МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

Физический факультет

Кафедра физико-технической информатики

Сидорова Анна Александровна

ДОКЛАД

Шины передачи данных

4 курс, группа №15312i

Новосибирск — 2018

Оглавление

1. Введение

Шины предназначены для соединения двух или более каких-либо устройств, включая такие как CPU, оперативная память и устройства ввода/вывода. По шинам передаются следующие сигналы:

– Данные
– Адрес
– Управляющие сигналы

Кроме того в шине могут быть линии питания и земли.

1.1 Последовательные и параллельные шины

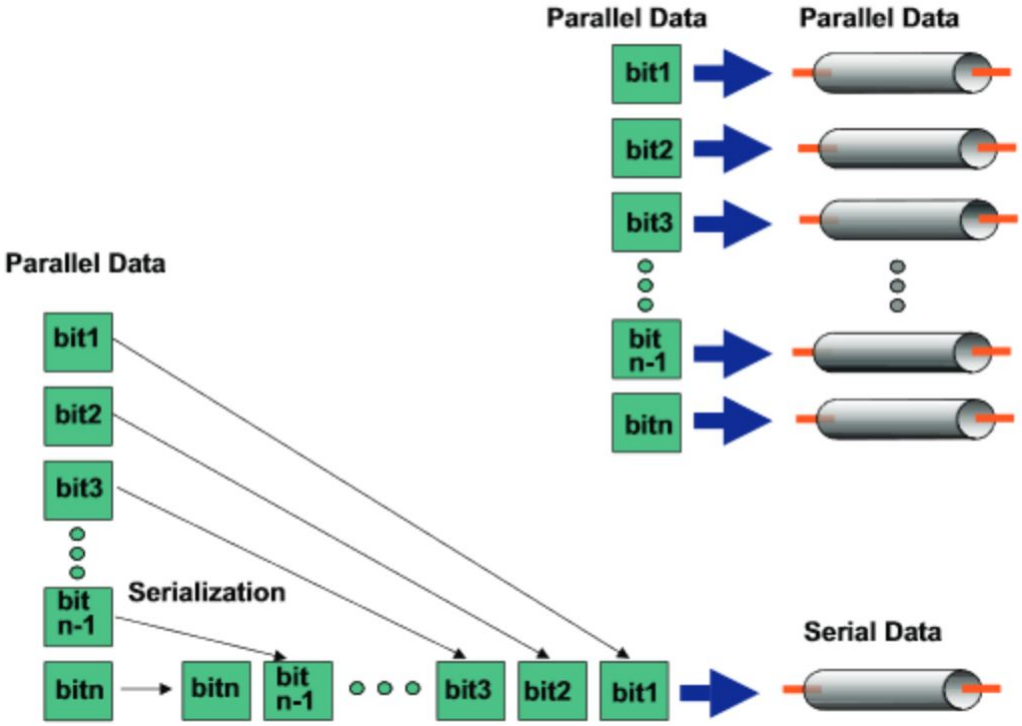

Рис. 1. Последовательная и параллельная передача данных [1]

Шины делятся на последовательные и паралелльные. Рис. 1 демонстрирует разницу между последовательной (*serial*) и параллельной (*parallel*) передачей данных. Шины параллельной передачи данных, такие как PCI, ATA, передают несколько битов одновременно, используя отдельную линию для каждого бита. Шины последовательной передачи данных, такие как USB, SPI, I2C, UART, передают биты один за другим, используя сериализацию и десериализацию. Очевидно, шины последовательной передачи данных

должны работать на более высокой частоте, чтобы достичь той же скорости передачи. Их примуществом является меньшее количество провода и занимаемого места, а так же отсутствие необходимости тщательного экранирования одной линии от другой на высоких частотах.

1.2 Синхронные и асинхронные шины

При **синхронной** передаче данных для управления потоком данных используются тактовые синхросигналы. При синхронной передаче кадры передаются через равные промежутки времени.

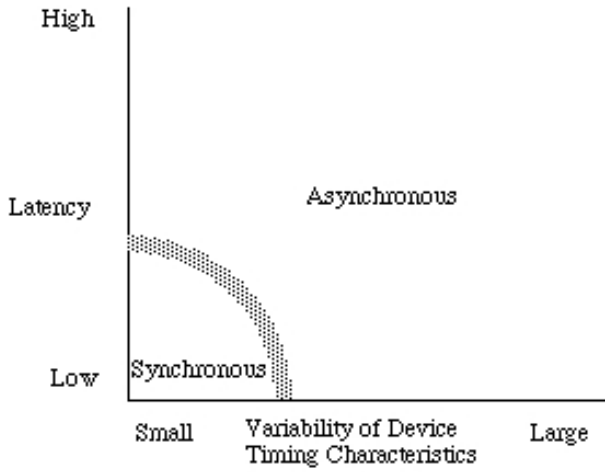

Рис. 2. Синхронные и асинхронные шины с точки зрения быстродействия и рабочей полосы частот [2]

Тактовая частота должна быть выбрана таким образом, чтобы сигнал от любой точки на шине мог достичь любой другой точки несколько раньше, чем завершится тактовый период, то есть шина должна допускать расхождение в моментах поступления тактовых импульсов. Ясно, что более короткие шины могут быть спроектированы на более высокую тактовую частоту.

