**;-----------------------------------------------------------------**

**;【Define】 pseudo name**

**//用來自定義新的名稱或位置去存取,紀錄其他東西**

**;-----------------------------------------------------------------**

**port4 EQU 0E8h //定義port4為位置0E8h**

**shift7LED EQU 0FEh //目的使七字節每次只有一個點亮**

**dispBuf EQU 40h //最左LED為40h,往右為41h、42h、43h**

**keyBuf1 EQU 44h //存取按鍵0～7的輸入，若有輸入顯示0**

**keyBuf2 EQU 45h //存取按鍵8～F的輸入，若有輸入顯示0**

**keyCode EQU 46h //用來存取按鍵碼**

**keyCheck EQU 47h //檢查是否和keyCode一樣，有無取到跳躍信號**

**debCounter EQU 48h //用來檢查是否過了跳躍信號**

**debTimes EQU 60 //用來延遲時間**

**t0Buf EQU 49h //;RAM**

**t0Count EQU 40 //;data**

**decBuf0 EQU 4Ah //;RAM**

**t1Buf EQU 4Bh //;RAM**

**t1Count EQU 5 //;data**

**decBuf1 EQU 4Ch //;RAM**

**bitFlag0 BIT p1.7 //Bitmemory:0:up,1:down(設置LED P1.7)**

**bitFlag1 BIT p1.0 //Bitmemory:0:up,1:down(設置LED P1.0)**

**byteBCD0 EQU 4Dh //;RAM**

**byteBCD1 EQU 4Eh //;RAM**

**byteBCD2 EQU 4Fh //;RAM**

**byteHex1 EQU 50h //;RAM**

**byteHex0 EQU 51h //;RAM**

**byteTime EQU 52h //**

**byteTH0 EQU 53h //**

**byteTL0 EQU 54h //**

**bitwaveGen BIT P1.5 //**

**bitTimingGate BIT P1.0 //;1-sec timing gate square wave out**

**byteDigit EQU 55h //**

**byteTemp EQU 56h //**

**First\_num EQU 57h //57h~58h用來存輸入的第一個數值**

**Sec\_num EQU 59h //59h~5Ah用來存輸入的第二個數值**

**Answer EQU 5Bh //5Bh~5ACh用來存相加之值**

**Save\_whom EQU 5Dh //**

**Clear EQU 5Eh //判斷是否要執行 clear**

**ORG 0000h ;Start assembly program**

**;===========================================**

**LJMP start**

**ORG 000bh**

**LJMP T0\_isr ;進入Timer 0 中斷服務程式**

**ORG 0013h**

**LJMP int1\_isr ;進入外部中斷1 中斷服務程式**

**ORG 0023h**

**LJMP Serial\_ISR**

**Start:**

**SETB P3.0**

**SETB P3.3 ;INT1’ for 1-sec timing gate signal input**

**SETB P3.5 ;T1 for waveform signal input**

**SETB bitwaveGen**

**;-----------------------------------------------------------------**

**;【Main】 program**

**//主程式,包含許多數值的初始化以及各個副程式呼叫運行**

**;-----------------------------------------------------------------**

**MOV A, #0ffh**

**MOV P1,#0FFh**

**MOV P2, A ;P2 as input port**

**MOV P3, A ;P3 as input port**

**MOV keyBuf1, #0ffh ;initial no key (0-7) pressed**

**MOV keyBuf2, #0ffh ;initial no key (8-F) pressed**

**MOV keyCode, #0ffh ;final key code buffer (0-15)**

**MOV keyCheck, #0ffh ;check final key code buffer**

**MOV debCounter, #0 ;debouncing counter**

**MOV dispBuf, #10**

**MOV dispBuf+1, #10**

**MOV dispBuf+2, #10**

**MOV dispBuf+3, #0**

**ACALL ClearLED**

**ACALL ResetAll**

**MOV byteDigit,#0**

**MOV DPTR, #Led\_table**

**reset: MOV R0, #dispBuf ;display buffer**

**MOV A, #shift7LED ;the left-most 7LED on**

**loop: MOV port4, A**

**ACALL Covled ;7-seg LED pattern conversion**

**SJMP Going**

**Clearing: ACALL ClearLED**

**Going: ACALL Getkeycode ;get pressed key code in keyCode**

**ACALL Delaykeydebounce ;delay key debounce**

**ACALL DisplayDigit**

**MOV R4, Clear**

**CJNE R4,#0, Clearing**

**INC R0 ;next display buffer**

**RL A ;rotate to the next 7LED on**

**JB ACC.4, loop ;check ACC.4 = 1?