Reporte Practica 3 - LCD, Keyboard, Timers

Diseño en sistemas de chip

Diego Armando Limón de León A01638247

Jose Miguel Figarola Prado A01632557

Giancarlo Franco Carrillo A01638108

Tecnológico de Monterrey

11 de mayo del 2021

In this lab the idea is to integrate aspects of the three modules we have seen so far: LCD display, 4x4 matrix keyboard and timers. Thus, before proceeding, be sure that your code for the LCD and the keyboard works properly. You can use either the 8- or 4-bit option for the LCD code (or even the functionalities provided by the I2C library)

The lab will be divided into two parts, described as follows:

Part 1

Menu and output management. Write a very simple program that displays the following message in the LCD screen

PRESS BUTTON

R: 1 B: 2 G: 3

Then write a function that recovers the data from the get_key() function (plus a decoder) and switch on a led depending on which key was pressed (1: Red, 2: Blue, 3: Green) and display

RED/BLUE/GREEN

LED IS ON!

The led should remain on for a few seconds, then go off and display the initial menu again.

Analisis

In the first part of the code, we initialize the functions required, as shown below in the line 2, we included the TextLCD module, this module helps us start the LCD in an easier way. With that said we have two different delay functions, micro and milliseconds, both functions will help us make a delay for the keypad and the LCD respectively. Further on, we initialize the keypad with their GPIOs and we got a function to get the pressed key from the matrix, also we have a decoder to tell which key was pressed.

As requested on the instructions, we have to use the RGB LEDs which are initialized in the LED_init function, then given by the decoder in the LED_set function we can turn on the LED requested.

```
1  #include "mbed.h"
2  #include "TextLCD.h"
3
4  TextLCD lcd(PTD0, PTD2 ,PTD3, PTD4, PTD5, PTD6, PTD7, TextLCD::LCD16x2); // rs, e, d4-d7
5  void delayMs(int n); //delay milisec
6  void delayUs(int n); //delay microsec
7  void keypad_init(void);
8  char keypad_getkey(void); //get value
9  void LED_init(void);
10 void LED_set(int value); //decoder
```

The main function has a while loop that prints in the first line a instruction “Press Button” and then in the second line it displays the options: R for red with the 1 key pressed; G for green with 2 key pressed; B for blue with the 3 key pressed.

```
12 int main() {
13     unsigned char key;
14 
15     while(1) {
16         lcd.locate(2,0);
17         lcd.printf("Press Button");
18         lcd.locate(2,1);
19         lcd.printf("R:1 G:2 B:3");
20         LED_init();
21         keypad_init();
22         key = keypad_getkey();
23         LED_set(key); //set LEDs according to the key code
24     }
25 }
26 
```

As explained in the introductory paragraphs, we have the initialization functions.

```
27 void keypad_init(void) {
28     SIM->SCGC5 |= 0x0800;  /* enable clock to Port C */
29     PORTC->PCR[0] = 0x103; /* PTD0, GPIO, enable pullup*/
30     PORTC->PCR[1] = 0x103; /* PTD1, GPIO, enable pullup*/
31     PORTC->PCR[2] = 0x103; /* PTD2, GPIO, enable pullup*/
32     PORTC->PCR[3] = 0x103; /* PTD3, GPIO, enable pullup*/
33     PORTC->PCR[4] = 0x103; /* PTD4, GPIO, enable pullup*/
34     PORTC->PCR[5] = 0x103; /* PTD5, GPIO, enable pullup*/
35     PORTC->PCR[6] = 0x103; /* PTD6, GPIO, enable pullup*/
36     PORTC->PCR[7] = 0x103; /* PTD7, GPIO, enable pullup*/
37     PTD->PDDR = 0x0F; /*make PTD7-0 as input pins */
38 }
39 
40 void LED_init(void) {
41     SIM->SCGC5 |= 0x400; /* enable clock to Port B */
42     SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
43     PORTB->PCR[18] = 0x100; /* make PTB18 pin as GPIO */
44     PTB->PDDR |= 0x40000; /* make PTB18 as output pin */
45     PTB->PSOR |= 0x40000; /* turn off red LED */
46     PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
47     PTB->PDDR |= 0x80000; /* make PTB19 as output pin */
48     PTB->PSOR |= 0x80000; /* turn off green LED */
49     PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
50     PTD->PDDR |= 0x02; /* make PTD1 as output pin */
51     PTD->PSOR |= 0x02; /* turn off blue LED */
52 }
```

