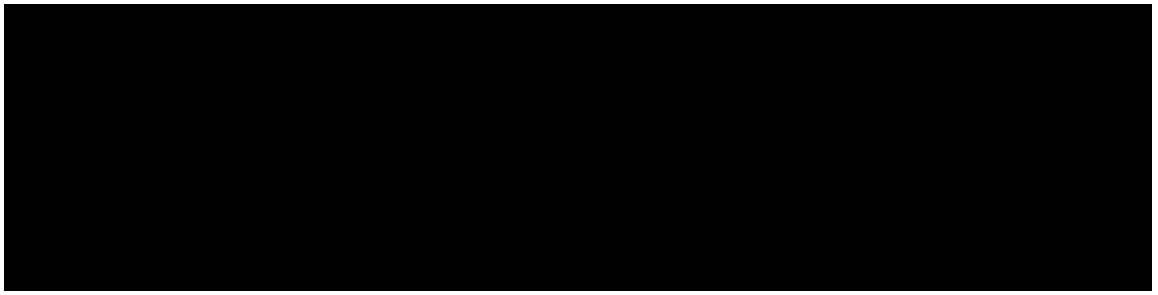


Материалы по дисциплине «Микропроцессорные системы»
Специальность 230113 «Компьютерные системы и комплексы»
Составитель: Торгашин Р.Г

ГОБУ СПО ВО Борисоглебский индустриальный техникум

Интерфейс SPI

SPI (англ. Serial Peripheral Interface, SPI bus — последовательный периферийный интерфейс, шина SPI) — последовательный синхронный стандарт передачи данных в режиме полного дуплекса, разработанный компанией Motorola для обеспечения простого и недорогого сопряжения микроконтроллеров и периферии. SPI также иногда называют четырёхпроводным (англ. four-wire) интерфейсом[1]. SPI - популярный интерфейс для последовательного обмена данными между микросхемами. Интерфейс SPI, наряду с I2C, относится к самым широко-используемым интерфейсам для соединения микросхем

Шина SPI организована по принципу 'ведущий-подчиненный'. В качестве ведущего шины обычно выступает микроконтроллер, но им также может быть программируемая логика, DSP-контроллер или специализированная ИС. Подключенные к ведущему шины внешние устройства образуют подчиненных шины. В их роли выступают различного рода микросхемы, в т.ч. запоминающие устройства (EEPROM, Flash-память, SRAM), часы реального времени (RTC), АЦП/ЦАП, цифровые потенциометры, специализированные контроллеры и др.

К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (англ. chip select) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участия в передаче по SPI.

В отличие от стандартного последовательного порта (англ. standard serial port), SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая (ведомая) периферия синхронизирует получение битовой последовательности с тактовым сигналом.

Главным составным блоком интерфейса SPI является обычный сдвиговый регистр, сигналы синхронизации и ввода/вывода битового потока которого и образуют интерфейсные сигналы. Таким образом, протокол SPI правильнее назвать не протоколом передачи данных, а протоколом обмена данными между двумя сдвиговыми регистрами, каждый из которых одновременно выполняет и функцию приемника, и функцию передатчика. Непременным условием передачи данных по шине SPI является генерация сигнала синхронизации шины. Этот сигнал имеет право генерировать только ведущий шины и от этого сигнала полностью зависит работа подчиненного шины.

В SPI используются четыре цифровых сигнала:
MOSI или SI — выход ведущего, вход ведомого (англ. Master Out Slave In). Служит для передачи данных от ведущего устройства ведомому.
MISO или SO — вход ведущего, выход ведомого (англ. Master In Slave Out). Служит для передачи данных от ведомого устройства ведущему.
SCLK или SCK — последовательный тактовый сигнал (англ. Serial Clock). Служит для передачи тактового сигнала для ведомых устройств.
CS или SS — выбор микросхемы, выбор ведомого (англ. Chip Select, Slave Select).

Электрическое подключение

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала (их основное и альтернативные обозначения см. в табл. 1). Самое простое подключение, в котором участвуют только две микросхемы, показано на рисунке 1.

[1] http://ru.wikipedia.org/wiki/Serial_Peripheral_Interface

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.

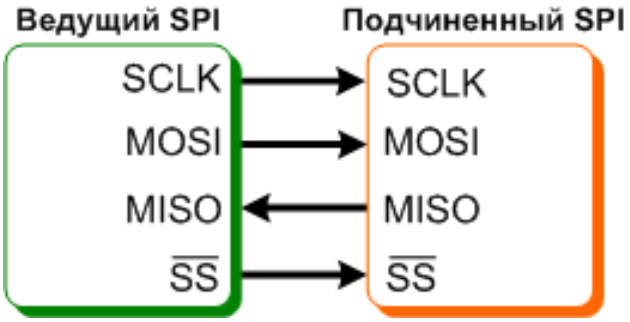


Рисунок 1 Простейшее подключение к шине SPI

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение (рис. 2), либо каскадное (последовательное) (рис. 3). 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 3. 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'.

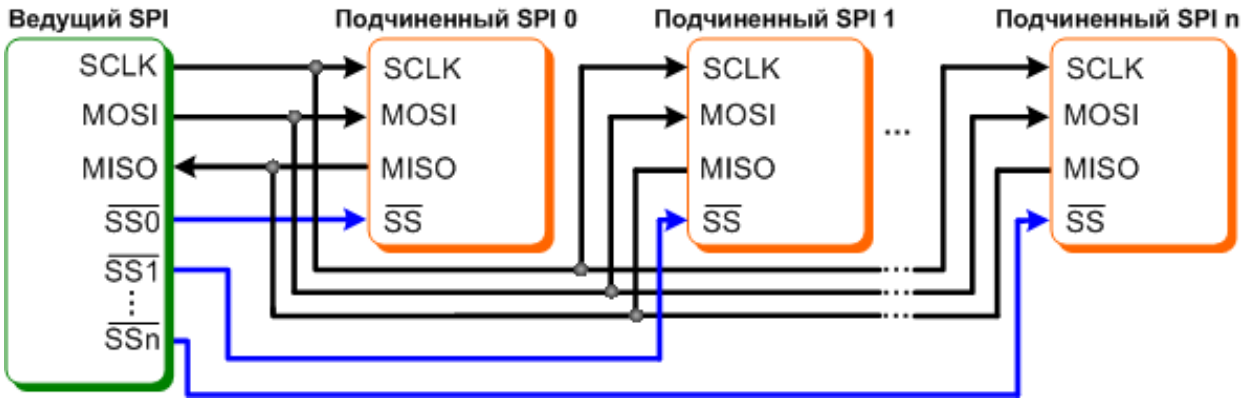


Рисунок 2 Независимое подключение к шине SPI

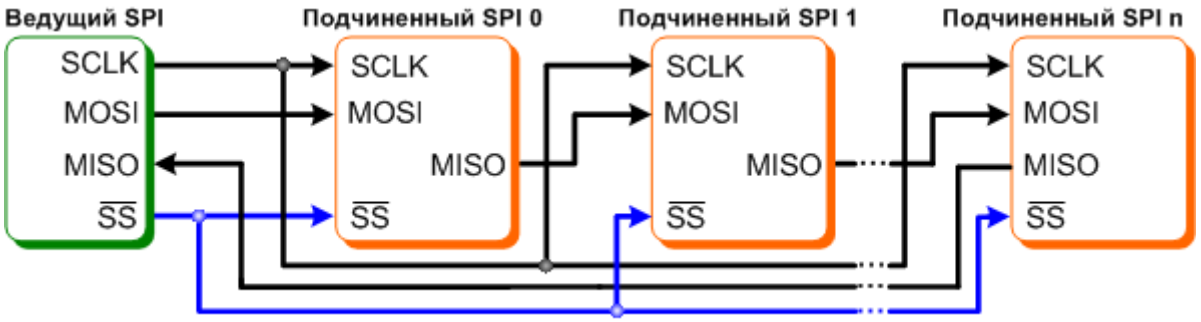


Рисунок 3 Каскадное подключение к шине SPI

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. Если к этому учесть, что в качестве первого фронта в цикле передачи может выступать нарастающий или падающий фронт, то всего возможно четыре варианта логики работы интерфейса SPI. 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 2.

