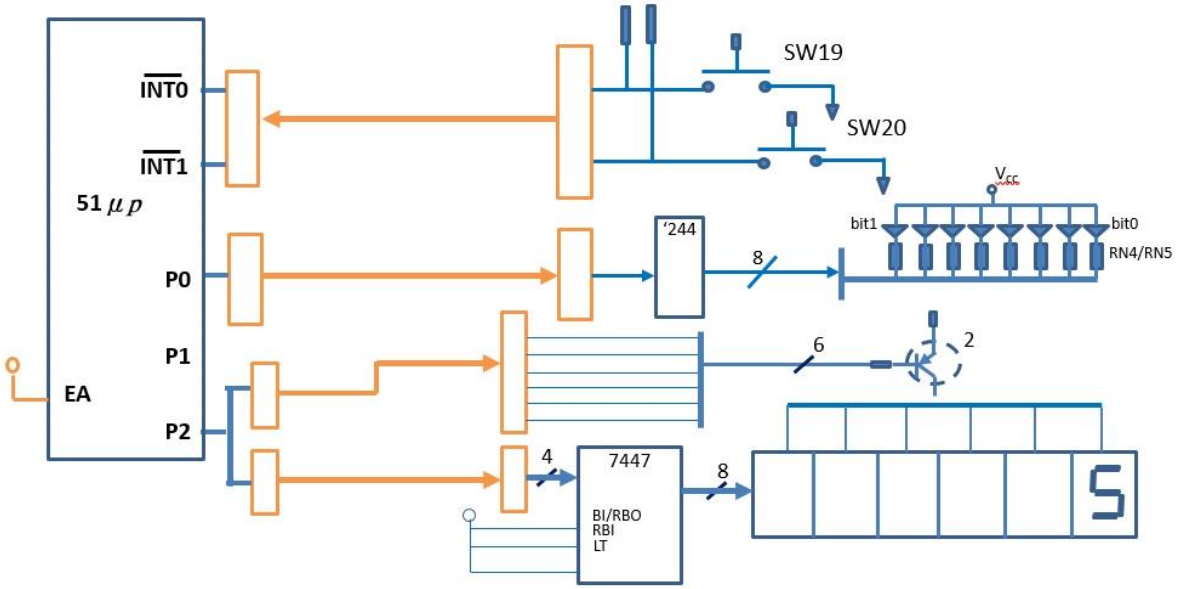
# Microprocessor Lab 6 Report

0416106 彭敬樺
0416109 周才錢

**Subject and Goal:**

This lab is about using µ-Vision 51IDE residing on MegaWin82G516 to:

- operations with two external interrupts and two timer interrupts
- built-in counter/timer setup and control for timing application
- time-up event detection by timer interrupt
- key entry event by external interrupt
- priority resolving among multi-interrupt




**Preparations:**

- Power cable and required connection from the output to the led input is established. Port P0 will control 8x1 discrete LED module. Post P1 and P2 will control 7-segment power module and six 7-seg LED DIGs respectively
- Check the correctness and check if there are any defective on the board by activating all 8x1 discrete LED module and 7-segment LED module using static/dynamic pattern display.

**Operating Procedure:**

- Jumper-wiring for circuit setup
- Check the 8x1 discrete LED module and 7-segment LED module to see if it's working or not by running code to turn all the light on.
- Code preparation
- Task execution:
    - Start IDE51 emulation,
    - Start execution and troubleshooting if necessary.

