第 6 章　其他進階的應用

6-0 前言

本章將應用前面幾章介紹的軟、硬體方法與元件，來組合完成有趣或更實用的電路；另外，也將介紹目前正夯【3D 印表機】的主角 ---- 步進馬達，在明瞭其驅動原理後，進而學習如何控制應用。學習完本章之後，相信讀者將會更加確信 ---- 單晶片的應用，在日常生活中真的無所不在；若設計者能用心、細心去研究與嘗試時，就能創造出更實用與人性化的物件裝置，為忙碌的人群帶來生活的便利與樂趣。期待讀者能逐步進入設計者的領域。

6-1 聲音控制

相關知識

聲音產生的原理

記得在第三章中，曾經介紹蜂鳴器的驅動，只要利用通電與斷電的方式，就可以使蜂鳴器發出聲響或停止靜音，只是其聲音有點單調(因為電壓型的蜂鳴器，只能發出固定頻率的聲音)；當然啦！若使用喇叭或脈波型的蜂鳴器，就可以使發出的聲音變得較有變化，也就是 ---- 具有音階(scale，音的高低，常稱為音符)與節拍(beat，即拍子，也就是音的長短，以下簡稱為音長)的變化。

雖然可以利用時間延遲的方式產生類似頻率的聲音，但畢竟誤差會較大；還記得在第五章中，曾經應用『計時中斷』來計時，在此將利用它來產生較準確的音階，其方法為----每當產生計時中斷時，就使輸出狀態反相一次(若原為 Hi 就變為 Low，若原為 Low 就變為 Hi)，如此就可以產生所需頻率的方波(註)。如表(1)所示為 C 調低音、中音與高音三個音域的頻率對照表，依據表中的頻率數值，就可計算出計時中斷所需的延時參數。

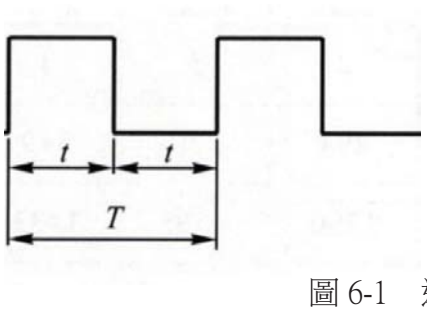
表(1)　C 調低、中、高音的頻率對照表

音階		1	2	3	4	5	6	7
		Do	Re	Mi	Fa	So	La	Si
高音	簡譜	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$	$\dot{6}$	$\dot{7}$
	頻率(Hz)	**1046**	1175	1318	1397	1568	1760	1976
中音	簡譜	1	2	3	4	5	6	7
	頻率(Hz)	**523**	587	659	698	784	880	988
低音	簡譜	$\underset{\cdot}{1}$	$\underset{\cdot}{2}$	$\underset{\cdot}{3}$	$\underset{\cdot}{4}$	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$
	頻率(Hz)	**262**	294	330	349	392	440	494

註：在此是由方波來組合成聲音，雖然與大自然界由弦波所組成的聲音有所不同，但對於人類耳朵而言，幾乎沒有差別。

延時參數的計算

若以中音的 Do 為例，其頻率約為 523Hz，所以其週期約為 1912 μs，而半個週期則為 956 μs，如圖 6-1 所示。



週期 $T = \frac{1}{f} \approx \frac{1}{523Hz} \approx 1912\,(\mu s)$

半週期 $t = \frac{T}{2} = \frac{1912}{2} = 956\,(\mu s)$

圖 6-1　週期與半週期的關係

以應用計時中斷 Timer 0 的 Mode 1 為例，若欲使每 956 μs 產生一次中斷，就必須將 16 bit 計數暫存器的值預設為 65536-956=64580 (=FC44H)，也就是將其高位元組數計暫存器 TH0 設為 0xFC、低位元組數計暫存器 TL0 設為 0x44，如此就可以發出中音的 Do 聲，其餘音階(符)的聲音皆可由此方法獲得；而表 6-1 所示為產生低、中、高音的延時參數表，提供往後讀者在設計程式時直接套用(註)。

