Module 9 - SPI, I2C, and Sensor Interfacing

1. Serial Peripheral Interface (SPI)

1.1 Overview

SPI is a synchronous serial communication protocol commonly used for fast peripheral communication in embedded systems. It typically uses four lines:

- SCK: Serial Clock
- MOSI: Master Out Slave In
- MISO: Master In Slave Out
- SS/CS: Slave Select / Chip Select

SPI is full-duplex, meaning data can be transmitted and received at the same time.

1.2 Communication Flow

1. The master pulls SS low to select a slave.
2. The master generates clock pulses on SCK.
3. Data shifts out on MOSI and shifts in on MISO.
4. The transfer ends when the expected bits/bytes are completed.
5. The master releases SS high.

1.3 SPI Clock Modes

- Mode 0: CPOL = 0, CPHA = 0
- Mode 1: CPOL = 0, CPHA = 1
- Mode 2: CPOL = 1, CPHA = 0
- Mode 3: CPOL = 1, CPHA = 1

1.4 SPI Registers (Detailed Bits)

SPDR - SPI Data Register

Function:

- Stores data to transmit and received data.

Bit	Name	Description

7	SPD7	Data bit 7
6	SPD6	Data bit 6
5	SPD5	Data bit 5
4	SPD4	Data bit 4
3	SPD3	Data bit 3
2	SPD2	Data bit 2
1	SPD1	Data bit 1
0	SPD0	Data bit 0

SPSR - SPI Status Register

Function:

- Indicates transfer completion and collision state.

Bit	Name	Description
7	SPIF	SPI interrupt/transfer complete flag
6	WCOL	Write collision flag
5	Reserved	Reserved
4	Reserved	Reserved
3	Reserved	Reserved
2	Reserved	Reserved
1	Reserved	Reserved
0	SPI2X	Double SPI speed (master mode)

SPCR - SPI Control Register

Function:

- Controls SPI enable state, mode, clock settings, and interrupts.

Bit	Name	Description
7	SPIE	Enable SPI interrupt
6	SPE	Enable SPI peripheral
5	DORD	Data order (1=LSB first, 0=MSB first)
4	MSTR	Master select (1=master, 0=slave)
3	CPOL	Clock polarity

Bit	Name	Description
2	CPHA	Clock phase
1	SPR1	Clock rate select bit 1
0	SPR0	Clock rate select bit 0

Clock-rate note:

- SPI clock rate in master mode is determined by SPR1, SPR0, and SPI2X.

1.5 SPI Assembly Examples

SPI Master

```
;------------------------
; Assembly Code SPI Master
;------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;------------------------
.global main
;================================================================
main:
.equ  SCK, 5
.equ  MOSI, 3
.equ  SS, 2
;----------------------------------------------------------------
      LDI   R17, (1<<MOSI)|(1<<SCK)|(1<<SS)
      OUT   DDRB, R17       ;set MOSI, SCK, SS as o/p
      ;--------------------------------------------------------
      LDI   R17, (1<<SPE)|(1<<MSTR)|(1<<SPR0)
      OUT   SPCR, R17       ;enable SPI as master, fsck=fosc/16
      ;--------------------------------------------------------
      LDI   R17, 0xAA       ;byte to be transmitted
      ;--------------------------------------------------------
again:CBI   PORTB, SS       ;enable slave device
      OUT   SPDR, R17       ;transmit byte to slave
      ;--------------------------------------------------------
loop: IN    R18, SPSR
      SBRS  R18, SPIF       ;wait for byte transmission
      RJMP  loop            ;to complete
```

```
      ;-----------------------------------------------------
      SBI   PORTB, SS       ;disable slave device
      ;-----------------------------------------------------
      RCALL my_delay        ;delay
      COM   R17             ;1's compliment of byte
      RJMP  again           ;repeat transmission
;=============================================================
my_delay:                   ;delay in ms
    LDI   R20, 255
l6: LDI   R21, 255
l7: LDI   R22, 40
l8: DEC   R22
    BRNE  l8
    DEC   R21
    BRNE  l7
    DEC   R20
    BRNE  l6
    RET
```

