Module 4:

COMMUNICATION PERIPHERALS & PROTOCOLS

1. Communication Interfaces in Embedded Systems

Embedded systems often communicate with external devices, such as sensors, displays, or other microcontrollers, using standard communication protocols. The **MSP430** microcontroller supports a range of communication interfaces for connecting with external devices.

1.1 USART (Universal Synchronous Asynchronous Receiver/Transmitter)

The **USART** module is used for serial communication and can operate in both **synchronous** and **asynchronous** modes. It is primarily used to transmit and receive data bit by bit over a single wire or pair of wires.

- **Asynchronous Mode**: Data is transmitted without a clock, typically using **start** and **stop bits** to frame the data.
- **Synchronous Mode**: Data is transmitted with a clock signal, which ensures that both sender and receiver are synchronized.

Key Features:

- Full-duplex communication (simultaneous send and receive).
- Typically used in UART (Universal Asynchronous Receiver/Transmitter) and SPI (Serial Peripheral Interface).

Example (MSP430 USART - Asynchronous Mode):

```
#include <msp430.h>

void USART_init() {
    P1SEL |= BIT1 + BIT2; // Configure P1.1 as RX and P1.2 as TX
    UCA0CTL1 |= UCSWRST; // Put USART in reset
    UCA0CTL1 = UCSSEL_2; // Use SMCLK
    UCA0BR0 = 104; // Set baud rate to 9600
    UCA0BR1 = 0;
    UCA0MCTL = UCBRS_1; // Set modulation for 9600 baud rate
    UCA0CTL1 &= ~UCSWRST; // Release USART from reset
}

void USART_transmit(char data) {
    while (!(IFG2 & UCA0TXIFG)); // Wait for TX buffer to be ready
    UCA0TXBUF = data; // Send data
}

int main() {
    USART_init(); // Initialize USART
    while (1) {
        USART_transmit('A'); // Transmit character 'A'
        __delay_cycles(1000000); // Delay
    }
    return 0;
}
```

1.2 USCI (Universal Serial Communication Interface)

The **USCI** module is more advanced and provides support for **UART**, **SPI**, and **I2C** communication. It can operate in both **synchronous** and **asynchronous** modes and is available in newer MSP430 devices.

- **Supports multiple protocols**: USART (UART), I2C, and SPI.
- **Flexible configuration** for different baud rates, data formats, and clock settings.

Example (MSP430 USCI - SPI Mode):

```
#include <msp430.h>

void USCI_SPI_init() {
    P1SEL |= BIT5 + BIT6 + BIT7; // Assign pins for SCLK, MOSI, MISO
    UCB0CTL1 = UCSWRST; // Put USCI in reset
    UCB0CTL0 = UCCKPL | UCMSB | UCMSTR | UCSYNC; // SPI mode, MSB first, synchronous
    UCB0CTL1 = UCSSEL_2; // Use SMCLK as clock source
    UCB0BR0 = 2; // Set baud rate (prescaler)
    UCB0BR1 = 0;
    UCB0CTL1 &= ~UCSWRST; // Release USCI from reset
}
```

```
void SPI_transmit(char data) {
    while (!(IFG2 & UCB0TXIFG)); // Wait for TX buffer to be ready
    UCB0TXBUF = data; // Send data
}

int main() {
    USCI_SPI_init(); // Initialize SPI
    while (1) {
        SPI_transmit('A'); // Transmit data
        __delay_cycles(1000000); // Delay
    }
    return 0;
}
```

1.3 USI (Universal Serial Interface)

The **USI** module is a simpler interface compared to **USCI** and is available on older MSP430 models. It supports basic **I2C** and **SPI** communication. USI is less flexible than USCI, as it only supports a subset of the serial communication protocols.

- Primarily used for **I2C** and **SPI**.
- Typically suitable for simpler applications requiring serial communication.

