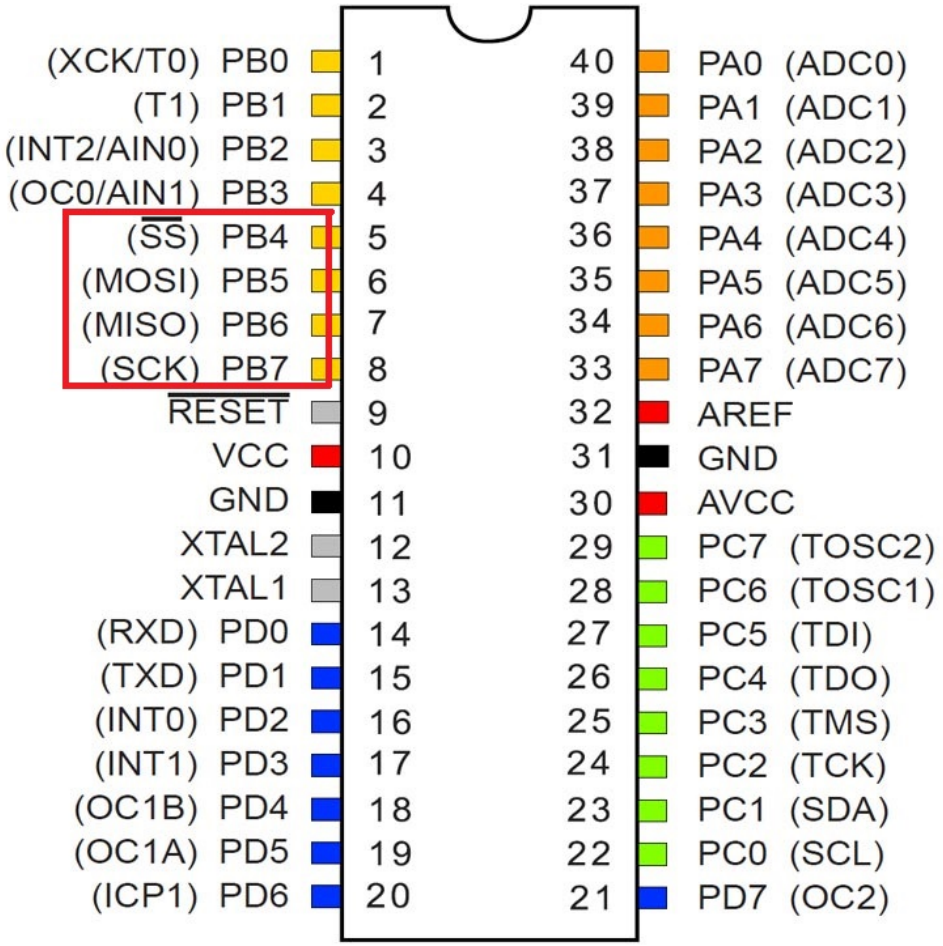
Done at Home
Программирование микроконтроллеров avr

AVR микроконтроллеры для начинающих (урок 10) SPI-интерфейс

admin | 13.01.2014 0 Comment

AVR микроконтроллеры для начинающих (урок 10) SPI-интерфейс

Аббревиатура SPI означает «Serial Peripheral Interface» или в русском варианте «последовательный периферийный интерфейс». Название говорит само за себя, данный интерфейс используется для работы с различными периферийными устройствами. Например, это могут быть различные ЦАП/АЦП, потенциометры, датчики, расширители портов ввода/вывода, различная память и даже более сложная периферия. Интерфейс SPI, наряду с I2C, относится к самым широко-используемым интерфейсам для соединения микросхем.



