實驗六結報： STM32 Keypad Scanning

0316323 薛世恩 0316213 蒲郁文

實驗目的：

1.了解 STM32 使用原理

2.了解如何使用 C code 控制 STM32

3.設計 7-Seg LED 和 Keypad 程式

實驗步驟：

(p1.c)

```
// these
functions
are
inside
the asm
file
```

```
extern void gpio_init();
extern void max7219_init();
extern void max7219_send(unsigned char address, unsigned char data);



/**
 * show data on 7-seg via max7219_send
 *
 * input:
 *   data: decimal value
 *   num_digs: number of digits to show on 7-seg
 *
 * return:
 *   0: success
 *   -1: illegal data range (out of 8 digits)
 */
```

```c
int display(int data, int num_digs)
{
     int data2 = data, i;
     for (i = 1; i <= num_digs; i++)
     {
          if (data % 10 < 0)
               max7219_send(i, -data % 10);
          else
               max7219_send(i, data % 10);
          data /= 10;
     }
     if (data2 < 0)
          max7219_send(num_digs, 10);
     for ( ; i <= 8; i++)
          max7219_send(i, 15);
     return (data > 99999999 || data < -9999999) ? -1 : 0;
}


int main()
{
     int student_id = 316323;
     gpio_init();
     max7219_init();
     display(student_id, 8);
}
```

(p1.s)

```
     .syntax
unified
```

```
     .cpu cortex-m4
     .thumb


.text
     .global max7219_init
     .global max7219_send
     .global gpio_init
```

```
        .equ RCC_AHB2ENR,  0x4002104C


        .equ DECODE_MODE,  0x09
        .equ DISPLAY_TEST, 0x0F
        .equ SCAN_LIMIT,   0x0B
        .equ INTENSITY,    0x0A
        .equ SHUTDOWN,     0x0C


        .equ MAX7219_DIN,  0x20 @ PA5
        .equ MAX7219_CS,   0x40 @ PA6
        .equ MAX7219_CLK,  0x80 @ PA7


        .equ GPIOA_BASE,   0x48000000
        .equ BSRR_OFFSET,  0x18 @ set bit
        .equ BRR_OFFSET,   0x28 @ clear bit


gpio_init:
        push {r0, r1, r2, lr}


        mov  r0, 0b00000000000000000000000000000001
        ldr  r1, =RCC_AHB2ENR
        str  r0, [r1]


        ldr  r1, =GPIOA_BASE @ GPIOA_MODER
        ldr  r2, [r1]
        and  r2, 0b11111111111111110000001111111111
        orr  r2, 0b00000000000000000101010000000000
        str  r2, [r1]
```

```
    add  r1, 0x4 @ GPIOA_OTYPER
    ldr  r2, [r1]
    and  r2, 0b11111111111111111111111100011111
    str  r2, [r1]

    add  r1, 0x4 @ GPIOA_SPEEDER
    ldr  r2, [r1]
    and  r2, 0b11111111111111110000001111111111
    orr  r2, 0b00000000000000000101010000000000
    str  r2, [r1]

    pop  {r0, r1, r2, pc}

max7219_init:
    push {r0, r1, lr}

    ldr  r0, =DECODE_MODE
    ldr  r1, =0xFF
    bl   max7219_send

    ldr  r0, =DISPLAY_TEST
    ldr  r1, =0x0
    bl   max7219_send

    ldr  r0, =SCAN_LIMIT
    ldr  r1, =0x7
    bl   max7219_send

    ldr  r0, =INTENSITY
    ldr  r1, =0xA
    bl   max7219_send
```

```
    ldr  r0, =SHUTDOWN
    ldr  r1, =0x1
    bl   max7219_send


    pop  {r0, r1, pc}


max7219_send:
    @ input parameter: r0 is ADDRESS , r1 is DATA
    push {r0, r1, r2, r3, r4, r5, r6, r7, r8, lr}
    lsl  r0, r0, 0x8
    add  r0, r1
    ldr  r1, =GPIOA_BASE
    ldr  r2, =MAX7219_CS
    ldr  r3, =MAX7219_DIN
    ldr  r4, =MAX7219_CLK
    ldr  r5, =BSRR_OFFSET
    ldr  r6, =BRR_OFFSET
    ldr  r7, =0x0F @ currently sending r7-th bit


max7219_send_loop:
    mov  r8, #0x1
    lsl  r8, r8, r7
    str  r4, [r1, r6] @ clk -> 0
    tst  r0, r8 @ ANDS but discard result
    beq  max7219_send_clear_bit
    str  r3, [r1, r5] @ din -> 1
    b    max7219_send_check_done


max7219_send_clear_bit:
    str  r3, [r1, r6] @ din -> 0
```

```
max7219_send_check_done:
     str  r4, [r1, r5] @ clk -> 1
     subs r7, #0x1
     bge  max7219_send_loop
     str  r2, [r1, r6] @ cs -> 0
     str  r2, [r1, r5] @ cs -> 1
     pop  {r0, r1, r2, r3, r4, r5, r6, r7, r8, pc}
```

