實驗七 計時/計數器

1. 學習重點

- 了解計時/計數器的性質
- 了解 8051 中與計時/計數器相關的暫存器
- 了解 8051 的計時/計數器中斷

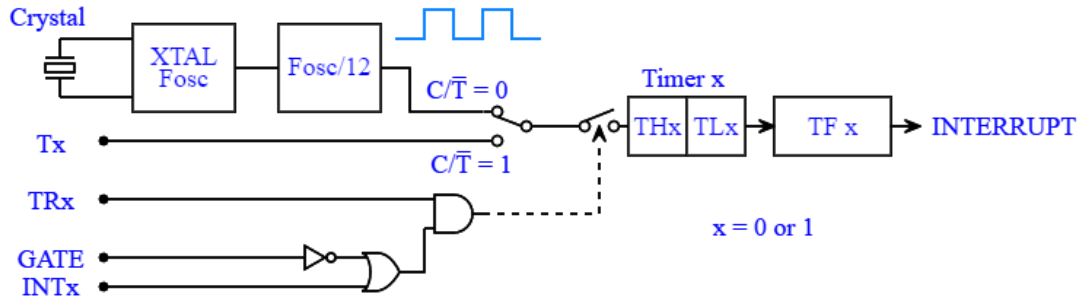
2. 材料清單

表 7-1、材料清單

器材名稱		數量
AT89S51		1
12MHz 石英震盪器		1
LED 二極體		8
按壓開關		2
電阻	1kΩ	9
	10 kΩ	1
電容	20pF	2
	10μF	1

3. 元件原理

計時/計數器 Timer/Counter



TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0
	Timer1				Timer0			

TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

圖 7-1、8051 計時/計數器邏輯分析圖

計時/計數器是一種常見的中斷裝置，其主要運作方式為持續累加一或多個暫存器，直到溢位時便觸發中斷，而計時與計數器的差別主要在於累加暫存器的時機 : 計時器依靠的是處理器的時脈計時，每個時脈都會累加計時的暫存器 ; 而計數器則是依靠接腳來接收外部的脈衝，利用脈衝的次數來計數。8051 提供的兩組的計時/計數器，而 8052 則提供了三組，以下與計時/計數器相關暫存器的說明以 8051 為主。

8051 中控制計時/計數器的暫存器主要有四個：TCON、TMOD、TH、TL，而與中斷相關的 IE、IP 則在前面的實驗已有介紹，此處不再贅述。因此若我們希望啟用計時/計數器中斷，我們需要設定至少 6 個暫存器，以下便依序來介紹這些暫存器。

計時/計數器控制暫存器（TCON 暫存器）

表 7-2、TCON 暫存器

	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON 暫存器為一個 8bits 的定址暫存器，僅後 4 bits 與計時/計數器相關，前 4 bits 為啟動其他中斷的設定使用。

- TFx 為 Timer flag，當 Timer x 的中斷觸發時，TFx 會被設為 1，當中斷結束後便會被設回 0。

- TRx 為啟用 Timer 的位元，當 TRx 為 1 時代表啟用 Timer x，反之則代表關閉 Timer x。

計時/計數器模式控制暫存器（TMOD 暫存器）

表 7-3、TMOD 暫存器

	7	6	5	4	3	2	1	0
TMOD	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
	Timer 1				Timer 0			

TMOD 暫存器為一個 8bits 的定址暫存器，兩組計時/計數器分別使用前四個與後四個位元。

- GATE 位元為控制計時/計數器是否為外部啟動，當 GATE 值為 1 時代表該計時/計數器的啟動並不只需要 TCON 暫存器的 TRx 位元設定為 1，還需要接腳 INTx 的電位為 HIGH 才可啟動；反之若 GATE 值為 0 時，僅需設定好 TRx 便可啟動計時/計數器。

- C/$\overline{T}$ 為計時器與計數器的選擇位元。當 C/$\overline{T}$ =0 時，Timer 為計時器，時脈來源為 crystal；當 C/$\overline{T}$ =1 時，Timer 則是計數器，時脈來源為 8051 的外部輸入腳。

