

微處理機 LAB 6 Interrupt

Due : 兩週後 早上 8:00

PART 1. (10%) 實作題

Lab 6.1.1 Timer interrupt – Change LED sparkling rate:

```
1 #include <stdio.h>
2 #include "stm321476xx.h"
3 #include "helper_functions.h"
4 #include "7seg.h"
5 #include "keypad.h"
6 #include "led_button.h"
7 #include "timer.h"
8
9 // Define pins for led (default use on-board led PA5)
10 #define LED_gpio GPIOA
11 #define LED_pin 5
12
13 // Define pins for button (default use on-board button PC13)
14 #define BUTTON_gpio GPIOC
15 #define BUTTON_pin 13
16
17 // Define Counter timer
18 #define COUNTER_timer TIM2
19
20 void SysTick_Handler() {
21     if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
22         // Toggle LED display
23         toggle_output(LED_gpio, LED_pin);
24     }
25 }
26
27 int main()
28 {
29     // Cause we want to use floating points we need to init FPU
30     FPU_init();
31     if(init_led(LED_gpio, LED_pin) != 0){
32         // Fail to init led
33         return -1;
34     }
35     if(init_button(BUTTON_gpio, BUTTON_pin) != 0){
36         // Fail to init button
37         return -1;
38     }
39     // Configure SysTick clk to 10 Mhz and interrupt every 0.5s
40     SystemClock_Config_Interrupt(10, 5000000);
41
42     // button_press_cycle_per_second (How many button press segments in a second)
43     int button_press_cycle_per_second = 20;
44     // Use to state how many cycles to check per button_press_cycle
45     int debounce_cycles = 50;
46     // Use to state the threshold when we consider a button press
47     int debounce_threshold = debounce_cycles*0.7;
48     // Used to implement negative edge trigger 0=not-presses 1=pressed
49     int last_button_state=0;
50
51     int blink_frequencies[] = {5000000, 1000000, 3000000};
52     int num_blink_frequencies = sizeof(blink_frequencies) / sizeof(blink_frequencies[0]);
53     int current_blink_frequency_index = 0;
54     int blink_frequency = blink_frequencies[1];
55     while(1) {
56         for(int a=0;a<button_press_cycle_per_second;a++){
57             // Simple Debounce without interrupt
58             int pos_cnt=0;
59             for(int a=0;a<debounce_cycles;a++){
60                 // If button press add count
61                 if(read_gpio(BUTTON_gpio, BUTTON_pin)==0){
62                     pos_cnt++;
63                 }
64                 delay_without_interrupt(1000/(button_press_cycle_per_second*debounce_cycles));
65             }
66             // Check if need to change state
67             if(pos_cnt>debounce_threshold){
68                 if(last_button_state==0){
69                     // Pressed button - Pos edge
70                     current_blink_frequency_index = (current_blink_frequency_index + 1) % num_blink_frequencies;
71                     blink_frequency = blink_frequencies[current_blink_frequency_index];
72                     SystemClock_Config_Interrupt(10, blink_frequency);
73                 }
74             }
75         }
76     }
```

```

75         else{
76             // Pressed button - Continued pressing
77             // Do nothing
78         }
79         last_button_state = 1;
80     }
81     else{
82         if(last_button_state==0){
83             // Released button - Not pressing
84             // Do nothing
85         }
86         else{
87             // Released button - Neg edge
88             // Do nothing
89         }
90         last_button_state = 0;
91     }
92 }
93 }
94 while(1);
95 return 0;
96 }