The keypad function that obtains the pressed key is given this way.

```
54 char keypad_getkey(void) {
55
56     int row, col;
57     const char row_select[] = {0x01, 0x02, 0x04, 0x08};
58     /* one row is active
59     check to see any key pressed */
60     PTC->PDDR |= 0x0F; /* enable all rows */
61     PTC->PCOR = 0x0F;
62     delayUs(2); /* wait for signal return */
63     col = PTC-> PDIR & 0xF0; /* read all columns */
64     PTC->PDDR = 0; /* disable all rows */
65     if (col == 0xF0)
66         return 0; /* no key pressed */
67     /* If a key is pressed, we need find out which key.*/
68     for (row = 0; row < 4; row++) {
69         PTC->PDDR = 0; /* disable all rows */
70
71         PTC->PDDR |= row_select[row]; /* enable one row */
72         PTC->PCOR = row_select[row]; /* drive active row low*/
73
74         delayUs(2); /* wait for signal to settle */
75         col = PTC->PDIR & 0xF0; /* read all columns */
76
77         if (col != 0xF0) break;
78         /* if one of the input is low, some key is pressed. */
79     }
80     PTC->PDDR = 0; /* disable all rows */
81     if (row == 4)
82         return 0; /* if we get here, no key is pressed */
83
84     /* gets here when one of the rows has key pressed
85     check which column it is*/
86
87     if (col == 0xE0) return row * 4 + 1; /* key in column 0 */
88     if (col == 0xD0) return row * 4 + 2; /* key in column 1 */
89     if (col == 0xB0) return row * 4 + 3; /* key in column 2 */
90     if (col == 0x70) return row * 4 + 4; /* key in column 3 */
91     return 0; /* just to be safe */
92 }
```

As mentioned before, we needed a decoder so the board can know which activity to do, given the key that was pressed.

```
void LED_set(int value) {
    /* use cases to control leds */
    switch(value) {
        case 1:
            PTB->PCOR = 0x40000;  /* turn on red LED */
            lcd.cls();
            lcd.locate(4,0);
            lcd.printf("RED");
            lcd.locate(2,1);
            lcd.printf("LED IS ON!");
            delayUs(2000);
            lcd.cls();
            PTB->PSOR = 0x40000; /* turn off red LED */
            break;
        case 2:
            PTB->PCOR = 0x80000;  /* turn on green LED */
            lcd.cls();
            lcd.locate(4,0);
            lcd.printf("GREEN");
            lcd.locate(2,1);
            lcd.printf("LED IS ON!");
            delayUs(2000);
            lcd.cls();
            PTB->PSOR = 0x80000; /* turn off green LED */
            break;
        case 3:
            PTD->PCOR = 0x02;  /* turn on blue LED */
            lcd.cls();
            lcd.locate(4,0);
            lcd.printf("BLUE");
            lcd.locate(2,1);
            lcd.printf("LED IS ON!");
            delayUs(2000);
            lcd.cls();
            delayUs(1000);
            PTD->PSOR = 0x02; /* turn off blue LED */
            break;
```

The remaining functions are the micro and milliseconds delay.

```
 /* Delay n milliseconds
The CPU core clock is set to MCGFLLCLK at
41.94 MHz in SystemInit(). */
void delayMs(int n)
{
    int i;
    int j;
    for(i = 0 ; i < n; i++)
        for(j = 0 ; j < 7000; j++) { }
}

/* Delay n milliseconds
The CPU core clock is set to MCGFLLCLK at
41.94 MHz in SystemInit(). */
void delayUs(int n)
{
    int i;
    int j;
    for(i = 0 ; i < n; i++)
        for(j = 0 ; j < 7000; j++) { }
}
```

Part 2

Ascending Timer: The goal of the program is to implement a simple timer, as the ones used in sports or music. The code should proceed as follows:

It should print a hello and stay there for 5 seconds (you can use simple delay for this), then a second message should appear in the first line asking for the user to introduce the number of seconds.

To make this interesting, the code should be able to accept at least two digits, so you need to introduce the get_key(function) into a while loop that only stops when you press another not numerical key (for instance the * or # key.)

The introduced value should set the TMP_MODULO register in one timer and start it, either immediately after the termination key was selected or by pressing the same key again.

Then the LCD should show a message Counting and showing the current count value.

Once the timer has elapsed, you can use a buzzer (if you have it) or a led to mark that the timer went to zero.

