# Laboratory Work No. 4

*EEPROM and Dynamic Display*

## 1. Goal and Task

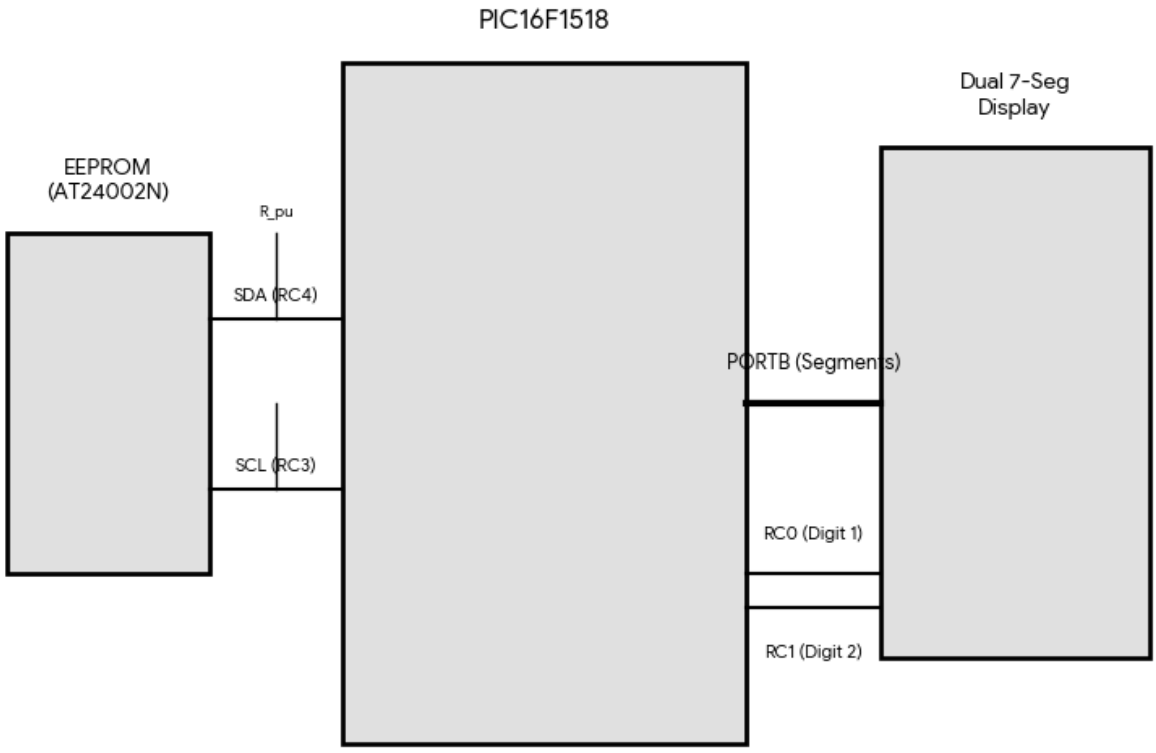
Goal: Learn to use external EEPROM memory connected via I2C interface and output information using dynamic display method.

Task: Create and connect a circuit according to the variant sequence number and program the microcontroller so that two 7-segment LED indicators connected to it dynamically display information read from EEPROM memory. If EEPROM is empty, the microcontroller writes the author's birth day into it.

## 2. Task Variant Data (Variant No. 1)

Based on the provided list, Variant 1 was selected:
- 7-Segment Data Port: PORTB
- 7-Segment Common Pins: PORTC.0 and PORTC.1
- Switching Period: ~5ms
- EEPROM Address: 0x00

