微處理機系統實習　Lab3

班級：資訊三乙 學號：D1210799 姓名：王建葦

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

Lab 3.1 - Buzzer 與 Keypad 整合：

透過按下 Keypad 的數字鍵，系統會根據輸入數值鳴叫相同次數的蜂鳴器（Buzzer），並於 LED 上以二進位方式顯示該數值。

設計概念為 **輸入（Keypad）—處理（控制邏輯）—輸出（Buzzer 與 LED）** 的流程，練習 GPIO 腳位輸出控制及按鍵掃描。

Lab 3.2 - 7-Segment 雙向跑馬燈顯示 “HOLA”：

以四個七段顯示器顯示 “HOLA” 字樣，並利用 Keypad 控制字串移動方向（← 往左、→ 往右）、暫停（P）、重置（R）。

主要概念為 **狀態機（State Machine）設計** 與 **多工掃描顯示（Multiplexing Display）**，並將重複的動作包裝成函式以提升程式可讀性與維護性。

二、【遭遇的問題】：

What problems you faced during design and implementation?

1. 在 Lab3.1 實作時，按鍵按下後蜂鳴器無法立即反應，且 LED 顯示數值有時會出現錯誤或不穩定閃爍。
2. 在 Lab3.2 中，七段顯示器的段位對應腳位（PE0~PE7）與顯示字型的 pattern 不一致，導致輸出的字母形狀錯誤。
3. 方向鍵控制 “HOLA” 的滾動時，出現方向顛倒或延遲不一致的問題。
4. 在連續顯示四個字元時，若延遲時間設定不當會造成明顯閃爍現象。

三、【解決方法】：

How did you solve the problems?

1. 針對蜂鳴器響應延遲問題，透過 pressed 旗標變數控制「按下—放開」的狀態轉換，確保蜂鳴器動作只在放開按鍵後執行。
2. 為了修正七段顯示器字型錯誤，依照電路圖比對每個 segment（A~G、DOT）對應的 PE 腳位，重新定義 H, O, L, A 的 pattern，例如 H = 0x2A, O = 0x82, L = 0x9B, A = 0x22。
3. 使用模組化函式 Segment_showPattern() 來簡化顯示邏輯，並透過狀態變數 scroll_direction、scrolling 控制字串滾動方向與暫停功能。
4. 加入 CLK_SysTickDelay() 微延遲迴圈控制顯示刷新時間，讓人眼看起來為持續亮顯而非閃爍。

在找尋這些問題的解決方法與問題點時，我有使用 ChatGPT 協助我找尋與解決問題。包含實驗結報的內容修改與潤飾都有使用 ChatGPT 協助。

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

1. 由於七段顯示器之硬體刷新速率與中斷機制未整合，仍有極少數情況會出現閃爍現象，推測與延遲時間設定或硬體干擾有關。
2. 若多次快速按下 Keypad，可能出現去彈跳（debounce）問題導致多次觸發，目前僅透過軟體延遲暫時改善，尚未使用硬體或中斷式去彈跳機制解決。

五、【程式碼】 ：

Lab 3.1:

Q1.c

```c
#include <stdio.h>
#include "NUC100Series.h"
#include "MCU_init.h"
#include "SYS_init.h"
#include "Scankey.h"

void Buzz(int number) {
    int i;
    for (i = 0; i < number; i++) {
        PB11 = 0;
        CLK_SysTickDelay(100000);
        PB11 = 1;
        CLK_SysTickDelay(100000);
    }
}


void Display_binary(int value) {
    switch(value) {
            // 1
            case 1 : PC12=1; PC13=1; PC14=1; PC15=0; break;
            // 2
            case 2 : PC12=1; PC13=1; PC14=0; PC15=1; break;
            // 3
            case 3 : PC12=1; PC13=1; PC14=0; PC15=0; break;
            // 4
            case 4 : PC12=1; PC13=0; PC14=1; PC15=1; break;
            // 5
            case 5 : PC12=1; PC13=0; PC14=1; PC15=0; break;
            // 6
            case 6 : PC12=1; PC13=0; PC14=0; PC15=1; break;
            // 7
            case 7 : PC12=1; PC13=0; PC14=0; PC15=0; break;
            // 8
            case 8 : PC12=0; PC13=1; PC14=1; PC15=1; break;
            // 9
            case 9 : PC12=0; PC13=1; PC14=1; PC15=0; break;
    }
}
```

```c

int main(void) {
    int key, lastKey = 0;
        int pressed = 0;

    SYS_Init();
    OpenKeyPad();


    GPIO_SetMode(PB, BIT11, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PC, BIT12 | BIT13 | BIT14 | BIT15, GPIO_MODE_OUTPUT);

    PB11 = 1;

    while (1) {
        key = ScanKey();

        if (key != 0 && pressed == 0) {
            // ????? ? ?? key
            pressed = key;
        }
        else if (key == 0 && pressed != 0) {
            // ????? ? ????
            Display_binary(pressed);
            Buzz(pressed);
            pressed = 0; // reset
        }
    }
}
```

