

# LTC\_KDM

## 簡介

堃喬 KDM 實習板提供按鍵組(Keypad)、七節顯示器與步進馬達的實習功能，如圖 1 所示。配合 89S51 線上燒錄實驗板(KT-89S51)，即可快速進行按鍵組與七節顯示器之掃描實習，以及步進馬達之操控實習。另外，也可搭配「例說 89S51-C 語言版」、「例說 8051」與「快學 8051」等書籍(新文京開發出版)，更可徹底學習單晶片之相關能力。

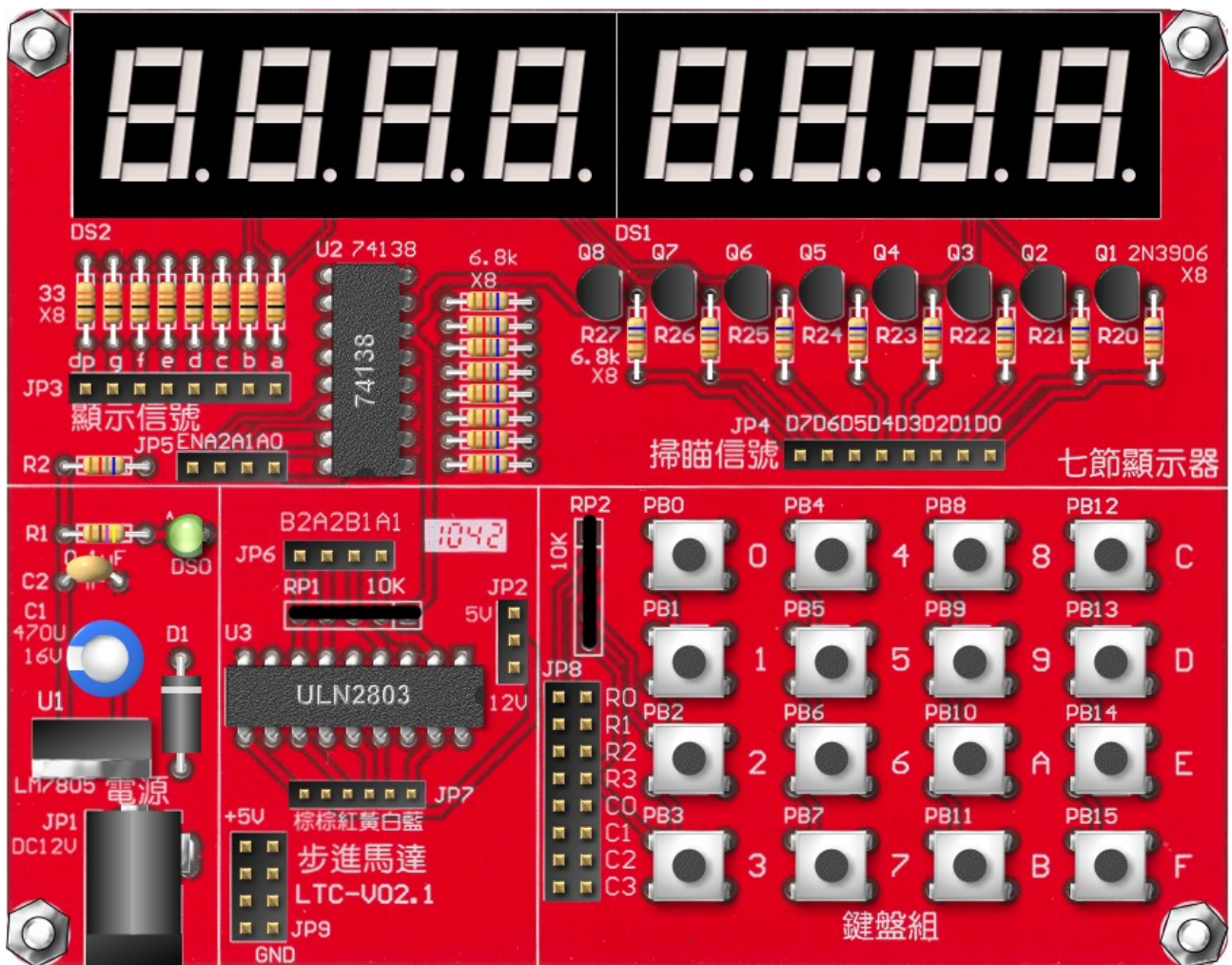


圖1 LTC-KDM 實習板

基本上，LTC-KDM 實習板必須搭配 89S51 線上燒錄實驗板使用，而 89S51 線上燒錄實驗板除提供快速實用的 89S51 燒錄功能外，也可直接進行 LED、蜂鳴器、指撥開關與 LCD 模組(選配)等的實驗。另外，89S51 線上燒錄實驗板也提供所有輸出入埠的擴充功能，可直接連接到 LTC-KDM，即可進行 4×4 按鍵組、8 位數七節顯示

示器與步進馬達之實習。所有實驗與實習，皆可透過 USB，進行線上燒錄/除錯與發展，不但可省去零件插拔的時間與風險，還可縮短時間，即時由電路反應，以調校程式，大大提升學習或開發單晶片電路的效率。

