**Лекция 1**

**Интерфейсы микроконтроллеров**

*Физический (аппаратный) интерфейс — способ взаимодействия физических устройств.* При попытке связать микроконтроллер с другими устройствами требуется знать определенный набор правил, методов и характеристик оборудования.

**Последовательный интерфейс UART/USART**

Универсальный асинхронный или универсальный синхронно/асинхронный приемопередатчик (Universal Synchronous/Asynchronous Receiver and Transmitter — **UART** или **USART**) — удобный и простой последовательный интерфейс для организации информационного канала обмена микроконтроллера с внешним миром. Способен работать в дуплексном режиме (одновременная передача и прием данных). Он поддерживает протокол стандарта RS-232, что обеспечивает возможность организации связи с персональным компьютером.

Изначально использовался в компьютерах для большинства периферийных устройств, таких как плоттер, удаленный принтер, мышь, внешний модем и т. д. До настоящего времени для последовательной связи IBM PC-совместимых компьютеров используются адаптеры с интерфейсом **RS-232С** (новое название EIA-232D). В современном IBM PC-совместимом компьютере может использоваться до четырех последовательных портов, имеющих логические имена соответственно COM1, COM2, COM3 и COM4. Основой последовательного адаптера является микросхема UART (Universal Asynchronous Receiver/Transmitter) — универсальный асинхронный приемопередатчик. Обычно используется микросхема UART 16550A. Она имеет 16-символьный буфер на прием и на передачу и, кроме того, может использовать несколько каналов прямого доступа в память DMA .

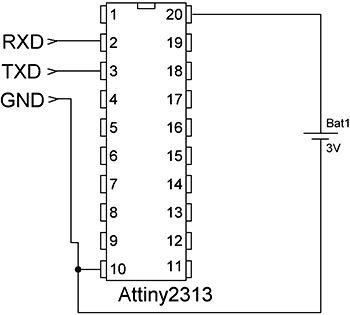


Рис. 1. Схема подключения микроконтроллера Attiny2313

Подключать UART надо, так сказать "наоборот" RXD к TXD, а TXD к RXD как на картинке ниже:

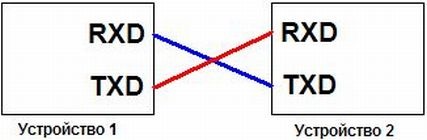


Рис.2 Схема соединения 2х устройств (общий минус не показан)

С помощью UART также можно можно связать микроконтроллер и компьютер, но есть одна проблема: у UART интерфейса логические уровни 0 и +5 вольт, а в компьютере логические уровни в интерфейсе RS-232 могут быть от -25 до -3 вольт и от +3 до +25 вольт. Для этого применяют специальный преобразователь уровней на микросхеме MAX232

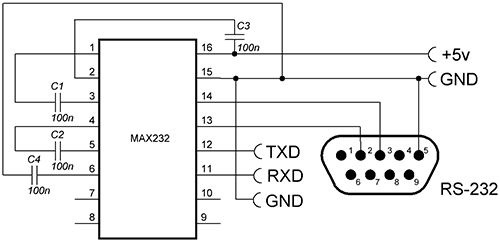


Рис. 3. Преобразователь уровней

Все сигналы UART передаются специально выбранными уровнями, обеспечивающими высокую помехоустойчивость связи. Отметим, что данные передаются в инверсном коде (логической единице соответствует низкий уровень, логическому нулю — высокий уровень.   
Формат передаваемых данных показан на рисунке 4. Собственно данные (5, 6, 7 или 8 бит) сопровождаются стартовым битом, битом четности и одним или двумя стоповыми битами.

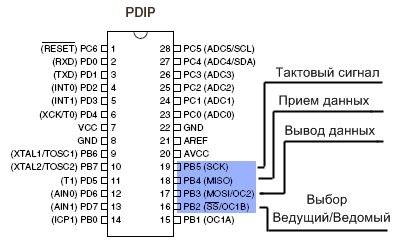


Рис. 4. Расположение выводов интерфейса SPI на микроконтроллере AtMega8

***Электрическое подключение***

Самое простое подключение, в котором участвуют только две микросхемы, показано на рисунке 6. Здесь, ведущий шины передает данные по линии MOSI синхронно со сгенерированным им же сигналом SCLK, а подчиненный захватывает переданные биты данных по определенным фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Представленную схему можно упростить исключением линии MISO, если используемая подчиненная ИС не предусматривает ответную передачу данных или в ней нет потребности. Одностороннюю передачу данных можно встретить у таких микросхем как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Таким образом, рассматриваемый вариант подключения подчиненной ИС требует 3 или 4 линии связи.

Чтобы подчиненная ИС принимала и передавала данные, помимо наличия сигнала синхронизации, необходимо также, чтобы линия SS была переведена в низкое состояние. В противном случае, подчиненная ИС будет неактивна. Когда используется только одна внешняя ИС, может возникнуть соблазн исключения и линии SS за счет жесткой установки низкого уровня на входе выбора подчиненной микросхемы. Такое решение крайне нежелательно и может привести к сбоям или вообще невозможности передачи данных, т.к. вход выбора микросхемы служит для перевода ИС в её исходное состояние и иногда инициирует вывод первого бита данных.

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала.



Рис. 6. Простейшее подключение к шине SPI

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение (рис. 7), либо каскадное (последовательное) (рис. 8).

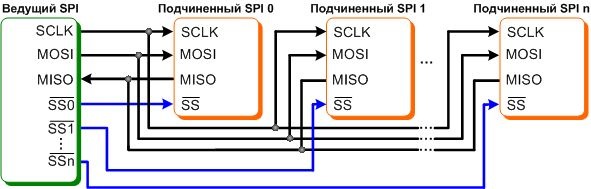


Рис. 7. Независимое подключение к шине SPI

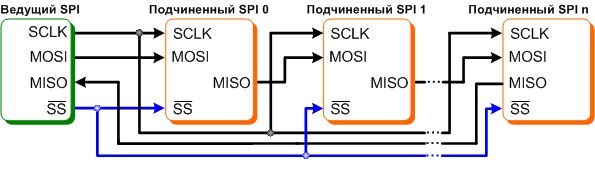


Рис. 8. Каскадное подключение к шине SPI

Независимое подключение более распространенное, т.к. достигается при использовании любых SPI-совместимых микросхем. Здесь, все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины, переводом того или иного сигнала SS в низкое состояние, задает, с какой подчиненной ИС он будет обмениваться данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации подчиненных микросхем (общее число линий связи равно 3+n, где n-количество подчиненных микросхем). Каскадное включение избавлено от этого недостатка, т.к. здесь из нескольких микросхем образуется один большой сдвиговый регистр.

Для этого выход передачи данных одной ИС соединяется со входом приема данных другой, как показано на рисунке 8. Входы выбора микросхем здесь соединены параллельно и, таким образом, общее число линий связи сохранено равным 4. Однако использование каскадного подключения возможно только в том случае, если его поддержка указана в документации на используемые микросхемы. Чтобы выяснить это, важно знать, что такое подключение по-английски называется 'daisy-chaining'.

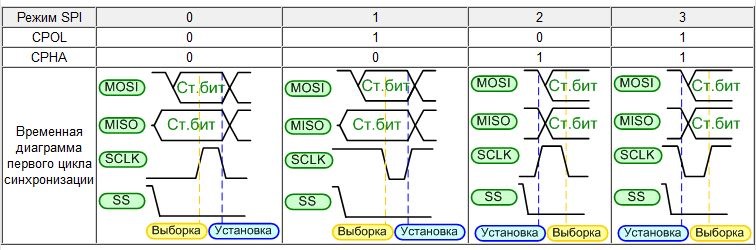
***Протокол передачи***

Протокол передачи по интерфейсу SPI предельно прост и, по сути, идентичен логике работы сдвигового регистра, которая заключается в выполнении операции сдвига и, соответственно, побитного ввода и вывода данных по определенным фронтам сигнала синхронизации. Установка данных при передаче и выборка при приеме всегда выполняются по противоположным фронтам синхронизации. Это необходимо для гарантирования выборки данных после надежного их установления. Эти варианты получили название режимов SPI и описываются двумя параметрами:

**CPOL** — исходный уровень сигнала синхронизации (если CPOL=0, то линия синхронизации до начала цикла передачи и после его окончания имеет низкий уровень (т.е. первый фронт нарастающий, а последний — падающий), иначе, если CPOL=1, — высокий (т.е. первый фронт падающий, а последний — нарастающий));

**CPHA** — фаза синхронизации; от этого параметра зависит, в какой последовательности выполняется установка и выборка данных (если CPHA=0, то по переднему фронту в цикле синхронизации будет выполняться выборка данных, а затем, по заднему фронту, — установка данных; если же CPHA=1, то установка данных будет выполняться по переднему фронту в цикле синхронизации, а выборка — по заднему).

Информация по режимам SPI обобщена в таблице.



Ведущая и подчиненная микросхемы, работающие в различных режимах SPI, являются несовместимыми, поэтому, перед выбором подчиненных микросхем важно уточнить, какие режимы поддерживаются ведущим шины. Аппаратные модули SPI, интегрированные в микроконтроллеры, в большинстве случаев поддерживают возможность выбора любого режима SPI и, поэтому, к ним возможно подключение любых подчиненных SPI-микросхем (относится только к независимому варианту подключения). Кроме того, протокол SPI в любом из режимов легко реализуется программно.

**Лекция 2**

**Микроконтроллеры**

**Классификация и структура микроконтроллеров**

Основная особенность современных микропроцессорных систем (МПС) состоит в завершении перехода от систем, выполненных на основе нескольких больших интегральных схем (ИС), к однокристальным системам, которые объединяют в одном кристалле все основные элементы ЭВМ:

* центральный процессор (ЦП),
* постоянное запоминающее устройство (ПЗУ),
* оперативное запоминающее устройство (ОЗУ)
* порты ввода/выводы (УВВ),
* устройства формирования временных интервалов - таймеры и т.д.

Микроконтроллер – это полноценный компьютер на одной микросхеме. Предназначен для управления различными электронными устройствами и осуществления взаимодействия между ними в соответствии с заложенной в микроконтроллер программой. В отличие от микропроцессоров, используемых в персональных компьютерах, микроконтроллеры содержат встроенные дополнительные устройства. Эти устройства выполняют свои задачи под управлением микропроцессорного ядра микроконтроллера.

Все типы МК можно условно разделить на три основных класса:  
•8-разрядные *МК* для встраиваемых приложений;  
•16- и 32-разрядные управляющие *МК*;

•цифровые сигнальные процессоры (DSP) для обработки данных.  
Основным классификационным признаком микроконтроллеров (МК) является разрядность данных, обрабатываемых *арифметико-логическим устройством* (АЛУ).  
По этому признаку они делятся на 4-, 8-, 16-, 32- и 64-разрядные.

Основные фирмы производители  
1. Intel – i8048, i8051 (CISC архитектура)  
2. Atmel – серия AT89 (CISC архитектура),серия AVR AT90xx, Atmega, ATxmega (RISC архитектура)  
3. Microchip – серия PIC (PIC12xx, PIC16xx и т.д.) (RISC архитектура)  
4. Motorola – серия М68 (М68НС05, НС08, НС11) (CISC архитектура)  
5. Texas Instruments – серия TMS320 – DSP (RISC архитектура)  
6. STMicroelectronics – серия STM8, STM32 (RISC архитектура)

Наиболее распространенным представителем семейства *МК* являются 8-разрядные приборы. потому что основная область их применения находится в системах управления реальными объектами, где применяются, в основном, алгоритмы с преобладанием логических операций, скорость обработки которых практически не зависит от разрядности процессора.

