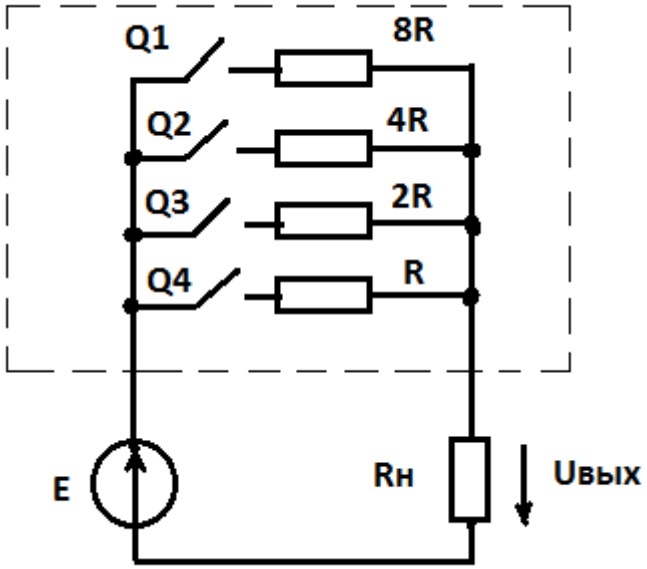
Цифро-аналоговое и аналого-цифровое преобразование (ЦАП и АЦП)

Цифро-аналоговые преобразователи (ЦАП) служат для преобразования информации из цифровой формы в аналоговый сигнал – суммирование токов и напряжений. Принцип работы ЦАП состоит в суммировании аналоговых сигналов, пропорциональных весам разрядов входного цифрового кода, с коэффициентами, равными нулю или единице в зависимости от значения соответствующего разряда кода.



ЦАП преобразует цифровой двоичный код $Q_4Q_3Q_2Q_1$ в аналоговую величину, обычно напряжение $U_{вых.}$. Каждый разряд двоичного кода имеет определенный вес i-го разряда вдвое больше, чем вес (i-1)-го. Работу ЦАП можно описать следующей формулой:

$$\mathbf{U_{вых}=e*(Q_1\ 1+Q_2*2+Q_3*4+Q_4*8+\ldots),\ (1)}$$

где e - напряжение, соответствующее весу младшего разряда, Q_i - значение i -го разряда двоичного кода (0 или 1).

Например, числу 1001 соответствует

$$\mathbf{U_{вых}=y*(1*1+0*2+0*4+1*8)=9*e,\ а\ числу\ 1100}$$

$$\mathbf{U_{вых}=e*(0*1+0*2+1*4+1*8)=12*e.}$$

Аналогово-цифровые преобразователи.

АЦП последовательного приближения (АЦППП).

Структурная схема АЦППП

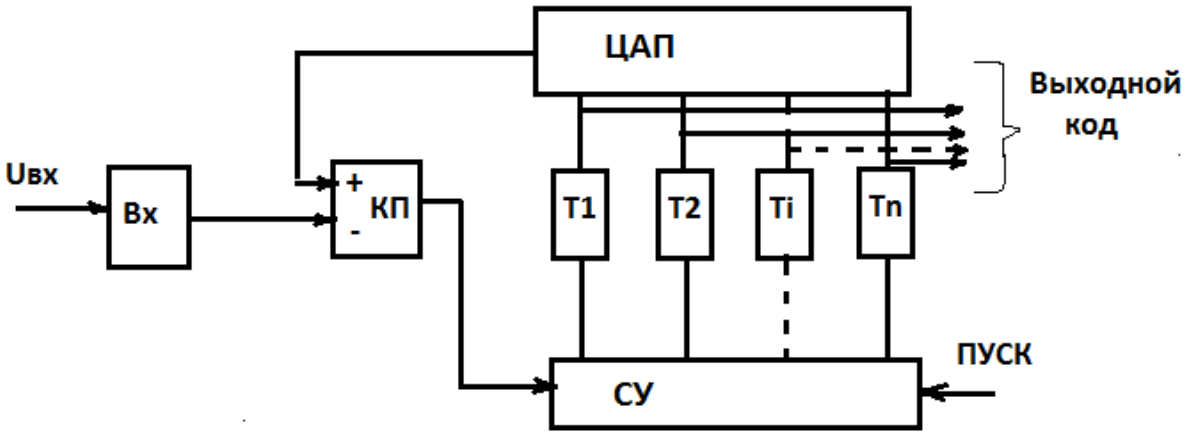
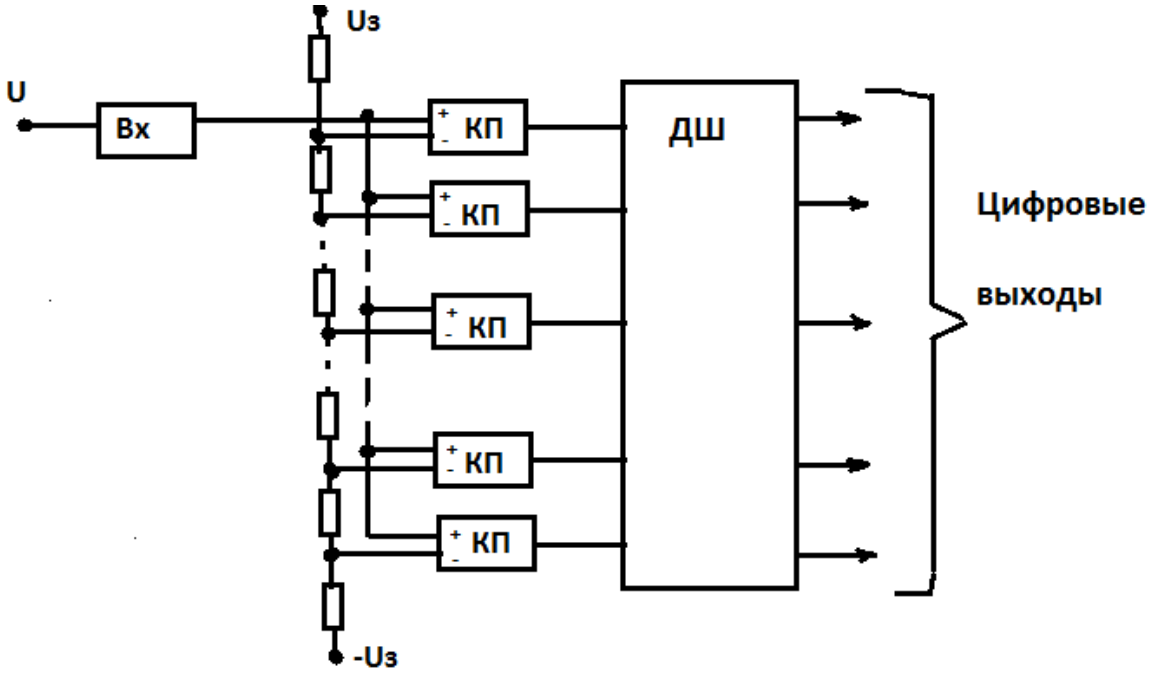


Схема работает следующим образом. Входной аналоговый сигнал Uвх перед началом преобразования запоминается схемой выборки – хранения ВХ. С помощью сдвигового регистра последовательно во времени каждый триггер Ti, начиная со старшего разряда, переводит в положение 1 соответствующий разряд ЦАП. Напряжение U1 (или ток) с выхода ЦАП сравнивается с входным аналоговым сигналом с помощью компаратора КП. Если U0 > U1, на выходе компаратора сохраняется низкий уровень и в триггере сохраняется единица, при U0 < U1 срабатывает компаратор и переводит триггер в положение 0. После окончания цикла на выходах триггеров получается двоичный код, соответствующий (при идеальных элементах) U0 с точностью до половины младшего разряда.

Погрешность АЦППП определяется неточностью ЦАП, зоной нечувствительности и смещением нуля компаратора, а также погрешностью схемы выборки – хранения.

АЦП параллельного типа (АЦППП). Обладает более высоким быстродействием, но сдержит большое количество компараторов.



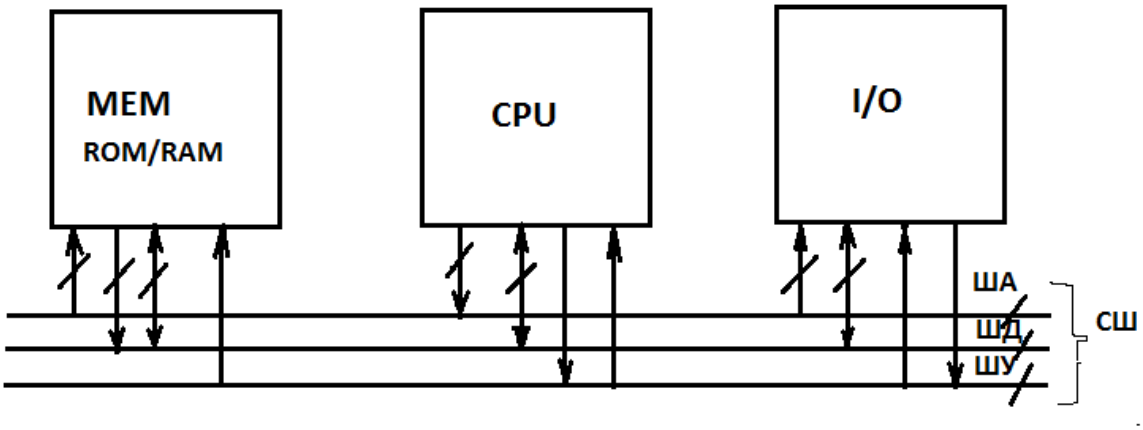
Здесь входная аналоговая величина U_0 с выхода схемы ВХ сравнивается с помощью $2^{n+1} - 1$ компараторов с $2(2^n-1)$ эталонными уровнями, образованными делителями из резисторов равного сопротивления. При этом срабатывают m младших компараторов, образующих на выходах схем И-НЕ нормальный единичный код, затем который с помощью специального кодопреобразователя ДШ преобразуется в двоичный выходной сигнал.

Погрешность АЦПП определяется неточностью и нестабильностью эталонного напряжения, резистивного делителя и погрешностями компараторов. Значительную роль могут играть входные токи компараторов, если делитель недостаточно низкоомный.

Микропроцессоры.

Структура микропроцессорной системы.

Любая микропроцессорная система состоит из трех основных компонентов: собственно микропроцессора, производящего операции над данными, области памяти, где хранятся коды программ работы процессора и обрабатываемые массивы данных, и области устройств ввода-вывода, состоящую из схем, адаптирующих процессорную систему к внешним устройствам. Соединение блоков микропроцессорной системы производится по системной шине (СШ).



