутствию обратной связи

Особенности централизованной СУ

* продолжительность рабочего цикла для каждого исполнительного органа, является как правило величиной постоянной;
* простота схем управления
* надежность в работе
* удобство обслуживания и наладки
* необходимость иметь дополнительные предохранительные устройства

Особенности децентрализованной системы управления

* отсутсвие сложной блокировки
* датчики расположены в рабочей зоне оборудования

Наиболее важным и характерным прищнаком любой СУ программного управления является способ задания программы обработки или движения, т.е программоносителя

* СУ копирами
* СУ с электронными кулачкками
* Система циклового программного управления (упоры, коммутаторы, штекерное табло)
* Система числовоого программного управления

**ЧПУ**

В зависимости от решаемых технологических задач системы ЧПУ делятся на 4 вида:

* позиционные управления положением
* прямоугольные системы управления рабочими перемещениями
* непрерывные контурные СУ
* комбинированные СУ

**Лк 2 Надёжность ПО автоматизированных систем управления технологического процесса**

OPC - OLE для управления процессами. Ключевыми словами для понимания изложения являются технологии Microsoft OLE (Object Linking and Embedding - связывание и встраивание объектов) и интеграция

COM/DCOM - Component Object Model (модель составных объектов)

Распределенная COM - это технология, введенная первоначально Microsoft для интеграции между различными офисными приложениями в Windows

К COM добавляются компоненты связанные с безопасностью

**Объекты COM**

Модель COM оперирует объектами, очень похожими на объекты в ОО ЯП, например C++

**Интерфейсы объектов**

* объединяет группу взамосвязанныз функций, предоставляемых объектом
* особенность - публичность
* используются после того, как опубликованы - после этого их никогда нельзя изменить

**Доступ к объектам**

* Указатель на интерфейс - то, с чем работает вызывающий процесс
* Чтобы получить указатель на интерфейс , нужно воспользоваться функцией QueryInterface
* Указатель на интерфейс передается при вызове объекта

**Обмен в COM**

Объект в COM - пассивная сторона, он предоставляет через интерфейсы свои функции, в этом смысле употребляется термин COM-сервер

Заправшиваемая программа - COM-клиент

OPC - сервер может поставлять данные “по подписке”, то есть сам инициализировать обмен с OPC - клиентами при их обновлении

**Регистрация**

Каждый объект при регистрации имеет уникальный идентификатор, называется GUID - Globally Unique Identifier. Регистрация деает доступной информацию о расположении объектов всем программам

**Обслуживание объектов**

Даже на одоном компьютере программы работают в своих адресных пространствах - это означает, что требуется кто-то , кто будет передавать вызовы из одного процесса в другой

**Удаленные объекты**

DCOM - расширение COM, которое позволяет добираться до объектов на другиз компьютерах. DCOM - системный сервис, делающий COM прозрачным в локальных сетях.

**Предоставление объектов**

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

**Использование объектов**

Если используем C++, для описания интерфейсов используюся .c и .h файлы. В этом случае говорят об Custom интерфейсах

Либо мы используем для скриптовых запросов так называемую автоматизацию. В этом случает используется функция IDispatch

**Реализация объектов**

Программирование COM - занятие не из легких. С помощью C-подобного языка MIDL описываются интерфейсы. С помощью компилятора MIDL.exe преобразуются в описанные выше файлы.Далее используется библиотека ATL

**Спецификации**

* OPC Common definitions and Interfaces - общие для всех OPC спецификаций интерфейсы
* Data Access Custom Interface Standard - спецификация COM интерфейсов для обмена оперативными данным, программирование на C++
* Data Access Automation Interface Standard - спецификация COM-интерфейсов для обмена оперативными данными, программироваие на языках по типу VB
* OPC Batch Custom Interface Specification - спецификация COM-интерфейсов конфигурирования оборудования, программирование на C++
* OPC Batch Automation Interface Specification - спецификация COM-интерфейсов для конфигурирования оборудования на языках типа VB
* OPC alarm and events interface specification - спецификация COM-интерфейсов для обслуживания событий и нештатных ситуаций, программирование на C++
* Historical Data Access Custom Interface Standard - спецификация COM-интерфейсов для работы с хранилищами данных, прогрмаммирование на C++
* OPC Security Custom Interface - спецификация COM-интерфейсов для обработки прав доступа к данным, программирование на C++

**OPC сервер**

**OPC Data Access**

стандарт, сокращенно DA

Предназначен для поставки оперативных данных от оборудования и/или к обородуванию. Для стандарта

**Данные**

Основной единицей данных в OPC является переменная (Item). Переменная может быть любого типа, допустимого в OLE: различные целые и вещественные типы, логический тип, строковый и т.д. Кроме того, переменная может быть массивов.

**Лк 3 Надёжность программного обеспечения АСУТП**

Одной из основных частей АСУ ТП чвлчется ПО, представляющее собой сов-ть взыимосвязанных и автономных программ, описаний, инструкций программиста м пользователя.

Надежность ПО есть свойство программного обеспечения своевременно выполнять в заранее указанных условиях эксплуатации вперед установленные функции.

Функция ПО АСУТП - своевременное получение некоторого результата или решения y при переработке входной …..

Решение о выполнении или невыполнении функций ПО вынужден принимать пользователь и, в меньшей степени разработчик программы.

Надежность устанавливается по результатам работы ПО, т.е при динамической проверке всех программ на множестве входной информации. - это определение надежности базируется на понятии отказа.

Отказы

* Случайные - наюлюдаются в случайные моменты времени работы УВМ или процессора.
* Неслучайные - обусловлены действием так называемых компьютерных вирусов

Сбой - случайное событие заключается в появлении неразумного результата.

Сбой ПО - это самоустраняющийся отказ программы, возникающий при некоторых, возможно случайных состояниях УВМ.

* Устойчивый отказ - в случайный момент процессорного времени в форме “неразумного” результата
* Причина отказа ПО - систематическая ошибка программы.

Ошибки бывают первичные и вторичные

| **Первичные** | **Вторичные** |
| --- | --- |
| Неточность в текстах программ, алгоритмах, возникают при подготовке носителей и документации ПО | Следствие ошибок вторичного типа |

Вторичные ошибки

* Вычислительные - неверная индексация и подсчет времени
* Логические
* Сопряжения интерфейсов

Ошибки первичного и вторичного типов порождаются на этапах разработки спецификаций на ПО, проектировани ПО, реализации программ

Отказы ПО при его эксплуатации имеет ряд отличий от отказов тех. элементов

* Отказ ПО не приводит к разрушению или поломке программного элемента
* Отказ ПО не коррелирован с процессорным и, тем более, астрономическим временем
* При длительном использовании программы становятся абсолютно устойчивыми

При устранении одних ошибок вносятся другие.

Основные показатели надежности ПО

* функция надежности или отказа
* функция надежности
* интенсивность отказов
* средняя наработка на отказ ПО

Вероятность безотказной работы программного комплекса

Понижению интенсивности лямбда способствует и перезод на другой более высокий ЯП

Другой путь повышения надежности связан с резервированием и повышением избыточности

Есть три вида резервирования

* временное - многократный прогон одних и тех же ненадежных программ и сравнении результатов расчета
* информационное -дублирование исходных и промежуточных данных
* программное - два и более ПО для получения одного и того же результата или выполнения одной и той же функции

# ****Лк 4 Наиболее важные элементы Automation Studio****

По pdf документу:

<https://s3.us-west-2.amazonaws.com/secure.notion-static.com/fca56a56-b68d-4e28-8f47-d2708d5a3e64/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_13.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-Credential=AKIAT73L2G45EIPT3X45%2F20230320%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-Date=20230320T092107Z&X-Amz-Expires=86400&X-Amz-Signature=8581521fb191d6dc2b13d95f30e9f001449a7dbdd8744e267d3a5c849c419d53&X-Amz-SignedHeaders=host&response-content-disposition=filename%3D%22Automation%2520Studio.pdf%22&x-id=GetObject>

**Лк 5 Проектирование микропроцессорных систем управления полиграфическим оборудованием**

Уровни представления микропроцессорной системы. Микропроцессорная система может быть описана на различных уровнях абстрактного представления. Существующую микропроцессорную систему можно описать на любом известном уровне представления, но вначальной стадии проектирования ее можно описать только на концептуальном уровне. Впроцессе разработки системы происходит переходот одного уровня ее представления к другому,более детальному. Каждая абстракция несет в себетолько информацию, которая соответствуетДанному уровню, и не содержит каких-либосведений относительно более низких уровней.