註：若欲改為使用 Timer 1 計時中斷，只需將 TH0 改為 TH1、TL0 改為 TL1 即可。

表 6-1　　低、中、高音的延時參數表

音階			Do	Re	Mi	Fa	So	La	Si
高音	簡譜		$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$	$\dot{6}$	$\dot{7}$
	頻率(Hz)		1046	1175	1318	1397	1568	1760	1976
	半週期(μs)		478	426	379	358	319	284	253
	延時參數	TH0	0xFE	0xFE	0xFE	0xFE	0xFE	0xFE	0xFF
		TL0	0x22	0x56	0x85	0x9A	0xC1	0xE4	0x03
中音	簡譜		1	2	3	4	5	6	7
	頻率(Hz)		523	587	659	698	784	880	988
	半週期(μs)		956	852	759	716	638	568	506
	延時參數	TH0	0xFC	0xFC	0xFD	0xFD	0xFD	0xFD	0xFE
		TL0	0x44	0xAC	0x09	0x34	0x82	0xC8	0x06
低音	簡譜		$\underset{\cdot}{1}$	$\underset{\cdot}{2}$	$\underset{\cdot}{3}$	$\underset{\cdot}{4}$	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$
	頻率(Hz)		262	294	330	349	392	440	494
	半週期(μs)		1908	1700	1515	1433	1276	1136	1012
	延時參數	TH0	0xF8	0xF9	0xFA	0xFA	0xFB	0xFB	0xFC
		TL0	0x8C	0x5C	0x15	0x67	0x04	0x90	0x0C

歌曲的轉換

若只有音階(符)的變化，不足以構成好聽的音樂，需配合適當的節拍(拍子)，才能產生悅耳的旋律。當定義一個音長的時間為 125ms，且 1 拍有 4 個音長時，則 1 拍的時間等於 500ms；以此類推，4 拍、3 拍、2 拍、$\frac{1}{2}$ 拍、$\frac{1}{4}$ 拍，其音長分別為 16、12、8、2、1；以下以小蜜蜂歌曲的前 4 小節(以簡譜表示)為例，來說明兩者的關係。

歌曲簡譜為　| 5 3 3 - | 4 2 2 - | 1 2 3 4 | 5 5 5 - |

音符代碼為　5 3 3　4 2 2　1 2 3 4　5 5 5

拍子代碼為　4 4 8　4 4 8　4 4 4 4　4 4 8

所以，該歌曲可轉換寫成　5,4,3,4,3,8,4,4,2,4,2,8,1,4,2,4,3,4,4,4,5,4,5,4,5,8 的方式來表示，即第一個數字表示其音符，而第二個數字則表示該音符的拍子(以音長為單位)，也就是 ---- 偶數位置的數字皆為奇數位置(音符)的拍子；如此轉換是為了方便應用於程式中。(註)

註：在簡譜中，若音符下方畫有一條線，其音長減半（即變成 1/2 拍），若音符後加上一點，則長加半。

手腦並用時間

1. 電話鈴聲

本程式利用 Timer 0 Mode 1 計時中斷方式完成電話鈴聲，由於鈴聲是兩種頻率(在此以 300kHz 與 600Hz)所組成的聲音，因而程式需使 300Hz 與 600Hz 的聲音交替發出聲響 25ms，即可模擬出類似電話的鈴聲。如圖 6-2 所示為電話鈴聲的電路、程式與流程圖，該程式的功能為----產生 300Hz 與 600Hz 頻率的方波，兩者循環發出聲響 25ms，持續 1 秒後，靜音 2 秒，如此週而復始。以下就讓筆者來大略說明程式中指令的作用：