**

**SJMP reset**

**;-----------------------------------------------------------------**

**;【Subroutine】 Interrupt Service Routine (ISR)**

**;External Interrupt 1**

**;-----------------------------------------------------------------**

**INT1\_ISR:**

**PUSH 0e0h ; push a**

**MOV byteHex0, Tl1**

**MOV byteHex1, TH1**

**ACALL hex2bcd**

**ACALL bcd2display**

**MOV Tl1, #0**

**MOV TH1, #0**

**POP 0e0h ; pop a**

**RETI ; 中斷服務程式之返回指令**

**;-----------------------------------------------------------------**

**;【Subroutine】 time 0 interrupt service routine**

**;-----------------------------------------------------------------**

**T0\_isr:**

**PUSH 0E0h ;push a**

**CLR TR0**

**MOV TH0, byteTH0**

**MOV TL0, byteTL0**

**SETB TR0**

**DJNZ t0Buf, t0isr\_exit**

**MOV t0Buf, byteTime**

**;====================== 時間到，Do something**

**CPL bitTimingGate**

**T0isr\_exit:**

**POP 0E0h ;pop a**

**RETI**

**;-----------------------------------------------------------------**

**;【Subroutine】 BCD(byteBcd2, byteBcd1, byteBcd0) ←**

**; Hex(byteHex1, byteHex0)**

**;-----------------------------------------------------------------**

**Hex2bcd:**

**PUSH 0e0h ; push a**

**MOV byteBcd0, #0**

**MOV byteBcd1, #0**

**MOV byteBcd2, #0**

**;-------------------------------------------------**

**; byteHex0 -> BCD -> (byteBcd1, byteBcd0)**

**;-------------------------------------------------**

**MOV A, byteHex0**

**MOV B, #100**

**DIV AB**

**MOV byteBcd1, a**

**MOV A, B**

**MOV B, #10**

**DIV AB**

**SWAP A**

**ORL A, B**

**MOV byteBcd0, a**

**;-------------------------------------------------**

**; byteHex1 -> BCD ->(byteBcd2, byteBcd1, byteBcd0)**

**; 處理 (56 + 200) ，留意 BCD**

**;-------------------------------------------------**

**MOV A, byteHex1**

**JZ Hex2bcd\_exit**

**MOV R2, A**

**MOV A, byteBcd0**

**Hex2bcd\_loop1:**

**ADD A, #56h ; 留意 BCD**

**DA A**

**JNC hex2bcd\_skip1**

**MOV R1, #byteBcd1**

**MOV B, #1**

**ACALL bcd\_daa**

**Hex2bcd\_skip1:**

**MOV R1, #byteBcd1**

**MOV B, #2**

**ACALL bcd\_daa**

**DJNZ R2, hex2bcd\_loop1**

**MOV byteBcd0, A**

**;-----------------**

**Hex2bcd\_exit:**

**POP 0E0h ; pop a**

**RET ; 服務程式之返回指令**

**;-----------------------------------------------------------------**

**;【Subroutine】 bcd daa, @R1, B**

**;-----------------------------------------------------------------**

**Bcd\_daa:**

**PUSH 0e0h**

**MOV A, @R1**

**ADD A, B**

**DA A**

**MOV @R1, A**

**JNC bcd\_daa\_skip**

**INC R1**

**INC @R1**

**bcd\_daa\_skip:**

**POP 0e0h**

**RET ; 服務程式之返回指令**

**;-----------------------------------------------------------------**

**;【Subroutine】(dispBuf, dispBuf+1)←byteBcd1**

**; (dispBuf+2, dispBuf+3)←byteBcd0**

**;-----------------------------------------------------------------**

**Bcd2display:**

**PUSH 0E0h**

**MOV A, byteBcd1**

**PUSH 0E0h**

**ANL A, #0fh**

**MOV dispBuf+1, A ; (dispBuf+1)←low nibble of byteBcd1**

**POP 0E0h**

**SWAP A**

**ANL A, #0fh**

**MOV dispBuf, A ; (dispBuf)←high nibble of byteBcd1**

**MOV A, byteBcd0**

**PUSH 0e0h**

**ANL