Синхронные протоколы требуют меньше сигнальных линий, проще для понимания, реализации и тестирования. Поскольку для реализации синхронного протокола практически не требуется дополнительной логики, эти шины могут быть быстрыми и дешевыми. С другой стороны, они менее гибки, поскольку привязаны к конкретной максимальной тактовой частоте и, следовательно, к конкретному уровню технологии. По этой причине существующие

шины часто не в состоянии реализовать потенциал производительности подключаемых к себе новых устройств.

По синхронному протоколу обычно работают шины «процессор-память».

Асинхронная шина не тактируется. Вместо этого обычно используется старт-стопный режим передачи и протокол рукопожатия (handshaking) между источником и приемником данных на шине. Эта схема позволяет гораздо проще приспособить широкое разнообразие устройств и удлинить шину без беспокойства о перекосе сигналов синхронизации и о системе синхронизации.

Шины ввода/вывода обычно асинхронные.

Главный недостаток: построить асинхронную шину гораздо дороже, чем синхронную.

Выбор типа шины (синхронной или асинхронной) определяет не только пропускную способность, но также непосредственно влияет на емкость системы ввода/вывода в терминах физического расстояния и количества устройств, которые могут быть подсоединены к шине. Асинхронные шины по мере изменения технологии лучше масштабируются.

2. Обзор существующих шин последовательной передачи данных

2.1 I2C

I2C шина представляет собой две двунаправленные линии связи - SDA (Serial Data Line) и SCL (Serial Clock Line). По SDA передаются данные, по SCL тактовый сигнал.

Устройства на шине подразделяются на ведущих (*Master*) и ведомых (*Slave*). Данные передаются поразрядно, старшим разрядом вперед. Каждый разряд сопровождается тактовым сигналом (рис. 3).

Начало и окончание передачи данных сопровождается специальными состояниями шины - START и STOP. Эти состояния формирует ведущее устройство. Также ведущий может сформировать состояние повторного старта (REPEATED START) до формирования состояния STOP (рис. 4).

Данные передаются по I2C шине со служебной информацией. Все вместе это называется пакетом. Существуют адресные пакеты и пакеты данных.

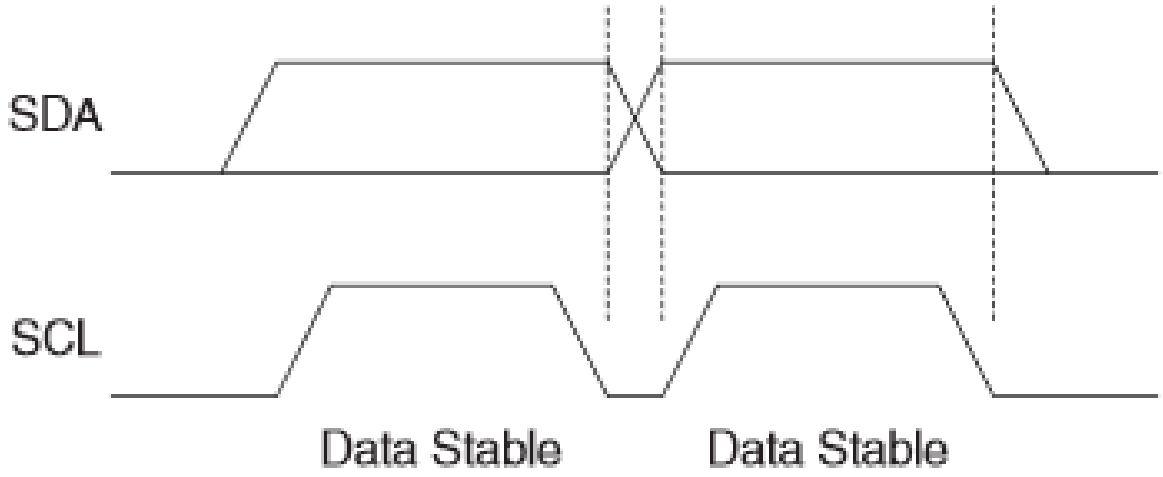

Рис. 3. Данные и тактовый сигнал на шине I2C [3]

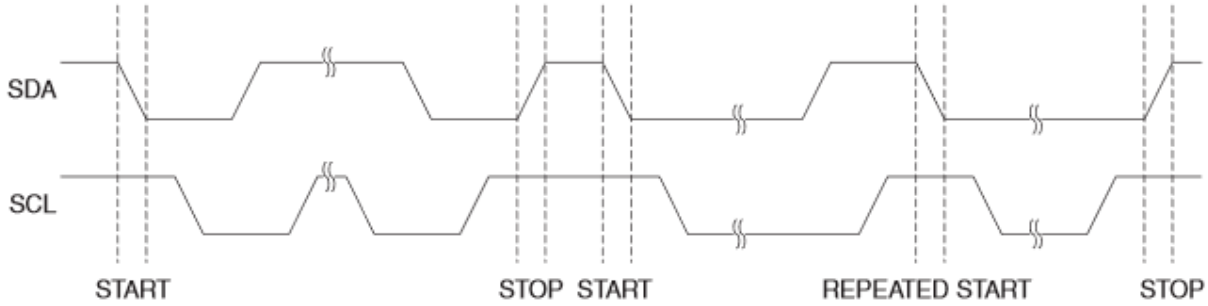

Рис. 4. START, STOP, REPEATED START на шине I2C [3]

Адресные пакеты (рис. 5) состоят из 7-и разрядного адреса, управляющего бита R/W и бита квитирования (весь пакет - 9 бит). Адресный пакет нужен, чтобы обратиться к конкретному устройству. Адрес 0000000 - это адрес общего вызова (обращение ко всем ведомым устройствам). R/W - определяет последующее направление передачи данных. Бит квитирования - это ответ ведомого устройства на принятый адрес. Если адрес распознан, ведомый выдает на линию SDA низкий уровень. В противном случае на линии удерживается высокий уровень.