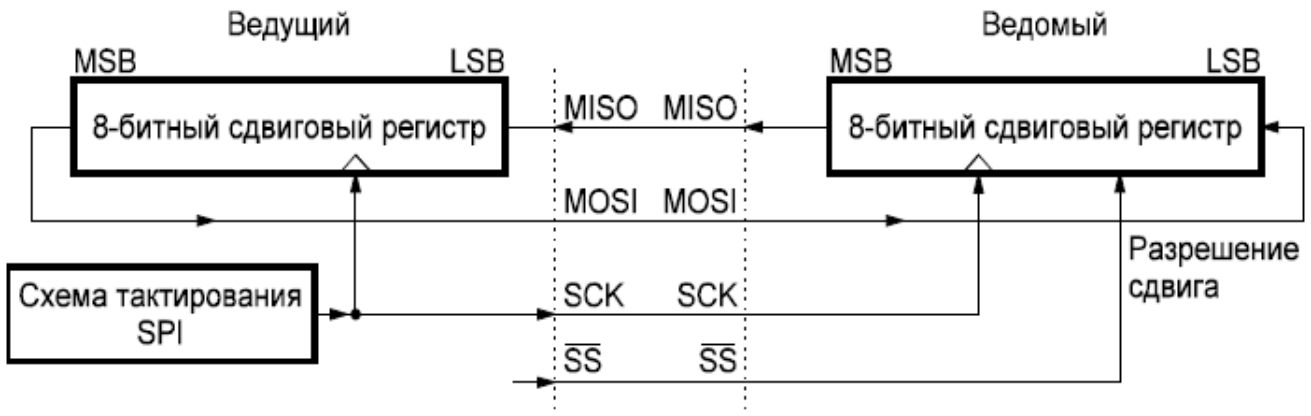
С технической точки зрения SPI – это синхронная четырёхпроводная шина. Она представляет собой соединение двух синхронных сдвиговых регистров, которые является центральным элементом любого SPI устройства. Для соединения используется конфигурацию ведущий/ведомый. Только ведущий может генерировать импульсы синхронизации. В схеме всегда только один ведущий, количество ведомых может быть различно. В общем случае выход ведущего соединяется со входом ведомого, и наоборот, выход ведомого соединяется со входом ведущего. При подаче импульсов синхронизации на выход SCK, данные выталкиваются ведущим с выхода MOSI, и захватываются ведомым по входу MISO. Таким образом, если подать количество импульсов синхронизации соответствующее разрядности сдвигового регистра, то данные в регистрах обменяются местами. Отсюда следует, что SPI всегда работает в полнодуплексном режиме.

Выходы SPI:

MOSI — Master Output, Slave Input (выход ведущего,вход ведомого). Данный сигнал предназначен для последовательной передачи данных от ведущего к ведо-
мому.
MISO — Master Input, Slave Output (вход ведущего, выход ведомого). Данный сигнал предназначен для последовательной передачи данных от ведомого к ведущему.
SCK — Serial Clock (сигнал синхронизации). Используется для синхронизации при передаче данных.
SS — Chip Select (выбор микросхемы). С помощью данного сигнала происходит активация ведомого устройства. Обычно он является инверсным, то есть низкий уровень считается активным.

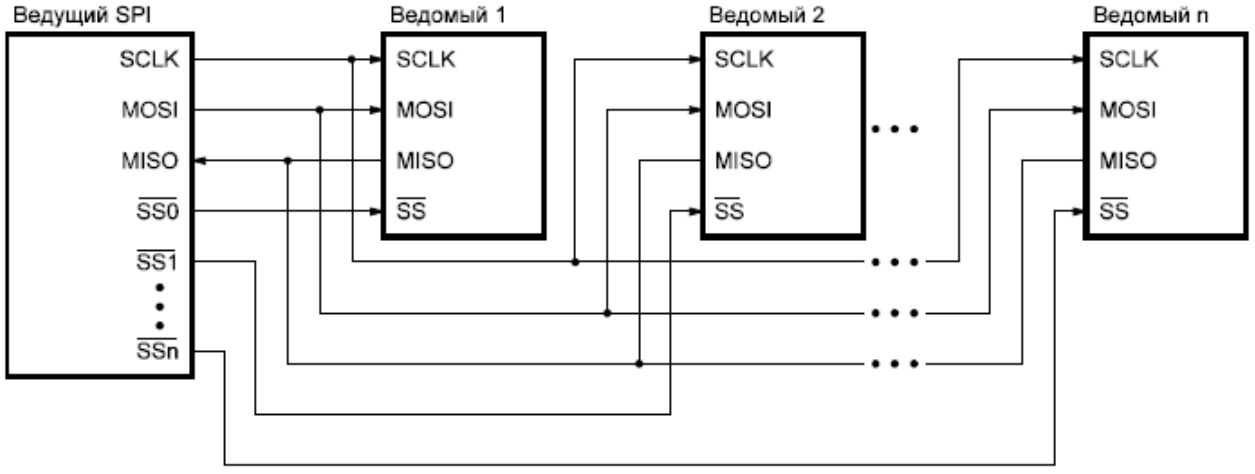
Подключение ведущего к ведомому:

SPI устроен просто – два сдвиговых регистра (master и slave)(ведущий и ведомый), плюс генератор на стороне master – все! Сдвиговые регистры замкнуты в кольцо по линиям MOSI и MISO, тактовый сигнал от генератора подается на оба сдвиговых регистра (на slave через линию SCK). До начала обмена данные помещаются в сдвиговые регистры, мастер запускает генератор, генератор «отщелкивает» 8 тактов, за эти 8 тактов сдвиговые регистры «меняются» содержимым.