**Микропроцессорная система** может быть описана, например, на одном из следующих уровней абстрактного представления:

1. “черный ящик”
2. структурный
3. программный
4. логический
5. системный

* На уровне **“черного ящика”** микропроцессорная системаописывается внешними спецификациями: перечисляютсявнешние характеристики.
* **Структурный уровень** создается компонентами микропроцессорной системы: микропроцессорами, запоминающими устройствами, устройствамиввода/вывода, внешними запоминающими устройствами,каналами связи. Микропроцессорная система описываетсяфункциями отдельных устройств и их взаимосвязью, атаюке информационными потоками.
* **Программный уровень** разделяется на два подуровня: *команд процессора* и *языковой*. Микропроцессорнаясистема интерпретируется как последовательностьоператоров или команд, вызывающих то или иноедействие над некоторой структурой данных.
* **Логический уровень** присущ исключительно дискретным системам. На этом уровне выделяются дваподуровня: переключательных схем и регистровых пересылок.
* **Подуровень переключательных схем** образуется вентилями и построенными на их основеоператорами обработки данных. Переключательные схемы подразделяются на комбинационные ипоследовательностные; первые в отличие от последнихне содержат запоминающих элементов. Поведение системы на этом уровне описывается алгеброй логики,моделью конечного автомата, входными/выходными последовательностями 1 и 0. Комбинационные схемы представляются таблицей истинности, в которойкаждому входному набору значений сигналов ставится всоответствие набор значений сигналов на выходах
* **Последовательностные схемы** могут описываться диаграммами или таблицами входов/выходов, вкоторых определены взаимно однозначные соответствия между входами схемы, внутреннимисостояниями (комбинациями значений элементовпамяти) и выходами. Подуровень регистровыхпересылок характеризуется более высокойстепенью абстрагирования и представляет собойописание регистров и передачу данных между ними.Он включает в себя две части: информационную и управляющую. Информационная часть образуетсярегистрами, операторами и путями передачиданных. *Управляющая часть* определяетзависящие от времени сигналы, инициирующиепересылку данных между регистрами.
* **Схемный уровень** образуется резисторамии конденсаторами. Показателями поведениясистемы на этом уровне служат напряжениеИ ТОК, Представляемые в функции времениили частоты. Этот уровень описаниядискретной системы широко используется вописаниях аналоговых систем и не являетсяни наинизшим из возможных, нидостаточным для полной характеристикисистемы.

**Этапы проектирования микропроцессорных систем**. Микропроцессорные системы по своей сложности, требованиям и функциям могут значительно отличаться надежностными параметрами, объемом программных средств, быть однопроцессорными и многопроцессорными,построенными на одном типе микропроцессорного набора или нескольких, и т. д. В связи с этим процесс проектирования может видоизменяться в зависимости от требований, предъявляемых к системам. Например, процесс проектирования МПС, отличающихся одна от другой содержанием ПЗУ, будет состоять из разработки программ и изготовления ПЗУ.

При проектировании многопроцессорных микропроцессорных систем, содержащих несколько типов микропроцессорных наборов, необходимо решать вопросы организации памяти, взаимодействия с процессорами, организации обмена между устройствами системы и внешней средой, согласования функционирования устройств, имеющих различнуюскорость работы, и т. д. **Ниже приведена примерная последовательность этапов, *типичных для создания микропроцессорной системы*:**

1. Формализация требований к системе.
2. Разработка структуры и архитектуры системы.
3. Разработка и изготовление аппаратных средств и программного обеспечения системы.
4. Комплексная отладка и приемосдаточные испытания

***Этап 1.*** На этом этапе составляются внешние спецификации, перечисляются функции системы, формализуется техническое задание (Т3) на систему, формально излагаются замыслы разработчика в официальной документации.

***Этап 2.*** На данном этапе определяются функцииотдельных устройств и программных средств,выбираются микропроцессорные наборы, на базекоторых будет реализована система,определяются взаимодействие междуаппаратными и программными средствами,временные характеристики отдельных устройств и программ.

***Этап 3.*** После определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты одновременно приступают к разработке и изготовлению соответственно опытного образца и программных средств. Разработка и изготовление аппаратуры 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. На каждом этапе проектирования МПС людьми могут быть внесены неисправности и приняты неверные проектныерешения. Кроме того, в аппаратуре могут возникнуть дефекты.

**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.

***Этал 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. Обнаружение ошибки и локализация неисправности являются сложной задачей по нескольким причинам: во-первых, из-за большого числа неисправностей; во-вторых, из-за того, что различные неисправности могут проявляться одинаковым образом. Так как отсутствуют модели субъективных неисправностей, указанная задача не формализована. Имеются определенные успехи в области создания методов и средств обнаружения ошибок и локализациифизических неисправностей. Эти методы и средства широко используются для проверки работоспособного состояния и диагностики неисправностей дискретных систем при проектировании, производстве и эксплуатации последних.

***Субъективные неисправности*** отличаются от *физических* тем, что после обнаружения, локализации и коррекции больше не возникают. Однако, как следует из перечня источников ошибок, субьективные 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.

**Процесс проектирования** - итерационный процесс. Неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции спецификаций, а следовательно, к началу проектирования всей системы. Обнаруживать неисправности необходимо как можно раньше, для этого надо контролировать корректность проекта на каждом этапе разработки.

**Проверка правлиьности 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.

Для контроля корректности проекта на каждом этапе проектирования необходимо проводить моделирование на различных уровнях абстрактного представления системы и проверку правильности реализации заданной модели путем тестирования. На этапе формализации требований контроль корректности особо необходим, поскольку многие цели проектирования не формализуются или не могут быть формализованы в принципе. Функциональная спецификация может анализироваться коллективом экспертов или моделироваться и проверяться в опытном порядке для выявления, достигаются ли желаемые цели. После утверждения функциональной спецификацииначинается разработка функциональных тестовыхпрограмм, предназначенных для установленияправильности функционирования системы в соответствиис ее функциональной спецификацией.

**Отладка микропроцессорных систем.**

О правильности функционирования микропроцессорной системы на уровне «черного ящика» с полностью неизвестной внутренней структурой можно говорить лишь тогда, когда произведены ее испытания, входе 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.

**Обнаружение ошибки и диагностика неисправности.**

Дефект не может быть обнаружен до тех пор, пока небудут созданы условия для возникновения из-за негонеисправности, результат которой должен быть, в свою очередь, передан на выход испытуемого объекта, длятого чтобы сделать неисправность наблюдаемой. Методиспытаний должен позволить генерировать тесты,ставящие испытуемый объект в условия, при которыхмоделируемые неисправности проявляли бы себя ввиде обнаруживаемых ошибок. Если испытуемыйобъект предназначен для эксплуатации, то приобнаружении ошибки необходимо произвестилокализацию неисправности с целью ее устраненияпутем ремонта или усовершенствования испытуемогообъекта.

**Диагностика неисправности** - процесс определенияпричины появления ошибки по результатам тестирования.

**Отладка** - процесс обнаружения ошибок и определение источников их появления по результатам тестирования припроектировании 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.

**Лк 6 Организация обеспечения АСУ**

Структура обеспечения АСУ

1. Организационное обеспечение – выработка общей концепции функционирования АСУ, разработка структуры системы, правил взаимодействия между ее подсистемами и отдельными элементами, обеспечивающих ее устойчивое и эффективное функционирование. Организационное обеспечение АСУ вводится как на действующих предприятиях путем модернизации, так и на новых.