## Code Preparation:

```
; ============================
; (1) 4 interrupt events in action
; (2) 1x8 LEDs
;    normal ON-OFF □ OFF-ON
;    INT0 □ L2R scanning
;    INT1 □ R2L scanning
; (3) 7-seg LEDs
;    DIG6-5: timing by second (TF1)
;    DIG4-1: timing by 10ms (TF0)
; (4) priority
;    priority order of the 4
;    interrupts  ???
; ============================
; port0: 1x8 discrete LEDs control
; port1: power-SW control for
;       7-seg LED module
; port 2: pattern control for
;      timing display on 7-seg LEDs
; 30H:   TMR1 10-2sec-register
; 31H:   TMR0 sec-register
; 32H:   TMR0 10-2sec-register
; 33H:   TMR0 10-2sec time-up flag
; 34H:   TMR1 10-2sec time-up flag
; 35H:   TMR1 counter: 0-19
; 36H:  normal pattern for 1x8 LEDs
; BITMAP 0H:   INT0 flag
; BITMAP 1H:   INT1 flag
; BITMAP 2H:   left-to-right Cy
; BITMAP 3H:   right-to-left Cy
; 40H:   1x8 LED pattern for INT0
; 41H:   1x8 LED pattern for INT1
org   0
jmp   start
org   03H
jmp   INT0
org   0BH
jmp   TMR0
org   13H
      jmp   INT1
      org   1BH
      jmp   TMR1
start:
      mov  sp, #50H
      mov  TMOD, #11H
      mov  TH0, #>(65536-10000)
      mov  TL0, #<(65536-10000)
      mov  TH1, #>(65536-50000)
      mov  TL1, #<(65536-50000)
      mov  IE, #8FH
      mov  IP, #0AH
      mov   30H, #0
      mov   31H, #0
      mov   32H, #0
      mov   33H, #0
      mov   34H, #0
      mov   35H, #0
      clr    0H
clr    1H
mov   40H, #0FFH
mov   41H, #0FFH

setb   TR0
      setb   TR1
      ;mov   P1, #0H
      ;mov   P2, #0H
      clr     C
      mov   2H, C
      mov   3H, C
 display_go:

 normcycle:
      mov   36H, #0FH
mov   R4, #2
mov   R5, #14
 halfcycle:
      mov   P0, 36H
 int_in_session:
mov   R6, #30
DIG6:   mov   A, 30H
      mov   B, #10
```

```
        div    A, B
        mov    P1, #0DFH
        mov    P2, A
        call   delay
 DIG5:  mov    A, B
        mov    P1, #0EFH
        mov    P2, A
        call   delay
DIG4:   mov    A, 31H
        mov    B, #10
        div    A, B
        mov    P1, #0F7H
        mov    P2, A
        call   delay
 DIG3:  mov    A, B
        mov    P1, #0FBH
        mov    P2, A
        call   delay
 Dig2:  mov    A, 32H
        mov    B, #10
        div    A, B
        mov    P1, #0FDH
        mov    P2, A
        call   delay
 DIG1:  mov    A, B
        mov    P1, #0FEH
        mov    P2, A
        call   delay
        djnz   R6, DIG6
        ;mov   36H, #0F0H
int1test:
        jnb    1H, int0test
        mov    A, 41H
        push   PSW
        mov    C, 3H
        rrc    A
        mov    3H, C
        pop    PSW
        mov    P0, A
        mov    41H, A
        jb     3H, int1_in_session
        clr    1H
int0test:
        jnb    0H, norm_cont
        mov    A, 40H
        push   PSW
        mov    C, 2H
        rlc    A
        mov    2H, C
        pop    PSW
        mov    P0, A
        mov    40H, A
        jb     2H, int0_in_session
        clr    0H
        jmp    norm_cont
int1_in_session:
int0_in_session:
        jmp    int_in_session
norm_cont:
        djnz   R5, midway
        mov    36H, #0F0H
        mov    R5, #14
        ;djnz  R4, halfcycle  ; Q0

        djnz   R4, midway   ; Q1
        jmp    display_go
midway:
jmp    halfcycle    ; Q2

 delay:  ; ? how long should the
        ; delay be for visual
        ; satisfaction?
        ret
TMR0:   push   PSW
       push    A
      mov    TH0, #>(65536-10000)
      mov    TL0, #<(65536-10000)
        setb   TR0
        inc    32H
        mov    A, 32H
        cjne   A, #100, ext1
        mov    32H, #0
```

```
        inc     31H
        mov     A, 31H
        cjne    A, #60, ext1
        mov     31H, #0
ext1:   pop     A
        pop     PSW
        reti

TMR1:   push    PSW
        push    A
        mov     TH1, #>(65536-50000)
        mov     TL1, #<(65536-50000)
        setb    TR1
        inc     35H
        mov     A, 35H
        cjne    A, #20, ext2
        mov     35H, #0
        inc     30H
        mov     A, 30H
        cjne    A, #60, ext2
        mov     30H, #0
ext2:   pop     A
        pop     PSW
        reti
INT0:   jb      0H, intext0
        push    A
        push    PSW
        setb    0H
        pop     PSW
        pop     A
intext0:
        reti
INT1:   jb      1H, intext1
        push    A
        push    PSW
        setb    1H
        pop     PSW
        pop     A
intext1:
        reti

        end
; ========================
```