方法：

此題跟之前作業幾乎一模一樣，只是熟悉參數傳遞而已(reg 要對到)，並無太大差異，只是要將 C code 跟 Assembly 分開來寫，然後用 C 呼叫 assembly 然後 push 後做完事情再 pop 然後 pop pc 回到原本 C code 該在之位置。重要的是 assembly 中函式要宣告 global 然後 C 那邊要 extern 作連結。

(p2.c)

```
#include
"stm32l476xx.h"
#include "utils.h"



#define XPORT GPIOC
#define YPORT GPIOB
#define X0 GPIO_Pin_0
#define X1 GPIO_Pin_1
#define X2 GPIO_Pin_2
#define X3 GPIO_Pin_3
#define Y0 GPIO_Pin_6
#define Y1 GPIO_Pin_5
```

```
#define Y2 GPIO_Pin_4
#define Y3 GPIO_Pin_3



unsigned int x_pin = {X0, X1, X2, X3};
unsigned int y_pin = {Y0, Y1, Y2, Y3};



// initial keypad gpio pin, X as output and Y as input
void keypad_init()
{
     RCC->AHB2ENR   |= 0b00000000000000000000000000000110;



     GPIOC->MODER   &= 0b11111111111111111111111100000000;
     GPIOC->MODER   |= 0b00000000000000000000000001010101;
     GPIOC->PUPDR   &= 0b11111111111111111111111100000000;
     GPIOC->PUPDR   |= 0b00000000000000000000000001010101;
     GPIOC->OSPEEDR &= 0b11111111111111111111111100000000;
     GPIOC->OSPEEDR |= 0b00000000000000000000000001010101;
     GPIOC->ODR     |= 0b00000000000000000000000000001111;



     GPIOB->MODER   &= 0b11111111111111111100000000111111;
     GPIOB->PUPDR   &= 0b11111111111111111100000000111111;
     GPIOB->PUPDR   |= 0b00000000000000000010101010000000;
     GPIOB->OSPEEDR &= 0b11111111111111111111111100000000;
     GPIOB->OSPEEDR |= 0b00000000000000000000000001010101;
}



/**
 * scan keypad value
 *
 * return:
 *   >=0: key pressed value
 *   -1: no key press
 */
```

```
signed char keypad_scan()
{
      XPORT->BSRR = X0;
      XPORT->BRR  = X1;
      XPORT->BRR  = X2;
      XPORT->BRR  = X3;


      if (GPIO_ReadInputDataBit(YPORT, Y0))
           return 15;
      if (GPIO_ReadInputDataBit(YPORT, Y1))
           return 7;
      if (GPIO_ReadInputDataBit(YPORT, Y2))
           return 4;
      if (GPIO_ReadInputDataBit(YPORT, Y3))
           return 1;


      XPORT->BRR  = X0;
      XPORT->BSRR = X1;
      XPORT->BRR  = X2;
      XPORT->BRR  = X3;


      if (GPIO_ReadInputDataBit(YPORT, Y0))
           return 0;
      if (GPIO_ReadInputDataBit(YPORT, Y1))
           return 8;
      if (GPIO_ReadInputDataBit(YPORT, Y2))
           return 5;
      if (GPIO_ReadInputDataBit(YPORT, Y3))
           return 2;


      XPORT->BRR  = X0;
      XPORT->BRR  = X1;
      XPORT->BSRR = X2;
      XPORT->BRR  = X3;
```

```
    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 14;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 9;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 6;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 3;


    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BSRR = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 13;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 12;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 11;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 10;


    return -1;
}


int main()
{
    gpio_init();
    max7219_init();
    keypad_init();
    while (1)
```

```
    {
        int input = keypad_scan();
        if (input >= 10)
            display(input, 2);
        else if (input >= 0)
            display(input, 1);
        else
            display(input, 0);
    }
}
```