Системная шина состоит из трх групп шин:

-шины адреса, на которую процессор выставляет адрес устройства в пространстве памяти или пространстве ввода-вывода, с которым будет поизводиться обмен информацией;

-шины данных, по которой производится обмен информацией;

-шины управления, по которой процессор посылает управляющие сигналы и получает запросы от устройств ввода-вывода.

Шина адреса однонаправлена, информация, следующая по ней, многоразрядна. Шина данных двунаправлена и информация, следующая по ней, также многоразрядна. Шина управления состоит из отдельных проводников, каждый из которых передает определенный сигнал управления. Формировать управляющие сигналы может процессорный блок и блок устройств ввода-вывода (запросы на процессорный блок).

Для полноценного обмена процессора по шине данных важны три момента: направление обмена (чтение/запись), обьект обмена (память/устройства ввода-вывода) и, наконец, структура информации (данные/код).

Устройства памяти.

Внутренняя память микропроцессорной системы по способу доступа к ячейкам накопителя делится на 3 типа: адресная память, память с последовательным доступом и ассоциативная память.

В адресной памяти доступ к любой ячейке накопителя возможен по любому выставленному на шине адресу, независимо от предыдущего обращения.

В памяти с последовательным доступом порядок обращения к ячейкам задается счетчиком адресов, который невозможно переустановить

в процессе работы с памятью. Таким образом, адрес обращения к каждой последующей ячейке отличается от предыдущего всегда на определенную величину.

В ассоциативной памяти ячейка накопителя содержит информацию, скопированную из основной адресной памяти. При этом некоторая часть адреса этой основной памяти, так называемый тег, или признак, также сохраняется в ассоциативной памяти. И, если вызываемый процессором адрес содержит такой тег, информация извлекается из накопителя

ассоциативной памяти.

Пример ассоциативной памяти CACH-память. Данная структура в настоящем курсе не рассматривается.

Память с последовательным доступом.

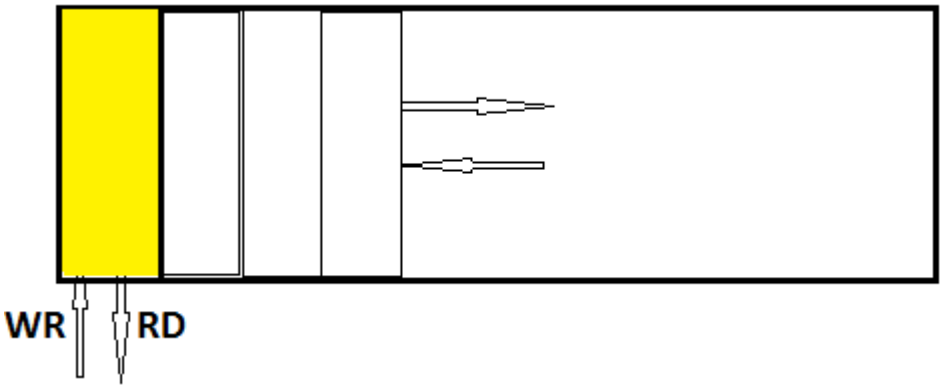
Примеры памяти с последовательным доступом: память **FIFO(first input, first output)** и память **LIFO(last input, first output),** или **stack**.

В FIFO процессы записи и считывания обсчитываются двумя счетчиками адресов. Оба счетчика суммирующие. Для исключения

ложного считывания (чтения ячейки, куда не производилась запись, счетчик записи имеет возможность реверса).

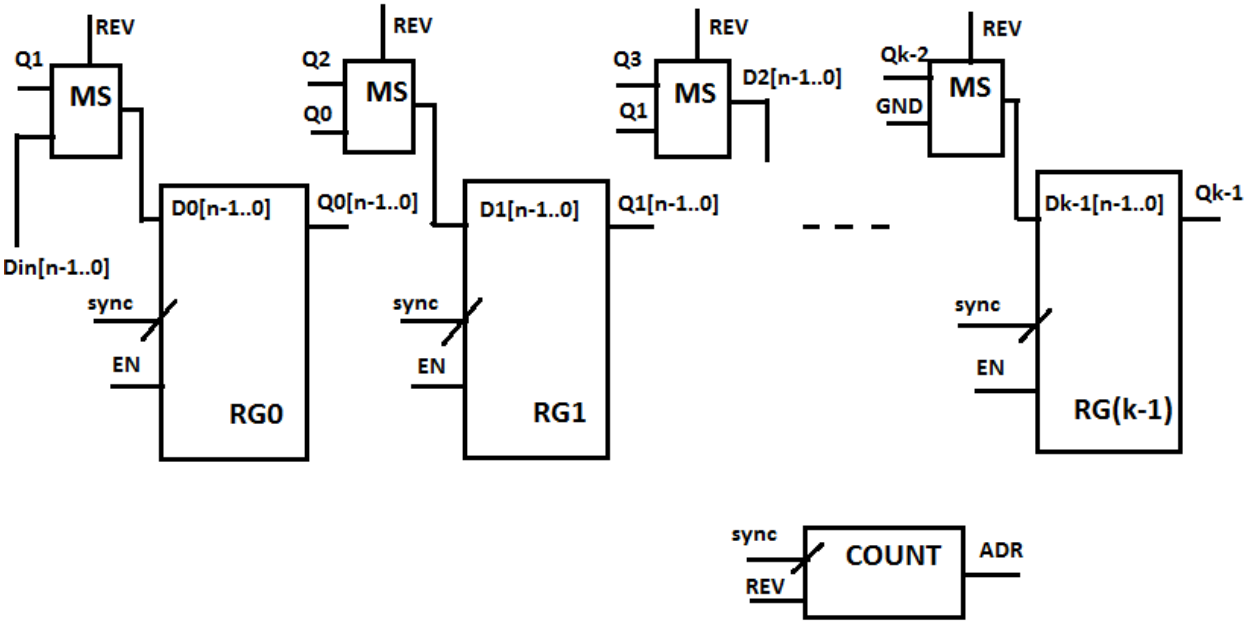
Внешние стрелки показывают информацию, доступную для просмотра.

В LIFO доступна для просмотра только одна точка, поэтому адреса записи и считывания обсчитываются одним реверсивным счетчиком.



Адресуемая ячейка, через которую производится запись и считывание информации называется вершиной стека.

На схеме показана структура n-разрядного стека глубиной k. Вершина стека – RG0.

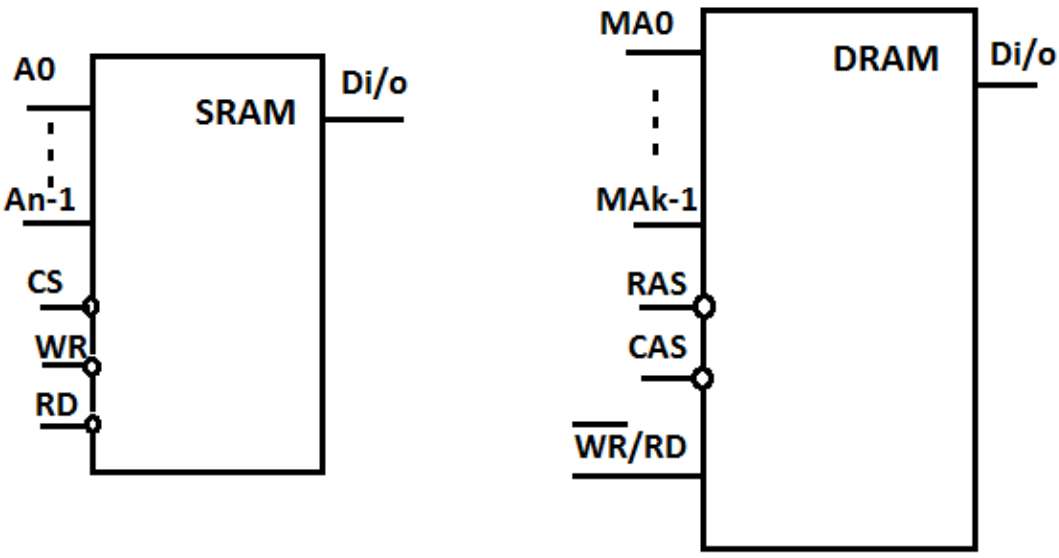


Адресная память.

Примеры адресной памяти: постоянные запоминающие устройства ROM и оперативные запоминающие устройства RAM.

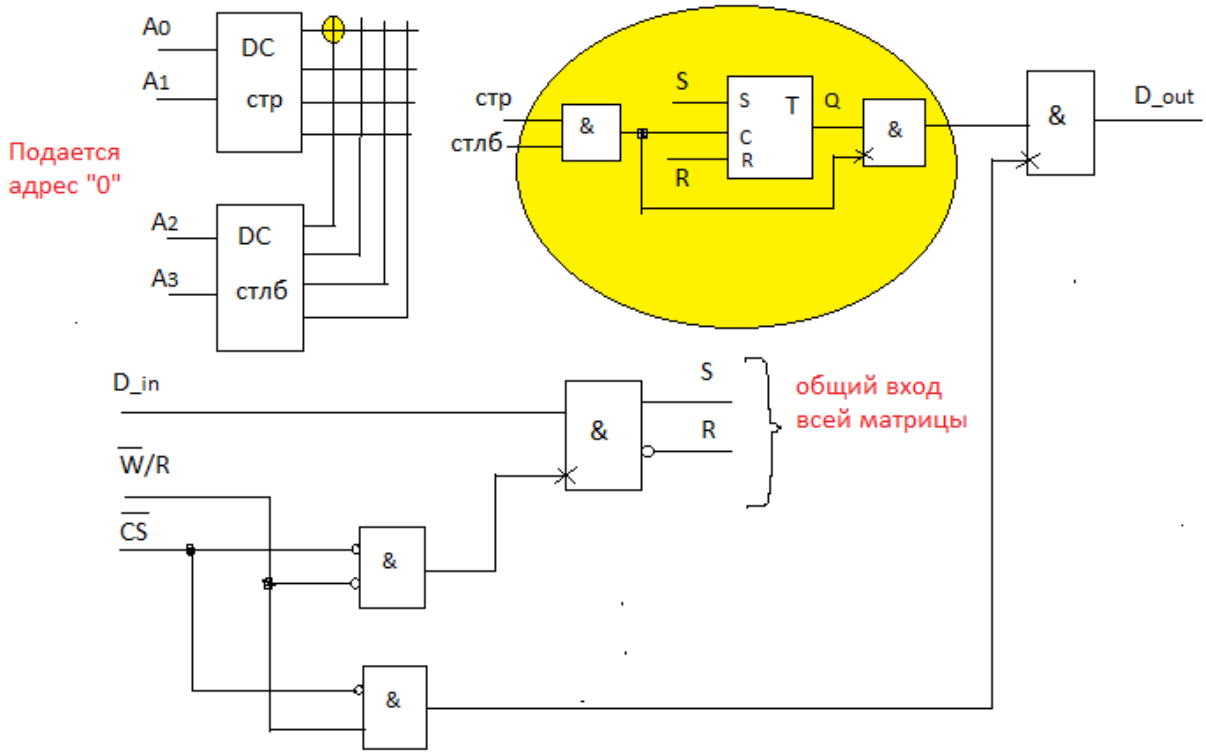
Оперативные запоминающие устройства.