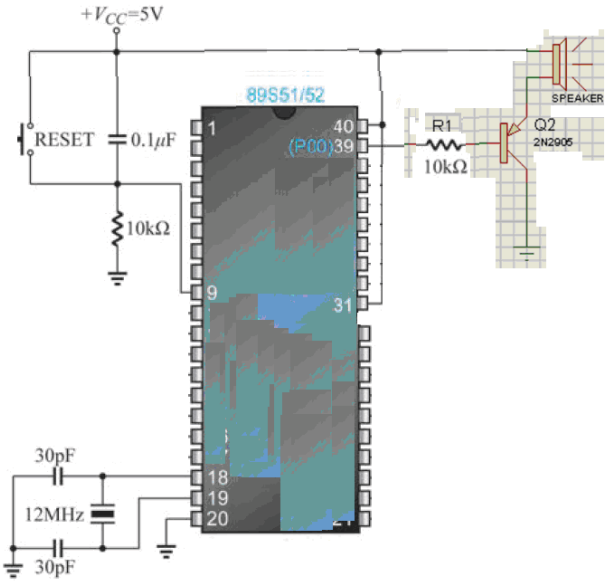


圖 6-2a　電話鈴聲的電路

電話鈴聲.c

```
1  /****  timer0中斷產生鈴聲(300Hz與600Hz組合) ****/
2  #include <regx52.h>
3  #define SP P0_0
4  unsigned char scale;
5  code char Ring_frequency[]={0xF9,0x7D,0xFC,0xBF};
6  /****  時間延遲    ****/
7  void delay_10us(unsigned int time)
8    {while(time>0) time--;}
```

```
/****  主程式   ****/
void main(void)
  {unsigned char i;
   EA=1;ET0=1;
   TMOD=0x01;
   while(1)
     {TR0=1; // 開始計時中斷,產生鈴聲
      for(i=0;i<20;i++) // 鈴響 1s
        {scale=0; //產生320Hz聲音
         delay_10us(2500); //25ms
         scale=1;  //產生480Hz聲音
         delay_10us(2500);  //25ms
        }
      TR0=0;              // 靜音 2s
      for(i=0;i<4;i++)
        {delay_10us(50000);}
     }
  }
```

```
/****  T0計時中斷    ****/
void timer0(void) interrupt 1
  {TH0=Ring_frequency[scale*2];
   TL0=Ring_frequency[scale*2+1];
   SP=!SP;
  }
```

