Protocoles de Communication dans le Véhicule

TP1: Communication Master-Slave I2C entre PIC16F877A et EEPROM 24LC250

Réalisé par:

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

Encadré par:

- **M. Anas HATIM**

Introduction:

Dans ce premier TP des protocoles de communication embarqués, l'objectif est de mettre en œuvre une communication I2C entre un microcontrôleur PIC16F877A et une mémoire EEPROM 24LC250. 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 I2C et un oscilloscope.

Les prototypes des fonctions utilisées:

```
#define RS PORTBbits.RB3
#define RW PORTBbits.RB4
#define EN PORTBbits.RB5

void I2C_Init(int);
void Start_Bit(void);
void Repeated_Start(void);
void Send_Byte_Data(uint8_t);
uint8_t Receive_Byte_Data(void);
void Send_ACK_Bit(void);
void Send_NACK_Bit(void);
void Stop_Bit(void);

void LCD_Init(void);
void LCD_Print_Byte(unsigned char);
void LCD_Print_Bytes(const unsigned char *, unsigned char);
void LCD_Control(unsigned char);
```

Les définitions des fonctions I2C:

```
void I2C_Init(int clock_init_value) {
    SSPADD = (unsigned char)clock_init_value;
    SSPCON = 0x28;
}
void Start_Bit() {
    SSPCON2bits.SEN = 1;
    while (SSPCON2bits.SEN);
    PIR1bits.SSPIF = 0;
}
void Repeated_Start() {
    SSPCON2bits.RSEN = 1;
    while (SSPCON2bits.RSEN);
    PIR1bits.SSPIF = 0;
}
void Send_Byte_Data(uint8_t data) {
    SSPBUF = data;
    while (!PIR1bits.SSPIF);
    PIR1bits.SSPIF = 0;

    if (SSPCON2bits.ACKSTAT) {
        Stop_Bit();
    }
}
uint8_t Receive_Byte_Data() {
    SSPCON2bits.RCEN = 1;
    while (!SSPSTATbits.BF);
    return SSPBUF;
}
```

```
void Send_ACK_Bit() {
    SSPCON2bits.ACKDT = 0;
    SSPCON2bits.ACKEN = 1;
    while (SSPCON2bits.ACKEN);
}
void Send_NACK_Bit() {
    SSPCON2bits.ACKDT = 1;
    SSPCON2bits.ACKEN = 1;
    while (SSPCON2bits.ACKEN);
}
void Stop_Bit() {
    SSPCON2bits.PEN = 1;
    while (SSPCON2bits.PEN);
    PIR1bits.SSPIF = 0;
}
```

