Student Name	蔡汉霖
Fuzhou University ID	*832002117*

Student Name	卿祺果
Fuzhou University ID	*832002127*

Report for EE302FZ Lab 5

Introduction

The following Figure 1 depicts a basic configuration of an embedded system based around the PIC16F877A. And Figure 2 below depicts a system overview of the embedded system and a bidirectional connection to a PC.

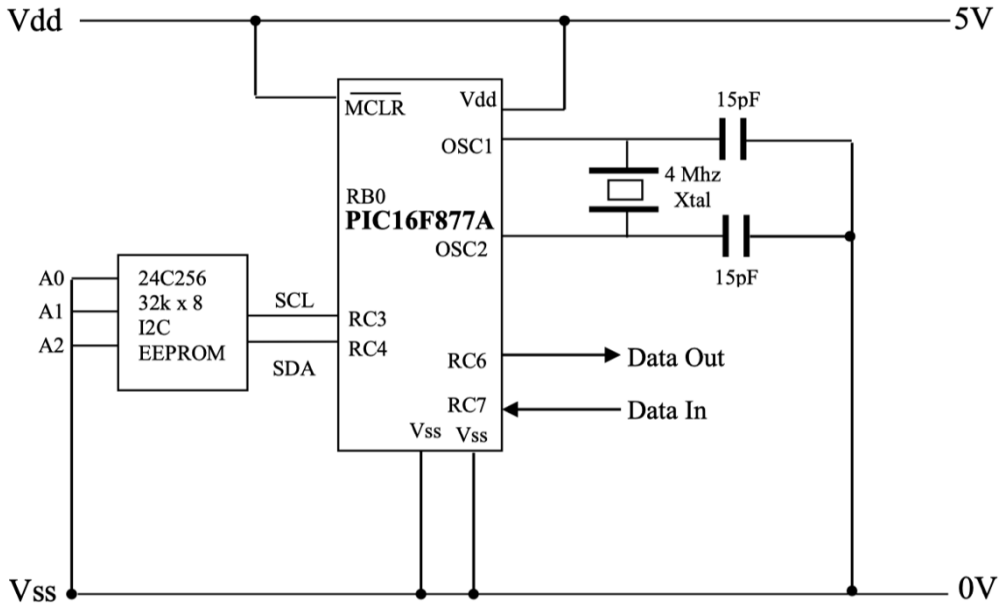

Figure 1 Basic Configuration

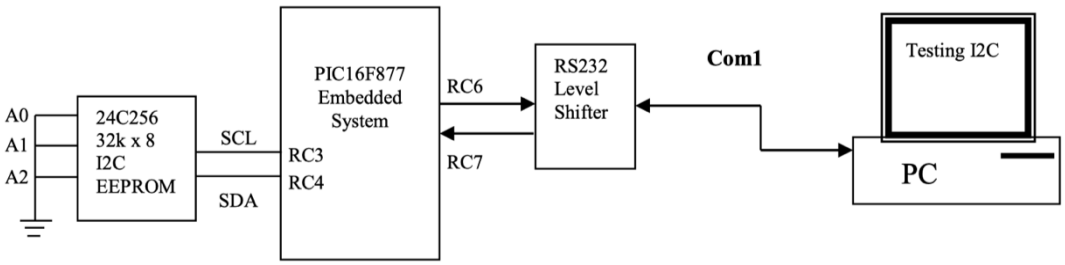

Figure 2 Basic Configuration

Software: MPLAB, WCHSerialPort (in macOS);
Equipment: PIC16F877A, PICkit-3, LED*4, PC and corresponding USB cables.

Part 1

A handout is provided for this question. With the provided handout,

(a) Write a function to WRITE a byte to the external EEPROM.

(b) Show how this function is used.

(c) Write a function to READ a byte from the external EEPROM.

(d) Show how this function is used. (40 marks).

The handwritten answers have been finished, as shown in the Figure 3 and 4.