- M1、M0 為決定計時/計數器模式的兩個位元，共四種模式，當 M1 為 0、M0 為 0 時則是 Mode 0，M1 為 1、M0 為 0 時則是 Mode 1，以此類推。
  Mode 0：該計時/計數器為 13-bits 的計時/計數器。
  Mode 1：該計時/計數器為 16-bits 的計時/計數器。
  Mode 2：該計時/計數器為具有自動載入功能 8-bits 的計時/計數器。
  Mode 3：該計時/計數器同時為一 8-bits 的計時器與一 8-bits 的計數器。

計時/計數器次數計算用暫存器（TH、TL 暫存器）

這兩個暫存器會依據計時計數器的 Mode 決定其功能及使用方法，在 Mode 0 和 Mode 1 中使用方法較為類似，皆是每個脈衝都會使 TL 的值加一，當 TL 溢位時則使 TH 加一，當 TH 溢位時則觸發中斷，差別在於 Mode 0 時 TL 僅使用了後 5 位元，Mode 1 時則使用了全部的 8 位元，因此 Mode 0 時可以將 TL 與 TH 視為一個 13-bits 的暫存器，當溢位時便觸發中斷，Mode 1 則是 16-bits 的暫存器。

在具有自動載入功能的 Mode 2，僅使用 TL 進行計時/計數，TH 的功能則變為當中斷觸發後，TL 需被設定的預設值，因在 Mode 1 和 0 時，當 TH 溢位觸發中斷後，TH 和 TL 都會變為 0x00，然而很多時候我們並不希望計時或計數的次數是從 0x00 開始計算直到溢位，因此在中斷結束後必須重新將 TH 和 TL 的值設定到我們希望的預設值；而 Mode 2 的自動載入功能便是省去重新設定 TH 和 TL 初始值的動作。

在 Mode 3 中，TL 是負責計時器的次數計算，TH 則是負責計數器的次數計算，TH 或 TL 暫存器溢位都會觸發中斷，差別只在於兩者的次數計算方式不一樣而已。

很多時候，為了追求較精準的計時，我們會希望在一個較好計算的脈衝次數後才進行中斷，再藉由中斷觸發的次數來精準計時，例如今天希望使用 Mode 1 的計時器計時一秒鐘，我們必須先計算出我們使用的設備時脈產生頻率為多少，本次實驗使用的石英震盪器頻率為 12MHz，而 8051 的時脈頻率為外部信號頻率除以 12，也就是說我們時脈產生的頻率為 1MHz，或是說每 10 萬分之一秒便會產生一次脈衝，若是我們使計時器每 5 萬次脈衝便中斷一次，20 次中斷後才執行我們希望計時一秒鐘後執行的程式碼，便能較精準的進行計時。為了達成 5 萬次脈衝中斷一次，我們需要將 TH 和 TL 設定為加上 5 萬後會恰好溢位的值，也就是

$$2^{16} - 50000 = 15536_{10} = 0011110010110000_2$$

將這串二進制的目標值前 8 位元放入 TL，後 8 位元放入 TH，便能使計時器在 5 萬的脈衝後觸發中斷。

4. 實驗內容

連接於 P2.0 的 LED 使用 delay function 作為閃爍的時間間隔；連接於 P2.1 的 LED 使用 Timer 0 的 interrupt 作為閃爍的時間間隔，調整參數,使兩個 LED 的閃爍時間間隔接近一致。

8051 程式-C 語言

本次實驗開始使用 C 語言，開啟一個新專案時，請加入 STARTUP.A51，如圖 7-2。

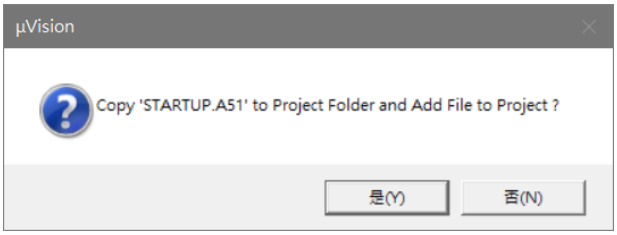


圖 7-2、Startup code

一般 8051 晶片都可以使用 header file reg51.h 或 reg52.h 來定義 8051 以及 8052 的暫存器，使用時在 C 語言程式中加入#include <reg51.h>或#include <reg52.h>。如圖 7-3，在 Keil C51 中以右鍵點選 Open document <reg51.h>，可以看到 header file reg51.h 的內容（如圖 7-4）。