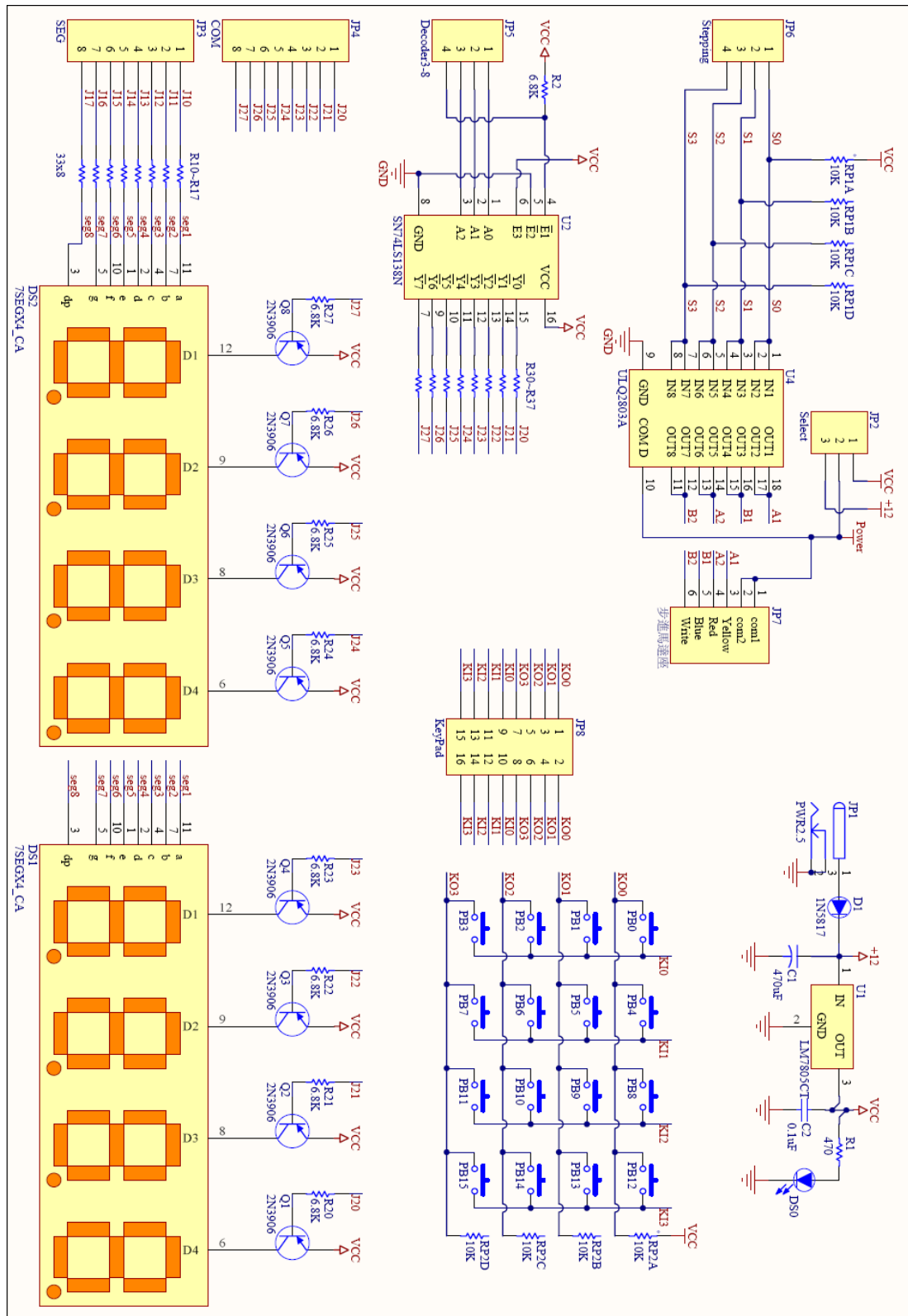


圖2 KDM 實習板之電路

如圖 2 所示為 LTC-KDM 實習板之電路圖，而其中的連接器說明如下：

- **JP1** 為電源連接器，在此配備一個 DC12V 的電源供應器(Adapter)，將此電源供應器插入 JP1(內正外負)，即可提供整塊电路板的電源。另外，若不使用此電源供應器，也可直接將+5V 電源(可由引入 89S51 線上燒錄實驗板)連接入 **JP9**(電路圖上未標示)，亦可取得電源。
- **JP2** 為步進馬達電源選擇器，進行步進馬達實習時，若步進馬達使用 12V 的電源，則在 **JP2** 上，將跳線(Jumper)跳接下方針腳與中間針腳；若步進馬達使用 5V 的電源，則在 **JP2** 上，將跳線跳接上方針腳與中間針腳；若不進行步進馬達實習時，務必把跳線拿開，否則會連續供電，造成 ULN2803 發燙，甚至損害。本實習板提供步進馬達的選配(LTC\_SM12)，此為每步 1.8 度 DC12V 的步進馬達，在此接用 12V 時，即可發揮步進馬達的力道，若使用 5V 時，此步進馬達亦可正常運作，唯轉矩很小。若接用其它步進馬達時，則須使用驅動電流 0.5A 以下之步進馬達。
- **JP3** 為七節顯示器之顯示信號輸入埠，由左而右分別為 dp、g、f、e、d、c、b、a，採低態動作信號。
- **JP4** 為七節顯示器之顯示位數選擇埠，進由左而右分別為 D7~D0，採低態動作信號。
- **JP5** 為 74138 之控制埠，其中 EN 接腳引入低態信號時，74138 才會解碼，而 A2~A0 為 74138 的輸入接腳。此控制埠未接任何信號時，將為高態，而 74138 的輸出也將為高態。74138 的輸出連接到七節顯示器之每個位數選擇線(**JP4**)。