Пакеты данных (рис. 6) состоят из байта данных и бита квитирования, то есть тоже имеют длину 9 бит. После приема каждого байта данных, принимающее устройство (приемник) отвечает передающему устройству (передатчику), устанавливая на линии SDA низкий уровень (это и есть бит квитирования). Если принимающее устройство получило последний байт или больше не может продолжать прием данных, оно должно "оставить"на линии SDA высокий уровень.

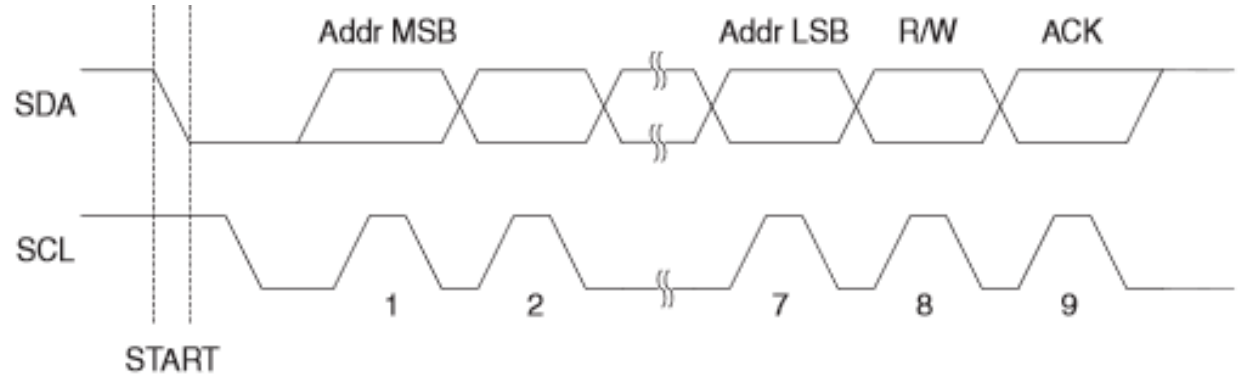

Рис. 5. Адресный пакет на шине I2C [3]

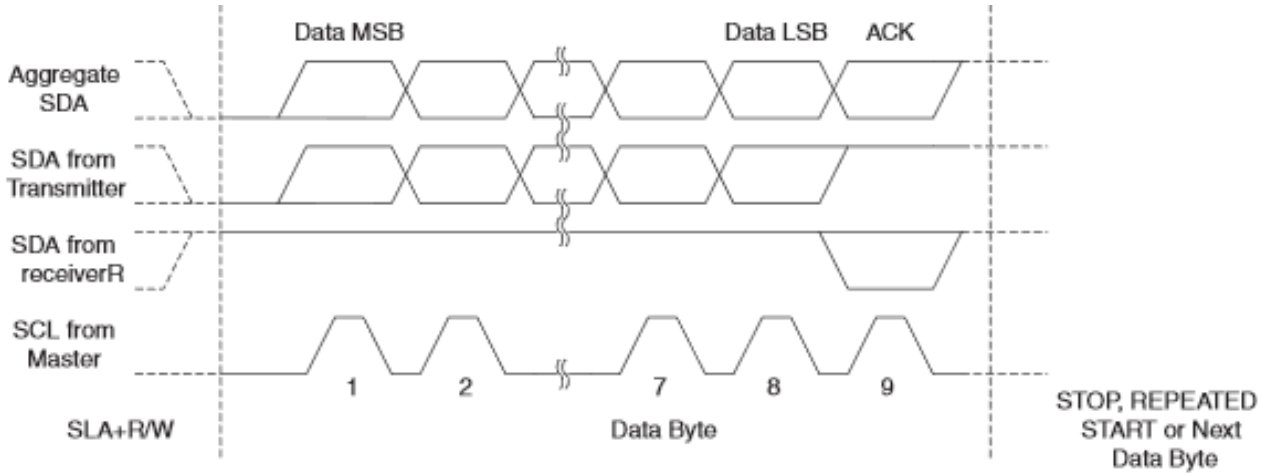

Рис. 6. Пакет данных на шине I2C [3]

Цикл обмена по I2C состоит из следующих шагов:

1. формирование состояния START
2. передача адресного пакета
3. передача пакетов данных
4. прием пакетов данных
5. формирование состояния STOP

2.2 Serial Peripheral Interface

Serial Peripheral Interface (далее SPI) представляет собой четырехпроводную синхронную шину, предназначенную для последовательного обмена данными между микросхемами. Данный интерфейс отличают простота использования и реализации, высокая скорость обмена и малая дальность действия [4].