**Первый микроконтроллер**

Первый микроконтроллер появился на свет в 1976 году, через 5 лет после создания первого микропроцессора Intel 4004.

Это была микросхема фирмы Intel, получившая имя i8048.  
Помимо центрального процессора, на кристалле находились 1 кбайт памяти  
программ, 64 байта памяти данных, два 8-битных таймера, тактовый генератор и 27 портов ввода/вывода.

Микроконтроллеры семейства 8048 использовались в игровых консольных приставках Magnavox Odyssey, в клавиатурах первых IBM PC и в ряде других устройств. Это была микросхема фирмы Intel, получившая имя i8048. Помимо центрального процессора, на кристалле находились 1 кбайт памяти программ, 64 байта памяти данных, два 8-битных таймера, тактовый генератор и 27 портов ввода/вывода.

**Закон Мура**. В 1965 году один из основателей Intel Гордон Мур отметил, что новые модели микросхем разрабатывались спустя более-менее одинаковые периоды времени — 18-24 месяца — после появления их предшественников, при этом количество используемых в них транзисторов при этом возрастала каждый раз примерно вдвое. Наблюдение Мура впоследствии блестяще подтвердилось, а обнаруженная им закономерность наблюдается и в наши дни. За 40 лет, истекшие с момента появления микропроцессора i4004 в 1971 году и вплоть до выпуска процессора Xeon Westmere-EX®, количество транзисторов выросло более чем в 1,3 миллиона раз – с 2 300 до 2,6 миллиардов.

**Отличительные признаки микроконтроллеров**

1. модульная организация, при которой на базе одного процессорного ядра  
(центрального процессора) проектируется ряд (линейка) МК, различающихся объемом и типом памяти программ, объемом памяти данных, набором периферийных модулей, частотой синхронизации;

2. использование закрытой архитектуры МК, которая характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса МК. Таким образом, МК представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается;

3. использование типовых функциональных периферийных модулей (таймеры, процессоры событий, контроллеры последовательных интерфейсов, аналогоцифровые преобразователи и др.), имеющих незначительные отличия в алгоритмах работы в МК различных производителей;

4. расширение числа режимов работы периферийных модулей, которые задаются в процессе инициализации регистров специальных функций МК.

Процессорное ядро включает в себя:

1. центральный процессор

2. внутреннюю контроллерную магистраль (ВКМ) в составе шин адреса, данных и управления;

3. схему синхронизации МК;

4. схему управления режимами работы МК, включая поддержку режимов пониженного энергопотребления, начального запуска (сброса) и т.д. При модульном принципе построения все МК одного семейства содержат процессорное ядро, одинаковое для всех МК данного семейства, и  
изменяемый функциональный блок, который отличает МК разных моделей.

**МОДЕЛИ МИКРОКОНТРОЛЛЕРОВ AVR**

В настоящее время в серийном производстве у Atmel находятся семейства AVR ATtiny, ATmega и ATxmega. Также все еще можно встретить в продаже чипы семейства Classic типа AT90

Classic AVR – это классические (первые) модели AVR, что следует и из их названия. В настоящее время они представлены специализированными микроконтроллерами, например для работы с USB, CAN, для формирования многоканального ШИМ сигнала.

Tiny AVR – недорогие микроконтроллеры с малым количеством выходных контактов, чаще в восьмивыводном исполнении. Используются в системах, где не требуются сложные вычислительные алгоритмы.

Mega AVR – мощные микроконтроллеры, включающие большой набор периферии. Это семейство имеет самое большое разнообразие моделей для выбора.  
AVR XMEGA – новое семейство компании Atmel переносит 8/16-битные

микроконтроллеры на новый уровень системных характеристик. В этой связи,

микроконтроллеры AVR XMEGA могут выступать в качестве эталонных 8/16-битных микроконтроллеров. Независимо от выбора семейства микроконтроллеров AVR, следует помнить, что cистема команд всех семейств совместима, поэтому возможен простой перенос программы со слабого на более мощный микроконтроллер

**Память**

В микроконтроллерах AVR реализована архитектура, в соответствии с которой разделены не только адресные пространства памяти программ и памяти данных, но и шины доступа к ним. Каждая из областей памяти данных расположена в своем адресном пространстве.

**Память программ**

Память программ в микроконтроллере AVR является энергонезависимой реализована по flash технологии и предназначена для хранения последовательности команд, управляющих функционированием микроконтроллера, и имеет 16-ти битную организацию.

Все AVR имеют Flash-память программ, которая может быть различного размера от 1 до 256 КБайт. Программа заносится во Flash-память AVR как с помощью специального устройства − программатора, так и с помощью последовательных интерфейсов − SPI- или JTAG. Практически все микроконтроллеры AVR обладают возможностью внутрисхемного программирования (функция ISP in system programming) через коммуникационный интерфейс SPI.

Все микроконтроллеры семейства Mega имеют возможность самопрограммирования, т. е. самостоятельного изменения содержимого свое памяти программ.

**Память данных**

Память данных разделена на три части:

1. регистровая память (набор внутренних регистров микроконтроллера),

2. оперативная память (ОЗУ – оперативное запоминающее устройство с произвольной выборкой или RAM – Random Access Memory),

3. энергонезависимая память (ЭСППЗУ – электрически стираемое перепрограммируемое постоянное запоминающее устройство или EEPROM − Electrically Erasable Programmable Read-Only Memory).

**Регистровая память**

Регистровая память включает 32 регистра общего назначения (РОН или GPR), объединенных в файл, и служебные регистры ввода/вывода (РВВ). И те и другие расположены в адресном пространстве ОЗУ, но не являются его частью. В области регистров ввода/вывода расположены различные служебные регистры (регистры управления микроконтроллером, регистры состояния и т. п.), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. Управление микроконтроллером заключается в управлении этими регистрами

**Энергонезависимая память данных**

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования микроконтроллерной системы, используется EEPROM-память. Все AVR имеют блок энергонезависимой электрически перезаписываемой памяти данных EEPROM от 64 байт до 4 кбайт. Этот тип памяти, доступный программе микроконтроллера непосредственно в ходе ее выполнения, удобен для хранения промежуточных данных, различных констант, коэффициентов, серийных номеров, ключей и т.п. EEPROM может быть загружена извне как через SPI интерфейс, так и с помощью обычного программатора. Число циклов стирание/запись – не менее 100 тыс.

**Оперативная память**

Внутренняя оперативная статическая память (Static RAM − SRAM) имеет байтовый формат организации и используется для оперативного хранения данных. Размер оперативной памяти может варьироваться у различных чипов от 64 байт до 4 кбайт. Число циклов чтения и записи в RAM не ограничено, но при отключении питающего напряжения вся информация теряется. Для некоторых микроконтроллеров возможна организация подключения внешнего статического ОЗУ объемом до 64 кбайтПериферия микроконтроллера

Периферия микроконтроллеров AVR включает:

• цифровые порты ввода/вывода (от 3 до 48 линий ввода и вывода),

• поддержку внешних прерываний (IRQ),

• таймеры-счетчики (T/C),

• широтно-импульсные модуляторы (PWM)

• сторожевой таймер (WD),

• аналоговые компараторы (AC),

• 10-разрядный 8-канальный АЦП (ADC),

• Последовательные интерфейсы UART, JTAG и SPI,

• устройство сброса по понижению питания (RESET)

**Цифровые порты ввода/вывода**

Порт ввода-вывода – логическое объединение сигнальных линий, через которое принимаются и передаются данные.

В зависимости от реализуемых функций различают следующие типы параллельных портов:

1. однонаправленные порты, предназначенные только для ввода или только для вывода информации;

2. двунаправленные порты, направление передачи которых (ввод или вывод) определяется в процессе инициализации МК;

3. порты с альтернативной функцией (мультиплексированные порты). Отдельные линии этих портов используются совместно со встроенными периферийными устройствами МК, такими как таймеры, АЦП, контроллеры последовательных интерфейсов;

4. порты с программно управляемой схемотехникой входного/выходного буфера.Порты ввода/вывода можно назвать Периферией Независимой от Ядра так как входы портов являются асинхронными, состояние портов сохраняется в спящем состоянии ядра.

**Входные элементы управления**

В качестве входных элементов управления могут использоваться

o тумблеры;

o джамперы;

o множественные переключатели;

o кнопки;

o другие элементы схемы

**Выходные элементы управления**

В качестве выходных элементов управления могут использоваться o элементы индикации (единичные светодиоды или светодиодные сборки);

o Электромагнитные реле;

o другие элементы схемы.Порт в МК выполняет роль устройства согласования между самим МК и объектом управления, которые в общем случае работают асинхронно.

Три типа алгоритмов обмена информацией между МК и внешним устройством через параллельные порты ввода/вывода:

1. режим простого программного ввода/вывода;

2. режим ввода/вывода со стробированием (синхронный);

3. режим ввода/вывода с полным набором сигналов подтверждения обмена (асинхронный).

**Типовая схема двунаправленного порта ввода/вывода**

МК.Порты ввода/вывода AVR имеют от 3 до 53 независимых линий "вход/выход". Каждая линия порта может быть запрограммирована на вход или на выход. Мощные выходные драйверы обеспечивают высокую токовую нагрузочную способность 20 мА на линию порта (втекающий ток) при максимальном значении 40 мА, что позволяет, например, непосредственно подключать к микроконтроллеру светодиоды и биполярные транзисторы. Общая токовая нагрузка на все линии одного порта не должна превышать 80 мА (все значения приведены для напряжения питания 5 В).

Архитектурная особенность построения портов ввода/вывода у AVR заключается в том, что для каждого физического вывода (пина, ножки) существует 3 бита контроля/управления, а не 2, как у других распространенных 8-разрядных микроконтроллеров. Это повышает скорость работы микроконтроллера при работе с внешними устройствами, особенно в условиях внешних электрических помех.

**Прерывания (INTERRUPTS)**

Система прерываний – одна из важнейших частей микроконтроллера.

Все микроконтроллеры AVR имеют многоуровневую систему прерываний. Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием. Для обработки каждого такого события разрабатывается отдельная программа, которую называют подпрограммой обработки запроса на прерывание (для краткости − подпрограммой прерывания), и размещается в памяти программ. При возникновении события, вызывающего прерывание, микроконтроллер сохраняет cодержимое счетчика команд, прерывает выполнение центральным процессором текущей программы и переходит к выполнению подпрограммы обработки прерывания.

После выполнения подпрограммы прерывания осуществляется восстановление предварительно сохраненного счетчика команд и процессор возвращается к выполнению прерванной программы.

Для каждого события может быть установлен приоритет. Понятие приоритет означает, что выполняемая подпрограмма прерывания может быть прервана другим событием только при условии, что оно имеет более высокий приоритет, чем текущее. В противном случае центральный процессор перейдет к обработке нового события только после окончания обработки предыдущего

**Таймеры/счетчики**

Микроконтроллеры AVR имеют в своем составе от 1 до 4 таймеров/счетчиков (TIMER/COUNTERS) с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника тактовой частоты, и как счетчики внешних событий.

Их можно использовать:

Таймеры/счетчики способны вырабатывать запросы на прерывания, переключая процессор на их обслуживание по событиям и освобождая его от необходимости периодического опроса состояния таймеров.

Поскольку основное применение микроконтроллеры находят в системах управления реального времени, таймеры/счетчики являются одним из наиболее важных элементов таких систем.

