|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name: | 蔡汉霖 |  | Student Name: | Hanlin CAI |
| Fuzhou University ID: | 832002117 |  | Maynooth University ID: | 20122161 |

## EE302FZ Project: Real-time Multiple LCD Display

**Introduction**

This report proposes **a real-time multiple LCD display project**. In a nutshell, the project has four different main functions, including all seven peripherals available in the PIC16F877A. The following Table 1 shows the seven different peripherals.



The following Figure 1 depicts a basic configuration of an embedded system based around the PIC16F877A and LCD.

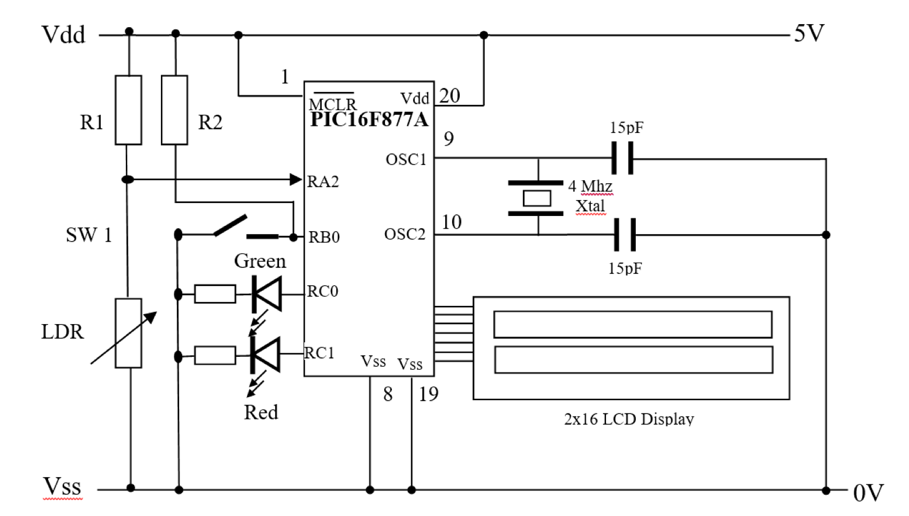


Figure 1 Basic Configuration

**What you need for reproducing this project:**

Software: MPLAB IDE, WCHSerialPort (macOS);

Equipment: PIC16F877A, PICkit-3, LCD, LED, and corresponding USB cables.

**The Table 2 below illustrates the main functions included in this project, where the blue fonts represent the** **peripherals, and the green fonts represent the LCD or GUI contents.**



**Part 1: Block diagrams of this project**

**The following Figure 2 shows the block diagram of the proposed project, all the four main functions have been described in detail.**

