國立臺灣科技大學

電子工程系

嵌入式系統設計實習

交流電的功率控制

|  |  |
| --- | --- |
| 指導老師： | 陳武田老師 |
|  |  |
| 實習組別： | 第十三組 |
|  |  |
| 實習組員： | 四電子三甲B9502004葉俊邑 |
|  | 四電子三甲B9502022楊竣宇 |
|  | 四電子三甲B9502028陳敬翔 |

報告日期：民國九十八年五月二十九日

§～catalog～§

[1. 實習目的： 3](#_Toc226102947)

[2. 實習說明： 3](#_Toc226102948)

[3. 實習原理： 3](#_Toc226102949)

[4. 設計內容： 21](#_Toc226102950)

[5. 設備及材料： 36](#_Toc226102951)

[6. 實做及查錯 36](#_Toc226102952)

[7. 實習結果 37](#_Toc226102953)

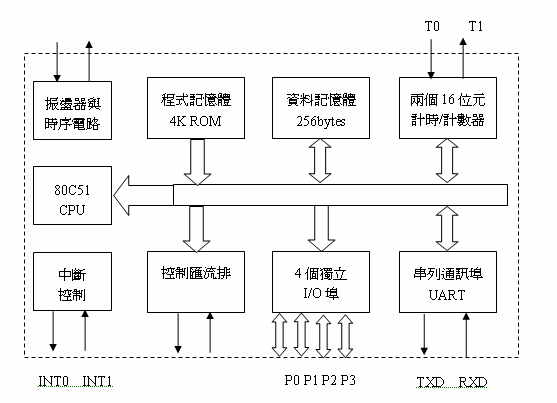
[8. 實習討論 38](#_Toc226102954)

[9. 實習心得 38](#_Toc226102955)

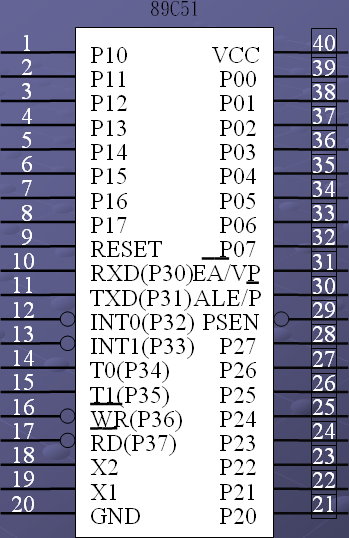
[10. 參考資料 39](#_Toc226102956)

[11. 附件 39](#_Toc226102957)

1. 實習目的：
   1. 了解MOC3041光繼電器及Q4008L4的使用，並用之設計出交流電功率控制器。
2. 實習說明：
   1. 可以設定時間讓使用燈泡10秒鐘半功率，全功率一段時間後，再恢復到半功率10秒鐘，接下來關閉。
   2. 有一個時鐘可以顯示目前的時間及用鍵盤設定功率控制時間。
3. 實習原理：
   1. 8051功能與結構：
      1. 8051的基本功能特性
         1. 8 位元CPU。
         2. 32條雙向可獨立定址的I/O埠。
         3. 4K程式記憶體(ROM)，外部可擴充至64K；
         4. 128byte資料記憶體(RAM)，外部可擴充至64K
         5. 2個16位元計時/計數器, 5個中斷源，
         6. 全雙工的串列通訊埠(UART)
         7. 具有布林運算能力。
      2. 8051內部結構圖：

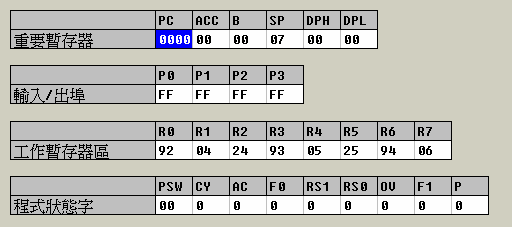


* + 1. 8051接腳電路：



* + 1. 8051內部組成：
       1. 中央處理單元(CPU)。
       2. 內部程式記憶體(ROM)-4KB。
       3. 內部資料記憶體(RAM)-256Bytes。
       4. 振盪與時序電路(12MHZ)。
       5. I/O埠(P0,P1,P2,P3)。
       6. 計時/計數器。
       7. 中斷控制電路。
       8. 串列通訊UART
    2. 一般通用暫存器：
       1. ACC：最重要的暫存器，運算與資料轉移都透過 ACC
       2. PC :程式計數器, 記載著程式下一個待執行指令位址。
       3. B 暫存器：用於乘法，除法指令的輔助暫存器。
       4. PSW 程式狀態字組：記錄程式運作時，CPU各種狀態。
       5. SP堆疊指標：重置(RESET)時，堆疊指標設為07H
       6. DPTR資料指標暫存器16位元暫存器。由DPH，DPL兩個 8位元暫存器組成。
       7. 工作暫存器：共有 RB0、RB1、RB2、RB3四組工作暫存器庫。每個暫存器庫有8個8位元暫存器，分別為R0、R1、R2、R3、R4、R5、R6、R7。

