8051 記憶體的規劃（一）
位元定址面面觀

文 / 旗威科技
編輯種籽 / Irene

還記得上一篇文章中我們提到有關於記憶體的規劃嗎？接下來一連串的文章是關於Data Memory 的規劃細節，而最先開始的是位元定址的介紹，並實際用一些程式範例來介紹。

位元定址對於8051 來說是一個相當好用，而且很節省Data Memory 的功能，它是將記憶體的寫入以單一位元（1 bit）的方式來儲存資料，常用的指令爲布林位元處理指令，不過在使用上還是有一些限制存在的。底下是我們所整理出來的一些重點：

壹、位元定址的使用限制：

A. 位元定址只能使用Data Memory的20H～2FH這個區段，超過這個區段就無法使用位元定址。

說明：

雖然說位元定址很好用，可是並不是所有的Data Memory都可以使用位元定址，我們看看下面兩段程式。

```
OK      REG     20H.0
ERR     REG     20H.1
;
START   MOV     R0,#0
        DJNZ R0,$
        MOV     SP,#50H
;
        MOV     P1,#FFH
        MOV     20H,#0
$LOOP   CALL ON
        MOV     A,20H
        CJNE A,#1,$LOOP
        CLR     P1.7        ;RLY1 ON
        JMP     $LOOP
;
ON  SETB OK
    CLR     ERR
    RET
```

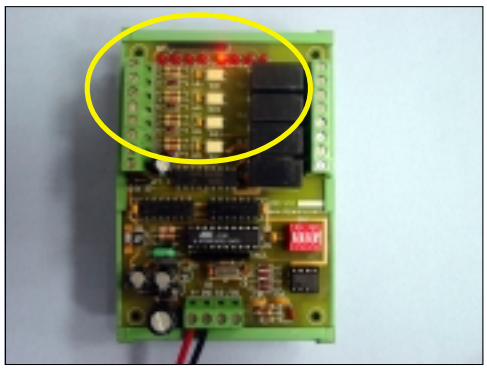
〔程式A-1〕正確的位元定址的寫法

```
OK      REG     30H.0
ERR     REG     30H.1
;
START   MOV     R0,#0
        DJNZ R0,$
        MOV     SP,#50H
;
        MOV     P1,#FFH
        MOV     30H,#0
$LOOP   CALL ON
        MOV     A,30H
        CJNE A,#1,$LOOP
        CLR     P1.7        ;RLY1 ON
        JMP     $LOOP
;
ON      SETB OK
        CLR     ERR
        RET
```

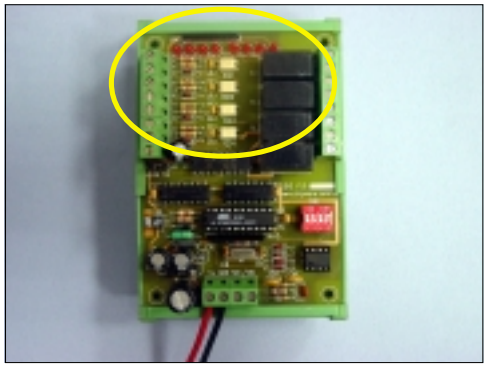
〔程式A-2〕錯誤的位元定址的寫法，與A-1程式的差別在於所宣告的位址不同。（藍色標示處）

這兩個程式的內容都一樣，差別只在於OK與ERR的旗標設定，A-1是用20H，A-2是用30H，這兩個程式會有什麼不同呢？我們利用旗威科技所研發的DIO-I控制板來做驗證。

執行A-1的程式時，RLY1順利啓動了；可是執行A-2的程式時，RLY1則是一動也不動。在語法的編輯上，由於我們所設定的OK與ERR都是位元定址的標準寫法，所以組譯器本身並不會發現這樣的寫法有錯誤，但身爲一個程式設計人員則必須要注意到這樣的細節。實際的執行結果也發現，30H並不能使用單一位元的定址方式，所以RLY1並不會啓動。