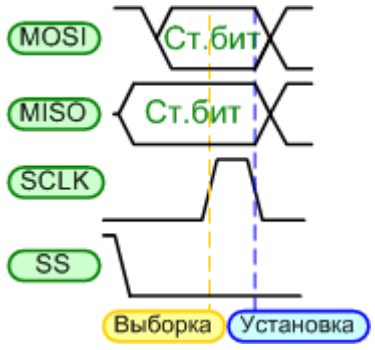
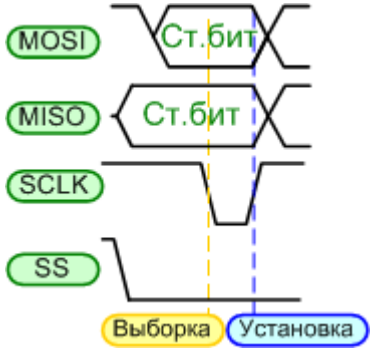
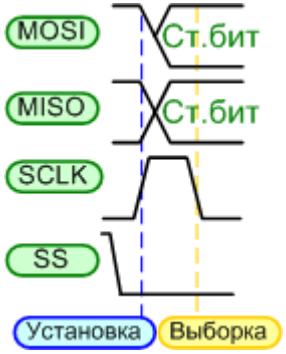
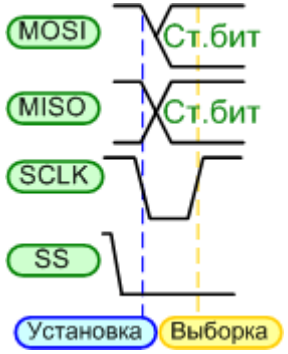
Ведущая и подчиненная микросхемы, работающие в различных режимах SPI, являются несовместимыми, поэтому, перед выбором подчиненных микросхем важно уточнить, какие режимы поддерживаются ведущим шины. Аппаратные модули SPI, интегрированные в микроконтроллеры, в большинстве случаев поддерживают возможность 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.

Таблица 1 Электрические сигналы шины SPI

Ведущий шины			Подчиненный шины		
Основное обозначение	**Альтернативное обозначение**	**Описание**	**Основное обозначение**	**Альтернативное обозначение**	**Описание**
MOSI	DO, SDO, DOUT	Выход последовательной передачи данных	MOSI	DI, SDI, DIN	Вход последовательного приема данных
MISO	DI, SDI, DIN	Вход последовательного приема данных	MISO	DO, SDO, DOUT	Выход последовательной передачи данных
SCLK	DCLOCK, CLK, SCK	Выход синхронизации передачи данных	SCLK	DCLOCK, CLK, SCK	Вход синхронизации приема данных
SS	CS	Выход выбора подчиненного (выбор микросхемы)	SS	CS	Вход выбора подчиненного (выбор микросхемы)

Таблица 2 Режимы SPI

Режим SPI	**0**	**1**	**2**	**3**
CPOL	0	1	0	1
CPHA	0	0	1	1
Временная диаграмма первого цикла синхронизации				

Реализация SPI в PIC18f4520

Интерфейс SPI реализован на базе модуля MSSP.
В режиме SPI используются 8 битные данные. Все четыре режима поддерживаются.
Обычно используются выводы:
SDO – RC5/SDO
SDI – RC4/SDI/SDA
SCK – RC3/SCK/SCL
SS – RA5/SS

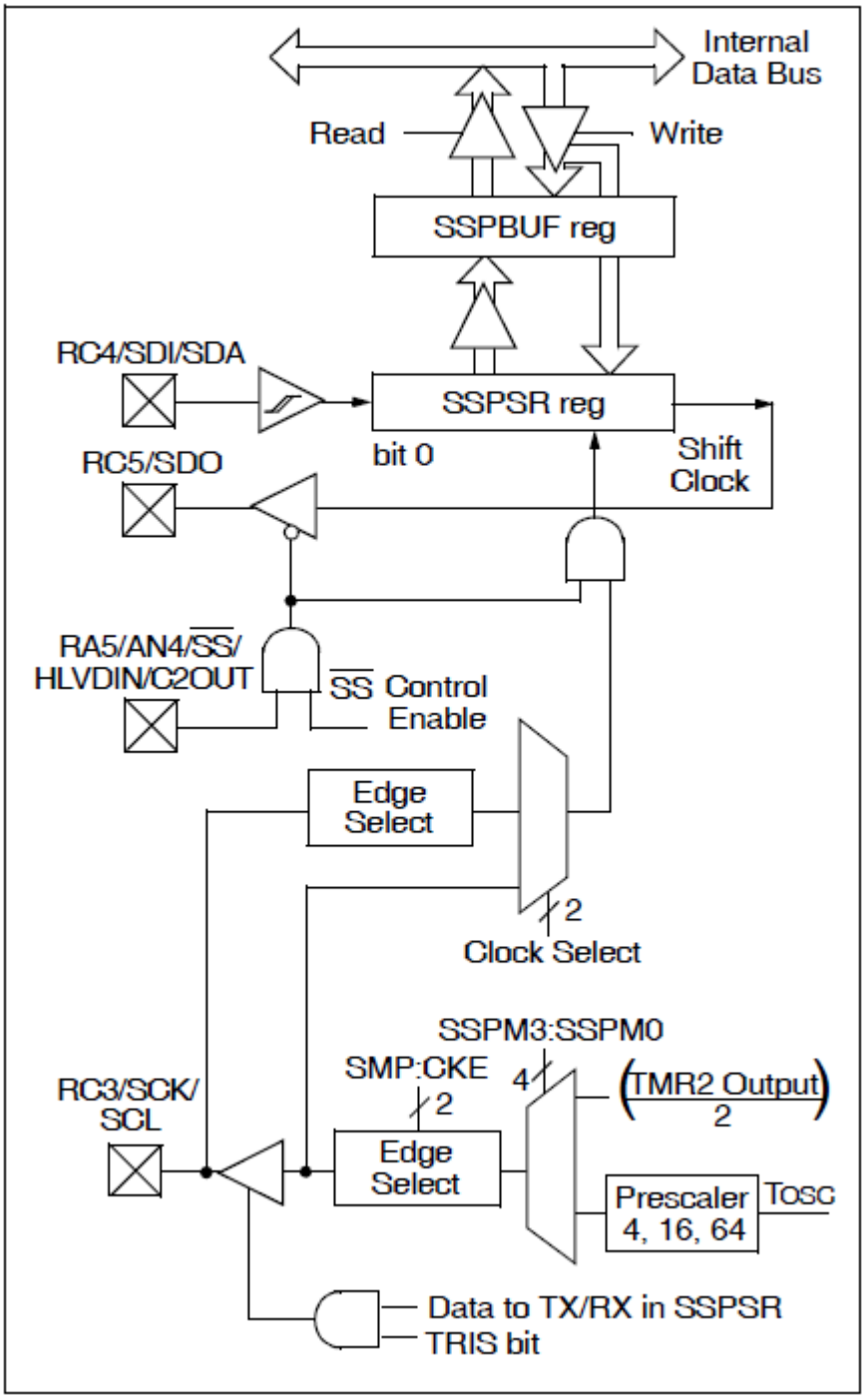


Рисунок 4 Диаграмма MSSP для режима SPI

Интерфейс I2C

I2C (рус. ай-ту-си/и-два-цэ) — последовательная шина данных для связи интегральных схем, разработанная фирмой Philips в начале 1980-х как простая шина внутренней связи для создания управляющей электроники. Используется для соединения низкоскоростных периферийных компонентов с материнской платой, встраиваемыми системами и мобильными телефонами. Название представляет собой аббревиатуру слов Inter-Integrated Circuit.