方法：

首先這題是在做 keyboard 的輸入輸出，跟電路問題，主要是將 pin 對好，然後做出 definition，當按下去時判斷被按下去的按鈕，然後只有在被按的時候才顯示，沒被按則不顯示，然後 display 部分則沿用上一題之 display。

(p3.c)

```
#include
"stm32l476xx.h"
#include "utils.h"


#define XPORT GPIOC
#define YPORT GPIOB
#define X0 GPIO_Pin_0
#define X1 GPIO_Pin_1
#define X2 GPIO_Pin_2
#define X3 GPIO_Pin_3
#define Y0 GPIO_Pin_6
```

```
#define Y1 GPIO_Pin_5
#define Y2 GPIO_Pin_4
#define Y3 GPIO_Pin_3



unsigned int x_pin = {X0, X1, X2, X3};
unsigned int y_pin = {Y0, Y1, Y2, Y3};
unsigned int total, len;
char set[14];
int rem=0;



void set_clear()
{
     for (int i = 0; i < 14; i++)
          set[i] = 0;
}



void set_insert(int i)
{
     if (i >= 0 && i < 14)
          set[i] = 1;
}



int set_reduce()
{
     int sum = 0;
     for (int i = 0; i < 14; i++)
          if (set[i])
               sum += i;
     return sum;
}



// initial keypad gpio pin, X as output and Y as input
void keypad_init()
```

```
{
    RCC->AHB2ENR   |= 0b00000000000000000000000000000110;


    GPIOC->MODER   &= 0b11111111111111111111111100000000;
    GPIOC->MODER   |= 0b00000000000000000000000001010101;
    GPIOC->PUPDR   &= 0b11111111111111111111111100000000;
    GPIOC->PUPDR   |= 0b00000000000000000000000001010101;
    GPIOC->OSPEEDR &= 0b11111111111111111111111100000000;
    GPIOC->OSPEEDR |= 0b00000000000000000000000001010101;
    GPIOC->ODR     |= 0b00000000000000000000000000001111;


    GPIOB->MODER   &= 0b11111111111111111100000000111111;
    GPIOB->PUPDR   &= 0b11111111111111111100000000111111;
    GPIOB->PUPDR   |= 0b00000000000000000010101010000000;
    GPIOB->OSPEEDR &= 0b11111111111111111111111100000000;
    GPIOB->OSPEEDR |= 0b00000000000000000000000001010101;
}



/**
 * scan keypad value
 *
 * return:
 *   >=0: key pressed value
 *   -1: no key press
 */
signed char keypad_scan()
{
    XPORT->BSRR = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 15;
```

```
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 7;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 4;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 1;


    XPORT->BRR  = X0;
    XPORT->BSRR = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 0;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 8;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 5;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 2;


    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BSRR = X2;
    XPORT->BRR  = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 14;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 9;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 6;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 3;
```

```
	XPORT->BRR  = X0;
	XPORT->BRR  = X1;
	XPORT->BRR  = X2;
	XPORT->BSRR = X3;


	if (GPIO_ReadInputDataBit(YPORT, Y0))
		return 13;
	if (GPIO_ReadInputDataBit(YPORT, Y1))
		return 12;
	if (GPIO_ReadInputDataBit(YPORT, Y2))
		return 11;
	if (GPIO_ReadInputDataBit(YPORT, Y3))
		return 10;


	return -1;
}


int main()
{
	gpio_init();
	max7219_init();
	keypad_init();
	total = 0;
	len = 0;
	int cnt=0;
	int input = -1, prev_input = -1;


	set_clear();


	while (1)
	{
```

```
            prev_input = input;
            input = keypad_scan();
            if(input==0)
                 rem=1;
            if (input == prev_input)
            {
                 cnt++;
                 if(cnt>12000)
                 {
                      cnt=0;
                      if(input<14)
                      goto A;
                 }
            }
            else if (input >= 14)
            {
                 total = 0;
                 len = 0;
                 set_clear();
                 display(total, len);
            }
            else if (input != -1)
            {
                 set_insert(input);
            }
            else
            {
                 input = set_reduce();
                 A:
                 set_clear();
                 if (input >= 10 && len + 2 <= 8)
                 {
                      total = total * 100 + input;
                      len += 2;
                 }
                 else if (input < 10 && input >= 0 && len + 1 <=
8)
                 {
```

```
                    if(input==0&&rem==0);
                    else
                    {
                        total = total * 10 + input;
                        len += 1;
                        rem=0;
                    }
                }
                display(total, len);
            }
        }
}
```