Lab 3.2:

```
15  void Display_7seg(uint16_t value)
16  {
17    uint8_t digit;
18      digit = value / 1000;
19      CloseSevenSegment();
20      ShowSevenSegment(3,digit);
21      CLK_SysTickDelay(5000);
22
23      value = value - digit * 1000;
24      digit = value / 100;
25      CloseSevenSegment();
26      ShowSevenSegment(2,digit);
27      CLK_SysTickDelay(5000);
28
29      value = value - digit * 100;
30      digit = value / 10;
31      CloseSevenSegment();
32      ShowSevenSegment(1,digit);
33      CLK_SysTickDelay(5000);
34
35      value = value - digit * 10;
36      digit = value;
37      CloseSevenSegment();
38      ShowSevenSegment(0,digit);
39      CLK_SysTickDelay(5000);
40  }
```

```c
void Segment_showPattern(int i, unsigned char pattern)
{
    CloseSevenSegment();
    // Set the pattern directly to the specified segment
    if (i >= 0 && i <= 3) {
        uint8_t temp, j;
        temp = pattern;

        // Set segment pattern bits (PE0-PE7) based on the library logic
        for (j = 0; j < 8; j++) {
            if ((temp & 0x01) == 0x01) {
                switch (j) {
                    case 0: PE0 = 1; break;
                    case 1: PE1 = 1; break;
                    case 2: PE2 = 1; break;
                    case 3: PE3 = 1; break;
                    case 4: PE4 = 1; break;
                    case 5: PE5 = 1; break;
                    case 6: PE6 = 1; break;
                    case 7: PE7 = 1; break;
                }
            } else {
                switch (j) {
                    case 0: PE0 = 0; break;
                    case 1: PE1 = 0; break;
                    case 2: PE2 = 0; break;
                    case 3: PE3 = 0; break;
                    case 4: PE4 = 0; break;
                    case 5: PE5 = 0; break;
                    case 6: PE6 = 0; break;
                    case 7: PE7 = 0; break;
                }
            }
            temp = temp >> 1;
        }

        // Enable the specific display position
        switch (i) {
            case 0: PC4 = 1; break;
            case 1: PC5 = 1; break;
            case 2: PC6 = 1; break;
            case 3: PC7 = 1; break;
        }
    }
}
```

```c
int main(void)
{
    int k=0;
    int j=0;

    unsigned char hola_patterns[4] = {0x2A, 0x82, 0x9B, 0x22}; // H, O, L, A patterns
    int hola_position = 0; // Current starting position of HOLA (0-3)
    int scrolling = 0; // 1 = scrolling, 0 = paused (default: paused)
    int scroll_direction = 1; // 1 = right, -1 = left
    unsigned int scroll_counter = 0;

    SYS_Init();

    OpenSevenSegment(); // for 7-segment
      OpenKeyPad();       // for keypad

      while(1) {
          k=ScanKey();

          // Handle HOLA scrolling controls
          if (k == 4) { // Right scroll - HOLA -> AHOL -> LAHO -> OLAH
              if (scrolling == 0) { // First time pressed or after pause
                  hola_position = (hola_position + 1) % 4; // Move immediately
              }
              scroll_direction = 1;
              scrolling = 1;
              scroll_counter = 0; // Reset counter for next scroll
          } else if (k == 6) { // Left scroll - HOLA -> OLAH -> LAHO -> AHOL
              if (scrolling == 0) { // First time pressed or after pause
                  hola_position = (hola_position - 1 + 4) % 4; // Move immediately
              }
              scroll_direction = -1;
              scrolling = 1;
              scroll_counter = 0; // Reset counter for next scroll
          } else if (k == 5) { // Pause
              scrolling = 0;
          } else if (k == 8) { // Reset to default HOLA
              hola_position = 0;
              scrolling = 0; // Reset to paused state
              scroll_direction = 1;
              scroll_counter = 0; // Reset counter
          }

```

```
131
132             // Display HOLA on 7-segment displays (one at a time for multiplexing)
133             for (j = 0; j < 4; j++) {
134                 int char_index = (hola_position + (3 - j)) % 4;
135                 uint8_t pattern = hola_patterns[char_index];
136
137                 CloseSevenSegment();
138
139                 // Set pattern for HOLA character directly using PE pins
140                 // Actual mapping: PE7=G, PE6=E, PE5=D, PE4=B, PE3=A, PE2=F, PE1=DOT, PE0=C
141                 PE0 = (pattern & 0x01) ? 1 : 0;  // C segment
142                 PE1 = (pattern & 0x02) ? 1 : 0;  // DOT segment
143                 PE2 = (pattern & 0x04) ? 1 : 0;  // F segment
144                 PE3 = (pattern & 0x08) ? 1 : 0;  // A segment
145                 PE4 = (pattern & 0x10) ? 1 : 0;  // B segment
146                 PE5 = (pattern & 0x20) ? 1 : 0;  // D segment
147                 PE6 = (pattern & 0x40) ? 1 : 0;  // E segment
148                 PE7 = (pattern & 0x80) ? 1 : 0;  // G segment
149
150                 // Enable display position
151                 switch(j) {
152                     case 0: PC4 = 1; break;
153                     case 1: PC5 = 1; break;
154                     case 2: PC6 = 1; break;
155                     case 3: PC7 = 1; break;
156                 }
157
158                 CLK_SysTickDelay(5000);  // Display time for each segment
159             }
160
161             // Auto-scroll HOLA every 1 second if scrolling is enabled
162             if (scrolling) {
163                 scroll_counter++;
164                 if (scroll_counter >= 45) { // Approximately 1 second (45 cycles ?  ~22ms/cycle)
165                     scroll_counter = 0;
166                     hola_position = (hola_position + scroll_direction + 4) % 4;
167                 }
168             }
169
170             CLK_SysTickDelay(1000); // Small delay for main loop
171         }
172 }
```