Analisis

The given code first show in the LCD a “Hello” for 5 seconds, the it implements the key pressed function that is then displayed on the LCD, by the given buttons, we have a time countdown that is shown in the LCD, after the countdown reaches 0, it turns on the red LED on the board and also it makes a buzzer sound with two different frequencies thanks to the PWM. Now that we have the TextLCD.h module, we now have implemented the code with Keypad.h module; also we have defined the frequencies as 262 and 494.

```
#include "mbed.h"
#include "TextLCD.h"
#include "Keypad.h"
#define c 262
#define d 494

TextLCD lcd(PTD0, PTD2, PTD3, PTD4, PTD5, PTD6, PTD7); // rs, rw, e, d4-d7
Keypad Kpad(PTC4, PTC3, PTC0, PTC7, PTC11, PTC10, PTC6, PTC5); // col 1-4, row 1-4
PwmOut buzzer(PTA5); //Set buzzer with pwm for different frequencies

void LED_init(void);
void counter(int n); //contador
int isA_number(char key); //is my key a numer or not
void delayMs(int n); //delay milisec with internal clock
bool keypress(char key); //check if a key has been pressed
```

As shown above, we now have a PwmOut that instantiates a buzzer that is connected to the PTA5 in the board. Also we got the LED_init to initialize the LED (red one in this case) from the board.

```
void LED_init(void)
{
    SIM->SCGC5 |= 0x400; /* enable clock to Port B */
    SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
    PORTB->PCR[18] = 0x100; /* make PTB18 pin as GPIO */
    PTB->PDDR |= 0x40000; /* make PTB18 as output pin */
    PTB->PSOR |= 0x40000; /* turn off red LED */
    PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
    PTB->PDDR |= 0x80000; /* make PTB19 as output pin */
    PTB->PSOR |= 0x80000; /* turn off green LED */
    PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
    PTD->PDDR |= 0x02; /* make PTD1 as output pin */
    PTD->PSOR |= 0x02; /* turn off blue LED */
}
```

The counter function receives a int number and starts the countdown until it reaches 0

```
void counter(int n)
{

    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Counting...");

    while(n != 0) {
        lcd.locate(0,1);
        lcd.printf("%d \n", n);
        PTD->PCOR = 0x02; /*turn on blue led*/
        delayMs(800); //stop one second
        PTD->PSOR |= 0x02; /* turn off blue LED */
        delayMs(200);
        n -= 1;
    }
```

We implemented a function called isA_number, this function biscally recognizes the key pressed from the matrix if it is a number or not.

```
92 int isA_number(char key)
93 {
94     //check if a numerical key has been pressed
95     for( int i=0; i<=9; i++) {
96         if((key-'0') == i) {
97             return i;
98         }
99     }
.00    //if we get here no numerical key has been pressed
.01    return -1; //return -1 in order to avoid an error
.02 }
```

The keypress function is a flag that detects if a key was pressed or not

```
86 bool keypress(char key)
87 {
88     //check if a key has been pressed this function will be used as a flag
89     return key != '/0';
90 }
```

As part 1, we need the implementation of both the micro and milliseconds delay.

```
123 /* Delay n milliseconds
124 The CPU core clock is set to MCGFLLCLK at
125 41.94 MHz in SystemInit(). */
126 void delayMs(int n)
127 {
128     int i;
129     SysTick->LOAD = 41940 - 1;
130     SysTick->CTRL = 0x5; /* Enable the timer and choose sysclk as the clock source */
131
132     for(i = 0; i < n; i++) {
133         while((SysTick->CTRL & 0x10000) == 0)
134             /* wait until the COUTN flag is set */
135         { }
136     }
137     SysTick->CTRL = 0;
138     /* Stop the timer (Enable = 0) */
139 }
```

Finally we have the main function, this main function has a while loop that shows the “Hello” in the LCD display for 5 seconds, and them displays the “Introduce Seconds:” in two different lines, as asked by the LCD, you have to press the number you want for the countdown and with “#” you introduce the numbers. A second while is implemented to show the seconds and the countdown. Then as requested in the instructions and described in the

