義隆電子股份有限公司
ELAN MICROELECTRONICS CORP.

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

提昇軟體效率的小程式

分享

筆者閒暇時總喜歡一個人窩在房裡拿烙鐵，焊電路板，在網路上游走，看到喜歡的 DIY 也一定仔細端詳，既使按圖施工也可以得到不少的樂趣，相信酷愛此道的人應該也不少，除了喜歡看看別人的作品，也可以互相比較一下看誰用的零件少，誰提供的功能強，誰的速度最快，所以經常很容易就蒐集到一些不錯的電路，日子久了就像堆積木一樣，可以一個方塊一個方塊的拿來用，吾人戲稱為積木設計法。將許多有用的電路組合在一起，又是一個新的東西。這種方式的確又快又經濟，符合現代人速食的觀念。不僅是硬體可以像堆積木一樣的收集起來，軟體當然也可以適用於積木法則，於是在不少有心人的努力之下，筆者也收集了 EM78 系列單晶片一些很好的程式庫，所以說麻雀雖小，五臟俱全。也因為這些程式庫極具參考價值，筆者不忍獨享，故決定將紊亂的筆記重新整理後公開出來，與熱愛此系列單晶片的朋友們一同分享。

EM78XXX 單晶片自從問世以來已經陸續推出十餘種不同等級的單晶片，小到 8Pin 的 78P152，大到 100Pin OTP 的 78P860，其組合語言指令都是一樣的，僅有 57 個，所以反覆練習幾次就能熟悉指令的用法。組合語言用在 I/O 控制非常容易，也有很高的效率，所以坊間的書籍大部份以討論控制為主顯，顯少專門探討軟體技巧的篇幅，其實老手都知道，關於晶片之控制往往用到時再去翻一翻 DATA BOOK，注意一下 TIMING，然後準備一部示波器，三兩下就可以搞定。反倒是演算法用的好不好會大大影響產品的穩定度，所以有經驗的程式設計師通常都有自己的一套睽花祕笈，所以要提昇自己的功力最好的方式除了多練習之外，看看別人的程式也會使你進步很快。

※BCD 轉換成 Binary※

由於 EM78XXX 是 8 位元的微控器，因此為了節省記憶體，我們的範例僅以一個 BYTE 存放兩位 BCD 數為例，數字的範圍在 0~99 之間，轉換後的結果放在 ACC，如果您需要更多的位數，相信您在看完之後應該不難自行修改才是。

程式一：這個範例程式共花費 13 個指令 CYCLE，需要兩個變數空間，執行後會影響到原 BCD 的內容。

```
MOV     A,BCD
MOV     TMP,A
MOV     A,@0x0F
AND     TMP,A
SWAP    BCD
AND     BCD,A
BC      PSW,0
RLC     BCD         ; *2
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
MOV     A,BCD
ADD     TMP,A
RLC     BCD
RLCA    BCD         ; *8
ADD     A,TMP
```

說明:在程式一中所採用的方式應該算是最多人知道的方式,也是一種最直覺的方法,先將 BCD 個位數保存起來,因為十位數必須要乘以 10,所以利用移位的技巧乘以 10 再加上個位數,所得的答案放入 ACC。

程式二:在程式一的缺點,就是在執行程式以後,原本 BCD 的內容已經在移位的過程中被破壞掉了,為了改善這項缺失,我們換一種方式看看。下面這個程式,我們企圖改善前面的缺失,共花費 11 個指令 CYCLE,仍需要兩個變數空間,但是執行後不會破壞原來 BCD 的內容。

```
SWAPA   BCD
MOV     TMP,A
MOV     A,@0x0F
AND     BCD,A
AND     TMP,A
BC      PSW,0
RLCA    TMP
SWAP    TMP
RRC     TMP
ADD     A,TMP
ADD     A,BCD
```

(3) 對於程式二的結果我們仍然不滿意,似忽稍嫌複雜,雖然速度有所改善,但在記憶體的分配上仍有餘地,所以我們再改善成程式三的型態。轉換過程只花費 10 個指令 CYCLE,而且只需要一個變數空間,執行之後也不會改變原來 BCD 的內容。

程式三:

```
MOV     A,@0x0f
AND     A,BCD
JBC     BCD,4
ADD     A,@10
JBC     BCD,5
ADD     A,@20
JBC     BCD,6
ADD     A,@40
JBC     BCD,7
ADD     A,@80
```

說明:看過以上三個範例,您是否覺得程式三最簡潔而且容易瞭解?的確寫程式是一項極具挑戰性的工作,而且還可以找到很多靈感及樂趣,想不到吧!

