VE373 Lab 5

Group 5:Qian Dong， Yifan Hu， Yinchen Ni, Jeffery Ma.

Target of this Lab

Implement a simple voltmeter with PIC32 microcontroller [^1].

Source Code

```
/*********************************************************************************
***********************************'
*                                                                  Lab 5.c
/*********************************************************************************
***********************************/
#include <p32xxxx.h>
#include <stdio.h>
#include "LCD.h"

volatile double voltage = 0;
char startStr[10];

void LCD_init() {
    DelayMsec(16);      //wait for 16 ms(over 15ms)
    RS = 0;             //send command
    E = 1;
    Data = LCD_IDLE;    //function set - 8 bit interface
    E = 0;
    DelayMsec(5);       //wait for 5 ms
    E = 1;
    Data = LCD_IDLE;    //function set - 8 bit interface
    E = 0;
    DelayUsec(200);     //wait for 200 us(over 100us)
    E = 1;
    Data = LCD_IDLE;    //function set
    E = 0;
    DelayMsec(5);
    E = 1;
    Data = 0b00100000; //function set - 4 bit interface
    E = 0;
    LCD_putchar(LCD_2_LINE_4_BITS);
    DelayMsec(100);
    LCD_putchar(LCD_DSP_CSR);
    DelayUsec(40);
    LCD_putchar(LCD_CLR_DSP);
    DelayMsec(5);
    LCD_putchar(LCD_CSR_INC);
}

/* Send one byte c (instruction or data) to the LCD */
void LCD_putchar(uchar c) {
    E = 1;
```

```
    Data = c;   //sending higher nibble
    E = 0;      //producing falling edge on E
    E = 1;
    Data <<= 4; //sending lower nibble through higher 4 ports
    E = 0;      //producing falling edge on E
}

/* Display a string of characters *s by continuously calling LCD_putchar() */
void LCD_puts(const uchar *s, int length) {
    // ...your code goes here
    // ...NOTE: must wait for at least 40 us after sending each character to
        // the LCD module.
    RS = 1;
    int i;
    for (i = 0; i < length; i++) {
        LCD_putchar(s[i]);
        DelayMsec(30);
    }
    RS = 0;
}

/* go to a specific DDRAM address addr */
void LCD_goto(uchar addr) {
    // ...send an address to the LCD
    // ...NOTE: must wait for at least 40 us after the address is sent
    // TODO;
    RS = 0;
    E = 1; Data = 0x80 + addr; E = 0;
    E = 1; Data <<=4;          E = 0;
    DelayUsec(40);
}

/* configure timer SFRs to generate num us delay*/
void DelayUsec(uchar num) {
    // ...your code goes here
    PR1 = num;
    flags.timer1_done = 0;
    T1CONSET = 0x8000;              // turn on the timer 1
    TMR1 = 0x0;                     // reset the timer 1
    while ( ! flags.timer1_done );  // loop until flag 04 (for timer 1) is set
    flags.timer1_done = 0;          // reset the flags
    T1CONCLR = 0x8000;                    // turn off timer
}

/* configure timer SFRs to generate 1 ms delay*/
void GenMsec() {
    // ...your code goes here
    DelayUsec(250);
    DelayUsec(250);
    DelayUsec(250);
    DelayUsec(250);
}

/* Call GenMsec() num times to generate num ms delay*/
void DelayMsec(uchar num) {
    uchar i;
    for (i=0; i<num; i++) {
        GenMsec();
```

```
    }
}

/* timer 1 interrupt handler */
#pragma interrupt timer_1_interrupt ipl3 vector 4
void timer_1_interrupt(void) {