Вот некоторые достоинства шины I2C:

- Требуется только две линии - линия данных (SDA) и линия синхронизации (SCL) Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу. В каждый момент времени существует простое отношение ведущий/ведомый: ведущие могут работать как ведущий-передатчик и ведущий-приёмник.
- Шина позволяет иметь несколько ведущих, предоставляя средства для определения коллизий и арбитраж для предотвращения повреждения данных в ситуации, когда два или более ведущих одновременно начинают передачу данных В стандартном режиме обеспечивается передача последовательных 8-битных данных со скоростью до 100 кбит/с, и до 400 кбит/с в “быстром” режиме.
- Встроенный в микросхемы фильтр подавляет всплески, обеспечивая целостность данных.
- Максимальное допустимое количество микросхем, подсоединённых к одной шине, ограничивается максимальной емкостью шины 400 пФ

2C-совместимые микросхемы позволяют ускорить процесс разработки от функциональной схемы до прототипа. Более того, поскольку такие микросхемы подключаются непосредственно к шине без каких-либо дополнительных цепей, появляется возможность модификации и модернизации системы прототипа путем подключения и отключения устройств от шины.

Вот некоторые достоинства I2C-совместимых микросхем, которые касаются конструкторов:

- Блоки на функциональной схеме соответствуют микросхемам, переход от функциональной схемы к принципиальной происходит быстро.
- Нет нужды разрабатывать шинные интерфейсы, т.к. шина уже интегрирована в микросхемы.
- Интегрированные адресация устройств и протокол передачи данных позволяют системе быть полностью программно определяемой.
- Одни и те же типы микросхем могут быть часто использованы в разных приложениях.
- Время разработки снижается, так как конструкторы быстро знакомятся с часто используемыми функциональными блоками и соответствующими микросхемами.
- Микросхемы могут быть добавлены или убраны из системы без оказывания влияния на другие микросхемы, подключенные к шине.
- Простая диагностика сбоев и отладка; нарушения в работе могут быть немедленно отслежены.
- Время разработки программного обеспечения может быть снижено за счет использования библиотеки повторно используемых программных модулей.

Помимо этих преимуществ, КМОП I2C-совместимые микросхемы предоставляют для конструкторов специальные решения, которые в частности привлекательны для портативного оборудования и систем с батарейным питанием:

- Крайне низкое потребление.
- Высокая стойкость к помехам.

- Широкий диапазон питающего напряжения.
- Широкий рабочий температурный диапазон.

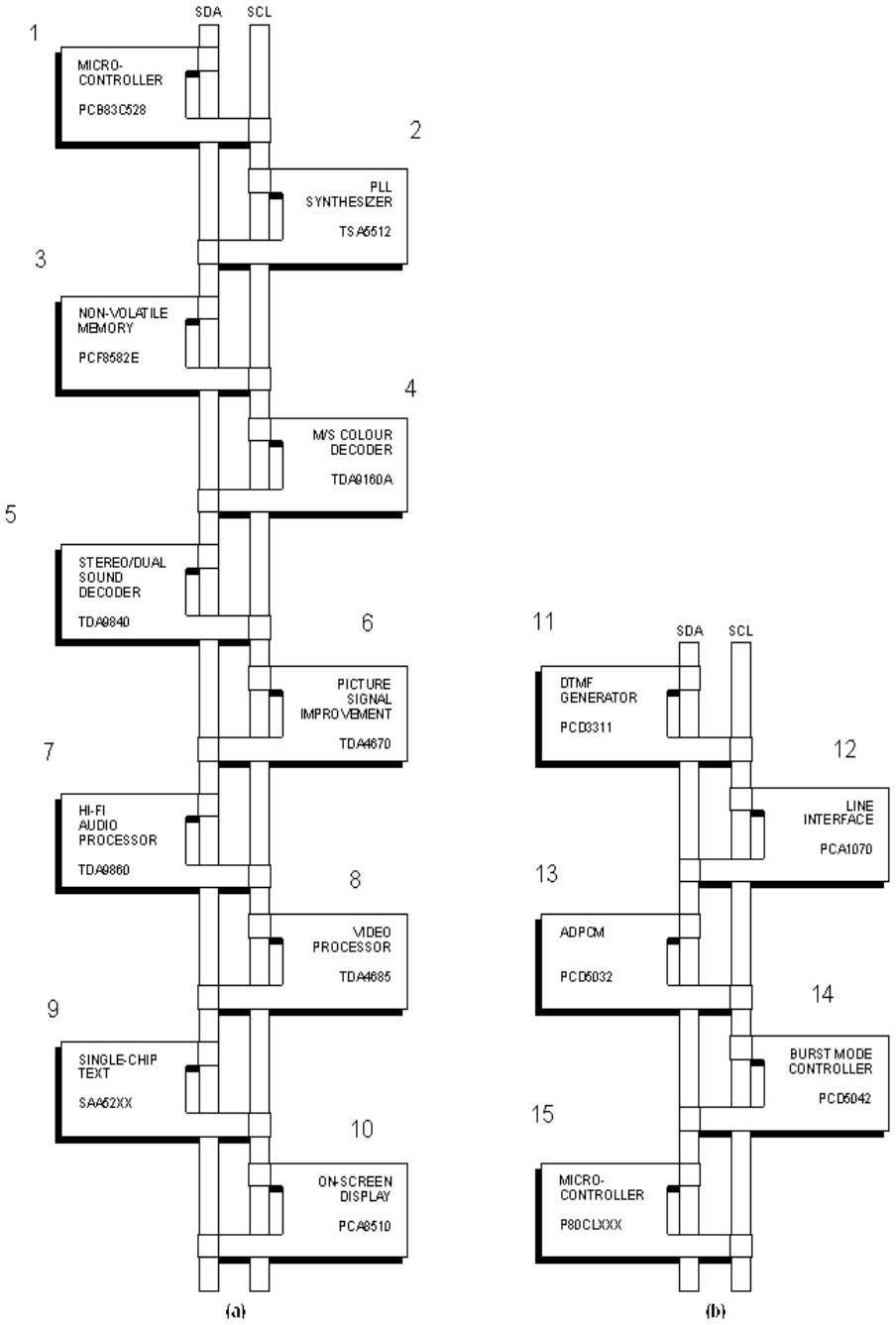


Рисунок 5 Два примера применения I2C a) высокоинтегрированный телевизор b) базовая станция радиотелефона стандарта DECT