※Binary 轉換成 BCD 碼※

下面的範例程式會將存放在 ACC 內的二進位數轉換成兩位 BCD 碼 (Compacted BCD Code),可轉換最大的 BCD 碼是 99。

```
    CLR     BCD
DIGIT_HI:
    ADD     A,@256-10
    JBS     PSW,FC
    JMP     DIGIT_LO
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
    INC     BCD
    JMP     DIGIT_HI
DIGIT_LO:
    ADD     A,@10
    SWAP    BCD
    OR      BCD,A
```

※減法的陷阱※

EM78 系列組合語言的減法指令是 SUB，使用這個指令時您得特別注意，因為 ACC 永遠都是減數，不可為被減數。SUB 指令的語法有以下三種：

```
    SUB     A,R (R-A→A)
    SUB     R,A (R-A→R)
    SUB     A,K (K-A→A)
```

也就是說如果我們想計算 A-2 的值，如果寫成：

```
    SUB     A,@2
```

其實是執行 2-A，解決方法如下：

```
    ADD     A,@256-2
或
    ADD     A,@254
```

※交換兩組暫存器的內容※

如果你覺得要交換兩組記憶體的內容一定要借用第三組變數，那麼您可以參考以下的方式，只是用了一些數學技巧就變得又快又簡單。

```
    MOV     A,REG1
    SUB     A,REG2
    ADD     REG1,A
    SUB     REG2,A
```

原理說明：

A=REG1

A=REG2-REG1

REG1=REG1+A

=REG1+(REG2-REG1)

=REG2

REG2=REG2-(REG2-REG1)

=REG1

※若 X>Y 就交換...※

延續上一個例子，此法用應用在 Bubble Sort 特別管用。

```
    MOV     A,X
    SUB     A,Y
    JBC     PSW,FC
    JMP     NO_CHANGE
    ADD     X,A
    SUB     Y,A
```

※2 補數※

2 補數加法經常代替減法，傳統上的做法是先取 1 補數，然後加 1。

```
    COM     REG
    INC     REG
```

或是可以利用另一種方式求得，所不同的是第二種方式會影響 PSW 暫存器。

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN
N.RD.SEC, 2. TAIPEI.
TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1,
SCIENCE-BASE INDUSTRIAL PARK,
HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
ADD     A,REG
SUB     A,REG
```

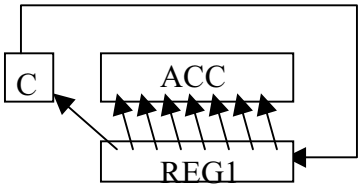
如果您所要求的數已經放在 ACC 裡面，那只要一行就能解決了。

```
SUB     A,@0
```

※旋轉位元組運算※

在 8051 指令中位元左旋有 RLC 與 RL 兩種指令區分,RLC 在 ACC 左旋時會連代將 CY 一併旋轉，而 RL 只會將 ACC 的 MSB 旋入 LSB。EM78XXX 指令只有 RLC,那麼要如何才能做到不帶 CY 旋轉呢?答案是旋轉兩次：

```
RLCA    REG1
RLC     REG1
```



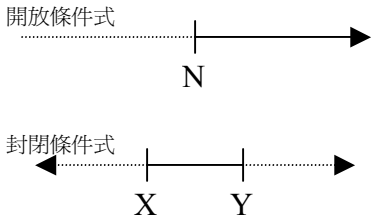
圖一

如圖一所示,第一次位元旋轉並沒有真正改變 REG1 的內容，目的是將 REG1 的 MSB 先放入 FC，第二次位元旋轉才將剛剛放在 FC 內的 MSB 旋入 LSB。同理，兩個 BYTES 不經 FC 的位元旋轉也是相同的原理。

```
RLCA    HI_BYTE
RLC     LO_BYTE
RLC     HI_BYTE
```

※範圍判斷※

寫程式免不了會碰到 IF..THEN.. 的場合,有些人覺得 EM78XXX 的條件判斷式太過繁瑣,所以筆者也將它們整理歸納一下。條件判斷式可分為開放區間條件式與封閉區間條件式來討論,以圖二來表示：



圖二

開放條件式是以 N 點為出發點,當待測值大於 N 或是小於等於 N 時的條件判斷，以 C 的語法描述如下：