    T1CONCLR = 0x8000;                // turn off the timer 1
    IFS0CLR = 0x0010;           // clear the flag ;l;hfor timer 1
    flags.timer1_done = 1;      // set up the flags
    TMR1 = 0;
}


void display (float data){
    sprintf(startStr,"   %-4.2lfV",data);
    RS=0;
    LCD_putchar(0x1); // clear display
    DelayMsec(30);
    LCD_goto(0x0);
    DelayMsec(30);
    RS=1;
    LCD_puts(startStr, 8);
}

void MCU_init(void) {
    /* setup I/O ports to connect to the LCD module */
    // let A,B,D,E all to be output
    TRISDCLR = 0xFFFF;
    TRISECLR = 0xFFFF;
    TRISACLR = 0xFFFF;
    // TRISBCLR = 0xFFFF;

    /* setup Timer to count for 1 us and 1 ms */
    // ...your code goes here
    SYSKEY = 0x0;               // Ensure OSCCON is lock
    SYSKEY = 0xAA996655;        // Unlock sequence part 1 of 2 back to back
instructions.
    SYSKEY = 0x556699AA;        // Unlock sequence part 2 of 2 back to back
instructions.
    OSCCONbits.NOSC = 0x0007;   // Write new osc src value to NOSC control bits
-- FRS, with original freqeuncy as 8 MHz
    OSCCONbits.FRCDIV = 0x3; // the prescale of FRC is 8
    OSCCONbits.PBDIV = 0x0;    // PBCLK is SYSCLK divided by 1. {(Not changed
here)Clock is multiplied by 15. PLL output is divided by 1} -- PBCLK has
freqeuncy 1 MHz
    OSCCONbits.OSWEN = 0x0001;  // Initiate clock switch by setting OSWEN bit.
    SYSKEY = 0x0;               // Write non-key value to perform a re-lock.

    while(OSCCONbits.OSWEN);    // Loop until OSWEN = 0. Value of 0 indicates
osc switch is complete.

    T1CON = 0x0;
    PR1 = 0xFFFF;
    T1CONSET = 0x8000;

    /* Configure Timer interrupts */
    INTCONbits.MVEC = 1;        // multi-vector mode
    IPC1SET = 0x000d;           // timer 1: priority is 3, subpriority is 1
```

```
    // IPC2SET = 0x000d;           // timer 2: priority is 3, subpriority is 1
    IFS0CLR = 0x0110;           // clear the flags for timer 1 and timer 2

    /* enable global and individual interrupts */
    asm( "ei" );                // enable interrupt globally
    IEC0SET = 0x0110;           // enable interrupt for timer 1 and timer 2
}

int main(void) {
    MCU_init();
    LCD_init();
    AD1PCFG = 0xFFF7; // PORTB = Digital; RB2 = analog
    AD1CON1 = 0x40;// Timer3 period match ends sampling, integer 16-bit
    AD1CHS = 0x00030000;// Connect RB3/AN3 as CH0 input

    AD1CSSL = 0;
    AD1CON3 = 0x0000;// Sample time is TMR3, TAD = internal TPB * 2
    AD1CON2 = 0x0000;// Interrupt at the end of each conversion
    // set TMR3 to time out every 2 ms
    T3CONSET = 0; // 16-bit mode, prescaler 1:1, internal clock
    TMR3 = 0x0; // Clear TMR3
    PR3 = 8000; // period = 2ms, frequency = 500 Hz
                // if SYSCLK = 80 MHz, then prescale = 1:32, PR3 = 5000
    T3CONSET = 0x8000; // Start TMR3
    AD1CON1SET = 0x8000; // turn ON the ADC
    AD1CON1SET = 0x0004;// start auto sampling every 2 mSecs
    while (1) {
        int i = 0;
        voltage = 0;
        for (; i < 50; i++)
        {
            while (!IFS1 & 0x0002){};// conversion done?
            voltage += (double)ADC1BUF0;
            IFS1CLR = 0x0002;// clear ADIF
        }
        display(voltage/1024.0/50*3.3); // display the average
    }// repeat
    return 0;
}
```