1. Микроконтроллер
2. ФАПЧ синтезатор
3. Флеш-память
4. Декодер цвета
5. Стереодекодер звука
6. Улучшение сигнала картинки
7. HI-FI аудиопроцессор
8. Видеопроцессор
9. Одночиповый текст
11. Генератор DTMF
12. Интерфейс телефонной линии
13. Кодек АДИКМ
14. Пакетный контроллер
15. Микроконтроллер

10. Экранный дисплей

SDA - линия данных, SCL - линия синхронизации

Шина I2C поддерживает любую технологию изготовления микросхем (НМОП, КМОП, биполярную). Две линии, **данных (SDA)** и **синхронизации (SCL)** служат для переноса информации. Каждое устройство распознается по уникальному адресу - будь то микроконтроллер, ЖКИ буфер, память или интерфейс клавиатуры - и может работать как передатчик или приёмник, в зависимости от назначения устройства. Обычно ЖКИ буфер - только приёмник, а память может как принимать, так и передавать данные. Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. Ведущий - это устройство, которое инициирует передачу данных и вырабатывает сигналы синхронизации. При этом любое адресуемое устройство считается ведомым по отношению к ведущему

Таблица 3 Классы устройств

Термин (англ)	Термин(рус)	Описание
Transmitter	Передатчик	Устройство, посылающее данные в шину
Receiver	Приемник	Устройство, принимающее с шины
Master	Ведущий	Начинает пересылку данных, вырабатывает синхроимпульсы, заканчивает пересылку данных
Slave	Ведомый	Устройство, адресуемое ведущим
Multi-master	-	Несколько ведущих могут пытаться захватить шину одновременно, без нарушения передаваемой информации
Arbitration	Арбитраж	Процедура, обеспечивающая Multi-master
Synchronization	Синхр.	Процедура синхронизации двух устройств

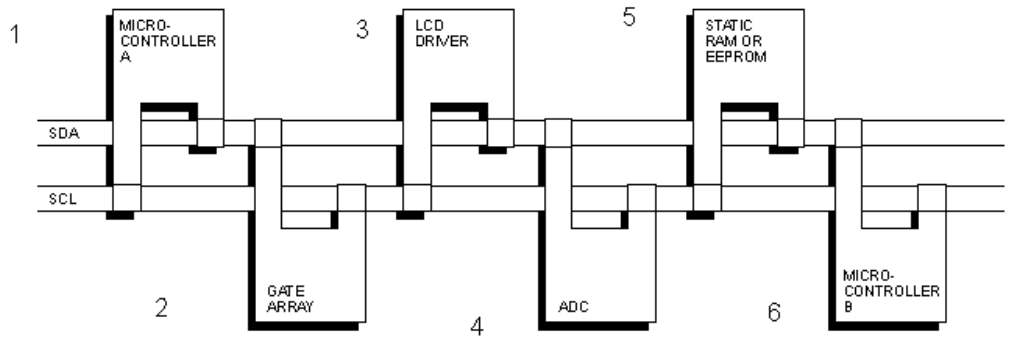


Рисунок 6 Пример конфигурации шины I2C с двумя микроконтроллерами

1. Микроконтроллер А
2. Массив
3. ЖКИ драйвер
4. АЦП
5. Статическая ОЗУ или ППЗУ
6. Микроконтроллер В

Шина I2C допускает несколько ведущих. Это означает, что более чем одно устройство, способное управлять шиной, может быть подключено к ней. Поскольку в качестве ведущих обычно выступают микроконтроллеры, давайте рассмотрим пример

пересылки данных между двумя микроконтроллерами, подключенными к шине (рис 3). Пример покажет взаимоотношения передатчик-приемник и ведущий-ведомый, существующие в шине I2C. Необходимо заметить, что эти отношения не постоянны, а зависят только от направления пересылки данных в данный момент времени. Пересылка данных будет происходить следующим образом:

1. Пусть микроконтроллер А желает послать информацию в микроконтроллер В:
   - микроконтроллер А (ведущий) адресует микроконтроллер В (ведомый)
   - микроконтроллер А (ведущий-передатчик) посылает данные микроконтроллеру В (ведомый-приёмник)
   - микроконтроллер А заканчивает пересылку
2. Пусть микроконтроллер А желает принять информацию от микроконтроллера В:
   - микроконтроллер А (ведущий) адресует микроконтроллер В (ведомый)
   - микроконтроллер А (ведущий-приемник) принимает данные от микроконтроллера В (ведомый-передатчик)
   - микроконтроллер А заканчивает пересылку

В обоих случаях ведущий (микроконтроллер А) генерирует синхроимпульсы и заканчивает пересылку.

Возможность подключения более одного микроконтроллера к шине означает, что более чем один ведущий может попытаться начать пересылку в один и тот же момент времени. Для устранения хаоса, который может возникнуть в данном случае, разработана процедура арбитража. Эта процедура основана на том, что все I2C-устройства подключаются к шине по правилу монтажного И.

Генерация синхросигнала - это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине. Сигнал синхронизации может быть изменен только если он “вытягивается” медленным ведомым устройством (путем удержания линии в низком состоянии), или другим ведущим, если происходит коллизия.

Как SDA, так и SCL являются двунаправленными линиями, подсоединенными к положительному источнику питания через подтягивающий резистор. Когда **шина свободна**, обе линии находятся в ВЫСОКОМ положении. Выходные каскады устройств, подключенных к шине, должны иметь открытый сток или открытый коллектор для обеспечения функции монтажного И. Данные по шине I^2C могут **передаваться со скоростью** до 100 кбит/с в стандартном режиме, и до 400 кбит/с в “быстром” режиме. Количество устройств, подключенных к шине, определяется единственным параметром - емкостью линии (до 400 пф). **Стандартные напряжения** +5 В или +3,3 В, однако допускаются и другие.

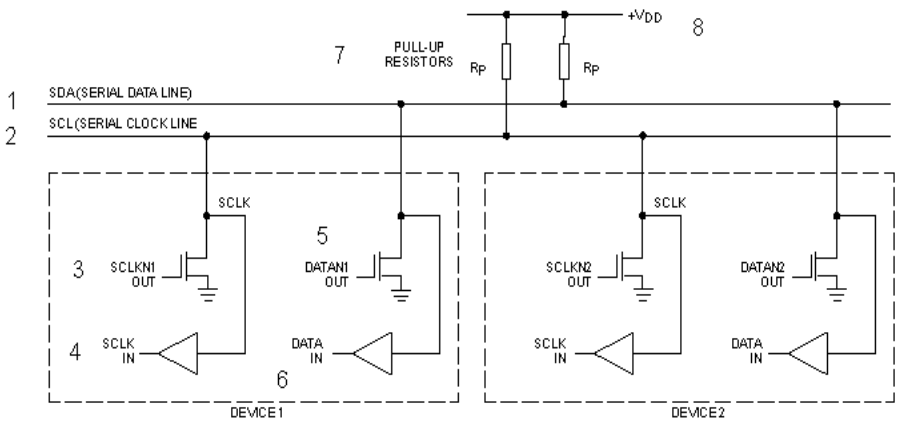


Рисунок 7 Подключение I2C-устройств к шин

1. SDA (линия данных)
5. Выход данных

2. SCL (линия синхронизации)
3. Выход синхронизации
4. Вход синхронизации
6. Вход данных
7. Подтягивающие резисторы
8. Напряжение питания

