طراحی سیستمهای ریزپردازنده

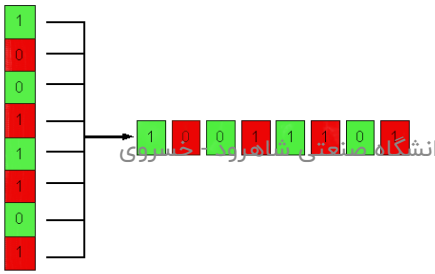
رابطهای سریال SPI و I2C

دانشکده مهندسی برق

دانشگاه صنعتی شاهرود

حسین خسروی

۱۴۰۰



Serial Peripheral Interface (SPI) Bus

- هیچ مشخصات رسمی برای رابط SPI وجود ندارد
  - مراجعه به برگه مشخصات قطعه مورد استفاده ضروری است
- مناسب برای برای اتصال مدارهای مجتمع روی بُرد
- پارامترهای مهم: فرکانس های کاری مجاز و نوع انتقال معتبر
- پیکربندی پایه – پیرو (یک یا چند پیرو)
- پایه (فرمانده) معمولا یک ریزپردازنده است

سایر ویژگیهای رابط SPI

- تولید کنندگان متعددی این رابط را در قطعات خود استفاده می کنند (ابتدا موتورولا)

- محصولات سازندگان مختلف به طور مستقیم با هم سازگار نیستند!
  - لیکن معمولا با تغییرات کوچک در پارامترها، امکان هماهنگی هست

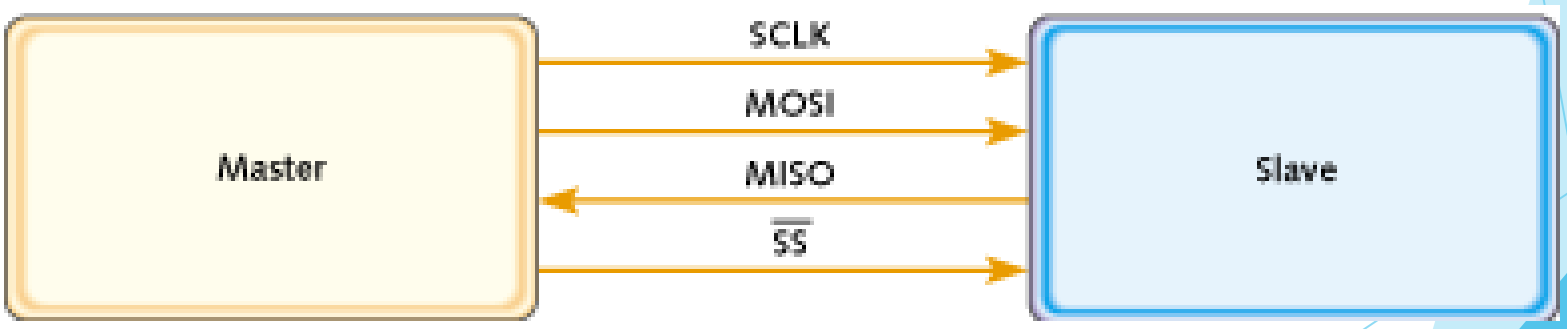
- برخی قطعاتی که از SPI استفاده می کنند:
  - Flash EEPROM, ADC, DAC, temperature sensor, digital IO,RTC digital potentiometer etc.

سیگنالهای SPI

- گذرگاه SPI شامل چهار سیگنال است:
- SCLK - Serial Clock (output from master)
- MOSI - Master Output, Slave Input (output from master)
- MISO - Master Input, Slave Output (output from slave)
- SS - Slave Select (active low; output from master)
  - No Slave Addressing
- سایر نامهایی که برای این سیگنالها استفاده می شود:
- SCK, CLK - Serial Clock (output from master)
- SDI, DI, SI - Serial Data In
- SDO, DO, SO - Serial Data Out
- SSEL - Slave Select