**Observation:**

- The code is running well, but some parts of the prepared code need to be fixed to prevent error popup. The addition of delay function will also let user see the sequence easier. All the wanted sequence react to the correct button being pressed in the switch.
- Simple execution flow description: Firstly, calculate the timer value at one point of the time by using the built-in counter in 89c51. The value that will be displayed then has to be calculated in BCD value before being feed into the port that will be responsible for the number display. Next, display the value to appropriate location of the LED module. The next part is the module to control the behavior of 8x1 LED segment if some input is detected or not.
- The bit 0 and 1 in BITMAP is used to handle and temporarily saved the condition of whether an interrupt for the 7-segment LED module is detected or not, when interrupt has been handled, then the value will be reset.
- The bit 2 and 3 in BITMAP is used to handle and temporarily saved the condition of whether an interrupt for the 8x1 LED module is detected or not, when interrupt has been handled, then the value will be reset.
- 40H and 41H is the one that holds the LED pattern for the 8x1 LED module
- Push pop of PSW when handling interrupt is unnecessary due to the push of Acc prior and the pop of Acc post the PSW push pop instruction

- The code written above is not a really bad way of arrangement if the goals is only to write a functioning code. However, by managing each wanted jobs into separate function, the code will be easier to read, edit, and implement in future modification