Процесс создания АСУ включает в себя следующие стадии:

1. Научно-исследовательские и опытно-конструкторские работы. Выработка общей концепции функционирования АСУ и формирования состава задач по управлению, которые разделяются по уровню иерархии объекта управления предприятием, цеха или участка.

2. Предпроектные работы. В процессе предпроектных и проектных работ главное внимание должно быть уделено проверка экономической эффективности спроектированных систем.

3. Изготовление АСУ, наладка и ввод ее в эксплуатацию.

2. Информационное обеспечение – совокупность директивных данных памяти, воплощённой нормативной справочной информации, слежения (сбора оперативной информации по учету и контролю), документооборот, кодирования и шифровка данных, технологическая схема обработки данных.

Сбор учетных данных, представляющий собой информационные потоки о состоянии объекта, может осуществляться тремя способами:

1. Автоматический режим – данные о состоянии объекта фиксируются с помощью датчиков и непосредственно вводятся в ЭВМ.

2. Автоматизированный режим – данные оцениваются оператором и вводятся в ЭВМ, но анализируется оператором.

3. Неавтоматизированный режим – данные считываются и анализируются оператором.

3. Алгоритмическое обеспечение

В процессе функционирования АСУ ТП управляющий вычислительный комплекс может решать следующие задачи:

1. Сбор данных по параметрам объекта, их обработка и регистрации.

2. Определение параметров процесса или объекта.

3. Анализ режима протекания процесса с целью выработки рекомендаций для управления или команд для непосредственного управления.

Нормальное функционирование АСУ ТП возможно в том случае, когда при решении указанных выше задач отдельные операции выполняются по определенным правилам и в необходимой последовательности.

Алгоритм – это предписание, определяющее содержание и последовательность выполнения операций, переводящих исходные данные в требуемый результат. Алгоритм представляется в виде схемы последовательности действия на входящую информацию.

4. Программное обеспечение.

Программа – данные, используемые для управления ЭВМ при обработке информации в соответствии с принятым алгоритмом управления.

Программирование – это процесс составления упорядоченных последовательностей действий (программ) для ЭВМ.

Программное обеспечение управляющего вычислительного комплекса – совокупность программ систем обработки информации и документов, необходимых для эксплуатации этих программ.

Программное обеспечение делится на:

- общее (системное)

Это математическое обеспечение, разрабатываемое одновременно с проектированием ЭВМ.

- специальное (функциональное)

В состав операционной системы входит специальный язык, файловая система и драйверы. Файловая система представляет собой хранилище программ и данных.

5. Технологическое обеспечение

Технологическое обеспечение – совокупность технологической документации, основу которой составляют инструкции для каждого пользователя и обслуживающего персонала. Кроме инструкций, в состав технологического обеспечения входят рекомендуемые технологические схемы прохождения заказов в производства, журналы учета прохождения заказов, документы, фиксирующие строки и исполнителей заказов на каждом этапе технологического процесса.

Доп. задания: написать про системы счисления, нарисовать блок-схему уравнения y = cosX

**Системы счисления**

* двоичная (0,1)
* восьмеричная (0,1,2,3,4,5,6,7)
* десятичная (0,1,2,3,4,5,6,7,8,9)
* шестнадцатеричная (цифры 0,1,2,3,4,5,6,7,8,9 и буквы A,B,C,D,E,F)
* двоично-десятичная (как десятичная, но каждая цифра числа записывается его двоичным аналогом)

**Блок-схема функции y = cos(x):**



# Лк 7 Проектирование программного обеспечения полиграфических машини систем обработки информации

## Синтез алгоритмов управления исполнительными системами

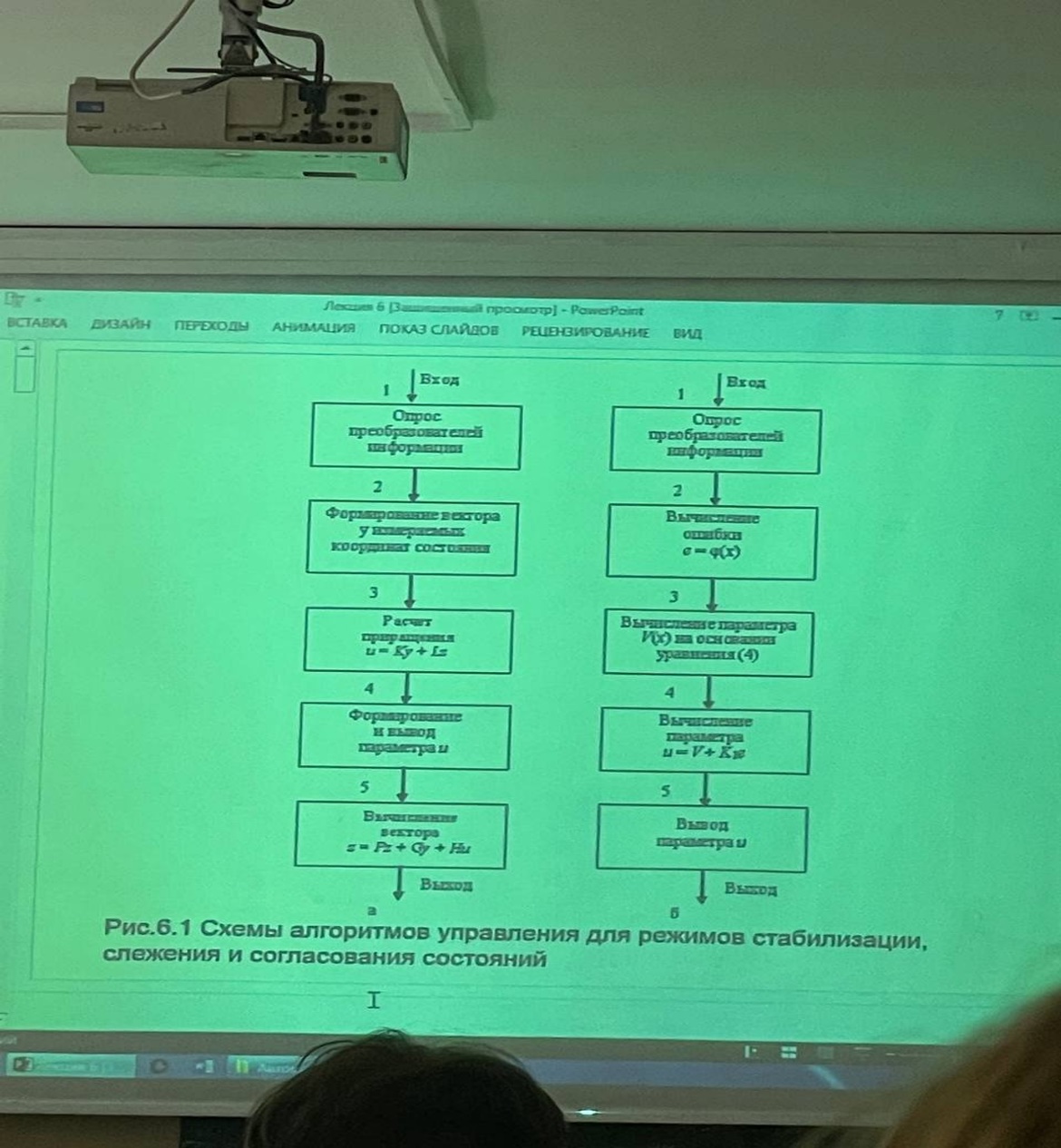
Новая апппаратная база - микро-ЭВМ контроллеры — позволяет

реалищзовать в системах автоматического управления отпимаольные законы управления и этим обеспечивает их более высокие статические и динамические показатели.

