Protocoles de Communication dans le Véhicule

TP2: Communication SPI entre PIC16F877A et EEPROM 25AA256

Réalisé par:

- **ELMADI Choaib**
- **ELHAZMIRI Ayoub**

Encadré par:

- **M. Anas HATIM**

Introduction:

Dans ce deuxième TP des protocoles de communication embarqués, l'objectif est de mettre en œuvre une communication SPI entre un microcontrôleur PIC16F877A et une mémoire EEPROM 25AA256. Le travail consiste à écrire des fonctions de lecture et d'écriture, développer un programme complet pour gérer cette communication, et simuler le tout sur Proteus avec l'observation des trames échangées via un débuggeur SPI et un oscilloscope.

Les prototypes des fonctions SPI:

```h
#ifndef SPI_FUNCTIONS_H
#define SPI_FUNCTIONS_H

#include <xc.h>

void SPI_Init(void);
void SPI_Write_Enable(void);
void SPI_Write_Data(unsigned char);
unsigned char SPI_Read_Data(void);

#endif // SPI_FUNCTIONS_H

```

Les définitions des fonctions SPI:

```c
#include "SPI_Functions.h"

void SPI_Init() {
    SSPCON = 0x20;
    SSPSTAT = 0x00;
}

void SPI_Write_Enable() {
    SSPBUF = 0x06;
    while (!PIR1bits.SSPIF);                // Set automatically
    PIR1bits.SSPIF = 0;
}

void SPI_Write_Data(unsigned char data) {
    SSPBUF = data;
    while (!PIR1bits.SSPIF);                // Set automatically
    PIR1bits.SSPIF = 0;
}

unsigned char SPI_Read_Data() {
    SSPBUF = 0x00;
    while (!SSPSTATbits.BF);                // Set automatically
    SSPSTATbits.BF = 0;
    return SSPBUF;
}

```

Les prototypes des fonctions mémoires:

```h
#ifndef MEM_FUNCTIONS_H
#define MEM_FUNCTIONS_H

#include <xc.h>

#include "SPI_Functions.h"

unsigned char data;

void Write_Byte_25AA256(unsigned char, unsigned char, unsigned char);
unsigned char Read_Byte_25AA256(unsigned char, unsigned char);

#endif // MEM_FUNCTIONS_H

```

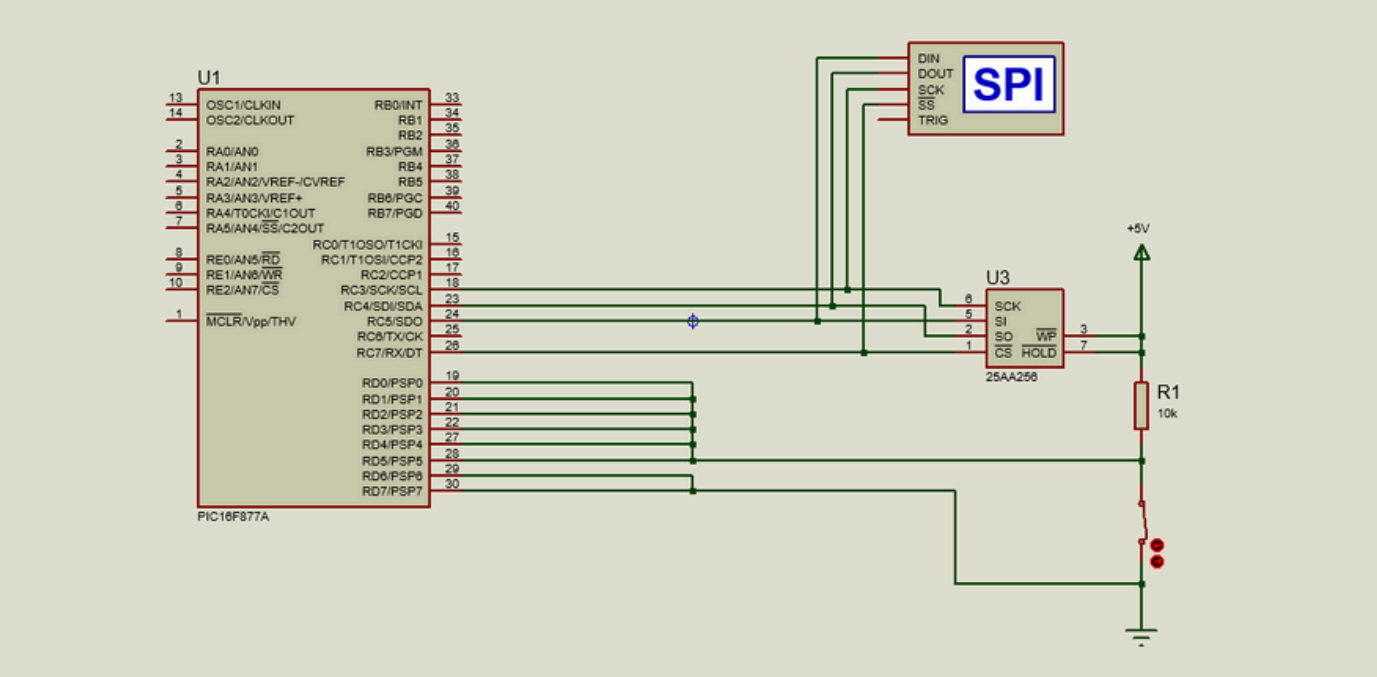
Les définitions des fonctions mémoires:

```c
#include "MEM_Functions.h"

void Write_Byte_25AA256(unsigned char ADS_H, unsigned char ADS_L, unsigned char data) {
    SPI_Write_Data(0×02);
    SPI_Write_Data(ADS_H);
    SPI_Write_Data(ADS_L);
    SPI_Write_Data(data);
}

unsigned char Read_Byte_25AA256(unsigned char ADS_H, unsigned char ADS_L) {
    SPI_Write_Data(0×03);
    SPI_Write_Data(ADS_H);
    SPI_Write_Data(ADS_L);
    data = SPI_Read_Data();

    return data;
}