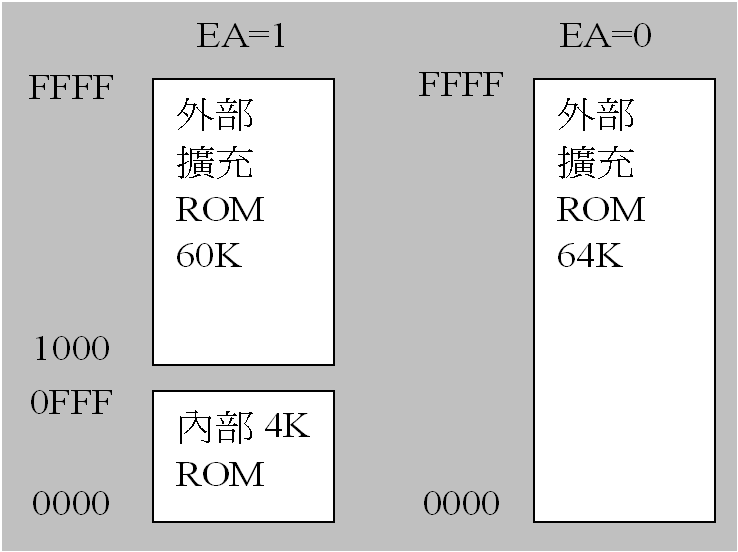
* + 1. 暫存器結構圖：



* + 1. PSW程式狀態字：

|  |  |  |
| --- | --- | --- |
| 位元 | 名稱 | 功能 |
| D7 | CY：進位旗標或借位旗標 | (Carry Flag)進位旗標，用來表示算術指令運算後的結果，其資料的bit 7是否有進位或借位。  加法運算時(ADD)的結果：有進位C=1，沒有進位C=0。  減法運算時(SUB)的結果：有借位C=1，沒有借位C=0。 |
| D6 | AC：半進位旗標或半借位旗標 | (Aux Carry Flag)半進位旗標，用來表示運算後資料的bit 3是否有向bit 4進位或借位。  加法運算時(ADD)的結果：有進位AC=1，沒有進位 AC=0。  減法運算時(SUB)的結果：有借位AC=1，沒有借位 AC=0。 |
| D5 | F0：通用位元 | 可作為一般的讀/寫位元。 |
| D4 | RS1：暫存器庫選擇位元1 | 暫存器庫選擇(Regiser Bank Select)位元1及位元0。  RS1 RS0 暫存器庫選擇  0 0 RB0(位址00h-07h)  0 1 RB1(位址08h-0Fh)  1 0 RB2(位址10h-17h)  1 1 RB3(位址18h-1Fh) |
| D3 | RS0：暫存器庫選擇位元0 |
| D2 | OV：溢位旗標 | (Over)溢位旗標，表示程式經算術或邏輯運算後的結果是否有溢位，若是OV=1，若不是 OV=0。 |
| D1 | - | 空位元 |
| D0 | P：同位元旗標 | (Parity)同位元旗標，表示累積器的內容為奇數個“1”則P=0，偶數個“1”則P=1。 |

* + 1. 8051記憶體介紹：



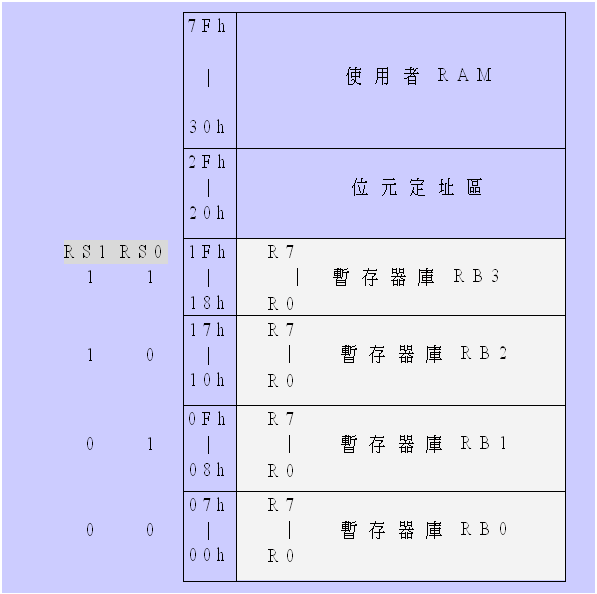
* + 1. 中斷向量位址：

|  |  |  |
| --- | --- | --- |
| 中斷 | 位址 | 功能 |
| RESET | 0000H | 系統重置啟始位址 |
| INT0 | 0003H | 外部中斷INT0向量 |
| INT1 | 0013H | 外部中斷INT1向量 |
| TIMER0 | 000BH | 計時計數中斷TIMER0向量 |
| TIMER1 | 001BH | 計時計數中斷TIMER1向量 |
| TIMER2 | 002BH | 計時計數中斷TIMER2向量 |
| UART | 0023H | 串列埠中斷向量 |

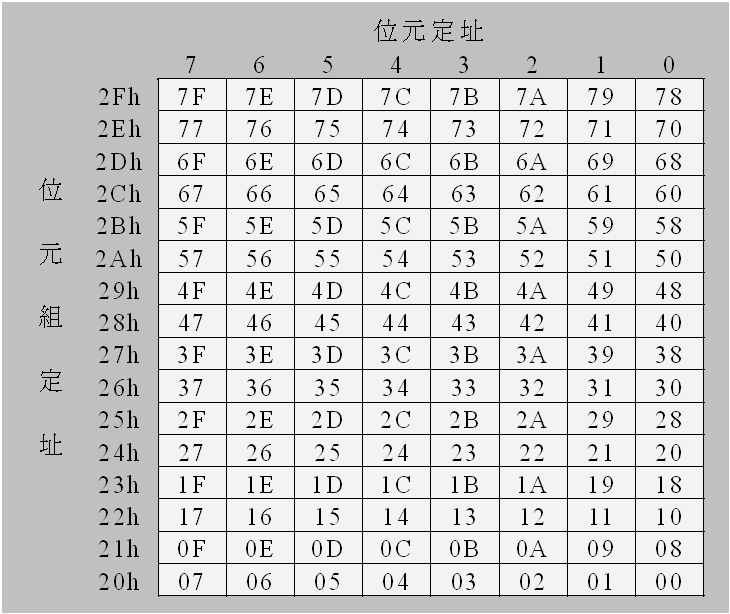
* + 1. 資料記憶體結構：



* + 1. 11、暫存器庫：



* + 1. 12、可位元定址區：



* + 1. 中斷向量位址：

|  |  |  |
| --- | --- | --- |
| 中斷 | 位址 | 功能 |
| RESET | 0000H | 系統重置啟始位址 |
| INT0 | 0003H | 外部中斷INT0向量 |
| INT1 | 0013H | 外部中斷INT1向量 |
| TIMER0 | 000BH | 計時計數中斷TIMER0向量 |
| TIMER1 | 001BH | 計時計數中斷TIMER1向量 |
| TIMER2 | 002BH | 計時計數中斷TIMER2向量 |
| UART | 0023H | 串列埠中斷向量 |

* 1. LCD液晶顯示器：
     1. LCD液晶顯示器功能：
        1. 14支接腳的IC。
        2. LCD顯示器內部具有字元產生器，因此它可以接收ASCII字元碼。
        3. 供許多LCD顯示方式的控制指令，例如清除顯示畫面、游標歸位、顯示On/Off、游標On/Off、閃爍顯示、游標移動等功能。
     2. LCD液晶簡介：
        1. LCD（Liquid Crystal Display）液晶顯示器，顯示方式可分為：文字型LCD與繪圖型LCD兩種。
        2. 常見文字型LCD有16字X2列、20字X2列、40字X2列幾種
        3. 介面以14支信號接腳最為常見。
     3. LCD接腳圖與功能：