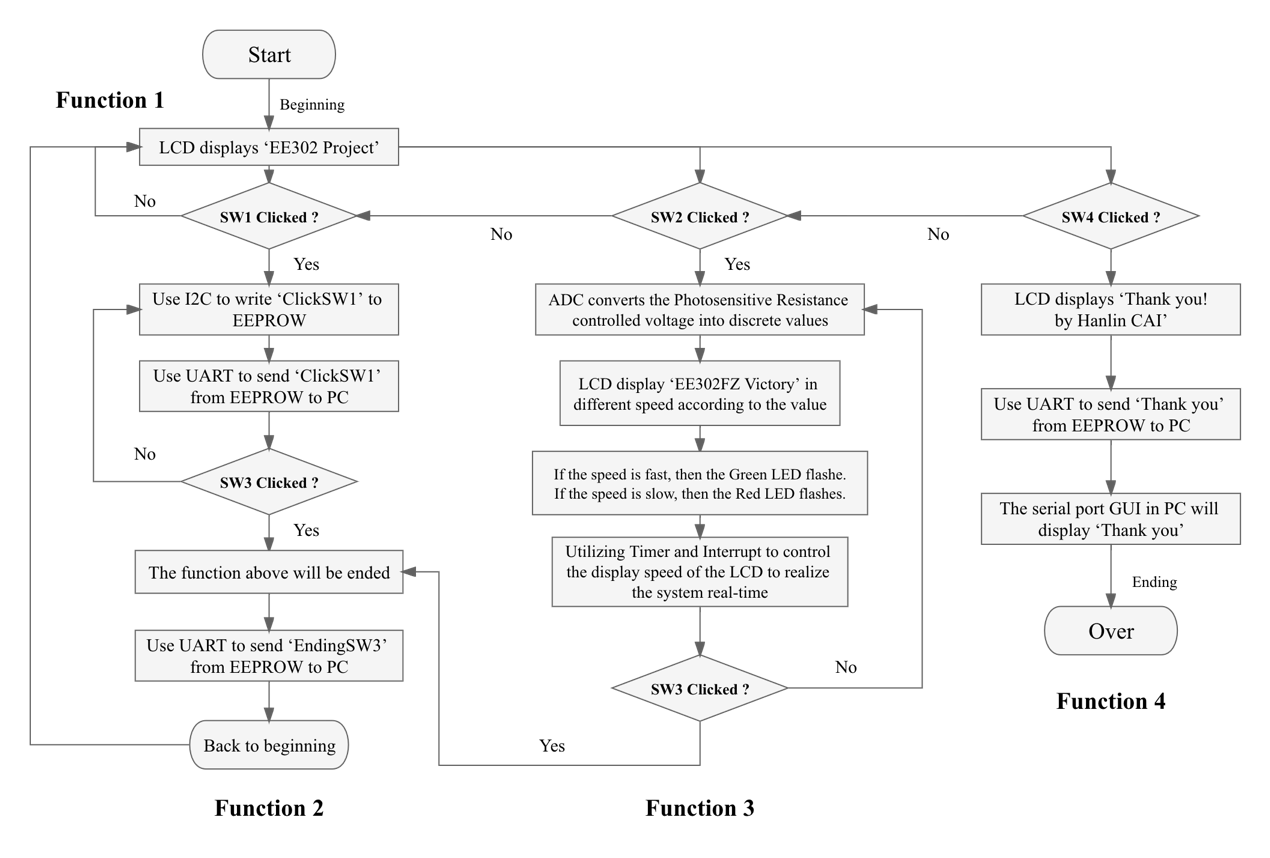


Figure 2 Block diagram of the proposed system

**Part 2: Real-time aspect of this project**

The proposed system is a soft real-time system, which means the meeting of deadline is not compulsory for every task, but the process should get processed and give the result. Therefore, this project utilizes the following peripherals and designs to realize the real-time aspect:

1. The operation of I2C and EEPROM can be finished within a specific time;
2. The data transfer using UART can be finished at a specific time;
3. The ADC can be done with a desired time;
4. Using Timer and Interrupt to control the speed of LCD display (Function 2), and the timer counts time accurately. Therefore, the proposed system is real-time.

**Acknowledgements**

I gratefully acknowledge Dr. Wu for her generous guidance and support during the EE302FZ course. I hope to thank Mr. Yanxiang Wang and Mr. Zhuoran Wang for their helpful advice. Lastly, I would like to thank the TA who carefully evaluate this report.

Hanlin CAI

16th Dec 2022

**Part 3: Programs of this project**

**The main C program is shown in the following Table 3.** **And the corresponding LCD and I2C head file are illustrated in the Table 4 and Table 5.**

