Rep\_Lab\_02\_B0829002

# 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。

# 先備知識

## SRF page register

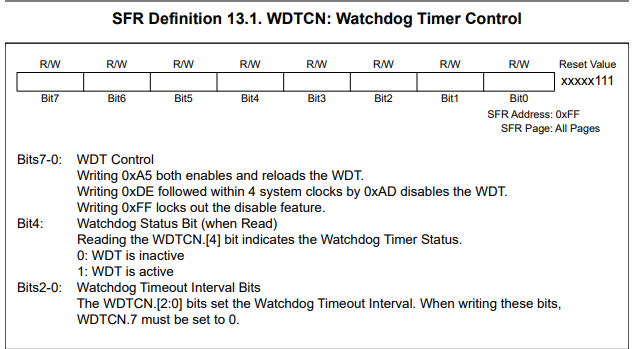
第一是要先找出SFR的位置，因為要先去定義PnMDIN和PnMDOUT用以作為按鈕和輸入和LED輸出，但因為在Datasheet上可以看到不是每個MDIN和MDOUT在每個page都可用，所以切到特定page。

一張含有 桌 的圖片

自動產生的描述

## 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0xDE和0xAD，可以關閉看門狗功能，確保程式可以正常運行而不會被不斷重啟。



## 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。

一張含有 文字, 螢幕擷取畫面, 文件, 字型 的圖片

自動產生的描述

# 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。

1. XBR2 equ 0e3h
2. P1MDIN equ 0adh
3. P2MDOUT equ 0a6h
4. WDTCN equ 0ffh
5. SFRPAGE equ 084h
6. P1 equ 090h
7. P2 equ 0a0h
8. CONFIG\_PAGE equ 0fh
9. LEGACY\_PAGE equ 00h
10. // close timer
11. mov WDTCN, #0deh
12. mov WDTCN, #0adh
13. // setup port config
14. mov SFRPAGE, #CONFIG\_PAGE
15. mov XBR2, #0c0h
16. mov P1MDIN, #0ffh
17. mov P2MDOUT, #0ffh
18. mov SFRPAGE, #LEGACY\_PAGE
19. mov P1, #00000000h // btn array reference to 090h
20. mov P2, #00000000h // output array reference to 0a0h
21. mov R0, #10000000b
22. mov B, #00000000b
23. // initial signal, status, btn and LED
24. mov 20h, #0
25. mov P1, #0
26. mov R1, #0
27. mov R2, #0

/\* 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。 \*/

1. Loop\_Begin:
2. lcall handleOnclick
3. // after btn state check, put val. to A, in order to do operation
4. mov R1, 20h
5. mov A, R1
6. rightRotate:
7. cjne A, #1, leftRotate // check event
8. lcall Delay
9. mov R3, A
10. mov A, R0
11. rr A
12. mov P2, A
13. mov R0, A
14. ljmp Loop\_Begin
15. // R\_L
16. leftRotate:
17. cjne A, #2, evenBlink // check event
18. lcall Delay
19. mov R3, A
20. mov A, R0
21. rl A
22. mov P2, A
23. mov R0, A
24. ljmp Loop\_Begin
25. // Blink
26. evenBlink:
27. cjne A, #3, oddBlink // check event
28. lcall Delay
29. mov R3, A
30. xrl B ,#01010101b
31. mov P2, B
32. ljmp Loop\_Begin
33. oddBlink:
34. cjne A, #4, Jump // check event
35. lcall Delay
36. mov R3, A
37. xrl B ,#10101010b
38. mov P2, B
39. ljmp Loop\_Begin
40. // if not
41. Jump:
42. ljmp Loop\_Begin
43. // button even handler

/\* 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\*/

1. handleOnclick:
2. mov R1, P1 // set the input val to reg.
3. mov A, R1
4. event1: // right rotate
5. cjne A, #10000000b, event2
6. mov 20h, #1
7. ret
8. event2:// left rotate
9. cjne A, #01000000b, event3
10. mov 20h, #2
11. ret
12. event3:// even num blink
13. cjne A, #00100000b, event4
14. mov 20h, #3
15. mov B, #0 // this is quite important, because we need to inital the LED array statt
16. ret
17. event4:// blink odd num.
18. cjne A, #00010000b, return
19. mov 20h, #4
20. mov B, #0 // like the upper one, or it will leave strange state
21. return:
22. ret

// 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

1. ; use the slow down the blick state, due to the cpu clock is quite fast
2. Delay: mov R1,#50
3. Delay0: MOV R2, #30
4. Delay1: MOV R3, #249
5. Delay2: DJNZ R3, Delay2
6. DJNZ R2, Delay1
7. djnz R1, Delay0
8. ret
9. 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。