1. для точного формирования временных интервалов,

2. подсчета внешних импульсов на выводах микроконтроллера,

3. формирования последовательности импульсов,

4. формировать широтно-импульсную модуляцию ШИМ (PWM) с программируемыми частотой и скважностью.

**Сторожевой таймер**

Сторожевой таймер (WDT − WatchDog Timer) предназначен для предотвращения катастрофических последствий от случайных сбоев программы. Он имеет свой собственный RC-генератор, работающий на частоте 1 МГц от встроенного генератора. Работа с аналоговыми сигналами

**Аналоговый компаратор**

Аналоговый компаратор (Analog Comparator) сравнивает напряжения на двух выводах (пинах) микроконтроллера. Результатом сравнения будет логическое значение, которое может быть прочитано из программы.

Выход аналогового компаратора можно включить на прерывание от аналогового компаратора. Пользователь может установить срабатывание прерывания по нарастающему или спадающему фронту или по переключению. Присутствует у всех современных AVR, кроме Mega8515

**Аналого-цифровой преобразователь**

Аналого-цифровой преобразователь (АЦП, ADС – analog to digital cjnverter) служит для получения числового значения напряжения, поданного на его вход. Этот результат сохраняется в регистре данных АЦП. Какой из выводов (пинов) микроконтроллера будет являться входом АЦП (их может быть до 8 у AVR), определяется числом, занесенным в соответствующий регистр.

**Последовательные периферийные интерфейсы**

Универсальный последовательный приемопередатчик (UART или USART)

Универсальный асинхронный или универсальный синхронно/ асинхронный приемопередатчик (Universal Synchronous/Asynchronous Receiver and Transmitter – UART или USART) – удобный и простой последовательный интерфейс для организации информационного канала обмена микроконтроллера с внешним миром. Способен работать в дуплексном режиме (одновременная передача и прием данных). Он поддерживает интерфейс стандарта RS-232, что обеспечивает возможность организации связи с персональным компьютером. (Для стыковки МК и компьютера обязательно понадобится схема сопряжения уровней сигналов. Для этого существуют специальные микросхемы, например MAX232).

**Последовательный периферийный интерфейс SPI**

Последовательный периферийный трехпроводный интерфейс SPI (Serial Peripheral Interface) предназначен для организации обмена данными между двумя устройствами. С его помощью может осуществляться обмен данными между микроконтроллером и различными устройствами, такими, как цифровые потенциометры, ЦАП/АЦП, FLASH-ПЗУ и др. С помощью этого интерфейса удобно производить обмен данными между несколькими микроконтроллерами AVR.

Кроме того, через интерфейс SPI может осуществляться программирование микроконтроллера. Двухпроводной последовательный интерфейс TWI Двухпроводной последовательный интерфейс TWI (Two-wire Serial Interface) является полным аналогом базовой версии интерфейса I2C (двухпроводная двунаправленная синхронно-асинхронная шина) фирмы Philips. Этот интерфейс позволяет объединить вместе до 128 различных устройств с помощью двунаправленной шины, состоящей из двух линий - линии тактового сигнала (SCL) и линии данных (SDA).

**Интерфейс JTAG**

Интерфейс JTAG был разработан группой ведущих специалистов по проблемам тестирования электронных компонентов (Joint Test Action Group) и был зарегистрирован в качестве промышленного стандарта IEEE Std 1149.1-1990.

Четырехпроводной интерфейс JTAG используется для тестирования печатных плат, внутрисхемной отладки, программирования микроконтроллеров. Многие микроконтроллеры семейства Mega имеют совместимый с IEEE Std 1149.1 интерфейс JTAG или debugWIRE для встроенной отладки. Кроме того, все микроконтроллеры Mega с флэш-памятью емкостью 16 кбайт и более могут программироваться через интерфейс JTAG.

**Тактовый генератор**

Тактовый генератор вырабатывает импульсы для синхронизации работы всех узлов микроконтроллера. Внутренний тактовый генератор AVR может запускаться от нескольких источников опорной частоты (внешний генератор, внешний кварцевый резонатор, внутренняя или внешняя RC-цепочка). Минимальная допустимая частота ничем не ограничена (вплоть до пошагового режима). Максимальная рабочая частота определяется конкретным типом микроконтроллера и указывается Atmel в его характеристиках, хотя практически любой AVR-микроконтроллер с заявленной рабочей частотой, например, в 10 МГц при комнатной температуре легко может быть "разогнан"до 12 МГц и выше.

**Система реального времени**

Система реального времени – (RTC – real time clock) реализована во всех микроконтроллерах Mega и в двух кристаллах "classic" - AT90(L)S8535. Таймер/счетчик RTC имеет отдельный предделитель, который может быть программным способом подключен или к источнику основной тактовой частоты, или к дополнительному асинхронному источнику опорной частоты (кварцевый резонатор или внешний синхросигнал). Для этой цели зарезервированы два вывода микросхемы. Внутренний осциллятор оптимизирован для работы с внешним "часовым" кварцевым резонатором

**Лекция 3**

**Двухпроводной последовательный интерфейс TWI / I2C**

Двухпроводной последовательный интерфейс **TWI** (Two-wire Serial Interface) является полным аналогом базовой версии интерфейса **I2C** (двухпроводная двунаправленная шина) фирмы Philips. Этот интерфейс позволяет объединить вместе до 128 различных устройств с помощью двунаправленной шины, состоящей из линии тактового сигнала (**SCL**) и линии данных (**SDA**).

Двухпроводной интерфейс (**TWI**) — двунаправленная двухпроводная последовательная шина передачи данных, совместимая со стандартными шинами **I2C** и **SMBus**.

Устройство, подключенное к шине, должно быть либо ведущим, либо подчиненным. Ведущее устройство инициирует передачу данных путем передачи адреса подчиненного устройства и типа передачи: чтение или запись. Если к шине подключено несколько ведущих устройств и некоторая их часть одновременно инициировала передачу, применяется механизм арбитража, который учитывает приоритет этих устройств.

Модуль **TWI** микроконтроллеров может работать и в роли ведущего, и в роли подчиненного устройства.

Ведущая и подчиненная работа полностью отделены друг от друга и предусматривают отдельное управление включением/отключением. Для этих функций также предусмотрены отдельные регистры управления и статуса, а также векторы прерываний. Потеря арбитража, ошибки, коллизии и удержание линии синхронизации обнаруживаются на аппаратном уровне и индицируются отдельными флагами статуса для ведущего и подчиненного режимов.

Ведущий модуль содержит программируемый генератор скорости связи. Даже при синхронизации системы низкими частотами, поддерживается возможность работы шины на частоте 100 и 400 кГц. При необходимости автоматического выполнения операций и снижения сложности программы могут быть разрешены команды **QUICK** и режим **SMART**.

Логика подчиненного модуля продолжает работать во всех экономичных режимах работы МК, в т.ч. **POWER DOWN**. Благодаря этому, подчиненный модуль способен возобновить активную работу МК при обнаружении совпадения адреса. При необходимости программного слежения за адресами, аппаратную функцию обнаружения совпадения адреса можно отключить. Потребность в этом может возникнуть, когда необходимо обнаруживать несколько различных адресов и реагировать на них. При необходимости автоматического выполнения действий и снижения сложности программы можно задействовать режим **SMART**.

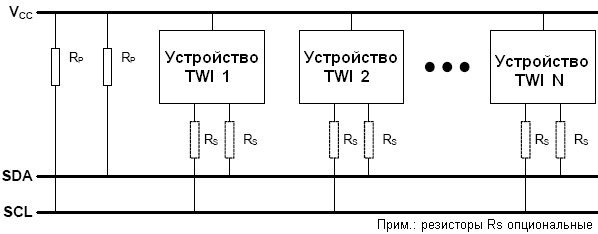
Модуль **TWI** содержит логику контроля состояния шины, которая накапливает информацию для обнаружения условий **START** и **STOP**, коллизий и ошибок шины. С её помощью можно определить состояние шины в ведущем режиме (IDLE, OWNER, BUSY или UNKNOWN).  
При необходимости подключения к внешнему драйверу шины по 4-проводному интерфейсу, внутренние драйверы модуля **TWI** можно отключить.

***Принцип действия шины TWI***

Двухпроводной интерфейс (**TWI**) подключается к простой двухпроводной и двунаправленной шине, которая состоит из двух линий: линия синхронизации (**SCL**) и линия последовательной передачи данных (**SDA**). Источниками сигналов для обеих линий являются схемы с открытым коллектором в выходном каскаде (используется принцип "монтажного И").

Шина **TWI** является простым и эффективным средством соединения по последовательной шине нескольких устройств.

Устройство, подключенное к шине, может быть либо ведущим (**master**), либо подчиненным (**slave**). Чтобы передача по шине стала возможной, к ней должно быть подключено как минимум одно ведущее устройство.



*Топология шины TWI*

Устройство может содержать, как логику ведущего устройства, так и логику подчиненного устройства. Кроме того, оно может эмулировать работу нескольких подчиненных устройств, путем реагирования на несколько адресов.

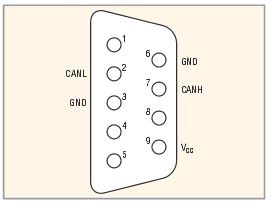
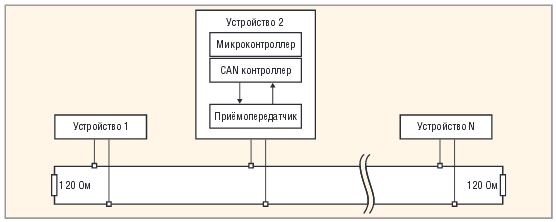
Ведущее устройство сигнализирует о начале транзакции генерацией на шине условия **START** (S). Затем, передается пакет адреса с адресом подчиненного устройства (**ADDRESS**) и указание желаемого направления передачи данных, т.е. чтение или запись (**R/W**). По завершении передачи всех пакетов данных (**DATA**), ведущее устройство генерирует на шине условие **STOP** (P) и транзакция завершается. После приема каждого байта получатель данных генерирует бит подтверждения (A или ACK) или неподтверждения (/A или NACK).

***Структура базовой транзакции шины TWI:***

Сигнал синхронизации транзакции генерирует ведущее устройство, однако, в целях снижения быстродействия шины, любое подключенное к шине устройство может увеличить длительность импульса низкого уровня.

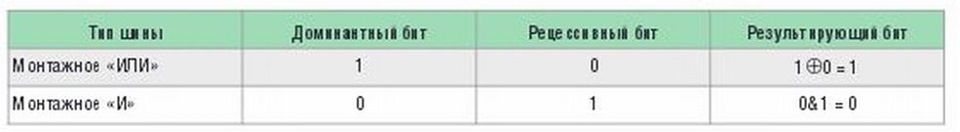
**CAN BUS**

**CAN** (Control Area Network) — последовательная магистраль, обеспечивающая увязку в сеть "интеллектуальных" устройств ввода/вывода, датчиков и исполнительных устройств некоторого механизма или даже предприятия. Характеризуется протоколом, обеспечивающим возможность нахождения на магистрали нескольких ведущих устройств, обеспечивающим передачу данных в реальном масштабе времени и коррекцию ошибок, высокой помехоустойчивостью.



Интерфейс **CAN** относится к типу сетей CR (Collision Resolution, разрешение коллизий), в отличие от сетей типа CD (Collision Detect, обнаружение коллизий), например **Ethernet**. Тип сетей CR обеспечивает приоритетный доступ к передаче сообщения, что необходимо для промышленных устройств.