- **JP6** 為步進馬達的控制埠，其接腳順序由左而右為  $\overline{B}$ 、 $\overline{A}$ 、B、A。
- **JP7** 為步進馬達的輸出埠，其接腳順序由左而右為棕、棕、紅、黃、白、藍，與本實習板的選配步進馬達或大部分市售步進馬達的連接線一樣。其中棕色線為共同點(Com.)，紅、黃、白與藍則是依步進馬達的磁極順序排列。
- **JP8** 為 4×4 按鍵組之連接埠，此為 8×2 的排針，左邊排針與右邊排針兩兩相連接，而由上而下依序為 R0~R3、C0~C3，換言之，R0~R3、C0~C3 各

有兩支排針可供使用。

- **JP9** 為 DC5V 之電源連接埠，此為 4×2 的排針，左邊四支排針都為 +5V，而右邊四支排針都為 GND。我們可將外部(89S51 線上燒錄實驗板)電源，透過此連接埠，接入本實習板，以提供所須電源；也可由本連接埠提供外部電路所須之電源。由 **JP1** 引用電源時，則本連接埠亦可提供電源。

堃喬股份有限公司所提供的 LTC-KDM 的實習板，可分為下列產品包裝：

1. **KDM 套件包**：LTC-KDM 電路板、電路板上之所有零件、DC12V 電源供應器、杜邦線(8pin、4pin、2pin 各 3 條)。
2. **KDM 完成品**：已裝配完成之 LTC-KDM 實習板、DC12V 電源供應器、杜邦線(8pin、4pin、2pin 各 3 條)。
3. **LTC\_SM12**：每步 1.8 度 DC12V 步進馬達。