```
if(number>n)
        ...    /* number 大於 N */
else
        ...    /* number 小於等於 N */
```

EM78XXX 組合語言寫法如下：

```
    MOV    A,@N+1
    SUB    A,Number
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
    JBC   PSW,FC
    JMP   LABEL_1          ; 大於N
    JMP   LABEL_2          ; 小於等於N
```

封閉式條件判斷是指待測值N是否在X與Y的範圍之內，若以C的語法描述：

```
if((number>=x) && (number<=y))
   ....  /* in range */
else
   ....  /* False */
```

如何以EM78組合語言做到呢?一般做法是以減法後的PSW做條件判斷,程式如下：

```
    MOV     A,@2
    SUB     A,number
    JBS     PSW,FC
    JMP     FALSE
    MOV     A,@y+1
    SUB     A,SI
    JBC     PSW,FC
    JMP     FALSE
IN_RANGE:
    ; ....
FALSE:
    ; ....
```

這個IF條件式要花費8個指令Cycle，還不算太複雜。但是還有個更簡潔的方法,以下用加法後的PSW(R3)做條件判斷,一共只要5行就清潔溜溜了。

```
    MOV     A,Number
    ADD     A,@255-y
    ADD     A,@y-x+1
    JBC     PSW,FC
    JMP     IN_RANGE
FALSE:
    ; ....
IN_RANGE:
    ; ....
```

說明：關鍵就在前三行，x表示條件式的下限值，y表示條件式的上限值，可以看得出仍是利用CY旗標製造的特效，不但精簡而且有點小聰明，許多老手都愛用,這也是他們口袋裡的秘密武器之一。如果您覺得不錯，不妨也收入錦囊中，爾後就可以依樣畫葫蘆了。

※ACC與暫存器內容交換※

這理我們要介紹一種快速的邏輯演算法， 只需要3個指令CYCLE，就可以將ACC的內容與暫存器的內容交換，不拖泥帶水，Very cute!

```
    XOR     Number,A
    XOR     A,Number
    XOR     Number,A
```

請讀者自行在紙上推算一次,就知道答案了。

※交換多組暫存器內容※

利用上面介紹的方法，可以推廣

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

到多組暫存器交換的例子上,下面的程式將 5 組 DATA 內容移位，第一筆暫存器的資料傳到第二筆暫存器內,第二暫存器的資料再傳送到第三筆暫存器內,依此類推,最後一筆資料則傳給第一個暫存器，形成一種位元組資料旋轉。

```
    MOV     A,@5
    MOV     COUNT,A
    MOV     A,@DATA1
    MOV     RSR,A
    MOV     A,DATA5
NEXT:
    XOR     INDIR,A
    XOR     A,INDIR
    XOR     INDIR,A
    INC     RSR
    DJZ     COUNT
    JMP     NEXT
```

※計算 MOD 2^N※

假如你剛好需要計算 ACC MOD X，且 X 剛好是 2 的 N 次方,使用 ACC AND (X-1) 是最快的方法了。例如要判斷 YEAR 是否為閏年,有個簡單的方法,可以排除一些非閏年的條件,只要不能被 4 整除者就不是閏年。所以可以用 YEAR AND 3 解決。

```
    MOV     A,@4-1
    AND     A,YEAR
    JBS     PSW,FZ
    JMP     NOLEAP
```

※清除一段連續的記憶體※

對於連續一段記憶體做讀寫最好的方式就是使用間接定址法,但是要注意在一些如 EM78447/811/860 等高階 MCU，記憶體 20H~3FH 又可以分成 4 組 BANK，如果之前沒有切換到正確的 BANK 會造成讀寫錯誤。下面的範例程式會將 BANK1 內的 32 個 BYTES 全部清為 0。

```
    INDIR == 0x00
    RSR   == 0x04
    COUNT == 0x10
    REG   == 0x20
    BANK1 == 0x40
    BANK2 == 0x80
    BANK3 == 0xC0
    MOV     A,@32
    MOV     COUNT,A
    MOV     A,@REG|BANK1
    MOV     RSR,A
NEXT:
    CLR     INDIR
    INC     RSR
    DJZ     COUNT
    JMP     NEXT
```

※計算一個 BYTE 中有多少個“1”※

這個小程式可以檢查出在某個 BYTE 中共有幾個 1,在某些演算法的過程可能會用得到，計算的結果放在 ACC。