Приоритетный доступ к передаче сообщений в стандарте ISO-11898 реализуют так называемые рецессивные и доминантные биты. В зависимости от типа шины, объединяющей устройства **CAN**, эти биты могут принимать значение лог. 0 либо лог. 1. В таблице приведены два примера организации шин, поясняющие образование управляющих битов.



***Кадры и их форматы***

Кроме рецессивного и доминантного битов, стандарт интерфейса CAN оперирует таким понятием, как кадр, который используется в протоколе обмена информацией между устройствами. Каждый кадр представляет собой единый набор нескольких служебных битов и байтов. Существуют следующие виды кадров:

* **кадр данных** (data frame) служит для передачи данных;
* **кадр запроса передачи** (remote frame) служит для запроса на передачу кадра данных с тем же идентификатором;
* **кадр перегрузки** (overload frame) обеспечивает промежуток между кадрами данных или кадрами запросов
* **кадр ошибки** (error frame) передаётся узлом, обнаружившим ошибку.

**Лекция 4**

**ВСТРОЕННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ. МИКРОКОНТРОЛЛЕРНАЯ ПЛАТФОРМА STM32**

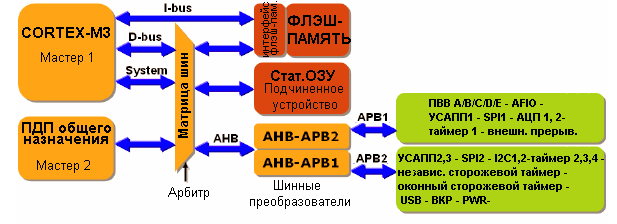
**Экономическая целесообразность использования микроконтроллеров STM32**

* Современный уровень развития технологии обеспечил возможность создания микроконтроллеров с 32- битной архитектурой, цена которых сопоставима с 8-битными микроконтроллерами
* 32-х битное ядро для реализации требует несколько десятков тысяч транзисторов, в то время как 256Кб flash-памяти — это   
  2 млн транзисторов. Соответственно, основное место площади современного контроллера занимает память, порты ввода/вывода и периферийные модули
* Уменьшение площади кристалла даже в 2 раза не уменьшает стоимость вдвое, так как львиную долю стоимости микроконтроллера составляет механическая обработка

Компания STMicroelectronics разрабатывала 32 -битные микроконтроллеры на основе ядер ARM7 и ARM9

Новое поколение 32- битных микроконтроллеров компании STMicroelectronics использует ядро CORTEX M3 и обеспечивает фантастическое соотношение цена/качество. При больших партиях стоимость микроконтроллера доходит до 1 евро

**Архитектура микроконтроллеров STM32**



Микроконтроллеры семейства STM32 выполнены на основе ядра Cortex-M3, которое подключено к flash-памяти по шине инструкций   
I-bus

Шина данных D-bus и системная шина System Cortex подключены к матрице высокоскоростных шин AHB

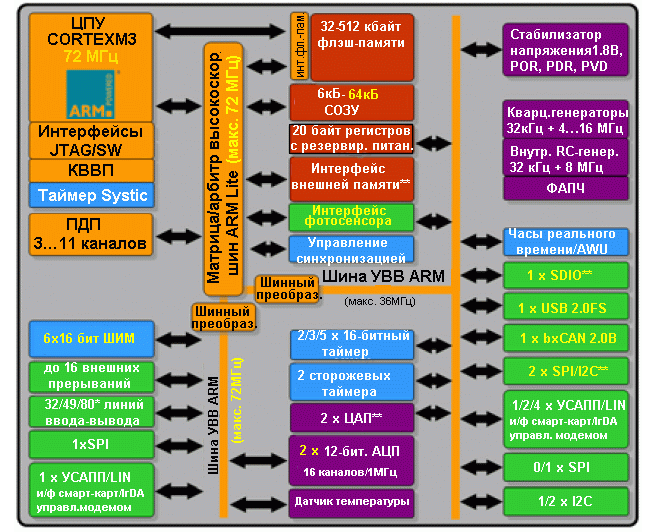
Внутреннее статическое ОЗУ подключено напрямую к матрице шин AHB, с которой также связан блок прямого доступа к памяти (ПДП).

**Обзор платформы STM32**

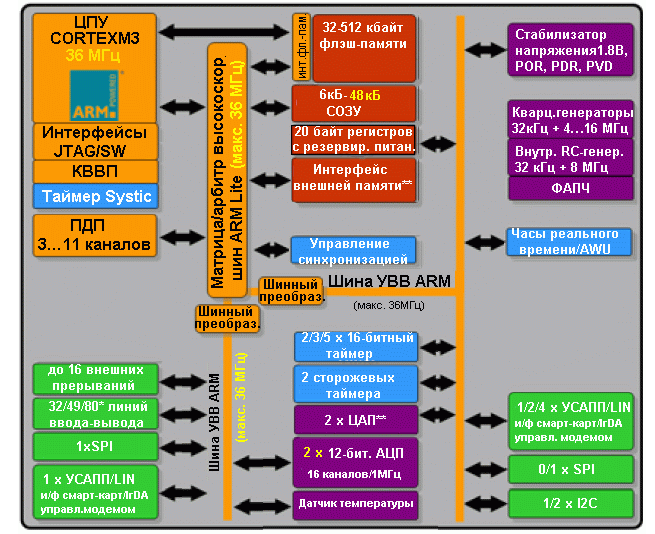
Микроконтроллеры STM32 изначально выпускались в 14 различных вариантах, разделенные на две группы:

* Performance Line, в которую вошли микроконтроллеры с тактовой частотой ЦПУ до 72 МГц;
* Access Line (тактовая частота до 36 МГц)

**Группа Performance Line** (рабочие частоты до 72 МГц)



**Группа Aссess Line** (тактовая частота до 36 МГц)



**АРХИТЕКТУРА**

**Контроллер вложенных векторных прерываний (NVIC)**

NVIC – неотъемлемая часть процессора Cortex-M3. Модуль NVIC в процессоре Cortex-M3 осуществляет обработку прерываний аппаратными средствами.

NVIC отвечает за генерацию прерываний на различные события:

* внешние — изменение логического уровня на входе ножки, пробуждение из режима сна;
* внутренние — завершение приёма/отправки данных, переполнение счётчика таймера и т.п.

NVIC имеет несколько типов конфигурации:

* Стандартная конфигурация: поддержка одного немаскируемого прерывания (NMI), 32 физических прерывания общего назначения с восемью уровнями приоритета;
* Улучшенная конфигурация: поддержка от 1 до 240 физических прерываний с количеством уровней приоритета до 256

**Шинная матрица (Bus matrix)**

Шинная матрица — это развитие идеи простого контроллера шины: здесь шины соединены так, что устройства могут взаимодействовать напрямую, не через ядро.

Архитектурой Cortex-M3 предусмотрены 4 шины, подключенных к матрице:

* ICode, для выборки инструкций и векторов прерываний — дляпользовательского кода. 32-битная шина AHB-Lite типа;
* DCode, для выборки/записи данных и отладочного доступа — для пользовательского кода. 32-битная шина AHB-Lite типа;
* System, для выборки инструкций и векторов прерываний, а также выборки/записи данных и отладочного доступа в системном пространстве — для внутренних компонентов МК. 32-битная шина AHB-типа;
* PPB (Private Peripheral Bus), для выборки/записи данных и отладочного доступа — для периферии. 32-битная шина APB-типа

**Ядро Cortex-M3**

Выполняет инструкции, производит вычисления в своём арифметико-логическом устройстве (АЛУ)

Его Гарвардская архитектура позволяет одновременно загружать инструкции и осуществлять доступ к памяти — благодаря этому, а также трёхступенчатому конвейеру, большинство инструкций выполняются за 1 такт

Ядро Cortex-M3 поддерживает набор инструкций Thumb-2, который содержит как 32-битные, так и 16-битные инструкции для сокращения объёма кода за счёт менее дальнобойных переходов; имеет 13 регистров общего назначения, снижая потребность в частом доступе к памяти

**СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ**

**Структура CMSIS**

CMSIS состоит из трех файлов:

* core\_m3.h - вспомогательные функции доступа к регистрам ядра;
* startup\_stm32f10x\_xx.s - набор файлов для каждой линейки семейства STM32, обеспечивающие инициализацию стека и таблицу векторов прерываний;
* system\_stm32f10x.h - файл начальной инициализации тактовой частоты микроконтроллера

**Стандартная библиотека STM32**

Стандартная библиотека для работы с периферийными модулями написана в соответствии со стандартом ANSI C и может использоваться с любым стандартизованным компилятором

Библиотека состоит из двух взаимодополняющих составляющих:

* заголовочных файлов и файлов реализации всей периферии микроконтроллеровSTM32 - STM32F10x\_StdPeriph\_Driver;
* заголовочных файлов и файлов реализации ядра ARM Cortex-M3

Вся функциональность периферийных модулей описана в заголовочных файлах и файлах реализации. Например, для портов ввода-вывода это два файла: stm32f10x\_gpio.h и stm32f10x\_gpio.c

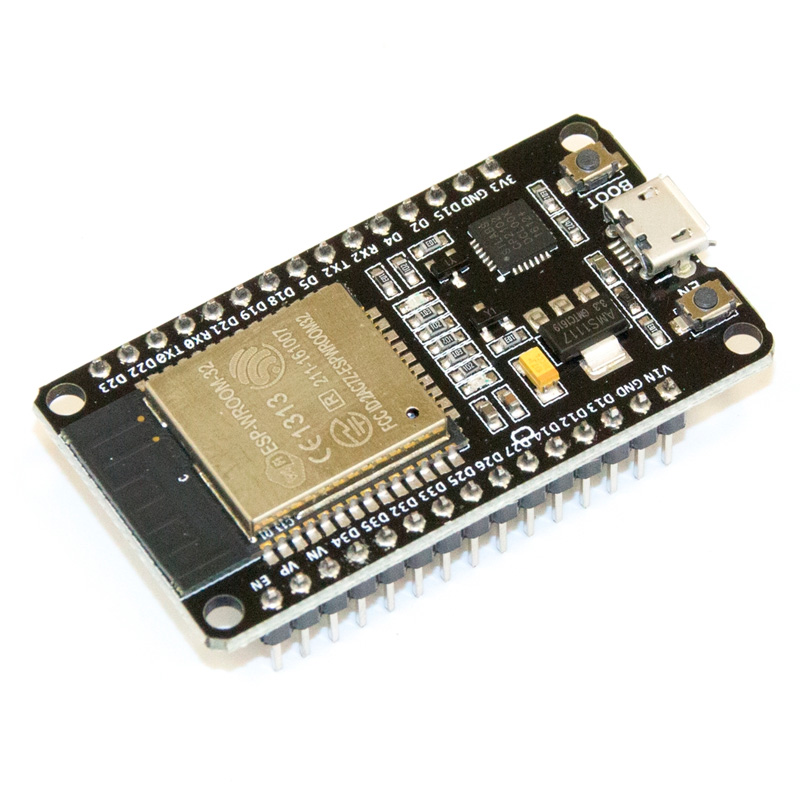
**Лекция 5**

**Микроконтроллер ESP32**

Микроконтроллер ESP32 – это одна из самых доступных и мощных платформ для создания умных ардуино-проектов с поддержкой WiFi. Придя на смену ESP8266, этот чип дал новые возможности для разработчиков, хотя по-прежнему остались старые проблемы с поддержкой и документацией.