Typical SPI Configuration

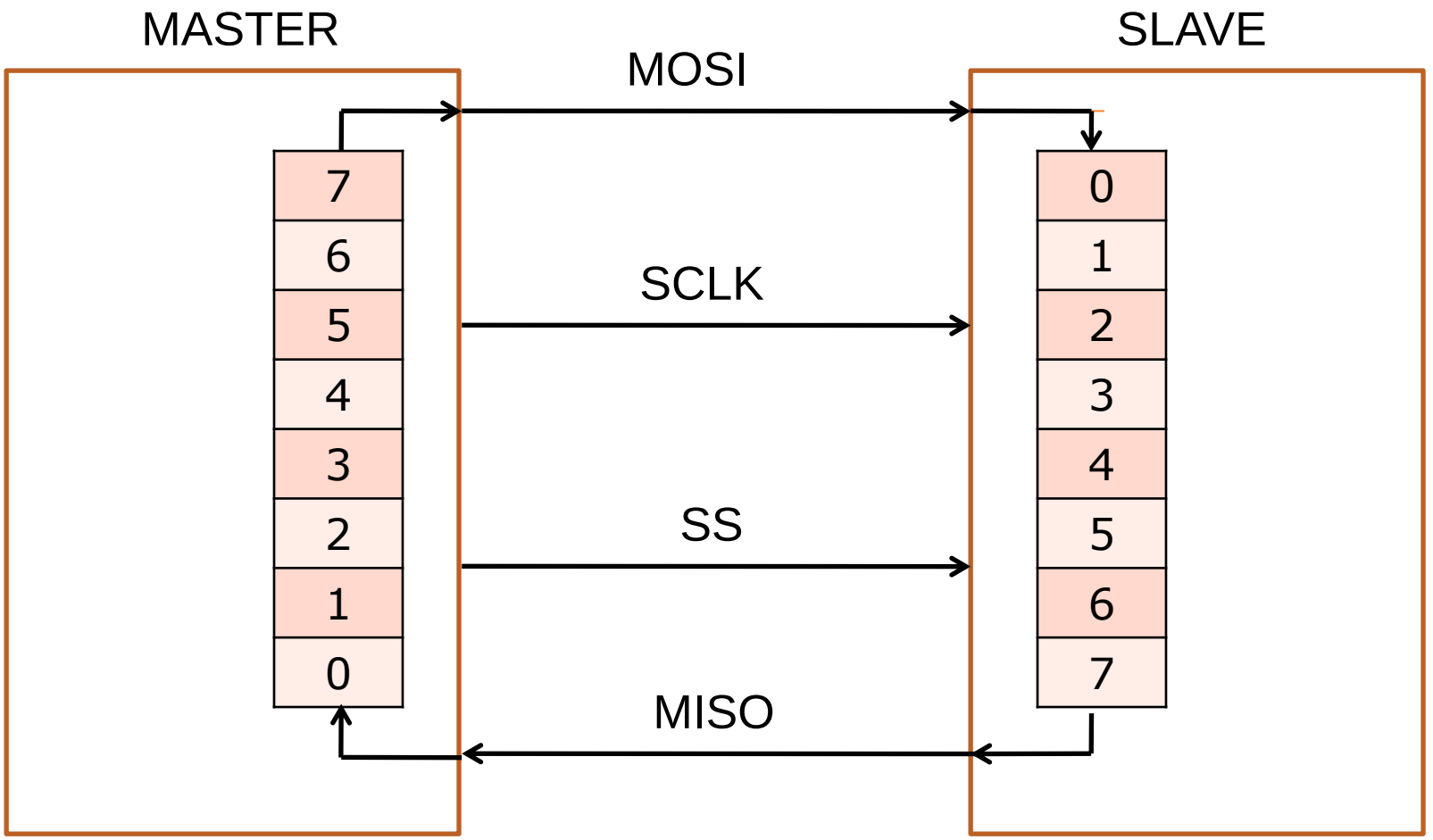
- فرمانده سیگنال Slave Select را صفر می کند و سپس پالسهای ساعت را ایجاد می کند
- فرکانس ساعت در این پروتکل مشخص نشده است و بسته به ویژگی های دستگاه پیرو، می تواند از ۱ تا ۷۰ مگاهرتز باشد
- سپس انتقال داده انجام می شود
- در نهایت فرمانده، پیرو را از حالت انتخاب خارج می کند



Simple master slave implementation

- During each SPI clock cycle, a full duplex data transmission occurs:
  - the master sends a bit on the MOSI line; the slave reads it from that same line
  - the slave sends a bit on the MISO line; the master reads it from that same line
- Not all transmissions require all four of these operations to be *meaningful* but they do happen.
- The number of bits transferred is not fixed but is usually a multiple of 8-bits.

Basic serial data transfer



The registers within the master and slave act like shift registers shifting one bit on every cycle of the SCLK.