|  |
| --- |
| Table 3: C Main Program |
| *This is the c* *program for the main function.* |
| /\*   \* File: main.c  \* Author: Hanlin CAI (20122161)  \* Comments: This is the main function for EE302FZ final porject.  \* Includes: Basci I/O + ADC + LCD + UART + I2C + Real-time  \* Latest update in 2022/12/15  \*/  // 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 <xc.h> // Include standard PIC library #include "ee302\_Lcd.h" // Include required header file for LCD functions #include "ee302\_I2C.h" // Include required header file for I2C functions  #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 SW1 RB0 // Assign Label SW1 to PortB bit 0 (RB0) #define SW2 RB1 // Assign Label SW2 to PortB bit 1 (RB1) #define SW3 RB2 // Assign Label SW2 to PortB bit 1 (RB1) #define SW4 RB3 // Assign Label SW2 to PortB bit 1 (RB1)  #define CLOSED 0 #define OPEN 1 #define HIGH 1 #define LOW 0 #define hi 0x11 #define lo 0x55 #define G\_led RC0 #define R\_led RC1  /\*\*\*\*\*\*\*\*\*\*Global variables\*\*\*\*\*\*\*\*\*\*/ int data1 = 0; int data2 = 0; unsigned char newData = 0; unsigned char AddData = 0; char ch = 0; int col = 16; int flag = 1; int i = 0;   /\*\*\*\*\*\*\*\*\*\*\*USER FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*/  //prototypes void setup(void); void loop(void); char receive(void); void send\_str(char \*str);  // 3 version of LCD display void LCDBegin(void); void LCDTitle(void); void LCDEnd(void);  // I2C function void readLDR\_data(void); void Write\_data(void); void Send\_data(void); void receive\_data(void); void Light();   void main() {  setup(); // do initialisation    LCDBegin(); // LCD display version 1 (Begin).    if (SW1 == CLOSED) // If SW1 closed then  {  for(;;){  Write\_data();  Send\_data();  \_\_delay\_ms(500);    if(SW3 == CLOSED) // If SW3 closed then  {  send\_str("Ending\_SW3");  break;  }  }  }    if (SW2 == CLOSED) // If SW2 closed then  {   for (;;) // endless loop  {   readLDR\_data();  loop();  LCDTitle();  send\_str("Working\_SW2");    if(SW3 == CLOSED) // If SW3 closed then  {  send\_str("Ending\_SW3");  break;  }  }   }    if (SW4 == CLOSED) // If SW4 closed then  {  for(;;){  LCDEnd();  send\_str("Thank you!");  \_\_delay\_ms(200);  }  } }  void setup(void) // Setup stuff {  PORTD = 0b11111111;  TRISD = 0b00000000;  TXSTA = 0x24; //Set TXEN bit to enable transmit.  //Set BRGH bit for Baud Rate table selection.  RCSTA = 0x90; //Set CREN bit for continuous read.  //Set SPEN bit to enable serial port.  SPBRG = 0x19; //Set Baud Rate as 9600   TRISC = 0xd8;  TRISD = 0x00;   T1CON = 0x21;  INTCON = 0xc0;  PIE1 = 0x21;  PIR1 = 0x00;  Lcd8\_Init(); // Required initialisation of LCD to 8-bit mode  TRISB = 0x07; // Set PORTB bit 0 as input   TRISC = 0xd8;  TRISD = 0x00;  PORTD = 0xff;   TXSTA = 0x24;  RCSTA = 0x90;  SPBRG = 0x19; // set Baud Rate   T1CON = 0x21;  INTCON = 0xc0;  PIE1 = 0x21;  PIR1 = 0x00;  //Set the ACD registers    TRISA = 0b00000101; // Set PORTA bits 0 and 2 are output  //TRISC = 0b00000000; // Set PORTC bit 1 and 0 as output  PORTC = 0b00000010;  ADCON0 = 0b01010001; // Set FOSC/8,RA2 as analog input and A/D converter module is powered up  ADCON1 = 0b00000010; // Set Left justified  OPTION\_REG &= 0b01111111;        TRISC = 0b1101100; // RC6 and RC7 must be set to inputs for USART.   TXSTA = 0x24; // Set TXEN bit to enable transmit.  // Set BRGH bit for Baud Rate table selection.  RCSTA = 0x90; // Set CREN bit for continuous read.  //Set SPEN bit to enable serial port.  SPBRG = 0x19; // Set Baud Rate to 9600  i2c\_init(); // Required initialisation of I2C        PORTD = 0b11111111;  TRISD = 0b00000000;  //TRISC = 0xC0; //RC6 and RC7 must be set to inputs for USART.   TXSTA = 0x24; //Set TXEN bit to enable transmit.  //Set BRGH bit for Baud Rate table selection.  RCSTA = 0x90; //Set CREN bit for continuous read.  //Set SPEN bit to enable serial port.  SPBRG = 0x19; //Set Baud Rate to 9600 }  void loop(void) {  if (data1 > flag)  {  T1CON = 0x01;  }  else if (data1 <= flag)  {  T1CON = 0x31;  } }   void \_\_interrupt() // Interrupt identifier  isr(void) // Here is interrupt function. {  if (TMR1IF)  {  TMR1IF = 0;  i++;  if (i == 2) {  col--;  i = 0;  } else {    }  }  else if (RCIF)  {  RCIF = 0;  ch = RCREG;  } }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  char receive(void) {  RCIF = 0;  while (!RCIF);  return RCREG; }   void send\_str(char \*str) {  int index = 0;  char ch = \*str;   while (ch != '\0')  {  ch = \*(str + index);  index++;  while (!TXIF)  ;  TXREG = ch;  } }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/     // 3 LCD Display Version // Beginning of the project, version 1. void LCDBegin(void) {  Lcd8\_Set\_Cursor(1,1); // select line 2 of LCD  Lcd8\_Write\_String("EE302 Project "); // display "EE302 Victory" on second line of LCD // Lcd8\_Set\_Cursor(2,1); // select line 2 of LCD // Lcd8\_Write\_String("by Hanlin CAI "); // display "EE302 Victory" on second line of LCD  \_\_delay\_ms(1000); }  // Once SW2 is closed, LCD version 2. void LCDTitle(void) {  if (col < 0)  {  col = 16;  }  Lcd8\_Clear();  Lcd8\_Set\_Cursor(1, col); // select line 1 of LCD  Lcd8\_Write\_String("EE302FZ"); // print "Project" on line 1 of LCD    Lcd8\_Set\_Cursor(2, col); // select line 2 of LCD  Lcd8\_Write\_String("Victory"); // display "EE302 Victory" on second line of LCD  \_\_delay\_ms(300); }  // Once SW4 is closed, LCD version 3. void LCDEnd(void) {  Lcd8\_Set\_Cursor(1,1); // select line 2 of LCD  Lcd8\_Write\_String("Thank you! "); // display "EE302 Victory" on second line of LCD  Lcd8\_Set\_Cursor(2,1); // select line 2 of LCD  Lcd8\_Write\_String("by Hanlin CAI "); // display "EE302 Victory" on second line of LCD  \_\_delay\_ms(1000); }  void Light(){   if(data1+data2\*0.1<1.5){  R\_led = HIGH;  G\_led = LOW;  }else{  R\_led = LOW;  G\_led = HIGH;    } }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // ADC function void readLDR\_data(void){  if(1){  \_\_delay\_ms(150);  \_\_delay\_us(50);  GO\_nDONE = 1;  while(GO\_nDONE){  continue;  }    if(ADRESH!=newData){  AddData = ADRESH;  data1 = (AddData\*5/255);  data2 = (AddData\*10\*5/255)%10;  }    Light();  newData = AddData;  \_\_delay\_ms(100);  } }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ // I2C function void Write\_data(void) {  unsigned char address\_hi = hi;  unsigned char address\_lo = lo;  unsigned char data[12] = "Click\_SW1";  int i = 0;  while (i <= 10)  {  write\_ext\_eeprom(address\_hi, address\_lo, data[i]);  address\_lo++;  i++;  } }  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);  } }  // This program is created by Hanlin CAI in 2022/12/15 // EE302FZ Final Project. |