```
    RRCA    DATA
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN
N.RD.SEC, 2. TAIPEI.
TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1,
SCIENCE-BASE INDUSTRIAL PARK,
HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
AND     A,@0x55
SUB     DATA,A
MOV     A,DATA
AND     A,@0x33
ADD     DATA,A
RRC     DATA
ADD     DATA,A
RRC     DATA
SWAPA   DATA
ADD     A,DATA
AND     A,@0x0F
```

※節省 NOP 指令的方法※

您還在為程式擠不下傷腦筋嗎? NOP 指令有時候在延遲指令時間很有用,假如你有連續兩個 NOP 指令可以用 JMP 到下一個指令的方式代替,因為這樣可以減少一個指令 BYTE,又可以達到相同的效果。

例如:

```
    NOP
    NOP
```

可以寫成:

```
    JMP     NEXT_INST
NEXT_INST:
    ;....
```

因為一個 NOP 花費一個指令 Cycle,但是一個 JMP 指令就需要 2 個指令 Cycle,雖然有時候會抱怨 JMP 指令會多花一點時間,但是想不到它也有如此妙用吧。

※LABEL 太多?※

寫組合語言最令人傷腦筋的問題之一就是稱是中到處是 label,這有兩個壞處,第一就是不小心就會造成 label 重複的問題,第二就是想不出適當的 label 名稱。如果您已經為 label 的命名問題腸枯思竭,給您提供一個小方法,程式中如果用「$」可以表示目前 PC 的位址,依此推論「$+2」表示 PC+2,「$-4」表示 PC-4,看看底下的例子您立刻就明白:

```
MOV     R,R
JBS     PSW,FZ
JMP     $+2
JMP     $-4
; ....
```

不過也要給您一個建議,label 有個要的意義就是具有註解的功能,特別是針對一些懶的寫註解的人格外重要。所以這個方法僅適合使用在重複性很高的程式片斷。

※SWITCH...CASE 敘述※

在程式設計的過程中,免不了常常會碰到多重選項的問題,利用 EM78XXX 的查表指令試試看,所以 TBL 除了當作一般查表指令以外,還可以當作多重條件判斷之用。

```
MOV     A,CASE
TBL
JMP     EVENT1      ; CASE=0
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
JMP     EVENT2      ; CASE=1
JMP     EVENT3      ; CASE=2
JMP     EVENT4      ; CASE=3
```

※多位元組的遞增及遞減運算※

因為 EM78XXX 是 8 位元的單晶片，如果要執行 8 位元以上的計算，必須將多個位元組看成是一個變數，以下我們舉例說明如何將一組 24 位元的變數，做到遞增及遞減運算。

遞增(Increment)：

```
MOV     A,@1
ADD     INT24,A
JBC     STATUS,FC
ADD     INT24+1,A
JBC     STATUS,FC
ADD     INT24+2,A
```

遞減(Decrement)：

```
MOV     A,@1
SUB     INT24,A
JBS     STATUS,FC
SUB     INT24+1,A
JBS     STATUS,FC
SUB     INT24+2,A
```

※判斷多位元組變數是否爲零※

使用簡單的邏輯運算指令，將多位元組 OR 在一起，然後依據 Z 旗標就可以判斷此多位元組變數是否為零了。

```
MOV     A,INT24
OR      A,INT24+1
OR      A,INT24+2
JBS     PSW,FZ
;...
```

※複製某些位元※

有時候我們需要將一些特定的幾個位元由某個暫存器複製給另一組暫存器，由於並非完全複製暫存器的內容，所以會多了一些抽取位元的步驟，現在我們找到一個方法，只要四個步驟就可以將指定的位元複製到另一組暫存器裡面，舉例說明，假設位元複製前(SOURCE)=44H，(TARGET)=5AH，如果我們希望將 SOURCE 的 BIT0~BIT2 複製到 TARGET，則執行程式後(SOURCE)=44H，(TARGET)=5CH。

```
MOV     A,SOURCE
XOR     A,TARGET
AND     A,@00000111B
XOR     TARGET,A
```

無論您希望複製哪幾個 BIT，只要將第三行程式 MASK 所需的位元即可。

※奇偶位元對調※

以下這段程式是根據 Dmitry Kiryashov 的演算法設計，假設原本 ACC 內所有位元的排列順序為 abcdefgh，交換後 ACC 順序變成 badcfehg，程式只有五行，頗耐人尋味。