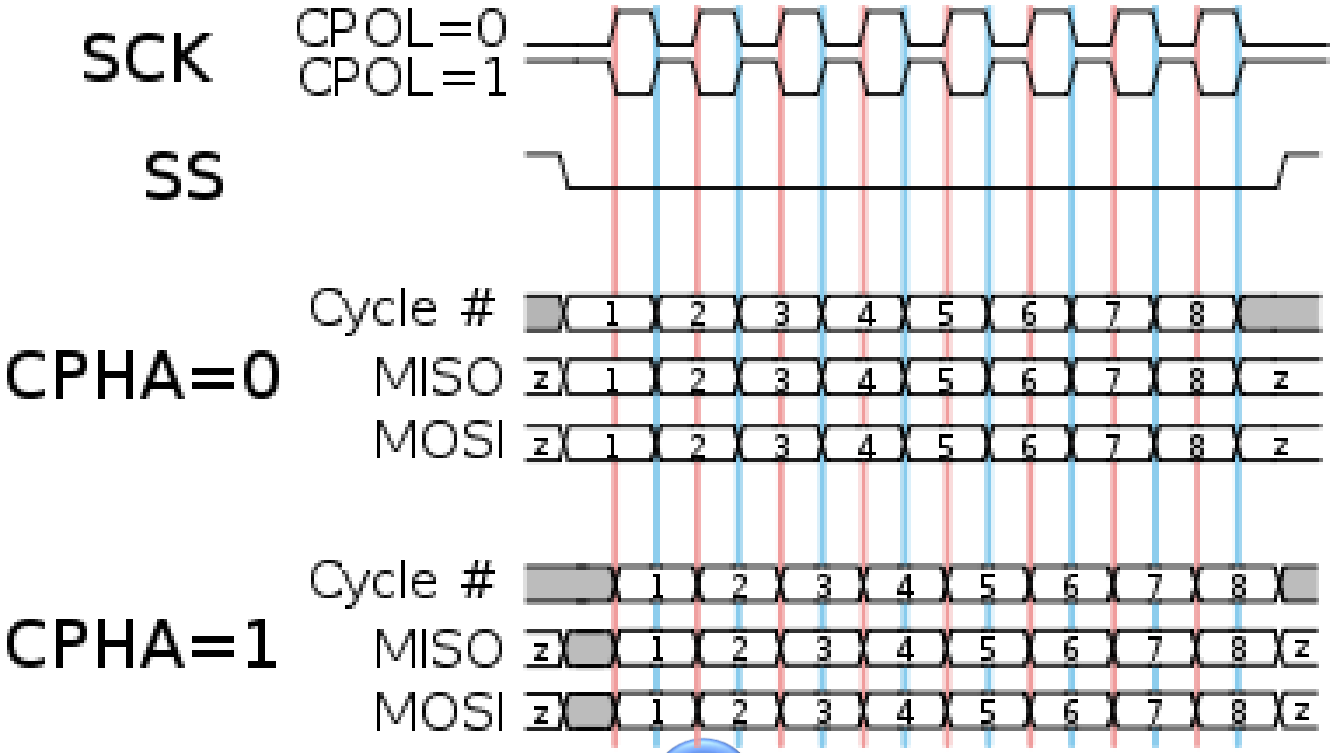
Data transfer details

- Most SPI interfaces have two configuration bits, called clock polarity (CPOL) and clock phase (CPHA).
- CPOL determines whether the shift clock's idle state is low (CPOL=0) or high (CPOL=1).
- CPHA determines on which clock edges data is shifted in and out (for CPHA=0, MOSI data is shifted out on falling edge, MISO data is shifted in on rising edge).
- For two SPI devices to talk to each other, they need to be set to use the same clock polarity and phase settings.

SPI Data Transfer Modes

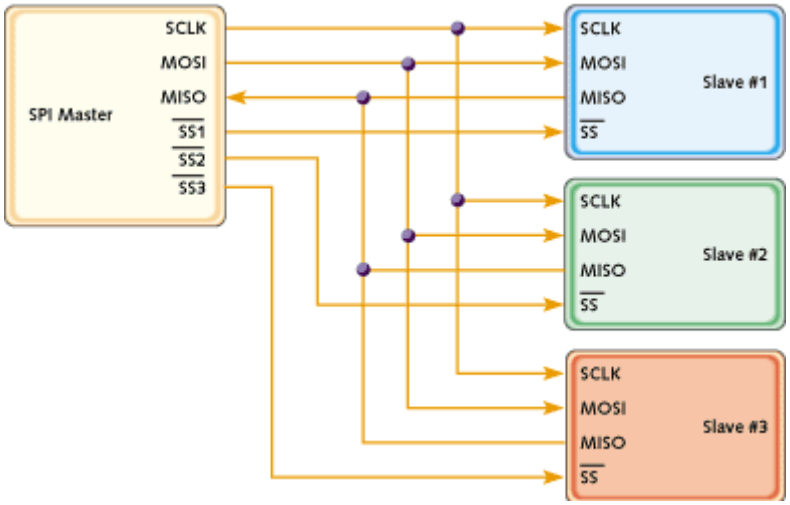
Mode	CPOL	CPHA	Active edge
0	0	0	Rising
1	0	1	Falling
2	1	0	Falling
3	1	1	Rising

- Modes 0 and 3 are the most common.
- With SPI modes 0 and 3, data is always latched in on the rising edge of SCK and always output on the falling edge of SCK.