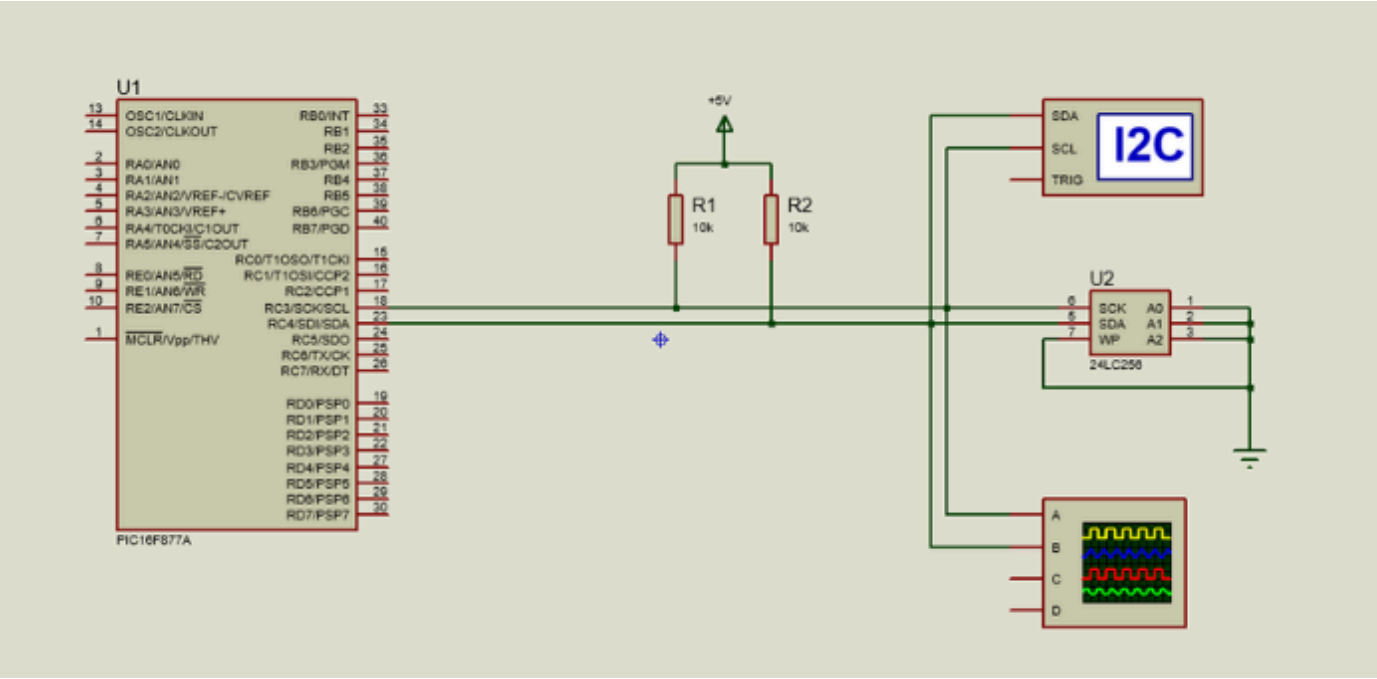
Le programme principal:

```c
void main(void) {
    TRISBbits.TRISB3 = 0;
    TRISBbits.TRISB4 = 0;
    TRISBbits.TRISB5 = 0;
    TRISD = 0x00;

    TRISCbits.TRISC3 = 1;                       // SCL as input
    TRISCbits.TRISC4 = 1;                       // SDA as input

    LCD_Init();

    LCD_Control(0x80);
    LCD_Print_Bytes((const unsigned char *)"Send: ", 6);
    LCD_Control(0xC0);
    LCD_Print_Bytes((const unsigned char *)"Receive: ", 9);

    int clock_init_value = 49;                  // For 100kHz baud-rate
    I2C_Init(clock_init_value);

    uint8_t slave_address_w = 0b10100000;       // Slave address and write bit
    uint8_t slave_address_r = 0b10100001;       // Slave address and read bit
    uint8_t memor_address_h = 0x11;             // Memory address high byte
    uint8_t memor_address_l = 0x01;             // Memory address low byte

    uint8_t data_to_write = 0x00;
    uint8_t received_data = 0x00;

    while (1) {
        Start_Bit();
        Send_Byte_Data(slave_address_w);        // Write the address to write to
        Send_Byte_Data(memor_address_h);
        Send_Byte_Data(memor_address_l);
        Send_Byte_Data(data_to_write);          // Write data
        Stop_Bit();

        __delay_ms(1000);

        Start_Bit();
        Send_Byte_Data(slave_address_w);        // Write the address to read from
        Send_Byte_Data(memor_address_h);
        Send_Byte_Data(memor_address_l);

        Repeated_Start();
        Send_Byte_Data(slave_address_r);
        received_data = Receive_Byte_Data();    // Read data
        Send_NACK_Bit();
        Stop_Bit();

        data_to_write++;
        __delay_ms(1000);
    }

    return;
}
```