SPI Slave

```
;------------------------
; Assembly Code SPI Slave
;------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;------------------------
.global main
;=============================================================
main:
;-------------------------------------------------------------
      LDI   R17, 0xFF
      OUT   DDRD, R17       ;set PORTD for o/p
      ;-----------------------------------------------------
      LDI   R17, (1<<SPE)
      OUT   SPCR, R17       ;enable SPI as slave
      ;-----------------------------------------------------
agn:  IN    R18, SPSR
      SBRS  R18, SPIF       ;wait for byte reception
      RJMP  agn
```

```
;-----------------------------------------------------
IN    R18, SPDR       ;i/p byte from data register
OUT   PORTD, R18      ;and o/p to PORTD
;-----------------------------------------------------
RJMP  agn             ;repeat reception
```

2. Inter-Integrated Circuit (I2C)

2.1 Overview

I2C is a synchronous two-wire serial bus:

- SDA: Serial Data
- SCL: Serial Clock

I2C supports multiple devices on the same bus using slave addressing and ACK/NACK handshaking.

2.2 I2C Transaction Flow

1. Master sends START condition.
2. Master sends Slave Address + R/W bit.
3. Slave replies with ACK.
4. Data bytes are exchanged with ACK/NACK after each byte.
5. Master sends STOP condition.

2.3 I2C Speed Modes

- Standard mode: up to 100 kHz
- Fast mode: up to 400 kHz
- Fast mode plus: up to 1 MHz
- High-speed mode: up to 3.4 MHz
- Ultra-fast mode (unidirectional): up to 5 MHz

2.4 I2C Registers (Detailed Bits)

TWSR - TWI Status Register

Function:

- Contains status code bits (TWS7:TWS3) and prescaler bits.

Bit	Name	Description
7	TWS7	Status code bit 7
6	TWS6	Status code bit 6
5	TWS5	Status code bit 5

Bit	Name	Description
4	TWS4	Status code bit 4
3	TWS3	Status code bit 3
2	Reserved	Reserved/unused
1	TWPS1	Prescaler select bit 1
0	TWPS0	Prescaler select bit 0

Prescaler mapping:

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWBR - TWI Bit Rate Register

Function:

- Sets the SCL clock generator division factor in master mode.

Bit	Name	Description
7	TWBR7	Bit-rate divider bit 7
6	TWBR6	Bit-rate divider bit 6
5	TWBR5	Bit-rate divider bit 5
4	TWBR4	Bit-rate divider bit 4
3	TWBR3	Bit-rate divider bit 3
2	TWBR2	Bit-rate divider bit 2
1	TWBR1	Bit-rate divider bit 1
0	TWBR0	Bit-rate divider bit 0

Clock formula:

SCL = F_CPU / (16 + 2 x TWBR x PrescalerValue)

TWCR - TWI Control Register

Function:

- Controls start/stop generation, ACK, interrupt, and enable state.

Bit	Name	Description
7	TWINT	TWI interrupt flag (set by hardware when operation completes)
6	TWEA	TWI enable acknowledge
5	TWSTA	TWI start condition request
4	TWSTO	TWI stop condition request
3	TWWC	TWI write collision flag
2	TWEN	TWI enable
1	Reserved	Reserved
0	TWIE	TWI interrupt enable

TWDR - TWI Data Register

Function:

- Holds the current transmitted/received byte.

Bit	Name	Description
7	TWD7	Data bit 7
6	TWD6	Data bit 6
5	TWD5	Data bit 5
4	TWD4	Data bit 4
3	TWD3	Data bit 3
2	TWD2	Data bit 2
1	TWD1	Data bit 1
0	TWD0	Data bit 0

TWAR - TWI Address Register

Function:

- Holds own slave address and general call control.

Bit	Name	Description
7	TWA6	Slave address bit 6
6	TWA5	Slave address bit 5

Bit	Name	Description
5	TWA4	Slave address bit 4
4	TWA3	Slave address bit 3
3	TWA2	Slave address bit 2
2	TWA1	Slave address bit 1
1	TWA0	Slave address bit 0
0	TWGCE	General call recognition enable

Common TWI Status Codes

Status Code	Meaning
0x08	START condition transmitted
0x10	Repeated START transmitted
0x18	SLA+W transmitted, ACK received
0x20	SLA+W transmitted, NACK received
0x28	Data transmitted, ACK received
0x30	Data transmitted, NACK received
0x38	Arbitration lost
0x40	SLA+R transmitted, ACK received
0x48	SLA+R transmitted, NACK received
0x50	Data received, ACK returned
0x58	Data received, NACK returned