Multiple Slaves

- If multiple slave devices exist, the master normally generates a separate slave select signal for each slave.



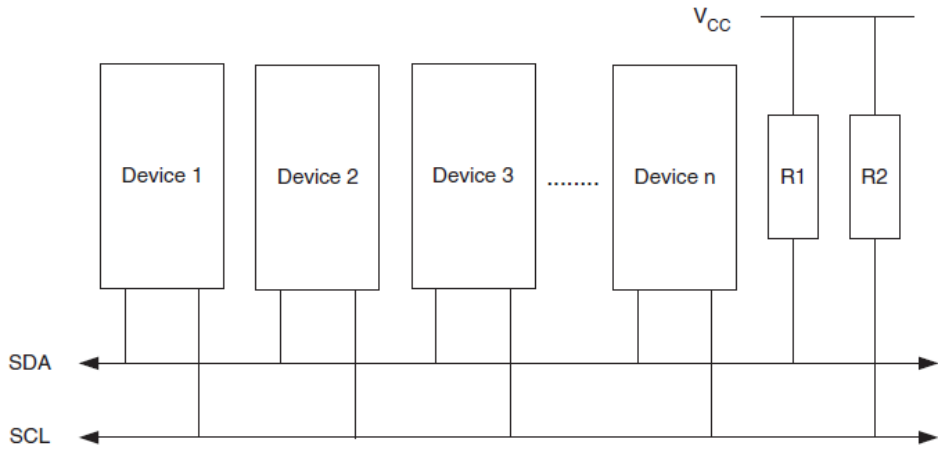
SPI Bus characteristics

- It is up to the master and slave devices to know whether a received byte is meaningful or not.
- No Acknowledgement
- Master doesn't even know if slave is present!
- Slaves can be thought of as IO devices of the master.

Example SPI devices

- 25LC020A - 2K SPI Bus Serial EEPROM
- TC77-5.0 - Thermal Sensor with SPI Interface
- MCP3201 - 2.7V 12-Bit A/D Converter with SPI Serial Interface
- MCP4822 - 12-Bit DAC with Internal VREF and SPI Interface
- MCP41010 - Single/Dual Digital Potentiometer with SPI Interface
- MCP6S92 - Single-Ended, Rail-to-Rail I/O, Low-Gain PGA

I2C or TWI



Introduction

- Serial communication protocol
- Meant for short distances “inside the box”
- Low complexity
- Low cost
- Low speed (a few Mbps at the fastest)

- **To be discussed**: Applications, protocols, tradeoffs, AVR support

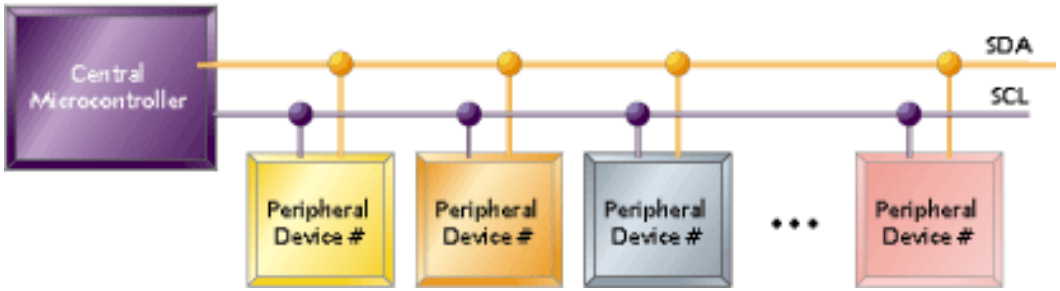
What is I²C?