2. Communication Protocols in Embedded Systems

2.1 SPI (Serial Peripheral Interface)

SPI is a high-speed synchronous serial communication protocol used for connecting a microcontroller to peripherals like sensors, memory devices, and displays. It uses four lines:

- **MOSI (Master Out Slave In)**: Data from master to slave.
- **MISO (Master In Slave Out)**: Data from slave to master.
- **SCLK (Serial Clock)**: Clock signal generated by the master.
- **SS (Slave Select)**: A signal indicating which slave device the master is communicating with.

Key Features:

- Full-duplex communication.
- Supports multiple slaves (using different chip select lines).

- High-speed data transfer.

Example: **SPI Master Sending Data** (MSP430 with USCI)

```
// Already shown in the previous code snippet for USCI_SPI_init().
```

2.2 I2C (Inter-Integrated Circuit)

I2C is a widely used communication protocol for short-distance communication between devices. Unlike SPI, I2C uses only two wires:

- **SDA (Serial Data)**: The data line, shared between devices.
- **SCL (Serial Clock)**: The clock line, also shared.

Key Features:

- **Multi-master support**: Multiple devices can act as master.
- **Slave addressing**: Devices are addressed using a unique address.
- **Two-wire interface**: Reduces the number of wires needed for communication.

Example (MSP430 I2C - USCI Mode):

```
// Initialize I2C (USCI)
#include <msp430.h>

void I2C_init() {
   P1SEL |= BIT6 + BIT7;  // Configure pins for SDA and SCL
   UCB0CTL1 = UCSWRST;  // Put USCI in reset
   UCB0CTL0 = UCMSTR | UCMODE_3 | UCSYNC; // Master mode, 7-bit addressing
   UCB0CTL1 = UCSSEL_2; // Use SMCLK
   UCB0BR0 = 10;  // Set baud rate
   UCB0BR1 = 0;
   UCB0CTL1 &= ~UCSWRST; // Release USCI from reset
}

void I2C_transmit(unsigned char data) {
   while (!(IFG2 & UCB0TXIFG)); // Wait for TX buffer to be ready
   UCB0TXBUF = data; // Send data
}
```

```
int main() {
   I2C_init();  // Initialize I2C
   I2C_transmit(0x55);  // Send data
   return 0;
}
```

2.3 USB (Universal Serial Bus)

USB is a widely used protocol for communication between computers and peripheral devices. It is designed to support plug-and-play functionality and high-speed data transfer.

- **Full-duplex communication**: Allows data to be transmitted and received simultaneously.
- **Master-slave architecture**: The host (usually a computer) controls the communication, while peripherals (e.g., keyboards, mice) are slaves.

Key Features:

- Supports **high-speed data transfer**.
- Power delivery to peripherals.
- **Plug-and-play** functionality.

2.4 CAN (Controller Area Network)

CAN is a robust communication protocol designed for embedded systems in automotive and industrial applications. It is widely used for real-time control and communication between microcontrollers, sensors, actuators, and other devices.

- **Multi-master**: Allows multiple devices to initiate communication.
- **Message priority**: CAN provides message prioritization based on ID, making it ideal for real-time systems.
- **Error detection**: CAN includes strong error detection mechanisms, ensuring data integrity.

Key Features:

- Multi-master support.
- High error detection and correction capabilities.
- **Real-time** and fault-tolerant communication.

3. Summary

- **USART** is used for asynchronous and synchronous serial communication.
- **USCI** is a more advanced interface than USART, capable of handling **UART**, **SPI**, and **I2C** protocols.

- **USI** is a simpler version of USCI and supports basic serial communication.
- **SPI** is used for fast data transfer between devices, using four lines: MOSI, MISO, SCLK, and SS.
- **I2C** is a two-wire protocol, widely used in embedded systems for communication with multiple devices.
- **USB** is a universal protocol designed for data exchange between peripherals and computers, supporting high-speed communication.
- **CAN** is used in critical applications requiring high-speed, real-time communication, typically in automotive or industrial environments.