圖 6-2b　　電話鈴聲的程式

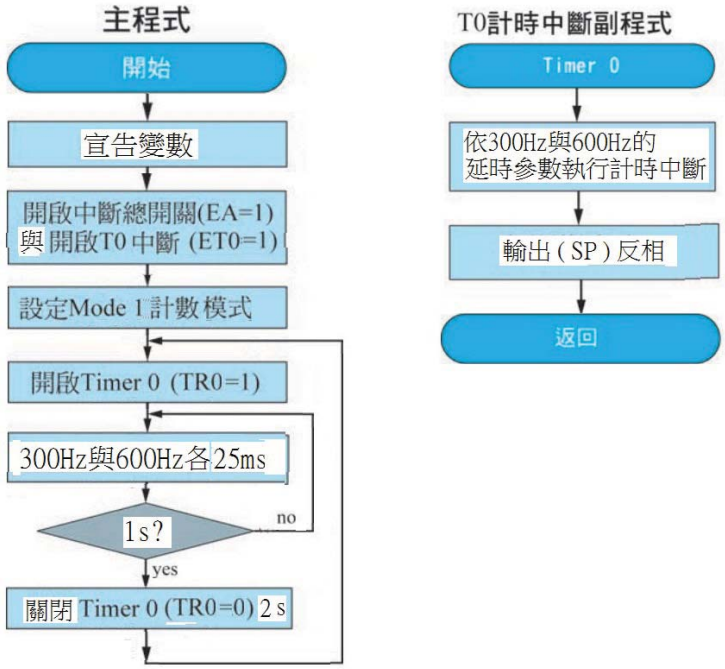


圖 6-2c　電話鈴聲的流程圖

第 3 列 SP 為輸出裝置(speaker)代號的設定，由 P0_0 輸出聲音。

第 4 列　宣告音階(scale，音符)為無號數字元變數。

第 5 列　宣告表格 Ring_frequency 為字元陣列並依序存放 300Hz 與 600Hz 的延時參數 0xF9、0x7D、0xFC、0xBF；當信號的頻率為 300Hz 時，其週期 $T=\frac{1}{f}=\frac{1}{300Hz}\approx 3333$ （μS)，半週期 $t=\frac{T}{2}\approx\frac{3333}{2}\approx 1667$ （μS)，所以其延時參數為 65536-1667＝$63869_{(10)}$＝$F97D_{(16)}$。當信號頻率為 600Hz 時，其週期 $T=\frac{1}{f}=\frac{1}{600Hz}\approx 1667$ （μS)，半週期 $t=\frac{T}{2}\approx\frac{1667}{2}\approx 833$ （μS)，所以其延時參數為 65536-833=$64703_{(10)}$=$FCBF_{(16)}$。

第 6～8 列 名稱為 delay_10us 的時間延遲副函數，每執行該副函數一次約需花費 10μs 的時間。

第 12 列 開啟所有中斷與 Timer 0 中斷的功能，可合併寫為 IE＝0x82。

第 13 列　由於 TMOD＝0x01，即 M1M0＝01，所以設定 Timer 0 計時工作為模式 1（Mode 1），16 bit 計數暫存器可計時的範圍為 0～65535 μS。

第 14 列　不斷執行第 15～25 列的指令。

第 15 列 啟動 Timer 0 的計時中斷，產生鈴聲。

第 16 列 鈴響 1s，即（10 μS x 2500 +10 μS x 2500）x 20＝1s。

第 17 列 配合計時中斷副函數中的第 29、30 列，取得 300Hz 的延時參數，使得高位元組數計暫存器 TH0＝0xF9、低位元組數計暫存器 TL0＝0x7D。

第 18 列 約 25ms 的時間延遲(等待)，其間每 1667 μs 即執行(產生)一次計時中斷副函數，也就是產生 300Hz 頻率的方波信號。

第 19 列 配合計時中斷副函數中的第 29、30 列，取得 600Hz 的延時參數，使得高位元組數計暫存器 TH0＝0xFC、低位元組數計暫存器 TL0 = 0xBF。

第 20 列 約 25ms 的時間延遲(等待)，其間每 833 μs 即執行(產生)一次計時中斷副函數，也就是產生 600Hz 頻率的方波信號。

第 22～24 列 關閉計時中斷(暫停鈴聲)約 2s，即 10 μS x 50000 x 4＝2s。

第 27～32 列 計時中斷副函數，依 scale 值取得 300Hz 或 600Hz 的延時參數，例如：當 scale＝0 時，高、低位元組數計暫存器 TH0、TL0 分別存入 0xF9、0x7D，因而產生 300Hz 的聲音；當 scale＝1 時，高、低位元組數計暫存器 TH0、TL0 分別存入 0xFC 、0xBF，因而產生 600Hz 的聲音。