При любом обмене данными по SPI одно из устройств является ведущим (*Master*), а другое ведомым (*Slave*). Ведущий переводит периферийное

устройство в активное состояние и формирует тактовый сигнал и данные. В ответ ведомое устройство передает ведущему свои данные. Передача данных в обе стороны происходит синхронно с тактовым сигналом.

Физически SPI реализуется на основе сдвигового регистра, который выполняет и функцию передатчика, и функцию приемника.

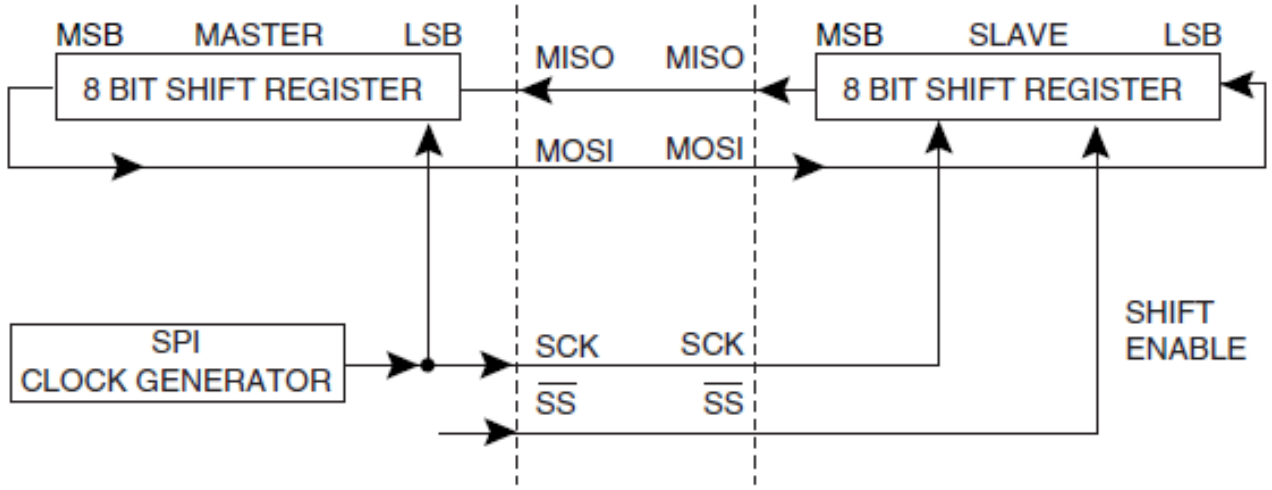

Рис. 7. Схема обмена данными по SPI [3]

Принцип обмена данными по SPI проиллюстрирован на рис. 7.

Сигналы, используемые данным интерфейсом, имеют следующее назначение:

- MOSI — Master Output / Slave Input. Выход ведущего / вход ведомого. Служит для передачи данных от ведущего устройства к ведомому.
- MISO – Master Input / Slave Output. Вход ведущего / выход ведомого. Служит для передачи данных от ведомого устройства к ведущему.
- SLK — Serial Clock. Сигнал синхронизации. Служит для передачи тактового сигнала всем ведомым устройствам.
- SS — Slave Select. Выбор ведомого. Служит для выбора ведомого устройства.

Протокол обмена по SPI аналогичен логике работы сдвигового регистра и заключается в последовательном побитном выводе/вводе данных по определенным фронтам тактового сигнала. Установка данных и выборка осуществляется по противоположным фронтам тактового сигнала.

2.3 UART

Для приема и передачи данных UART использует две линии данных (рис. 8) и земля:

– передающая данные (TX)

– принимающая данные (RX)

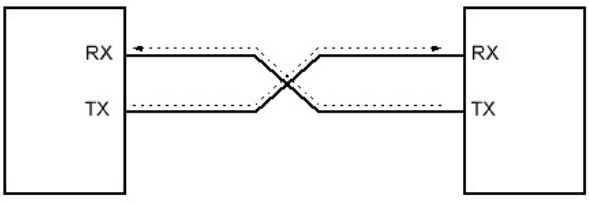

Рис. 8. Подключение UART

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 соответствует количество бит в секунду. Существует общепринятый ряд стандартных скоростей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400;460800; 921600 бод; Приемник и передатчик заранее договариваются о том на какой частоте будет идти обмен. Это очень важный момент! Если скорость передатчика и приемника не будут совпадать, то передачи может не быть вообще, либо будут считаны не те данные.

Байты данных отправляются в пакетах (рис. 9).