2.5 I2C Assembly Examples

I2C Master Transmit

```
;-------------------------
; Assembly Code - Master Tx
;-------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----------------------
.global main
;============================================================
main:
```

```
    CBI   DDRC, 3         ;pin PC3 is i/p
    ;-------------------------------------------------------
    RCALL I2C_init        ;initialize TWI module
    ;-------------------------------------------------------
l1: SBIS  PINC, 3
    RJMP  l1              ;wait for "transmit" button press
    ;-------------------------------------------------------
    RCALL I2C_start       ;transmit START condition
    LDI   R27, 0b10010000 ;SLA(1001000) + W(0)
    RCALL I2C_write       ;write slave address SLA+W
    LDI   R27, 0b11110101 ;data byte to be transmitted
    RCALL I2C_write       ;write data byte
    RCALL I2C_stop        ;transmit STOP condition
    ;-------------------------------------------------------
    RJMP  l1              ;go back for another transmit
;=============================================================
I2C_init:
    LDI   R21, 0
    STS   TWSR, R21       ;prescaler = 0
    LDI   R21, 12         ;division factor = 12
    STS   TWBR, R21       ;SCK freq = 400kHz
    LDI   R21, (1<<TWEN)
    STS   TWCR, R21       ;enable TWI
    RET
;=============================================================
I2C_start:
    LDI   R21, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
    STS   TWCR, R21       ;transmit START condition
    ;-------------------------------------------------------
wt1:LDS   R21, TWCR
    SBRS  R21, TWINT      ;TWI interrupt = 1?
    RJMP  wt1             ;no, wait for end of transmission
    ;-------------------------------------------------------
    RET
;=============================================================
I2C_write:
    STS   TWDR, R27       ;copy SLA+W into data register
    LDI   R21, (1<<TWINT)|(1<<TWEN)
    STS   TWCR, R21       ;transmit SLA+W
    ;-------------------------------------------------------
```

```
wt2:LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt2             ;wait for end of transmission
    ;-------------------------------------------------------
    RET
;=============================================================
I2C_stop:
    LDI   R21, (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)
    STS   TWCR, R21       ;transmit STOP condition
    RET
```

I2C Slave Receive

```
;-------------------------
; Assembly Code - Slave Rx
;-------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;------------------------
.global main
;=============================================================
main:
    LDI   R21, 0xFF
    OUT   DDRD, R21         ;port D is o/p
    CBI   DDRC, 3           ;pin PC3 is i/p
;-------------------------------------------------------------
agn:RCALL I2C_init          ;initialize TWI module
    RCALL I2C_listen        ;listen to bus to be addressed
    RCALL I2C_read          ;read data byte
    OUT   PORTD, R27        ;and o/p to port D
;-------------------------------------------------------------
l1: SBIS  PINC, 3
    RJMP  l1                ;wait for "listen" button press
;-------------------------------------------------------------
    LDI   R26, 0
    OUT   PORTD, R26        ;clear port D
    RJMP  agn               ;& go back & listen to bus
;=============================================================
I2C_init:
    LDI   R21, 0b10010000
```

```
    STS   TWAR, R21         ;store slave address 0b10010000
    LDI   R21, (1<<TWEN)
    STS   TWCR, R21         ;enable TWI
    LDI   R21, (1<<TWINT)|(1<<TWEN)|(1<<TWEA)
    STS   TWCR, R21         ;enable TWI & ACK
    RET
;=============================================================
I2C_listen:
    LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  I2C_listen        ;wait for slave to be addressed
    RET
;=============================================================
I2C_read:
    LDI   R21, (1<<TWINT)|(1<<TWEA)|(1<<TWEN)
    STS   TWCR, R21         ;enable TWI & ACK
    ;-----------------------------------------------------------
wt: LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt                ;wait for data byte to be read
    ;-----------------------------------------------------------
    LDS   R27, TWDR         ;store received byte
    RET
```

3. DHT11 Sensor Interfacing

3.1 Sensor Fundamentals

A sensor captures physical phenomena from the real world and converts them into electrical signals (analog or digital) for computation.