第 31 列 每次執行使 SP 反相一次(0 變 1 或 1 變 0)，因而可產生 300Hz 或 600Hz 的方波。

另外，若覺得電話鈴聲不夠大聲，可將單一電晶體的放大電路改為達林頓對(darlington pair)的放大電路，如圖 6-3 所示，即可獲得較大音量的效果。

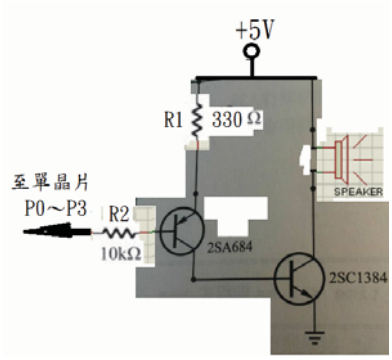


圖 6-3 達林頓對的放大電路

2. 電子音樂

以下圖 6-4 所示為小蜜蜂歌曲的簡譜，依照在前面介紹的方法，將該歌曲轉換可供程式讀取(使用)的資料如下：

C 4/4

| 5 3 3 - | 4 2 2 - | 1 2 3 4 | 5 5 5 - |
嗡嗡嗡 嗡嗡嗡 大家一起 勤作工

| 5 3 3 - | 4 2 2 - | 1 3 5 5 | 3 - - - |
嗡嗡嗡 嗡嗡嗡 做工趣味 濃

| 2 2 2 2 | 2 3 4 - | 3 3 3 3 | 3 4 5 - |
天暖花好 不做工 將來哪裡 好過冬

| 5 3 3 - | 4 2 2 - | 1 3 5 5 | 1 - - - |
嗡嗡嗡 嗡嗡嗡 別學懶惰 蟲

圖 6-4 小蜜蜂歌曲

第一列 | 5 3 3 - | 4 2 2 - | 1 2 3 4 | 5 5 5 - |

轉換成 5,4,3,4,3,8,4,4,2,4,2,8,1,4,2,4,3,4,4,4,5,4,5,4,5,8

第二列 | 5 3 3 - | 4 2 2 - | 1 3 5 5 | 3 - - - |

轉換成 5,4,3,4,3,8,4,4,2,4,2,8,1,4,3,4,5,4,5,4,3,16

第三列 | 2 2 2 - | 2 3 4 - | 3 3 3 3 | 3 4 5 - |

轉換成 2,4,2,4,2,8,2,4,3,4,4,8,3,4,3,4,3,4,3,4,3,4,4,4,5,8

第四列 | 5 3 3 - | 4 2 2 - | 1 3 5 5 | 1 - - - |

轉換成 5,4,3,4,3,8,4,4,2,4,2,8,1,4,3,4,5,4,5,4,1,16

、

由於喇叭發出聲音的電路與上一實習相同，所以在此省略其電路。如圖 6-5 所示為產生小蜜蜂歌曲的程式與流程圖，該程式的功能為 ---- 依歌曲簡譜的轉換資料，讀取其音階(音符)與節拍(拍子)，產生相對應的頻率與音長，使喇叭發出所需的聲音與節奏，因而產生美妙的音樂。以下就讓筆者來大略說明程式中指令的作用：

小蜜蜂.c

```
/**** 中音Do為1、中音Re為2、…、中音Si為7              ****/
/**** 低音Si為0、高音Do為8、高音Re為9、高音Mi為10     ****/
/**** 半拍為2、1拍為4、2拍為8、3拍為12                ****/
#include <regx52.h>
#define SP P0_0 //同 sbit SP=P0^0;
code char scale_table[]={0xfc,0x0c, //低音Si
                         0xfc,0x44, //中音Do
                         0xfc,0xac, //中音Re
                         0xfd,0x09, //中音Mi
                         0xfd,0x34, //中音Fa
                         0xfd,0x82, //中音So
                         0xfd,0xc8, //中音La
                         0xfe,0x06, //中音Si
                         0xfe,0x22, //高音Do
                         0xfe,0x56, //高音Re
                         0xfe,0x85}; //高音Mi

code char song[]={5,4,3,4,3,8,      4,4,2,4,2,8,   // 小蜜蜂 歌曲
                  1,4,2,4,3,4,4,4,  5,4,5,4,5,8,
                  5,4,3,4,3,8,      4,4,2,4,2,8,
                  1,4,3,4,5,4,5,4,  3,16,
                  2,4,2,4,2,4,2,4,  2,4,3,4,4,8,
                  3,4,3,4,3,4,3,4,  3,4,4,4,5,8,
                  5,4,3,4,3,8,      4,4,2,4,2,8,
                  1,4,3,4,5,4,5,4,  1,16,0xff}; //0xff為結束
unsigned char scale;
/****   時間延遲  ****/
void delay_10us(unsigned int time)
  {while(time>0) time--;}

void main(void)
  {unsigned char i,beat,j;//beat為節拍
   EA=1;ET0=1;
   TMOD=0x01;
   TR0=1;
   while(1)
     {scale=song[i];
      if(scale==0xff) // 停止
        {TR0=0;SP=1;while(1);}
      else
        {beat=song[i+1];
         for(j=0; j<beat; j++)
           {delay_10us(12500);} //以125ms為音拍基本單位
         TR0=0;  //斷開每一小節，暫停計時中斷
         delay_10us(12500);
         TR0=1; //恢復計時中斷
         i=i+2;
        }
     }
  }
```

```
50 /****  計時中斷  ****/ //取得音階參數
51 void timer0(void) interrupt 1
52   {TH0=scale_table[scale*2];
53    TL0=scale_table[scale*2+1];
54    SP=!SP;
55   }
```

