

Rep_Lab_03_B0829002

Problem Description:

在這個實驗中，我們的目標是寫一個 Program，使其能夠在不使用 `delay function` 的情況下改變按鈕的狀態。為了達到這一目的，我們選擇使用 8051 CPU 提供的 Timer 功能。這個 Timer 不僅可以用來進行按鈕狀態的改變，還可以用來觸發 interrupt。

Lab2 的方法中，當按鈕被按下時，系統通常會使用 `delay function` 來產生一段時間的延遲，這樣做的目的是為了避免按鈕的抖動或是其他不希望的行為。但這種方法的缺點是它會導致系統在這段延遲時間內無法進行其他操作。

因此，我們選擇使用 8051 CPU 的 Timer 功能來替代 `delay function`。當 Timer 到達設定的時間時，它會自動觸發一個 interrupt，這樣我們就可以在這個 interrupt 中進行按鈕狀態的改變，而不需要等待 `delay function` 的完成。這種方法的優點是它可以確保系統在整個過程中都保持活躍，並且可以即時地對按鈕的操作進行反應。透過使用 8051 CPU 提供的 Timer 和 interrupt 功能，我們可以設計出一個既快速又可靠的按鈕狀態改變系統，而不需要依賴傳統的 `delay function`。

Code and explanations:

在這裡我們利用 8051 的 datasheet 設定 Timer config, TMOD 設為 XXXX1001，GATE0 為 0 忽略外部訊號，C/T0 為 0 選擇 T0M 的計時模式，並使用 T0M1、T0M0 的 0/1 設定為 16 bit 計時模式。[1] [2]



圖 1: Timer 位址和變數設定



圖 2: Timer 基本狀態和數值設定

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x89
SFR Page: 0

- Bit7:** GATE1: Timer 1 Gate Control.
0: Timer 1 enabled when TR1 = 1 irrespective of /INT1 logic level.
1: Timer 1 enabled only when TR1 = 1 AND /INT1 = logic 1.
- Bit6:** C/T1: Counter/Timer 1 Select.
0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.4).
1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).
- Bits5-4:** T1M1-T1M0: Timer 1 Mode Select.
These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

圖 3: Timer 設定 Datasheet

TCON 設為 XXX10000 以啟動 timer0。當 timer0 overflow 時，TF0 自動設為 1 並觸發中斷。

SFR Definition 23.1. TCON: Timer Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable
SFR Address: 0x88 SFR Page: 0								
Bit7: TF1: Timer 1 Overflow Flag. Set by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine. 0: No Timer 1 overflow detected. 1: Timer 1 has overflowed.								
Bit6: TR1: Timer 1 Run Control. 0: Timer 1 disabled. 1: Timer 1 enabled.								

圖 4: Timer 模式設定 Datasheet

接下來我們在設定 Interrupt 的 config:

IE 設定為 10000010，再將 EA, ET0 設為 0 來開啟中斷模式以及打開利用 Timer 來中斷。

SFR Definition 12.11. IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable
SFR Address: 0xA8 SFR Page: All Pages								
Bit7: EA: Enable All Interrupts. This bit globally enables/disables all interrupts. It overrides the individual interrupt mask settings. 0: Disable all interrupt sources. 1: Enable each interrupt according to its individual mask setting.								

圖 5: Interrupt 設定 Datasheet

最後我們再設定 OSCICN 使 SYSCLK 為頻率的 1/8，並調整 CKCON 使 Timer 為 SYSCLK 的 1/12（由 SCA0 和 SCA1 決定）。