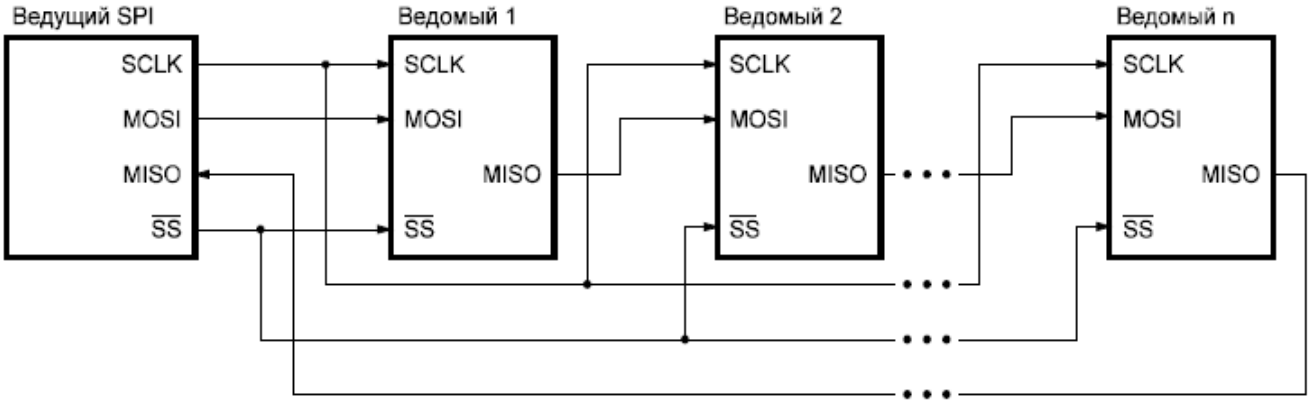


При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение, либо каскадное (последовательное). Независимое подключение более распространенное, т.к. достигается при использовании любых SPI-совместимых микросхем. Здесь, все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины, переводом того или иного сигнала SS в низкое состояние, задает, с какой ведомой микросхемой он будет обмениваться

данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации 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 микросхемы соединяется со входом приема данных другой1. Входы выбора микросхем здесь соединены параллельно и, таким образом, общее число линий связи сохранено равным 4.



Независимое подключение



Каскадное подключение

Ведущий или ведомый:

Если SPI настроен как ведущий (Master), то управление линией SS происходит не автоматически. Данная операция должна быть выполнена программно перед началом сеанса связи. После этого, запись в регистр данных SPI инициирует генерацию импульсов синхронизации и аппаратный сдвиг 8-ми бит в подчиненное устройство. По окончании сдвига одного байта генератор импульсов синхронизации SPI останавливается, при этом устанавливая флаг окончания передачи (SPIF). Если установлен бит SPIE в регистре SPCR, то разрешается прерывание SPI и по окончании передачи байта будет генерирован запрос на прерывание. Мастер может продолжить сдвигать следующий байт, если записать его в регистр SPDR, или подать сигнал окончания пакета путем установки низкого уровня на линии SS . Последний принятый байт сохраняется в буферном регистре.
В режиме подчиненного (Slave), интерфейс SPI находится в состоянии ожидания, в котором MISO переводится в третье состояние, до тех пор, пока на выводе SS присутствует высокий уровень. В этом состоянии программа может обновлять содержимое регистра данных SPI (SPDR), но при этом входящие импульсы синхронизации не сдвигают данные до подачи низкого уровня на вывод SS . После того как один байт был полностью сдвинут, устанавливается флаг окончания передачи SPIF. Если установлен бит

разрешения прерывания SPI (SPIE) в регистре SPCR, то установка флага SPIF приводит к появлению запроса на прерывание. Подчиненный может продолжать размещать новые данные для передачи в регистр SPDR перед чтением входящих данных. Последний принятый байт хранится в буферном регистре.

Table 18-1. Выводы SPI

Выводы	Направление в режиме ведущего	Направление в режиме ведомого
MOSI	**Определяется пользователем**	**Вход**
MISO	**Вход**	**Определяется пользователем**
SCK	**Определяется пользователем**	**Вход**
$\overline{SS}$	**Определяется пользователем**	**Вход**

В микроконтроллере atmega32 для работы с модулем SPI используются три регистра:

- управляющий регистр SPCR,
- статусный регистр SPSR,
- регистр данных SPDR.