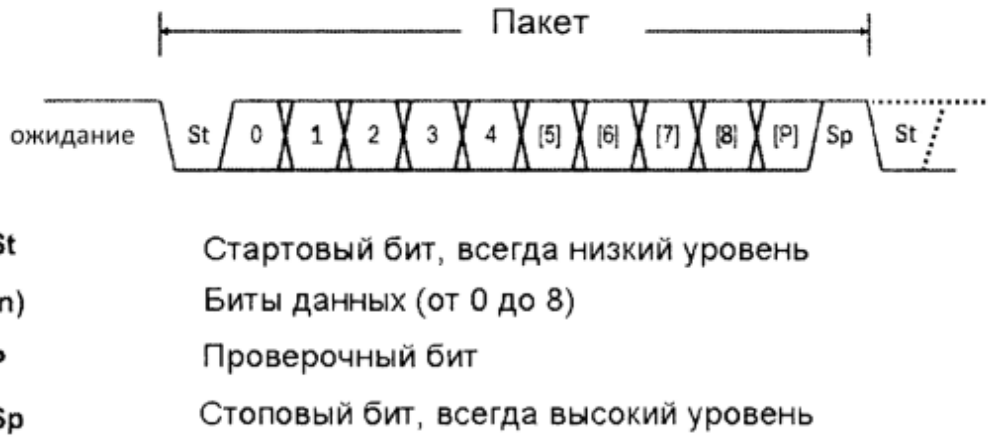

Рис. 9. UART-пакет

2.4 1-Wire

Интерфейс 1-Wire был предложен фирмой Dallas Semiconductor в конце 90-х годов прошлого века. Системы 1-Wire привлекательны благодаря легкости монтажа, низкой стоимости устройств, возможности распознавать устройство при подключении к функционирующей сети, большому числу устройств в сети и т.д. [5]

Типичная система 1-Wire состоит из управляющего контроллера (мастера или ведущего) и одного или нескольких устройств (ведомых), присоединенных к общей шине.

Режим передачи данных по шине 1-Wire – полудуплексный: мастер и ведомые устройства передают данные по очереди.

Пока шина свободна, она подтягивается к положительному уровню питания и удерживается в нем. В этот момент все устройства на шине заряжают внутренний конденсатор и получают энергию. Каждая транзакция через интерфейс 1-Wire начинается с того, что мастер передает импульс Reset: как только мастер хочет обратиться к какому-либо устройству, он опускает шину в ноль на заданный промежуток времени и возвращает её назад, затем ждет ответа – если на шине присутствует хотя бы один ведомый, он обязан ответить кратковременным опусканием же шины в ноль. Далее ведущий передаёт адрес, к кому конкретно он хочет обратится, команду и получает какие-либо данные.

2.5 Сравнение характеристик

Сравним [6] вышеописанные протоколы по следующим параметрам: скорость передачи данных, тип передачи данных (синхронно/асинхронно), сложность аппаратного обеспечения, количество линий, наличие дуплекса (таблица 1).

В результате сравнения мы пришли к заключению, что лучше использовать SPI, если есть только один ведущий и несколько ведомых устройств, так как SPI хорошо подходит для этого случая и наиболее быстродействующий. Когда имеется несколько ведущих устройств, помимо нескольких ведомых устройств, следует предпочесть I2C. В случае двух устройств, следует

	UART	SPI	I2C	1-Wire
Максимальная скорость	460 Кб/сек.	20 Мб/сек	3.4 Мб/сек	16 Кб/сек
Тип	Асинхронная	Синхронная	Синхронная	Асинхронная
Сложность	Низкая	Средняя	Высокая	Низкая
Дуплекс	Full	Full	Half	Half
Количество проводов	2	4	2	1

Таблица 1

Сравнение I2C, UART, SPI и 1-Wire

рассмотреть UART: его легко применять и с ним просто работать во многих периферийных устройствах.

3. Примеры использования интерфейсов передачи данных

Рассмотрим интерфейсы передачи данных на примере микроконтроллера Atmega16A. Согласно документации [3], данный микроконтроллер предоставляет возможность работы с интерфейсами TWI, SPI и USART.

3.1 TWI

TWI (Two Wire Interface) является аналогом I2C интерфейса фирмы Philips, только в случае AVR, не поддерживает высокие скорости передачи данных (свыше 400 kbit/s). Atmel использует другое название по лицензионным причинам.

3.1.1 Регистры

TWBR – TWI Bit Rate Register

Любой аппаратный модуль, реализующий интерфейс, имеет регистры задающие скорость передачи данных. В TWI модуле эту функцию выполняет регистр TWBR и два младших разряда статусного регистра TWSR - $TWPS_1$ и $TWPS_0$. Тактовая частота микроконтроллера, частота SCL сигнала и значение регистров TWBR и TWSR связаны следующим соотношением:

$$F_{scl} = \frac{F_{cpu}}{16 + 2TWBR * 4^{TWPS}}$$

, где $TWPS$ – значение младших разрядов регистра TWSR ($TWPS = 2TWPS_1 + TWPS_0$).

Рассмотрим пример расчета: тактовая частота микроконтроллера 16 МГц, нужно задать частоту SCL сигнала 100 кГц. Допустим, мы выбрали нулевое значение для битов $TWPS_1$ и $TWPS_0$. Выразив $TWBR$, получим:

$$TWBR = \frac{\frac{16000000}{100000} - 16}{2 * 4^0} = 72$$

При выборе частоты SCL сигнала нужно руководстоваться [4]:

1. Возможностями устройства, с которым производится обмен по I2C.
2. Тактовой частотой микроконтроллера. Она накладывает естественное ограничение на возможность задания максимальной частоты.
3. Электрическими характеристиками сети I2C. При большом количестве устройств и большой протяженности сети, передача данных на максимальной скорости может оказаться невозможной - сигналы будут искажаться из-за паразитной емкости сети или помех.

TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 10. TWI Control Register [3]

Рассмотри биты управляющего регистра TWCR (рис. 10):

– TWINT – флаг прерывания TWI модуля. Этот бит устанавливается аппаратно, когда TWI модуль завершает текущую операцию, а очищается программно, записью единицы.
– TWEA – разрешение бита подтверждения. Если бит TWEA установлен в 1, TWI модуль формирует сигнал подтверждения (ACK), когда это требуется.
– TWSTA – флаг состояния START. Когда этот бит устанавливается в 1, TWI модуль проверяет не занята ли шина и формирует состояние START.

– TWSTO – флаг состояния STOP. Когда этот бит устанавливается в 1 в режиме ведущего, TWI модуль выдает на шину состояние STOP и сбрасывает этот бит.
– TWWC – флаг конфликта записи. Этот флаг устанавливается аппаратно, когда выполняется запись в регистр данных (TWDR) при низком значении бита TWINT, то есть когда TWI модуль уже выполняет какие-то операции.
– TWEN – бит разрешения работы TWI модуля.
– TWIE – разрешение прерывания TWI модуля.

TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Рис. 11. TWI Control Register [3]

Статусный регистр TWSR (рис. 11) отражает состояние TWI модуля и двухпроводной шины, а также содержит разряды, задающие коэффициент деления частоты SCL сигнала.

Биты TWS7..TWS3 содержат статусный код. Биты доступны только для чтения, статусный код устанавливается TWI модулем аппаратно, после выполнения различных операций, например после формирования состояния START, передачи пакета данных и так далее. По значению статусного кода можно судить об успешности операции.

TWDR – TWI Data Register

В регистр данных мы помещаем то, что хотим передать/забираем то, что получили с шины.

TWAR – TWI (Slave) Address Register

Если микроконтроллер выступает в роли ведомого, ему необходим адрес, на который он будет отзываться. Регистр TWAR (рис. 12) предназначен для хранения 7-и разрядного адреса. Младший разряд (TWGCE) этого регистра

разрешает/запрещает микроконтроллеру отзываться на общие вызовы, то есть на пакеты с адресом 0000000.

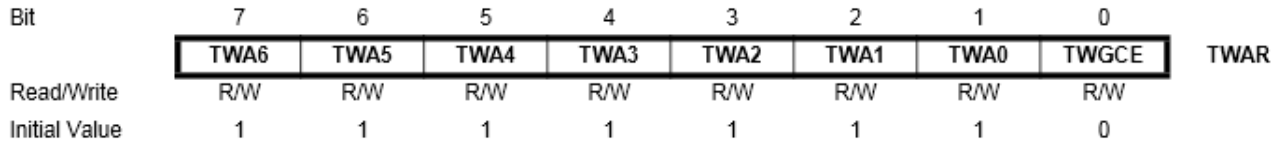

Рис. 12. TWI (Slave) Address Register [3]

3.1.2 Программный код

Листинг 3.1

Цикл передачи данных по TWI

```
1  // We want to transmit DATA to SLA_W address
2
3  // Send START condition
4  TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
5  // Wait for TWINT Flag set. This indicates that the START
       condition has been transmitted
6  while (!(TWCR & (1<<TWINT)));
7  // Check value of TWI Status Register. Mask prescaler bits.
       If status different from START go to ERROR
8  if ((TWSR & 0xF8) != START)
9      ERROR();
10 // Load Address into TWDR Register. Clear TWINT bit in TWCR
       to start transmission of address
11 TWDR = SLA_W;
12 TWCR = (1<<TWINT) | (1<<TWEN);
13 // Wait for TWINT Flag set. This indicates that the SLA+W
       has been transmitted, and ACK/NACK has been received
14 while (!(TWCR & (1<<TWINT)));
15 // Check value of TWI Status Register. Mask prescaler bits.
       If status different from MT_SLA_ACK go to ERROR
16 if ((TWSR & 0xF8) != MT_SLA_ACK)
17     ERROR();
18 // Load DATA into TWDR Register. Clear TWINT bit in TWCR to
       start transmission of data
19 TWDR = DATA;
20 TWCR = (1<<TWINT) | (1<<TWEN);
```

```
// Wait for TWINT Flag set. This indicates that the DATA has
    been transmitted, and ACK/NACK has been received.
while (!(TWCR & (1<<TWINT)));
// Check value of TWI Status Register. Mask prescaler bits.
    If status different from MT_SLA_ACK go to ERROR
if ((TWSR & 0xF8) != MT_DATA_ACK)
    ERROR();
// Transmit STOP condition
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
```

3.2 SPI

3.2.1 Регистры

SPCR

Кофигурация модуля SPI устанавливается с помощью регистра SPCR (рис. 13).

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 13. SPI Control Register [3]

Рассмотрим этот регистр побитово:

- SPIE — разрешает /запрещает прерывания от модуля SPI. Если бит установлен в 1, прерывания от SPI разрешены.
- SPE — включает/выключает модуль SPI. Если бит установлен в 1, модуль SPI включен.
- DORD — определяет порядок передачи данных. Когда бит установлен в 1, содержимое регистра данных передается младшим битом вперед. Когда бит сброшен, то старшим битом вперед.
- MSTR — определяет режим работы микроконтроллера. Если бит установлен в 1, микроконтроллер работает в режиме Master (ведущий). Если бит сброшен – в режиме Slave (ведомый). Обычно микроконтроллер работает в режиме master.