Типовыми режимами работы электропривода является разгон, тороможение, реверс; стабилизация определенных параметров при сильно изменяющихся внешних воздействиях; слежение в том числе программное, при наличии внешних воздействий; согласование конечных состояний ркжимов пуска - тороможения с начльным состоянием режимов стабилизации или слежения

Наивысшее качество процессов может быть достигнуто в том случае, когда для каждого режима работы электропривода используется регулятор соответствующей структуры, настройка которого зависит от изменений параметров объекта управления и внешних воздействий.

Рассмотрим возможные



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 1 этого алгоритма опрашиваются преобразователи информации, используемые в системе управления и измеряющие состояние электропривода и внешних воздействий. Формирования вектора у в 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 воздействия, рассчитанное в блоке 3 на предыдущем такте. Такое расширение вектора измеренных координат позволяет компенсировать запаздывание, обусловливаемое управляющей ЭВМ. Это запаздывание равно времени выполнения расчетов в блоках 1-4.

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 с являются матрицами-строками, если электродвигатель управляется по одному входу (по цепи якоря или по цели возбуждения), и имеют две строки, если электродвигатель управляется по двум входам. Выбором матриц K и L обеспечиваются заданные свойства собственного движения системы управления электроприводом, а также компенсируются доминирующие возмущающие воздействия. Для расчета матриц K 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. Это оценка используется для расчета управляющего воздействии в следующем такте. Выбором матриц P, G и H можно обеспечить требуемую скорость сходимости оценки, фильтрацию ошибок измерения, экстраполяцию отсчетов, снимаемых с преобразователей информации, а также требуемый порядок астелома системы.

Одновременно требуется обеспечить выполнение некоторых соотношений координат вектора состояния (фазовых координат хрида ф(х) = 0, определяющих, например, условия постоянства Настоты вращения двигателя, мощности т. п. в пространстве состояний рассматриваемое уравнение задает некоторое многообразие, в частности поверхность, отклонение от которого характеризуется ошибкой

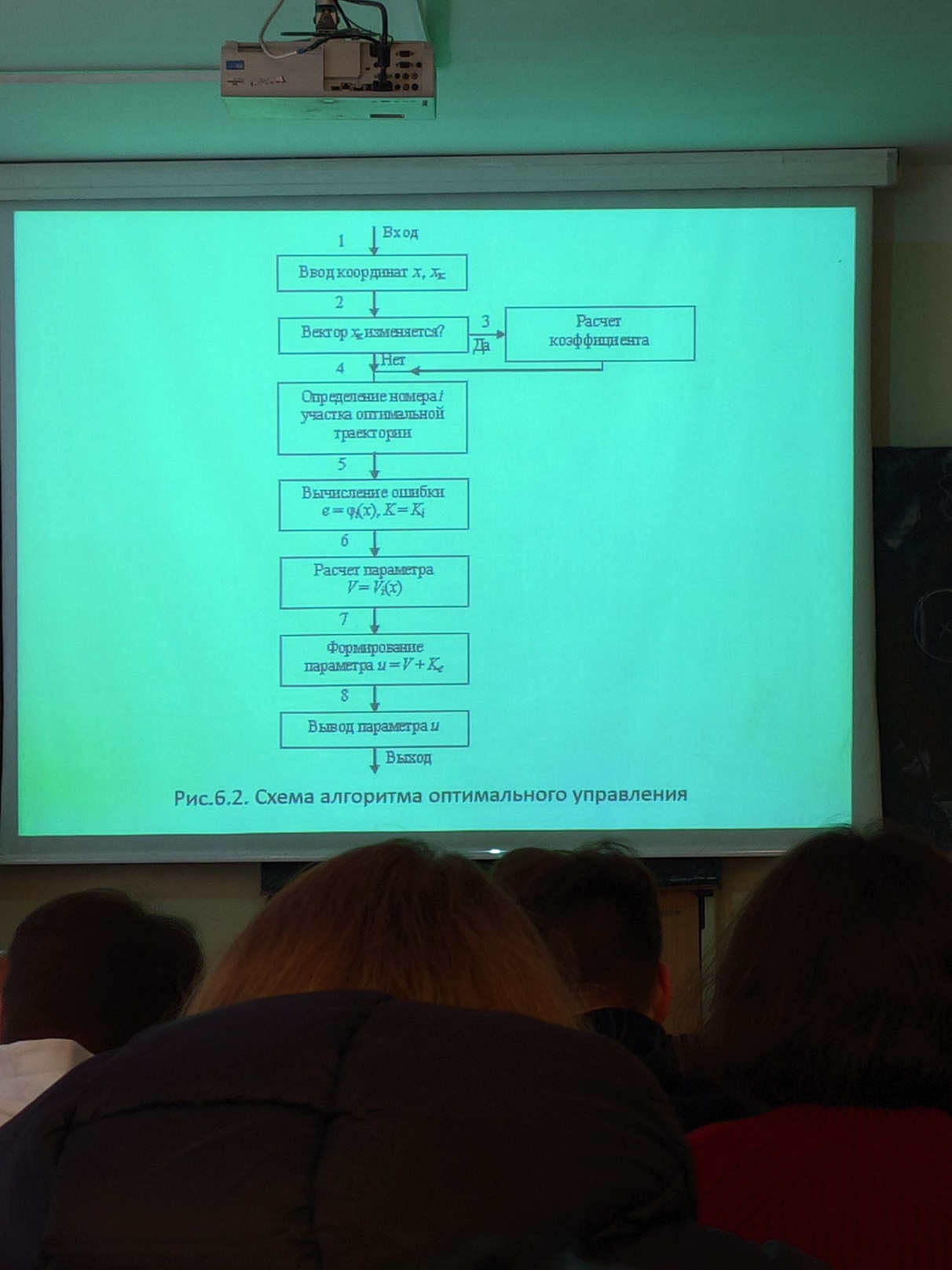
**e = ф(х)**

Алгоритм управления, обеспечивающий минимизацию этой ошибки, имеет вид v = V[x] +K₁e.

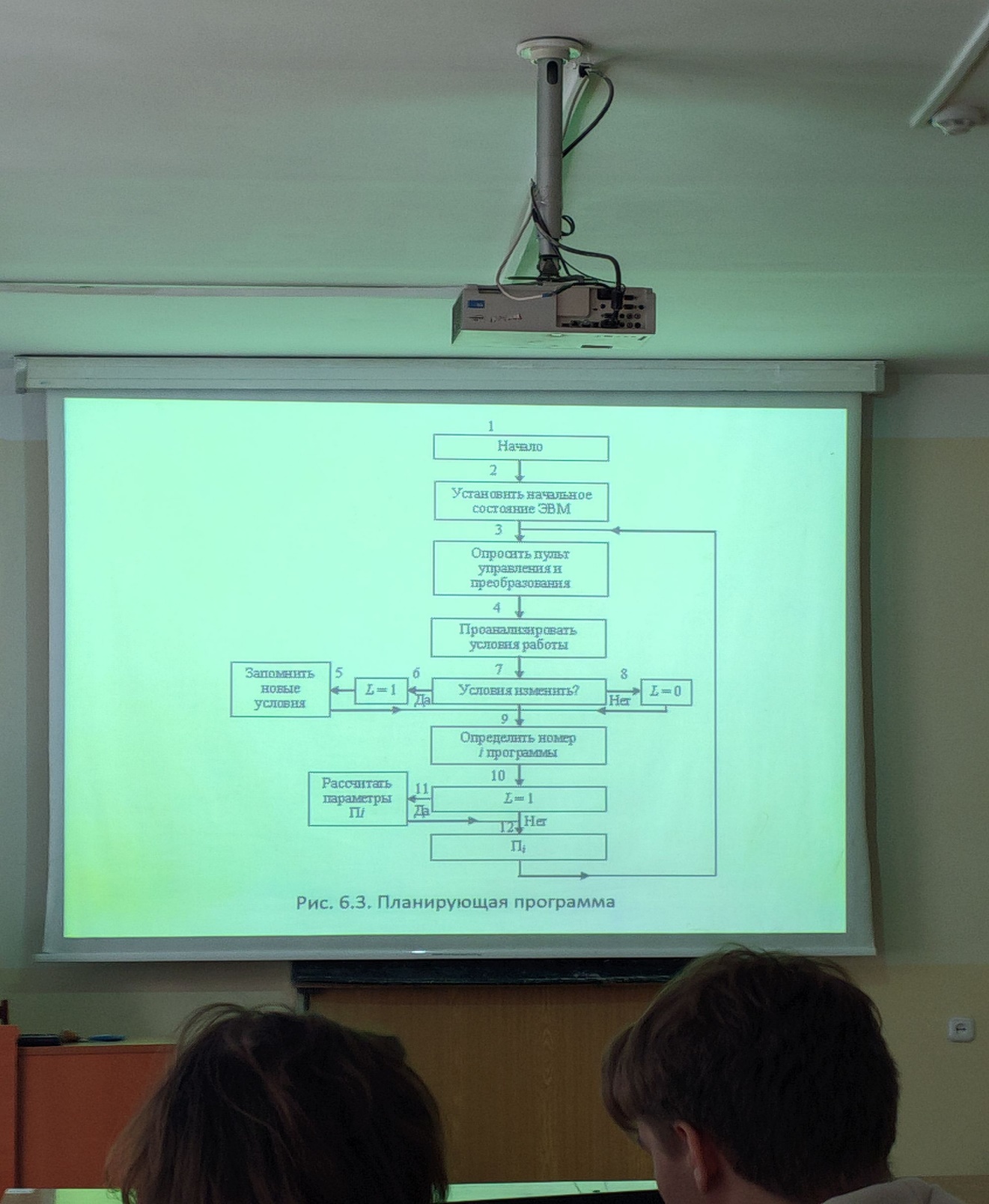
Аналитическое выражение для составляющей V(x) может быть палучено из условия = 0. Матрица K1 обеспечивает желаемое качество отработки ошибки и выбирается с помощью математического обеспечения, используемого при синтезе регуляторов стабилизации или слежения.