**Modified code:**

- Modified code with loop applied to the scanning body:

```
; ============================
; (1) 4 interrupt events in action
; (2) 1x8 LEDs
;    normal ON-OFF ? OFF-ON
;    Control_int0 ? L2R scanning
;    Control_int1 ? R2L scanning
; (3) 7-seg LEDs
;    DIG6-5: timing by second (TF1)
;    DIG4-1: timing by 10ms (TF0)
; (4) priority
;    priority order of the 4
;    interrupts  ???
; ============================
; port0: 1x8 discrete LEDs control
; port1: power-SW control for
;       7-seg LED module
; port 2: pattern control for
;      timing display on 7-seg LEDs
; 30H:   TMR1 10-2sec-register
; 31H:   TMR0 sec-register
; 32H:   TMR0 10-2sec-register
; 33H:   TMR0 10-2sec time-up flag
; 34H:   TMR1 10-2sec time-up flag
; 35H:   TMR1 counter: 0-19
; 36H:  normal pattern for 1x8 LEDs
; BITMAP 0H:   Control_int0 flag
; BITMAP 1H:   Control_int1 flag
; BITMAP 2H:   left-to-right Cy
; BITMAP 3H:   right-to-left Cy
; 40H:   1x8 LED pattern for Control_int0
; 41H:   1x8 LED pattern for Control_int1
org   0
jmp   start
org   03H
jmp   Control_int0
org   0BH
jmp   TMR0
org   13H
      jmp   Control_int1
      org   1BH
      jmp   TMR1
start:
      mov  sp, #50H
      mov  TMOD, #11H
      mov      TH0,    #11011000b;#>(65536-10000)
      mov      TL0,    #11110000b;#<(65536-10000)
      mov      TH1,    #00111100b;#>(65536-50000)
      mov      TL1,    #10110000b;#<(65536-50000)
      mov  IE, #8FH
      mov  IP, #0AH
      mov   30H, #0
      mov   31H, #0
      mov   32H, #0
      mov   33H, #0
      mov   34H, #0
      mov   35H, #0
      clr    0H
clr    1H
mov   40H, #0FFH
mov   41H, #0FFH

setb   TR0
      setb   TR1
      ;mov   P1, #0H
      ;mov   P2, #0H
      clr     C
      mov   2H, C
      mov   3H, C
```

```
display_go:

normcycle:
    mov   36H, #0FH
mov   R4, #2
mov   R5, #14
halfcycle:
    mov   P0, 36H
int_in_session:
mov   R6, #30
DIG6:   mov   A, 30H
    mov   B, #10
    div    AB
    mov   P1, #0DFH
    mov   P2, A
    call    delay
DIG5:   mov   A, B
    mov   P1, #0EFH
    mov   P2, A
    call    delay
DIG4:   mov   A, 31H
    mov   B, #10
    div    AB
    mov   P1, #0F7H
    mov   P2, A
    call    delay
DIG3:  mov    A, B
    mov   P1, #0FBH
    mov   P2, A
    call    delay
Dig2:   mov   A, 32H
    mov   B, #10
    div    AB
    mov   P1, #0FDH
    mov   P2, A
    call    delay
DIG1:  mov    A, B
    mov   P1, #0FEH
    mov   P2, A
    call    delay
    djnz   R6, DIG6
    ;mov   36H, #0F0H
int1test:
    jnb     1H, int0test
    mov    A, 41H
    push    PSW
    mov    C, 3H
    rrc      A
    mov    3H, C
    pop     PSW
    mov    P0, A
    mov    41H, A
    jb      3H, int1_in_session
    clr     1H
int0test:
    jnb     0H, norm_cont
    mov    A, 40H
    push    PSW
    mov    C, 2H
    rlc      A
    mov    2H, C
    pop     PSW
    mov   P0, A
    mov   40H, A
    jb     2H, int0_in_session
    clr    0H
    jmp   norm_cont
int1_in_session:
int0_in_session:
    jmp   int_in_session
norm_cont:
    djnz   R5, midway
    mov   36H, #0F0H
    mov   R5, #14
    ;djnz   R4, halfcycle  ; Q0

    djnz   R4, midway   ; Q1
    jmp   display_go
midway:
jmp    halfcycle    ; Q2

 delay:  push  2
```

```
        push  3
        mov  R2, #2
dd1:    mov  R3, #250
        djnz  R3, $
        djnz  R2, dd1
        pop  3
        pop  2
        ret

TMR0:   push   PSW
        push    0E0H ;A
        mov         TH0,  #11011000b;#>(65536-10000)
        mov         TL0,  #11110000b;#<(65536-10000)
        setb    TR0
        inc     32H
        mov     A, 32H
        cjne    A, #100, ext1
        mov     32H, #0
        inc     31H
        mov     A, 31H
        cjne     A, #60, ext1
        mov     31H, #0
ext1:   pop      0E0H ;A
        pop      PSW
        reti

TMR1:   push   PSW
        push    0E0H ;A
        mov       TH1,  #00111100b;  #>(65536-50000)
        mov       TL1,  #10110000b;  #<(65536-50000)
-
        setb    TR1
        inc     35H
        mov     A, 35H
        cjne    A, #20, ext2
        mov     35H, #0
        inc     30H
        mov     A, 30H
        cjne    A, #60, ext2
        mov     30H, #0
 ext2:  pop      0E0H ;A
        pop      PSW
        reti
 Control_int0:    jb      0H, intext0
push    0E0H ;A
        push    PSW
        setb     0H
        pop     PSW
        pop     0E0H ;A
 intext0:
        reti
 Control_int1:    jb      1H, intext1
push    0E0H ;A
        push    PSW
        setb    1H
        pop     PSW
        pop     0E0H ;A
 intext1:
        reti

        end
; ==========================
```