|  |  |
| --- | --- |
|  |  |

* + 1. LCD內部結構與功能：
       1. 文字型LCD內部結構包括指令暫存器（IR）、資料暫存器（DR）、顯示記憶體（DD RAM）、與字元產生器ROM（CG ROM）。
       2. DD RAM：對應LCD顯示器上的記憶體。當ASCII資料寫入DD RAM，對應的位址就會顯示該字元。
       3. CG ROM：LCD內部有預存192個5x7點陣字型。這些資料只能讀取，不能更改，所以只要將ASCII寫入DD RAM，對應的點陣字型會顯示在指定的位址上。
       4. 區分為指令暫存器（IR）與資料暫存器（DR），由RS來選擇。
       5. IR主要作用是接受所下達的各項控制指令，諸如清除顯示內容、游標位移、顯示資料RAM（DD RAM）的位址以及字型產生ROM（CG ROM）的位址等等指令。
       6. DR主要作用存取DD RAM與CG ROM中的資料。當欲將資料寫入DD RAM或CG ROM時，並非直接寫入，而是透過DR作為緩衝。LCD執行寫入的程序是先將資料載入DR，然後再自動轉換至DD RAM或CG ROM。
    2. 暫存器之選擇與控制介面信號：

|  |  |  |  |
| --- | --- | --- | --- |
| E | RS | R/W | 作用 |
| 1 | 0 | 0 | 寫入指令暫存器（IR） |
| 1 | 0 | 1 | 讀取忙碌旗標（BF）或位址計數器（AC） |
| 1 | 1 | 0 | 寫入資料暫存器（DR） |
| 1 | 1 | 1 | 讀取資料暫存器（DR） |

* + 1. LCD晶片輸出入：



* + 1. 計算LCD記憶體位址：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 80H | 81 H | 82 H | … | 91 H | 92 H | 93 H |
| C0 H | C1 H | C2 H | … | D1 H | D2 H | D3 H |

* + 1. 電路製作：
       1. 一組8位元的輸出埠（P0）當資料線（DB0～DB7）。
       2. 三條輸出點（P1.0～P1.2）當控制線（E，RW，RS）。
       3. RL接地，或接可變電壓，調整明亮度。
    2. 程式流程：
       1. 設定LCD使用模式為8位元，5X10點矩陣字型，兩行字顯示。
       2. 設定螢幕與游標狀態。
       3. 清除螢幕（CMD=1）。
       4. 設定顯示位址。
       5. 將資料寫入資料暫存器。
  1. 鍵盤掃描控制電路：

4X4按鍵的控制電路，使用8051埠28條I/O線做16個按鍵的鍵盤掃描，由P2.0～P2.3送出掃描信號，而由P2.4～P2.7讀取按鍵資料返回碼。以程式掃描的方式來偵測湔一按鍵按下，一次掃描一行四個按鍵，掃描的順序如下：

* + 1. 送出掃描信號1110共掃描第一行的四個按鍵，讀取按鍵資料，判斷該行是否有鍵按下，若有鍵按下，則連接至被按下的該鍵返回線狀態為0。
    2. 送出掃描信號1101以掃描第二行的四個按鍵，讀取按鍵資料，判斷該行是否有鍵按下。
    3. 送出掃描信號1011以掃描第三行的四個按鍵，讀取按鍵資料，判斷該行是否有鍵按下。
    4. 送出掃描信號0111以掃描第四行的四個按鍵，讀取按鍵資料，判斷該行是否有鍵按下。
    5. 回到步驟Ⅰ繼續做按鍵掃描。
    6. 按鍵編號、掃描信號與讀取按鍵資料返回碼：

以上的步驟連續地重覆，若有按鍵被按下，就將該按鍵解碼出來，至於如何解碼，可以使用雙重迴圈做計數編號，當某一按鍵按下時，其按鍵編號便是計數編號，有關按鍵編號、掃描信號及讀取按鍵資料返回碼列表如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 鍵號 | 按鍵資料輸入碼 | | | | 掃描輸出信號 | | | | 所偵測的按鍵 |
|  | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |  |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | K0鍵 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | K1鍵 |
| 2 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | K2鍵 |
| 3 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | K3鍵 |
| 4 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | K4鍵 |
| 5 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | K5鍵 |
| 6 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | K6鍵 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | K7鍵 |
| 8 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | K8鍵 |
| 9 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | K9鍵 |
| 10 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | K10鍵 |
| 11 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | K11鍵 |
| 12 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | K12鍵 |
| 13 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | K13鍵 |
| 14 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | K14鍵 |
| 15 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | K15鍵 |

* 1. 中斷計時計數與串列通訊：
     1. MCS51的中斷簡介
        1. 單晶片在處理外部輸出入訊號的方式有兩種：
           1. PIO（程式I/O-ProgramI/O）：

透過程式指令讀取I/O埠狀態。但這種輸入方式無法在第一時間立即處理某些具有時效性的外部輸入訊號。

* + - * 1. IIO（中斷I/O-InterruptI/O）：

可立即中斷目前正在執行的程式，並做立即的回應與處理，本章將介紹中斷的原理，型式，與應用。

* + 1. MCS51的中斷型式：
       1. 外部中斷：

8051的外部中斷是指來自晶片硬體接腳INT0、INT1的中斷。這兩個外部中斷的觸發方式有低準位觸發（lowleveltrigger）與負緣觸發（fallingedgetrigger）兩種，可經由TCOM暫存器的ITX（X=0或1）位元設定。

* + - 1. 計時/計數中斷：

當8051的計時/計數器（TIMER0、TIMER1）產生溢位時，溢位旗標TF0（或TF1）會自動設為1，直到CPU跳到對應的中斷向量位址，執行中斷服務程式時，才會自動將TF0（或TF1）清除為1。

* + - 1. 串列埠中斷：

當8051的發射中斷旗標TI或接收中斷旗標RI為1時，會產生串列中斷請求。在中斷服務程式中，必須用指令清除TI與RI，因為硬體不會自動清除這兩個位元。

* + 1. MCS-51的外部中斷：
       1. MCS-51有5個中斷源，2個外部硬體中斷，2個計時/計數中斷，1個串列埠傳輸中斷：
          1. 外部中斷：