На основании сказанного блок-схема алгоритма управления согласованием конечного и начального состояний режимов разгона и стабилизации имеет вид, изображенный на рисунке 6.1, 6. В блоке 1 опрашиваются преобразователи информации и в ЭВМ вводятся координаты вектора состояния и электропривода. Если число преобразователей информации меньше числа координат вектора состояния и если выход преобразователей сильно зашумлен, то после блока 1 необходимо выполнить блок оценки координат вектора состояния. В рассматриваемом алгоритме предполагается, что все координаты вектора состояния измеряются допустимой ошибкой. В блоке 2 вычисляется ошибка е, характеризующая отклонение изображающей точки от заданного многообразия. В блоке 3 вычисляется основная составляющая управляющего воздействия V(x). обеспечивающая движение по заданному многообразию.

Полное управляющее воздействие и вычисляется блоке 4 как сумма основной составляющей V(x) и стабилизирующей составляющей, обеспечивающей устойчивое движение. Рассчитанное управляющее воздействие выводится на объект управления. Принципиальное решение задачи синтеза оптимального управления электроприводом получено для различных критериев качества. Если в процессе синтеза удается получить оптимальное управление как функцию состояния uopt=u(x), то реализация такого управления на ЭВМ не встречает принципиальных затруднений. Однако для нелинейного электропривода такая ситуация встречается редко. В процессе синтеза оптимального управления обычно удается получить только типы управляющих воздействий и последовательность их переключения, причем моменты переключения установить не удается. Алгоритм функционирования ЭВМ, реализующей подобное оптимальное управление, может быть синтезирован следующим образом. Для каждого (-го типа управляющих воздействий, составляющих в совокупности оптимально управление, численными методами рассматриваются участки оптимальной траектории для различных комбинаций начальных и конечных условий.



В блоке 1 осуществляется ввод координат вектора состояниях 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 ф (х) = 0. В блоке 4 путем проверки логических условий, полученных на этапе синтеза, определяется номер і участка оптимальной траектории. Затем рассчитывается отклонение е изображающей точки от соответствующего многообразия и выбирается коэффициент усиления стабилизирующей составляющей управляющего воздействия (блок 5). В блоке 6 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 имеет релейный характер.



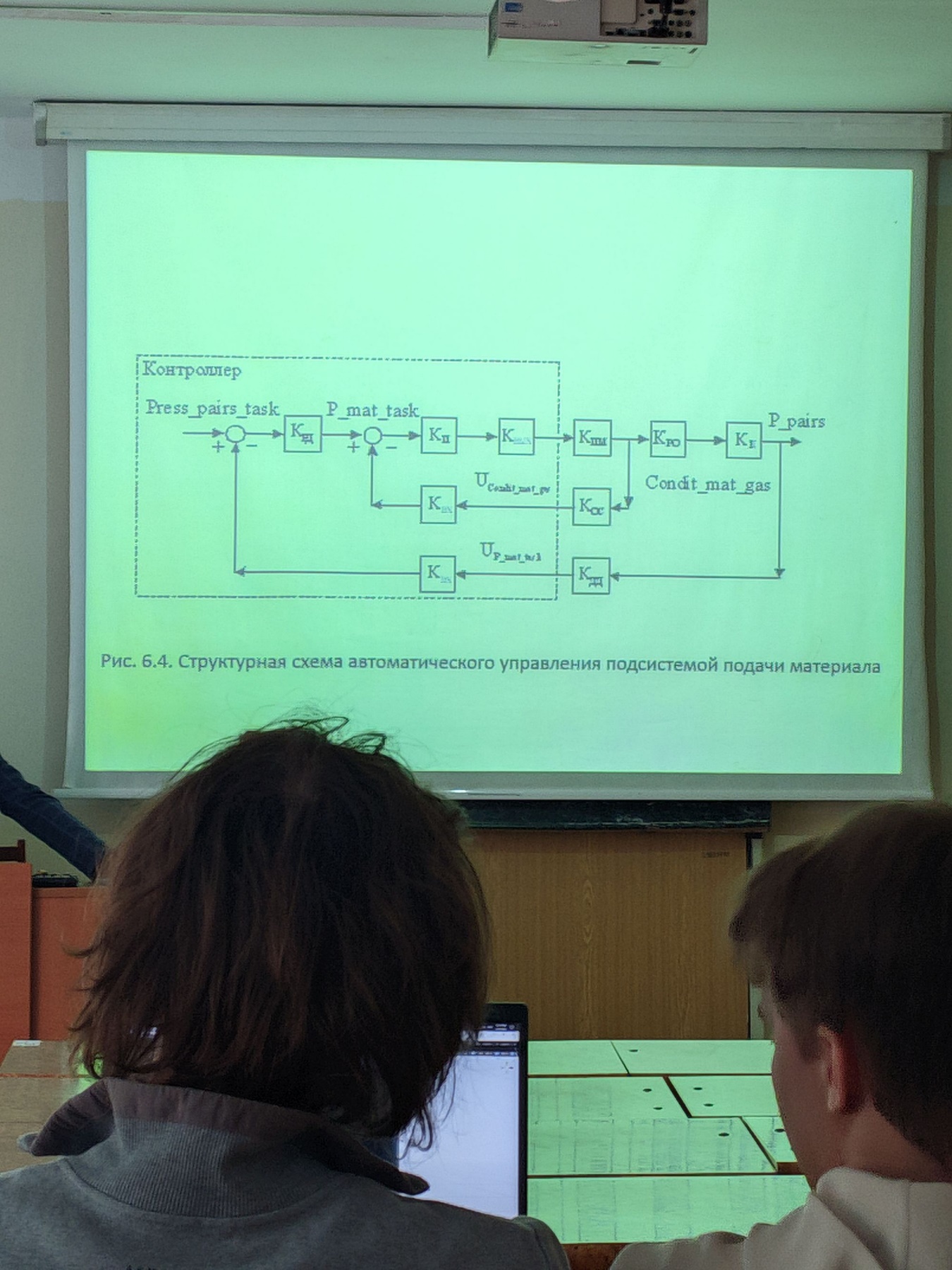
Все рассмотренные программы выполняются на одной и той же управляющей ЭВМ. Тип программы, т. е. структура регулятора выбирается с помощью специальной планирующей программы (рис. 6 3), исходными данными для которой являются команды пульта управления, а также показания преобразователей информации о состоянии 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 пульта управления, а также показани 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 6) При этом в блоке 5 запоминаются новые условия работы. Если же условия работы не изменяются, то в блоке 8 логической переменной L присваивается нулевое значение.