Управляющий регистр SPCR

7	6	5	4	3	2	1	0	
SPIE	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**	**SPCR**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

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.
MTSR – 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 - полярность синхронизации. Если данный бит равен лог. «1», то SCK имеет высокий уровень в состоянии ожидания. Если CPOL=0, то SCK имеет низкий уровень в состоянии ожидания.
CPHA – Фаза синхронизации.
SPR1 и **SPR0** – определяют частоту тактового сигнала SPI модуля, то есть скорость обмена. Максимально возможная скорость обмена всегда указывается в спецификации периферийного устройства.

Статусный регистр SPSR

7	6	5	4	3	2	1	0	
SPIF	**WCOL**	**–**	**–**	**–**	**–**	**–**	**SPI2X**	**SPSR**
R	R	R	R	R	R	R	R/W	
0	0	0	0	0	0	0	0	

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.

Table 18-4. Relationship Between SCK and the Oscillator Frequency

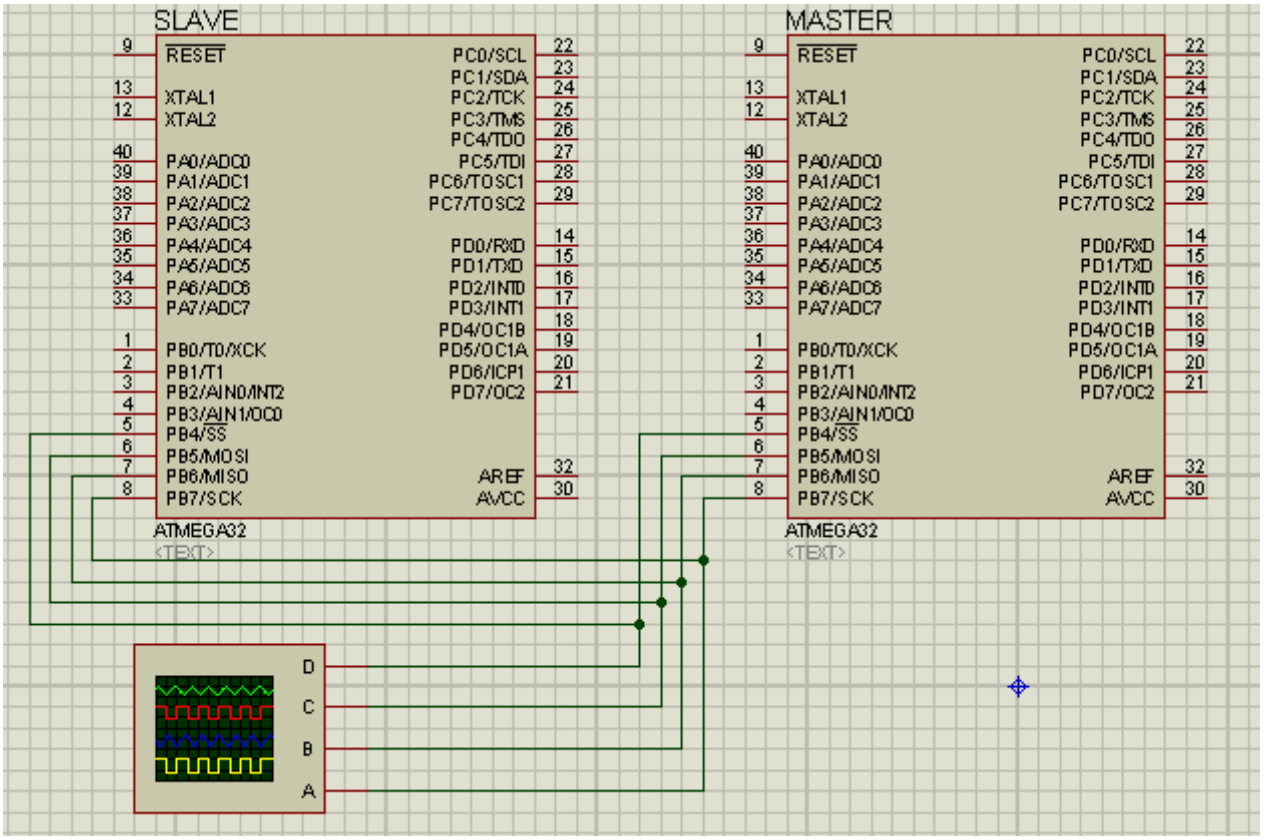
SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Регистр данных SPDR

7	6	5	4	3	2	1	0	
MSB							**LSB**	**SPDR**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
X	X	X	X	X	X	X	X	Undefined

Запись данных в этот регистр инициирует передачу данных SPI модулем. При чтении этого регистра, считывается содержимое буфера сдвигового регистра SPI модуля.