- Shorthand for an “Inter-integrated circuit” bus
- Developed by Philips Semiconductor for TV sets in the 1980’s
- I²C devices include EEPROMs, thermal sensors, and real-time clocks

- I²C bus has three speeds:
  - Slow (under 100 Kbps)
  - Fast (400 Kbps)
  - High-speed (3.4 Mbps) – I²C v.2.0

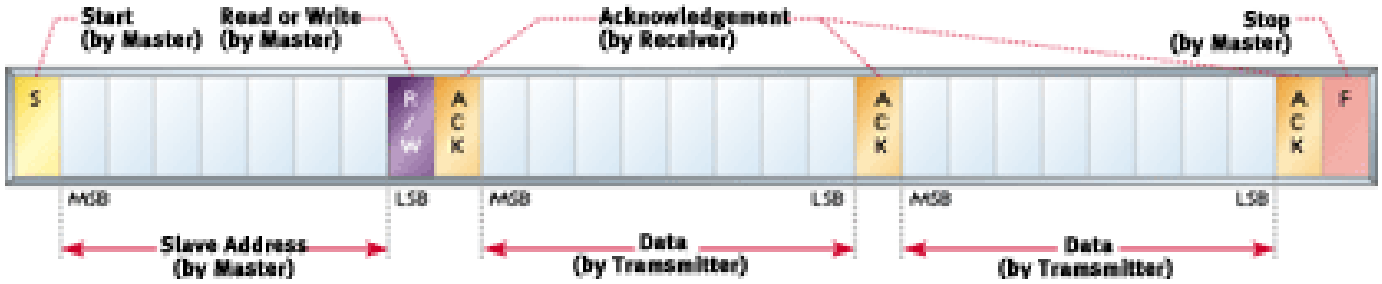
- Limited to about 10 feet for moderate speeds

I²C Bus Configuration



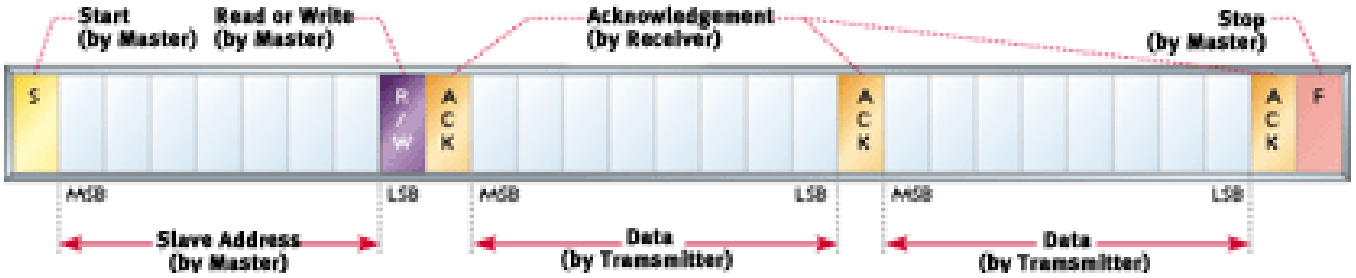
- 2-wire serial bus – Serial data (SDA) and Serial clock (SCL)
- Half-duplex, synchronous, multi-master bus
- No chip select or arbitration logic required
- Lines pulled high via resistors, pulled down via open-drain drivers

I²C Protocol



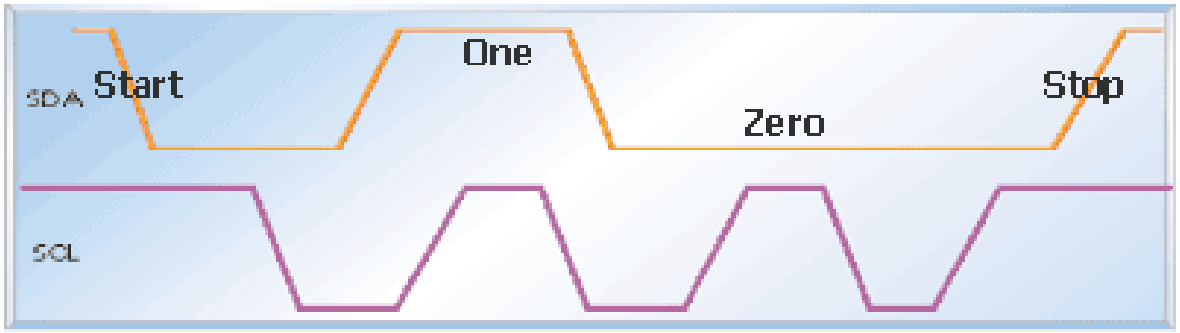
1. Master sends start condition (S) and controls the clock signal
2. Master sends a unique 7-bit slave device address
3. Master sends read/write bit (R/W) :
   0 - slave receive,
   1 - slave transmit
4. Receiver sends acknowledge bit (ACK)
5. Transmitter (slave or master) transmits 1 byte of data

I²C Protocol (cont.)