В блоке 9 в реультате проверки логических условий, определяющих последовательность переключения структруры регулятора, находится номер i программы расчета управляющего воздействия, которая выполняется в данное время. При изменившихся условиях работы (L = 1 в блоке 10) в блоке 11 рассчитывается параметр программы П, (элементы матричных коэффициентов алгортима стабилизации, коэффициенты…)

## **7.2 Математическое описание управления технологическим процессом**

Математическое описание процесса управления подачи материала. Подачу необходимого количества материала обеспечивает исполнительный механизм подачи материала. Основой математического описания автоматического управления подсистемой подачи материала является Структурная схема, представленная на рис. 6.4. На схеме приняты следующие обозначения: -Press\_pairs\_task заданная скорость подачи материала;

* Up\_mat\_task - выходное напряжение датчика подачи материала;
* Ucondit\_mat\_gas выходное напряжение датчика степени открытия исполнительного механизма подачи материала;
* P\_pairs - значение выхода материала; -P\_mat\_task - задание на исполнительный механизм;



* Condit\_mat\_gas состояние исполнительного механизма подачи материала;
* Квх - коэффициент передачи аналоговых входов контроллера К;
* Квых - коэффициент передачи аналоговых выходов контроллера K;
* Кдд - коэффициент передачи датчика выхода материала;
* Кос - коэффициент передачи датчика степени открытия исполнительного механизма;
* Ким коэффициент передачи исполнительного механизма;
* Кро- коэффициент передачи регулирующего органа;
* Крд - коэффициент передачи регулятора;
* Кп - коэффициент передачи управляющего воздействия;
* К,к- коэффициент связи выходного и входного параметров.

**7.3 Графическая среда разработки программного обеспечения для микроконтроллеров с архитектурой AVR «Algorithm Builder»**

Среда обеспечивает полный цикл разработки (ввод-вывод, отладку)

Логическая структрура программы становится более наглядной, Основное предназначение — максимальное приведение интерфейса разработки к природе человеческого восприятия.

Время разработки сокращается в 3-5 раз в сравнении с ассемблером

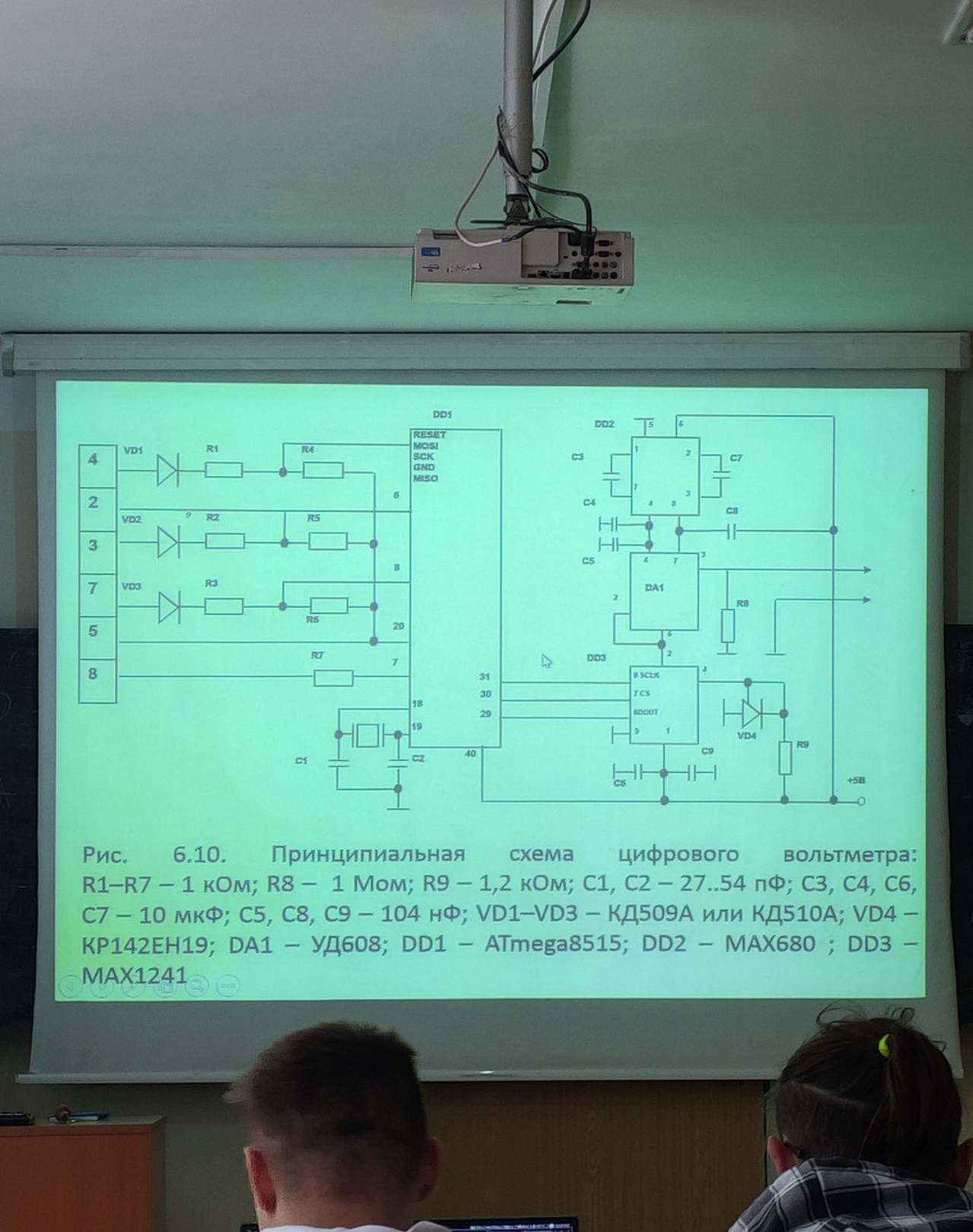
ПО предназначено для ОС Windows.

Разработка сводится к формированию блоков

Блоки находятся в пункте меню Elements

* field - поле
* label - метка - обозначает место в алгоритме, куда можно осуществление условных и безусловных переходов, к метке должен примыкать один из векторов перехода
* vertex - вершина блока - по отображению и назначению идентичен метке, но, в отличие от нее, задает расположение блока на рабочей плоскости и всегда является его началом, имя вершине назначается, только если она ялвляется началом подпрограммы или макроса
* condition - условный переход, представляет собой овальный контур, внутри которого вписывается условие перехода и возможный вектор в виде ломаной линии со стрелкой на конце, возле которой возможн необязательное имя вектора. Конец вектора должен либо заканчиваться на какой-то метке, или вершине, либо на отрезке другого вектора, либо иметь имя адресуемой метки.
* jmp vector - относительный короткий безусловный переход. Представляет собой ломаную линию, исходящую из середины блока со стрелкой на конце, аналогичную вектору объекта “Condition”
* setter - настройщик перефирийных устройств -серый прямоугольник, внутрь которого вписано имя настраиваемого перефирийного компонента микроконтроллера, такого как таймер, АЦП, регистр маски прерываний и пр. Настройщик предназначен для формирования последовательности операций микроконтроллера, которые обеспечивают загрузку необходимых констант в соответствующие управляющие регистры ввода-вывода в соответствии с выбранными свойствами. Перед испольщованием этого элемента, тип микроконтроллера должен быть определен (пункт меню Options\Project options закладка “Chip”)
* text - строка текстового редактора

**7.4 Принципиальная схема цифрового вольтметра**



Описание работы устройства. По сути устройство, представляет собой цифровой вольтметр [4-5]. На входе вольтметра стоит операционный усилитель (DA1), имеющий высокое входное сопротивление. За операционным усилителем следует АЦП (DD2), позволяющий оцифровать интересующее нас напряжение для последующей передачи в микроконтроллер. Микроконтроллер DD1 является главным управляющим звеном устройства, так как он считывает информацию из Ацп общается с персональным И компьютером по последовательному каналу. В устройство также входит преобразователь питания +/-10 В для операционного усилителя. Значение, посылаемое B компьютер, лежит в диапазоне 0...4095 (что соответствует разрядности АЦП), 0 соответствует входному уровню ОВ, 4095- уровню 2,5B, зависимость линейная.