Пример:



Напишем 2-а кода: для ведущего и ведомого. Ведущий будет каждую секунду отправлять ведомому по SPI интерфейсу число насчитанных секунд. Ведомый будет принимать это значение и выводить его на PORTA.

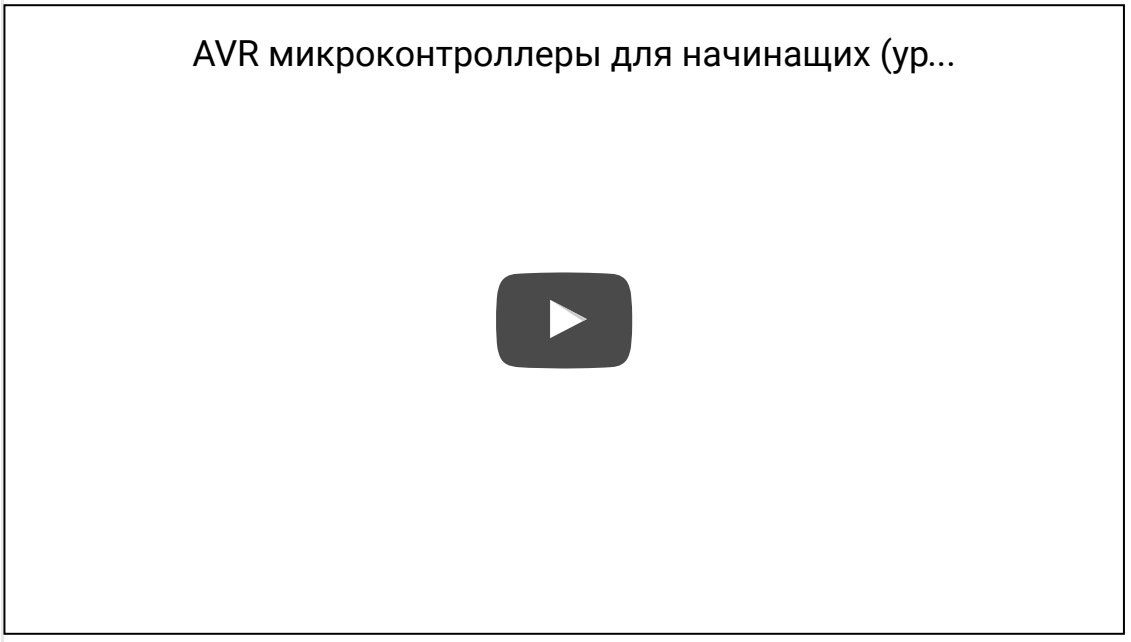
Ведущий (MASTER) :

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

// Функция инициализации мастера шины SPI
void SPI_MasterInit(void)
{
// Установка выводов SPI на вывод
DDRB = 0xFF;
//Включение SPI, режима ведущего, и установка частоты тактирования fclk/128
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);
}

/* Функция передачи байта данных outData. Ожидает окончания
передачи и возвращает принятый по ножке MOSI байт */
unsigned char SPI_MasterTransmit(char outData)
{
// Начало передачи
SPDR = outData;
// Ожидание окончания передачи
while(!(SPSR & (1<<SPIF))) ;
return SPDR; //возвращаем принятый байт
}

int main(void) {
char i=0;
SPI_MasterInit();
while(1)
{
i+=1;
SPI_MasterTransmit(i);
```

```
32 _delay_ms(1000);
33 }
34 }
```

Ведомый (SLAVE) :

```
1  #define F_CPU 1000000UL
2  #include <avr/io.h>
3  #include <util/delay.h>
4  #include <avr/interrupt.h>
5
6  // Функция инициализации мастера шины SPI
7  void SPI_SlaveInit(void)
8  {
9  // Установка выводов MISO и SCK на вход
10 DDRB = 0x00;
11 //Включение SPI, режима ведомого, разрешаем прерывания
12 SPCR = (1<<SPE)|(1<<SPIE);
13 }
14
15 ISR(SPI_STC_vect) //прерывание по приему байта
16 {
17 PORTA=SPDR; //выводим байт на PORTA
18 }
19
20 int main(void)
21 {
22 SPI_SlaveInit();
23 sei();
24 while(1);
25 }
```



Рубрика: Все посты AVR Уроки AVR

Вордпресс