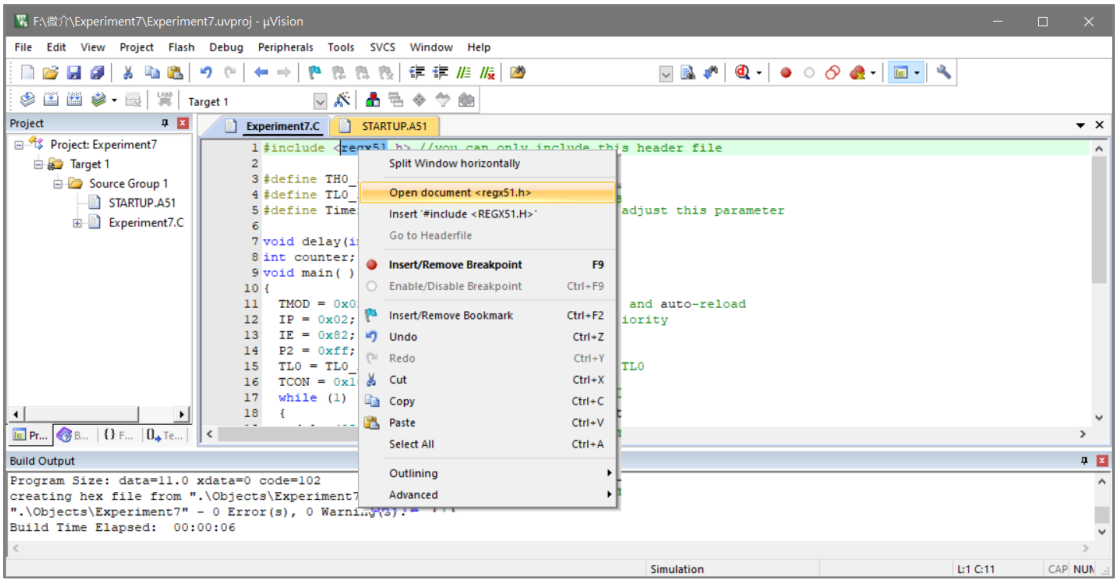


圖 7-3、header file - reg51.h

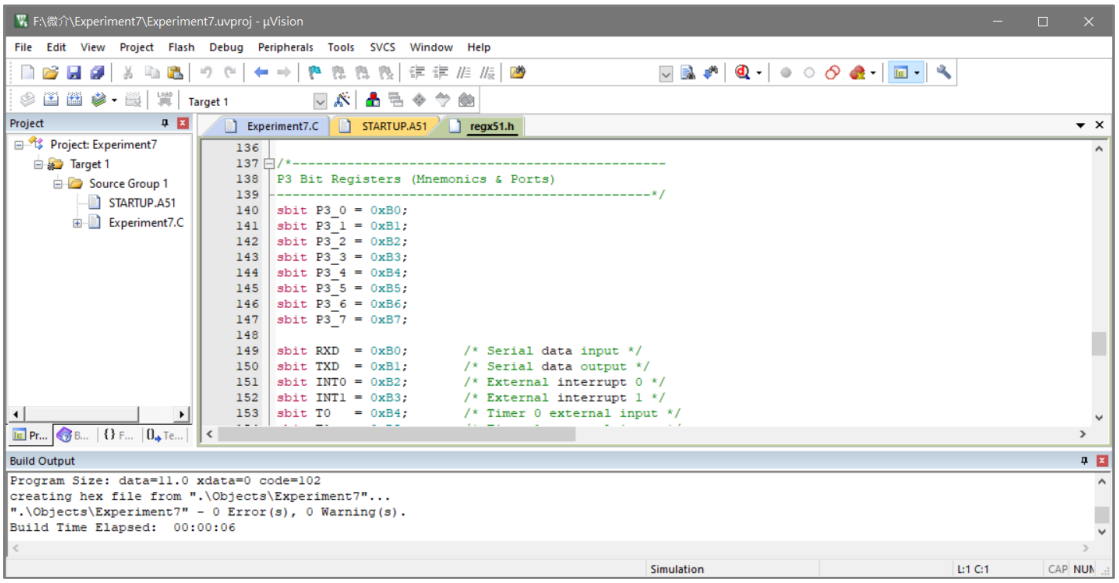


圖 7-4、header file - reg51.h

在 C 語言中，觸發中斷時會自動呼叫中斷函式，該函式有指定的宣告格式：

```
void 中斷函式名稱（void）interrupt 中斷型號
{
函式內容...
}
```

中斷型號可參考表 7-4，亦可在 header file reg51h 中找到 8051 的中段型號（如圖 7-5）。