**7.5 Моделирование ПО микроконтроллера в среде MPLAB**

MPLAB IDE - интегрированная среда разработки для микроконтроллеров PICmicro фирмы Microchip Technology Incorporated.

Оно позволяет

- получить откомпилированный код программы

- наблюдать выполнение программы с момощью симулятора, или в реальном масшабе времени, используя эмулятор

- определять время выполнения программы и ее частей

- работать с программаторами

- просматривать текущее значение переменных и специальных регистров в ходе выполнения программы

В графе Cycles выводится число машинных циклов, израсходованных на выполнение фрагмента программы после обнуления счетчика кнопкой Zero.

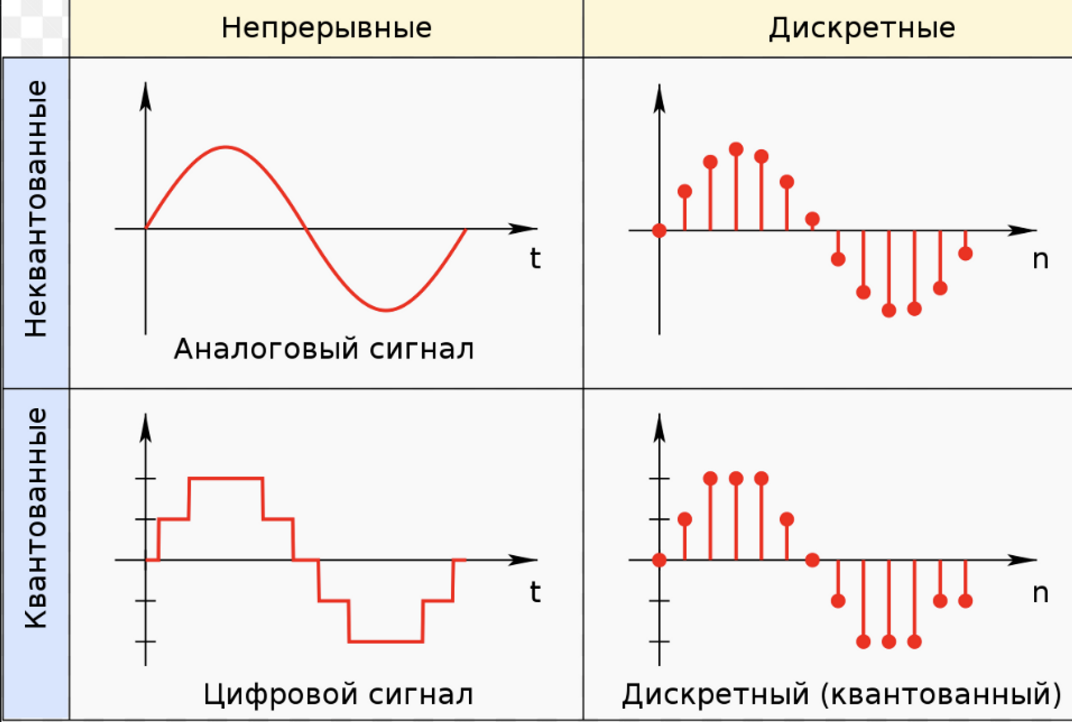
Для более детального изучения работы программы в пакете MPLAB предусмотрены функции точек останова и точек трассировки по которым можно досконально изучить поведение программы не загружая ее в микроконтроллер.

После отладки программа компилируется и полученный hex-файл загружается микроконтроллер с помощью программатора.

Программа у правления скоростью вращения электродвигателем постоянного тока для микроконтроллера PIC16F84, работающего на частоте 10 мГц (полностью совместима с PIC16C84). При записи программы в память микроконтроллера в слове конфигурации должны быть указаны: тип генератора - HS, Watchdog timer - ON, Power-up timer – ON

**ЛК 8**

Цифровой и аналоговый сигнал - графическое представление



Что такое сигнал? Виды сигналов

Сигнал данных \*\*\*\*– форма представления сообщения данных с помощью физической величины, изменения одного или нескольких параметров которой, отображает его изменение.

Сигналы бывают цифровыми, аналоговыми, дискретными, квантоваными, двоичными

*Отладчик*

В Automation Studio отладчик представляет собой инструмент, который позволяет программистам отслеживать и отлаживать программы, написанные для автоматизированных систем управления (АСУ). С помощью отладчика можно следить за выполнением программы пошагово, контролировать значения переменных и условий, проверять корректность работы алгоритмов и тестировать их на различных условиях.

*Отладчик в Automation Studio позволяет выполнять следующие задачи*:

1. Установка точек останова - программист может установить точки останова в программе, чтобы при выполнении кода отладчик останавливался на определенном месте.

2. Пошаговое выполнение кода - отладчик позволяет выполнить код программы пошагово, чтобы программа могла быть проверена на каждом шаге.

3. Контроль за значениями переменных - отладчик позволяет отслеживать значения переменных на каждом шаге выполнения программы.

4. Проверка состояния системы - отладчик позволяет проверить состояние системы и диагностировать возможные ошибки.

5. Использование команд отладки - отладчик предоставляет множество команд отладки, которые позволяют выполнять различные задачи в процессе отладки.

*Многозадачность*

Многозадачность - это способ организации работы компьютерной системы, при котором на одном компьютере или в одной программе может одновременно выполняться несколько задач, каждая из которых работает независимо от остальных.

Многозадачность может быть реализована на уровне операционной системы, которая распределяет ресурсы компьютера между запущенными задачами, или на уровне приложения, которое само управляет своими потоками выполнения.

Многозадачность позволяет повысить эффективность использования компьютера, ускорить выполнение задач и улучшить пользовательский опыт. Однако для правильной работы многозадачности необходимо уметь правильно распределять ресурсы и управлять потоками выполнения, чтобы избежать конфликтов и необходимости выполнения задач в последовательном порядке.

*Какие существуют топологии. Преимущества и недостатки*

1. Звездообразная топология: в этой топологии все устройства сети подключены к одному центральному узлу. Достоинства: простота управления и обслуживания, возможность быстро обнаружить и устранить неисправности в сети. Недостатки: ненадежность центрального узла - если он выходит из строя, вся сеть может остановиться.

2. Шина: в этой топологии все устройства сети подключены к одной линии, которая называется шиной. Достоинства: простота управления, низкая стоимость и легкость установки. Недостатки: отказ одного устройства или поломка шины может привести к полной остановке сети.

3. Кольцевая: в этой топологии устройства сети соединены в кольцо. Достоинства: высокая производительность, поскольку данные передаются только в одном направлении. Недостатки: отказ одного устройства может привести к проблемам со всей сетью.

4. Деревообразная: в этой топологии устройства сети соединены в виде дерева с корневым узлом в вершине. Достоинства: хорошая масштабируемость, возможность добавлять новые узлы без перестройки всей сети. Недостатки: сложность управления, высокая стоимость.

5. Смешанная: в этой топологии комбинируются несколько типов топологий. Достоинства: возможность комбинировать преимущества разных типов топологий. Недостатки: сложность управления, высокая стоимость.

