Lab 16:閃黃燈, 按鍵調整週期(2020 Final Lab)

1. 連接著P1.1的是一顆黃色的LED燈, 透過按鍵可以調整其週期, 也可以暫停/恢復燈號的閃爍.
2. 黃燈閃爍的亮與滅維持相同的時間. 原始設定為亮300微秒, 滅300微秒, ;重複此過程.
3. 按鍵'4'的用途是亮與滅各增加其時間30微秒, 也就是按一次變成330微秒, 兩次360微秒, 上限為450微秒.
4. 按鍵'6'的用途是亮與滅各減少其時間30微秒, 也就是按一次變成270微秒, 兩次240微秒, 下限為150微秒.
5. 按鍵'4'與按鍵'6'可以交替使用, 未達限值時都有增加或減少的作用, 達到上限再按'4'或者達到下限再按'6', 亮滅時間就不再變動.
6. 按鍵'5'的功用是凍結燈號的閃爍, 轉為長亮或長滅. 再按一次'5', 就會恢復閃爍.
7. 參考邏輯圖, P0.6~P0.4經過一個AND 閘之後, 接到P3.3, 也就是INT1
8. EdSim面板的keypad, 應該會顯示AND Gate Enable, 否則請按下調整
9. 使用中斷來偵測按鍵的好處是主程式可以放假, 最後停在JMP $
10. 提示:
    a、 Timer 0的週期設定為30 micro-seconds, 另外搭配上倍數(R7), 就可以得出30*R7 micro-seconds的週期
    b、 Timer 0請使用auto-reload 的功能,溢位的偵測請使用中斷
    c、 負責keypad的程式移動到外部中斷的EX1ISR
    d、 EdSim的按鍵'4', '5'與'6'都在同一個row.
    e、 INT1 使用falling edge trigger, key的按下動作會觸發中斷, key的鬆開就沒有影響
    f、 不需要調整中斷的優先順序, 但是需要啟動(enable)中斷
    g、 T0ISR與EX1ISR要放在適當的位址
    h、 善用breakpoints, 確認
        i. timer的週期正確, 而且會觸發T0ISR的執行,
        ii. 按鍵會觸發EX1ISR的執行,
        iii. 重要暫存器的數值正確
    i、 Update Freq. 可以調整顯示的快慢, 數值也可以手動輸入.
    j、 如果讀取port的pin腳發生問題, 可以先輸出'1'到pin腳, 再讀取之.
    k、 程式請放置到適當的位址, SP的初值設定為40H
    l、 程式的寫作禮儀, 如註解的編寫, 程式的縮排都請滿足.
11. [重要評分提醒]黃燈的閃爍請獨立出一個名為"FLASH_YELLOW"的副程式, 其中只有兩行程式"CPL P1.1; RET", 教師將以此測量週期與評定分數

12. Two square wave program

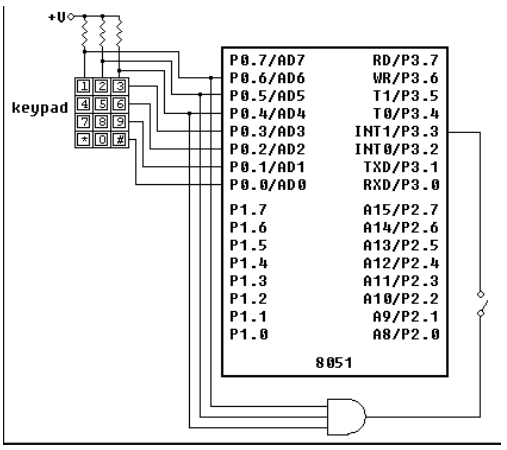
```
        ORG   0000H          ;Reset
        LJMP  MAIN
        ORG   000BH          ;Timer0 vector
        LJMP  T0ISR
        ORG   001BH          ;Timer1 vector
        LJMP  T1ISR
        ORG   0030H          ;Main program entry
MAIN:   MOV   TMOD, #12H ;timer1 mode 1,timer0 mode 2
        MOV   TH0, # -71     ;7KHz using timer0
        SETB  TR0            ;Start timer0
        SETB  TF1            ;Force timer1 interrupt
        MOV   IE, #8AH       ;enable both intrrrupts
        SJMP  $              ;do nothing
```

```
T0ISR:  CPL   P1.7          ;Toggle port bit
        RETI                ;Return to main program
T1ISR:  CLR   TR1
        MOV   TH1, #0FCH  ; -1000 1ms high
        MOV    TL1, #18H    ;1ms low delay
        SETB  TR1            ;Start timer1
        CPL    P1.6
        RETI
        END
```

13. Keypad scan program

```
start:

        MOV R0, #0              ; clear R0 - the first key is key0

        ; scan row0
        SETB P0.3               ; set row3
        CLR P0.0                ; clear row0
        CALL colScan            ; call column-scan subroutine
        JB F0, finish           ; | if F0 is set, jump to end of program
                                ; | (because the pressed key was found and its number is in  R0)

        ; scan row1
        SETB P0.0               ; set row0
        CLR P0.1                ; clear row1
        CALL colScan            ; call column-scan subroutine
        JB F0, finish           ; | if F0 is set, jump to end of program
                                ; | (because the pressed key was found and its number is in  R0)

        ; scan row2
        SETB P0.1               ; set row1
        CLR P0.2                ; clear row2
        CALL colScan            ; call column-scan subroutine
        JB F0, finish           ; | if F0 is set, jump to end of program
                                ; | (because the pressed key was found and its number is in  R0)

        ; scan row3
        SETB P0.2               ; set row2
        CLR P0.3                ; clear row3
        CALL colScan            ; call column-scan subroutine
        JB F0, finish           ; | if F0 is set, jump to end of program
                                ; | (because the pressed key was found and its number is in  R0)


        JMP start               ; | go back to scan row 0
                                ; | (this is why row3 is set at the start of the program
                                ; | - when the program jumps back to start, row3 has just been scanned)

finish:
        JMP $                   ; program execution arrives here when key is found - do nothing


; column-scan subroutine
colScan:
        JNB P0.4, gotKey        ; if col0 is cleared - key found
        INC R0                  ; otherwise move to next key
        JNB P0.5, gotKey        ; if col1 is cleared - key found
        INC R0                  ; otherwise move to next key
        JNB P0.6, gotKey        ; if col2 is cleared - key found
        INC R0                  ; otherwise move to next key
        RET                     ; return from subroutine - key not found
gotKey:
        SETB F0                 ; key found - set F0
        RET                     ; and return from subroutine
```

14. Keypad logic diagram


+V
keypad
1 2 3
4 5 6
7 8 9
* 0 #
P0.7/AD7
P0.6/AD6
P0.5/AD5
P0.4/AD4
P0.3/AD3
P0.2/AD2
P0.1/AD1
P0.0/AD0
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0
RD/P3.7
WR/P3.6
T1/P3.5
T0/P3.4
INT1/P3.3
INT0/P3.2
TXD/P3.1
RXD/P3.0
A15/P2.7
A14/P2.6
A13/P2.5
A12/P2.4
A11/P2.3
A10/P2.2
A9/P2.1
A8/P2.0
8051