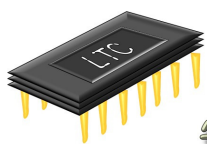
而上述產品之購買資訊如下：

購 買 方 式	
堃喬股份有限公司 <a href="http://www.ltc.com.tw">www.ltc.com.tw</a> 台北縣三重市興德路 111 號 5 樓	
電話：02-29992993 傳真：02-85121770 學校業務專員：李俊宏 0961-011-119	
連絡郵件： <a href="mailto:joe@ltc.com.tw">joe@ltc.com.tw</a> , <a href="mailto:apple@ltc.com.tw">apple@ltc.com.tw</a>	
湯城 3C 電子廣場門市 台北縣三重市重新路五段 609 巷 10 號地下一樓 P5	
電話：02-29955706 傳真：02-29955705	
郵政匯款帳號：	
戶名：堃喬股份有限公司	帳號：19379918

項 次	品 名	規 格	數 量	單 價
1	KDM 套件包	含電源供應器	1	Call
2	KDM 完成品	含電源供應器	1	Call
3	LTC-SM12	步進馬達	1	Call

◆ 團體訂購，另有優惠！(金額未達 NT2000 元，另收台灣地區運費 NT120 元，大陸地區運費 NTCall 元)





# LTC\_KDM 實務應用一

本應用範例是要應用 LTC-KDM 實習板上的 74138，做為 8 位數七節顯示器的位數掃描信號，以顯示「20110125」。而此設計可對應到「例說 89S51-C 語言版」的第 5 章(5-26 頁)。在此分為兩個部分，第一部分為線路連接，第二部分為程式設計與燒錄。