|  |
| --- |
| Table 4: LCD head file |
| *This is the head file* *(.h) for the LCD display function.* |
| /\*   \* File: ee302\_Lcd.h  \* Author: Hanlin CAI (20122161)  \* Latest update in 2022/12/14  \* Comments: This is the head file for LCD display function.  \* Based on the open-source code implemented by JMaloco.  \*/   /\*\*\*\*\*\*\*\*\*\*\*USER FUNCTIONS\*\*\*\*\*\*\*\*\*\*\* 1. Lcd8\_Init() - Must be called to initialise LCD. - Note what SFRs are effected and be sure not to overwrite these in yourt program initialisation.  2. Lcd8\_Clear() - Call this to Clear LCD display  3. Lcd8\_Set\_Cursor(char a, char b) - The function sets the position of the cursor on the LCD. A = Line (1 or 2); B = Position (1 - 16).  4. Lcd8\_Write\_Char(char a) - Write a character to the LCD e.g. Lcd8\_Write\_Char('A');  5. Lcd8\_Write\_String(char \*a) - Write a string to the LCD e.g. Lcd8\_Write\_Char("Hello");  6. Lcd8\_Shift\_Right() - Shift the displayed characters one place to the right.  7. Lcd8\_Shift\_Left() - Shift the displayed characters one place to the left. \*/  #include <xc.h> #ifndef \_XTAL\_FREQ  // Unless already defined assume 4MHz system frequency  // This definition is required to calibrate \_\_delay\_us() and \_\_delay\_ms()  #define \_XTAL\_FREQ 4000000 #endif  #define RS RE0 #define RW RE1 #define EN RE2 #define D0 RD0 #define D1 RD1 #define D2 RD2 #define D3 RD3 #define D4 RD4 #define D5 RD5 #define D6 RD6 #define D7 RD7 //LCD Functions Developed by electroSome /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ //LCD 8 Bit Interfacing Functions void Lcd8\_Port(char a) {  if(a & 1)  D0 = 1;  else   D0 = 0;    if(a & 2)  D1 = 1;  else  D1 = 0;    if(a & 4)  D2 = 1;  else  D2 = 0;    if(a & 8)  D3 = 1;  else  D3 = 0;    if(a & 16)  D4 = 1;  else  D4 = 0;   if(a & 32)  D5 = 1;  else  D5 = 0;    if(a & 64)  D6 = 1;  else   D6 = 0;    if(a & 128)  D7 = 1;  else  D7 = 0; }  void Lcd8\_Cmd(char a) {   RS = 0; // => RS = 0  Lcd8\_Port(a); //Data transfer  EN = 1; // => E = 1  \_\_delay\_ms(5);  EN = 0; // => E = 0 }   //=============USER FUNCTIONS=============  void Lcd8\_Init() {  TRISE = 0x00;  TRISD = 0x00;  ADCON1 = 0x07;  RE1 = 0;    Lcd8\_Port(0x00);  RS = 0;  \_\_delay\_ms(25);  ///////////// Reset process from datasheet /////////  Lcd8\_Cmd(0x30);  \_\_delay\_ms(5);  Lcd8\_Cmd(0x30);  \_\_delay\_ms(15);  Lcd8\_Cmd(0x30);  /////////////////////////////////////////////////////  Lcd8\_Cmd(0x38); //function set  Lcd8\_Cmd(0x0C); //display on,cursor off,blink off  Lcd8\_Cmd(0x01); //clear display  Lcd8\_Cmd(0x06); //entry mode, set increment }   Lcd8\_Clear() {  Lcd8\_Cmd(1); }  void Lcd8\_Set\_Cursor(char a, char b) {  if(a == 1)  Lcd8\_Cmd(0x80 + b);  else if(a == 2)  Lcd8\_Cmd(0xC0 + b); }   void Lcd8\_Write\_Char(char a) {  RS = 1; // => RS = 1  Lcd8\_Port(a); //Data transfer  EN = 1; // => E = 1  \_\_delay\_ms(4);  EN = 0; // => E = 04 }  void Lcd8\_Write\_String(char \*a) {  int i;  for(i=0;a[i]!='\0';i++)  Lcd8\_Write\_Char(a[i]); }  void Lcd8\_Shift\_Right() {  Lcd8\_Cmd(0x1C); }  void Lcd8\_Shift\_Left() {  Lcd8\_Cmd(0x18); }  // End LCD 8 Bit Interfacing Functions // Lastly modified by Hanlin CAI // EE302FZ Final Project. |