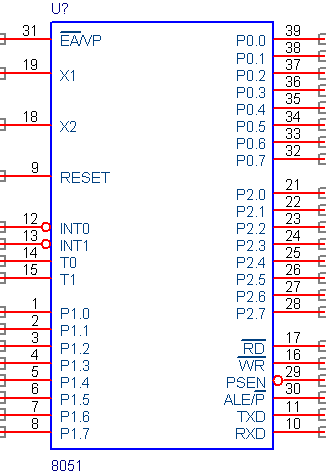
INT0、INT1（P3.2，P3.3）。

* + - * 1. 計時/計數中斷：

T0，T1（P3.4，P3.5）。

* + - * 1. 串列通訊輸出/入：

TXD，RXD（P3.0，P3.1）。



* + 1. MCS-51的中斷功能：
       1. 中斷源的致能方法（使用IE暫存器）。
       2. 外部中斷觸發方式（TCON）。
       3. 中斷優先順序（使用IP暫存器）。
       4. 中斷向量（中斷向量表位址）。
       5. 中斷處理流程。
    2. 中斷致能控制（IE暫存器）：
       1. 中斷源利用中斷致能暫存器（IE暫存器）來控制其致能與禁能。當中斷源被致能時，中斷才能被CPU接受。八位元例如，第0個外部中斷與第1個計時器中斷，IE必須令為#89H=#10001001B，即指令為＜MOVIE，#89H＞，第INT0與第TIMER1中斷。
       2. 當IE小於或等於80H（即IE=#0xxxxxxxB，或IE=#10000000B），所有中斷皆無效（即EA=1，或EX0~EX5皆為0）。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IE.7 | IE.6 | IE.5 | IE.4 | IE.3 | IE.2 | IE.1 | IE.0 |
| EA | 未用 | ET2 | ES | ET1 | EX1 | ET0 | EX0 |

* + - 1. 功能說明：



* + 1. 外部中斷觸發方式（TCON暫存器）：
       1. INT0、INT1外部中斷的觸發方式有兩種：
          1. 低準位觸發（lowleveltrigger）：

第0外部中斷（INT0），使用CLRIT0指令或IT0=0;

第1外部中斷INT1，使用CLRIT1指令或IT0=0;

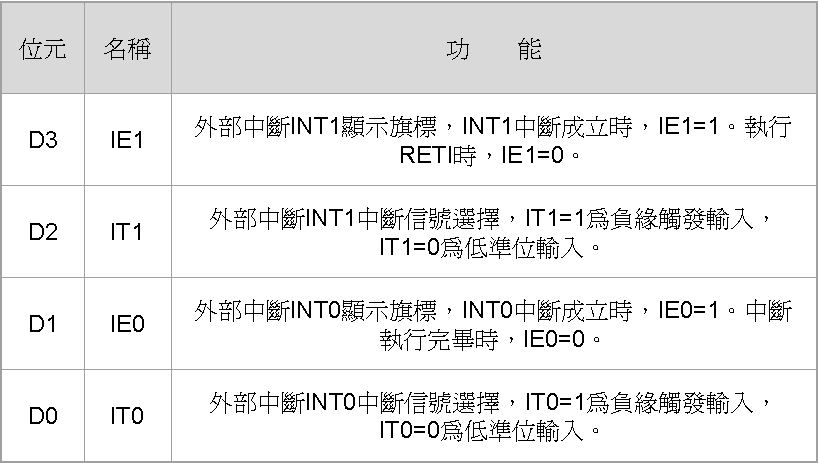
* + - * 1. 負緣觸發（fallingedgetrigger）：

第0外部中斷（INT0），使用SETBIT0指令或IT0=1;

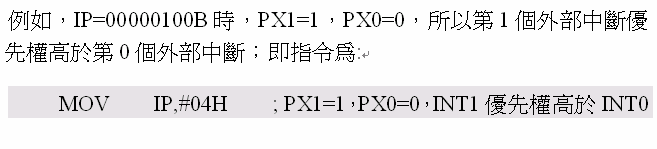
第1外部中斷INT1，使用SETBIT1指令或IT1=1;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TCON.7 | TCON.6 | TCON.5 | TCON.4 | TCON.3 | TCON.2 | TCON.1 | TCON.0 |
| TF | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

* + - 1. 功能說明：



* + 1. 中斷優先順序控制（IP暫存器）：
       1. MCS-51所有中斷源都是使用中斷優先權暫存器（IP暫存器）來控制中斷優先順序。



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IP.7 | IP.6 | IP.5 | IP.4 | IP.3 | IP.2 | IP.1 | IP.0 |
| 未用 | 未用 | PT2 | PS | PT1 | PX1 | PT0 | PX0 |

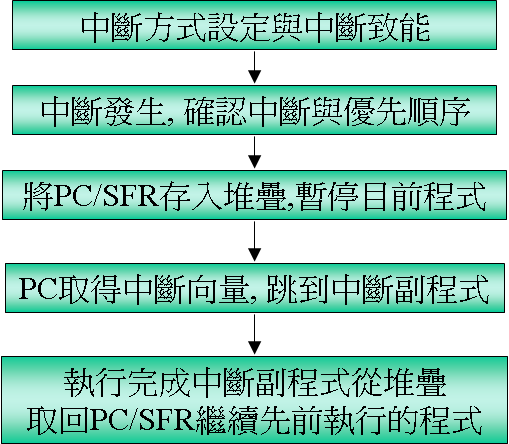
* + - 1. 功能說明：



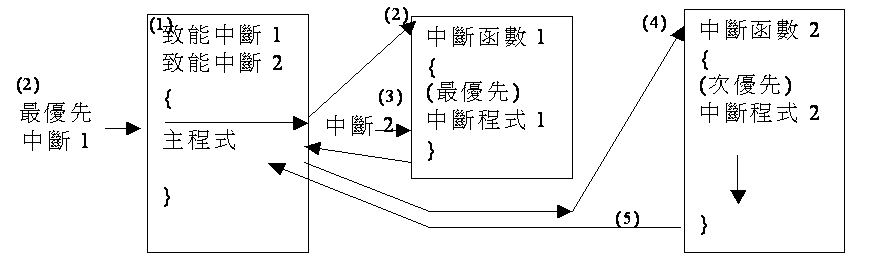
* + 1. 中斷向量位址：

|  |  |  |
| --- | --- | --- |
| 中斷 | 位址 | 功能 |
| RESET | 0000H | 系統重置啟始位址 |
| INT0 | 0003H | 外部中斷INT0向量 |
| INT1 | 0013H | 外部中斷INT1向量 |
| TIMER0 | 000BH | 計時計數中斷TIMER0向量 |
| TIMER1 | 001BH | 計時計數中斷TIMER1向量 |
| TIMER2 | 002BH | 計時計數中斷TIMER2向量 |
| UART | 0023H | 串列埠中斷向量 |

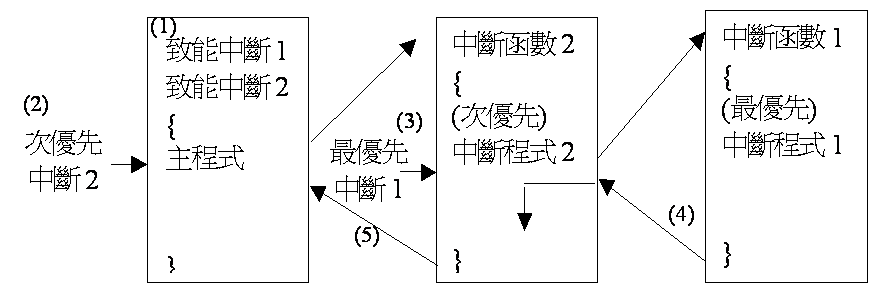
* + 1. 中斷處理流程：



* + 1. 中斷程式的工作方式：
       1. 最優先中斷先輸入：



* + - 1. 次優先中斷先輸入：



1. 設計內容：
   1. 程式碼：

//#include <AT89X51.H>

//#include <REGX51.H>

#include <AT89X51.H>

#define RS P2\_7

#define RW P2\_6

#define Enable P2\_5

#define Out\_sig P2\_4

int mode=0;