圖 6-5a　　小蜜蜂歌曲的程式

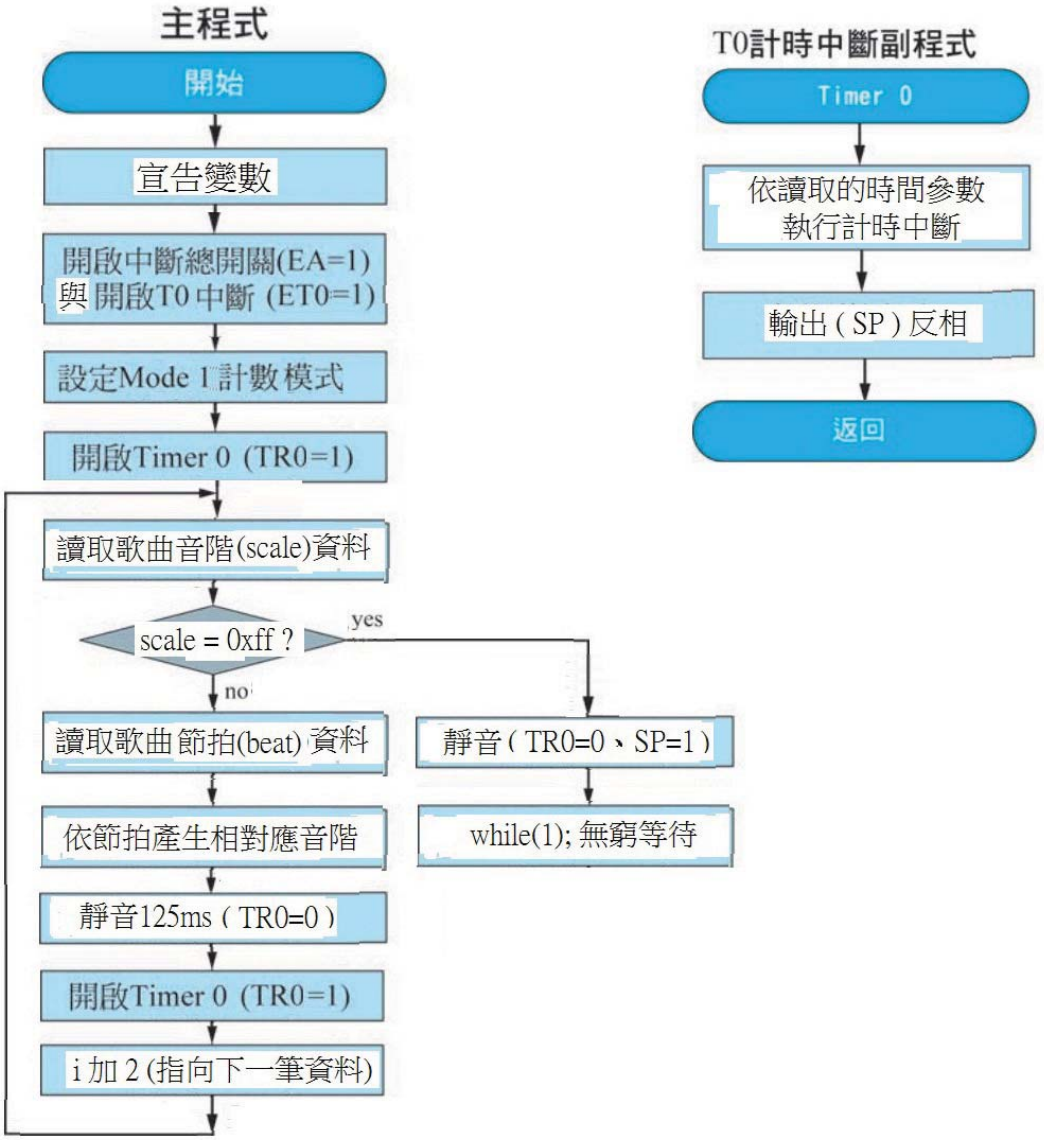


圖 6-5b　　小蜜蜂歌曲的流程圖

第 5 列　　SP 為輸出裝置（speaker）代號的設定，由此(P0_0)輸出小蜜蜂歌曲。

第 6～16 列　宣告(定義)音階(音符)表格 scale_table(低、中、高音)為字元陣列，並依序存放發出特定頻率聲音(Do、Re、Mi、…)的延時參數。

第 17～24 列　宣告(定義)小蜜蜂歌曲表格 song 為字元陣列，由於程式中的表格是由 0 開始(偶數)，所以在此應改為 ---- 表格 song 內的偶數位置

為音符，而其奇數位置為該音符的拍子；另外，在此自行定義 0xff 表示歌曲結束。

第 25 列　宣告音符(scale)為無號數字元變數。

第 26～28 列　名稱為 delay_10us 的時間延遲副函數，每執行該副函數一次約需花費 10μs 的時間。

第 32 列　開啟所有中斷與 Timer 0 中斷的功能，可合併寫為 IE＝0x82。

第 33 列　由於 TMOD＝0x01，設定 Timer 0 計時工作模式 1 (Mode 1)，16 bit 計數暫存器可計時的範圍為 0～65535 μs。