```

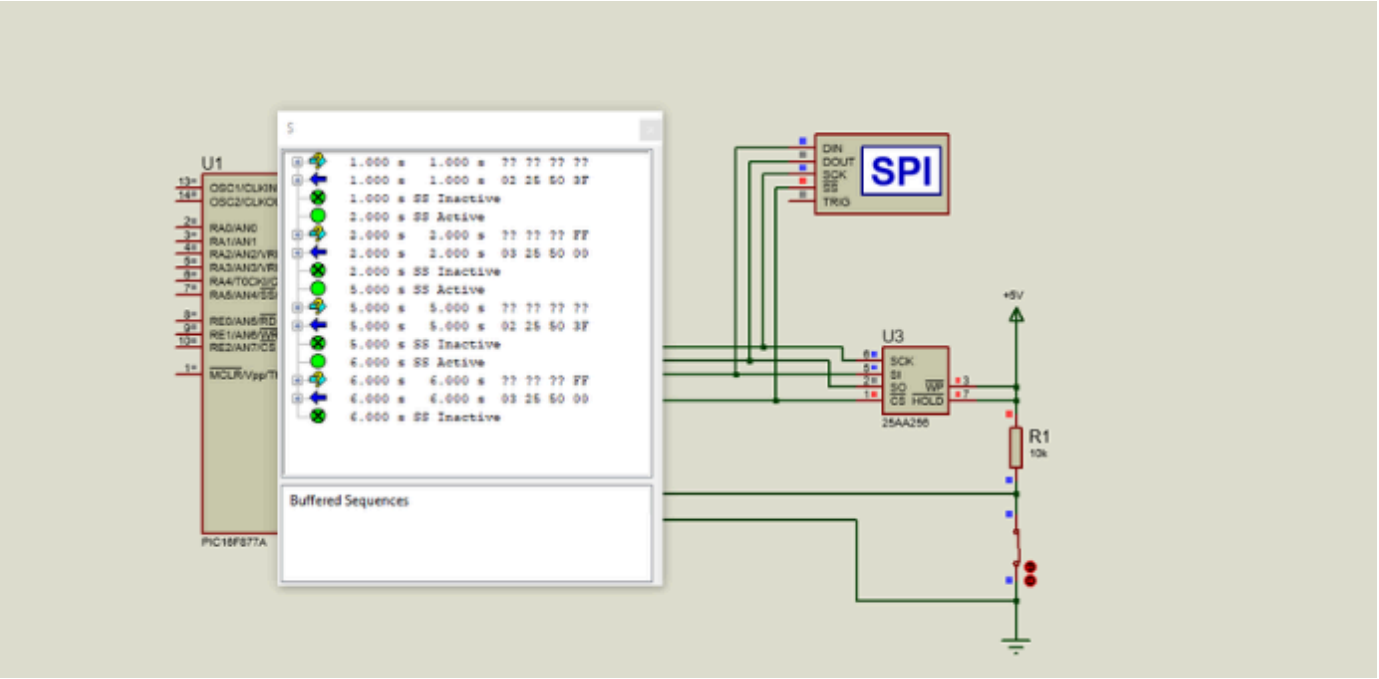
Le programme principal:

```c
#include <xc.h>

#include "MEM_Functions.h"

#define _XTAL_FREQ 20000000

#define ADDRESS_H 0x25
#define ADDRESS_L 0x50

unsigned char data_read;

void main(void) {
    TRISCbits.TRISC3 = 0;       // SCK
    TRISCbits.TRISC4 = 1;       // SDI
    TRISCbits.TRISC5 = 0;       // SDO
    TRISCbits.TRISC7 = 0;       // CS
    TRISD = 0xFF;

    SPI_Init();
    __delay_ms(1000);

    while (1) {
        PORTCbits.RC7 = 0;      // Activate slave
        Write_Byte_25AA256(ADDRESS_H, ADDRESS_L, PORTD);
        PORTCbits.RC7 = 1;      // Deactivate slave
        __delay_ms(1000);

        PORTCbits.RC7 = 0;
        data_read = Read_Byte_25AA256(ADDRESS_H, ADDRESS_L);
        PORTCbits.RC7 = 1;
        __delay_ms(3000);
    }
}

```

Proteus setup:



Proteus setup avec SPI Debugger:



Conclusion:

En conclusion, la communication SPI entre le PIC16F877A et la mémoire EEPROM 25AA256 a été correctement réalisée. La vérification des données lues et écrites a été effectuée à l'aide du débuggeur SPI et l'oscilloscope, permettant d'observer les échanges et valider le bon fonctionnement du protocole.