Вследствие различных технологий микросхем (КМОП, НМОП, биполярная), которые могут быть подключены к шине, **уровни логического нуля** (“НИЗКИЙ”) и логической единицы (“ВЫСОКИЙ”) не фиксированы и зависят от соответствующего уровня Vdd. Один синхроимпульс генерируется на каждый пересылаемый бит

Данные на линии SDA должны быть **стабильными** в течение ВЫСОКОГО периода синхроимпульса. ВЫСОКОЕ или НИЗКОЕ состояние линии данных должно меняться, только если линия синхронизации в состоянии НИЗКОЕ

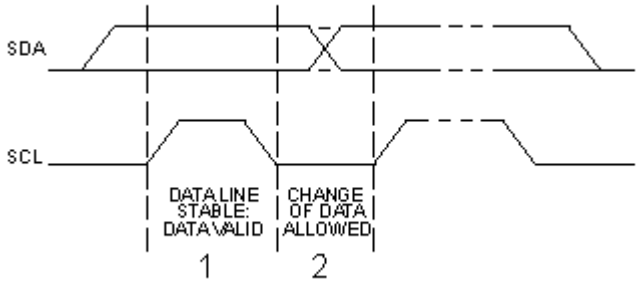


Рисунок 8 Пересылка бита в шине I2C 1- Линия данных находится в стабильном состоянии, данные определены 2- Допускается изменение данных

Специальные ситуации на шине отмечают сигналы START и STOP.

Переход линии SDA из ВЫСОКОГО состояния в НИЗКОЕ, в то время как SCL находится в ВЫСОКОМ состоянии означает START.

Переход линии SDA из НИЗКОГО состояния в ВЫСОКОЕ при SCL в ВЫСОКОМ состоянии означает STOP.

Сигналы СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после сигнала СТАРТ. Шина считается освободившейся через определенное время после сигнала СТОП.

Определение сигналов СТАРТ и СТОП устройствами, подключенными к шине достаточно легко, если в них встроены необходимые цепи. Однако микроконтроллеры без таковых цепей должны осуществлять считывание значения линии SDA как минимум дважды за период синхронизации для того, чтобы определить переход состояния

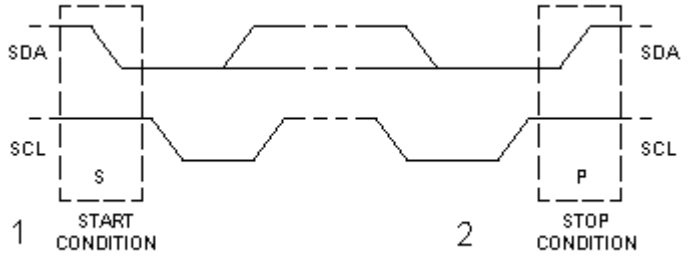


Рисунок 9 Сигналы START и STOP

Каждый **байт, передаваемый по линии SDA**, должен состоять из 8 бит.

Количество байт, передаваемых за один сеанс связи неограниченно.

Каждый байт должен оканчиваться битом подтверждения.

Данные передаются, начиная с наиболее значащего бита.

Если приёмник не может принять еще один целый байт, пока он не выполнит какую-либо другую функцию (например, обслужит внутреннее прерывание), он может удерживать линию SCL в НИЗКОМ состоянии, переводя передатчик в состояние ожидания. Пересылка данных продолжается, когда приёмник будет готов к следующему байту и отпустит линию SCL.

В некоторых случаях, необходимо использовать другой формат данных (например, CBUS). Посылка, которая передается с таким адресом, может быть закончена выдачей сигнала СТОП, даже если это происходит во время передачи байта. В этом случае подтверждение не генерируется.

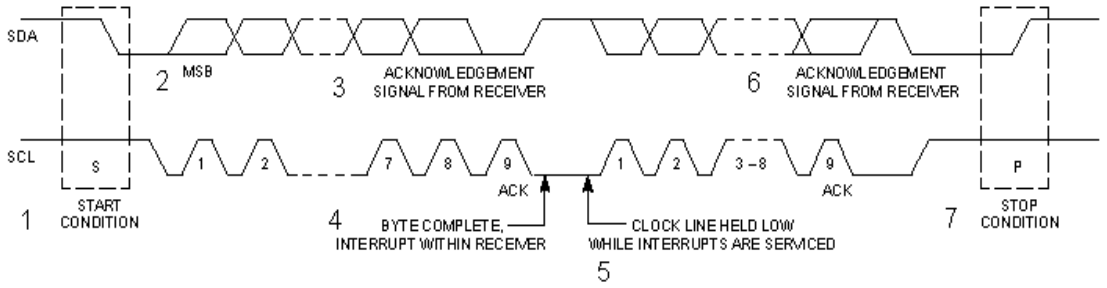


Рисунок 10 Пересылка данных по шине I2C

1. Сигнал СТАРТ
2. Старший разряд байта
3. Сигнал подтверждения от приёмника
4. Прием байта завершен. Прерывание внутри приемника
5. Синхролиния удерживается в низком состоянии, пока обслуживается прерывание
6. Сигнал подтверждения от приемника
7. Сигнал СТОП

Подтверждение при передаче данных обязательно. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает (ВЫСОКОЕ) линию SDA в течение синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение ВЫСОКОГО состояния синхроимпульса подтверждения в стабильно НИЗКОМ состоянии. Конечно, время установки и удержания также должны быть приняты во внимание.

Обычно, приёмник, который был адресован, обязан генерировать подтверждение после каждого принятого байта, исключая те случаи, когда посылка начинается с адреса CBUS.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), линия данных должна быть оставлена в ВЫСОКОМ состоянии. После этого ведущий может выдать сигнал СТОП для прерывания пересылки данных.

Если ведомый-приёмник подтвердил свой адрес, но через некоторое время больше не может принимать данные, ведущий также должен прервать пересылку. Для этого ведомый не подтверждает следующий байт, оставляет линию данных в ВЫСОКОМ состоянии и ведущий генерирует сигнал СТОП.

Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путем не подтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал СТОП или повторить сигнал СТАРТ

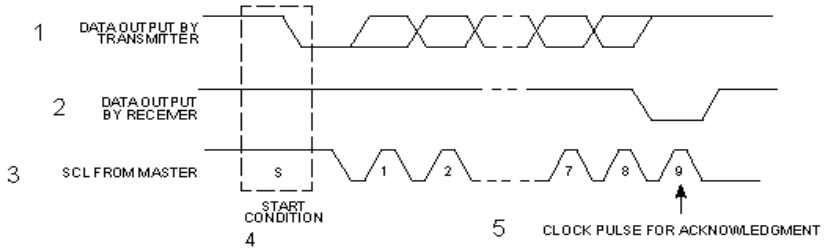


Рисунок 11 Подтверждение

1. Данные, переданные передатчиком
2. Данные, переданные приёмником
3. SCL от ведущего
4. Сигнал СТАРТ
5. Синхроимпульс подтверждения

Процедура адресации на шине I2C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый выбирается ведущим для работы. Исключение составляет адрес “Общего вызова”, который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако, устройства могут быть сделаны игнорирующими этот адрес. Второй байт посылки общего вызова определяет действие, которое должны произвести устройства

Процедура адресации на шине I2C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый выбирается ведущим для работы. Исключение составляет **адрес “Общего вызова”**, который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако, устройства могут быть сделаны игнорирующими этот адрес. Второй байт посылки общего вызова определяет действие, которое должны произвести устройства

Первые семь битов первого байта образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных.