Оперативные запоминающие устройства делятся на 2 класса по структуре матриц накопителя. Это динамические ОЗУ (DRAM), имеющие ячейки накопителя емкостного типа, и статические ОЗУ (SRAM), в которых накопители строятся на основе триггерных ячеек. Обращение к ячейкам накопителя DRAM не может совершаться одновременно по строкам и столбцам из-за инерционности емкости. При считывании с DRAM информация стирается (емкость разряжается) и необходимо время на восстановление информации в ячейке. Кроме того, емкостным ячейкам, не участвующим в процессе обмена, необходима регенерация (поддержание заряда). Такая память обладает низким быстродействием, но имеет низкую себестоимость производства.



Память SRAM, имеющая триггерную матрицу накопителя, наоборот, обладает сверхвысоким быстродействием. Если рассматривать матрицу накопителя, имеющую 2^n ячеек, то в SRAM имеется n адресных входов, распределенных на дешифраторы строки и столбца, вход CS (chip select),

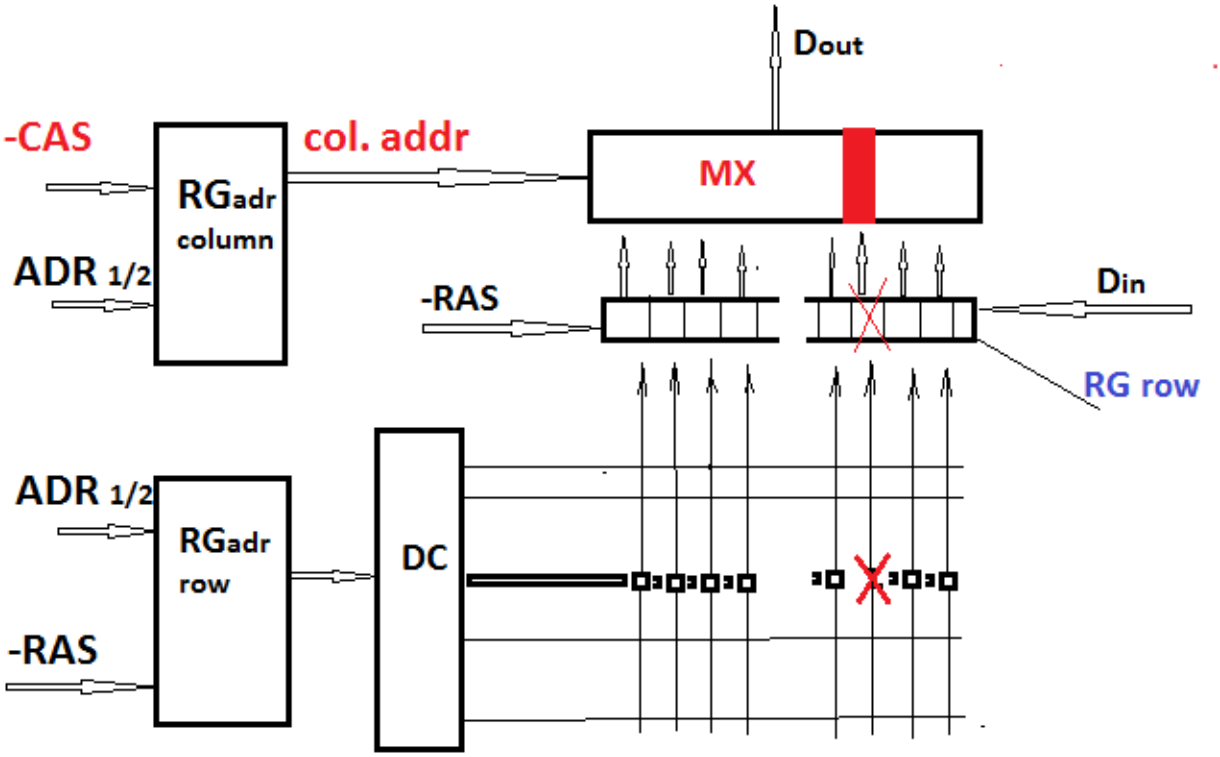
позволяющий подключить именно данный кристалл к шине и входы управления буферами записи и считывания.



На рисунке изображена внутренняя структура схемы SRAM на 4 адресных входа, на которые подаем адрес «0»(0000). Пересечение строки и столбца матрицы накопителя, на которых при этом будут уровни «1» выделено желтым. Видно, что будет активна ячейка синхронного RS-триггера, на вход С которой поступит «1». В ячейку будет записана информация, поступившая на входы S и R всей матрицы. На рисунке видно, как работают входы CS(активный уровень «0») и W/R(для записи активный уровень «0», а для чтения «1»). Получаемые на выходах элементов 2И уровни «1» открывают буфер записи или буфер чтения. Считывание с ячейки накопителя также управляется с элемента 2И на линиях выходов дешифраторов строк и столбцов.

Оперативные запоминающие устройства. DRAM.

В DRAM адресные входы подключаются к шине через мультиплексоры, выделяющие адреса строки и столбца. K=n/2. Вход RAS (row address strobe) (строб строки) активен, когда подается адрес строки, вход CAS (column address strobe) (строб столбца) активен после подачи адреса столбца.



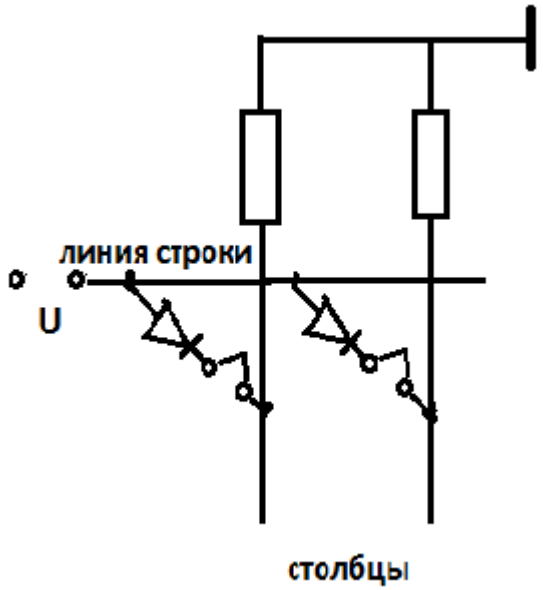
На рисунке изображен процесс поиска ячейки и считывания информации.

При записи строки в защелку строки информация в ячейках разрушается, поэтому во время выдачи информации одновременно идет восстановление строки из защелки. Строки, не выбранные дешифратором при активном RAS, подвергаются регенерации.

Постоянные запоминающие устройства.

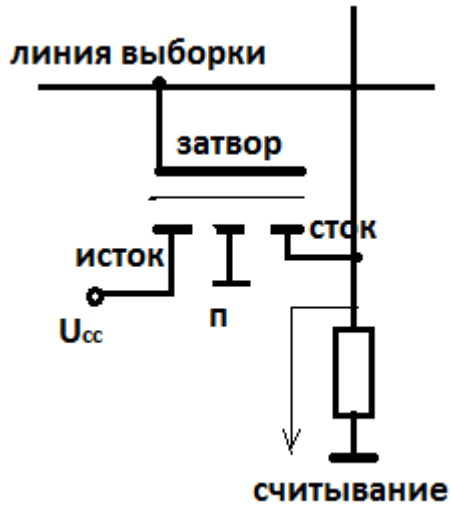
В матрице накопителя постоянного запоминающего устройства строки подключены к выходам дешифратора адреса, а столбцы – к шине данных.

Постоянные запоминающие устройства, имеющие диодные накопители, могут программироваться однократно (ROM, PROM).



Фрагмент матрицы накопителя PROM. В исходном состоянии все узлы в состоянии «1».

Постоянные запоминающие устройства, матрицы накопителей которых построены на полевых транзисторах, возможно многократно перепрограммировать. Это могут быть схемы с ультрафиолетовым стиранием информации (EPROM), или с электрическим стиранием (EEPROM).



Узел матрицы накопителя на основе полевого транзистора.

Основа – МОП-структура с плавающим или двойным затвором.

Плавающий затвор: между затвором и каналом вводится дополнительная область, вызывающая стекание в нее заряда. При снятии

напряжения с затвора заряд сохраняется и удерживает транзистор в запертом состоянии. Пороговое напряжение настолько велико, что поле не создается.

В настоящее время ультрафиолетовое стирание практически не применяется, термин EPROM используют для схем на основе плавающего затвора, которые допускают только полное стирание информации, и имеют значительно меньший ресурс для перезаписи, чем схемы на основе двойного затвора.

Блок CPU.

Производительность процессора. Архитектура процессоров.

Если процессор работает с тактовой частоой F, то время T=1/F называется тактом. Время выполнения тестовой задачи можно рассчитать через такт

T x C x I,

где C – количество тактов на инструкцию, а I – количество инструкций на задачу.

Соответственно, чем меньше времени затрачивается на решение тестовой задачи, тем производительность процессора выше. В указанном выше выражении уменьшение T ограничено свойствами структуры, поэтому изменение производительности можно достичь изменением I или C.

Рассмотрим две основные архитектуры процессорного ядра. RISC – процессоры (Reduced Instruction Set Computer) и CISC - процессоры (Complete Instruction Set Computer).

Любой тип процессора выполняет инструкции, непрерывным потоком поступающие из памяти по шине данных. Выполнение инструкции можно разбить на 5 этапов:

1 – выборка кода из памяти по выставленному на адресной шине адресу,

2 – дешифрация кода,

3 – исполнение,

4 – получение результата,

5 – обратная загрузка результата.

Для ускорения процесса работа производится **конвейерным** способом, т.е. в каждый момент времени одновременно выполняются разные этапы следующих подряд команд.

Рассмотренный выше случай – пятиступенчатый конвейер, но для разных процессоров возможно объединение 4 и 5 или 3, 4 и 5 этапов, в этих случаях мы имеем четырех- или трехступенчатый конвейер.

Для CISC – процессоров характерны сложные многотактовые инструкции, производители этих процессоров старались увеличить производительность засчет уменьшения I. Но это приводило к приостановке конвейера, а, следовательно, снова снижало производительность процессора.

RISC – процессоры выполняют простые однотактовые операции. Они, в отличие от CISC не могут выполнять сложные задачи, зато для них C = 1,а так как операции обмена с пространством памяти в RISC выделены в отдельную группу, конвейер работает практически безостановочно и производительность высока.