例如，欲啟用外部中斷 0，且中斷函式名稱命名為 ex0_interrupt，因此宣告函式 void ex0_interrupt(void) interrupt 0。

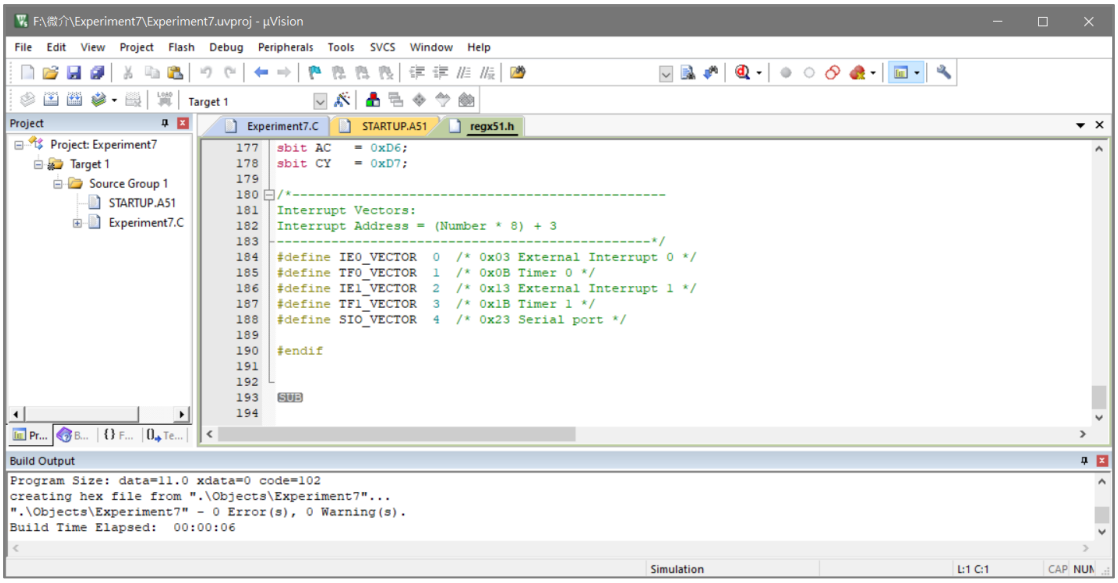


圖 7-5、reg51.h – Interrupt Vector

表 7-4、8051 中斷型號

中斷源	中斷向量	中斷型號
外部中斷 0	0x03	0
計時/計數器 0	0x08	1
外部中斷 1	0x13	2
計時/計數器 1	0x1B	3
串列埠	0x23	4