第 34 列　啟動 Timer 0 的計時中斷，開始產生歌曲的聲音。

第 35　不斷執行第 36～48 列的指令。

第 36～38 列　讀取一個 song 表格資料(即音符資料，在程式表格的偶數位置)，若資料為 0xff 就停止計時中斷(結束歌曲)，若不是則依讀取的音符代碼，由第 52、53 列取得相對應的延時參數，產生音符所需的頻率(聲音)。

第 39 列　若讀取的資料非 0xff，則執行第 40～47 列的指令。

第 40～42 列　若讀取的資料非 0xff，則再讀取一個 song 表格資料，產生相對應的節拍(拍子)；在此以 125ms 為音長時間單位，一拍則有 4 個音長，即 0.5s 時間。

第 43～44 列　每個聲音結束後，即靜音 125ms 時間，使每個聲音有分(斷)開的感覺。

第 45 列　恢復計時中斷(準備繼續產生下一個聲音)。

第 46 列　讀取下一筆聲音資料；由於每筆聲音資料包含音符、拍子資料，所以加 2。

第 50～55 列　計時中斷副函數，依 scale 值取得相對應聲音的延時參數，例如 scale＝5 時，由於高、低位元組數計暫存器 TH0、TL0 分別存入 0xFD、0x82，因而發出中音 So 的聲音。每執行一次計時中斷副函數，就使 SP 值反相一次，因而產生歌曲所需聲音的頻率。