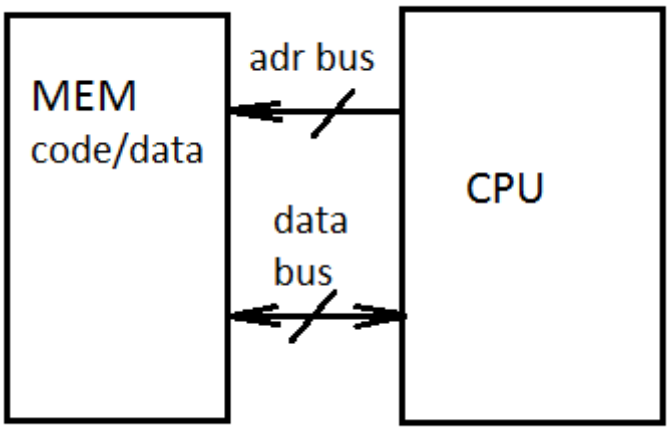
В настоящее время классические структуры в их первоначальном виде уже не используются. Процессорные системы строятся, в основном, на основе модифицированных RISC-ядер.

Типы архитектуры микропроцессорных систем.

В настоящее время существуют два типа архитектуры микропроцессорных систем – Принстонская, или архитектура фон-Неймана и Гарвардская.

В 1945 г. американский математик Джон фон Нейман сформулировал основные принципы работы современных компьютеров. Им была

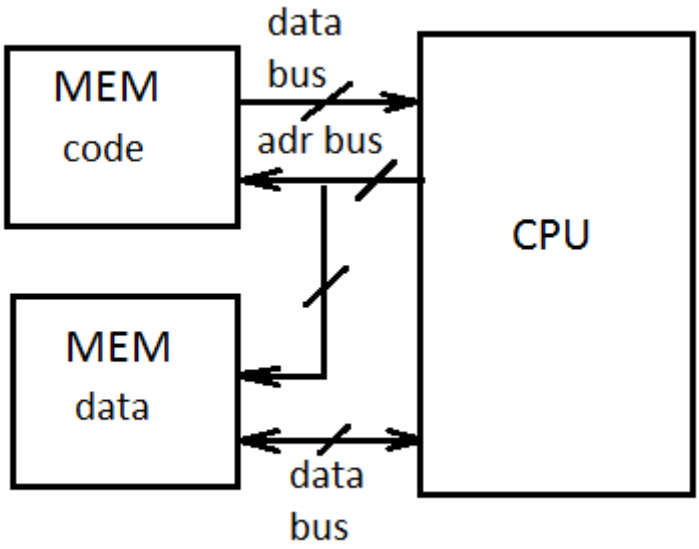
предложена архитектура, получившая его имя (von Neumann architecture) и предполагающая хранение программ и данных в общей памяти (1946 г.).



Такая архитектура наиболее характерна для микропроцессоров, ориентированных на использование в компьютерах. Примером могут служить микропроцессоры семейства x86. Эти микропроцессоры относятся к CISC-процессорам.

Структура ядра CISC-процессора предполагает наличие малого количества регистров общего назначения, к тому же имеющих строго определенные функции. Эти функции обусловлены наличием большого количества указателей и счетчиков, входящих в состав ядра и позволяющих выполнять циклические операции, записанные в одной инструкции. При выполнении операций в АЛУ процессор может пользоваться операндами как хранящимися в регистрах общего назначения, так и в пространстве памяти, выделенном под данные. Таким образом, для написания программ под CISC=процессор используется большее количество адресаций данных, чем при программировании под RISC-процессор. Команды CISC-процессора выполняются за несколько тактов, что позволяет не выстраивать коммутационные КЦУ в цепочку, а использовать один коммутатор и служебный регистр (например, в командах пересылок).

Архитектура, предполагающая раздельное использование памяти программ и данных, носит название гарвардской (Harvard architecture). Гарвардская архитектура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность.



Такая архитектура была создана под использование RISC-процессоров.

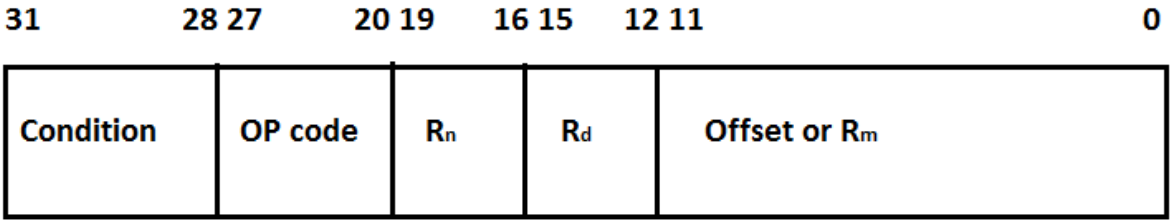
Работа микропроцессорной системы происходит под действием команд языка низкого уровня (Ассемблера). Структура команды представлена адресами блоков, задействованных в операции, адресами источников и приемников информации. Кроме того, в команде могут содержаться операнды. Операнды могут быть данными, необходимыми для загрузки в регистр, или адресами (прямыми, или смещениями).

Любой микропроцессор состоит из нескольких блоков. Условно их можно назвать:

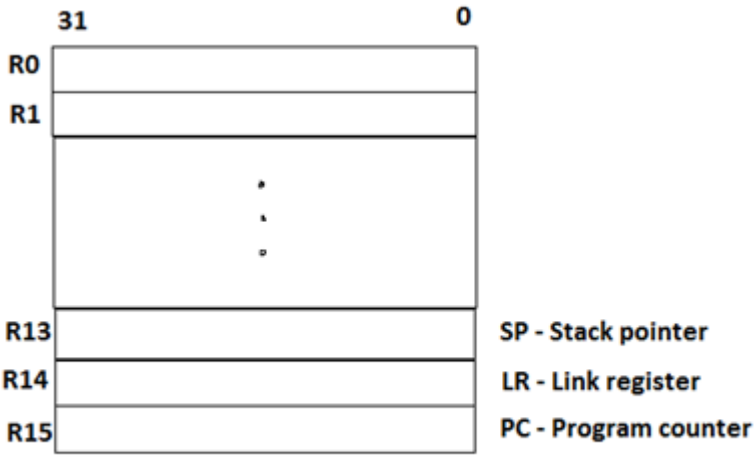
1. – блок очереди команд,
2. – блок дешифрации команд,
3. – блок регистров (внутренняя память).
4. – блок производства операций, включающий: АЛУ, умножитель, делитель, сдвиговый регистр.

Как мы уже рассматривали, из блока памяти команды поступают в процессор непрерывным потоком. На входе в процессор с шины данных команды записываются в регистр очереди. Такой регистр содержит 3 команды одновременно, такой формат является оптимальным, так как программы обычно построены по разветвленным алгоритмам.

При считывании с регистра очереди команды дешифрируются, т.е. адресная информация, содержащаяся в каждом поле формата, поступает на свой дешифратор для выработки сигналов управления.



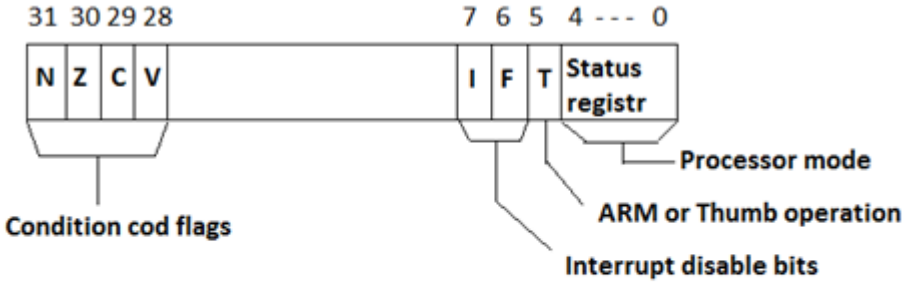
Структура ядра RISC-процессора предполагает наличие большой внутренней памяти, состоящей из регистров общего назначения, не имеющих дополнительных специальных функций. В некоторых семействах микропроцессоров исключение составляет так называемый «регистр нуля». Этот регистр не предназначен для записи информации, в нем хранится «0». Кроме того, в общее число регистров общего назначения входят указатель стека указатель связей и программный счетчик. В этих регистрах всегда записываются адреса: в указатель стека – адрес вершины стека в исполняемой программе, а в указатель связей – адрес выхода из основной программы в вызванную область для возможности возврата.



С помощью команд прямой и обратной загрузки происходит обмен между регистрами общего назначения и памятью данных.

АЛУ – арифметико-логическое устройство. Содержит сумматор и простую комбинаторную логику. Связано с регистром флагов.

Регистр флагов ARM-процессора



CPSR - Current Program Status Register

Различают флаги состояний и системные флаги.

Флаги состояний характеризуют состояние результата операции.

N(S) – знак.

Z – ноль,

C – перенос,

V – переполнение (overflow).

Флаги состояний устанавливаются командами, выполняемыми в операционном блоке (АЛУ), но только теми, которые для этого определены.

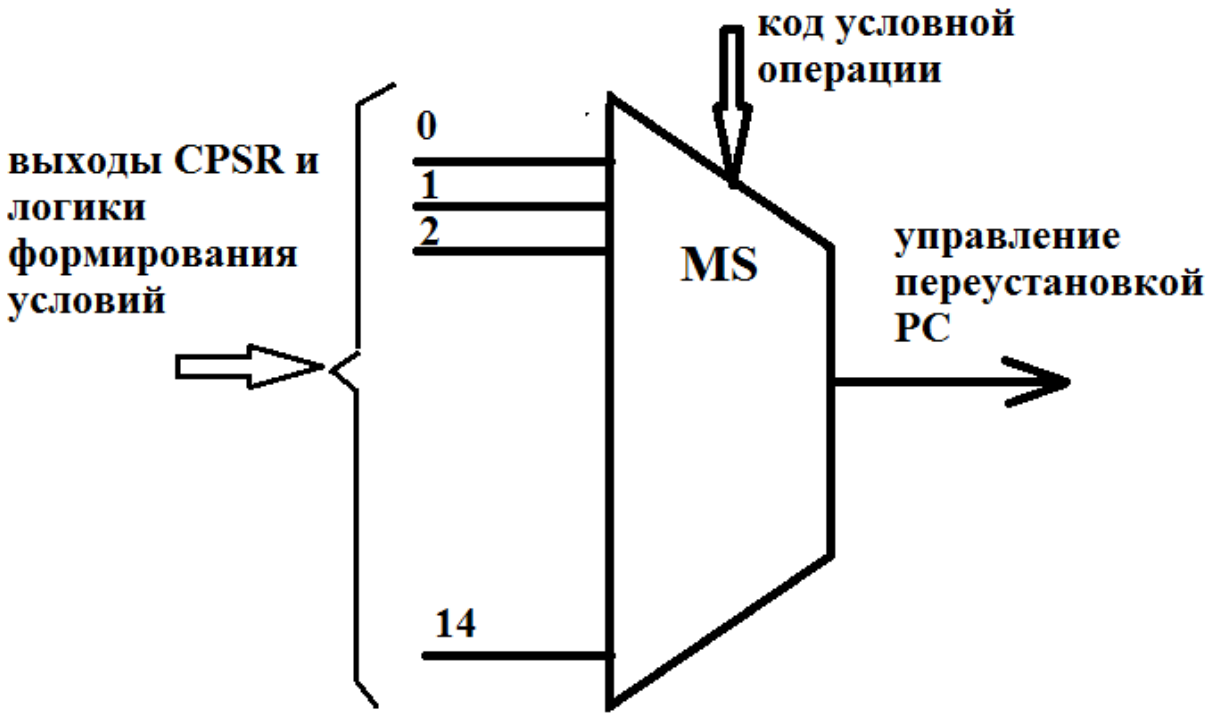
Флаги состояний используются командами условных действий.