“Ноль” означает, что ведущий будет записывать информацию в выбранного ведомого.

“Единица” означает, что ведущий будет считывать информацию из ведомого.

После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

Адрес ведомого может состоять из фиксированной и программируемой частей. Вероятно, что в системе будет несколько таких одинаковых устройств, поэтому при помощи программируемой части адреса становится возможным подключить к шине максимально возможное количество таких устройств. Количество программируемых бит в адресе зависит от количества свободных выводов микросхемы. Например, если устройство имеет 4 фиксированных и 3 программируемых адресных битов, всего 8 одинаковых устройств может быть подключено к шине.

Интерфейс USART (UART)

Интерфейс USART — последовательный универсальный синхронно-асинхронный приемо-передатчик. В отличии от I2C или SPI, в которых передача бита осуществляется по сигналу синхронизации, **передача данных в USART осуществляется через равные промежутки времени в одной линии в каждом направлении**. Этот временной промежуток определяется заданной скоростью USART и указывается в бодах (битах в секунду). Существует общепринятый ряд стандартных скоростей: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 бод.

Помимо бит данных USART автоматически вставляет в поток синхронизирующие метки, так называемые **стартовый и стоповый биты**. При приёме эти лишние биты удаляются. Обычно стартовый и стоповый биты отделяют один байт информации (8 бит), однако встречаются реализации USART, которые позволяют передавать по 5, 6, 7, 8 или 9 бит. Отделённые стартом и стопом биты являются минимальной посылкой. USART позволяет 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 данных) состоянием входа и выхода USART является логическая «1». Стартовый бит всегда логический «0», поэтому приёмник USART 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-бит 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, USART фиксирует ошибку.

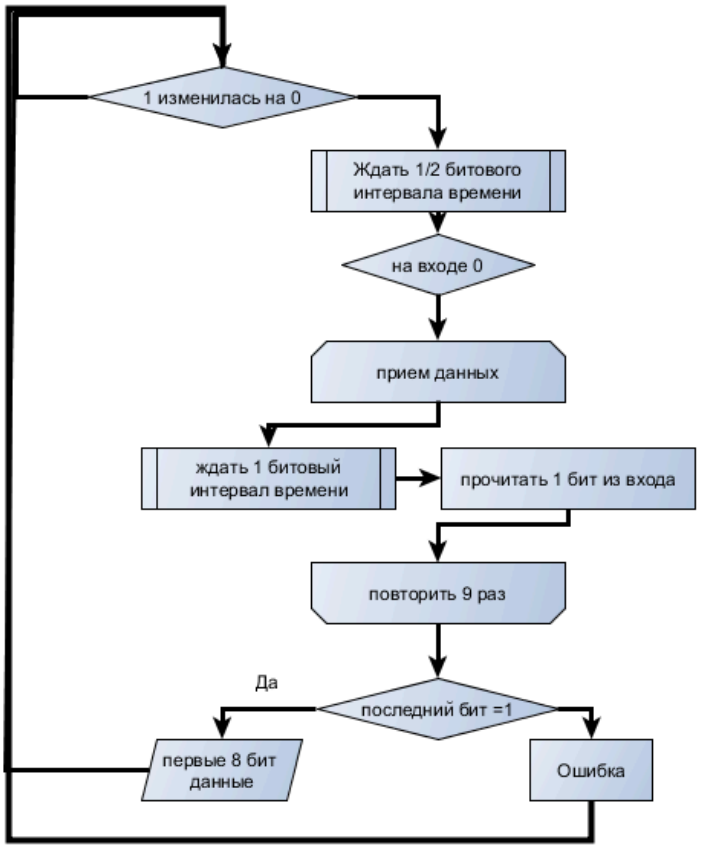


Рисунок 12 Алгоритм приема данных

Рисунок 13 Байты при передаче по USART

Для формирования временных интервалов передающий и приёмный USART имеют источник точного времени (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины (а лучше хотя бы четверти) битового интервала. Для 8-бит посылки 0,5/9,5 = 5 % (в реальности не более 3 %). Поскольку эта сумма ошибок приёмника и передатчика плюс возможные искажения сигнала в линии, то рекомендуемый допуск на точность тактирования USART не более 1,5 %.

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 92160 бит/с или 11 520 байт/с.

Контроль чётности

В протоколе USART имеют возможность автоматически контролировать целостность данных методом контроля битовой чётности. Когда эта функция включена, последний бит данных («бит чётности») всегда принимает значение 1 или 0, так чтобы количество единиц в байте всегда было четным.

Например:

1 0 0 1 0 0 1 **1** = в байте 4 единицы

1 0 1 1 1 1 1 **0** = в байте 6 единиц

Управление потоком

В старые времена устройства с USART могли быть настолько медлительными, что не успевали обрабатывать поток принимаемых данных. Для решения этой проблемы модули USART снабжались отдельными выходами и входами управления потоком. При заполнении входного буфера логика принимающего USART выставляла на соответствующем выходе запрещающий уровень, и передающий USART приостанавливал передачу.

Позже управление потоком возложили на коммуникационные протоколы, и надобность в отдельных линиях управления потоком постепенно исчезла.

Физическая реализация

USART это протокол обмена, т.е. он определяет способ формирования бита, параметры передачи байта, скорость передачи и прочее.

А вот физическая реализация у USARTа может быть различная. Например для передачи данных внутри одной платы сигналы передаются уровнями +5В и 0В. Для передачи данных на длинные расстояния и между устройствами применяются другие физические уровни напряжений и стандарты такие как: токовая петля (4-20 мА), RS-232 (COM-порт), RS-485 и тому подобные.

Для преобразования «контроллерных» уровней 0-5В в «стандартные» существует огромное количество специализированных микросхем, например ADM202 для RS-232.

Модуль USART контроллера PIC

Модуль USART может работать в трех режимах:

- Асинхронный, полный дуплекс;
- Ведущий синхронный полудуплекс;
- Ведомый Синхронный полудуплекс.

В режиме **ведущего/ведомого синхронного полудуплекса** передача ведется аналогично протоколам I2C или SPI — состояние линии данных изменяется по тактовому

сигналу на линии синхронизации. В режиме полудуплекса единовременно разрешена передача только в одном направлении (при передаче запрещен прием).

В **асинхронном режиме** сигнал тактируется по временным интервалам (стандартной скоростью), а передача возможно одновременно в обоих направлениях, по отдельным линиям связи.

Модуль USART поддерживает 8 и 9 битную передачу. Девятый бит может использоваться например как бит четности, хотя попадались мне специфические устройства которые использовали девятый бит как информационный.

Перед работой с модулем USART выводы модуля RC6/TX/CK и RC7/RX/DT должны быть настроены на вход (TRISC<7:6>).