For DHT11 specifically:

- It measures temperature and humidity.
- It is commonly used in educational and basic room-monitoring applications.
- Typical module-noted range:
- Temperature: 0 to 50 deg C
- Humidity: 20% to 90% RH

3.2 DHT11 Variants and Wiring

DHT11 can be found as:

- bare sensor package,
- breakout module.

Breakout modules are typically easier to wire and often include support components.

3.3 DHT11 Protocol and Timing

DHT11 uses a single-wire, pulse-width-based digital protocol. Communication sequence:

1. MCU sends start signal (low pulse around 18-20 ms, then high).
2. DHT11 sends response pulse.
3. DHT11 transmits 40-bit serial data.
4. Bit value is represented by pulse width:

- short high pulse = 0
- long high pulse = 1

3.4 40-bit Data Format

Byte	Data Field
1	Humidity integer part

Byte	Data Field
2	Humidity decimal part
3	Temperature integer part
4	Temperature decimal part
5	Checksum

Checksum rule:

Checksum = (Byte1 + Byte2 + Byte3 + Byte4) & 0xFF

3.5 Why Accurate Delay Matters

DHT11 decoding depends on timing precision. If delays are too short or too long:

- start handshake may fail,
- bit sampling can shift,
- decoded bytes can become invalid,
- checksum mismatch may occur.

3.6 DHT11 Assembly Example

```
;------------------------
; Assembly Code
;------------------------
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;------------------------
.global main
;====================================================================
main:
;------------
    LDI   R17, 0xFF
    OUT   DDRC, R17     ;set port C for o/p
    OUT   DDRD, R17     ;set port D for o/p
;--------------------------------------------------------------------
agn:RCALL delay_2s      ;wait 2s for DHT11 to get ready
;--------------------------------------------------------------------
;send start signal
;------------
    SBI   DDRB, 1       ;pin PB0 as o/p
```

```
    CBI   PORTB, 1      ;first, send low pulse
    RCALL delay_20ms    ;for 20ms
    SBI   PORTB, 1      ;then send high pulse
;-----------------------------------------------------------------
;wait for response signal
;---------------
    CBI   DDRB, 1       ;pin PB0 as i/p
w1: SBIC  PINB, 1
    RJMP  w1            ;wait for DHT11 low pulse
w2: SBIS  PINB, 1
    RJMP  w2            ;wait for DHT11 high pulse
w3: SBIC  PINB, 1
    RJMP  w3            ;wait for DHT11 low pulse
;-----------------------------------------------------------------
    RCALL DHT11_reading ;read humidity (1st byte of 40-bit data)
    MOV   R19, R18
    RCALL DHT11_reading
    RCALL DHT11_reading ;read temp (3rd byte of 40-bit data)
;-----------------------------------------------------------------
    OUT   PORTD, R19    ;o/p temp byte to port C
    OUT   PORTC, R18    ;o/p humidity byte to port D
    RJMP  agn           ;go back & get another sensor reading
;=================================================================
DHT11_reading:
    LDI   R17, 8        ;set counter for receiving 8 bits
    CLR   R18           ;clear data register
    ;-------------------------------------------------------
w4: SBIS  PINB, 1
    RJMP  w4            ;detect data bit (high pulse)
    RCALL delay_timer0  ;wait 50us then check bit value
    ;-------------------------------------------------------
    SBIS  PINB, 1       ;if bit=1, skip next instruction
    RJMP  skp           ;else bit=0
    SEC                 ;C=1
    ROL   R18           ;shift in 1
    RJMP  w5
skp:LSL   R18           ;shift in 0
    ;-------------------------------------------------------
w5: SBIC  PINB, 1
    RJMP  w5            ;wait for low pulse
```

```
;-----------------------------------------------------
DEC   R17
BRNE  w4
RET
```

4. SPI vs I2C Comparison

Aspect	SPI	I2C
Signal lines	SCK, MOSI, MISO, SS	SDA, SCL
Duplex mode	Full-duplex	Half-duplex-style exchange
Addressing	No built-in addressing	Built-in slave addressing
Typical speed	Generally faster	Generally slower
Wiring complexity	More wires, simple protocol	Fewer wires, richer bus protocol