Таблица условий.

Код условия	Обозначение условия	Наименование условия	Комбинация флагов
0	EQ	Равно (ноль)	Z = 1
1	NE	Не равно (не ноль)	Z = 0
2	CS/HS	Перенос (беззнаковое больше или такое же)	C = 1
3	CC/LO	Нет переноса (беззнаковое меньше)	C = 0
4	MI	Знак минус	N = 1
5	PL	Знак плюс	N = 0
6	VS	Переполнение	V = 1
7	VC	Нет переполнения	V = 0
8	HI	Беззнаковое больше	nC v Z = 0
9	LS	Беззнаковое меньше или такое же	nC v Z = 1
10	GE	Знаковое больше, чем или равно	N + V = 0
11	LT	Знаковое меньше, чем	N + V = 1
12	GT	Знаковое больше, чем	Z v (N + V) = 0
13	LE	Знаковое меньше, чем или равно	Z v (N + V) = 1

14	AL	Все условия	

Очевидно, что выходы регистра флагов (условий) через логические цепочки подключаются к входам мультиплексора, а команда условного действия в своем коде содержит адрес необходимого входа.



При работе команд, использующих АЛУ (арифметико-логическое устройство), операнды поступают только из внутренней памяти ядра, что экономит время обработки. Команды RISC-процессора, в основном, выполняются за один такт, все КЦУ, участвующие в процессе обработки, как коммутирующие, так и кодопреобразующие образуют цепочки и имеют общую задержку, не превышающую время такта. Команды пересылок между регистрами выполняются как арифметическая операция сложения с 0.

Системные флаги описывают характеристики системы. Это может быть режим работы системы, возможность совершения прерывания, или указание формата используемых команд.

Разряды 7 и 6 CPSR указывают на возможность прерываний от внешних источников (IRQ – по обычным линиям запроса, FIQ – по скоростным).

Установка «1» при невозможности обслуживания запроса.

5-й разряд указывает, какое кодирование применяется в исполняемых операциях. Возможно ARM-кодирование с 32-разрядными инструкциями и Thumb – кодирование с 16-разрядными. Наибольшее применение получил синтаксис Thumb-2. В нем смешанный формат инструкций.

Код, записанный в младших разрядах, определяет способ функционирования процессорной системы.

	CPRS$_{4-0}$	Operating Mode
1.	10000	User
7.	10001	FIQ
6.	10010	IRQ
3.	10011	Supervisor
4.	10111	Abort
5.	11011	Undefined
2.	11111	System

1. User – основной способ работы для прикладных программ. Способ непривилегированный, имеет ограниченный обмен с системными ресурсами.
2. System – открывает полный доступ к системным ресурсам.
3. Supervisor – включается, когда программное прерывание делает невозможным дальнейшее исполнение программы (аварийное завершение). Кроме того, включается при сбросе или выключении питания.
4. Abort – включается при попытке программы осуществить обмен с запрещенным участком памяти.
5. Undefined – включается при заявлении несуществующей инструкции.
6. IRQ – режим реагирования на обычные запросы на прерывания о внешних источников.

7. FIQ – режим реагирования на запросы от внешних устройств на прерывание по скоростным линиям запросов. Используется для обслуживания запросов, требующих немедленного обслуживания.

Прерывание – это процесс, позволяющий приостановить выполнение основной программы на время выполнения подпрограммы.

Прерывания: аппаратные, программные и исключительные ситуации. Исключительные ситуации также можно отнести к программным прерываниям, но они происходят на этапе компиляции или отладки.

Первые действия процессора при поступлении сигнала о прерывании:

PC ➔LR (banked);

CPSR ➔SPSR (banked);

Changes bits 4—0 of CPSR, sets **I** and **F**;

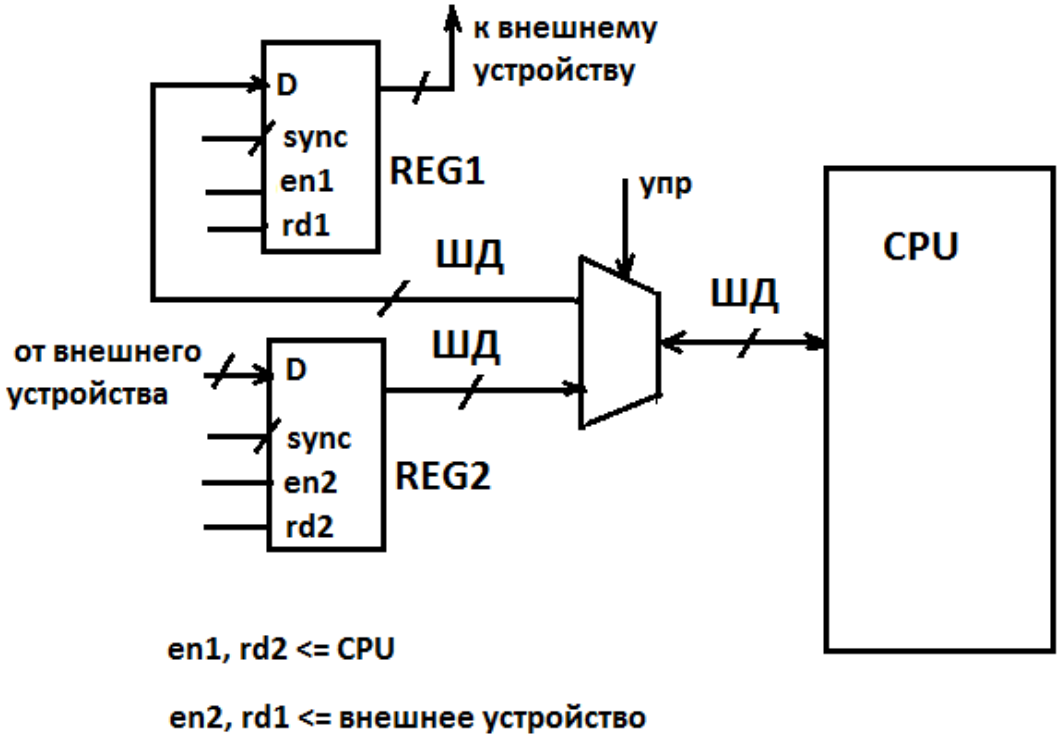
Vector address ➔ PC.

Таблица векторов прерываний

Address	Exception	Priority	Mode entered
0x000	Reset	1	Supervisor
0x004	Unimplemented instruction	6	Undefined
0x008	Software interrupt	-	Supervisor
0x00C	Instruction access violation	5	Abort
0x010	Data access violation	2	Abort
0x018	IRQ	4	IRQ
0x01C	FIQ	3	FIQ

Устройства ввода-вывода.

Основной принцип ввода и вывода информации основан на взаимодействии регистров.



Если внутри МПС передача информации производится параллельным способом, то все внешние передачи осуществляются последовательно.

Основные последовательные интерфейсы.

UART (USART), USB – высокоскоростные,

SPI, I2C - низкоскоростные.

UART (прототип RS232).

Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной **скоростью UART** и для конкретного соединения указывается в бодах (что в данном случае соответствует битам в секунду). Существует

общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод.

Основные регистры данных: регистр приема и регистр передачи.

Кроме регистров приема и передачи имеет два адресуемых делителя частоты (старший и младший байты адресуются отдельно), регистры управления линией и модемом, регистры состояния линии и модема, регистр разрешения прерываний и регистр-идентификатор прерываний.

Делители служат для хранения констант, изменяющих коэффициент деления тактовой частоты, чтобы обеспечить определенную скорость передачи.

Перед началом работы необходимо записать управляющее слово по адресу регистра управления линией. В формате управляющего слова определяется:

1) - доступ к регистрам приема/передачи или к регистрам выбора скорости;

2) - нормальная передача символов или старт (рассоединение);

3) – наличие контроля и тип контроля (паритет, непаритет);

4) - количество стоповых бит;

5) - количество разрядов в символе.

1. Первое управляющее слово всегда имеет в старшем разряде «1», что дает обращение к регистрам-делителям. После установки необходимой скорости записывается управляющее слово, имеющее в старшем разряде «0». Оно и будет определять регламент работы приема и передачи.
2. В следующем разряде записывается «1 » при рассоединении и ожидании старта. В штатном режиме записывается «0».
3. Далее три разряда (5,4 и 3) служат для определения контроля. Если в р3 записан «0», то контроль отсутствует, и состояние остальных разрядов не имеет смысла. Если в р3 записана «1», то значение р5 и р4 будут обозначать следующее:

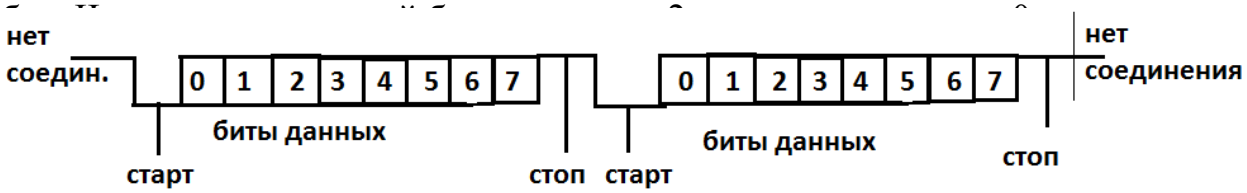
P5	P4	Тип контроля
0	0	Дополнение до чета
0	1	Дополнение до нечета
1	0	Контроль четности
1	1	Контроль нечетности

Многие реализации UART имеют возможность автоматически контролировать целостность данных методом контроля битовой чётности. Когда эта функция включена, последний бит данных в минимальной посылке («бит чётности») контролируется логикой UART и содержит информацию о чётности количества единичных бит в этой минимальной посылке. Различают контроль на четность (англ. *Even parity*), когда сумма количества единичных бит в посылке является четным числом, и контроль на нечетность (англ. *Odd parity*), когда эта сумма нечетна. При приеме такой посылки UART может автоматически контролировать бит четности и выставлять соответствующие признаки правильного или ошибочного приема.