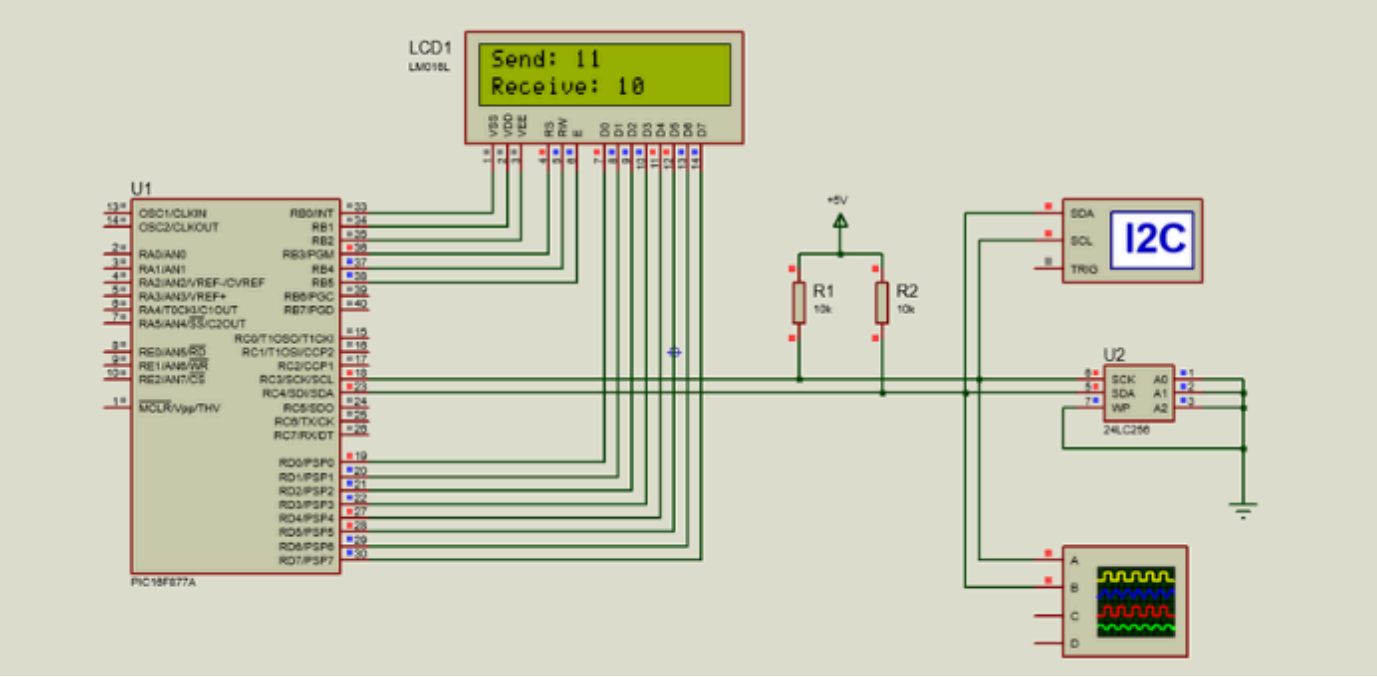
Les définitions des fonctions LCD:

```
void LCD_Init() {
    LCD_Control(0x38);                          // Setup 16X2 LCD model
    LCD_Control(0x06);                          // Auto increment cursor to next element
    LCD_Control(0x0C);                          // LCD on and cursor off
    LCD_Control(0x01);                          // Clear screen
}
void LCD_Print_Byte(unsigned char data) {
    EN = 1;                                     // Enabled
    RS = 1;                                     // Data register
    RW = 0;                                     // Write
    PORTD = data;
    __delay_ms(5);
    EN = 0;                                     // Disabled
}
void LCD_Print_Bytes(const unsigned char *str, unsigned char len) {
    unsigned char i;
    for (i=0; i<len; ++i) {
        LCD_Print_Byte(str[i]);
    }
}
void LCD_Control(unsigned char cmd) {
    EN = 1;                                     // Enabled
    RS = 0;                                     // Instruction register
    RW = 0;                                     // Write
    PORTD = cmd;
    __delay_ms(5);
    EN = 0;                                     // Disabled
}
```

Proteus setup sans LCD:



Proteus setup avec LCD:



Conclusion:

En conclusion, la communication I2C entre le PIC16F877A et la mémoire EEPROM 24LC250 a été correctement réalisée. Pour vérifier les données lues et écrites, un écran LCD a été utilisé afin d'afficher les résultats de manière claire et directe.