方法：

主要是在顯示數字跟多按鍵還有長按短按做處理，多按鍵時，我們用一陣列去儲存到底哪些按鍵在其他按鍵被按下去時同時也被按下去，然後再將其加總輸出做為 input，在處理短按時則是在時間內(有一個 cnt 被按住時會一直加直到某值他就會連續顯示)放開算短按一次，長按則時則是 cnt 一直往上加，直到某值時算是長按然後連續顯示並且 cnt 歸 0，我們實作數字則是用 total 並控制他的運算然後將其 display。

(p4.c)

```
#include
"stm32l476xx.h"
#include "utils.h"
```

```
#define XPORT GPIOC
#define YPORT GPIOB
#define X0 GPIO_Pin_0
#define X1 GPIO_Pin_1
#define X2 GPIO_Pin_2
#define X3 GPIO_Pin_3
#define Y0 GPIO_Pin_6
#define Y1 GPIO_Pin_5
#define Y2 GPIO_Pin_4
#define Y3 GPIO_Pin_3


unsigned int x_pin = {X0, X1, X2, X3};
unsigned int y_pin = {Y0, Y1, Y2, Y3};
float total;
int len;
char set[14];
int rem = 0;
int prev_is_num = 0;
int error = 0;


void set_clear()
{
     for (int i = 0; i < 14; i++)
          set[i] = 0;
}


void set_insert(int i)
{
     if (i >= 0 && i < 14)
          set[i] = 1;
}


int set_reduce()
```

```
{
	int sum = 0;
	for (int i = 0; i < 14; i++)
		if (set[i])
			sum += i;
	return sum;
}



int cal_len(int a)
{
	int sum = 0;
	while (a > 0)
	{
		a /= 10;
		sum++;
	}
	return sum;
}



// initial keypad gpio pin, X as output and Y as input
void keypad_init()
{
	RCC->AHB2ENR   |= 0b00000000000000000000000000000110;



	GPIOC->MODER   &= 0b11111111111111111111111100000000;
	GPIOC->MODER   |= 0b00000000000000000000000001010101;
	GPIOC->PUPDR   &= 0b11111111111111111111111100000000;
	GPIOC->PUPDR   |= 0b00000000000000000000000001010101;
	GPIOC->OSPEEDR &= 0b11111111111111111111111100000000;
	GPIOC->OSPEEDR |= 0b00000000000000000000000001010101;
	GPIOC->ODR     |= 0b00000000000000000000000000001111;



	GPIOB->MODER   &= 0b11111111111111111100000000111111;
	GPIOB->PUPDR   &= 0b11111111111111111100000000111111;
```

```
    GPIOB->PUPDR   |= 0b00000000000000000010101010000000;
    GPIOB->OSPEEDR &= 0b11111111111111111111111100000000;
    GPIOB->OSPEEDR |= 0b00000000000000000000000001010101;
}



/**
 * scan keypad value
 *
 * return:
 *   >=0: key pressed value
 *   -1: no key press
 */
signed char keypad_scan()
{
    XPORT->BSRR = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;



    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 15;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 7;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 4;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 1;



    XPORT->BRR  = X0;
    XPORT->BSRR = X1;
    XPORT->BRR  = X2;
    XPORT->BRR  = X3;



    if (GPIO_ReadInputDataBit(YPORT, Y0))
```

```
        return 0;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 8;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 5;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 2;


    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BSRR = X2;
    XPORT->BRR  = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 14;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 9;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 6;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
        return 3;


    XPORT->BRR  = X0;
    XPORT->BRR  = X1;
    XPORT->BRR  = X2;
    XPORT->BSRR = X3;


    if (GPIO_ReadInputDataBit(YPORT, Y0))
        return 13;
    if (GPIO_ReadInputDataBit(YPORT, Y1))
        return 12;
    if (GPIO_ReadInputDataBit(YPORT, Y2))
        return 11;
    if (GPIO_ReadInputDataBit(YPORT, Y3))
```

```
          return 10;



     return -1;
}



float stack_num[100];
int stack_ope[100];
int top_num = -1;
int top_ope = -1;



int main()
{
     asm("\
          LDR.W R0, =0xE000ED88      ;\
          LDR   R1, [R0]             ;\
          ORR   R1, R1, #(0xF << 20) ;\
          STR   R1, [R0]             ;\
          DSB                        ;\
          ISB                        ;\
     ");
     gpio_init();
     max7219_init();
     keypad_init();
     total = 0;
     len = 0;
     int input = -1, prev_input = -1;
     set_clear();
     display(0, 0);
     while (1)
     {
          prev_input = input;
          input = keypad_scan();
          if(input == 0)
               rem = 1;
          if (input == prev_input);
```

```
else if (input == 14) // Clear
{
     for (int i = 0; i <= 99; i++)
     {
          stack_num[i] = 0;
          stack_ope[i] = 0;
     }
     top_num = -1;
     top_ope = -1;
     total = 0;
     len = 0;
     set_clear();
     prev_is_num = 0;
     error = 0;
     display(total, len);
}
else if (input == 15) // =
{
     error = prev_is_num ? error : 1;
     top_num++;
     stack_num[top_num] = total;
     while (top_ope != -1)
     {
          int operator;
          operator = stack_ope[top_ope];
          top_ope--;
          if (operator == 10) // +
          {
               float temp1, temp2;
               temp2 = stack_num[top_num];
               top_num--;
               temp1 = stack_num[top_num];