Контроль чётности (пример для 7-битной посылки)

данные	**количество единичных бит**	**Бит четности**	
		even	**odd**
0000000	**0**	**0**	**1**
1010001	**3**	**1**	**0**
1101001	**4**	**0**	**1**
1111111	**7**	**1**	**0**

4. Следующий разряд содержит информацию о количестве стоповых



Помимо собственно информационного потока, UART автоматически вставляет в поток синхронизирующие метки, так называемые **стартовый и стоповый биты**. При приёме эти лишние биты удаляются из потока. Обычно стартовый и стоповый биты обрамляют один байт информации (8 бит), однако встречаются реализации UART, которые позволяют передавать по 5, 6, 7, 8 или 9 бит. Обрамленные стартом и стопом биты являются минимальной посылкой. Некоторые реализации UART позволяют вставлять два стоповых бита при передаче для уменьшения вероятности рассинхронизации приёмника и передатчика при плотном трафике. Приёмник игнорирует второй стоповый бит, воспринимая его как короткую паузу на линии.

Принято соглашение, что пассивным (в отсутствие потока данных) состоянием входа и выхода UART является логическая 1. Стартовый бит всегда логический 0, поэтому приёмник UART ждёт перепада из 1 в 0 и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Если в этот момент на входе всё ещё 0, то запускается процесс приёма минимальной посылки. Для этого приёмник отсчитывает 9 битовых длительностей подряд (для 8-битных данных) и в каждый момент фиксирует состояние входа. Первые 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда 1, если реально принятое значение иное, UART фиксирует ошибку.

5. Разряды 0 и 1 определяют количество разрядов в символе (от 5 до 8).

Поскольку синхронизирующие биты занимают часть битового потока, то результирующая **пропускная способность** UART не равна скорости соединения. Например, для 8-битных посылок формата 8-N-1 синхронизирующие биты занимают 20 % потока, что для физической

скорости 115 200 бод даёт битовую скорость данных 92 160 бит/с или 11 520 байт/с.

В регистре состояния сообщается о заполненности регистров приема и передачи, об ошибках приема, о прерывании отсоединения, о переполнении регистров приема и передачи.

Регистр разрешения прерывания позволяет определить источник разрешенного прерывания (модем или линия); разрешение, если регистр передачи пуст и разрешение, если получаемые данные не имеют доступа к регистру приема.

Интерфейс *USB* (Universal Serial Bus - Универсальный Последовательный Интерфейс)

Предназначен для подключения периферийных устройств к персональному компьютеру. Позволяет производить обмен информацией с периферийными устройствами на трех скоростях (спецификация *USB 2.0*):

- Низкая скорость (*Low Speed* - LS) - 1,5 Мбит/с;
- Полная скорость (*Full Speed* - FS) - 12 Мбит/с;
- Высокая скорость (*High Speed* - HS) - 480 Мбит/с.

Для подключения периферийных устройств используется 4-жильный кабель: питание +5 В, сигнальные провода *D*+ и *D*-, общий провод.

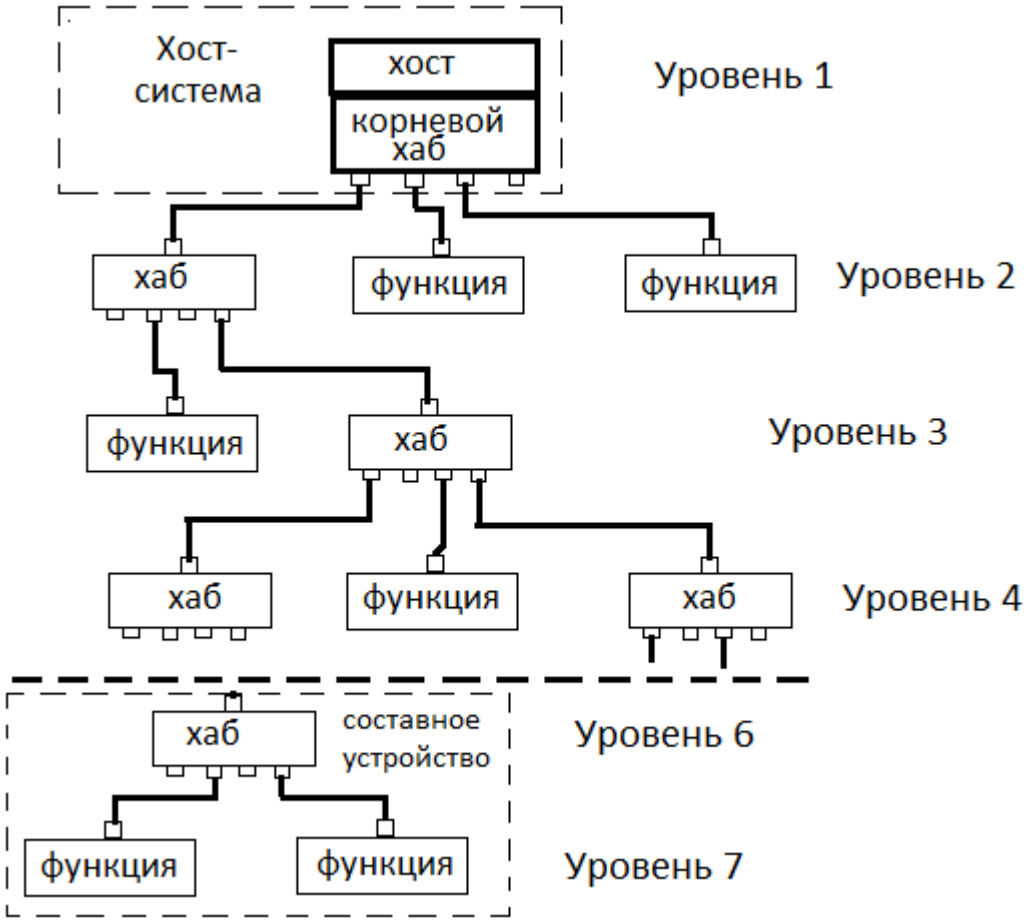
Структура USB

Хост – всегда основное ведущее устройство. (Персональный компьютер.)

Хаб – концентратор. Содержит контроллер и повторитель.

Регистры хаба-контроллера связываются с хостом и обеспечивают опознавание функции, и ее связь с хостом. Хаб-повторитель обеспечивает энергетический режим работы шины.

Функция – оконечное устройство, подключаемое к хосту. Всегда ведомое.



Порт хаба или функции, подключаемый к хабу более высокого уровня, называется восходящим портом (upstream port), а порт хаба, подключаемый к хабу более низкого уровня, или к функции, называется нисходящим портом(downstream port).

Все передачи данных по интерфейсу инициируются хостом, однако приемником или передатчиком может быть как хост, так и функция. Передача пакетная.

В интерфейсе USB испольуется несколько разновидностей пакетов:

- ***пакет-признак*** (*token paket*) описывает тип и направление передачи данных, адрес устройства и порядковый номер конечной точки (КТ - адресуемая часть USB-устройства).
- ***пакет с данными*** (*data packet*) содержит передаваемые данные;
- ***пакет согласования*** (*handshake packet*) предназначен для сообщения о результатах пересылки данных

Типы каналов и пересылок (передач).

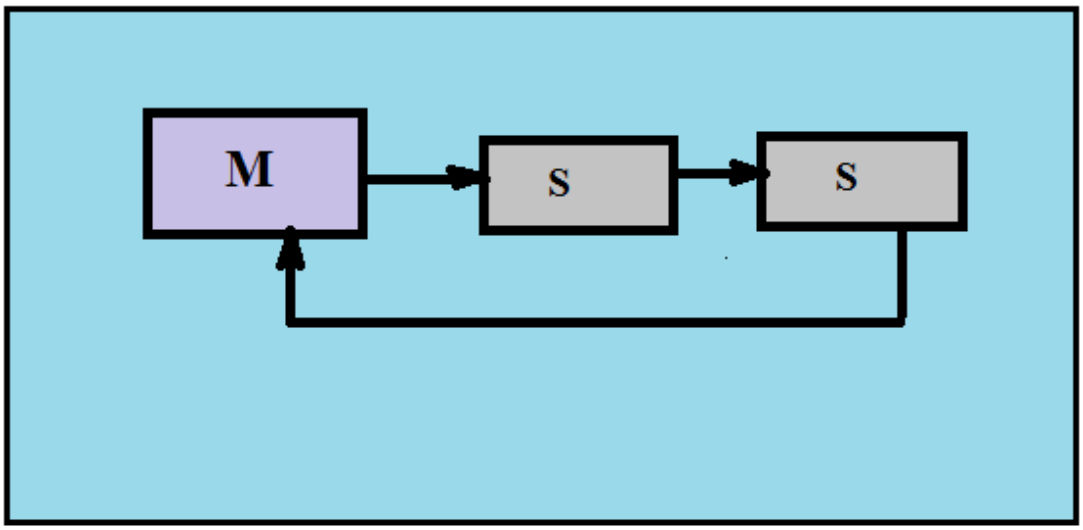
- · Каналы сообщений. Являются двунаправленными каналами. Возникает канал при отсылке хостом запроса в устройства, и управляет передачей только хост. Каналы сообщений используется для передач только управляющего типа.
- · Потоковые каналы. Являются однонаправленными. Эти передачи могут контролироваться не только хостом, но и устройством. Используется для передач данных типа прерывание, групповая пересылка, изохронная.
- В зависимости от типа передаваемых данных, предъявляемых требований к скорости обработки, задержки доставки и т.п. определены следующие типы передач.
- · Управляющие передачи. Используются для конфигурирования устройств во время подключения и выполнения других специфических функций над устройством, включая организацию новых каналов.
- · Прерывания. Используются для спонтанных, но гарантированных передач с гарантированными скоростями и задержками. Используются обычно для передачи введенных данных от клавиатуры или сведений об изменении положения указателя мыши, в устройствах обратной связи, и т.д.
- · Групповая пересылка. Используется для гарантированной передачи данных больших объемов без предъявленных требований к скоростям и задержкам. Занимает под себя всю свободную пропускную способность шины. В любой момент доступная полоса может быть урезана при необходимости осуществления передач других видов с более высоким приоритетом, или добавлена, при освобождении другими устройствами. Обычно такие передачи используется между принтерами, сканерами, накопителями и др.
- · Изохронные передачи. Используются для потоковых передач данных в реальном времени. Резервируют определенную полосу пропускания шины, гарантируют определенные величины задержек доставки, но не гарантируют доставку (в случае обнаружения

ошибки повторной передачи не происходит). Передачи этого вида используются для передачи аудио и видео трафика.