**Описание микроконтроллера ESP32**

Фирма Espressif выпустила мощный недорогой микроконтроллер ESP32 летом 2016 года. Устройство представляет собой систему на кристалле, построенную по технологии TSMC 40 нм, с Wi-Fi и Bluetooth контроллерами. Оно оснащено двухъядерным 32-битным процессором, который работает на частотах 80, 160 или 240 МГц. Также в систему интегрированы антенные коммутаторы, радиочастотные компоненты, фильтры, усилители, модули управления питанием. Подключается ESP32 к компьютеру через обычный USB провод.



Микроконтроллер ESP32

**Характеристики чипа**

Технические характеристики ESP32:

* Двух- или одноядерный 32-битный процессор Tensilica Xtensa LX6;
* Тактовая частота – 160 или 240 МГц;
* 520 Кб SRAM;
* Максимальный ток потребления 260 мА, в спящем режиме – 10 мА;
* Стандарты беспроводной связи – Wi-Fi: 802.11 b / g / N, Bluetooth: v4.2 BR/EDR and BLE;
* Наличие датчиков температуры, Холла, тач-сенсоров;
* Инфракрасное дистанционное управление;
* Можно подключать двигатели и светодиоды через ШИМ разъем;
* Стандарт IEEE 802.11 с поддержкой WFA, WPA/WPA2 и WAPI;
* Возможность безопасной загрузки;
* Шифрование флэш диска.

Расположение выводов зависит от производителя. Например, есть плата ESP32 DEVKIT V1 DOIT, у которой 36 контактов. Распиновка представлена на рисунке:

Одним из самых популярных модулей является ESP-WROOM-32. Распиновка также приведена на картинке.

**Отличия esp32 от esp8266**

Платформы ESP8266 и ESP32 произведены одной компанией Espressif. Микроконтроллер ESP32 отличается от своего предшественника улучшенными характеристиками,  увеличенным функционалом и большим объемом памяти. Стоимость нового прибора дороже примерно в 2 раза.

Важным преимуществом ESP32 является более быстрый Wi-Fi и Bluetooth. В ESP32 установлен более мощный процессор, позволяющий реализовывать сложные проекты. Эта платформа подходит для приложений, в которых требуется интернет или новые интерфейсы. Для более дешевых разработок используется ESP8266.

Объем памяти у нового устройства ESP32 увеличен – 512 Кб против 160 Кб ESP8266. Также ESP32 отличается большим количеством выводов GRIO. К нескольким контактам на ESP32 прикреплены емкостные сенсорные датчики и датчик температуры. На обоих устройствах контакты GRIO можно использовать по-разному.  ESP32 имеет 18 12-битных АЦП каналов. У его предшественника есть всего 1 10-битный вывод АЦП.

Мощность процессора значительно влияет на скорость работы. Модуль ESP32 показывает рекордную производительность по сравнению с предшественником ESP8266. Загрузка страницы с длинным скетчем и множеством графики занимает секунды.

Из недостатков ESP32 можно выделить отсутствие библиотек для поддержки сенсоров и малое количество драйверов. Это связано с тем, что плата появилась в продаже недавно. Но учитывая все преимущества микроконтроллера и его перспективы, эта проблема будет решена уже в ближайшее время.

**Лекция 6**

**Что такое сервопривод и как им управлять?**

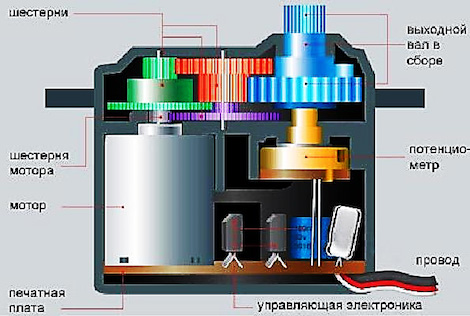
Сервопривод – это такой вид привода, который может точно управлять параметрами движения. Другими словами, это двигатель, который может повернуть свой вал на определенный угол или поддерживать непрерывное вращение с точным периодом.

**Где используют?**

В проектах ардуино робототехники серво часто используется для простейших механических действий:

* Повернуть дальномер или другие датчики на определенный угол, чтобы измерить расстояние в узком секторе обзора робота.
* Сделать небольшой шаг ногой, движение конечностью или головой.
* Для создания роботов-манипуляторов.
* Для реализации механизма рулевого управления.

**Устройство сервопривода**



Вся схема управления серво находится внутри корпуса, управляющие сигналы и питание подаются, как правило, идут по трем проводам: земля, напряжение питания и управляющий сигнал.

**Материалы шестерней сервопривода**

У большинства сервоприводов связующим звеном между валом и внешними элементами является шестеренка, поэтому очень важно, из какого материала она сделана. Наиболее доступных вариантов два: металлические или пластмассовые шестерни. В более дорогих моделях можно найти элементы из карбона и даже титана.

**Преимущества серводвигателей**

Широкое использование сервоприводов связано с тем, что они обладают стабильной работой, высокой устойчивостью к помехам, малыми габаритами и широким диапазоном контроля скорости. Важными особенностями сервоприводов являются способность увеличивать мощность и обеспечение обратной информационной связи. И этого следует, что при прямом направлении контур является передатчиком энергии, а при обратном – передатчиком информации, которая используется для улучшения точности управления

**Отличия серво и обычного двигателя**

Включая или выключая обычный электрический двигатель, мы можем сформировать вращательное движение и заставить двигаться колеса или другие предметы, прикрепленные к валу. Движение это будет непрерывным, но для того, чтобы понять, на какой угол повернулся вал или сколько оборотов он сделал, потребуется устанавливать дополнительные внешние элементы: энкодеры. Сервопривод уже содержит все необходимое для получения информации о текущих параметрах вращения и может самостоятельно выключаться, когда вал повернется на необходимый угол.



**Подключение серводвигателя к ардуино**

Сервопривод обладает тремя контактами, которые окрашены в разные цвета. Коричневый провод ведет к земле, красный – к питанию +5В, провод оранжевого или желтого цвета – сигнальный. К Ардуино устройство подключается через макетную указанным на рисунке образом. Оранжевый провод (сигнальный) подключается к  цифровому пину, черный и красный – к земле и питанию соответственно.

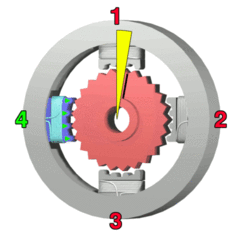
**Заключение**

Сервоприводы играют очень важную роль для многих проектов Ардуино, от робототехнических до систем умного дома. Все, что связано с движением,  традиционно требует особых знаний и создать полноценный правильно работающий привод – непростая задача. Но с помощью серводвигателей можно во многих случаях упростить задачу, поэтому серво постоянно используется даже в проектах начального уровня.

**Лекция 7**

**Устройство управления шаговым двигателем.**

**Принцип работы шагового двигателя.**



Шаговый электродвигатель — это синхронный бесщёточный электродвигатель с несколькими обмотками, в котором ток, подаваемый в одну из обмоток статора, вызывает фиксацию ротора. Последовательная активация обмоток двигателя вызывает дискретные угловые перемещения (шаги) ротора.

**Описание**

Конструктивно шаговые электродвигатели состоят из статора, на котором расположены обмотки возбуждения, и ротора, выполненного из магнитомягкого или из магнитотвёрдого материала. Шаговые двигатели с магнитным ротором позволяют получать больший крутящий момент и

**Преимущества и недостатки**

Преимущества

Главное преимущество шаговых приводов — точность. При подаче потенциалов на обмотки шаговый двигатель повернётся строго на определённый угол.

К приятным моментам можно отнести стоимость шаговых приводов, в среднем в 1,5-2 раза дешевле сервоприводов. Шаговый привод, как недорогая альтернатива сервоприводу, наилучшим образом подходит для автоматизации отдельных узлов и систем, где не требуется высокая динамика.

Недостатки

Возможность «проскальзывания» ротора — наиболее известная проблема этих двигателей. Это может произойти при превышении нагрузки на валу, при неверной настройке управляющей программы (например, ускорение старта или торможения не адекватно перемещаемой массе), при приближении скорости вращения к резонансной. Наличие датчика позволяет обнаружить проблему, но автоматически скомпенсировать её без остановки производственной программы возможно только в очень редких случаях. Чтобы избежать проскальзывания ротора, как один из способов, можно увеличить мощность двигателя.

**Лекция 8**

**Микросхема ESP8266** – один из самых популярных инструментов для организации беспроводной связи в проектах умного дома. С помощью беспроводного контроллера можно организовывать связь по интерфейсу WiFi, обеспечивая проектам Arduino выход в интернет и возможность дистанционного управления и сбора данных. На основе ESP8266 созданы такие популярные платы как **WeMos** и **NodeMcu**, а также огромное количество самодельных проектов. В этой статье, мы узнаем, что из себя представляет ESP82266, какие бывают ее разновидности, как работать с ESP8266 в среде Arduino IDE.

ESP8266 – микроконтроллер с интерфейсом WiFi, который имеет возможность исполнять программы из флеш-памяти.  Устройство было выпущено в 2014 году китайской фирмой Espressif и практически сразу же стало популярным.

Контроллер недорогой, обладает небольшим количеством внешних элементов и имеет следующие технические параметры:

* Поддерживает Wi-Fi протоколы 802.11 b/g/n с WEP, WPA, WPA2;
* Обладает 14 портами ввода и вывода, SPI, I2C, UART, 10-бит АЦП;
* Поддерживает внешнюю память до 16 МБ;
* Необходимое питание от 2,2 до 3,6 В, потребляемый ток до 300 мА в зависимости от выбранного режима.

Важной особенностью является отсутствие пользовательской энергонезависимой памяти на кристалле. Программа выполняется от внешней SPI ПЗУ при помощи динамической загрузки необходимых элементов программы. Доступ к внутренней периферии можно получить не из документации, а из API набора библиотек. Производителем указывается приблизительное количество ОЗУ – 50 кБ.

Особенности:

* Удобное подключение к компьютеру – через USB кабель, питание от него же;
* Наличие встроенного преобразователя напряжения 3,3В;
* Наличие 4 Мб флеш-памяти;
* Встроенные кнопки для перезагрузки и перепрошивки;
* Все порты выведены на плату на две гребенки с шагом 2,5 мм.

**Основные отличия Ардуино от ESP8266**

* ESP8266 имеет больший объем флеш-памяти, при этом у ESP8266 отсутствует энергонезависимая память;
* Процессор ESP8266 быстрее, чем у Ардуино;
* Наличие Wi-Fi у ESP8266;
* ESP8266 потребляеn больше тока, чем для Ардуино;

**Программирование ESP8266 в Arduino IDE**

Программный комплект разработчика esp8266 включает в себя:

* Компилятор из пакета GNU Compiler Collection.
* Библиотеки, стеки протоколов WiFi, TCP/IP.
* Средство загрузки информации в программу контроллера.
* Операционная IDE.
* **Основные функции языка Wiring**. Управлять портами GPIO можно точно так же, как и пинами на плате Ардуино: pinMode, digitalRead, digitalWrite, analogWrite. Команда analogRead(А0) позволяет считать значения АЦП. При помощи команды analogWrite (pin, value) можно подключить ШИМ на нужном выходе GPIO. При value=0 ШИМ отключается, максимальное значение достигает константы, равной 1023.С помощью функций attachInterrupt, detachInterrupt можно выполнять прерывание на любом порте GPIO, кроме 16.
* **Тайминг и delay**. Используя команды millis и micros можно вернуть мс и мкс, которые прошли с момента старта. Delay позволяет приостановить исполнение программы на нужное время. Также функция delay(…) позволяет поддерживать нормальную работу Wi-Fi, если в скетче присутствуют большие элементы, которые выполняются более 50 мс. Yield() – аналог функции delay(0).
* **Serial и Serial1 (UART0 и UART1).** Работа Serial на ESP8266 аналогична работе на ардуино. Запись и чтение данных блокируют исполнение кода, если FIFO на 128 байт и программный буфер на 256 байт заполнены. Объект Serial пользуется аппаратным UART0, для него можно задать пины GPIO15 (TX) и GPIO13 (RX) вместо GPIO1(TX) и GPIO3(RX). Для этого после функции Serial.begin(); нужно вызвать Serial.swap();. Аналогично Serial1 использует UART1, который работает на передачу. Необходимый пин для этого GPIO2.