```

PART 2. (40%) 實作題

Lab6.2: Keypad external interrupt

```

1  #include <stdio.h>
2  #include "stm321476xx.h"
3  #include "helper_functions.h"
4  #include "7seg.h"
5  #include "keypad.h"
6  #include "led_button.h"
7  #include "timer.h"
8
9  // Define Counter timer
10 #define COUNTER_timer TIM2
11
12 // Define pins for 7seg
13 #define SEG_gpio GPIOC
14 #define DIN_pin 3
15 #define CS_pin 2
16 #define CLK_pin 1
17
18 // Define pins for keypad
19 #define COL_gpio GPIOA
20 #define COL_pin 6 // 6 7 8 9
21 #define ROW_gpio GPIOB
22 #define ROW_pin 3 // 3 4 5 6
23
24 int now_col = 3;
25 int keyCnt = 0, keyValue = -1;
26
27 void SysTick_Handler() {
28     if(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk){
29         reset_push(COL_gpio, now_col+COL_pin);
30         now_col = (now_col+1)%4;
31         set_push(COL_gpio, now_col+COL_pin);
32     }
33 }
34
35 void EXTIKeyPad_Handler(int r){
36     int nowKey = keypad[r][(now_col+3)%4];
37     if(nowKey == keyValue){
38         keyCnt++;
39     }else{
40         keyCnt = 0;
41     }

```

```

41     }
42     keyValue = nowKey;
43     if(keyCnt >=5){
44         keyCnt = 5;
45         display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, keyValue, 2);
46     }
47 }
48
49 void EXTI_Setup(){
50     // Enable SYSCFG clock
51     RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
52     // Select output bits
53     SYSCFG->EXTICR[0] &= ~SYSCFG_EXTICR1_EXTI3_Msk;
54     SYSCFG->EXTICR[0] |= (1 << SYSCFG_EXTICR1_EXTI3_Pos);
55     SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI4_Msk;
56     SYSCFG->EXTICR[1] |= (1 << SYSCFG_EXTICR2_EXTI4_Pos);
57     SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI5_Msk;
58     SYSCFG->EXTICR[1] |= (1 << SYSCFG_EXTICR2_EXTI5_Pos);
59     SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI6_Msk;
60     SYSCFG->EXTICR[1] |= (1 << SYSCFG_EXTICR2_EXTI6_Pos);
61
62     // Enable interrupt
63     EXTI->IMR1 |= EXTI_IMR1_IM3;
64     EXTI->IMR1 |= EXTI_IMR1_IM4;
65     EXTI->IMR1 |= EXTI_IMR1_IM5;
66     EXTI->IMR1 |= EXTI_IMR1_IM6;
67
68     // Enable Falling Edge
69     EXTI->FTSR1 |= EXTI_FTSR1_FT3;
70     EXTI->FTSR1 |= EXTI_FTSR1_FT4;
71     EXTI->FTSR1 |= EXTI_FTSR1_FT5;
72     EXTI->FTSR1 |= EXTI_FTSR1_FT6;
73
74     // Enable NVIC
75     NVIC_EnableIRQ(EXTI3_IRQn);
76     NVIC_EnableIRQ(EXTI4_IRQn);
77     NVIC_EnableIRQ(EXTI9_5_IRQn);
78 }
79
80 void EXTI3_IRQHandler(){
81     if(EXTI->PR1 & EXTI_PR1_PIF3_Msk){
82         EXTIKeyPad_Handler(0);
83         EXTI->PR1 |= EXTI_PR1_PIF3_Msk;
84     }
85 }
86
87 void EXTI4_IRQHandler(){
88     if(EXTI->PR1 & EXTI_PR1_PIF4_Msk){
89         EXTIKeyPad_Handler(1);
90         EXTI->PR1 |= EXTI_PR1_PIF4_Msk;
91     }
92 }
93
94 void EXTI9_5_IRQHandler(){
95     if(EXTI->PR1 & EXTI_PR1_PIF5_Msk){
96         EXTIKeyPad_Handler(2);
97         EXTI->PR1 |= EXTI_PR1_PIF5_Msk;
98     }
99     if(EXTI->PR1 & EXTI_PR1_PIF6_Msk){
100         EXTIKeyPad_Handler(3);
101         EXTI->PR1 |= EXTI_PR1_PIF6_Msk;
102     }
103 }
104
105 int main()
106 {
107     // Cause we want to use floating points we need to init FPU
108     FPU_init();
109     if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
110         // Fail to init 7seg
111         return -1;
112     }