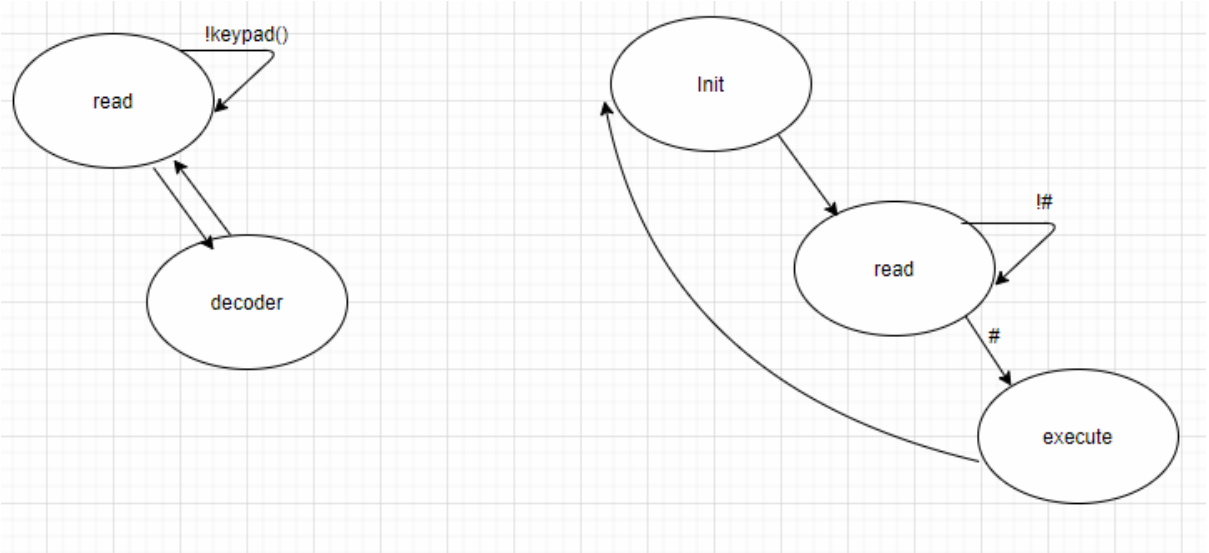
introduction of the analisis, a red LED from the board turns on, a buzzer starts making sounds (two different sounds) and in the LCD it is displayed “Time’s up”.

```
int main()
{
    char key; //value from the keyboard
    int num; //value from the keyboard
    int cont; //total amount of seconds of delay
    int pos; //pos for the cont
    LED_init();
    while(1) {
        lcd.locate(0,0);
        lcd.printf("Hello");
        delayMs(5000);
        lcd.cls();
        pos = 0;
        cont = 0;
        key = '*'; //reset all variables
        lcd.locate(0,0);
        lcd.printf("Introduce");
        lcd.locate(0,1);
        lcd.printf("Seconds: ");

        while(key != '#') {
            key = Kpad.ReadKey();
            num = isA_number(key);
            delayMs(200);
            //we do this code only if there is a numerical char
            if(num != -1 && keypress(key)) { //condition + flag
                cont = cont*10 + num;
                lcd.locate(8+pos,1);
                lcd.printf("%d \n", num); //display the seconds
                pos += 1; //get to the next position
            }
        }//we get out of the cicle when user press # or *
        //once here we call our timer
        counter(cont);
        PTB->PCOR = 0x40000; /*turn on red led*/
        lcd.cls();
        lcd.locate(0,0);
        lcd.printf("Time's up");
        buzzer.period(1/float(c)); // set PWM period
        buzzer = 0.5;// set duty cycle
        wait(0.5);
        buzzer.period(1/float(d));// set PWM period
        buzzer = 0.5;// set duty cycle
        wait(0.5);
        delayMs(2000);
        PTB->PSOR = 0x40000; /*turn off red led*/
        lcd.cls();
        buzzer = 0;
        //we go back again
    }
}
```

Finite state machines

The diagrams given below describe the states from each part of the practice, the left one describes how the code from part 1 works and the right is for part 2.



“Read” state in part one waits for a key to be pressed, if no key is pressed, it calls itself back, when a key is read it gets processed by the “decoder” state so that a LED turns on depending on what the decoder selects, also an auxiliary message is sent to the LCD.

“Init” state for the second part, resets and initializes variables and flags that will be used in the “read” state, a “hello” message is displayed for 5 seconds. Further on, “read” state waits for a numerical key to be pressed or the “#” key which means “enter”, also the numerical pressed keys given to the circuit are displayed in the LCD and saved in a variable. Finally, “execute” state starts the countdown and when it reaches 0, several alerts are triggered, then the program goes back again to the “init” state.

Video

Link del video para mejor visualización: https://youtu.be/SF5u2DvYqEk