其餘的部分與 Lab2 大同小異，利用 button status 和 Timer 來進行 output 的改變，每當 Timer 數到我們設定的數目時，確認經由 Btn 設定的 variable 去判斷要輸出給 output LED 哪些訊號（RightRotate, LeftRotate, EvenBlink, OddBlink），最後再將對應的 status output 輸出到 LED 上。

```

1  Loop_Begin:
2
3      // R0 is for LED so it cant be used at other plac
4  e      mov    R1, 20h
5
6      // check the status sign
7      mov    A, R1
8      rotateRightFunc: // right shift
9          cjne A, #1, rotateLeftFunc
10         mov    A, R0
11         rr     A
12         mov    P2, A
13         mov    R0, A
14         ret
15     rotateLeftFunc: // left shift
16         cjne A, #2, blinkEvenFunc
17         mov    A, R0
18         rl     A
19         mov    P2, A
20         mov    R0, A
21         ret
22     blinkEvenFunc: // blink even
23         cjne A, #3, blinkOddEven
24         xrl    B, #01010101b
25         mov    P2, B
26         ret
27     blinkOddEven: // blink odd
28         cjne A, #4, Jump
29         xrl    B, #10101010b
30         mov    P2, B
31         ret
32     Jump:
33         ret

```

圖 6: 按鈕監聽 Function



```
1  handleOnClick:
2      mov     R1, P1
3      mov     A,  R1
4      rotateRight: // RR
5          cjne  A,  #10000000b, rotateLeft
6      t       mov  20h,  #1
7          ret
8
9      rotateLeft://  RL
10         cjne  A,  #01000000b, blinkEven
11         mov   20h,  #2
12         ret
13     blinkEven://    BlinkEven
14         cjne  A,  #00100000b, blinkOdd
15         mov   20h,  #3
16         mov   B,  #0
17         ret
18     blinkOdd://    BlinkOdd
19         cjne  A,  #00010000b, return
20         mov   20h,  #4
21         mov   B,  #0
22         ret
23     return:
24         ret
```

圖 7: 按鈕狀態轉換 Function

Difficulties and Problem:


這次實驗的主要困難在理解 **Timer** 的運作和熟習設定。除此之外，其他部分主要是 Lab02 的方法去與本次的 **Timer** 進行串接。

Discussions or what I have learned:

本次實驗學習到如何使用 **Timer** 進行 **Interrupt** 的操作，發現利用 **Timer** 可以更加流暢的進行 **Btn** 狀態的轉換，感覺之後除了 **Btn** 外應該會有更多可以利用 **Timer interruption** 對 **CPU** 進行更精準和更有效率的數值和位址操作。

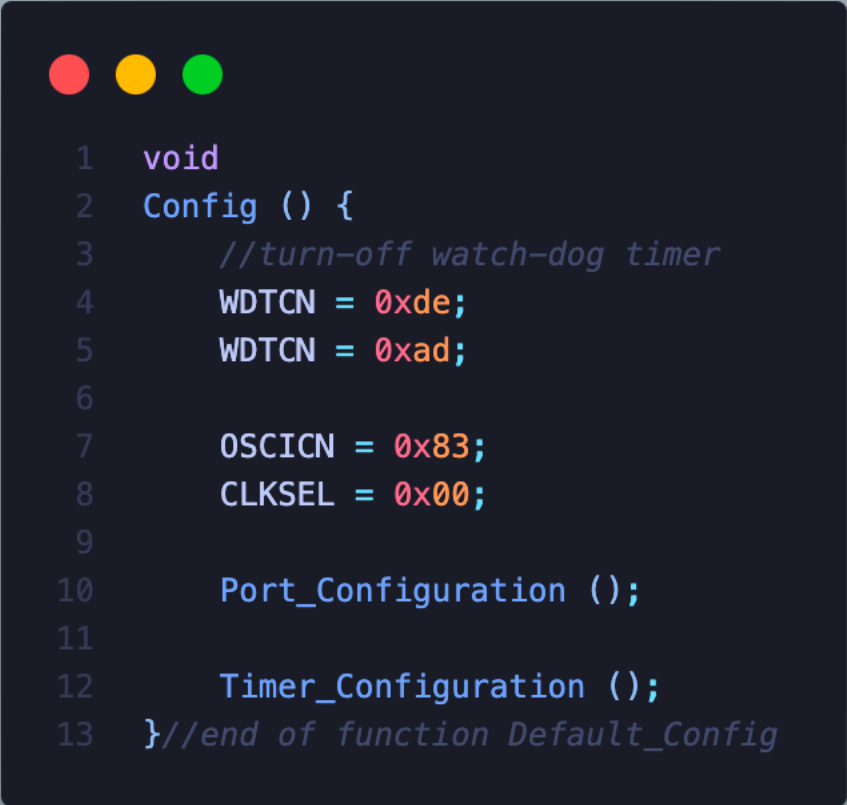
C code explanation:

接下來利用 **Caption** 進行 **C code** 解釋:



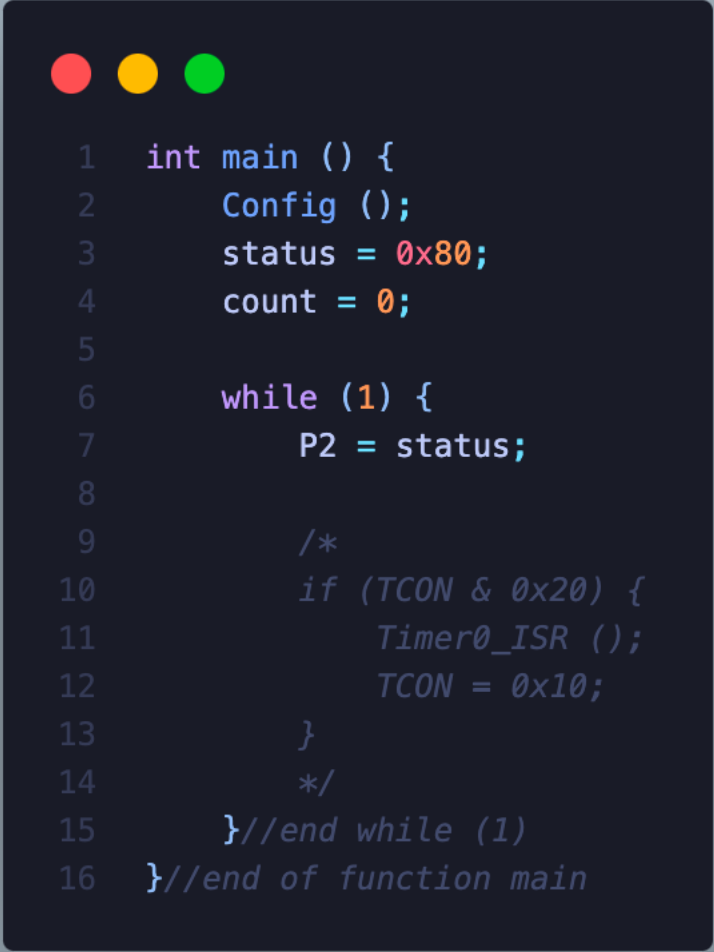
```
1  #include "C8051F040.h"
2
3  void Port_Configuration () {
4      //initialize SFR setup page
5      SFRPAGE = CONFIG_PAGE;           // Switch to configuration page
6      //setup the cross-bar and configure the I/O ports
7      XBR2 = 0xc0;
8      P2MDOUT = 0xff;
9      //set to normal mode
10     SFRPAGE = LEGACY_PAGE;
11 }//end of function Port_Configuration
12
13 void Timer_Configuration () {
14     TMOD = 0x01;
15     TCON = 0x10;
16     CKCON = 0x10;
17     IE = 0x82;
18     TL0 = 0;
19     TH0 = 0;
20 }//end of function Timer_Configuration
```

圖 8：透過 **Port_Configuration** 和 **Timer_Configuration** functions 進行 **Port**, **Page** 和 **Timer** 的設定（參見 Figure 2.）