– CPOL и CPHA — определяют в каком режиме работает SPI модуль. Требуемый режим работы зависит от используемого периферийного устройства.
– SPR1 и SPR0 -– определяют частоту тактового сигнала SPI модуля, то есть скорость обмена. Максимально возможная скорость обмена всегда указывается в спецификации периферийного устройства.

SPSR

Статусный регистр SPSR (рис. 14) предназначен для контроля состояния SPI модуля, кроме того он содержит дополнительный бит управления скоростью обмена.

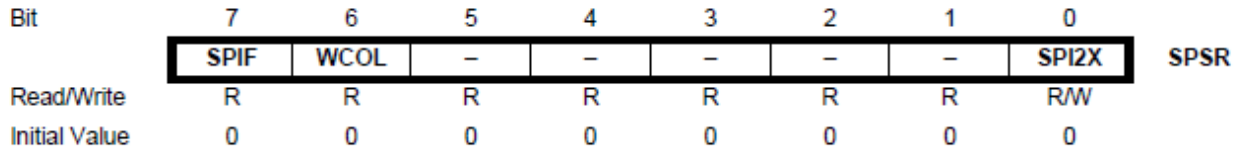

Рис. 14. SPI Status Register [3]

Рассмотрим значение каждого бита:

– SPIF – флаг прерывания от SPI. Он устанавливается в 1 по окончании передачи байта данных. Если разрешены прерывания модуля, одновременно с установкой этого флага генерируется прерывание от SPI. Также этот флаг устанавливается в 1 при переводе микроконтроллера из режима master в режим slave с помощью вывода SS.

  Сброс флага происходит аппаратно, при вызове подпрограммы обработки прерывания или после чтения регистра SPSR с последующим обращением к регистру данных SPDR.
– WCOL – флаг конфликта записи. Флаг устанавливается в 1, если во время передачи данных выполняется попытка записи в регистр данных SPDR. Флаг сбрасывается аппаратно после чтения регистра SPSR с последующим обращением к регистру данных SPDR.
– SPI2X – бит удвоения скорости обмена. Установка этого разряда в 1 удваивает частоту тактового сигнала SCK. Микроконтроллер при этом должен работать в режиме master.

SPDR

Для передачи и приема данных предназначен регистр SPDR (SPI Data Register). Запись данных в этот регистр инициирует передачу данных SPI модулем. При чтении этого регистра, считывается содержимое буфера сдвигового регистра SPI модуля.

3.2.2 Программный код

Минимальный программный код для работы с SPI модулем состоит из двух функций:

- функции инициализации.
- функции передачи/приема байта данных

Master

Листинг 3.2

Функции для работы с SPI

```
1  void SPI_MasterInit(void)
2  {
3      /* Set MOSI and SCK output, all others input */
4      DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
5      /* Enable SPI, Master, set clock rate fck/16 */
6      SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
7  }
8
9  void SPI_MasterTransmit(char cData)
10 {
11     /* Start transmission */
12     SPDR = cData;
13     /* Wait for transmission complete */
14     while (!(SPSR & (1<<SPIF)));
15 }
```

Slave

Листинг 3.3

Функции для работы с SPI

```
1  void SPI_SlaveInit(void)
2  {
```

```
3       /* Set MISO output, all others input */
4       DDR_SPI = (1<<DD_MISO);
5       /* Enable SPI */
6       SPCR = (1<<SPE);
7   }
8
9   char SPI_SlaveReceive(void)
10  {
11      /* Wait for reception complete */
12      while (!(SPSR & (1<<SPIF)));
13      /* Return data register */
14      return SPDR;
15  }
```

3.3 USART

USART это более гибкий в настройки UART с дополнительными возможностями. Например в USART можно регулировать длину слова с более большим диапазоном (от 5 до 9), чем в UART (от 8 до 9). В USART возможна как асинхронная, так и синхронная передача данных (в UART только асинхронная). При синхронной передачи помимо линий данных и питания, используется дополнительная линия с синхросигналом. С такой конфигурацией USART уже пересекается с интерфейсом SPI и его можно использовать как «ведущий» в интерфейсе SPI.

Мы будем рассматривать классический случай, когда интерфейс асинхронный. В таком режиме USART полностью совместим с UART [3].

3.3.1 Регистры

UDR – USART I/O Data Register

Регистр данных UART. При передаче в него записываются данные, которые необходимо отправить, а при чтении — принятые данные.

UCSRA – USART Control and Status Register A

См. рис 15.

– RXC – флаг окончания приема данных. Устанавливается в 1 при наличии несчитанных данных и сбрасывается в 0 по окончании приема данных.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Рис. 15. UCSRA [3]

- TXC – флаг окончания передачи данных. Сбрасывается в 0 по окончании передачи данных и устанавливается в 1 при наличии непереданных данных.
- UDRE – флаг, означающий готовность регистра UDR получать новые данные. Когда UDRE равен 1, регистр UDR пуст и готов к приему новых данных.
- FE – флаг ошибки фрейма.
- DOR – флаг переполнения регистра данных.
- PE – флаг ошибки четности
- U2X – бит, позволяющий увеличить скорость передачи вдвое. При записи 1 в данный бит предделитель тактовой частоты модуля UART уменьшается вдвое, что позволяет вдвое увеличить скорость передачи данных.
- MPCM – мультипроцессорный режим коммуникации.