```

```

112     }
113     if(init_keypad(ROW_gpio, COL_gpio, ROW_pin, COL_pin) != 0){
114         // Fail to init keypad
115         return -1;
116     }
117
118     // Set Decode Mode to non-decode mode
119     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE,0xFF);
120     // Set Scan Limit to digit 0 only
121     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT,0xFF);
122     // Wakeup 7seg
123     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN,0x01);
124
125     // Configure SysTick clk to 10 Mhz and interrupt every 0.1s
126     SystemClock_Config_Interrupt(10, 10000);
127     // Init Interrupts
128     EXTI_Setup();
129
130     while(1);
131     return 0;
132 }

```

PART 3. (50%) 實作題

Lab6.3: 製作簡單鬧鐘

```

1  #include <stdio.h>
2  #include "stm32l476xx.h"
3  #include "helper_functions.h"
4  #include "7seg.h"
5  #include "keypad.h"
6  #include "led_button.h"
7  #include "timer.h"
8
9  // Define Counter timer
10 #define COUNTER_timer TIM2
11
12 // Define pins for 7seg
13 #define SEG_gpio GPIOC
14 #define DIN_pin 3
15 #define CS_pin 2
16 #define CLK_pin 1
17
18 // Define pins for keypad
19 #define COL_gpio GPIOA
20 #define COL_pin 6 // 6 7 8 9
21 #define ROW_gpio GPIOB
22 #define ROW_pin 3 // 3 4 5 6
23
24 // Define pins for button (default use on-board button PC13)
25 #define BUTTON_gpio GPIOC
26 #define BUTTON_pin 13
27
28 int num = 0, tmp = 0;
29
30 int alarm_state = 0;
31 // state 0: wait for input
32 // state 1: alarm
33
34 // button_press_cycle_per_second (How many button press segments in a second)
35 #define button_press_cycle_per_second 20
36 // Use to state how many cycles to check per button_press_cycle
37 #define debounce_cycles 50
38 // Use to state the threshold when we consider a button press
39 #define debounce_threshold (debounce_cycles*0.7)
40 // Used to implement negative edge trigger 0=not-presses 1=pressed
41 int last_button_state = 0;
42
43 //
44 void NVIC_Configuration() {

```

```

44 void NVIC_Configuration() {
45     NVIC_SetPriority(EXTI15_10_IRQn, 0); // 最高優先順序
46     NVIC_SetPriority(SysTick_IRQn, 1); // 次高優先順序
47
48     // Enable SYSCFG clock
49     RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
50
51     // Select output bits
52     SYSCFG-> EXTICR[3] &= ~SYSCFG_EXTICR4_EXTI13_Msk;
53     SYSCFG-> EXTICR[3] |= (2 << SYSCFG_EXTICR4_EXTI13_Pos);
54
55     // Enable interrupt
56     EXTI->IMR1 |= EXTI_IMR1_IM13;
57
58     // Enable Falling Edge
59     EXTI->FTSR1 |= EXTI_FTSR1_FT13;
60
61     // Enable NVIC
62     NVIC_EnableIRQ(EXTI15_10_IRQn);
63 }
64
65 // SysTick timer中斷處理函數
66 void SysTick_Handler(void) {
67     if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk && alarm_state == 1) {
68         if (num > 0)
69             num--;
70         if (num <= 0)
71         {
72             timer_init(TIM2, 383, 20);
73             timer_start(TIM2);
74             display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, 0, 1);
75         }
76     }
77 }
78
79 // User button 外部中斷處理函數
80 void EXTI15_10_IRQHandler(void){
81     if(EXTI->PR1 & EXTI_PR1_PIF13_Msk && alarm_state == 1){
82         for(int a=0;a<button_press_cycle_per_second;a++){
83             // Simple Debounce without interrupt
84             int pos_cnt=0;
85             for(int a=0;a<debounce_cycles;a++){
86                 // If button press add count
87                 if(read_gpio(BUTTON_gpio, BUTTON_pin)==0){
88                     pos_cnt++;
89                 }
90                 delay_without_interrupt(1000/(button_press_cycle_per_second*debounce_cycles));
91             }
92             // Check if need to change state
93             if(pos_cnt>debounce_threshold){
94                 // Pressed
95                 num = 0;
96                 alarm_state = 0;
97                 timer_stop(TIM2);
98                 EXTI->PR1 |= EXTI_PR1_PIF3_Msk;
99                 last_button_state = 1;
100                 tmp = 0;
101                 num = 0;
102             }
103             else{
104                 if(last_button_state==0){
105                     // Released button - Not pressing
106                     // Do nothing
107                 }
108                 else{
109                     // Released button - Neg edge
110                     // Do nothing
111                 }
112                 last_button_state = 0;
113             }
114         }
115     }
116 }
117
118 int keypad_space[16][2] = {{0, 0}, {0, 1}, {0, 2}, {0, 3},
119                             {1, 0}, {1, 1}, {1, 2}, {1, 3},
120                             {2, 0}, {2, 1}, {2, 2}, {2, 3},
121                             {3, 0}, {3, 1}, {3, 2}, {3, 3}};
122
123 int get_keyboard_input()
124 {
125     for (int i = 0; i < 16; i++)
126     {
127         int a = keypad_space[i][0];