//mode = 0 -> clock mode

//mode = 1 -> set clock mode

//mode = 2 -> calculator

int KeyTemp=0xff;

int key=0;

int KeyData=0xff;

char col=0;

char row;

char one=0;

char zero=0;

char ScanLine=0x08;

unsigned char state=0;

//above is for KeyScan function used

int timecnt1=0; //count 10 second then go to mode 0

//int timecnt2=0; //count 10 second then go to mode 0

int timenum[6]={0};//stored clock number

int secondcnt = 0;//use to count every second

char flag\_timeout;

//function

void delay(void);

void CheckBusy(void);

void KeyScan(void);

void WriteIns(char instruction);

void InitialLCD(void);

void WriteData(char i);

void Clock(void);

void ShowTime(void);

//timer 0

void T0\_int(void) interrupt 1

{

TH0=(65536-50000)/256;

TL0=(65536-50000)%256;

secondcnt++;

if(secondcnt == 20)

{

if(flag\_timeout==1)

{

timecnt1++;

//timecnt2++;

if(timecnt1==10)

{

timecnt1=0;

flag\_timeout=0;

if(state==1)

state=2;

if(state==3)

state=0;

//timecnt2=0;

//mode=0;

//WriteIns(0x01);

//WriteIns(0x82);

}

}

Clock();

secondcnt=0;

}

}

//clock function

void Clock(void)

{

timenum[0]++;

if(timenum[0]>9)

{

timenum[1]++;

timenum[0]=0;

if(timenum[1]>5)

{

timenum[2]++;

timenum[1]=0;

if(timenum[2]>9)

{

timenum[3]++;

timenum[2]=0;

if(timenum[3]>5)

{

timenum[4]++;

timenum[3]=0;

//hour

if( (timenum[4]>9)&(timenum[5]<2) )

{

timenum[5]++;

timenum[4]=0;

}

if( (timenum[4]>3)&(timenum[5]==2) )

{

timenum[5]=0;

timenum[4]=0;

}

}

}

}

}

}

void ShowTime(void)

{

WriteIns(0xc4);

WriteData(0x30+timenum[5]);

WriteData(0x30+timenum[4]);

WriteData(0x3A);

WriteData(0x30+timenum[3]);

WriteData(0x30+timenum[2]);

WriteData(0x3A);

WriteData(0x30+timenum[1]);

WriteData(0x30+timenum[0]);

}

//void ShowTime2(void)

//{

//}

main()

{

int resultnum[4]={0};

int inputnum[4]={0};

//int result=0;

char inputcnt=0;

//char calcmd=0;

char flag\_mode2=0;

char flag\_mode1=0;

//char flag\_next=0;

char setcursor=5;

//float refloat=0;

//float ftemp1,ftemp2;

unsigned int shinecnt=0;

//int resultmod = 0;

//int tempf1=0,tempf2=0;

int timenum\_temp[6]={0};

int timenum\_start[6]={0};

int timenum\_end[6]={0,5,9,5,3,2};

int start\_flag=0;

int end\_flag =0;

//unsigned long start\_t=0;

//unsigned long end\_t=0;

//unsigned long current\_t=0;

unsigned int fre\_cnt=0;

int setmode=0;

P0=0x00;

P2=0x00;

P3=0x0F;

TMOD=0x11;

IE=0x8a;

TH0=(65536-50000)/256;

TL0=(65536-50000)%256;

TR0=1;

InitialLCD();

WriteIns(0x01);

WriteIns(0x82);

WriteIns(0x0e);

WriteIns(0x0c);

while(1)

{

KeyScan();//detect key

if( KeyData != 0xff )//if any have key

{

if( (KeyData==0x0A)|(KeyData==0x0B)|(KeyData==0x0F) )//if key is 15 then set mode=1

{

mode = 1;

if(KeyData==0x0A)

{

setmode=1;

}

if(KeyData==0x0B)

{

setmode=2;

}

if(KeyData==0x0F)

{

//setmode=3;

}

}

else

{

if(flag\_mode1 == 0)

mode=0;

//setmode=0;

//if(flag\_mode1 != 1)

// mode = 0;

}

//flag\_timeout = 1 ;

timecnt1=0;

//timecnt1=0;

fre\_cnt=0;

}

if( mode == 0 )

{

if(secondcnt<3)

{

ShowTime();

if(start\_flag==1)

{

if(

(timenum\_start[0]==timenum[0])&

(timenum\_start[1]==timenum[1])&

(timenum\_start[2]==timenum[2])&

(timenum\_start[3]==timenum[3])&

(timenum\_start[4]==timenum[4])&

(timenum\_start[5]==timenum[5])

)

{

flag\_timeout = 1;

state=1;

WriteIns(0x80);

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

start\_flag=0;

}

}

if(end\_flag==1)

{

if(

(timenum\_end[0]==timenum[0])&

(timenum\_end[1]==timenum[1])&

(timenum\_end[2]==timenum[2])&

(timenum\_end[3]==timenum[3])&

(timenum\_end[4]==timenum[4])&

(timenum\_end[5]==timenum[5])

)

{

flag\_timeout = 1;

state=3;

WriteIns(0x88);

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

WriteData(' ');

end\_flag=0;

}

}

}

switch(state)

{

case 1:

case 3:

if(fre\_cnt<50)

{

Out\_sig = 0;

}

else

{

Out\_sig = 1;

}

break;

case 2:

Out\_sig = 1;

break;

default:Out\_sig = 0;break;

}

fre\_cnt++;

if(fre\_cnt>=100)

fre\_cnt=0;

flag\_mode1 = 0;

//flag\_mode2 = 0;

}

if( mode == 1 )

{

//flag\_mode2 = 0;

if(flag\_mode1 == 0)

{

flag\_mode1 = 1;

TR0=0;

setcursor=5;

timenum\_temp[0]=0;

timenum\_temp[1]=0;

timenum\_temp[2]=0;

timenum\_temp[3]=0;

timenum\_temp[4]=0;

timenum\_temp[5]=0;