```
MOV     REG,A
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
    AND     A,@0x55
    ADD     REG,A
    RRC     REG
    ADD     A,REG
```

※中斷程式不需保留 ACC 及 PSW 的方法※

中斷程式一定要保留 ACC 及 PSW 嗎?那倒未必!特別是如果您使用的是 EM78P152/156 之類的迷你級的 MCU，RAM SIZE 都特別小，如果您只需要讓 TCC 中斷做簡單的計數工作，只要小心使用指令,就可以避免中斷程式會破壞到 ACC 及 PSW。原因是有些指令並不會對 PSW 產生影響,有些指令不需要經過 ACC。首先設定好預除器，並且讓 TCC Free Run。下面的例子完全沒用到 ACC 及 PSW。

```
        ORG     0
        JMP     INIT
        ORG     8
TCCINT:
        BC      RF,TCIF   ;清除中斷旗標
        INC     COUNTER
        RETI
```

※Multiple Task 管理與狀態機※

Multiple Task 就是將 CPU 時間平均分配(也可以是不平均分配)給多個 Task,所以在程式中會有一個時間管理者，依照指定的時間對指定的 Task 服務，沒有分配到時間的 Task 必需等候時間到來才能執行。

```
TCCINT:
    MOV     R10,A
    SWAP    R10
    SWAPA   PSW
    MOV     R11,A
    INC     TASK
    MOV     A,@4
    SUB     A,TASK
    JBS     PSW,FC
    JMP     ENDINT
    CLR     TASK
ENDINT:
    BC      ISR,TCIF
    SWAPA   R11
    MOV     PSW,A
    SWAPA   R10
    RETI
;-------------------------
MAIN:
    MOV     A,@0x21
    CONTW
    CLR     TCC
    CLR     ISR
    MOV     A,@0x01
    IOW     IOCF
    CLR     TASK
START:
    MOV     A,TASK
    TBL
    JMP     TASK0
    JMP     TASK1
    JMP     TASK2
    JMP     TASK3
    JMP     TASK4
```

TAIPEI: 6TH FL, NO. 42 CHUNG SHAN N.RD.SEC, 2. TAIPEI. TAIWAN, R.O.C.
TEL:886-2-5223065 FAX:886-2-5516348
HSIN-CHU: 7F-1, NO.9, PROSPERITY RD.1, SCIENCE-BASE INDUSTRIAL PARK, HSIN-CHU, TAIWAN, R.O.C.
TEL:886-3-5787505 FAX:886-3-5779095

```
;-------------------------
TASK0:
     ; ....
     JMP      START
TASK1:
     ; ....
     JMP      START
TASK2:
     ; ....
     JMP      START
TASK3:
     ; ....
     JMP      START
TASK4:
     ; ....
     JMP      START
```

上面這個程式將 TCC 規劃為 62.5ms 中斷一次(系統震盪選用 32.768KHz)，所以 Task 每 62.5ms 會切換到下一個 Task，也就是說每個 Task 都能夠平均分享 CPU 的時間，這就是分時多工的原理。至於中斷程式部分不是必需的，可一情況決定是否要由 TCC 安排時間的管理。狀態機(State Machine)是根據目前所在的 State 所產生的條件，來決定下一個狀態，所以程式原理和上面這個例子大同小異，所不同的是，我們應該把標示為 TASKn 的 Label 視為一個單獨的 State，然後根據某些條件將最後面的 JMP 轉移到另外一個 State。在這裡時間控制也不一定要用到，視需求決定。例如：

```
TASK1:
     MOV      A,INPUT
     JBS      PSW,FZ
     JMP      TASK2
     JMP      TASK3
```

說明：如果 INPUT=0 的話，將由目前所在的 TASK1 轉移到 TASK3 執行，否則狀態轉移到 TASK2。

後記

戲法人人會變，只是巧妙各有不同，希望筆者提供的這些小技巧對於喜歡玩單晶片的讀者能夠有所助益，我們不僅只是強調硬體應該節省，在軟體技巧上也應改多發展一些好的演算法，如此才能雙管齊下，對症下藥。吾人期盼藉此拋磚引玉能激發您更多的創意，寫出更精簡的程式，也期盼您的指教。