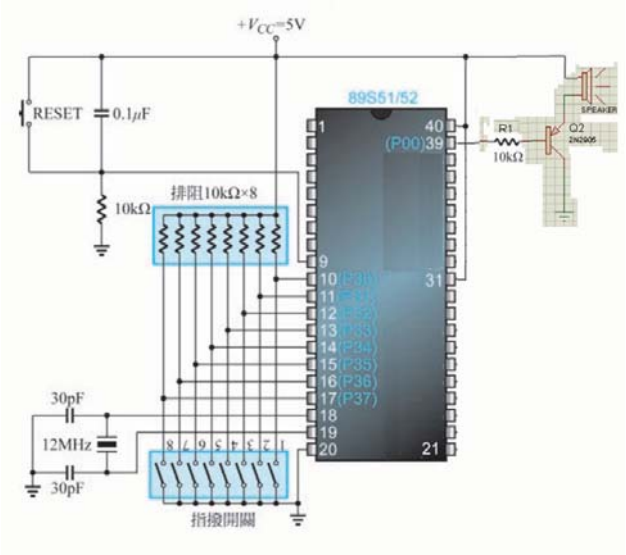
不曉得讀者學習完本小節後，有沒有深深覺得單晶片的應用真的很有趣，除了控制功能外，竟然還會唱歌(演奏音樂)。學好上面的例子，底下的練功時間就可以大展身手一番；多做、多想、多問問，持之以恆，是提升程式設計功力的捷徑(不二法門)哦！

練功時間

1. 嘗試將圖 6-2a 電路的 PNP 電晶體改為 NPN 電晶體，看看(聽聽)有何不同？原因為何？
2. 將電話鈴聲的兩種頻率 300Hz 與 600Hz，改為 265Hz 與 350Hz，並使每個頻

率各發出 0.5 秒聲響，循環持續不斷，聽聽看類似什麼聲音？

3. 利用指撥開關，組成一個電子琴電路，如圖(1)所示；其功能如下：只撥出 1 的指撥開關時，發出 Do 的音，只撥出 2 的指撥開關時，發出 Re 的音，以此類推。



圖(1)

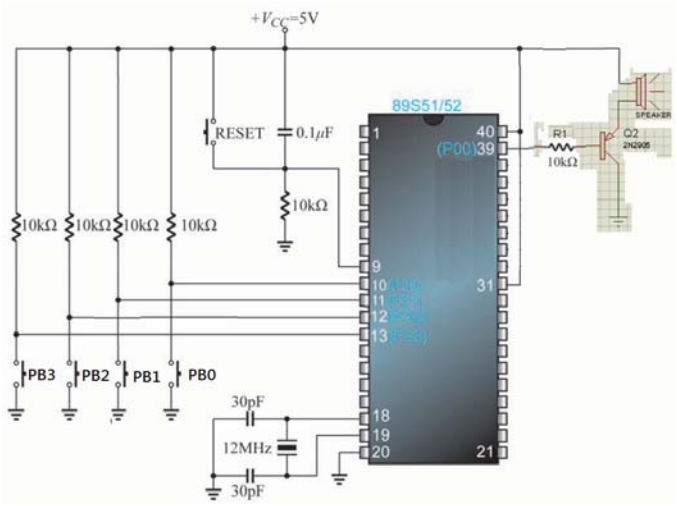
4. 嘗試將電子音樂的程式(小蜜蜂歌曲)改為演奏『兩隻老虎』或其他歌曲。

圖(2) 兩隻老虎的歌曲

5. 嘗試將範例中程式的時間延遲副函數改用計時中斷方式，使歌曲的拍子更為準確。

   提示：使用 Timer 1，參考 5-2 節 10 分鐘倒數計時程式。

6. 利用按鈕開關(或指撥開關)，組成一個點唱機電路，如圖(3)所示，每當按下不同開關時，就會演奏不同的歌曲。



圖(3)

7. 嘗試讓電子音樂(歌曲)不斷循環演奏。
8. 嘗試找出有休止符的歌曲並轉換成電子音樂程式。