**Выводы по ESP8266**

С помощью плат на основе микросхемы ESP8266 вы можете добавить в свои проекты возможности “большого интернета”, сделав их гораздо более интеллектуальными. Дистанционное управление, сбор и анализ данных на сервере, обработка голоса и работа с изображением – все это становится доступным, когда мы подключаем наш проект по WiFi к интернету. В следующих статьях мы подробно рассмотрим то, как можно программировать устройства на базе **esp8266**, а также уделим внимание таким популярным платам как **WeMos** и **NodeMcu**.

**Лекция 9**

**Гироскопы МЭМС (Микроэлектромеханические системы)**

**Что такое гироскоп?**

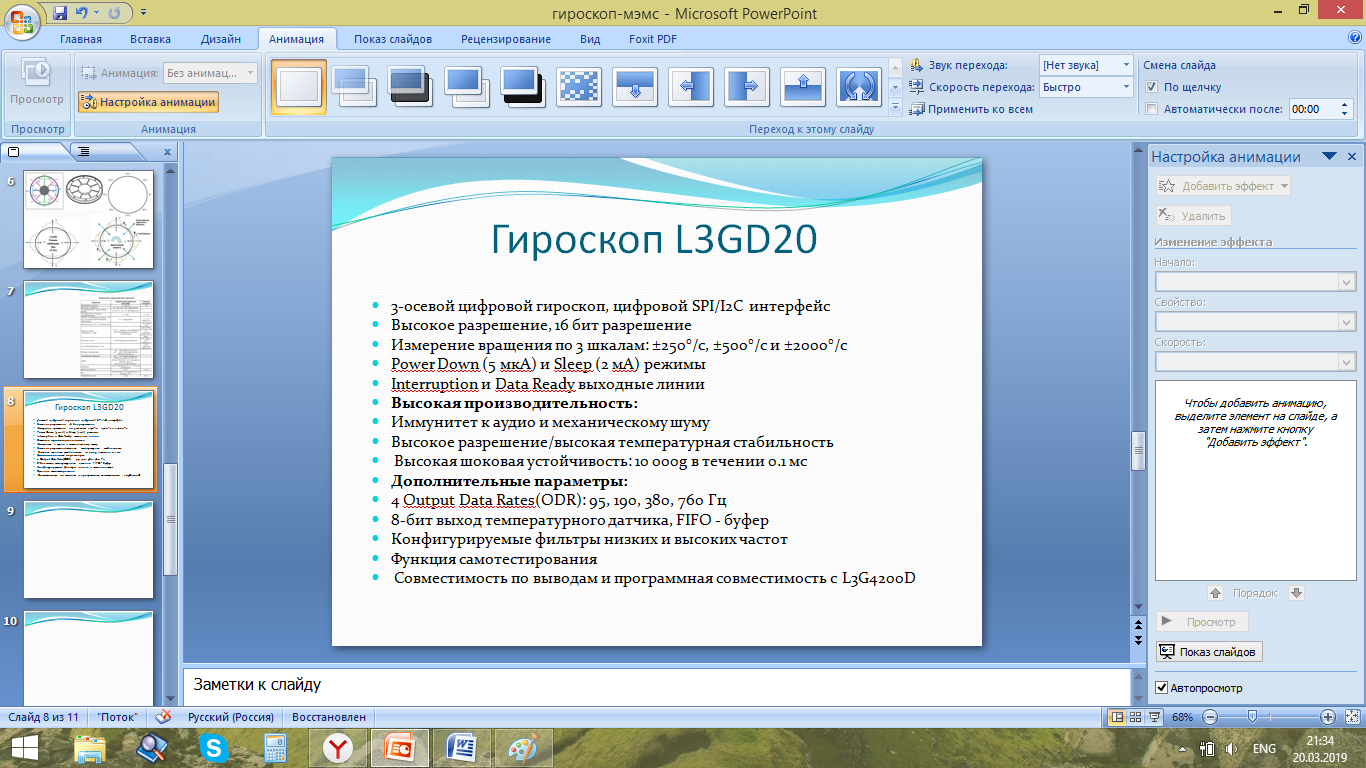
Гироскоп - это устройство, способное реагировать на изменение углов ориентации  тела, на котором оно установлено, относительно инерциальной системы отсчета.

**Одноосевой МЭМС-датчик угловой скорости (гироскоп) с вибрирующим кремниевым кольцом**

Данный кремниевый цифровой гироскоп разработан с учетом требований к низкой стоимости изделия и экономичному энергопотреблению для систем навигации и наведения нового поколения. Он способен измерять угловую скорость до ± 1,0 рад/с и имеет два режима вывода: аналоговый сигнал напряжения, линейно-пропорциональный угловой скорости, и цифровой по протоколу SPI®.

Как правило, подобные гироскопы выпускаются в герметичных керамических LCC корпусах которые можно устанавливать на системные платы. Датчик состоит из пяти основных компонентов: кремниевый кольцевой МЭМС сенсор (MEMS-ring), основание из кремния (Pedestal), интегральная микросхема гироскопа (ASIC), корпус (Package Base), крышка (Lid).

**Гироскоп L3GD20**



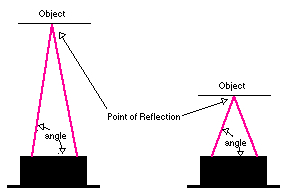
**Лекция 10**

**Виды датчиков**



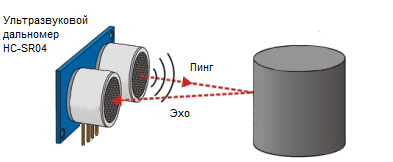
Для определения расстояния до какого-либо объекта используются различные датчики дальности. Одними из них являются инфракрасный датчик и ультразвуковой, которые широко используются на платформе Arduino.

**Принцип работы (инфракрасный)**



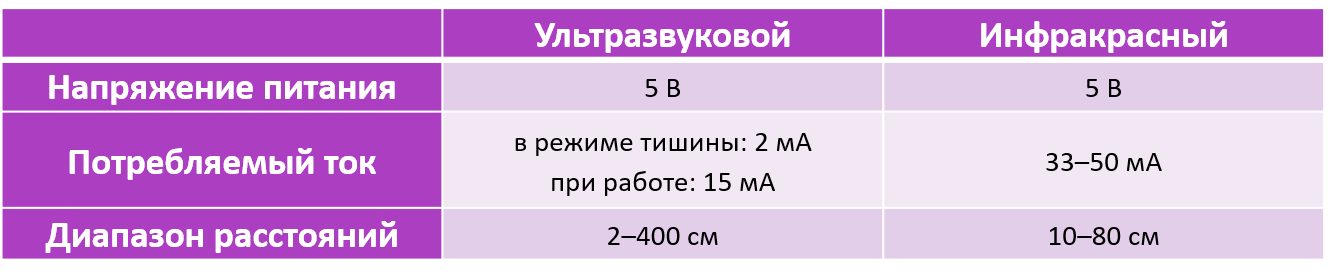
Датчик фиксирует отражение от объекта луча светодиода на удалении не более 0,8 м. “Угол зрения” можно представить в виде конуса с диаметром в средней части около 0,16 м. Излучение отражается под углом и возвращается на воспринимающий элемент датчика. Чем дальше объект, тем острее угол.

**Принцип работы (ультразвуковой)**



Ультразвуковой датчик HC-SR04, или, как его еще называют, сонар, подает звуковые импульсы (пинг) на частоте 40 кГц и ловит отраженный сигнал (эхо). По времени передачи импульса и его возврату определяется дальность.

**Сравнение**



Оба датчика могут использоваться не только в качестве измерителей расстояния. Например, закрепив и подключив к платформе, они могут использоваться как «глаза» робота, которые не позволят ему слепо передвигаться, врезаясь во все подряд. Напротив, он сможет объезжать препятствия, служить сигнализацией, строить карту помещения. Так, с помощью датчиков можно сделать робота, который найдет выход из лабиринта. Каждый из датчиков имеет свои плюсы и минусы.

**2 семестр**

Лекция 1

**Операционные системы реального времени (RTOS)**

Операционная система — это программный интерфейс между пользователем и аппаратурой вычислительной машины.

Операционная система – это набор программ, контролирующих работу прикладных программ и системных приложений и исполняющих роль интерфейса между пользователями, программистами, прикладными программами, системными приложениями и аппаратным обеспечением компьютера.

Список основных сервисов, предоставляемых типичными операционными системами.

1. Разработка программ: ОС представляет программисту разнообразные инструменты разработки приложений: редакторы, отладчики и т.п. Ему не обязательно знать, как функционируют различные электронные и электромеханические узлы и устройства компьютера. Часто пользователь не знает даже системы команд процессора, поскольку он может обойтись мощными высокоуровневыми функциями, которые представляет ОС.

2. Исполнение программ. Для запуска программы нужно выполнить ряд действий: загрузить в основную память программу и данные, инициализировать устройства ввода-вывода и файлы, подготовить другие ресурсы. ОС выполняет всю эту рутинную работу вместо пользователя. 5 ОС как расширенная виртуальная машина

3. Доступ к устройствам ввода-вывода. Для управления каждым устройством используется свой набор команд. ОС предоставляет пользователю единообразный интерфейс, который скрывает все эти детали и обеспечивает программисту доступ к устройствам ввода-вывода с помощью простых команд чтения и записи. Если бы программист работал непосредственно с аппаратурой компьютера, то для организации, например, чтения блока данных с диска ему пришлось бы использовать более десятка команд с указанием множества параметров. После завершения обмена программист должен был бы предусмотреть еще более сложный анализ результата выполненной операции.

4. Контролируемый доступ к файлам. При работе с файлами управление со стороны ОС предполагает не только глубокий учет природы устройства ввода-вывода, но и знание структур данных, записанных в файлах. Многопользовательские ОС, кроме того, обеспечивают механизм защиты при обращении к файлам.

5. Системный доступ. ОС управляет доступом к совместно используемой или общедоступной вычислительной системе в целом, а также к отдельным системным ресурсам. Она обеспечивает защиту ресурсов и данных от несанкционированного использования и разрешает конфликтные ситуации. 6 ОС как расширенная виртуальная машина

6. Обнаружение ошибок и их обработка. При работе компьютерной системы могут происходить разнообразные сбои за счет внутренних и внешних ошибок в аппаратном обеспечении, различного рода программных ошибок (переполнение, попытка обращения к ячейке памяти, доступ к которой запрещен и др.). В каждом случае ОС выполняет действия, минимизирующие влияние ошибки на работу приложения (от простого сообщения об ошибке до аварийной остановки программы).