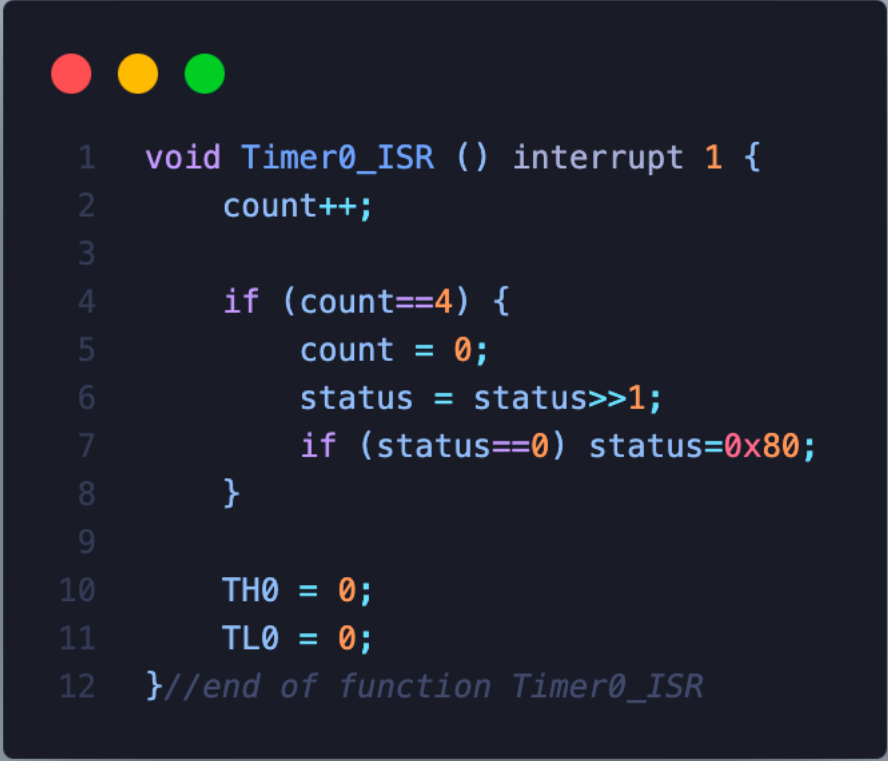
```
1  void
2  Config () {
3      //turn-off watch-dog timer
4      WDTCN = 0xde;
5      WDTCN = 0xad;
6
7      OSCICN = 0x83;
8      CLKSEL = 0x00;
9
10     Port_Configuration ();
11
12     Timer_Configuration ();
13 } //end of function Default_Config
```

圖 9：關閉看門狗計時器



```
1  int main () {  
2      Config ();  
3      status = 0x80;  
4      count = 0;  
5  
6      while (1) {  
7          P2 = status;  
8  
9          /*  
10         if (TCON & 0x20) {  
11             Timer0_ISR ();  
12             TCON = 0x10;  
13         }  
14         */  
15     } //end while (1)  
16 } //end of function main
```

圖 10: 初始化 Count, Status 以及對上二圖進行設定 (參見 Figure 8., Figure 9.)



```
1 void Timer0_ISR () interrupt 1 {  
2     count++;  
3  
4     if (count==4) {  
5         count = 0;  
6         status = status>>1;  
7         if (status==0) status=0x80;  
8     }  
9  
10    TH0 = 0;  
11    TL0 = 0;  
12 }//end of function Timer0_ISR
```

圖 11: 處理 Timer interrupt 時會做 status 轉換

Reference:

1. Ting, M. (2023, July 8). *[SDCC for 8051] 04-timer*. Medium.
[https://medium.com/%E9%96%B1%E7%9B%8A%E5%A6%82%E7%BE%8E/sdcc-for-](https://medium.com/%E9%96%B1%E7%9B%8A%E5%A6%82%E7%BE%8E/sdcc-for-8051-timer)
2. *8051 timer*. 8051 Timer. (2009, March 17). <https://jyhshin3.blogspot.com/2009/03/8051-timer.html>