}

else

{

switch(KeyData)//set clock number

{

case 0:

case 1:

case 2:

case 3:

case 4:

case 5:

case 6:

case 7:

case 8:

case 9:

switch(setcursor)

{

case 0:

case 2:

timenum\_temp[setcursor] = KeyData;

break;

case 1:

case 3:

if(KeyData <6)

timenum\_temp[setcursor] = KeyData;

else

timenum\_temp[setcursor] = 5;

break;

case 4:

if( (timenum\_temp[5]==2)&(KeyData>3) )

timenum\_temp[4] = 3;

else

timenum\_temp[4] = KeyData;

break;

case 5:

if(KeyData<3)

{

if( (timenum\_temp[4]>3)&(KeyData==2) )

{

timenum\_temp[5] = 1;

}

else

timenum\_temp[5] = KeyData;

}

else

{

if(timenum\_temp[4]<4)

timenum\_temp[5] = 2;

else

timenum\_temp[5] = 1;

}

break;

default:break;

}

setcursor--;

if(setcursor<0)

setcursor=5;

break;

case 12:break;

case 13:break;

case 14:break;

case 15:

mode = 0;

TR0=1;

flag\_mode1 =0;

switch(setmode)

{

case 1:

timenum\_start[0] = timenum\_temp[0];

timenum\_start[1] = timenum\_temp[1];

timenum\_start[2] = timenum\_temp[2];

timenum\_start[3] = timenum\_temp[3];

timenum\_start[4] = timenum\_temp[4];

timenum\_start[5] = timenum\_temp[5];

WriteIns(0x80);

WriteData('S');

WriteData(0x30+timenum\_start[5]);

WriteData(0x30+timenum\_start[4]);

WriteData(0x30+timenum\_start[3]);

WriteData(0x30+timenum\_start[2]);

WriteData(0x30+timenum\_start[1]);

WriteData(0x30+timenum\_start[0]);

start\_flag=1;

break;

case 2:

end\_flag=1;

timenum\_end[0] = timenum\_temp[0];

timenum\_end[1] = timenum\_temp[1];

timenum\_end[2] = timenum\_temp[2];

timenum\_end[3] = timenum\_temp[3];

timenum\_end[4] = timenum\_temp[4];

timenum\_end[5] = timenum\_temp[5];

WriteIns(0x88);

WriteData('E');

WriteData(0x30+timenum\_end[5]);

WriteData(0x30+timenum\_end[4]);

WriteData(0x30+timenum\_end[3]);

WriteData(0x30+timenum\_end[2]);

WriteData(0x30+timenum\_end[1]);

WriteData(0x30+timenum\_end[0]);

if(timenum\_end[1]==0)

{

if(timenum\_end[0]==0)

{

timenum\_end[1]=5;

}

else

{

timenum\_end[1]=4;

timenum\_end[0]=10-timenum\_end[0];

}

if(timenum\_end[2]!=0)

{

timenum\_end[2]--;

}

else

{

timenum\_end[2]=9;

if(timenum\_end[3]!=0)

{

timenum\_end[3]--;

}

else

{

timenum\_end[3]=5;

if(timenum\_end[4]!=0)

{

timenum\_end[4]--;

}

else

{

timenum\_end[4]=3;

if(timenum\_end[5]!=0)

{

timenum\_end[5]--;

}

else

{

timenum\_end[5]=2;

}

}

}

}

}

else

{timenum\_end[1]=timenum\_end[1]-1;}

break;

default:break;

}

setmode =0 ;

break;

default:break;

}

shinecnt++;

if(shinecnt==500)

shinecnt=0;

if(shinecnt>=250)//shine which will be setted

{

switch(setcursor)

{

case 0:

WriteIns(0xc4+0x07);

break;

case 1:

WriteIns(0xc4+0x06);

break;

case 2:

WriteIns(0xc4+0x04);

break;

case 3:

WriteIns(0xc4+0x03);

break;

case 4:

WriteIns(0xc4+0x01);

break;

case 5:

WriteIns(0xc4);

break;

}

WriteData(' ');

}

else

{

WriteIns(0xc4);

WriteData(0x30+timenum\_temp[5]);

WriteData(0x30+timenum\_temp[4]);

WriteData(0x3A);

WriteData(0x30+timenum\_temp[3]);

WriteData(0x30+timenum\_temp[2]);

WriteData(0x3A);

WriteData(0x30+timenum\_temp[1]);

WriteData(0x30+timenum\_temp[0]);

}

}

}

KeyData = 0xff;

}

}

void InitialLCD(void)

{

WriteIns(0x38);

WriteIns(0x38);

WriteIns(0x38);

WriteIns(0x38);

//WriteIns(0x08); /\* off display \*/

WriteIns(0x01); /\* clear buffer \*/

WriteIns(0x0c); /\* on display \*/

WriteIns(0x06); /\* set input mode \*/

} /\* InitialLCD \*/

void WriteIns(char instruction)

{

RS=0;

RW=0;

Enable=1;

P0=instruction;

Enable=0;

CheckBusy();

} /\* WriteIns \*/

void WriteData(char i)

{

RS=1;

RW=0;

P0=i;

Enable=1;

P0=i;

Enable=0;

CheckBusy();

} /\* WriteData \*/

void CheckBusy(void)

{

char i=0x80;

while(i&0x80)

{

RS=0;

RW=1;

Enable=1;

i=P0;

Enable=0;

delay();

}

} /\* CheckBusy \*/

void delay(void)

{

int i;

for(i=0;i<30;i++)

;

}

void KeyScan(void)

{

int KeyStatus;

P3=~ScanLine;

KeyStatus=~P3;

KeyStatus&=0xf0;

for(row=0;row<4;row++)

{

if(KeyStatus==0x80)

{

one=0;

if(KeyTemp!=key)

{

KeyTemp=key;

one=0;

zero=1;

}

else

{

zero++;

if(zero==5)

KeyData=KeyTemp;

}

}

key+=1;

KeyStatus<<=1;

} /\* row \*/

ScanLine>>=1;

if(ScanLine==0)

ScanLine=0x08;

col++;

if(col==4)

{

col=0;

key=0;

one++;

if(one==5)

{

zero=0;

KeyTemp=0xff;

KeyData=0xff;