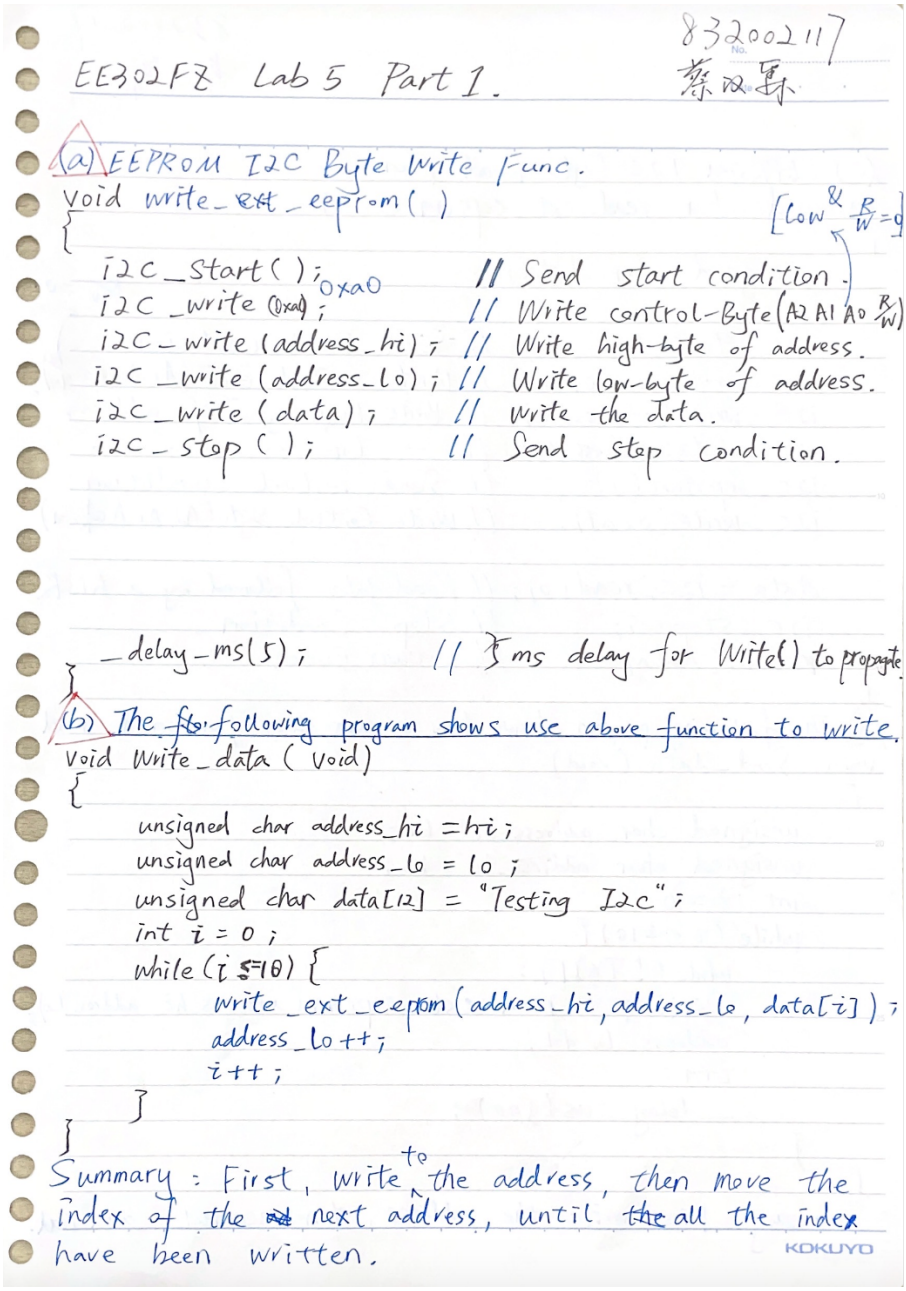

Figure 3 Handwritten answers for Q1 (Page1)

