Rep_Lab_02_B0829002

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

I. 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 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。

SFR Definition 13.1. WDTCN: Watchdog Timer Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								xxxxx111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xFF
SFR Page: All Pages

Bits7-0: WDT Control
Writing 0xA5 both enables and reloads the WDT.
Writing 0xDE followed within 4 system clocks by 0xAD disables the WDT.
Writing 0xFF locks out the disable feature.
Bit4: Watchdog Status Bit (when Read)
Reading the WDTCN.[4] bit indicates the Watchdog Timer Status.
0: WDT is inactive
1: WDT is active
Bits2-0: Watchdog Timeout Interval Bits
The WDTCN.[2:0] bits set the Watchdog Timeout Interval. When writing these bits, WDTCN.7 must be set to 0.

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

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	—	T4EXE	T4E	UART1E	EMIFLE	CNVST0E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xE3
SFR Page: F

Bit7: WEAKPUD: Weak PullUp Disable Bit.
0: Weak pullups globally enabled.
1: Weak pullups globally disabled.
Bit6: XBARE: Crossbar Enable Bit.
0: Crossbar disabled. All pins on Ports 0, 1, 2, and 3, are forced to Input mode.
1: Crossbar enabled.
Bit5: UNUSED. Read = 0, Write = don't care.
Bit4: T4EXE: T4EX Input Enable Bit.
0: T4EX unavailable at Port pin.
1: T4EX routed to Port pin.
Bit3: T4E: T4 Input Enable Bit.
0: T4 unavailable at Port pin.
1: T4 routed to Port pin.
Bit2: UART1E: UART1 I/O Enable Bit.
0: UART1 I/O unavailable at Port pins.
1: UART1 TX and RX routed to 2 Port pins.
Bit1: EMIFLE: External Memory Interface Low-Port Enable Bit.
0: P0.7, P0.6, and P0.5 functions are determined by the Crossbar or the Port latches.
1: If EMI0CF.4 = '0' (External Memory Interface is in Multiplexed mode)
P0.7 (/WR), P0.6 (/RD), and P0.5 (ALE) are 'skipped' by the Crossbar and their output states are determined by the Port latches and the External Memory Interface.
1: If EMI0CF.4 = '1' (External Memory Interface is in Non-multiplexed mode)
P0.7 (/WR) and P0.6 (/RD) are 'skipped' by the Crossbar and their output states are determined by the Port latches and the External Memory Interface.
Bit0: CNVST0E: ADC0 External Convert Start Input Enable Bit.
0: CNVST0 for ADC0 unavailable at Port pin.
1: CNVST0 for ADC0 routed to Port pin.

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。

```
4.      XBR2 equ 0e3h
5.      P1MDIN equ 0adh
6.      P2MDOUT equ 0a6h
7.      WDTCN equ 0ffh
8.      SFRPAGE equ 084h
9.      P1 equ 090h
10.     P2 equ 0a0h
11.
12.     CONFIG_PAGE equ 0fh
13.     LEGACY_PAGE equ 00h
14.
15.     // close timer
16.     mov WDTCN, #0deh
17.     mov WDTCN, #0adh
18.
19.     // setup port config
20.     mov SFRPAGE, #CONFIG_PAGE
21.     mov XBR2, #0c0h
22.     mov P1MDIN, #0ffh
23.     mov P2MDOUT, #0ffh
24.     mov SFRPAGE, #LEGACY_PAGE
25.
26.     mov P1, #00000000h // btn array reference to 090h
27.     mov P2, #00000000h // output array reference to 0a0h
28.
29.     mov R0, #10000000b
30.     mov B, #00000000b
31.
32.     // initial signal, status, btn and LED
33.     mov 20h, #0
34.     mov P1, #0
35.     mov R1, #0
36.     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。 */

```
37.     Loop_Begin:
38.     lcall handleOnclick
39.     // after btn state check, put val. to A, in order to do operation
40.     mov R1, 20h
41.     mov A, R1
42.
43.     rightRotate:
44.     cjne A, #1, leftRotate // check event
```

```
    lcall Delay
    mov R3, A
    mov A, R0
    rr A
    mov P2, A
    mov R0, A
    ljmp Loop_Begin

    // R_L
    leftRotate:
    cjne A, #2, evenBlink // check event
    lcall Delay
    mov R3, A
    mov A, R0
    rl A
    mov P2, A
    mov R0, A
    ljmp Loop_Begin

    // Blink
    evenBlink:
    cjne A, #3, oddBlink // check event
    lcall Delay
    mov R3, A
    xrl B ,#01010101b
    mov P2, B
    ljmp Loop_Begin

    oddBlink:
    cjne A, #4, Jump // check event
    lcall Delay
    mov R3, A
    xrl B ,#10101010b
    mov P2, B
    ljmp Loop_Begin

    // if not
    Jump:
    ljmp Loop_Begin

    // 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*/

```
    handleOnclick:
    mov R1, P1 // set the input val to reg.
    mov A, R1
    event1: // right rotate
    cjne A, #10000000b, event2
    mov 20h, #1
    ret

    event2:// left rotate
    cjne A, #01000000b, event3
    mov 20h, #2
    ret
    event3:// even num blink
    cjne A, #00100000b, event4
```

```
100.    mov 20h, #3
101.    mov B, #0 // this is quite important, because we need to inital the LED
        array statt
102.    ret
103.    event4:// blink odd num.
104.    cjne A, #00010000b, return
105.    mov 20h, #4
106.    mov B, #0 // like the upper one, or it will leave strange state
107.
108.    return:
109.    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

110.    ; use the slow down the blick state, due to the cpu clock is quite fast
111.    Delay: mov R1,#50
112.    Delay0: MOV R2, #30
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。