- **Приоритеты передач по USB-шине**

- Все операции по передаче данных инициируются хост-системой независимо от того, принимает ли она данные или пересылает в периферийное устройство. Все операции, которые необходимо выполнить, хранятся в виде четырех списков по типам передач:
- - изохронные передачи;
- - передачи прерываний;
- - передачи управляющих команд;
- - передачи данных больших объемов.

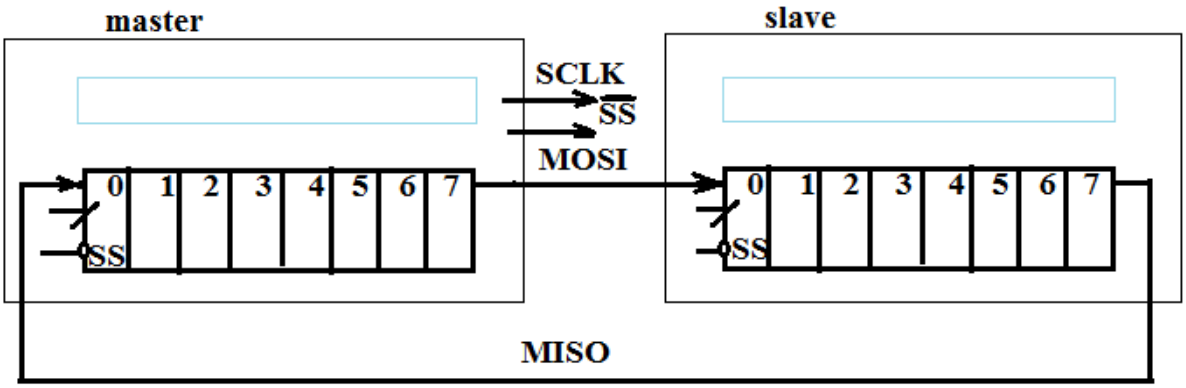
К устройствам, поддерживающим обмен внутри процессорной системы, относятся шины SPI и I2C.



\

Интерфейс (шина) SPI.

Последовательный периферийный интерфейс (шина) служит для обмена информацией между микросхемами. Построение шины основано на строгой иерархии «ведущий – подчиненные». В качестве ведущего обычно выступает микроконтроллер, а подчиненными могут быть различные схемы памяти, таймеры, АЦП и ЦАП и т.п. Организация представляет собой цепь из последовательных регистров, выделяемых в пространстве объединяемых микросхем.



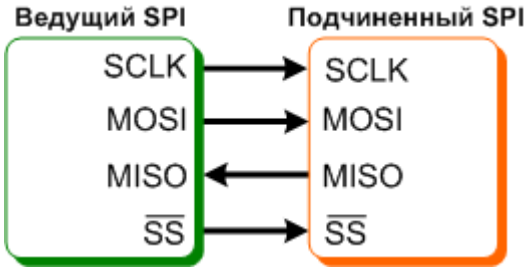
Для выбора ведомого ведущее устройство посылает в его направлении низкий уровень по линии slave select (SS).

Передача и прием ведутся одновременно, пакетами. Чаще всего длина пакета составляет 8 бит, но это не является обязательным условием. Сдвиг производится по тактовой частоте, генерируемой ведущим устройством. Ведомые устройства используют синхросигнал для определения моментов изменения битов на линии данных, при этом ведомые устройства никак не могут влиять на частоту следования битовых интервалов. Как в ведущем устройстве, так и в ведомом устройстве имеется счетчик импульсов синхронизации (битов). Счетчик в ведомом устройстве позволяет последнему определить момент окончания передачи пакета. Счетчик сбрасывается при выключении подсистемы SPI, такая возможность всегда имеется в ведущем устройстве. В ведомом устройстве счетчик обычно сбрасывается деактивацией интерфейсного сигнала SS.

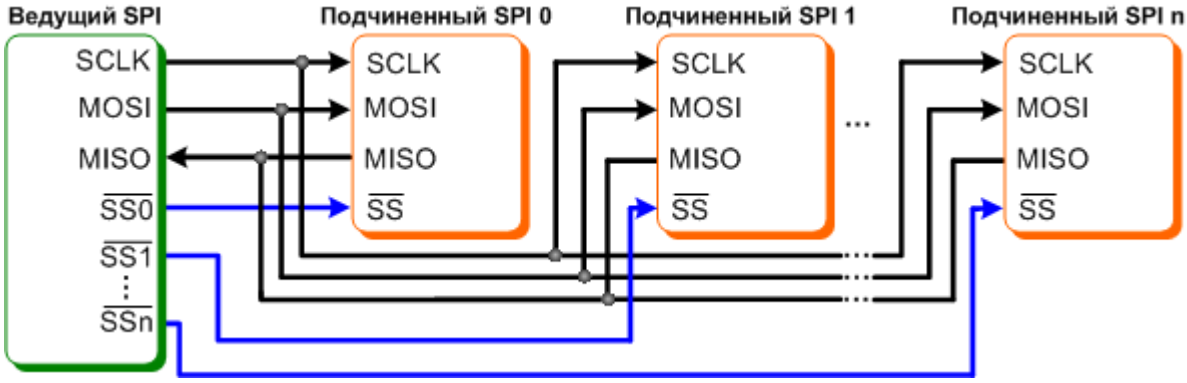
Так как действия ведущего и ведомого устройства тактируются одним и тем же сигналом, то к стабильности этого сигнала не предъявляется никаких требований, за исключением ограничения на длительность полупериодов, которая определяется максимальной рабочей частотой более медленного устройства. Это позволяет использовать SPI в системах с

низкостабильной тактовой частотой, а также облегчает программную эмуляцию ведущего устройства.

Самым простым будет подключение только одного ведомого устройства.

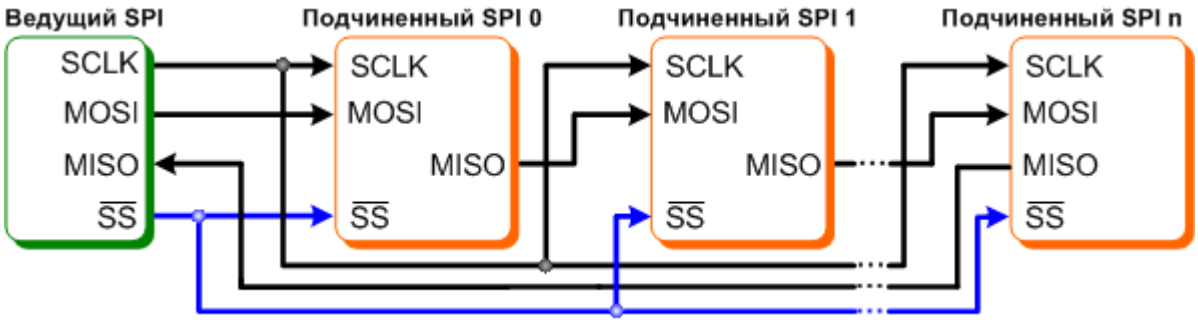


Если в структуре обмена присутствует несколько ведомых устройств, то их подключения к ведущему возможны одним из 2-х способов: независимое подключение или каскадное подключение.



Независимое подключение к шине SPI.

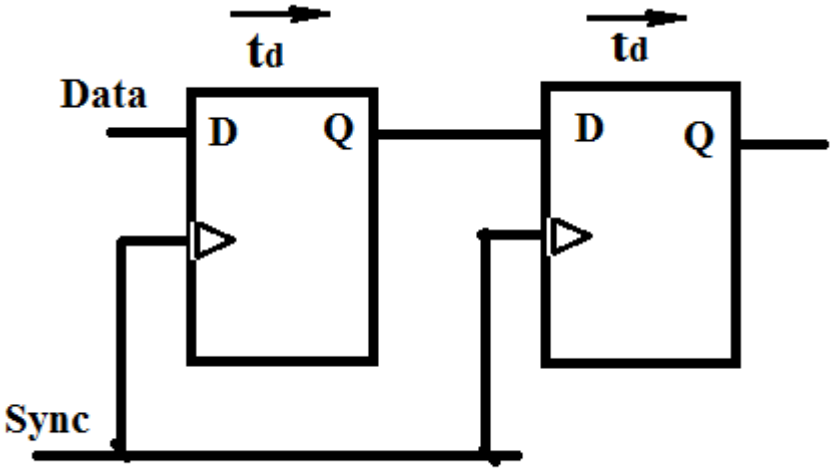
При таком подключении ведущему необходимо иметь несколько выходов для формирования сигналов выбора подключаемого ведомого. Если это нежелательно, то используется каскадное подключение.



Каскадное подключение к шине SPI(возможно при совместимости ведомых устройств по режимам работы).

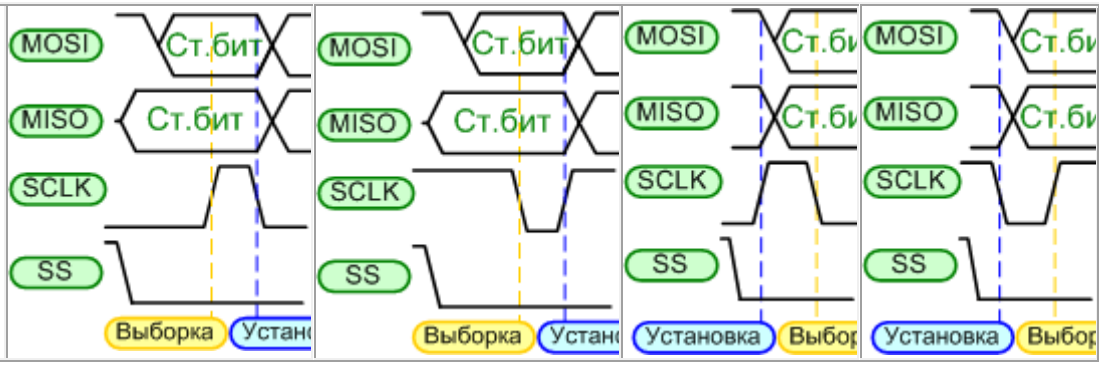
Режимы работы SPI

Функционирование любого регистра сдвига предполагает установку информации в каждом разряде с некоторой задержкой относительно ее выборки. Любое изменение уровня импульса синхронизации (фронт или спад) происходит за какое-то конечное время, которое не должно превышать время задержки триггера. Поэтому, для предотвращения сбоев при сдвиге информации, задержку триггера увеличивают (преимущественно за счет увеличения выходной емкости).



