Reporte Practica 4 - LCD, Keyboard, Interrupts

Diseño en sistemas de chip

Diego Armando Limón de León A01638247

Jose Miguel Figarola Prado A01632557

Giancarlo Franco Carrillo A01638108

Tecnológico de Monterrey

xx de abril del 2021

In this lab the idea is to integrate aspects of the four modules we have seen so far: LCD display, 4x4 matrix keyboard, timers and interrupts. 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.

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

Part 1.Simple GPIO Interrupt.

Implement the example one seen in class: the main program toggles the red LED continuously and can be interrupted by a push button connected to port PTA1. The interrupt routine service (ISR) for this button is simply toggle the green LED for a short time, then moves back to tread mode to toggle the red LED again.

Analisis

As first part of the code, we have got to include the necessary module of mbed, create the objects from the InterruptIn module and initialize the functions we are going to use.

```
#include "mbed.h"

InterruptIn blue(PTA1);
InterruptIn green(PTA2);

int IRQ; //global variable

void flip(void);
void no_flip(void);
void LED_init(void);
void PORTA_IRQHandler(void);
void delayMs(int n);
```

As we are going to use the internal LEDs from the KL25z board, the LED_init function declares the GPIOs that are going to be needed.

```
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 */
}
```

When a push button is pressed while the code is running, the code is going to stop whatever it is doing, calling it an interruption ¿What is the interruption going to do? Written in the main function that is going to be described later on the document, the code is send to PORTA_IRQHANDLER function, this function will toggle the green LED whenever the button is pressed. As we can see in the code, we have got to clear the IRQ flag.

```
52 void PORTA_IRQHandler(void)
53 {
54     /* toggle green LED (PTB19) three times */
55     for (int i = 0; i < 3; i++) {
56         PTB->PDOR &= ~0x80000; /* turn on green LED */
57         delayMs(500);
58         PTB->PDOR |= 0x80000; /* turn off green LED */
59         delayMs(500);
60     }
61     //clear interrupt flag
62     IRQ=0;
63 }
```

Whenever a push button is pressed, it is important to clear the flag or just don change the flag state, depending on the button state. For that piece of code we have two functions, flip and no_flip. As we can see flip function changes the IRQ state so the interruption can start its code.

```
65 void flip(void)
66 {
67     IRQ=1;
68 }
69
70 void no_flip(void)
71 {
72     IRQ = 0;
73 }
74
```

As every other code described in the practices, we need a delayMs function.

```
75 /* Delay n milliseconds */
76 void delayMs(int n)
77 {
78     int i;
79     int j;
80     for(i = 0 ; i < n; i++) {
81         for (j = 0; j < 7000; j++) {}
82     }
83 }
```

Finally to integrate the functions needed, the main function calls the LED_init function, initializes the push buttons as PullUps and states which function depending on the state, might be done, while there is no interruptions, the red LED will toggle every half second, and when the IRQ state change because of a pressed button, the code will do the PORTA_IRQHANDLER function.

```
14 int main()
15 {
16     LED_init();
17     //initialize both buttons
18     blue.mode(PullUp);
19     blue.rise(&flip);  // attach the function to the raising edge
20     blue.fall(&no_flip);  // attach function to the falling edge
21
22     green.mode(PullUp);
23     green.rise(&flip);  // attach the function to the raising edge
24     green.fall(&no_flip);  // attach function to the falling edge
25
26     IRQ=0;
27
28     while(1) {
29         PTB->PTOR |= 0x40000; /* toggle red LED */
30         delayMs(500);
31         if(IRQ==1) {
32             PORTA_IRQHandler();
33         }  // end if
34     } // end while
35 }
```

Part 2.Distinguishing interrupts from different pins.

In this part, you should implement the second example seen in class: two buttons are connected to the KL25Z board though the port A (PTA1 and PTA2), and both can interrupt the main process running in thread mode is the processor (again, just the red LED being toggled continuously).

As there is just one interrupt for the port A, we need to implement a mechanism for differentiating between the two ports; this can be done through flags in the ISFR register. Please remember to properly disable interrupts before the initialization code in your program. Also, is important to clear the interrupt after having served it, otherwise we might never go back to the main program!!!

Analisis

While part 1 implements just one push button, this part 2 requires a second push button, the functions from part 1 are still the same but in the PORTA_IRQHANDLER function, we need the necessary changes. First of all we need to know which button was pressed so we are going to use button.read() function from InterruptIn to read the state from the buttons.

```
InterruptIn blue(PTA1);
InterruptIn green(PTA2);
```

We gave different names to the buttons to differentiate from each other. If the green button is pushed, as stated in the IRQ_HANDLER function, the code will stop and will toggle the green LED; if the blue button is pressed, it will toggle the blue LED.

```
54 void PORTA_IRQHandler(void)
55 {
56     int i;
57     if(green.read() == 1) {
58         /* toggle green LED (PTB19) three times */
59         for (i = 0; i < 3; i++) {
60             PTB->PDOR &= ~0x80000; /* turn on green LED */
61             delayMs(500);
62             PTB->PDOR |= 0x80000; /* turn off green LED */
63             delayMs(500);
64         }
65     }
66     if(blue.read() == 1) {
67         for (i = 0; i < 3; i++) {
68             PTD->PDOR &= ~0x02; /* turn on blue LED */
69             delayMs(500);
70             PTD->PDOR |= 0x02; /* turn off blue LED */
71             delayMs(500);
72         }
73     }
74     //clear interrupt flag
75     IRQ=0;
76 }
```

Part 3. Event counter.