7. Учет использования ресурсов. Хорошая ОС имеет средства учета использования различных ресурсов и отображения параметров производительности вычислительной системы. Эта информация важна для настройки (оптимизации) вычислительной системы с целью повышения ее производительности

Управление ресурсами включает решение ряда общих, не зависящих от типа ресурса задач: • планирование ресурса – определение, какому процессу, когда и в каком качестве (если ресурс может выделяться частями) следует выделить данный ресурс; • удовлетворение запросов на ресурсы – выделение ресурса процессам; • отслеживание состояния и учет использования ресурса – поддержание оперативной информации о занятости ресурса и распределенной его доли; • разрешение конфликтов между процессами, претендующими на один и тот же ресурс.

Система реального времени - это конкретная система, связанная с реальным объектом. Она включает в себя необходимые аппаратные средства, операционную систему и прикладное программное обеспечение. Операционная система реального времени – это только инструмент, помогающий построить конкретную систему реального времени.

Систему реального времени можно разделить как бы на три слоя:

1. Ядро — содержит только строгий минимум, необходимый для работы системы: управление задачами, их синхронизация и взаимодействие, управление памятью и устройствами ввода/вывода. Размер ядра очень ограничен: часто несколько килобайт.

2. Система управления — содержит ряд дополнительных сервисов, расширяющих возможности ядра: расширенное управление памятью, вводом/выводом, задачами, файлами и т.д., обеспечивает также взаимодействие системы и управляющего/управляемого оборудования.

3. Система реального времени — содержит систему управления и набор утилит: средства разработки (компиляторы, отладчики и т.д.), средства визуализации (взаимодействия человека и операционной системы).

**Лекция 2**

Стандарты ОСРВ

Большие различия в спецификациях ОСРВ и огромное количество существующих микроконтроллеров выдвигают на передний план проблему стандартизации в области систем реального времени.

Наиболее ранним и распространенным стандартом ОСРВ является стандарт POSIX (IEEE Portable Operating System Interface for Computer Environments, IEEE 1003.1). Первоначальный вариант стандарта POSIX появился в 1990 г. и был предназначен для UNIXсистем, первые версии которых появились в 70-х годах прошлого века. Спецификации POSIX определяют стандартный механизм взаимодействия прикладной программы и операционной системы и в настоящее время включают набор более чем из 30 стандартов. Для ОСРВ наиболее важны семь из них (1003.1a, 1003.1b, 1003.1c, 1003.1d, 1003.1j, 1003.21, 1003.2h), но широкую поддержку в коммерческих ОС получили только три первых.

Стандарт 1003.1a (OS Definition) содержит базовые интерфейсы ОС – поддержку единственного процесса, поддержку многих процессов, управление заданиями, сигналами, группами пользователей, файловой системой, файловыми атрибутами, управление файловыми устройствами, блокировками файлов, устройствами ввода/вывода, устройствами специального назначения, системными базами данных, каналами, очередями FIFO, а также поддержку языка C.

Стандарт 1003.1b (Realtime Extensions) содержит расширения реального времени – сигналы реального времени, планирование выполнения (с учетом приоритетов, циклическое планирование), таймеры, синхронный и асинхронный ввод/вывод, ввод/вывод с приоритетами, синхронизация файлов, блокировка памяти, разделяемая память, передача сообщений, семафоры.

Стандарт 1003.1c (Threads) касается функций поддержки многопоточной обработки внутри процесса – управление потоками, планирование с учетом приоритетов, мьютексы (специальные синхронизирующие объекты в межпроцессном взаимодействии, подающие сигнал, когда они не захвачены каким-либо потоком), приоритетное наследование в мьютексах, переменные состояния (condition variables).

Стандарт 1003.1d включает поддержку дополнительных расширений реального времени – семантика порождения новых процессов (spawn), спорадическое серверное планирование, мониторинг процессов и потоков времени выполнения, тайм-ауты функций блокировки, управление устройствами и прерываниями.

Стандартизация POSIX позволяет решать следующие задачи: • интеграция информационных систем из компонент различных изготовителей; • эффективность реализаций и разработок, благодаря точности спецификаций и соответствию стандартным решениям, отражающим передовой научно-технический уровень; • эффективность переноса прикладного программного обеспечения, благодаря использованию стандартизованных интерфейсов и прозрачности механизмов реализации сервисов систем.

Стандарт ARINC-653 (Avionics Application Software Standard Interface) разработан компанией ARINC в 1997 г. Этот стандарт определяет универсальный программный интерфейс APEX (Application/Executive) между ОС авиационного компьютера и прикладным ПО.

Стандарт OSEK/VDX является комбинацией стандартов, которые изначально разрабатывались в двух отдельных консорциумах, впоследствии слившихся.

Первые ОС разрабатывались как монолитные системы без четко выраженной структуры. Для построения монолитной системы необходимо скомпилировать все отдельные процедуры, а затем связать их вместе в единый объектный файл.

Как альтернатива классическому варианту архитектуры ОС, часто используется микроядерная архитектура ОС. В привилегированном режиме остается работать только небольшая часть ОС, называемая микроядром. Микроядро защищено от остальных частей ОС и приложений

В то же время признаны следующие достоинства микроядерной архитектуры: • единообразные интерфейсы; • простота расширяемости; • высокая гибкость; • возможность переносимости; • высокая надежность; • поддержка распределенных систем; • поддержка объектно-ориентированных ОС.

**Лекция 3**

Операционные системы реального времени

Подразделяют ОСРВ на два класса — системы «жесткого» и «мягкого» реального времени. Системы «жесткого» РВ имеют минимальные объем и время отклика, но обладают ограниченными сервисными средствами. Типичным примером ОСРВ этого класса служит VxWorks. Системы «мягкого» РВ требуют большего объема памяти, имеют более длительное время отклика, но удовлетворяют широкому спектру требований пользователя по режиму обслуживания задач, уровню предоставляемого сервиса.

Также ОСРВ можно разделить на системы специализированные и универсальные. Специализированной ОСРВ называется система, где конкретные временные требования априори определены. Универсальная СРВ должна уметь выполнять произвольные временные задачи без применения специальной техники. Разработка таких систем является сложной задачей, хотя обычно требования, предъявляемые к таким системам, мягче, чем требования для специализированных систем

OS-9 компании Microware относится к классу UNIX-подобных операционных систем реального времени. Благодаря модульному объектно-ориентированному дизайну OS-9 является чрезвычайно гибко конфигурируемой высокопроизводительной системой реального времени

Характеристики использования OS-9: • Поддержка HOST-систем (IBM PC(MS Windows 3.xx, 95, NT), IBM RS6000/AIX, Sun4/SunOS/Solaris,HP9000 S/700, SGI IRIS/IRIX) • Широкая поддержка сторонних разработчиков программного обеспечения • Широкая поддержка разработчиков аппаратных средств промышленной автоматизации • Программные продукты для «вертикальных» рынков (мобильная беспроводная коммуникация, устройства с минимальным потреблением энергии, мультимедиа) • Специальные программные средства и лицензионная политика для OEM. • Более 5 млн. установленных копий • Более 800 OEM-партнеров

ОСРВ VxWorks компании Wind River Systems является встраиваемой ОС реального времени для систем жесткого реального времени. Значительная часть используемого в Интернет сетевого оборудования, коммутаторы, маршрутизаторы, серверы удаленного доступа и устройства широкополосного доступа работают под управлением VxWorks.

Wind River Linux компании Wind River Systems Inc. — коммерческий дистрибутив ОС Linux, предназначенный для разработки встраиваемых приложений. Это полнофункциональный и комплект, позволяющий разработчикам устройств достигать большей результативности с меньшими затратами, исключая из Linux-проектов множество ресурсоемких рутинных задач

Проект OSEK/VDX (Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen/Vehicle Distributed eXecutive) предназначался для разработки стандарта открытой архитектуры ОС и стандарта API для систем автомобильной промышленности. Его разработку финансировали BMW, Bosch, DaimlerChrysler, Siemens, Opel и Volkswagen.

LynxOS — операционная система жесткого реального времени, поддерживающая многопроцессные и многопотоковые приложения, которая предназначена для специализированной и телекоммуникационной аппаратуры. Эта ОС является полностью детерминированной и обладает POSIX-, UNIX- и Linux-совместимостью. Областями применения ОС LynxOS являются также сложные системы безопасности

Windows CE (также известна как WinCE) — это вариант операционной системы Microsoft Windows для наладонных компьютеров, смартфонов и встраиваемых систем. Сегодня Windows CE (Compact Edition/Compact Embedded) не является «урезанной» версией Windows для настольных ПК, она основана на совершенно другом ядре и является операционной системой реального времени с набором приложений, основанных на Microsoft Win32 API.

Microsoft Windows «в чистом виде» нельзя отнести к операционным системам реального времени. Одним из возможных решений является использование совместно с Microsoft Windows подсистемы реального времени, исполняющейся на том же процессоре (если процессор один) или на выделенном процессоре(-ах) (если их несколько). Этот подход использован фирмой VenturCom в продукте RTX (Real Time Extension). Сущность подхода заключается в использовании модифицированного HAL (Hardware Abstraction Level). Изменять ядро Microsoft не разрешает, а исходный код HAL предоставляет своим партнерам, одним из которых является VenturCom.

RTEMS (англ. Real-Time Executive for Multiprocessor Systems) — свободная операционная система реального времени с открытым исходным кодом, разработанная для встраиваемых систем. Изначально буква "М" означала Missile (ракетный), потом Military (военный). В настоящее время в версиях RTEMS, реализованных на языке С, буква "М" подразумевает Multiprocessor, в версиях на языке Ada буква "M" подразумевает Military. Разработчик системы компания OAR (On-Line Applications Research Corporation, США). Система была создана по заказу министерства обороны США для использования в системах управления ракетными комплексами.

FreeRTOS — это многозадачная, мультиплатформенная, бесплатная операционная система жесткого реального времени с открытым исходным кодом. Разработана компанией Real Time Engineers Ltd. специально для встраиваемых систем.

Операционная система QNX является разработкой канадской компании QNX Software System Ltd. по заказу Минобороны США. Впервые система появилась на рынке в 1981 году. Среди пользователей QNX значатся такие компании, как Du Pont, Eastman Kodak, General Mills, General Motors, Motorola, Texaco.

**Лекция 4**

Микроядро QNX Neutrino

Микроядерная операционная система построена на основе миниатюрного ядра, обеспечивающего минимальные службы для произвольной группы взаимодействующих процессов, которые, в свою очередь, обеспечивают функциональность более высокого уровня. Первичным здесь становится модульный принцип, тогда как стремление к компактности является вторичным.

ОС QNX Neutrino действует как своего рода «программная шина», позволяющая динамически присоединять/отсоединять модули ОС по мере необходимости. Большинство служб по обеспечению работы в реальном масштабе времени и по планированию потоков реализуется прямо в микроядре, эти службы могут работать даже без дополнительных модулей ОС.

Микроядро в ОС QNX Neutrino — очень компактно и предназначено для выполнения только нескольких базовых функций. • управление потоками (thread services) посредством POSIX-примитивов для создания потоков; • управление сигналами (signal services) посредством примитивов сигналов; • обмен сообщениями (message-passing services), с помощью которого микроядро выполняет трассировку всех сообщений, пересылаемых между всеми потоками внутри системы; • синхронизация (synchronization services) посредством примитивов синхронизации потоков; • планирование (scheduling services), с помощью которого микроядро осуществляет планирование потоков в реальном времени на основе различных алгоритмов; • управление таймерами (timer services), с помощью которого микроядро обеспечивает большой набор POSIX-таймеров; • управление процессами (process management services), выполняемое администратором процессов, вместе с которым ядро образует единый модуль procnto. Администратор процессов предназначен для управления процессами, памятью и пространством имен путей (pathname space).