6. Receiver issues an ACK bit for the byte received
7. Repeat 5 and 6 if more bytes need to be transmitted.
8.a) For write transaction (master transmitting), master issues stop condition (P) after last byte of data.
8.b) For read transaction (master receiving), master does not acknowledge final byte, just issues stop condition (P) to tell the slave the transmission is done

I²C Signals



Sample Signals

- Start – high-to-low transition of the SDA line while SCL line is high
- Stop – low-to-high transition of the SDA line while SCL line is high
- Ack – receiver pulls SDA low while transmitter allows it to float high
- Data – transition takes place while SCL is low, valid while SCL is high

I²C Features

- “Clock stretching” – when the slave (receiver) needs more time to process a bit, it can pull SCL low. The master waits until the slave has released SCL before sending the next bit.
- “General call” broadcast – addresses every device on the bus
- 10-bit extended addressing for new designs. 7-bit addresses all exhausted

AVR Support for I²C

- Atmel calls it “Two-wire Serial Interface” (or TWI)
- Supported by all AVR 8-bit µC except ATTiny and AT90

ATmega323 TWI mode when TWEN in TWCR is set:

- PC0=SCL, PC1=SDA
- TWBR – sets bit rate
- TWCR – controls start, stop, ack generation, indicates M/S, T/R
- TWDR – contains byte transmitted/received
- TWAR – contains slave address
- TWSR – indicates status of TWI Bus (start condition transmitted, ACK received, … 26 total states)

I²C Tradeoffs

Advantages:

- Good for communication with on-board devices that are accessed occasionally.
- Easy to link multiple devices because of addressing scheme
- Cost and complexity do not scale up with the number of devices

Disadvantages:

- The complexity of supporting software components can be higher than that of competing schemes (for example, SPI).

انواع دسترسی به رابط I2C در کدویژن

- برای دسترسی به رابط دو سیمه و استفاده از آن در AVR بوسیله نرم افزار کد ویژن به دو صورت می‌توان عمل کرد:
  - دسترسی به واسط I2C با استفاده از واحد سخت افزاری TWI
  - دسترسی به واسط I2C به صورت نرم افزاری با اضافه کردن فایل سرآیه i2c.h

تفاوت های استفاده از واسط سخت افزاری و نرم افزاری

- در صورت استفاده از واسط سخت افزاری تنها میتوان پایه های SDA و SCL میکرو را استفاده نمود در حالی که در حالت نرم افزاری هر دو پایه دلخواه را می شود استفاده کرد.

- در حالت نرم افزاری، بخشی از پردازنده، درگیر تولید پالس های کلاک و ارسال داده می شود در حالی که در واسط سخت افزاری، یک واحد مجزا درگیر می شود و سرعت برنامه بیشتر است.

- در صورت استفاده از I2C نرم افزاری، با اضافه شدن فایل سرآیه‌ی مربوطه میتوان از توابع آماده موجود استفاده کرده اما در حالت سخت افزاری ظاهرا فایل سرآیه‌ی آماده ای نیست و باید با ثباتها کار کنید!

توابع در حالت نرم افزاری

- void i2c_init(void)
  - این تابع گذرگاه TWI نرم افزاری را روی مقادیرSDA و SCL اولیه فراخوانی و راه اندازی می کند. به همین دلیل باید قبل از فراخوانی توابع دیگر این تابع را صدا زد.
- unsigned char i2c_start(void)
  - با اجرای این تابع یک وضعیت start ایجاد می شود و اگر گذرگاه I2C آزاد باشد ، مقدار ۱ توسط این تابع باز می گردد وگرنه صفر بر می گردد
- void i2c_stop(void)
  - یک وضعیت توقف ایجاد می کند
- unsigned char i2c_read(unsigned char ack)
  - یک بایت می خواند. اگر پارامتر ورودی صفر باشد، تقاضای acknowledge بعد از خواندن، نمی کند وگرنه تقاضا می کند.

توابع I2C نرم افزاری

- unsigned char i2c_write(unsigned char data)
  - داده را روی باس می نویسد. خروجی ۱ است اگر گیرنده ack بدهد وگرنه صفر برمی گرداند.