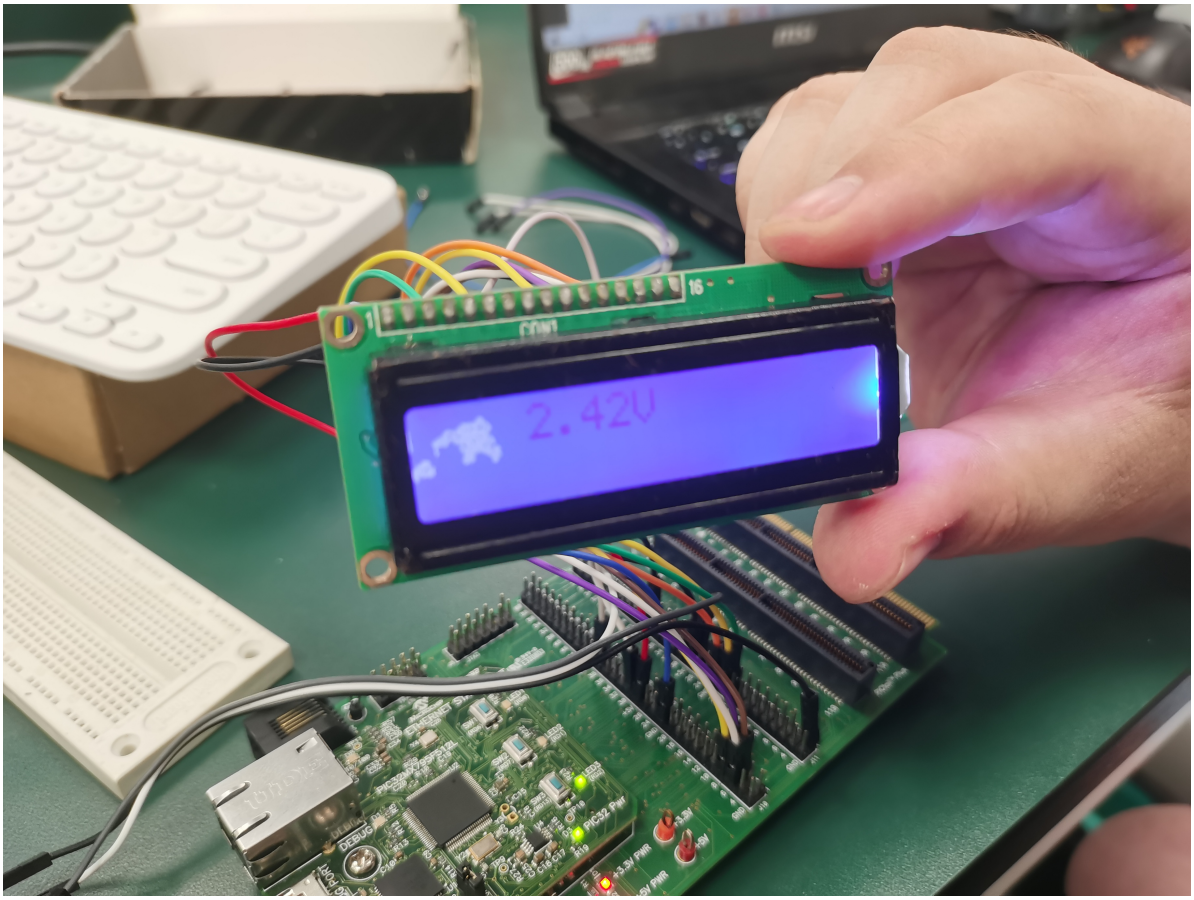
Notice: Here the LCD.h is the header file used in Lab3.

Key Notes

1. Use `OSCCONbits` to set up the clock source, peripheral clock divider, need to enter `SYSKEY` before the change.
2. Use `AD1CON` , `AD1CHS` to control the ADC module of PIC32.

Experiment Result

For different value of resistance:



Result Analysis and discussion

From the LCD ,we can view the measurement of different value of voltage on the resistance.

As the VDD is 3.3V, from the picture we can find its about 73 percent and 54 percent of full resistance.