В регистрах шины SPI выборка и установка производятся по противоположным фронтам импульса синхронизации. Исходя из этого, SPI имеет 4 режима работы.

Режим SPI	0	1	2	3
CPOL	0	1	0	1
CPHA	0	0	1	1

Временная диаграмма первого цикла синхронизации	

CPOL – исходный уровень полярности сигнала синхронизации;
CPHA – исходная фаза цикла обмена.

Преимущества интерфейса SPI.

- Полнодуплексная передача данных по умолчанию.
- Возможность произвольного выбора длины пакета.
- Возможность использования в системах с низкостабильной тактовой частотой.
- Адрес ведомого устройства не передается в структуре пакета.
- В отличие от параллельных интерфейсов имеет только 4 вывода.

Недостатки.

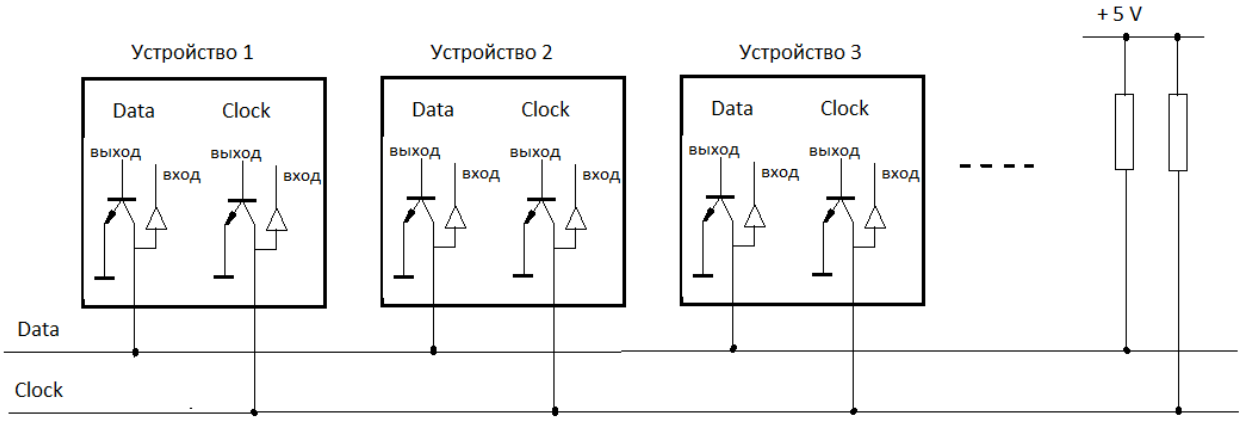
- Ведомое устройство не может управлять потоком данных.
- Нет подтверждения приема данных со стороны ведомого устройства (ведущее устройство может передавать данные «в никуда»).
- Нет определенного стандартом протокола обнаружения ошибок.
- Наличие множества вариантов реализации интерфейса.
- Отсутствие поддержки горячего подключения устройств.
- Необходимо больше выводов, чем для интерфейса I2C.

Интерфейс (шина I2C - Inter-integrated circuit bus)

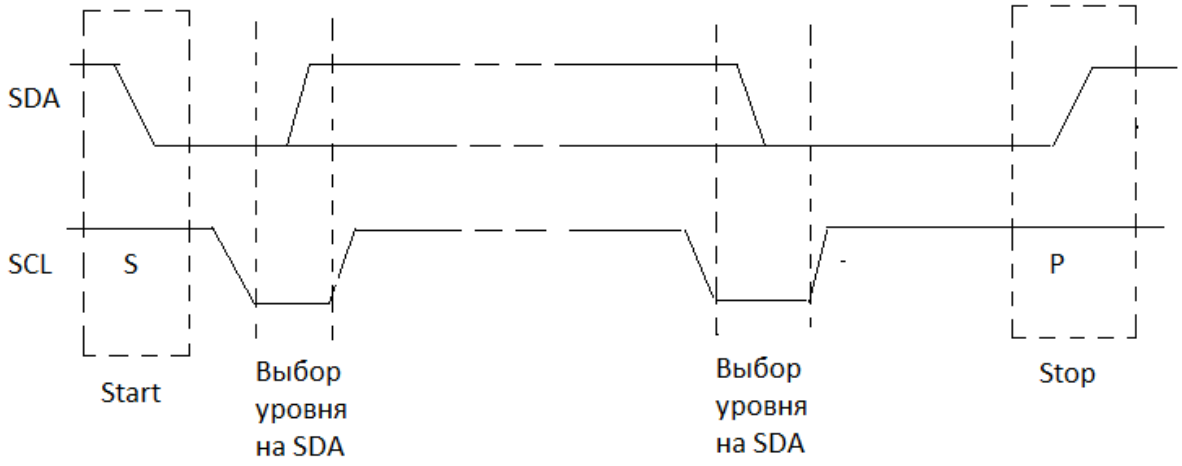
Сетевой последовательный интерфейс, осуществляющий связь ведущих и ведомых по 2-проводной схеме. К шине может быть подключено до 127 устройств, если это не превысит предельную емкость (400 пф).

По скорости передачи различают три режима работы шины: стандартный (S) - 100Кбит/сек, быстрый (F)– 400Кбит/сек и высокоскоростной режим (Hs) – до 3,4Мбит/сек. Первые два режима функционируют практически одинаково. Передача пакетная, объем пакета 8 бит.

Электрическое подключение к проводам.

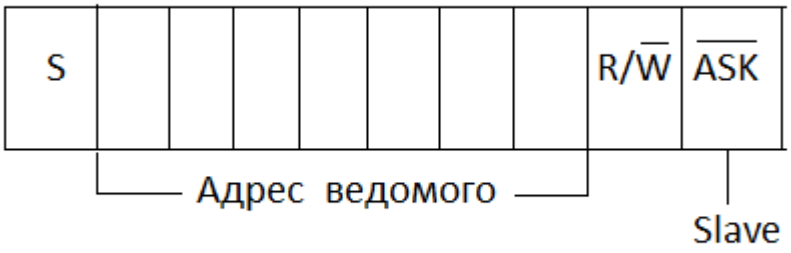


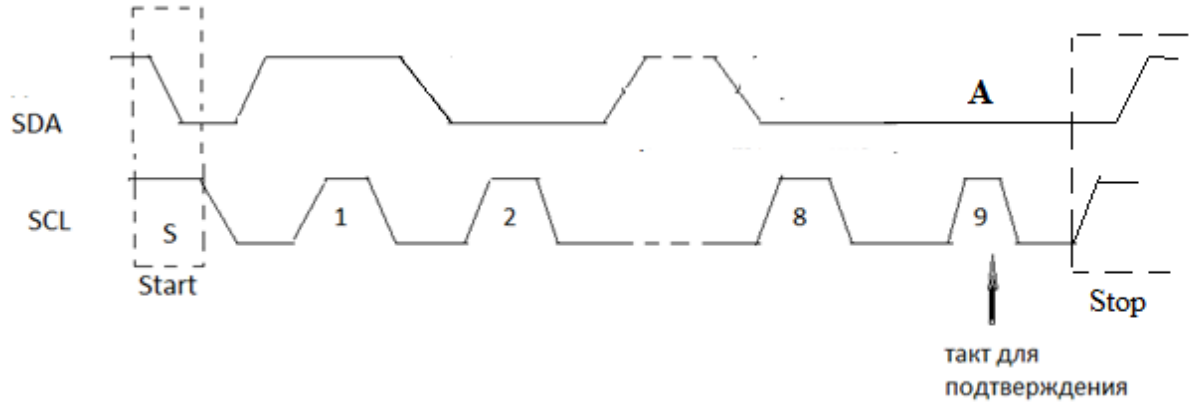
Изначально провода данных и синхронизации подтянуты к «1». Ведущее устройство, первым выставившее на линию **данных** «0» при неактивной линии **синхронизации** может начинать обмен. Синхронизацию всегда генерирует ведущее устройство. При низком уровне SCL происходит выбор данных и их передача, при высоком – считывание приемником. Первая посылка всегда идет от ведущих и содержит адреса ведомых, при этом определяется и направление передачи. Далее следует пересылка пакетов информации. По окончании передачи ведомый должен передать в позиции подтверждения «1», после чего ведущий отпускает линию.

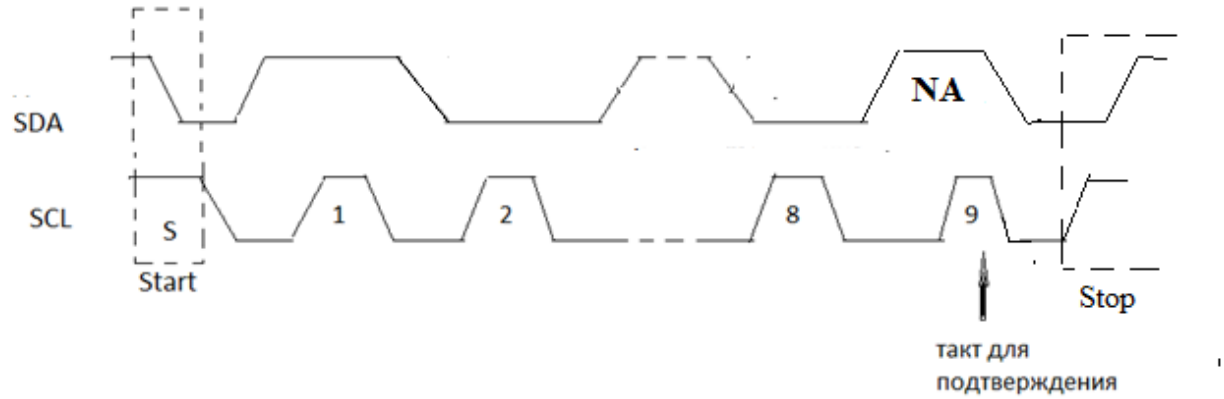


Адресация устройств на шине.
Ведущее устройство выставляет адрес ведомого и ждет подтверждения. Адрес ведомого устройства занимает 7 бит, 8-ой бит указывает дальнейшее направление передачи информации. Далее следует подтверждение. Все устройства системы сравнивают 7 бит после старта со своими адресами,

устройство, адрес которого был набран, становится ведомым и выдает сигнал подтверждения. После получения сигнала подтверждения ведущее устройство должно придержать линию синхронизации на низком уровне, пока ведомое ее не отпустит. Иначе синхронизация оборвется.







Арбитраж.

Ведущим становится устройство, выставившее низкий уровень на провод данных первым перед появлением низкого уровня на проводе синхронизации.

Синхронизацию задает ведущее устройство, но ведомое, если оно не имеет достаточного быстродействия, может замедлять обмен данными.

Таким образом, низкий уровень на проводе синхронизации появляется от устройства с самой высокой частотой, а высокий уровень появляется, когда его сформирует устройство с самой низкой частотой.