SPI Serial Port
(in AVR Microcontrollers)
Contents

• Serial communication with SPI
• Serial communication programming in C
• Reference: Chapter 17 of the Mazidi’s book
Serial ports in AVR
Serial ports in AVR

- **USART**: and old and widely used standard (pins 14 and 15 of ATmega32)
- **SPI**: very high-speed (~ two orders of magnitude faster than USART) synchronous serial port
  - For example for high-speed data transfer to EPROM
  - Pins 5 to 8 of Atmega32
- **I2C (Inter IC)**
  - Connection using a shared bus connecting up to 128 devices
  - Pins 22 and 23 of ATmega32
SPI

• SPI : serial peripheral interface
• Originally started by Motorola (now Freescale),
  – Now, a widely used standard adapted by many semiconductor chip companies including Atmel
• Very faster than USART
SPI wiring

- Uses only 2 pins for data transfer
  - SDI (Din)
  - SDO (Dout)
- Requires a SCLK (shift clock) pin: to synchronize the data transfer
- Requires a CE (chip enable) pin: used to initiate and terminate the data transfer
- Called MOSI, MISO, SCK, and SS in AVR
3-wire SPI

- Alternative: a 3-wire interface bus
  - SCLK and CE, and only a single pin for data transfer
- The SPI 4-wire bus can become a 3-wire interface when the SDI and SDO data pins are tied together
SPI architecture

- Two 8-bit shift registers, one in the master and the other in the slave side
- A clock generator in the master side
  - generates the clock for the shift registers
• Connections:
• Serial-out pin of the master shift register to the serial-in pin of the slave shift register by MOSI (Master Out Slave In)
• Serial-in pin of the master shift register to the serial-out pin of the slave shift register by MISO (Master In Slave Out)
• After 8 clock pulses, the contents of the two shift registers are interchanged
SS pin in SPI

• Slave Select (SS) pin of the SPI bus is used to select a slave to initiate and terminate the data transfer
• Makes sense in devices connected to several slaves
• In master mode:
  – SS=0: enable slave
  – SS=1: disable slave
  – Active low!
SPI timing

- Synchronization by clock
  - CPOL (clock polarity)
  - CPHA (clock phase)
- At CPOL = 0 the base value of the clock is zero, while at CPOL = 1 the base value of the clock is one
SPI timing

- Synchronization by clock
  - CPOL (clock polarity)
  - CPHA (clock phase)

- CPHA = 0 means sample (read) on the first clock edge, while CPHA = 1 means sample on the second clock
SPI registers in AVR

- SPSR (SPI Status Register)
- SPCR (SPI Control Register)
- SPDR (SPI Data Register)
SPSR (SPI Status Register)

<table>
<thead>
<tr>
<th>SPIF</th>
<th>WCOL</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>SPI2X</th>
</tr>
</thead>
</table>

**Bit 7 – SPIF (SPI Interrupt Flag)**

In master mode, this bit is set in two situations: when a serial transfer is completed, or when SS pin is an input and is driven low by an external device. Setting the SPIF flag to one will cause an interrupt if SPIE in SPCR is set and global interrupts are enabled.

**Bit 6 – WCOL (Write COLLision Flag)**

The WCOL bit is set if you write on SPDR during a data transfer.

**Bit 0 – SPI2X (Double SPI Speed)**

When the SPI is in master mode, setting this bit to one doubles the SPI speed.
SPCR (SPI Control Register)

Bit 7 – SPIE: SPI Interrupt Enable
Setting this bit to one enables the SPI interrupt.

Bit 6 – SPE: SPI Enable
Setting this bit to one enables the SPI.

Bit 5 – DORD: Data Order
This bit lets you choose to either transmit MSB and then LSB or vice versa. The LSB is transmitted first if DORD is one; otherwise, the MSB is transmitted first.

Bit 4 – MSTR: Master/Slave Select
If you want to work in master mode then set this bit to one; otherwise, slave mode is selected.
SPCR (SPI Control Register)

<table>
<thead>
<tr>
<th>SPIE</th>
<th>SPE</th>
<th>DORD</th>
<th>MSTR</th>
<th>CPOL</th>
<th>CPHA</th>
<th>SPR1</th>
<th>SPR0</th>
</tr>
</thead>
</table>

**Bit 3 – CPOL: Clock Polarity**
This bit sets the base value of the clock when it is idle. At CPOL = 0 the base value of the clock is zero while at CPOL = 1 the base value of the clock is one.

**Bit 2 – CPHA: Clock Phase**
CPHA = 0 means sample on the leading (first) clock edge, while CPHA = 1 means sample on the trailing (second) clock.

**Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**
These two bits control the SCK rate of the device in master mode.
SCK frequency

<table>
<thead>
<tr>
<th>SPI2X</th>
<th>SPR 1</th>
<th>SPR 0</th>
<th>SCK frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>F_{osc}/4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>F_{osc}/16</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>F_{osc}/64</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>F_{osc}/128</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F_{osc}/2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>F_{osc}/8</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F_{osc}/32</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>F_{osc}/64</td>
</tr>
</tbody>
</table>
SPDR (SPI Data Register)

- When a new data is written to this register, the SCK is generated for 8 cycles and then stops.
SPI Operation

• SPI can be used for simple data transfer
  – Just write data to SPDR
  – 8 clocks are generated automatically to transfer data
  – Slave can never start sending data

• Standard mechanisms for master to read and write data from a memory address
  – First send address (and also tell if the transfer is read or write), then send/receive data
Steps for writing data to an SPI device

- **Single byte:**
  - Make CE = 0 to begin writing
  - The 8-bit address is shifted in, one bit at a time, with each edge of SCLK
    - A7 = 1 for the write operation
  - After all 8 bits of the address are sent in, the SPI device expects to receive the data belonging to that address location immediately
  - The 8-bit data is shifted in one bit at a time, with each edge of the SCLK
  - Make CE = 1 to indicate the end of the write cycle
Multi-byte burst write

- Burst mode: storing to consecutive locations
- Provide the address of the first location, followed by the data for that location
- While CE = 0, consecutive bytes are written to consecutive memory locations
  - The SPI device internally increments the address location as long as CE is LOW
Steps for reading data from an SPI device

• Single byte:
  – Make CE = 0 to begin reading
  – The 8-bit address is shifted in one bit at a time, with each edge of SCLK
    » A7 = 0 for the read operation
  – After all 8 bits of the address are sent in, the SPI device sends out data belonging to that location
  – The 8-bit data is shifted out one bit at a time, with each edge of the SCLK
  – Make CE = 1 to indicate the end of the read cycle
Multi-byte burst read

- Much like the burst write
- The addresses are incremented by SPI device as long as CE is enable
SPI programming in C

• Write an AVR program to initialize the SPI for master, with CLCK frequency = Fosc/16, and then transmit 'G' via SPI repeatedly. The received data should be displayed on Port A.
SPI programming in C

- Write an AVR program to initialize the SPI for master, with CLCK frequency = Fosc/16, and then transmit 'G' via SPI repeatedly. The received data should be displayed on Port A.

```c
#include <avr/io.h>
#define MOSI 5
#define SCK 7
int main (void)
{
    DDRB = (1<<MOSI)|(1<<SCK); //MOSI and SCK are output
    DDRA = 0xFF; //Port A is output
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); //enable SPI as master
    while(1){
        SPDR = 'G'; //start transmission
        while(!(SPSR & (1<<SPIF)));
        PORTA = SPDR; //move received data to Port A
    }
    return 0;
}
```
SPI programming in C

- Write an AVR program to initialize the SPI for slave, with CLCK frequency = Fosc/4, and then show the received data on Port D

```c
DDRB=0001,0000 // MISO as output other pins input
DDRD=0xFF; //port D as output
SPCR= 1100,0000 // enable interrupt, enable SPI, clk=f/4
SPSR=0;
.asm ("sei")

interrupt [SPI] void spi_isr()

while (1)
{
    PORTD=SPDR;
}
```

![SPI register settings](image)
SPI in PCs

• Basic Input/Output System (BIOS):
  – The first software run by a PC when powered on
  – Initialize and test the system hardware components, and to load a bootloader or an operating system

• BIOS is stored on programmable ROM
  – The ROM is connected to Southbridge of motherboard by SPI
LPC interface

• Before SPI, LPC used to be adopted to connect to BIOS
• LPC: Low-pin count bus
• Before 2006, BIOS used to connect to Southbridge through LPC interface
  – Now, some laptops use LPC for bios
• A 7-wire bus
SPI in a Modern Southbridge

- ICH8 Southbridge by Intel
- Two chip select signals to connect to two ROMs
Serial Flash Command Set

- SPI just provides a mechanism to exchange data
- Every protocol can be made on top of SPI
- Example: A SPI-based flash device must support a set of commands in order to be interoperable with the SPI port of ICH8

<table>
<thead>
<tr>
<th>Commands</th>
<th>OPCODE</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Status</td>
<td>01h</td>
<td>If command is supported, 01h must be the opcode</td>
</tr>
<tr>
<td>Program Data</td>
<td>02h</td>
<td>Write Data / Program Data</td>
</tr>
<tr>
<td>Read Data</td>
<td>03h</td>
<td></td>
</tr>
<tr>
<td>Write Disable</td>
<td>04h</td>
<td></td>
</tr>
<tr>
<td>Read Status</td>
<td>05h</td>
<td></td>
</tr>
<tr>
<td>Write Enable</td>
<td>06h</td>
<td>If command is supported, 06h must be the opcode</td>
</tr>
</tbody>
</table>