## KDM-1

## 電路連接

yifer

本範例的電路圖，如圖 3 所示，在此將應用 89S51 線上燒錄實驗板與 LTC-KDM 實習板，以實現其功能。

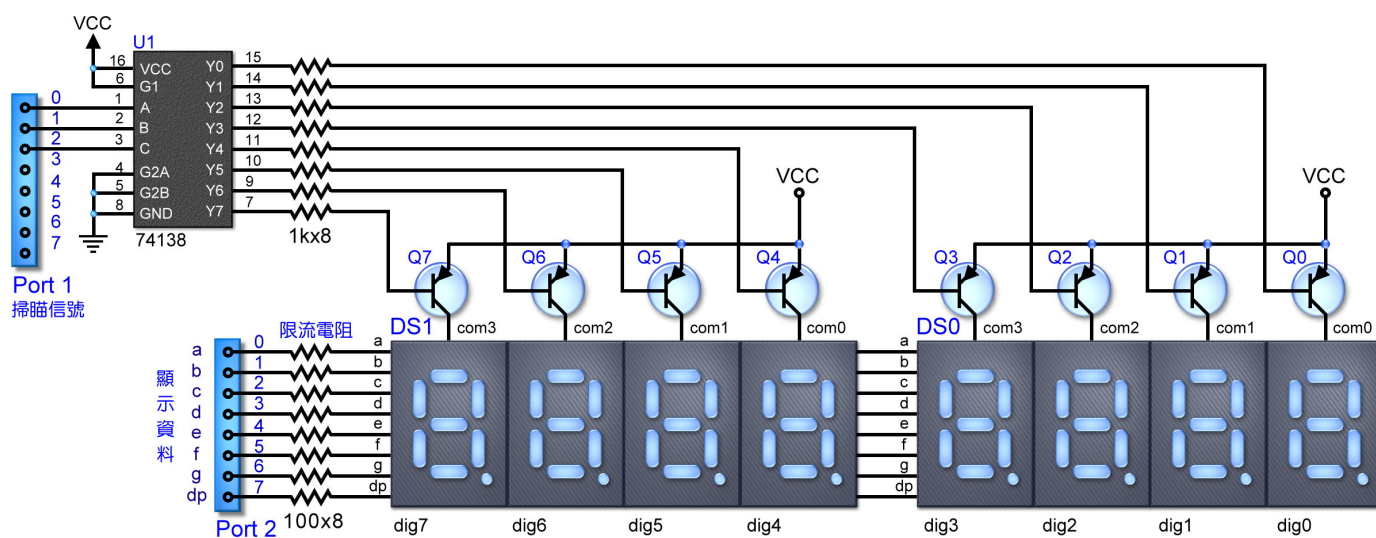


圖3 實習電路

線路連接接步驟如下：

1. 本單元不使用步進馬達，將 **JP2** 的短路環拿掉(除非要使用步進馬達，否則保持開路)。
2. 使用 8pin 的杜邦線，一端連接 89S51 線上燒錄實驗板的 Port 2，另一端連接的 LTC-KDM 實習板，其中 Port 2.0 連接到 **JP3** 的 a(最右邊)，如 4 圖所示。
3. 使用 4pin 的杜邦線，一端連接 89S51 線上燒錄實驗板的 P1.0~P1.3，另一端連接 LTC-KDM 實習板的 **JP5**，其中 P1.0 連接到 **JP5** 的 A0(最右邊)，如 4 圖所示。
4. 使用 2pin 的杜邦線，一端連接 89S51 線上燒錄實驗板的 **CN2**(下方為+5V、上方為 GND)，