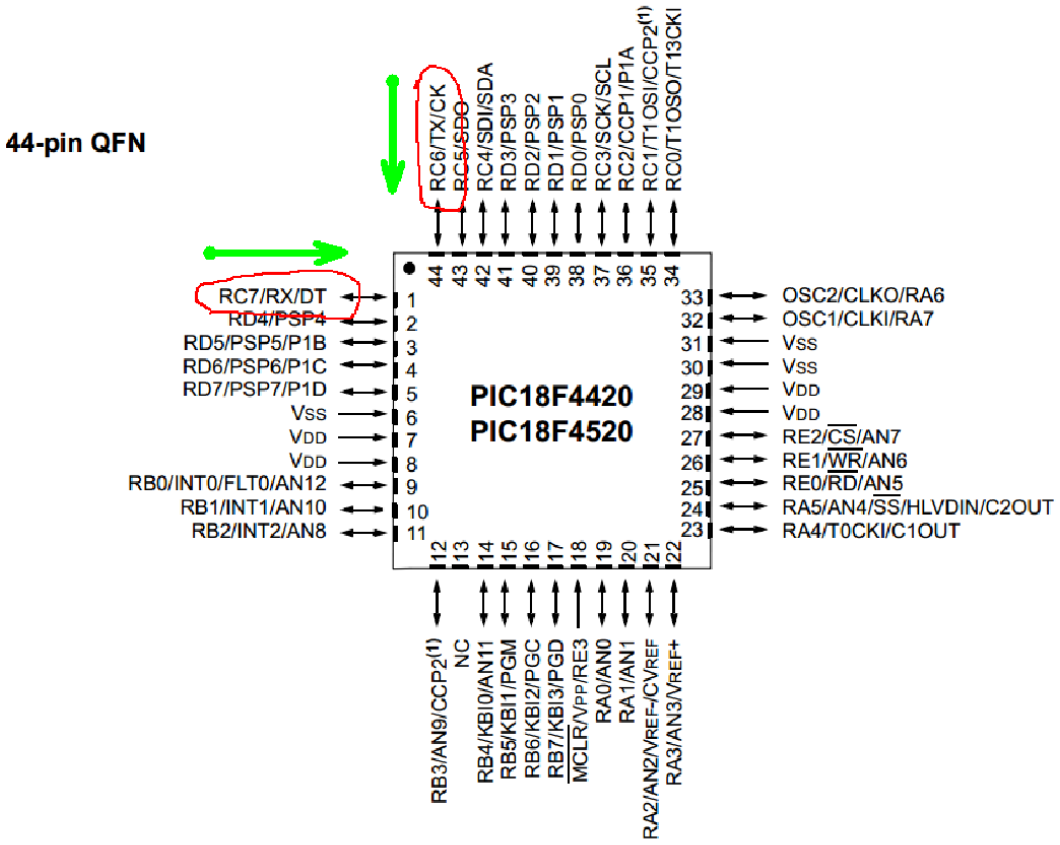


Рисунок 14 Выводы RC6 и RC7 в PIC18F4520 корпус 44-pin QFN

Обмен данными

Параметры связи:

- Частота кварца контроллера: 4 МГц
- Скорость: 19200 бит/сек;
- Кол-во бит: 8 (без контроля четности);

В работе модуля USART участвуют следующие регистры:

- TXSTA — Регистр управления и статуса передатчика;
- RCSTA — Регистр управления и статуса приемника;
- SPBRG — Регистр генератора скорости USART;
- TXREG — Регистр данных передатчика;
- RCREG — Регистр данных приемника;
- TSR — Сдвиговый регистр передатчика (программно не доступный).

Описание регистров управления

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-Bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-Bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be cleared by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit, CREN)
0 = No overrun error

bit 0 **RX9D:** 9th Bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

Генератор частоты обмена USART BRG

BRG используется для работы USART в синхронном ведущем и асинхронном режимах . BRG представляет собой отдельный 8- разрядный генератор скорости обмена в бодах , период которого определяется значением в регистре SPBRG. В асинхронном режиме бит BRGH (TXSTA<2>) тоже влияет на скорость обмена (в синхронном режиме бит BRGH игнорируется). Ниже указаны формулы для вычисления скорости обмена в бодах при различных режимах работы модуля USART (относительно внутреннего тактового сигнала микроконтроллера). Учитывая требуемую скорость и F OSC , выбирается самое близкое целое значение для записи в регистр SPBRG (от 0 до 255), рассчитанное по формулам. Затем рассчитывается ошибка скорости обмена

SYNC	BRGH = 0	BRGH = 1
0	(Асинхронный) Скорость обмена = F_{osc} / (64 (X + 1))	(Асинхронный) Скорость обмена = F_{osc} / (16 (X + 1))
1	(Синхронный) Скорость обмена = F_{osc} / (4 (X + 1))	(Синхронный) Скорость обмена = F_{osc} / (4 (X + 1))

Рисунок 15 Формулы расчета скорости

X = значение регистра SPBRG (от0 до255)

Например

Пусть:

F_{osc} = 16 МГц

Скорость приема/передачи = 9600 бит/с

BRGH = 0

SYNC = 0

Тогда:

Желаемая скорость 9600 = 16 000 000/(64*(X+1))

X = [25.042] = 25

Вычисленная скорость = 16 000 000/(64*(25+1)) = 9615

Ошибка = 100*(Вычисленное - Желаемое)/Желаемое

Ошибка = 100*(9615 - 9600)/9600 = 0,16%

В некоторых случаях выгоднее использовать высокоскоростной режим (BRGH=1) так как это дает меньшую ошибку.

Запись нового значения в регистр SPBRG сбрасывает таймер BRG, гарантируя немедленный переход на новую скорость.

Имя	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Сброс POR, BOR	Другие сбросы
TXSTA	CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	Регистр генератора скорости USART								0000 0000	0000 0000

Рисунок 16 Регистры и биты связанные с работой генератора BRG

В Datasheet можно найти заранее рассчитанные скорости обмена. Например:

Скорость обмена (К)	Fosc = 20 МГц			Fosc = 16 МГц			Fosc = 10 МГц		
	Реальная скорость	Ошибка %	Значение SPBRG (десят.)	Реальная скорость	Ошибка %	Значение SPBRG (десят.)	Реальная скорость	Ошибка %	Значение SPBRG (десят.)
0,3	Нет	-	-	Нет	-	-	Нет	-	-
1,2	Нет	-	-	Нет	-	-	Нет	-	-
2,4	Нет	-	-	Нет	-	-	Нет	-	-
9,6	Нет	-	-	Нет	-	-	9,166	+1,73	255
19,2	19,53	+1,73	255	19,23	+0,16	207	19,23	+0,16	129
76,8	76,92	+0,16	64	76,92	+0,16	51	7576	-1,36	32
96	96,15	+0,16	51	95,24	-0,79	41	96,15	+0,16	25
300	294,1	-1,96	16	307,69	+2,56	12	312,5	+4,17	7
500	500	0	9	500	0	7	500	0	4
Максим.	5000	-	0	4000	-	0	2500	-	0
Миним.	19,53	-	255	15,625	-	255	9,766	-	255

Рисунок 17 Скорости обмена в синхронном режиме

Скорость обмена (К)	Fosc = 20 МГц			Fosc = 16 МГц			Fosc = 10 МГц		
	Реальная скорость	Ошибка %	Значение SPBRG (десят.)	Реальная скорость	Ошибка %	Значение SPBRG (десят.)	Реальная скорость	Ошибка %	Значение SPBRG (десят.)
9,6	9,615	+0,16	129	9,615	+0,16	103	9,615	+0,16	64
19,2	19,230	+0,16	64	19,230	+0,16	51	18,939	-1,36	32
38,4	37,878	-1,36	32	38,461	+0,16	25	39,062	+1,7	15
57,6	56,818	-1,36	21	58,823	+2,12	16	56,818	-1,36	10
115,2	113,636	-1,36	10	111,111	-3,55	8	125	+8,51	4
250	250	0	4	250	0	3	Нет	-	-
625	625	0	1	Нет	-	-	625	0	0
1250	1250	0	0	Нет	-	-	Нет	-	-

Рисунок 18 Скорости обмена в асинхронном режиме BRGH=1

И еще одно уточнение

Модуль USART аппаратный, и при своей работе он генерирует прерывания, поэтому удобнее работать именно по прерываниям. Флаги прерываний TXIF и RCIF доступны только для чтения и не могут быть сброшены программно, поэтому управлять приемом и передачей будем разрешая/запрещая прерывания. Можно конечно включать и отключать соответствующие модули, но отключение например передатчика переводит его линию в третье состояние, а это может неправильно истолковано приемником другой микросхемы (хотя и это решается подтяжкой к питанию))). Обработка принятой информации и формирование ответа производится в основном цикле программы.