مثال نوشتن در EEPROM مدل AT24C08A 8k

Two-wire Serial EEPROM

1K (128 x 8)

2K (256 x 8)

4K (512 x 8)

8K (1024 x 8)

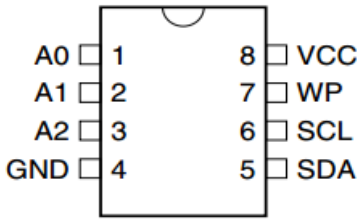
16K (2048 x 8)

AT24C01A(1)
AT24C02(2)
AT24C04
AT24C08A
AT24C16A(3)

Table 1. Pin Configuration

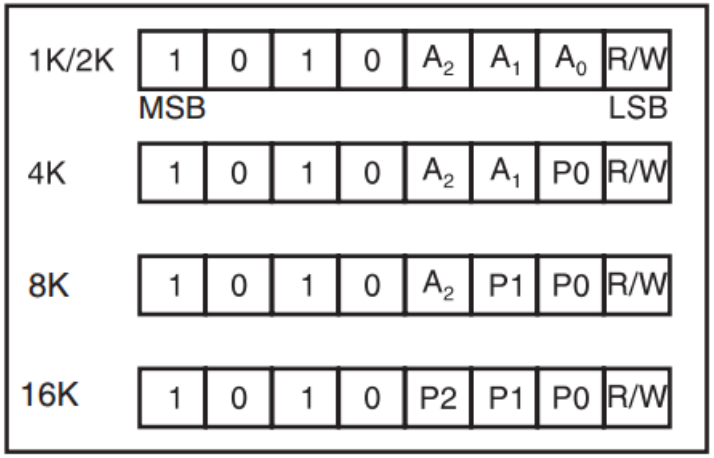
Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect
GND	Ground
VCC	Power Supply

8-lead PDIP



آدرس دهی قطعه

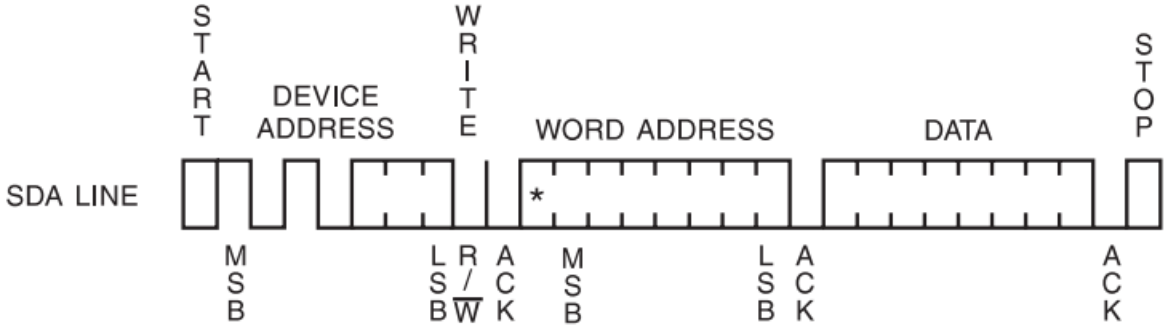
- سه پایه A0,A1,A2 آدرس قطعه را مشخص می کند که شرکت اتمل برای اینکه امکان استفاده از چندین حافظه EEPROM در یک گذرگاه I2C را داشته باشیم، آن را تعبیه کرده است. بنابراین تا ۸ آی سی حافظه را میتوان روی گذرگاه قرار داد.
- LSB بیانگر خواندن یا نوشتن خواهد بود



نوشتن

- برای نوشتن در این آی سی از پروتکل I2C به دو صورت خاص استفاده می شود. اولی برای نوشتن یک بایت در آن و دومی نوشتن یک صفحه که متشکل از چندین بایت پشت سر هم است
- در نوشتن به صورت بایتی ابتدا آدرس قطعه مشخص می شود، سپس آدرس خانه ای از حافظه که میخواهیم در آن بنویسیم مشخص می شود و در نهایت داده ی مورد نظر برای ذخیره ارسال می شود.