另一端連接的 **LTC-KDM 實習板**，其中+5V 連接到 **JP9** 的左排針、GND 連接到 **JP9** 的右排針，如 4 圖所示。

5. 使用 USB 線，一端連接 **89S51 線上燒錄實驗板** 的 **CN1**(USB 埠)，另一端連接電腦上的 USB 埠(安裝驅動程式的连接埠)。

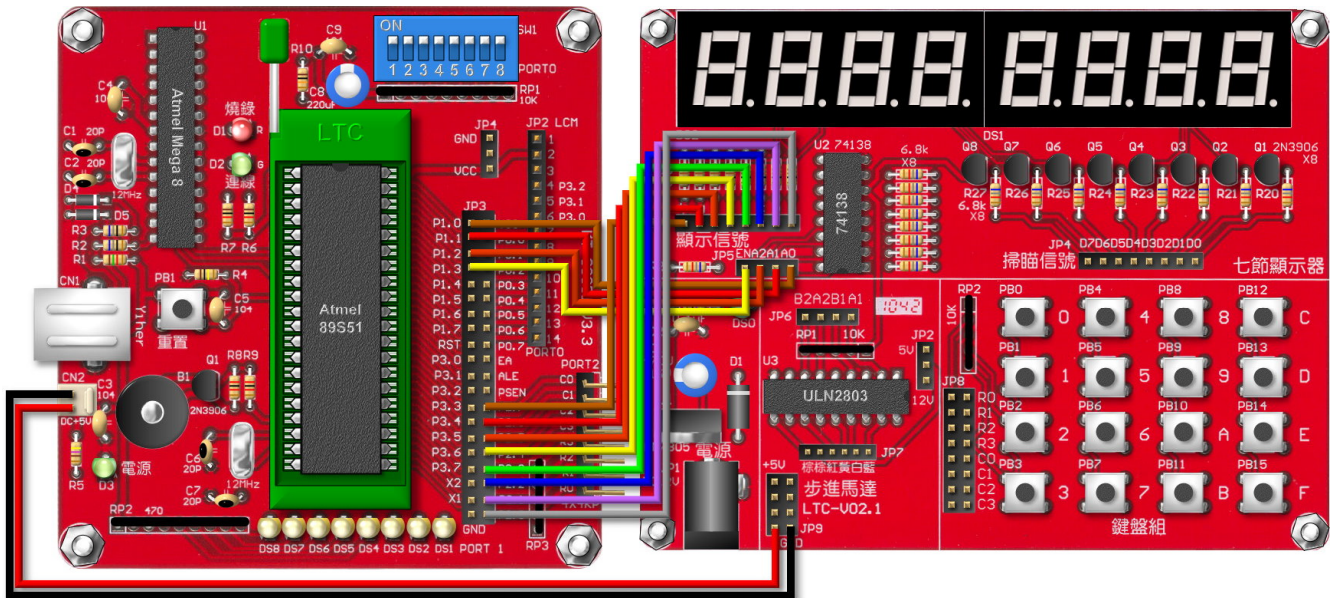


圖4 線路連接圖

## KDM-1

## 程式設計

yifan

本程式如下所示(可在光碟片中找到)，在 Keil C 裡設計完成，並建構產生可執行檔(kdm-1.hex)。而程式的說明與在 Keil C 的操作，詳閱「[例說 89S51-C 語言版](#)」。

```
#include <reg51.h>
/*宣告驅動信號陣列*/
char code TAB[10]={0xc0, 0xf9, 0xa4, 0xb0, 0x99,
                   0x92, 0x83, 0xf8, 0x80, 0x98 };
char disp[8]={ 2,0,1,1,0,1,2,5 }; // 宣告顯示資料
char i,j; // 宣告變數 i,j
void delay1ms(int);
//=====
main()
{
    while(1) // while 迴圈開始
    {
        for(i=0;i<8;i++) // for 敘述開始
        {
            j=disp[7-i]; // 取出顯示數字
            P2=TAB[j]; // 轉換成驅動信號，並輸出到 P2
            P1=i; // 輸出掃描信號
            delay1ms(2); // 延遲 2ms
        } // 結束 for 敘述
    } // 結束 while 敘述
} // 主程式結束
```

```
//=====
void delay1ms(int x)
{ int i,j;
  for(i=0;i<x;i++)
    for(j=0;j<120;j++);
}
//=====
```

產生可執行檔後，將它燒錄到 89S51 線上燒錄實驗板裡的 89S51 晶片。則在 LTC-KDM 實習板上，即顯示「20110125」。

## LTC\_KDM 實務應用二

本應用範例不使用 LTC-KDM 實習板上的 74138，直接驅動 8 位數七節顯示器的位數掃描信號，產生跑馬燈文字，顯示的為字為「27553390」。而此設計可對應到「例說 89S51-C 語言版」的第 5 章(5-40 頁，ch05-4-3.c)，但將原本四位數，擴充為 8 位數。同樣地，在此分為兩個部分，第一部分為線路連接，第二部分為程式設計與燒錄。