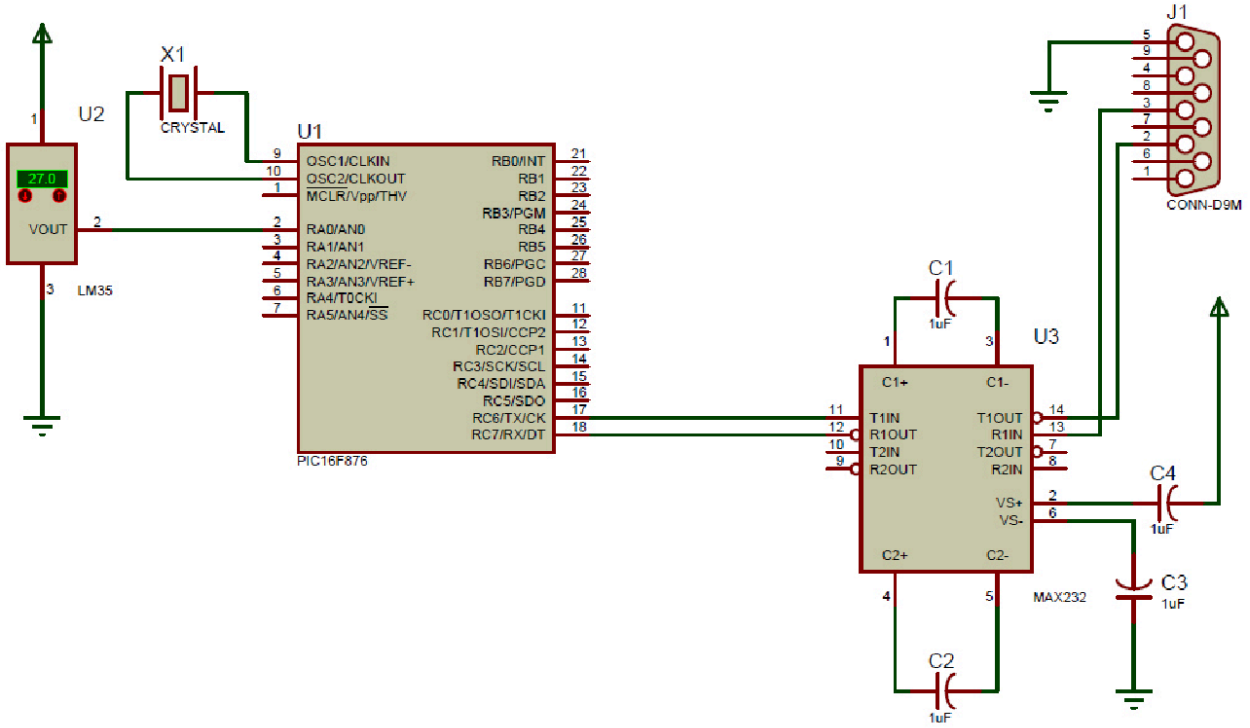


Рисунок 19 Реализация RS-232(COM) для PIC16

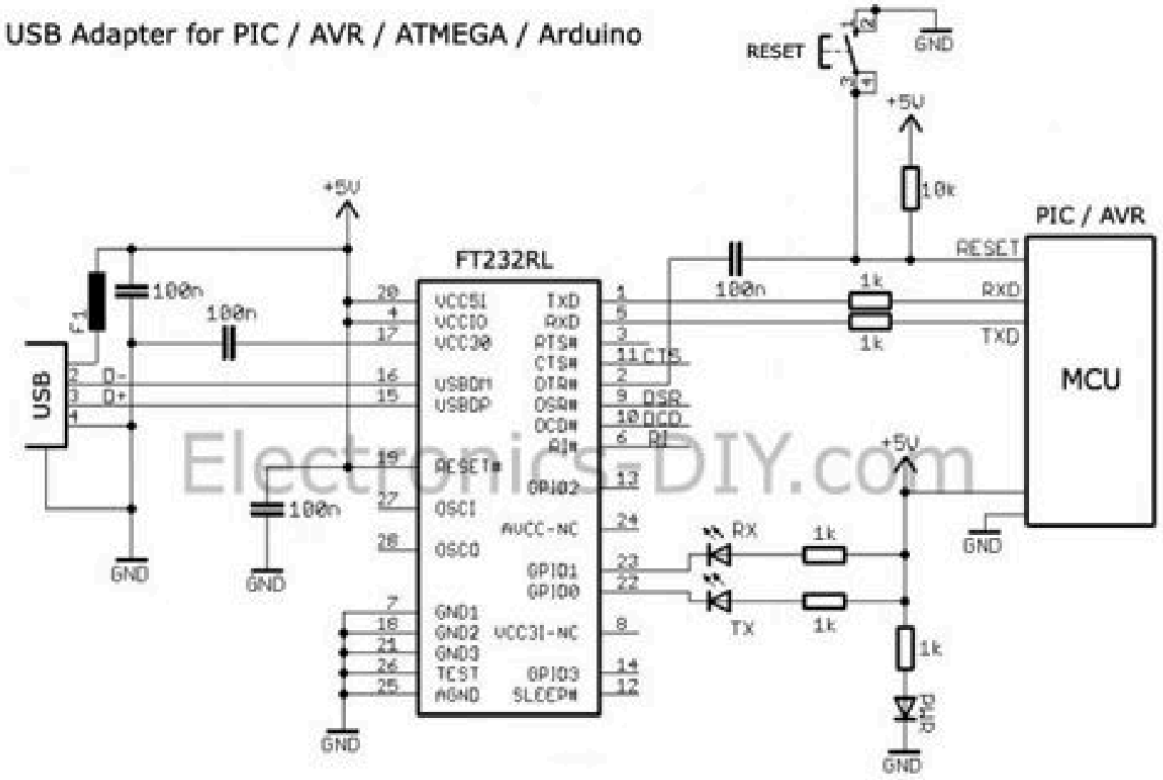


Рисунок 20 Реализация USB через UART протоколк

Вопросы для контроля

Источники

http://www.gaw.ru/html.cgi/txt/interface/iic/iic_3.htm

Меню MPLAB и описание функций – URL - http://disall.narod.ru/book/function.htm

PIC18F2420/2520/4420/4520 Data Sheet – URL - http://ww1.microchip.com/downloads/en/devicedoc/39631a.pdf

Барри Брей Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением C и Ассемблера – «МК-Пресс», СПб: «КОРОНА-ВЕК», 2008

PIC18F2420/2520/4420/4520 Data Sheet – URL - http://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf

USART/UART – url - http://picdevices.ru/eksperimentyi/usart-uart-apparatnyiy.html#more-1048