832002111
蔡以軍

(c) EEPROM I2C Byte Read Func.

```
unsigned char read_ext_eeprom ( )
{
    unsigned char data;

    i2c_start();              // Send Start Condition
    i2c_write(0xa0);          // Write Control Byte(A2 A1 A0 R/W).
    i2c_write(address_hi);    // Write high byte of address
    i2c_write(address_lo);    // ... low ... ... ...
    i2c_repStart();           // Send restart condition
    i2c_write(0xa1);          // Write Control Byte(A2 A1 A0 R/W=1)

    data = i2c_read(0);       // Read data followed by a NACK.
    i2c_stop();               // Stop Condition
    return(data);             // Return Func.
}
```

R/W = 0

(d) The following program shows how to use above function to read

```
void Send_data (void)
{
    unsigned char address_hi = hi;
    unsigned char address_lo = lo;
    int i = 0;
    while (i <=10) {
        while (! TXIF);
        TXREG = read_ext_eeprom(address_hi, address_lo);
        address_lo ++;
        i++;
        __delay_us(500);
    }
}
```

Summary: First write the address, then reconnect, and read.

Campus

Figure 4 Handwritten answers for Q1 (Page2)

Part 2

Sketch out the Start Condition and Stop Condition for I2C. You can reference the 24C256 EEPROM Datasheet provided on Moodle. (10 marks).

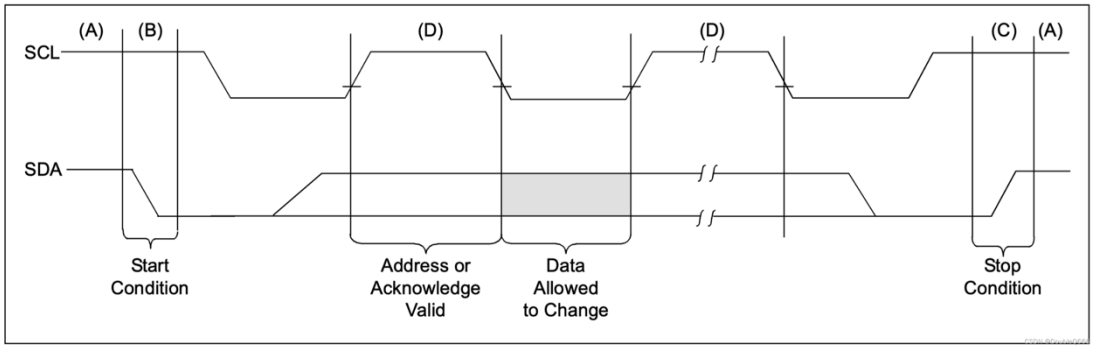

Figure 5 Data transfer sequence on the serial bus (I2C)

The above Figure 5 illustrates the data transfer sequence on the serial bus. Referring to Figure 5, the Start Condition and Stop Condition for I^2C can be described as follows:

1. **The Start Condition: when bus operation is started, SCL remains high and SDA switches from high-level to low-level.**
2. **The Stop Condition: when bus operation is stopped, SCL remains high and SDA switches from low-level to high-level.**

Part 3

An I2C library with an I2C Byte Write and I2C Byte Read function will be provided for this part once Part 1 is complete. Write a program which writes the following string "Testing I2C" to the I2C EEPROM. Then reads these characters back and then sends the characters read from the I2C EEPROM to the PC via the UART. Use best coding practices.