}

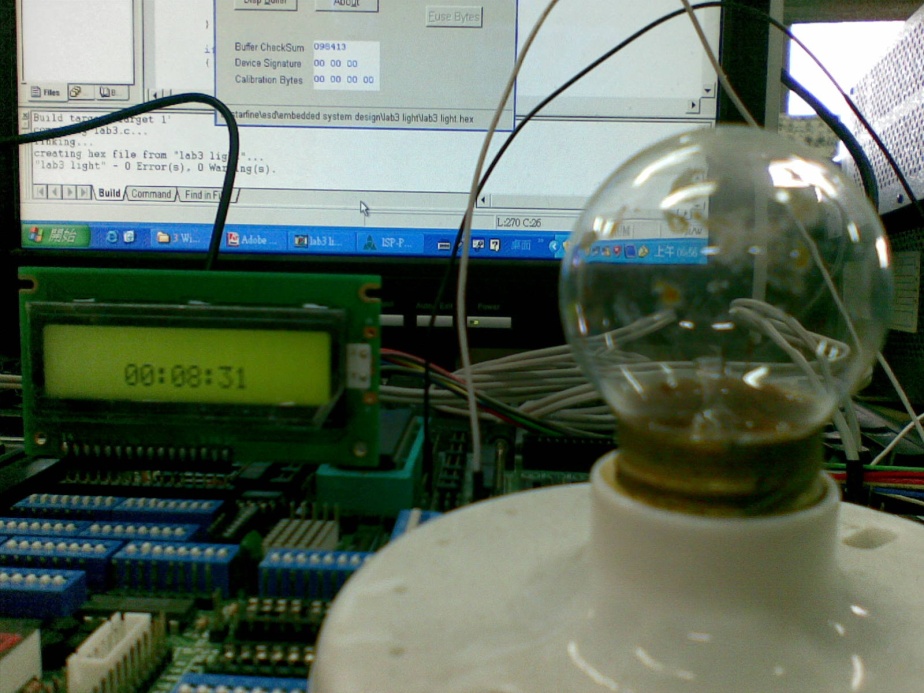
}

} /\* KeyScan \*/

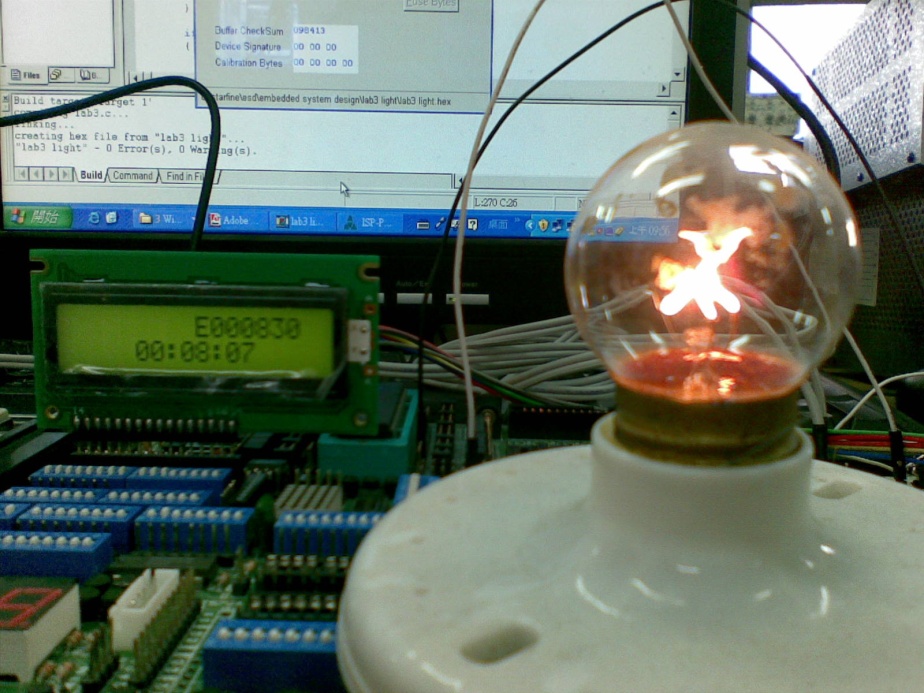
1. 設備及材料：
   1. 德源科技8051及AVR綜合實驗板
2. 實做及查錯

實驗一開始，我們就馬上去借了一組燈泡，而且我們知道這次實驗很危險因為要直接接上110伏特的電壓，所以我們對於麵包板上的一切過程，都特別的小心，儘可能防止一切錯誤和短路的問題發生。