A, #0fh**

**MOV dispBuf+3, A ; (dispBuf+3)←low nibble of byteBcd0**

**POP 0e0h**

**SWAP A**

**ANL A, #0fh**

**MOV dispBuf+2, A ; (dispBuf+2)←high nibble of byteBcd0**

**POP 0e0h**

**RET ; 服務程式之返回指令**

**;-------------------------------------------**

**;【Subroutine】 get key code, keyBuf1 <- P2, keyBuf2 <- P3,**

**;Use R2 as checking counter, then the final key code is**

**;keyCode <- R2**

**;-----------------------------------------------------------------**

**Getkeycode:**

**PUSH 0E0h ;push a**

**MOV A, P2 ;input from p2**

**MOV keyBuf1, A ;keyBuf1 for code (0-7)**

**MOV A, P3 ;input from p3**

**MOV A, #0FFh**

**MOV keyBuf2, A ;keyBuf2 for code (8, 9, a-f)**

**;=============================**

**MOV A, keyBuf2**

**CJNE A, #0FFh, nextcode0**

**MOV A, keyBuf1**

**MOV R2, #0**

**nextcode1:**

**JNB 0E0h, gotkeycode**

**INC R2**

**RR A**

**CJNE R2, #8, nextcode1**

**SJMP exitgetkeycode**

**nextcode0:**

**MOV A, keyBuf2**

**MOV R2, #8**

**nextcode2:**

**JNB 0E0h, gotkeycode**

**INC R2**

**RR A**

**CJNE R2, #16, nextcode2**

**SJMP exitgetkeycode**

**gotkeycode:**

**MOV keyCode, r2 ; save the final key code**

**exitgetkeycode:**

**POP 0E0h ;pop a**

**RET**

**;-----------------------------------------------------------------**

**;【Subroutine】get the stable key**

**;-----------------------------------------------------------------**

**Delaykeydebounce:**

**PUSH 0e0h ; PUSH A**

**ACALL Delay**

**INC debCounter**

**MOV A, debCounter**

**CJNE A, #debTimes, delayexitt ;debouncing times**

**MOV debCounter, #0 ;reset debouncing counter**

**MOV A, keyCode**

**CJNE A, #0FFh, havekey**

**Delayexitt:**

**LJMP delayexit**

**havekey:**

**CJNE A, keyCheck, unmatchkeyy**

**CJNE A,#0,NOT\_0**

**ACALL ScrollDigit**

**LJMP Turn\_off**

**Unmatchkeyy:**

**LJMP unmatchkey**

**NOT\_0: CJNE A,#1,NOT\_1**

**MOV A,byteDigit**

**ACALL Displaykeypattern**

**ACALL Save\_num**

**MOV Clear,#0**

**LJMP Turn\_off**

**NOT\_1: CJNE A,#2,NOT\_2**

**SETB P1.7**

**ACALL ClearLED**

**ACALL ResetAll**

**LJMP Turn\_off**

**NOT\_2: CJNE A, #3, NOT\_3**

**SETB P1.7**

**CLR bitwaveGen**

**SETB P1.6**

**ACALL SetModes**

**MOV Clear,#0**

**LJMP key\_reset**

**NOT\_3: CJNE A,#4,NOT\_4**

**MOV Save\_whom,#Sec\_num**

**MOV C,P1.7**

**MOV P1,#0FFh**

**MOV P1.7,C**

**MOV byteDigit,#0**

**ACALL ClearLED**

**SJMP key\_reset**

**NOT\_4: CJNE A,#5,NOT\_5**

**ACALL Sum**

**ACALL ShowAnswer**

**ACALL Delete\_zero**

**JB P1.7,Of//while of**

**ACALL InitialSerial**

**ACALL TxDString**

**SJMP Reseting**

**Of: MOV SCON,#0**

**Reseting: MOV C,P1.7**

**MOV P1,#0FFh**

**MOV P1.7,C**

**MOV byteDigit,#0**

**SJMP key\_reset**

**NOT\_5: CJNE A,#6,NOT\_6**

**MOV Clear,#0**

**CPL P1.7**

**JB P1.7,Off//while off**

**ACALL InitialSerial**

**SJMP key\_reset**

**Off: MOV SCON,#0**

**SJMP key\_reset**

**NOT\_6: CJNE A, #7, NOT\_7**

**SETB bitwaveGen**

**CLR P1.6**

**SETB P1.7**

**ACALL setModes**

**MOV Clear,#0**

**SJMP key\_reset**

**NOT\_7: CJNE A, #8, NOT\_8**

**SJMP