В микроядре ОС QNX Neutrino существуют системные вызовы (kernel calls), которые служат для управления следующими объектами: • потоками (threads); • сообщениями (message passing); • сигналами (signals); • часами (clocks); • таймерами (timers); • обработчиками прерываний (interrupt handlers); • семафорами (seamphores); • блокировками взаимного исключения, или мутексами (mutexes);

Все службы ОС, за исключением тех, которые выполняются обязательным модулем микроядра/администратора процессов (procnto), обрабатываются посредством стандартных процессов. Система может содержать следующие компоненты: • администраторы файловых систем; • администраторы устройств символьного ввода/вывода; • графический пользовательский интерфейс (Photon); • сетевой администратор; • стек протоколовTCP/IP.

Мультипрограммирование (многозадачность, multitasking) – это такой способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются несколько программ. Чтобы поддерживать мультипрограммирование, ОС должна определить для себя внутренние единицы работы, между которыми будут разделяться процессор и другие ресурсы компьютера.

Подсистема управления процессами и потоками мультипрограммной ОС выполняет функции: создание процессов и потоков; • обеспечение процессов и потоков необходимыми ресурсами; • изоляция процессов; • планирование выполнения процессов и потоков; • диспетчеризация потоков; • организация межпроцессного взаимодействия; • синхронизация процессов и потоков; • завершение и уничтожение процессов и потоков.

С каждым потоком связывается: • Счетчик выполнения команд • Регистры для текущих переменных • Стек • Состояние Потоки делят между собой элементы своего процесса: • Адресное пространство • Глобальные переменные • Открытые файлы • Таймеры • Семафоры • Статистическую информацию. В остальном модель идентична модели процессов.

Преимущества ULT-потоков: • Такую многопоточность можно реализовать на ядре, не поддерживающим многопоточность. • Более быстрое переключение, создание и завершение потоков. • Процесс может иметь собственный алгоритм планирования. Недостатки ULT-потоков: • Отсутствие прерывания по таймеру внутри одного процесса. • При использовании блокирующего системного запроса (процесс переводится в режим ожидания, например: чтение с клавиатуры, а данные не поступают) все остальные потоки блокируются. • Сложность реализации.

Главный недостаток связан с необходимостью двукратного переключения режимов пользовательский – ядро, ядро – пользовательский для передачи одного потока к другому в рамках одного и того же процесса.

Потоки создаются и уничтожаются динамически, их количество внутри одного процесса может изменяться в значительных пределах. В дополнение к показанным переходам, поток может перейти из любого состояния (кроме DEAD) в состояние READY.

RUNNING – поток выполняется процессором; READY – поток ожидает выполнения, пока процессор занят выполнением другого потока равного или более высокого приоритета; CONDVAR – поток блокирован на условной переменной (например, при вызове функции pthread\_condvar\_wait()); INTERRUPT – поток блокирован и ожидает прерывания (т. е. поток вызвал функцию InterruptWait()); JOIN – поток блокирован и ожидает завершения другого потока (например, при вызове функции pthread\_join()); MUTEX – поток блокирован блокировкой взаимного исключения (например, при вызове функции pthread\_mutex\_lock()); NANOSLEEP – поток находится в режиме ожидания в течение короткого периода времени (например, при вызове функции nanosleep()); Потоки и процессы NET\_REPLY – поток ожидает ответ на сообщение от другого узла сети (т.е. поток вызвал функцию MsgReply\*()); NET\_SEND – поток ожидает получения импульса или сигнала от другого узла сети (т. е. поток вызвал функцию MsgSendPulse(), MsgDeliverEvent() или SignalKill()); RECEIVE – поток блокирован на операции получении сообщения (например, при вызове функции MsgReceive()); REPLY – поток блокирован при ответе на сообщение (т. е. при вызове функции MsgSend() и получении сообщения сервером); SEM – поток ожидает освобождения семафора (т. е. поток вызвал функцию SyncSemWait()); DEAD – поток завершен и ожидает завершения другого потока;

SEND – поток блокируется при отправке сообщения (т. е. после того, как поток вызвал функцию MsgSend(), но получения сообщения сервером еще не произошло); SIGSUSPEND – поток блокирован и ожидает сигнал (т. е. поток вызвал функцию sigsuspend()); SIGWAITINFO – поток блокирован и ожидает сигнал (т. е. поток вызвал функцию sigwaitinfo()); STACK – поток ожидает выделения виртуального адресного пространства для своего стека (родительский поток вызывает функцию ThreadCreate()); STOPPED – поток блокирован и ожидает сигнал SIGCONT; WAITCTX – поток ожидает доступности нецелочисленного контекста (например, с плавающей запятой); WAITPAGE – поток ожидает выделения физической памяти для виртуального адреса; WAITTHREAD – поток ожидает завершения создания дочернего потока (т. е. поток вызвал функцию ThreadCreate())

**Лекция 5**

Планирование потоков

При каждом вызове ядра, исключении или аппаратном прерывании, в результате которого управление передается ядру QNX, выполнение активного потока временно приостанавливается. Действие планирования совершается в момент изменения состояния любого потока независимо от того, в каком процессе поток расположен. Планирование потоков осуществляется по всем процессам сразу

Активный поток блокируется, если он должен ожидать какое-либо событие (например, ответ на запрос механизма обмена сообщениями, освобождение мутекса и т. д.). Блокированный поток удаляется из очереди готовности (ready queue), после чего запускается поток с наивысшим приоритетом. Когда блокированный поток разблокируется, он помещается в конец очереди готовности на соответствующий приоритетный уровень.

Активный поток блокируется, если он должен ожидать какое-либо событие (например, ответ на запрос механизма обмена сообщениями, освобождение мутекса и т. д.). Блокированный поток удаляется из очереди готовности (ready queue), после чего запускается поток с наивысшим приоритетом. Когда блокированный поток разблокируется, он помещается в конец очереди готовности на соответствующий приоритетный уровень.

Активный поток самостоятельно освобождает процессор (sched\_yield()) и помещается в конец очереди готовности на данном уровне приоритета. После этого запускается поток с наивысшим приоритетом (в том числе им может быть поток, который только что отдал управление).

Каждому потоку назначается свой приоритет. Планировщик выбирает поток для выполнения в соответствии с приоритетом каждого потока, находящегося в состоянии готовности (READY).

Всего в ОС QNX Neutrino поддерживается до 256 уровней приоритетов. Приоритет выполнения каждого непривилегированного (nonroot) потока может изменяться в пределах от 1 до 63 (наивысший приоритет), независимо от его дисциплины планирования (scheduling policy). Только привилегированные (root) потоки (т. е. потоки, действующий uid которых равен 0) могут иметь приоритет выше 63.

Для работы с различными приложениями в ОС QNX Neutrino используются следующие алгоритмы планирования: • FIFO-планирование; • циклическое планирование (планирование по круговому алгоритму) (round-robin scheduling); • спорадическое планирование (sporadic scheduling).

FIFO-планирование и циклическое планирование применяются только в случаях, когда два или более потоков, имеющих одинаковый приоритет, находятся в состоянии готовности.

FIFO-планирование В алгоритме FIFO-планирования поток продолжает выполняться до тех пор, пока он: • принудительно не освободит ресурсы управления (например, при блокировке); • не будет вытеснен потоком с более высоким приоритетом.

Циклическое планирование В алгоритме планирования циклического типа поток продолжает выполняться до тех пор, пока он: • принудительно не освободит ресурсы управления; • не будет вытеснен потоком с более высоким приоритетом; • не израсходует свой квант времени (timeslice).

Алгоритм спорадического планирования обычно используется для задания верхнего лимита на время выполнения потока в пределах заданного периода времени. Этот метод необходим при выполнении монотонного частотного анализа (Rate Monotonic Analysis) системы, обслуживающей как периодические, так и апериодические события. По сути, данный алгоритм позволяет потоку обслуживать апериодические события, не препятствуя своевременному выполнению других потоков или процессов в системе.

Для управления этим спорадическим переходом используются следующие параметры: • начальный бюджет потока (initial budget) (C) —время, за которое поток может выполняться с нормальным приоритетом (N), перед тем как получить пониженный приоритет (L); • пониженный приоритет (low priority) (L) — приоритетный уровень, до которого приоритет потока будет снижен. При пониженном приоритете (L) поток выполняется в фоновом режиме. Если поток имеет нормальный приоритет (N), он выполняется с приоритетом переднего плана; • период пополнения (replenishment period) (T) — период времени, в течение которого поток может расходовать свой бюджет выполнения (execution budget). • максимальное число текущих пополнений (max number of pending replenishments) — это значение устанавливает ограничение на количество выполняемых операций пополнения, тем самым ограничивая объем системных ресурсов, выделяемых на дисциплину спорадического планирования.

Наиболее простыми из служб синхронизации являются мутексы. Мутекс (от англ. mutex, mutual exclusion lock) служит для обеспечения монопольного доступа к данным, которые совместно используются несколькими потоками. Операциями захвата мутекса (с помощью функции pthread\_mutex\_lock()) и освобождения мутекса (с помощью функции pthread\_mutex\_unlock()) обычно обрамляются участки кода, который обращается к совместно используемым данным (обычно это критическая секция кода)

Если поток, имеющий более высокий приоритет, чем владелец мутекса, пытается захватить мутекс, то действующий приоритет текущего владельца устанавливается равным приоритету блокированного потока, ожидающего мутекс. Владелец вернется к своему исходному приоритету после того, как он освободит мутекс. Такая схема не только обеспечивает минимальное время ожидания мутекса, но и решает классическую проблему инверсии приоритетов.

Семафоры

Еще одним средством синхронизации являются семафоры (semaphores), которые позволяют потокам увеличивать (с помощью функции sem\_post()) или уменьшать (с помощью функции sem\_wait()) значение счетчика на семафоре для управления блокировкой потока (операции "post" и "wait" соответственно).

Условная переменная (condvar – сокр. от condition variable) используется для блокировки потока по какому-либо условию во время выполнения критической секции кода. Условие может быть сколь угодно сложным и не зависит от условной переменной. Однако условная переменная всегда должна использоваться совместно с мутексом для проверки условия.

Барьер — это механизм синхронизации, который позволяет скоординировать работу нескольких взаимодействующих потоков (например, при матричных вычислениях) таким образом, чтобы каждый из них остановился в заданной точке в ожидании остальных потоков, прежде чем продолжить свою работу.

Ждущие блокировки (sleepon locks) работают аналогично условным переменным, за исключением некоторых деталей. Как и условные переменные, ждущие блокировки (pthread\_sleepon\_lock()) могут использоваться для блокировки потока до тех пор, пока условие не станет истинным (аналогично изменению значения ячейки памяти). Но в отличие от условных переменных (которые должны существовать для каждого проверяемого условия), ждущие блокировки применяются к одному мутексу и динамически создаваемой условной переменной независимо от количества проверяемых условий..

Службы обмена сообщениями (Send/Receive/Reply), используемые в ОС QNX Neutrino, осуществляют неявную синхронизацию посредством блокировок. Во многих случаях они могут заменить собой другие службы синхронизации. Кроме того, службы обмена сообщениями — единственные примитивы синхронизации и межзадачного взаимодействия (кроме именованных семафоров, основанных на механизме обмена сообщениями), которые могут работать в сети.