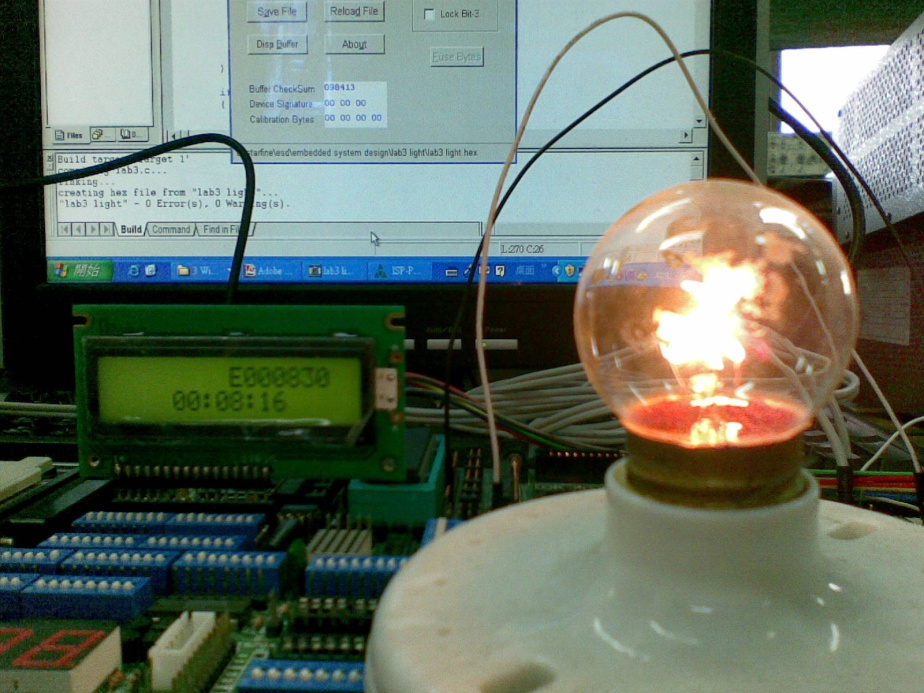
2. 實習結果
   1. 時間未到→0功率



* 1. 開始後十秒與結束前十秒→半功率



* 1. 開始十秒後與結束十秒前→全功率



1. 實習討論

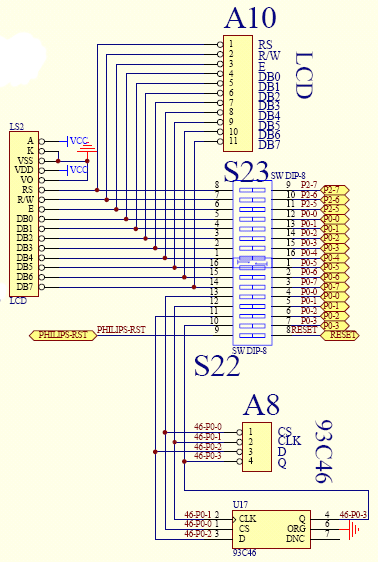
本實習成仍有一些小細節仍需加強與改善，條列如下：

* 1. 實習板上之數字矩陣鍵盤趨於老舊而導致難以操作。
  2. 可能還要多做一些除錯方面的設計，不然結束時間大於開始時間，是不被允許的

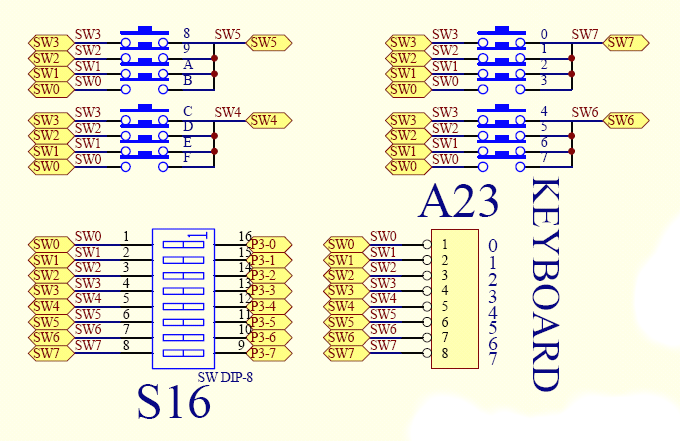
1. 實習心得

外接電壓的實驗果然很危險，這種經驗我們三個人都有過，還記得以前在修類似實驗的時候，常常因為板子有問題，還是接觸不良，燒壞了不少8051，等到查出原因的時候進度以落後別人一大截了，所以這次特別的用心，反覆檢查確定無誤的時候才允許接上電源，實驗很好玩，因為多了個燈泡險的格外的新鮮有趣，加上可以隨意的控制它的亮度，終於有種擺脫程式介面的感覺，以前都是在電腦前寫c語言，在電腦前秀出成果，很少有機會可以與外界的實體聯合在一起，這是個不錯的實驗。

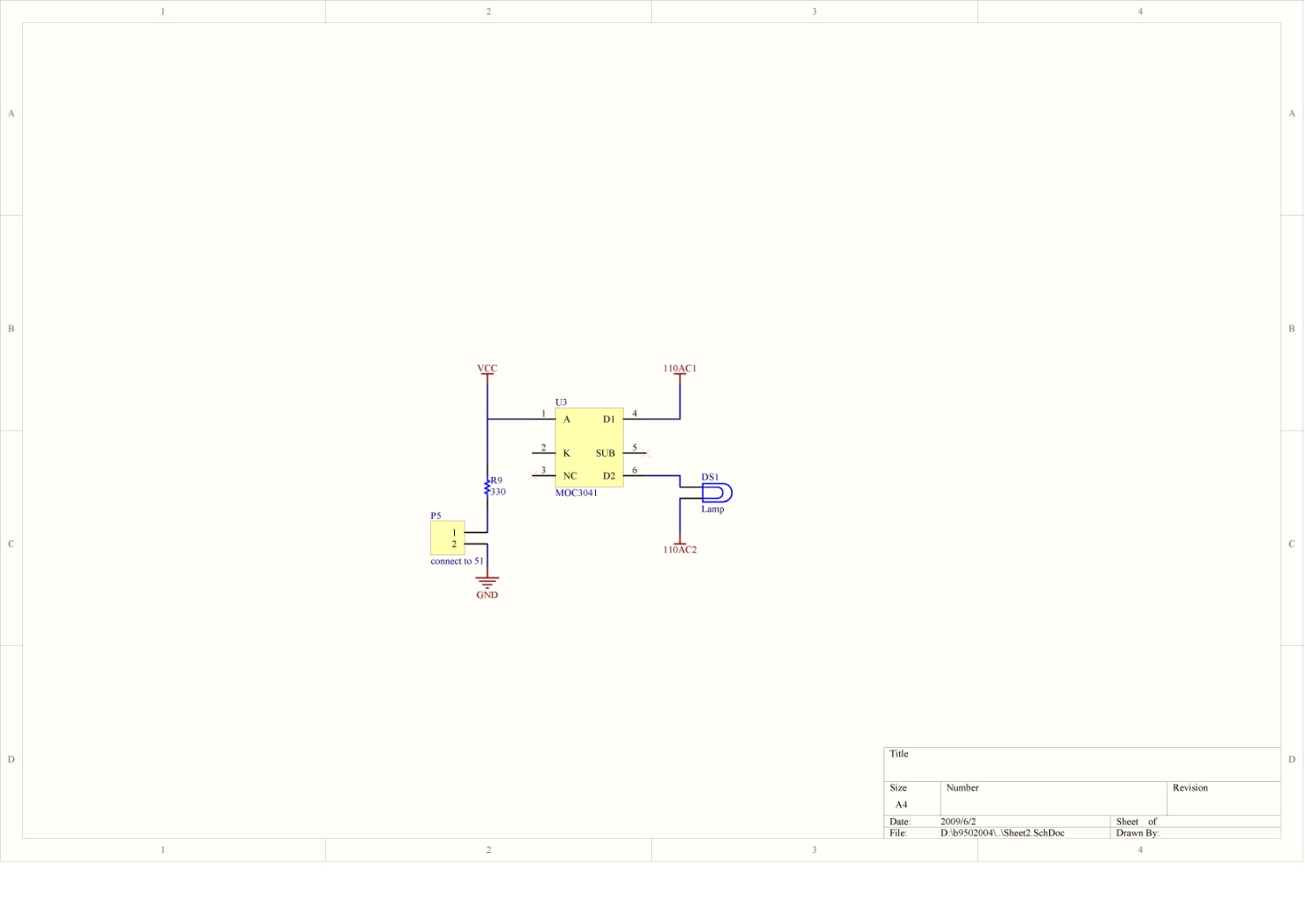
2. 參考資料
   1. 楊明豐，碁峰資訊股份有限公司，「8051單晶片C語言設計實務：使用Keil C」
   2. 實習板功能介紹光碟，各單元電路圖／完整電路圖.pdf
3. 附件
   1. 完整電路圖：
      1. LCD：



* + 1. KEY BOARD：



* + 1. 外接電路電路圖



國立臺灣科技大學

電子工程系

嵌入式系統設計實習

報告結束

感謝老師指導與詳閱