|  |
| --- |
| Table 5: I2C head file |
| *This is the head file (.h) for the I2C function.* |
| /\*   \* File: ee302\_I2C.h  \* Author: Hanlin CAI (20122161)  \* Latest update in 2022/12/14  \* Comments: This is the head file for I2C transfer function.  \* Based on the open-source code implemented by JMaloco.  \*/   /\*\*\*\*\*\*\*\*\*\*\*USER FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*  1.i2c\_init() - Must be called to initialise I2C device. - Note what SFRs are effected (TRISC) and be sure not to overwrite these in yourt program initialisation.  2.write\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo, unsigned char data);  3.unsigned char read\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo); - Returns a single character read. \*/  #include <pic.h> #define \_XTAL\_FREQ 4000000 unsigned char data[20];   /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*typedef for data types \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ typedef signed char BYTE; typedef signed short WORD; typedef signed long DWORD; typedef float FLOAT; typedef unsigned char UBYTE; typedef unsigned int UWORD; typedef unsigned long UDWORD;   #define TRUE 1 #define FALSE 0 #define HIGH 1 #define LOW 0  #define RX\_BUFFER\_SIZE 20  /\*\* T R I S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ #define INPUT\_PIN 1 #define OUTPUT\_PIN 0   void i2c\_init(void); void write\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo, unsigned char data); unsigned char read\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo); unsigned char i2c\_write( unsigned char i2cWriteData ); int i2c\_read( unsigned char ack ); void i2c\_stop(void); void i2c\_repStart(void); void i2c\_start(void); void i2c\_waitForIdle(void); void write\_string(unsigned char address\_hi,unsigned char address\_lo, const char\* ptr); void read\_string(unsigned char address\_hi,unsigned char address\_lo, unsigned char data[], int length);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void i2c\_waitForIdle(void) {  while (( SSPCON2 & 0x1F ) | R\_nW ) {}; // wait for idle and not writing }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void i2c\_start(void) {  i2c\_waitForIdle();  SEN=1; }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void i2c\_repStart(void) {  i2c\_waitForIdle();  RSEN=1; }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void i2c\_stop(void) {  i2c\_waitForIdle();  PEN=1; }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int i2c\_read( unsigned char ack ) { unsigned char i2cReadData; i2c\_waitForIdle();  RCEN=1;  i2c\_waitForIdle();  i2cReadData = SSPBUF;  i2c\_waitForIdle();  if ( ack )  {  ACKDT=0; //ACK  }  else  {  ACKDT=1; //NACK  }  ACKEN=1; // send acknowledge sequence  return( i2cReadData ); }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char i2c\_write( unsigned char i2cWriteData ) {  i2c\_waitForIdle();  SSPBUF = i2cWriteData; return ( ! ACKSTAT ); // function returns '1' if transmission is acknowledged }   //==================================================== // // MAIN USER FUNCTIONS // // - I2C Initialisation // - EEPROM Byte Write // - EEPROM Byte Read  // //====================================================   void i2c\_init(void) {  // Do in main code TRISC = 0b00011000; // set SCL and SDA pins as inputs  SSPCON = 0x38; // set I2C master mode  SSPCON2 = 0x00;  SSPADD = 0x0A; // 100k at 4Mhz clock  CKE=1; // use I2C levels   SMP=1; // disable slew rate control   PSPIF=0; // clear SSPIF interrupt flag  BCLIF=0; // clear bus collision flag }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ void write\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo, unsigned char data)  {  i2c\_start(); //Send Start Condition  i2c\_write(0xa0); //Write Control Byte (A2,A1,A0 all low, R/W = 0)  i2c\_write(address\_hi); //Write high byte of address   i2c\_write(address\_lo); //Write low byte of address   i2c\_write(data); //Write data  i2c\_stop(); //Send Stop condition  \_\_delay\_ms(5); //Necessary 5ms delay for write to propagate }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  unsigned char read\_ext\_eeprom(unsigned char address\_hi,unsigned char address\_lo) {  unsigned char data;   i2c\_start(); //Send Start Condition  i2c\_write(0xa0); //Write Control Byte (A2,A1,A0 all low, R/W = 0)  i2c\_write(address\_hi); //Write high byte of address   i2c\_write(address\_lo); //Write low byte of address   i2c\_repStart(); //Send reStart Condition  i2c\_write(0xa1); //Write Control Byte (A2,A1,A0 all low, R/W = 1)  data=i2c\_read(0); //Read Data followed by a NACK  i2c\_stop(); //Send Stop condition  return(data); }  // Lastly modified by Hanlin CAI // EE302FZ Final Project. |