**ЛК 9**

Структура сети B ETHERNET Powerlink выделяются области реального времени и области, где

реального времени не требуется. Это разделение соответствует типично концепция машины и технологического процесса. Оно также удовлетворяет рас хушим требованиям безопасности с целью предотвращения хакерских атак на уровне машин или ущерба от ошибочной передачи данных на верхних иерархи ческих уровнях сети. Фактические требования режима реального времени удов летворяются в области реального времени. Менее критические с временной точ рання дане прозрачным образом передаются между областью реального времени и стандартной областью с использованием стандартных IP четко разграничение между сетью машины и сетью предприятия с самого нача да предотвращает потенциальные угрозы безопасности, сохраняя полную, прозрачность данных.

1. *IP-адрес* (англ. Internet Protocol address) — это уникальный числовой идентификатор, присваиваемый каждому устройству в компьютерной сети, который используется для идентификации и адресации устройств в сети. IP-адрес состоит из четырех чисел, разделенных точками, каждое из которых может быть от 0 до 255. Например, 192.168.1.1 - это IP-адрес роутера или компьютера в локальной сети.

2. *Маска сети* (англ. subnet mask) — это числовой параметр, который используется для определения того, какая часть IP-адреса устройства относится к адресу сети, а какая — к адресу устройства в этой сети. Маска сети задает количество бит, которые используются для определения адреса сети. Например, если маска сети состоит из 24 бит (255.255.255.0), то первые три числа IP-адреса указывают на адрес сети, а последнее число указывает на адрес устройства в этой сети.

3. *Протокол передачи данных* (англ. data transmission protocol) — это стандарт, который определяет, как данные передаются между устройствами в компьютерной сети. Протоколы передачи данных определяют способ кодирования данных, формат заголовков и другие параметры, которые используются для обеспечения надежной и эффективной передачи данных в сети. Например, протоколы передачи данных включают TCP (Transmission Control Protocol), UDP (User Datagram Protocol) и ICMP (Internet Control Message Protocol).

4. *Роутер* (англ. router) и *коммутатор* (англ. switch) являются сетевыми устройствами, используемыми для соединения нескольких устройств в одну сеть. Однако, у этих устройств есть некоторые различия.

Роутер является устройством, которое обеспечивает маршрутизацию данных между различными сетями. Роутеры используются для соединения локальной сети с интернетом или другой удаленной сетью. Роутеры могут также выполнять функции брандмауэра, которые обеспечивают защиту сети от внешних угроз и контролируют доступ к ресурсам сети.

Коммутатор является устройством, которое обеспечивает коммутацию данных между устройствами внутри локальной сети. Коммутаторы используются для соединения компьютеров, серверов и других сетевых устройств в одну локальную сеть. Коммутаторы обычно имеют более высокую пропускную способность и ниже задержку, чем роутеры, что делает их более подходящими для использования в локальной сети с высоким трафиком данных.

Таким образом, основное различие между роутером и коммутатором заключается в их функциях. Роутеры используются для соединения различных сетей, в то время как коммутаторы используются для соединения устройств в одной локальной сети.

*Семиуровеневая модель ISO/OSI*

1. Физический уровень (Physical Layer): определяет физические аспекты передачи данных, такие как метод передачи данных, тип кабеля и интерфейсы.

2. Канальный уровень (Data Link Layer): обеспечивает надежную передачу данных между двумя устройствами в локальной сети, выполняя управление доступом к среде передачи данных, обнаружение ошибок и их исправление.

3. Сетевой уровень (Network Layer): определяет маршрутизацию и пересылку данных между различными сетями, а также управляет трафиком на сетевом уровне.

4. Транспортный уровень (Transport Layer): обеспечивает надежную передачу данных между двумя устройствами на разных сетях, управляет качеством обслуживания (QoS) и контролирует поток данных.

5. Сеансовый уровень (Session Layer): устанавливает, управляет и завершает сеансы связи между устройствами.

6. Уровень представления (Presentation Layer): обеспечивает стандартные методы кодирования, шифрования и форматирования данных для их передачи между устройствами.

7. Прикладной уровень (Application Layer): предоставляет приложениям доступ к сети и реализует прикладные протоколы, такие как HTTP, FTP, SMTP и т.д.

*Режимы работы устройств*

ETHERNET Powerlink совместимое устройство может работать в следующих рабочих режимах:

* Базовый Ethernet-режим: Устройство работает прямо в существующих сетях Ethernet, когда не требуется передача данных в режиме реального време. Этот режим используется по умолчанию после включения устройства
* Предпусковой режим: В ходе запуска системы или после подсоединения устройств к существующей сети, данные иницилизации и конфигурации - загружаются по асинхронному каналу.
* Рeжим ETHERNET Powerlink: После завершения процедуры загрузки, устройство работает в реальном масштабе времени. Управляющий узел проверяет синхронизацию. Время цикла зависит от объема синхронных и асинхронных данных, качество узлов. Базовый цикл состоит из следующих фаз:
  + Фаза запуска: Все сетевые узлы синхронизируются с часами управляющего узла
  + Изохронная фаза: Управляющий узел присваивает каждому узлу фиксированое временное окно для передачи крических с временной точки зрения данных. На этой фазе все другие узлы всегда могут принимать все данные (принцип издатель /подписчик)
  + Авторская фаза. Управляющий узел предоставляет право передавать конкретные данные одному конкретному капу. На этой фазе используются IP протоколы и адресация.

Важные, но все ещё критические с временной точки данные могут передаваться в течение более длинных циклов, чем базовый цикл

**ЛК 10**

Языки программирования высокого и низкого уровня отличаются по абстракции и близости к машинному коду. Вот их определения, преимущества и недостатки:

1. Языки программирования высокого уровня:

- Определение: Языки программирования высокого уровня разработаны для облегчения разработки программ, предоставляя абстракции и инструменты для более высокого уровня анализа и проектирования. Они удобны для разработчиков, так как позволяют писать код, который легче понять и поддерживать.

- Преимущества:

- Читаемость: Код на языках высокого уровня обычно более читабелен и понятен людям, поскольку используются более естественные конструкции и ключевые слова.

- Переносимость: Языки высокого уровня обычно абстрагируются от аппаратных особенностей и операционных систем, что делает программы переносимыми на разные платформы.

- Продуктивность: благодаря высокоуровневым конструкциям и библиотекам, разработка программ на языках высокого уровня может быть более быстрой и удобной.

- Недостатки:

- Производительность: поскольку языки высокого уровня абстрагируются от низкоуровневых деталей, они могут быть менее эффективными по производительности по сравнению с языками низкого уровня.

- Ограничения: Некоторые специфические задачи могут быть сложнее реализовать на языках высокого уровня из-за ограничений абстракции и уровня детализации.

- Примеры: Python, Java, C#, JavaScript, Ruby.

2. Языки программирования низкого уровня:

- Определение: Языки программирования низкого уровня предоставляют прямой доступ к аппаратным ресурсам компьютера и машинному коду. Они обладают более непосредственным управлением аппаратурой и позволяют разработчику работать на более низком уровне абстракции.

- Преимущества:

- Производительность: Языки низкого уровня позволяют полностью использовать ресурсы компьютера и обеспечивают более эффективную работу, особенно для задач требовательных к производительности.

- Управление ресурсами: Разработчик имеет более прямой контроль над памятью, регистрами и другими ресурсами компьютера, что может быть полезным для оптимизации и специфических сценариев.

- Недостатки:

- Сложность: Языки низкого уровня требуют от разработчика более глубокого понимания аппаратуры и более трудоемкой разработки, поскольку низкоуровневые детали должны быть учтены.

- Портируемость: Код на языках низкого уровня обычно зависит от конкретной аппаратной платформы и операционной системы, что затрудняет переносимость.

- Примеры: машинные языки, Assembler.

Каждый из этих языков имеет свои особенности и подходит для разных типов задач. Выбор языка программирования зависит от требований проекта, сроков, производительности, уровня абстракции и других факторов.