               // top_num--;
               // top_num++;
               stack_num[top_num] = temp1 + temp2;
          }
          else if (operator == 11) // -
          {
```

```
                    float temp1, temp2;
                    temp2 = stack_num[top_num];
                    top_num--;
                    temp1 = stack_num[top_num];
                    // top_num--;
                    // top_num++;
                    stack_num[top_num] = temp1 - temp2;
               }
               else if (operator == 12) // *
               {
                    float temp1, temp2;
                    temp2 = stack_num[top_num];
                    top_num--;
                    temp1 = stack_num[top_num];
                    // top_num--;
                    // top_num++;
                    stack_num[top_num] = temp1 * temp2;
               }
               else if (operator == 13) // /
               {
                    float temp1, temp2;
                    temp2 = stack_num[top_num];
                    top_num--;
                    temp1 = stack_num[top_num];
                    // top_num--;
                    // top_num++;
                    stack_num[top_num] = temp1 / temp2;
               }
          }
          total = error ? -1 : stack_num[top_num];
     }
     else if (input == 10) // +
     {
          top_num++;
          stack_num[top_num] = total;
          total = 0;
          len = 0;
          while (stack_ope[top_ope] == 12) // *
```

```
        {
            float temp1, temp2;
            temp2 = stack_num[top_num];
            top_num--;
            temp1 = stack_num[top_num];
            // top_num--;
            // top_num++;
            top_ope--;
            stack_num[top_num] = temp1 * temp2;
        }
        while (stack_ope[top_ope] == 13) // /
        {
            float temp1, temp2;
            temp2 = stack_num[top_num];
            top_num--;
            temp1 = stack_num[top_num];
            // top_num--;
            // top_num++;
            top_ope--;
            stack_num[top_num] = temp1 / temp2;
        }
        top_ope++;
        stack_ope[top_ope] = 10; // +
        error = prev_is_num ? error : 1;
        prev_is_num = 0;
    }
    else if (input == 11) // -
    {
        top_num++;
        stack_num[top_num] = total;
        total = 0;
        len = 0;
        while (stack_ope[top_ope] == 12) // *
        {
            float temp1, temp2;
            temp2 = stack_num[top_num];
            top_num--;
            temp1 = stack_num[top_num];
```

```
            // top_num--;
            // top_num++;
            top_ope--;
            stack_num[top_num] = temp1 * temp2;
        }
        while (stack_ope[top_ope] == 13) // /
        {
            float temp1, temp2;
            temp2 = stack_num[top_num];
            top_num--;
            temp1 = stack_num[top_num];
            // top_num--;
            // top_num++;
            top_ope--;
            stack_num[top_num] = temp1 / temp2;
        }
        top_ope++;
        stack_ope[top_ope] = 11; // -
        error = prev_is_num ? error : 1;
        prev_is_num = 0;
    }
    else if (input == 12) // *
    {
        top_num++;
        top_ope++;
        stack_num[top_num] = total;
        total = 0;
        len = 0;
        stack_ope[top_ope] = 12; // *
        error = prev_is_num ? error : 1;
        prev_is_num = 0;
    }
    else if (input == 13) // /
    {
        top_num++;
        top_ope++;
        stack_num[top_num] = total;
        total = 0;
```

```
                len = 0;
                stack_ope[top_ope] = 13; // /
                error = prev_is_num ? error : 1;
                prev_is_num = 0;
           }
           else if (input != -1)
           {
                set_insert(input);
           }
           else
           {
                input = set_reduce();
                set_clear();
                if ((input < 10 && input >= 1) || (!input &&
rem))
                     prev_is_num = 1;
                if (input >= 10 && len + 2 <= 3)
                {
                     total = total * 100 + input;
                     len += 2;
                }
                else if (input < 10 && input >= 0 && len + 1 <=
3)
                {
                     if (input == 0 && rem == 0);
                     else
                     {
                          total = total * 10 + input;
                          len += 1;
                          rem = 0;
                     }
                }
                if (total > 0)
                     len = cal_len(total);
                else if (total < 0)
                     len = cal_len(-total) + 1;
                displayf(total, len);
           }
```

```
        }
    }
```