UCSRB – USART Control and Status Register B

См. рис 16.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 16. UCSRB [3]

- RXCIE – Бит, разрешающий или запрещающий генерацию прерывания по окончании приема. При записи 1 в данный бит прерывание по окончании приема разрешено, при записи 0 — запрещено.
- TXCIE – При записи 1 в данный бит прерывание по окончании передачи данных разрешено, при записи 0 — запрещено.

– UDRIE – Бит, разрешающий или запрещающий возникновение прерывания по флагу UDRE.
– RXEN – Запись 1 в данный бит включает приемник UART модуля, запись 0 — выключает.
– TXEN – Запись 1 в данный бит включает передатчик UART модуля, запись 0 — выключает.
– UCSZ2 – В паре с битами UCSZ1 и UCSZ0 задает число передаваемых бит. То есть мы можем передавать не только побайтно, но и по 5, 6, 7, 8, 9 бит.
– RXB8 – девятый бит принимаемых данных при передаче по 9 бит. Должен быть считан перед операциями с регистром UDR.
– TXB8 – девятый бит отсылаемых данных при передаче по 9 бит. Должен быть записан перед записью других бит в UDR.

UCSRC – USART Control and Status Register C

См. рис. 17.

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Рис. 17. UCSRC [3]

– URSEL – бит выбора между регистрами UCSRC и UBRRH. Слегка странный бит, однако, достаточно запомнить, что при записи в него 1, работа происходит с регистром UCSRC, а при записи 0 — с регистром UBRRH. То есть последующие биты будут записаны в соответствующий регистр.
– UMSEL– при записи 1 в данный бит происходит синхронная передача данных, а при записи 0 – асинхронная.
– UPM0,UPM1 – Данные биты позволяют настроить контроль четности передаваемых данных.
– USBS – Данный бит позволяет выбрать число стоп-битов. При записи 1 в данный бит число стоп-битов станет равно 2, а при записи 0 — 1.

– UCSZ1, UCSZ0 – В совокупности с битом UCSZ2 регистра UCSRB данные биты позволяют выбрать число передаваемых бит.
– UCPOL – Данный бит позволяет настроить, по какому фронту будет происходить обмен данными при синхронном режиме передачи. При работе в асинхронном режиме передачи необходимо установить данный бит в 0.

UBRRL and UBRRH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Рис. 18. UBRR [3]

Регистры UBRRL и UBRRH (рис. 18) отвечают за настройку скорости работы приемопередатчика. Следует помнить, что при работе с регистром UBRRH бит URSEL должен быть равен 0.

Для получения необходимой скорости работы UART значение UBRR(Пары регистров UBRRH и UBRRL) рассчитывается по следующей формуле:

$$UBRR = \frac{F_{cpu}}{16B} - 1$$

Где F_{cpu} — частота работы микроконтроллера (Гц), B — необходимая скорость работы UART (Бит/сек).

Например, необходимо настроить скорость передачи 9600 бит/ сек при работе микроконтроллера на частоте 8 МГц:

$$UBRR = \frac{8000000}{16 * 9600} - 1 = 51.083333$$

Округляем до целого числа, $UBRR = 51$. И уже данное число записываем в регистры UBRRL:UBRRH.

3.3.2 Программный код

Инициализация

Листинг 3.4

Функции для работы с SPI

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1

void main(void)
{
    // ...
    USART_Init (MYUBRR);
    // ...
}

void USART_Init(unsigned int ubrr)
{
    /* Set baud rate */
    UBRRH = (unsigned char) (ubrr >> 8);
    UBRRL = (unsigned char) ubrr;
    /* Enable receiver and transmitter */
    UCSRB = (1<<RXEN)|(1<<TXEN);
    /* Set frame format: 8data, 2stop bit */
    UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Передача данных

Листинг 3.5

Функции для работы с SPI

```
void USART_Transmit(unsigned char data)
{
    /* Wait for empty transmit buffer */
    while (!(UCSRA & (1<<UDRE)));
    /* Put data into buffer, sends the data */
    UDR = data;
}
```

Приём данных

Листинг 3.6

Функции для работы с SPI

```
unsigned char USART_Receive(void)
{
    /* Wait for data to be received */
    while (!(UCSRA & (1<<RXC)));
    /* Get and return received data from buffer */
    return UDR;
}
```

Список литературы

1. Comparing Bus Solutions / F. Alicke [et al.]. — 03/2000. — URL: www.ti.com/lit/an/slla067c/slla067c.pdf.
2. *Weems C.* Lecture 12: Buses.
3. Atmega16a datasheet / Atmel corporation. — 2014. — URL: ww1.microchip.com/downloads/en/devicedoc/atmel-8154-8-bit-avr-atmega16a_datasheet.pdf.
4. *Бобков П.* Учебный курс AVR. — 2012. — URL: chipenable.ru.
5. *Елисеев Н.* Интерфейс 1-Wire: устройство и применение. — 2007.
6. *Patrick J.* Serial Protocols Compared. — 05/2002. — URL: www.embedded.com/design/connectivity/4023975/Serial-Protocols-Compared.