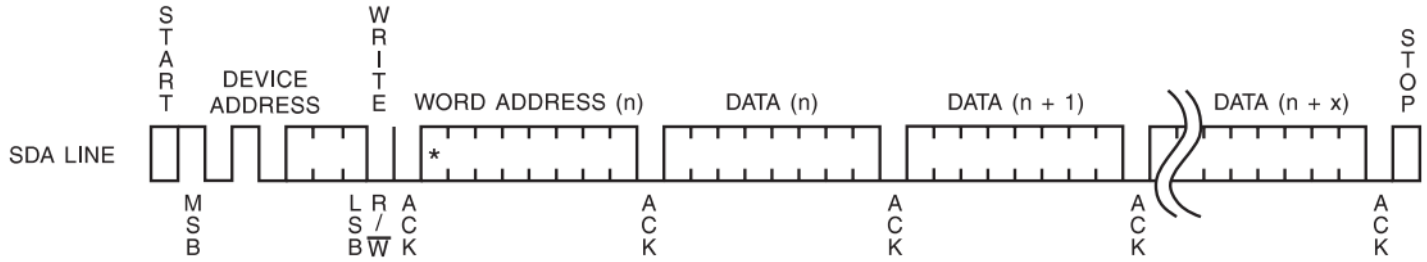
Byte Write



نوشتن بلوکی از بایتها

- نوشتن صفحه ای همانند نوشتن بایتی است با این تفاوت که بعد از ارسال بایت اول، بایت های بعدی پشت سر هم ارسال می‌شوند و سپس سیگنال توقف ارسال خواهد شد.

Figure 9. Page Write



کد برای خواندن و نوشتن در AT24C08A 8k

```
/* include the I2C bus functions
   The I2C bus connections and bit rate must be specified in the
   Project|Configure|C Compiler|Libraries|I2C menu */
#include <i2c.h>


/* function declaration for delay_ms */
#include <delay.h>


#define I2C_7BIT_DEVICE_ADDRESS 0x50 //1010 000*
//کم ارزشترین بیت، بیانگر خواندن یا نوشتن است
#define EEPROM_BUS_ADDRESS (I2C_7BIT_DEVICE_ADDRESS << 1)
```

کد خواندن

```
/*read a byte from the EEPROM */
unsigned char eeprom_read(unsigned int address)
{
   unsigned char data;
   i2c_start();
   i2c_write(EEPROM_BUS_ADDRESS | 0); //0 means write →
   /*send MSB of address */
   i2c_write(address >> 8);
   /*send LSB of address */
   i2c_write((unsigned char)address);
   i2c_start();
   i2c_write(EEPROM_BUS_ADDRESS | 1); //1 means read →
   data = i2c_read(0);
   i2c_stop();
   return data;
}
```

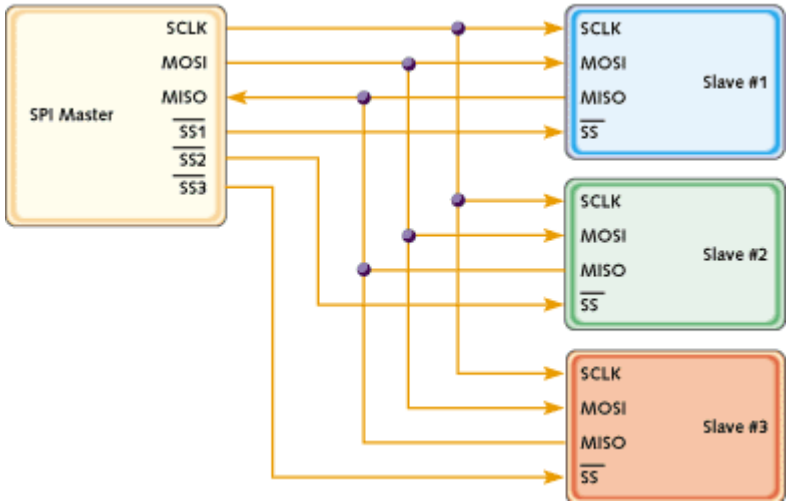
کد برای نوشتن بایت

```
/* write a byte to the EEPROM */
void eeprom_write(unsigned int address, unsigned char data)
{
    i2c_start();
    i2c_write(EEPROM_BUS_ADDRESS | 0);
    /*send MSB of address */
    i2c_write(address >> 8);
    /*send LSB of address */
    i2c_write((unsigned char)address);
    i2c_write(data);
    i2c_stop();
    /* 10ms delay to complete the write operation */
    delay_ms(10);
}
```

برنامه تست

```
void main(void)
{
    unsigned char i;
    /* initialize the I2C bus */
    i2c_init();
    /* write the byte 55h at address AAh */
    eeprom_write(0xaa, 0x55);
    /* read the byte from address AAh */
    i = eeprom_read(0xaa);
    while (1); /* loop forever */
}
```

SPI vs. I²C



- For point-to-point, SPI is simple and efficient
  - Less overhead than I²C due to lack of addressing, plus SPI is full duplex.
- For multiple slaves, each slave needs separate slave select signal
  - More effort and more hardware than I²C

Summary

- I^2C and SPI provide good support for communication with slow peripheral devices that are accessed intermittently, mainly EEPROMs and real-time clocks
- I^2C easily accommodates multiple devices on a single bus.
- SPI is faster, but gets complicated when there is more than one slave involved.