## 3. Circuit Diagram



# Laboratory Work No. 4

*EEPROM and Dynamic Display*

## 4. Code with Comments

```
/* * File:   LabWork4.c
 * Author: Halil Ibrahim Bekli
 *
 * Device: PIC16F1518
 * Description:
 * 1. Checks external I2C EEPROM (AT24002N).
 * 2. If empty (0xFF), writes "Birth Day" (e.g., 25).
 * 3. Reads the value back.
 * 4. Displays the value on two 7-segment LEDs using Dynamic Multiplexing (Timer0).
 */

#include <xc.h>

// --- Configuration Bits ---
#pragma config FOSC=INTOSC, WDTE=OFF, PWRTE=ON, MCLRE=ON, CP=OFF, BOREN=OFF
#pragma config CLKOUTEN=OFF, WRT=OFF, STVREN=ON, BORV=LO, LPBOR=OFF, LVP=OFF

#define _XTAL_FREQ 16000000

// --- User Defines ---
#define EEPROM_ADDR_W 0xA0  // Write Address of AT24002N
#define EEPROM_ADDR_R 0xA1  // Read Address of AT24002N
#define MY_BIRTH_DAY  25    // <--- CHANGE THIS TO YOUR BIRTH DAY
#define MEMORY_LOC    0x00  // EEPROM Address to use

// --- Global Variables ---
const unsigned char segmentCodes[] = {
    0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F
};

volatile unsigned char digit1_data = 0; // Tens digit pattern
volatile unsigned char digit2_data = 0; // Ones digit pattern
volatile unsigned char active_digit = 0; // 0 = Digit 1, 1 = Digit 2

// --- I2C Functions ---
void I2C_Init(void) {
    SSP1CON1 = 0b00101000;    // I2C Master mode
    SSP1CON2 = 0x00;
    SSP1ADD = 39;             // 100kHz @ 16MHz Fosc
    SSP1STAT = 0x00;
    TRISCbits.TRISC3 = 1;     // SCL
    TRISCbits.TRISC4 = 1;     // SDA
}

void I2C_Wait(void) {
    while ((SSP1CON2 & 0x1F) || (SSP1STATbits.R_nW));
}

void I2C_Start(void) {
    I2C_Wait();
    SSP1CON2bits.SEN = 1;
}

void I2C_Stop(void) {
    I2C_Wait();
    SSP1CON2bits.PEN = 1;
```

```
}

void I2C_RepeatedStart(void) {
    I2C_Wait();
    SSP1CON2bits.RSEN = 1;
}

void I2C_Write(unsigned char data) {
    I2C_Wait();
    SSP1BUF = data;
}

unsigned char I2C_Read(unsigned char ack) {
    unsigned char incoming;
    I2C_Wait();
    SSP1CON2bits.RCEN = 1;      // Enable Receive
    I2C_Wait();
    incoming = SSP1BUF;         // Read buffer
    I2C_Wait();
    SSP1CON2bits.ACKDT = !ack;  // 0 = ACK, 1 = NACK
    SSP1CON2bits.ACKEN = 1;     // Send ACK/NACK bit
    return incoming;
}

// --- EEPROM Functions ---
void EEPROM_WriteByte(unsigned char addr, unsigned char data) {
    I2C_Start();
    I2C_Write(EEPROM_ADDR_W);
    I2C_Write(addr);
    I2C_Write(data);
    I2C_Stop();
    __delay_ms(10);
}

unsigned char EEPROM_ReadByte(unsigned char addr) {
    unsigned char data;
    I2C_Start();
    I2C_Write(EEPROM_ADDR_W);
    I2C_Write(addr);
    I2C_RepeatedStart();
    I2C_Write(EEPROM_ADDR_R);
    data = I2C_Read(0); // Read with NACK
    I2C_Stop();
    return data;
}

// --- Interrupt Service Routine ---
void __interrupt() ISR(void) {
    if (TMR0IF) {
        PORTCbits.RC0 = 0;
        PORTCbits.RC1 = 0;

        if (active_digit == 0) {
            PORTB = digit1_data;
            PORTCbits.RC0 = 1;
            active_digit = 1;
        } else {
            PORTB = digit2_data;
            PORTCbits.RC1 = 1;
            active_digit = 0;
```

# Laboratory Work No. 4

*EEPROM and Dynamic Display*

```
        }

        TMR0 = 100;
        TMR0IF = 0;
    }
}

// --- Main Application ---
void main(void) {
    unsigned char readValue;
    unsigned char tens, ones;

    OSCCON = 0b01111010; // 16 MHz
    ANSELB = 0; ANSELC = 0;
    TRISB = 0x00;
    TRISCbits.TRISC0 = 0;
    TRISCbits.TRISC1 = 0;
    PORTB = 0x00; PORTC = 0x00;

    I2C_Init();

    readValue = EEPROM_ReadByte(MEMORY_LOC);

    if (readValue == 0xFF) {
        EEPROM_WriteByte(MEMORY_LOC, MY_BIRTH_DAY);
        readValue = MY_BIRTH_DAY;
    }

    tens = readValue / 10;
    ones = readValue % 10;

    digit1_data = segmentCodes[tens];
    digit2_data = segmentCodes[ones];

    OPTION_REG = 0b00000101; // Prescaler 1:64
    TMR0 = 100;
    INTCONbits.TMR0IE = 1;
    INTCONbits.GIE = 1;

    while(1) { }
}
```

# Laboratory Work No. 4

*EEPROM and Dynamic Display*

## 5. Program Algorithm

1. Initialize System: Set Internal Oscillator to 16MHz.
2. Configure GPIO: Set PORTB as Output (Segments), RC0 & RC1 as Output (Digits), and RC3 & RC4 as Inputs (I2C lines).
3. Initialize I2C Module: Master mode, 100kHz clock speed.
4. Read EEPROM: Read byte from address 0x00.
5. Check Data: If read value is 0xFF (Empty), write 'Birth Day' (25) to address 0x00 and update the local variable.
6. Data Processing: Split the number into Tens and Ones digits. Convert digits to 7-segment bit patterns using a lookup table.
7. Configure Timer0: Set Prescaler to 1:64 and preload TMR0 for ~5ms interval. Enable Timer0 Interrupt and Global Interrupts.
8. Main Loop: Enter an infinite loop (all processing is done in ISR).
9. Interrupt Service Routine (ISR):
   a. Check Timer0 Overflow Flag.
   b. Turn off both digits (RC0=0, RC1=0) to prevent ghosting.
   c. Check 'active_digit' flag:
      - If 0: Output 'Tens' pattern to PORTB, Set RC0=1 (Activate Digit 1), Set active_digit=1.
      - If 1: Output 'Ones' pattern to PORTB, Set RC1=1 (Activate Digit 2), Set active_digit=0.
   d. Reset TMR0 value and clear Interrupt Flag.

## 6. Conclusions

**In this laboratory work, I successfully implemented a system to interface an external EEPROM (AT24002N) with a PIC16F1518 microcontroller using the I2C communication protocol. Key achievements include:**

**- Configuring the MSSP module for I2C Master mode operations (Start, Stop, Write, Read, Ack/Nack).**

**- Implementing a logic to check for uninitialized memory (0xFF) and writing default data (Birth Day).**

**- Utilizing Timer0 interrupts to create a dynamic display (multiplexing) driver for a dual 7-segment LED display. This method allows for flicker-free display of data while freeing up the main processor loop.**

**The work demonstrated the practical application of serial memory storage and interrupt-driven display techniques.**