5. 實驗電路圖

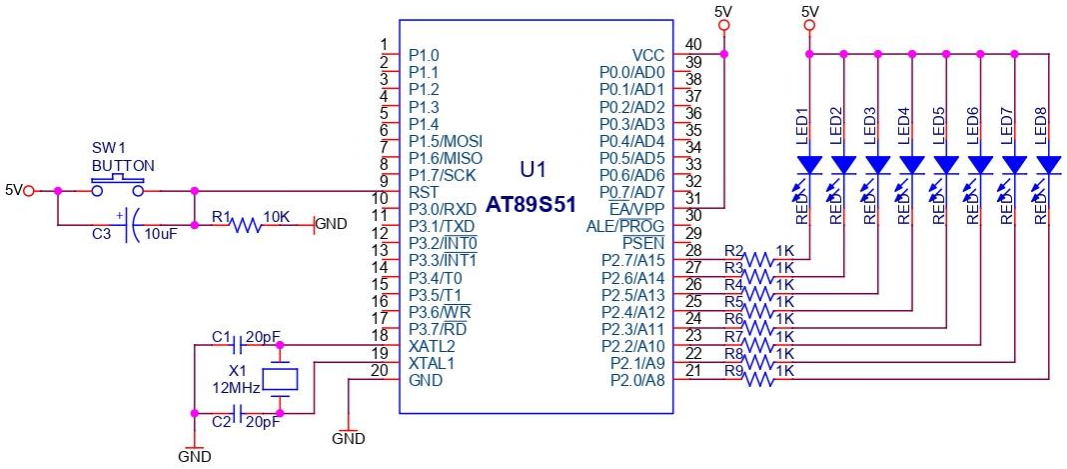


圖 7-2、實驗七基礎題參考電路圖

6. 軟體流程圖

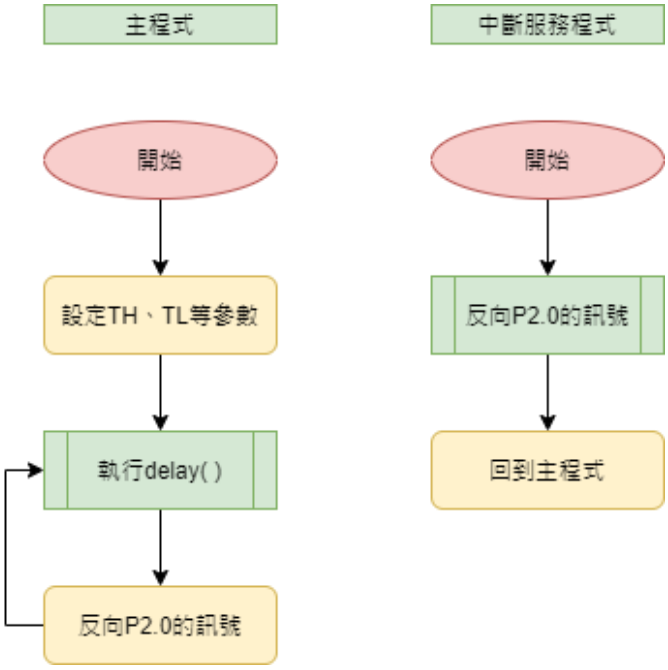


圖 7-3、實驗七基礎題參考軟體流程圖

7. 範例程式碼

```
#include <regx51.h>   // include header file for 8051

#define TH0_init 0x06 //TH0 = 256 - 250
#define TL0_init 0x06 //TL0 = 256 - 250
#define Timer0_int_exe_time 2000 //the parameter can be changed

void delay(int t);
int counter;
void main( )
{
   TMOD = 0x02;    //set timer0 to mode 2(8-bits and auto-reload)
   IP = 0x02;          //timer0 interrupt has high priority
   IE = 0x82;          //enable timer0 interrupt
   P2 = 0xff;
   TL0 = TL0_init; TH0 = TH0_init;   //set TH0 & TL0
   TCON = 0x10;      //enable timer0
   while (1)
   {
      delay(10000); // adjust this parameter to match timer
      P2_0 = ~P2_0; //inverse P2.0
   }
}

void timer0_interrupt(void) interrupt 1 // 'interrupt 1' is int vector of INT0
```

```
25  {
26      counter++;
27      if (counter == Timer0_int_exe_time) //250clock cycle * 2000 = 0.5 second
28      {
29          P2_1 = ~P2_1; //inverse P2.0
30          counter = 0;
31      }
32  }
33  //delay function
34  void delay(int t)
35  {
36      for (; t>0; t--) ;
37  }
```

8. 整理的題目，選擇/是非題