key\_reset**

**NOT\_8: CJNE A, #9, key\_reset**

**SJMP key\_reset**

**Turn\_off: CLR TR1**

**CLR TR0**

**CLR IT1**

**ORL P1,#01100001B**

**Key\_reset:**

**MOV keyCode, #0ffh ; reset for next key check**

**MOV keyCheck, #0ffh ; reset for next key check**

**SJMP delayexit //**

**unmatchkey:**

**MOV keyCheck, keyCode //**

**delayexit:**

**POP 0e0h //**

**RET //**

**;-------------------------------------------**

**;【Subroutine】InitialSerial**

**;-------------------------------------------**

**InitialSerial:**

**MOV IE,#10010000B**

**MOV TMOD ,#00100000B**

**MOV SCON ,#01010000B**

**MOV TH1,#-6**

**MOV A,PCON**

**SETB ACC.7**

**MOV PCON,A**

**SETB P3.0**

**SETB TR1**

**RET**

**;-------------------------------------------**

**;【Subroutine】 TxDString**

**;-------------------------------------------**

**TxDString:**

**PUSH 0E0h**

**MOV R2,#4**

**MOV R1,#dispBuf**

**NotOver: MOV SBUF,@R1**

**Zero: JNB TI,Zero**

**INC R1**

**DJNZ R2,NotOver**

**POP 0E0h**

**RET**

**;-------------------------------------------**

**;【Subroutine】 Serial\_ISR**

**;-------------------------------------------**

**Serial\_ISR:**

**PUSH 0E0h**

**JB TI,Hereee**

**MOV A,SBUF**

**ACALL Displaykeypattern**

**CLR RI**

**SJMP ExitSerial**

**Hereee: CLR TI**

**ExitSerial:**

**POP 0E0h**

**RETI**

**;-------------------------------------------**

**;【Subroutine】ClearLED**

**;-------------------------------------------**

**ClearLED: PUSH 0E0h //**

**MOV Clear, #0FFh //**

**MOV Port4,#11110111B //**

**MOV dispBuf+3, #0 //**

**PUSH 0 //**

**MOV R0,#43h //**

**ACALL Covled //**

**POP 0 //**

**MOV A,#10 //**

**ACALL Setdigits //**

**POP 0E0h //**

**RET //**

**;-------------------------------------------**

**;【Subroutine】ResetAll**

**;-------------------------------------------**

**ResetAll: PUSH 0E0h //**

**MOV byteDigit,#0 //**

**MOV First\_num,#0 //**

**MOV First\_num+1,#0 //**

**MOV Sec\_num,#0 //**

**MOV Sec\_num+1,#0 //**

**MOV Answer,#0 //**

**MOV Answer+1,#0 //**

**MOV Save\_whom,#First\_num//**

**POP 0E0h**

**RET //**

**;-------------------------------------------**

**;【Subroutine】Save\_num**

**;-------------------------------------------**

**Save\_num:**

**PUSH 0 //**

**PUSH 1 //**

**PUSH 0E0h //**

**MOV R0,Save\_whom //**

**MOV A,Save\_whom //**

**INC A //**

**MOV R1,A //**

**MOV A,@R0 //**

**SWAP A //**

**ANL A,#0F0h //**

**MOV @R0,A //**

**MOV A,@R1 //**

**SWAP A //**

**ANL A,#0Fh //**

**ORL A,@R0 //**

**MOV @R0,A //**

**MOV A,@R1 //**

**SWAP A //**

**ANL A,#0F0h //**

**ORL A,bytedigit //**

**MOV @R1,A //**

**POP 0E0h //**

**POP 1 //**

**POP 0 //**

**RET //**

**;-------------------------------------------**

**;【Subroutine】Sum**

**;-------------------------------------------**

**Sum: PUSH 0 //**

**PUSH 1 //**

**PUSH 0E0h //**

**MOV R0,#First\_num+1 //**

**MOV R1,#Sec\_num+1 //**

**MOV A,@R0 //**

**ADD A,@R1 //**

**DA A //**

**MOV Answer+1,A //**

**DEC R0 //**

**DEC R1 //**

**MOV A,@R0 //**

**ADDC A,@R1 //**

**DA A //**

