### 1. 題目描述

這次的實驗要實作一個用按鈕觸發不同 LED 輸出事件的 program,因為會使用到按 鈕和 LED,所以我們要對這兩者做一些在 8051 板上的宣告、定義變數。

在本次實驗不僅會用到 PnMDIN、PnMDOUT 還會設定到 WDTCN(看門狗暫存器)、 SRF page(配置面頁暫存器),還有最重要的如何去 implement 按鈕按下的延遲設定以及 如何 handleOnClick 到不同事件以輸出不同的燈號。

## 2. 先備知識

### SRF page register

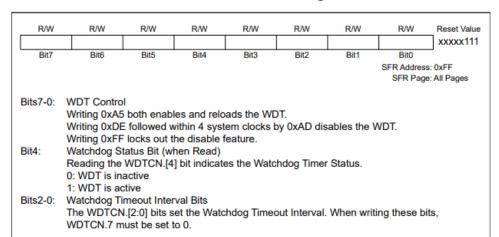
第一是要先找出 SFR 的位置,因為要先去定義 PnMDIN 和 PnMDOUT 用以作為按鈕 和輸入和 LED 輸出,但因為在 Datasheet 上可以看到不是每個 MDIN 和 MDOUT 在每個 page 都可用,所以切到特定 page。

P0	0x80	All Pages	Port 0 Latch
P0MDOUT	0xA4	F	Port 0 Output Mode Configuration
P1	0x90	All Pages	Port 1 Latch
P1MDIN	0xAD	F	Port 1 Input Mode Configuration
P1MDOUT	0xA5	F	Port 1 Output Mode Configuration
P2	0xA0	All Pages	Port 2 Latch
P2MDIN	0xAE	F	Port 2 Input Mode Configuration
P2MDOUT	0xA6	F	Port 2 Output Mode Configuration
P3	0xB0	All Pages	Port 3 Latch
P3MDIN	0xAF	F	Port 3 Input Mode Configuration
P3MDOUT	0xA7	F	Port 3 Output Mode Configuration

### **II. WDTCN register**

在本次實驗中,由於需要等待使用者輸入,程式可能會等待很長的時間。為了避免程式在閒置時被看門狗所中斷,需要先將看門狗功能關閉。看門狗是一種計時器暫存器,當系統運行異常或閒置過長時,它會自動重啟系統,以確保系統的正常運行。但在某些情況下,如此文所述的等待使用者輸入,如果不關閉看門狗,它可能會不斷地重啟系統甚至崩潰。因此,透過將WDTCN(看門狗暫存器)輸入特定的值,如OxDE

#### SFR Definition 13.1. WDTCN: Watchdog Timer Control

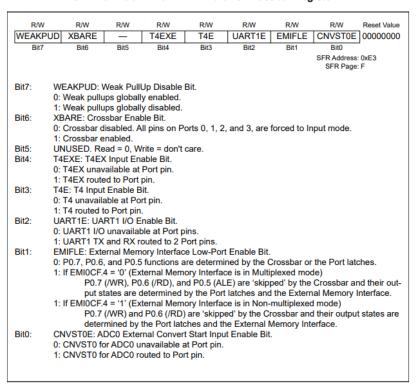


### 定義 Port

接著我們要定義 port 利用在輸出和輸入,將 XBAR2 的第五和第六個 bit 設為 1 調整成輸出模式。

#### C8051F040/1/2/3/4/5/6/7

#### SFR Definition 17.3. XBR2: Port I/O Crossbar Register 2



## 3. 代碼描述

以下為定義上述先備知識所提到的頁數暫存器、PnMDIN、PnMDOUT、看門狗暫存器和 XBR2 的輸出模式,還有定義 P1 到 button register array 跟 p2 到 LED

```
register array∘
     XBR2 equ 0e3h
4.
5.
     P1MDIN equ Oadh
     P2MDOUT equ 0a6h
6.
7.
     WDTCN equ Offh
     SFRPAGE equ 084h
8.
9.
     P1 equ 090h
10.
     P2 equ 0a0h
11.
     CONFIG PAGE equ Ofh
12.
     LEGACY PAGE equ 00h
13.
14.
15.
     // close timer
     mov WDTCN, #0deh
16.
17.
     mov WDTCN, #0adh
18.
19.
     // setup port config
20.
     mov SFRPAGE, #CONFIG_PAGE
     mov XBR2, #0c0h
21.
22.
     mov P1MDIN, #0ffh
23.
     mov P2MDOUT, #0ffh
24.
     mov SFRPAGE, #LEGACY PAGE
25.
     mov P1, #00000000h // btn array reference to 090h
26.
     mov P2, #00000000h // output array reference to 0a0h
27.
28.
29.
     mov R0, #1000000b
     mov B, #0000000b
30.
31.
     // initial signal, status, btn and LED
32.
33.
     mov 20h, #0
     mov P1, #0
34.
     mov R1, #0
35.
     mov R2, #0
36.
      /* 以下為 Loop_Begin 是程式中的主要函式,它會持續運行但不會重複執行。當這個函
      式開始運行時,首先會呼叫 Call_Func 函式。接著,它會根據 Call_Func 的回傳值
      (儲存在 20h 位置)來判定要進行哪一種操作:左移(leftRotate)、右移
      (rightRotate)、偶數 LED 閃爍(evenBlink)、基數 LED 閃爍(oddBlink)。每當要執行
      這些操作之前,程式都會先呼叫 delay 函式,目的是為了讓動作的效果能夠被人眼所捕
      捉,這是基於人眼的視覺暫留效果。 */
     Loop Begin:
37.
     lcall handleOnclick
     // after btn state check, put val. to A, in order to do operation
39.
     mov R1, 20h
40.
     mov A, R1
41.
43.
     rightRotate:
```