The goal of this last part of the lab is to integrate concepts of interrupts and timers, and put together a simple application involving the LCD and the 4x4 matrix keyboard. The main idea is to reuse the code of part 1 of this lab, but instead of toggling LEDs, we will integrate into the second part of the previous lab (ascending timer): when the counter is counting, if interrupt from PTA1 is activated, the code should execute and ISR that halts the counter and shows a message such as “PAUSED”, the main program can resume if you press the * key

Analisis

As we have seen in the other practices, we have integrated the LCD and the Keypad, so having the TextLCD and Keypad modules. What now changes is the implementation of interruption declared as pause with the Interruptin module. Basically the code displays a “Hello” message on the LCD for 5 seconds, and then asks to input the number of seconds to count, when the time is up, a buzzer starts to ring and a “Time’s up” message is displayed on the LCD. First of all we start with the initializations.

```
1  //implementacion codigo para la parte 3 practica 4 clearing interrupt flag doesn´t work
2  //we'll be using board reset to start the program
3  #include "mbed.h"
4  //librerias de apoyo
5  #include "TextLCD.h" //peter gresherd 2010 RW
6  #include "Keypad.h" //grant phillips
7  #define c 262
8  #define d 294
9  //interrupciones
10 InterruptIn pause(PTA1);//pausa
11 int IRQ;
12 void PORTA_IRQHandler(void);
13 void flip(void);
14 void no_flip(void);
15 //inicializacion de actuadores
16 TextLCD lcd(PTD0, PTD2, PTD3, PTD4, PTD5, PTD6, PTD7); // rs, rw, e, d4-d7
17 Keypad Kpad(PTC5, PTC6, PTC10, PTC11, PTC7, PTC0, PTC3, PTC4); // col 1-4, row 1-4
18 PwmOut buzzer(PTA12); //buzzer init
19 void LED_init(void);
20 //funciones varias
21 void main_code(void);
22 int isA_number(char key); //is my key a numer or not
23 bool keypress(char key); //check if a key has been pressed
24 void task1(void); //routine for a interrupt
25 //global variables
26 char key; //value from the keyboard
27 int cont; //total amount of seconds of delay
28 //time functions
29 void counter(int n); //contador
30 void delayMs(int n);
31
```

As we can see in the code we have a main_code that is in charge of the interruptions characteristics and calls the main function. Task1 function runs in the main function after the “Hello” and the “Introduce Seconds:”.

```
46 void main_code(void)
47 {
48     int num; //value from the keyboard
49     int pos; //pos for the cont
50     while(1) {
51         lcd.locate(0,0);
52         lcd.printf("Hello");
53         delayMs(5000);
54         lcd.cls();
55         pos = 0;
56         cont = 0;
57         key = '-'; //reset all variables
58         lcd.locate(0,0);
59         lcd.printf("Introduce");
60         lcd.locate(0,1);
61         lcd.printf("Seconds: ");
62         while(key != '#') {
63             delayMs(20); //debounce
64             key = Kpad.ReadKey();
65             num = isA_number(key);
66             delayMs(200);
67             //we do this code only if there is a numerical char
68             if(num != -1 && keypress(key)) { //condition + flag
69                 cont = cont*10 + num;
70                 lcd.locate(8+pos,1);
71                 lcd.printf("%d \n", num); //display the seconds
72                 pos += 1; //get to the next position
73             }
74         }//we get out of the cicle when user press # or *
75         //once here we call our timer
76         counter(cont);
77         task1();
78         //we go back again
79     }
80 }
```

In the counter function we basically run a descendent counter and display the count on the LCD, then it reaches the task1 function where it displays a “Time´s up” messages and activates the buzzer.

```
115 void counter(int n)
116 {
117
118     lcd.cls();
119     lcd.locate(0,0);
120     lcd.printf("Counting...");
121
122     while(n != 0) {
123         if(IRQ == 1) {
124             PORTA_IRQHandler();
125         }
126         lcd.locate(0,1);
127         lcd.printf("%d \n", n);
128         PTD->PCOR = 0x02; //turn on blue led
129         delayMs(800); //stop one second
130         PTD->PSOR |= 0x02; /* turn off blue LED */
131         delayMs(200);
132         n -= 1;
133     }
134 }
```

```
164 void task1(void)
165 {
166     lcd.cls();
167     lcd.locate(0,0);
168     lcd.printf("Time's up");
169     PTB->PCOR = 0x40000; //turn on red led
170     buzzer.period(1/float(c));     // set PWM period
171     buzzer = 0.5;                   // set duty cycle
172     wait(1.0);
173     buzzer.period(1/float(d));     // set PWM period
174     buzzer = 0.5;                   // set duty cycle
175     wait(1.0);
176     PTB->PSOR = 0x40000; // turn off red led
177     lcd.cls();
178     buzzer = 0;
179 }
```

While the counter function is running, we can activate an interruption by pressing the push button. What it does is that it pauses the counter and displays a message on the LCD that says “PAUSED”. To continue with the program you need to press the “*” from the keypad. This interruption is given by the PORTA_IRQHANDLER.

```
146 void PORTA_IRQHandler(void)
147 {
148     delayMs(20);//debounce
149     if(pause != 0) {
150         lcd.cls();
151         lcd.locate(0,0);
152         lcd.printf("PAUSED");
153         while(key != '*' && keypress(key)) {
154             key = Kpad.ReadKey();
155         }
156         if(cont != 0) {
157             lcd.cls();
158             lcd.locate(0,0);
159             lcd.printf("Counting...");
160         }
161     }
162 }
```

Video

Link of the video for a better visualization: https://youtu.be/LBoPdyYF7nM