〔圖1〕DIO-I 執行 A-1 的狀況，RLY1 的 LED 是亮著的。



〔圖2〕DIO-I 執行A-2 的狀況，RLY1 的LED 並未點亮。

B． 同一位元組的不同位元不可直接搬移。

說明：

在8051中並沒有爲『搬移單一位元資料』這個功能設計特別的指令，因此在搬移一個位元（1 bit）的資料時，還是必須使用MOV的指令，以一個位元組（1 Byte）的搬移方式來搬移一個位元（1 bit）的資料。這樣講好像很模糊，我們實際用例子來說明好了。

```
OK      REG     20H.0
ERR     REG     20H.1
;
START   MOV     R0,#0
        DJNZ R0,$
        MOV     SP,#50H
;
        MOV     P1,#FFH
        MOV     20H,#0
$LOOP   CALL ON
        MOV     A,20H
        CJNE A,#3,$LOOP
        CLR     P1.7        ;RLY1 ON
        JMP     $LOOP
;
ON      SETB ERR
        MOV     OK,ERR
        RET
```

〔程式B-1〕直接搬移位元資料的程式寫法，如果位元設定在同一個位元組時，會發生搬移的錯誤。（藍色標示處）

```
OK      REG     20H.0
ERR     REG     20H.1
;
START   MOV     R0,#0
        DJNZ R0,$
        MOV     SP,#50H
;
        MOV     P1,#FFH
        MOV     20H,#0
$LOOP   CALL ON
        MOV     A,20H
        CJNE A,#3,$LOOP
        CLR     P1.7        ;RLY1 ON
        JMP     $LOOP
;
ON      SETB ERR
        MOV     C,ERR
        MOV     OK,C
        RET
```

〔程式B-2〕間接搬移位元資料的程式寫法，也是較好的程式寫法。（藍色標示處）

比較兩個檔案，不一樣的地方只有在呼叫副程式ON的地方，一個是利用直接搬移，另一個是利用間接搬移，這兩個程式同樣經過DIO-I控制板測試，左邊的程式沒有任何的動作，而右邊的程式則使RLY1啓動，原因是MOV的指令只能用來搬移一個位元組的資料，也就是說：MOV　20H.0,20H.1和MOV　20H,20H的動作是一樣的，這樣一來無論你怎麼搬動，資料就是搬不過去，所以在寫程式的時候一定要特別小心這樣的情況。

貳、活用位元定址的法則：

A．　位元定址適用於狀態的儲存與判斷。

說明：

一般來說，位元定址通常是用來儲存狀態用的，如果用一個Byte來儲存一個單純的True-False狀態，對記憶體空間的使用來說，是一種很浪費的用法，而位元定址的好處在於可以用一個bit來儲存資料，這樣一來，16個Bytes一共可以儲存128個狀態值，對於一般中小型的程式已經相當够用了！

B．　位元定址適合用來處理具有正負號的數值資料。

說明：

具有正負號的數值，其最高位元通常用來儲存正負號，如果處理運算這一類的數值，20H～2FH這個區段的Data Memory是最適合不過的，因爲我們可以利用SETB及CLR直接改變其正負號，也可以用JB與JNB來判斷正負號，這對算術運算來說是最快也最有效的方法。

C．　熟悉布林位元處理指令的使用限制。

說明：

不是所有的布林位元處理指令都必須在可位元定址的Data Memory中才能執行的。如JB與JNB就適用於全部的Data Memory，因爲這些指令並沒有『定址』的動作，它只是純粹的判斷動作而已，然而JBC是一個特例，它同JB與JNB一樣具有判斷位元資料的功能，此外，它還可以進行非位元定址的Data Memory之位元資料清除，不過因爲只能清除而已。熟悉這些指令的功能與用法，可以讓程式更精簡也更具彈性。

有了以上的概念，您是否對位元定址的用法有更深一步的認識了呢？當然，撰寫Assembler沒有絕對的方式，如果您有任何更好的意見及看法，歡迎隨時來函指教。

參考資料：

1.『8051單晶片徹底研究－基礎篇』／林伸茂著／旗標出版股份有限公司

2. DIO-I控制板使用手冊　／　旗威科技