```

```

128     int a = keypad_space[i][0];
129     int b = keypad_space[i][1];
130     if (check_keypad_input_one(ROW_gpio, COL_gpio, ROW_pin, COL_pin, a, b))
131     {
132         while (check_keypad_input_one(ROW_gpio, COL_gpio, ROW_pin, COL_pin, a, b) != 0)
133             continue;
134         return keypad[a][b];
135     }
136 }
137 return -1;
138 }
139
140 int main()
141 {
142     // Cause we want to use floating points we need to init FPU
143     FPU_init();
144
145     if (init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0)
146     {
147         // Fail to init 7seg
148         return -1;
149     }
150     if (init_keypad(ROW_gpio, COL_gpio, ROW_pin, COL_pin) != 0)
151     {
152         // Fail to init keypad
153         return -1;
154     }
155     if (init_button(BUTTON_gpio, BUTTON_pin) != 0){
156         // Fail to init button
157         return -1;
158     }
159     // Set Decode Mode to non-decode mode
160     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0xFF);
161     // Set Scan Limit to digit 0 only
162     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0xFF);
163     // Wakeup 7seg
164     send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);
165
166     // begin with 0
167     display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, 0, 1);
168
169     GPIO_init_AF();
170     timer_enable(TIM2);
171     PWM_channel_init();
172
173     while (1)
174     {
175         if (alarm_state == 0)
176         {
177             tmp = get_keyboard_input();
178             if (tmp != -1)
179                 num = num * 10 + tmp;
180             display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, num, num_digits(num));
181             for(int a=0;a<button_press_cycle_per_second;a++){
182                 // Simple Debounce without interrupt
183                 int pos_cnt=0;
184                 for(int a=0;a<debounce_cycles;a++){
185                     // If button press add count
186                     if(read_gpio(BUTTON_gpio, BUTTON_pin)==0){
187                         pos_cnt++;
188                     }
189                     delay_without_interrupt(1000/(button_press_cycle_per_second*debounce_cycles));
190                 }
191                 // Check if need to change state
192                 if(pos_cnt>debounce_threshold){
193                     if(last_button_state==0 && num!=0){
194                         // Pressed button - Pos edge
195                         alarm_state = 1;
196                         // 設定中斷優先順序
197                         NVIC_Configuration();
198                         // Configure SysTick clk to 10 Mhz and interrupt every 1s
199                         SystemClock_Config_Interrupt(10, 10000000);
200                     }
201                     else{
202                         // Pressed button - Continued pressing
203                         // Do nothing
204                     }
205                     last_button_state = 1;
206                 }
207                 else{
208                     if(last_button_state==0){
209                         // Released button - Not pressing
210                         // Do nothing
211                     }
212                     else{

```

```
211         }
212         else{
213             // Released button - Neg edge
214             // Do nothing
215         }
216         last_button_state = 0;
217     }
218 }
219 }else if(alarm_state == 1){
220     display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, num, num_digits(num));
221 }
222 }
223 while(1);
224 return 0;
225 }
```