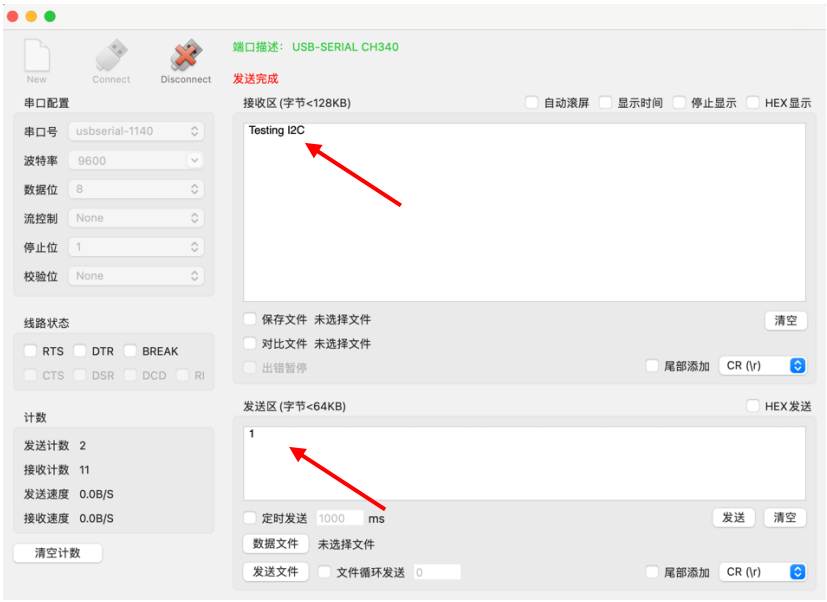

Figure 6 Testing result of the following program

As shown in the above Figure 6, the testing result have been given. And the complete code of C program is shown in the following Table 1.

Table 1

The C program based on MPLAB

```
#include "I2C_EE302.h"
// Implemented by Hanlin Cai & Qiguo Qing
// CONFIG
#pragma config FOSC = XT
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config BOREN = OFF
#pragma config LVP = OFF
#pragma config CPD = OFF
#pragma config WRT = OFF
#pragma config CP = OFF
// Oscillator Selection bits (XT oscillator)
// Watchdog Timer Enable bit (WDT disabled)
```

```
// Power-up Timer Enable bit (PWRT disabled)
// Brown-out Reset Enable bit (BOR disabled)
// Low-Voltage (Single-Supply) In-Circuit Serial
// Programming Enable bit (RB3 is digital I/O, HV // on MCLR must be used for
programming)
// Data EEPROM Memory Code Protection bit //(Data EEPROM code protection off)
// Flash Program Memory Write Enable bits (Write // protection off; all program
memory may be
// written to by EECON control)
// Flash Program Memory Code Protection bit //(Code protection off)
// Include standard PIC library
#ifndef _XTAL_FREQ
// Unless already defined assume 4MHz system frequency
// This definition is required to calibrate the delay functions, __delay_us() and
__delay_ms()
#define _XTAL_FREQ 4000000
#endif

#define LED1 RC0
#define LED2 RC1

#define ON 1
#define OFF 0

void setup() {
    i2c_init();

    PORTC = 0x00;
    TRISC = 0xC0;
    TXSTA = 0x24;
    RCSTA = 0x90;
    SPBRG = 0x19;
}

void send(char* str) {
    for (int i = 0; str[i]; i++) {
        if (str[i] < 0 || str[i] >= 256) {
            continue;
        }
        while (!TXIF);
        TXREG = str[i];
    }
    while (!TXIF);
    TXREG = 0x0D;
```

```
    __delay_ms(1);
}

char recv() {
    while (!RCIF);
    char ret = RCREG;
    RCREG = 0;
    return ret;
}

void write_str(char* str) {
    for (int i = 0; str[i]; i++) {
        write_ext_eeprom(0, i, str[i]);
    }
}
void main() {
    setup();

    // 往 EEPROM 中写入数据
    write_str("Testing I2C");

    int len = 11;

    // 当 PC 向单片机发送 1 时开始读数据
    while (1) {
        char op = recv();
        if (op == '1') {
            break;
        }
    }

    for (int i = 0; i < len; i++) {
        char x = read_ext_eeprom(0, i);

        // 向 PC 发送 EEPROM 中读到的数据
        while (!TXIF);
        TXREG = x;
        __delay_ms(100);
    }
}
```

Summary for Lab 5

In Lab 5, we have learned the basic knowledge of I^2C and we explored how to utilize PIC DIP-40 to verify the function of our C programs. Thanks to Dr. Wu for her guidance.

Hanlin CAI 832002117

Qiguo QING 832002127

In 2022/12/01