**MOV Answer,A //**

**POP 0E0h //因先前有 PUSH A,所以要 POP A**

**POP 1 //因先前有 R1,所以要 POP R1**

**POP 0 //因先前有 R0,所以要 POP R0**

**RET //返回**

**;-------------------------------------------**

**;【Define】 Delete\_zero**

**//將高位沒用的 0 刪除**

**//Ex0012 刪除最左邊的兩個 0**

**;-------------------------------------------**

**Delete\_zero:**

**PUSH 0E0h //進入副程式前先 PUSH A**

**PUSH 1 //因為會用到 R1,因此先 PUSH R1**

**MOV R1,#dispBuf //將 R1 給值為 6Ah,結果的最高位**

**Againnn: CJNE @R1,#0,Exit\_delete //若 R1!=0,跳至 Exit\_delete**

**MOV @R1,#0FFh //若 R1=0,將@R1 給值為 0FFh(無字型)**

**INC R1 //R1++,換下一位**

**CJNE R1,#dispBuf+3,Againnn //若 R1!=6Dh 表尚未處理完,**

**//跳至 Againnn 繼續**

**Exit\_delete:**

**POP 1 //因先前有 PUSH R1,所以要 POP R1**

**POP 0E0h //因先前有 PUSH A,所以要 POP A**

**RET //返回**

**;-------------------------------------------**

**;【Subroutine】ShowAnswer**

**;-------------------------------------------**

**ShowAnswer:**

**PUSH 0E0h //**

**PUSH 1 //**

**MOV R4,#0 //**

**MOV R1,#dispBuf+1 //**

**Not\_over: MOV A,Answer //**

**MOV B,Answer //**

**ANL B,#0Fh //**

**MOV @R1,B //**

**SWAP A //**

**ANL A,#0Fh //**

**DEC R1 //**

**MOV @R1,A //**

**INC R4 //**

**CJNE R4,#1, OK //**

**MOV Answer,Answer+1 //**

**MOV R1,#dispBuf+3 //**

**SJMP Not\_over //**

**OK: POP 1 //**

**POP 0E0h //**

**RET //**

**;-------------------------------------------**

**;【Subroutine】ScrollDigit**

**;-------------------------------------------**

**ScrollDigit:**

**PUSH 0E0h //**

**INC byteDigit //**

**MOV A,byteDigit //**

**CJNE A,#10,Go //**

**MOV byteDigit,#0 //**

**Go: POP 0E0h //**

**RET //**

**;-------------------------------------------**

**;【Subroutine】DisplayDigit**

**;-------------------------------------------**

**DisplayDigit:**

**PUSH 0E0h //**

**MOV A,byteDigit //**

**MOV R2,#5 //**

**MOV byteTemp,#0 //**

**R2\_NOT\_0: RRC A //**

**PUSH 0E0h //**

**MOV A,byteTemp //**

**RLC A //**

**MOV byteTemp,A //**

**POP 0E0h //**

**DJNZ R2,R2\_NOT\_0 //**

**MOV A,byteTemp //**

**CPL A //**

**ANL A,#00011110B //**

**ANL P1,#11100001B //**

**ORL P1,A //**

**POP 0E0h //**

**RET //**

**;-----------------------------------------------------------------**

**;【Subroutine】Covdecimal**

**//將數值轉換成十進制(藉由DIV AB)**

**//並將高位給byteBCD1存取, 低位給byteBCD1存取**

**;-----------------------------------------------------------------**

**Covdecimal:**

**PUSH 0E0h //進入副程式前先 PUSH A**

**MOV B, #10 //將B給值為10**

**DIV AB //進行A/B,商數存在A,餘數存在B**

**MOV byteBCD1, A //將A值給byteBCD1記住**

**MOV byteBCD0, B //將B值給byteBCD0記住**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Subroutine】 SetModes**

**;-----------------------------------------------------------------**

**SetModes:**

**PUSH 0e0h //進入副程式前先 PUSH A**

**JB bitwaveGen, here //**

**MOV IE, #10000010B //**

**MOV TMOD, #00000001B //**

**MOV byteTime, #1 //**

**MOV byteTH0, #0FAh //**

**MOV byteTL0, #0D5h //**

**SETB tr0 //**

**MOV dispbuf, #0Ch //**

**MOV dispBuf+1, #0Dh //**

**MOV dispBuf+2, #0Ch //**

**MOV dispBuf+3, #0Dh //**

**SJMP exitsetmodes //**

**Here: MOV IE, #10000110B //**

**MOV TMOD, #11010001B //**

**SETB IT1 //**

**SETB TR1 //**

**MOV Tl1, #1 //**

**MOV TH1, #1 //**

**MOV byteTime, #40 //**

**MOV byteTH0, #60 //**

**MOV byteTL0, #176 //**

**SETB TR0 //啟動Timer0(產生1s High 1s Low)**

**MOV dispbuf, #0Bh //將dispBuf給值為0Bh(字型表第B個)**

**MOV dispBuf+1, #0Bh //將dispBuf+1給值為0Bh(字型表第B個)**

**MOV dispBuf+2, #0Bh //將dispBuf+2給值為0Bh(字型表第B個)**

**MOV dispBuf+3, #0Bh //將dispBuf+3給值為0Bh(字型表第B個)**

**Exitsetmodes:**

**POP 0e0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Subroutine】Displaykeypattern**

**//將LED座向左傳遞的顯示**

**;-----------------------------------------------------------------**

**Displaykeypattern:**

**PUSH 0E0h //進入副程式前先 PUSH A**

**MOV dispBuf,dispBuf+1 //先將左二LED數值給左一LED**

**MOV dispBuf+1,dispBuf+2 //再將左三LED數值給左二LED**

**MOV dispBuf+2,dispBuf+3//然後左一LED數值給右二LED**

**MOV dispBuf+3, A //將取得之按鍵碼給右一的LED**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Subroutine】Setdigits**

**//將A值給dispBuf至dispBuf+3**

**;-----------------------------------------------------------------**

**Setdigits:**

**PUSH 0E0h //進入副程式前先 PUSH A**

**MOV dispBuf, A //將dispBuf給值為A**

**MOV dispBuf+1, A //將dispBuf+1給值為A**

**MOV dispBuf+2, A //將dispBuf+2給值為A**

**MOV dispBuf+3, A //將dispBuf+3給值為A**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Subroutine】 time delay**

**//用在時間延遲,透過迴圈去浪費時間造成delay效果**

**;-----------------------------------------------------------------**

**Delay:**

**MOV R4, #1 //R4給值1**

**delay0: MOV R5, #2 //R5給值2**

**delay1: MOV R6, #100 //R6給值100**

**delay2: MOV R7, #100 //R7給值100**

**delay3: DJNZ R7, delay3 //若R7!=0,跳至delay3**

**DJNZ R6, delay2 //若R6!=0,跳至delay2**

**DJNZ R5, delay1 //若R5!=0,跳至delay1**

**DJNZ R4, delay0 //若R4!=0,跳至delay0**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Subroutine】 7-seg LED pattern conversion**

**//主要是向Led\_teble取字形並交給P0去做顯示**

**;-----------------------------------------------------------------**

**Covled:**

**PUSH 0E0h // 進入副程式前先PUSH A**

**MOV A, @R0 //取R0內容當地址,到該地址取內容給A**

**MOVC A, @a+dptr //將dptr+A(偏移量)到Led\_table取字型**

**MOV P0, A //取到的字形傳至P0做輸出**

**POP 0E0h //因先前有PUSH A,所以要POP A**

**RET //返回**

**;-----------------------------------------------------------------**

**;【Fixed data】 for table lookup**

**//用在事先存取好使 LED 亮 0~9,0.~9.時該給何值**

**//方便直接取值出去做輸出**

**;-----------------------------------------------------------------**

**Led\_table:**

**DB 0c0h //當偏移值為 0 時,取 0c0h,字型 0,回去給 A**

**DB 0F9h //當偏移值為 1 時,取 0F9h,字型 1,回去給 A**

**DB 0A4h //當偏移值為 2 時,取 0A4h,字型 2,回去給 A**

**DB 0B0h //當偏移值為 3 時,取 0B0h,字型 3,回去給 A**

**DB 99h //當偏移值為 4 時,取 99h,字型 4,回去給 A**

**DB 92h //當偏移值為 5 時,取 92h,字型 5,回去給 A**

**DB 82h //當偏移值為 6 時,取 82h,字型 6,回去給 A**

**DB 0D8h //當偏移值為 7 時,取 0D8h,字型 7,回去給 A**

**DB 80h //當偏移值為 8 時,取 80h,字型 8,回去給 A**

**DB 90h //當偏移值為 9 時,取 90h,字型 9,回去給 A**

**DB 0FFh //當偏移值為 10 時,取 0FFh,字型無,回去給 A**

**DB 0BFh //當偏移值為 11 時,取 0BFh,字型-,回去給 A**

**DB 0F7h //當偏移值為 12 時,取 0F7h,字型\_,回去給 A**

**DB 0C8h //當偏移值為 13 時,取 0C8h,字型ㄇ,回去給 A**

**;-------------------------------------------**

**END //結束程式**