**TASK 2 EXPLANATION AND ANSWER**

Here the microcontroller used is PIC16F8076A and the IDE is MPLAB. Data sheet of PIC16F8076A attached. In order to perform SPI there will be a master and in this case only one slave is used. A character ‘a’ is going to transmit from master to slave. So, two programs required, one is for master and other for slave.

**Pseudocode for Master (SPI)**

* First develop the program for master by using MPLAB IDE.
* Configuration should be done before header files like the type of oscillator, oscillator frequency etc.
* After header filer declare the crystal frequency as 4 mega hertz.
* Declare character ‘a’ and transmission register.
* Initialize the transmission register as 0X24 and baud rate as 25 (9600 bits per second).
* For performing SPI two registers such as SSPSTAT and SSPCON should initialize.
* Initialize the value for SSPSTAT and SSPCON as 0X80 and 0X20.
* Declare the pins of port C as input/output (Pin C3 as output C4 as input and C5 as output).
* Transfer the character ‘a’ to TXREG and store it in SSPBUF.
* End of program.

**Pseudocode for Slave (SPI)**

* First develop the program for master by using MPLAB IDE.
* Configuration should be done before header files like the type of oscillator, oscillator frequency etc.
* After header filer declare the crystal frequency as 4 mega hertz.
* Declare character ‘a’ and transmission register.
* Initialize the reception register as 0X90 and baud rate as 25 (9600 bits per second).
* For performing SPI two registers such as SSPSTAT and SSPCON should initialize.
* Initialize the value for SSPSTAT and SSPCON as 0X80 and 0X20.
* Declare the pins of port C as input/output (Pin C3 as input C4 as input and C5 as output).
* Transfer the character ‘a’ to from SSPBUF and store it in RCREG.
* End of program.

**MASTER PROGRAM**

// CONFIG

#pragma config FOSC = XT // Oscillator Selection bits (XT oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = ON // Brown-out Reset Enable bit (BOR enabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include <xc.h>

#define \_XTAL\_FREQ 4000000

void main()

{

char a;

int i;

TXSTA=0X24;

RCSTA=0X90;

SPBRG=25;

SSPSTAT=0X80;

SSPCON=0X20;

TRISC3=0;

TRISC4=1;

TRISC5=0;

while(1)

{

while(RCIF==0);

a=RCREG;

TXREG=a;

while(TRMT==0);

SSPBUF=a;

while(SSPIF==0);

}

}

**SPI SLAVE PROGRAMME**

// CONFIG

#pragma config FOSC = XT // Oscillator Selection bits (XT oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR disabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define \_XTAL\_FREQ 4000000

#include <xc.h>

void main()

{

RCREG=0X90;

SSPSTAT=0X00;

SSPCON=0X25;

TRISC3=1;

TRISC4=1;

TRISC5=0;

TRISB=0X00;

while(1)

{

while(SSPIF==0);

RCREG=SSPBUF;

SSPIF=0;

}

}