(utils.h)

```
#ifndef
UTILS_H_
#define UTILS_H_



// these functions are inside the asm file
extern void gpio_init();
extern void max7219_init();
extern void max7219_send(unsigned char address, unsigned char data);



/**
 * show data on 7-seg via max7219_send
 *
 * input:
 *   data: decimal value
 *   num_digs: number of digits to show on 7-seg
 *
 * return:
 *   0: success
 *   -1: illegal data range (out of 8 digits)
 */
int display(int data, int num_digs)
{
    int show_dec_pt = 0;
    if (num_digs <= -1000)
    {
        num_digs = -1000 - num_digs ;
        show_dec_pt = 1;
    }
    num_digs = num_digs > 8 ? 8 : num_digs;
    int data2 = data, i;
    for (i = 1; i <= num_digs; i++)
    {
```

```
            if (data2 < 0 && i == num_digs);
            else if (show_dec_pt && i == 3 && data % 10 < 0)
                max7219_send(i, -data % 10 | 0b10000000);
            else if (show_dec_pt && i == 3)
                max7219_send(i, data % 10 | 0b10000000);
            else if (data % 10 < 0)
                max7219_send(i, -data % 10);
            else
                max7219_send(i, data % 10);
            data /= 10;
        }
        if (data2 < 0)
            max7219_send(num_digs, 10);
        for ( ; i <= 8; i++)
            max7219_send(i, 15);
        return (data > 99999999 || data < -9999999) ? -1 : 0;
}



#define GPIO_Pin_0  0b0000000000000001
#define GPIO_Pin_1  0b0000000000000010
#define GPIO_Pin_2  0b0000000000000100
#define GPIO_Pin_3  0b0000000000001000
#define GPIO_Pin_4  0b0000000000010000
#define GPIO_Pin_5  0b0000000000100000
#define GPIO_Pin_6  0b0000000001000000
#define GPIO_Pin_7  0b0000000010000000
#define GPIO_Pin_8  0b0000000100000000
#define GPIO_Pin_9  0b0000001000000000
#define GPIO_Pin_10 0b0000010000000000
#define GPIO_Pin_11 0b0000100000000000
#define GPIO_Pin_12 0b0001000000000000
#define GPIO_Pin_13 0b0010000000000000
#define GPIO_Pin_14 0b0100000000000000
#define GPIO_Pin_15 0b1000000000000000



int GPIO_ReadInputDataBit(GPIO_TypeDef *a, uint16_t b) {
```

```
        return a->IDR & b;
}



int displayf(float data, int num_digs)
{
        if (num_digs > 8)
                return display(-1, 2);
        if ((int) (data * 100) % 100)
                return display(data * 100, -1002 - num_digs);
        else
                return display(data, num_digs);
}



#endif /* UTILS_H_ */
```

方法：將數字用中序(2 個 stack 一個存數字一個存運算子)然後將數字做計算，如果七段顯示器無法顯示或連續兩個運算子或是最後一個是運算子，我們就讓它顯示-1，然後在做中序的時候要注意運算子優先度問題，否則結果會錯誤，然後為了用浮點數要 enable FPU 然後修改 display 成可以顯示浮點數的型態，並且讓他在負數的時候也可以顯示。

心得與感想：LAB 真的一次比一次難，幾乎都要熬夜兩天才做得出來，然後 bug 越來越多，然後條件一直增加，實在是有點辛苦，又很難猜中到底實驗要我們做的條件是什麼，例如這次長按短按，問了 10 個人吧，沒有人能百分之百確定到底是要我們做出什麼功能，每個人猜想的都不一樣，題目描述希望能夠更詳細一些，demo 影片也可以指出更多應該注意之地方，不過其實還滿有成就感的，而且可以用 C 語言真的是比較輕鬆，終於不用每天想到底哪個 reg 是什麼東西，然後可以自由的用變數，不用 ldr 又 str，然後 mov 來 mov 去，真的太開心了！