cjne A, #1, leftRotate // check event

44.

```
lcall Delay
45.
46.
      mov R3, A
47.
      mov A, R0
      rr A
48.
      mov P2, A
49.
      mov R0, A
50.
51.
      ljmp Loop_Begin
52.
      // R L
53.
54.
      leftRotate:
      cjne A, #2, evenBlink // check event
55.
      lcall Delay
56.
57.
      mov R3, A
58.
      mov A, R0
      rl A
59.
      mov P2, A
60.
61.
      mov R0, A
62.
      ljmp Loop_Begin
63.
      // Blink
64.
      evenBlink:
65.
      cjne A, #3, oddBlink // check event
66.
      lcall Delay
67.
68.
      mov R3, A
69.
      xrl B ,#01010101b
70.
      mov P2, B
      ljmp Loop_Begin
71.
72.
73.
      oddBlink:
      cjne A, #4, Jump // check event
74.
      lcall Delay
75.
      mov R3, A
76.
      xrl B ,#10101010b
77.
      mov P2, B
78.
79.
      ljmp Loop_Begin
80.
81.
      // if not
      Jump:
82.
83.
      ljmp Loop_Begin
84.
      // button even handler
85.
      /* 按鈕事件處理 Function,藉由此函示來定義回傳不同 state 給 Loop_Begin 來顯示不
      同的 LED Output 狀態*/
86.
      handleOnclick:
      mov R1, P1 // set the input val to reg.
87.
88.
      mov A, R1
      event1: // right rotate
89.
      cjne A, #10000000b, event2
90.
91.
      mov 20h, #1
      ret
92.
93.
      event2:// left rotate
94.
      cjne A, #01000000b, event3
95.
96.
      mov 20h, #2
97.
      ret
98.
      event3:// even num blink
99.
      cjne A, #00100000b, event4
```

```
100.
      mov 20h, #3
      mov B, #0 // this is quite important, because we need to inital the LED
101.
      array statt
102.
      ret
     event4:// blink odd num.
103.
      cjne A, #00010000b, return
      mov 20h, #4
105.
      mov B, #0 // like the upper one, or it will leave strange state
106.
107.
      return:
108.
     ret
109.
      // 以下為定義如何處理按鈕的 delay 以達到不會按一個按鈕太快跳到不同 state 的狀態
110. ; use the slow down the blick state, due to the cpu clock is quite fast
      Delay: mov R1,#50
111.
     Delay0: MOV R2, #30
112.
113.
      Delay1: MOV R3, #249
114.
     Delay2: DJNZ R3, Delay2
115.
     DJNZ R2, Delay1
116. djnz R1, Delay0
117.
      ret
118.
     end
```

# 4. 問題難點和解決方法

在進行實驗時,我發現了一個問題。當 delay 的時間設定得較長時,如果在這段時間內按下按鍵,由於系統可能正在執行 delay,它可能不會立即偵測到新的按鍵輸入。因此,使用者可能需要長時間按住按鍵才能確保系統能夠讀取到新的狀態。

為了呈現四種不同的狀態,我使用了一個變數 state 來儲存當前的狀態。當使用者按下按鍵時,系統會偵測並記錄所按的按鍵,並將其轉換為相對應的狀態。接著,系統會根據當前的 state 執行相對應的功能,如左移、右移、偶數閃爍、基數閃爍。如果沒有新的按鍵輸入,系統會繼續執行上一次的狀態動作。

### 5. 結論

整體來說,一旦了解了 8051 CPU 的架構,再加上之前的 Lab01 實驗,我已經對需要使用的指令有了基本認識。不過,我也發現了 delay 的問題,這是一個需要改進的地方。接下來在 Lab03 中,我們可以用 Timer 來解決這個問題來優化 program。