### KDM-2

### 電路連接

yiher

本範例的電路圖，如圖 5 所示，在此將應用 89S51 線上燒錄實驗板與 LTC-KDM 實習板，以實現其功能。

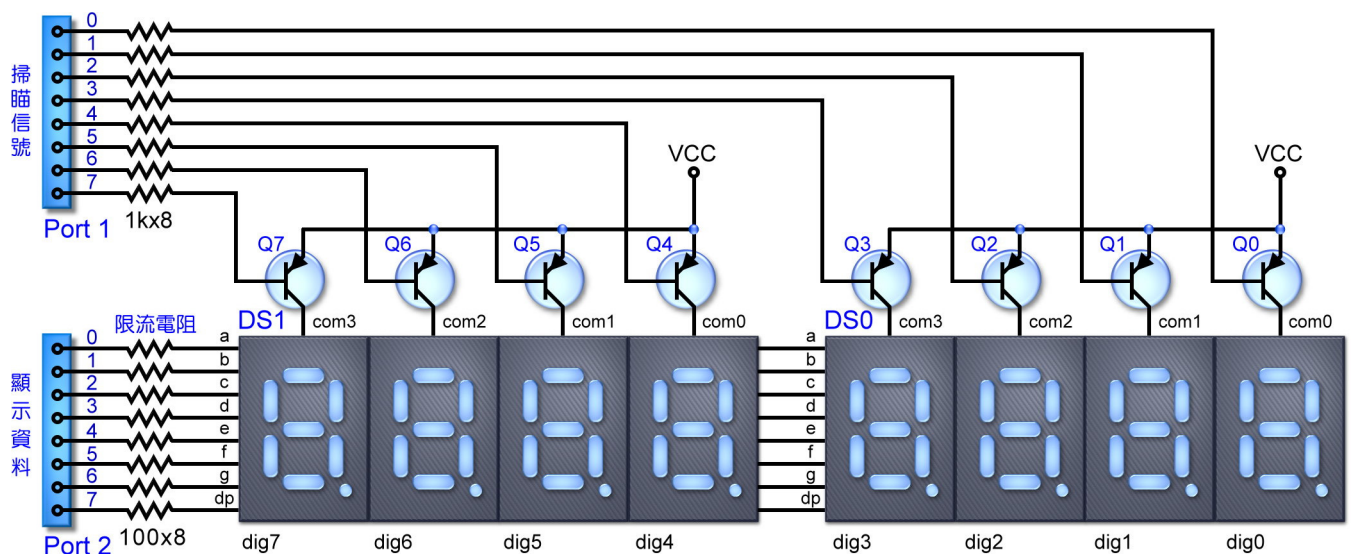


圖 5 實習電路

線路連接接步驟如下：



1. 本單元不使用步進馬達，將 **JP2** 的短路環拿掉(除非要使用步進馬達，否則保持開路)。
2. 使用 8pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 Port 2，另一端連接的 **LTC-KDM 實習板**，其中 Port 2.0 連接到 **JP3** 的 a(最右邊)，如 6 圖所示。
3. 使用 8pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 Port 1，另一端連接 **LTC-KDM 實習板** 的 **JP4**，其中 P1.0 連接到 **JP4** 的 D0(最右邊)，如 6 圖所示。
4. 使用 2pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 **CN2**(下方為+5V、上方為 GND)，另一端連接的 **LTC-KDM 實習板**，其中+5V 連接到 **JP9** 的左排針、GND 連接到 **JP9** 的右排針，如 4 圖所示。
5. 使用 USB 線，一端連接 **89S51 線上燒錄實驗板** 的 **CN1**(USB 埠)，另一端連接電腦上的 USB 埠(安裝驅動程式的连接埠)。

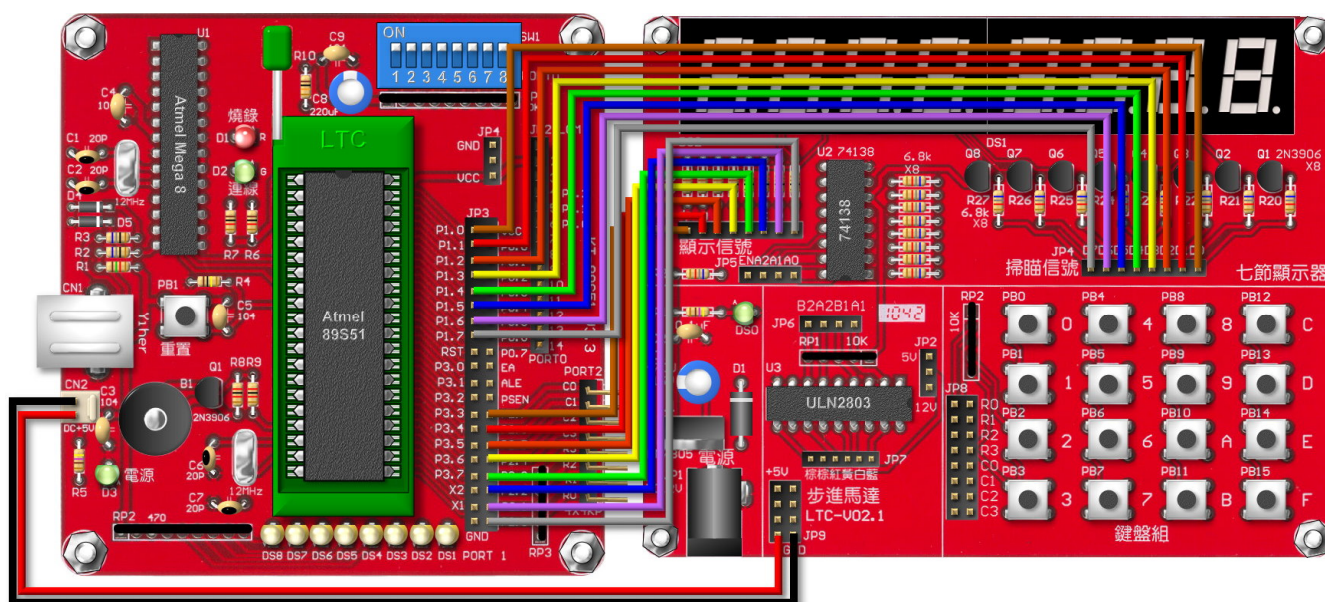


圖6 線路連接圖

## KDM-2

## 程式設計

yifer

本程式如下所示(可在光碟片中找到)，在 Keil C 裡設計完成，並建構產生可執行檔(kdm-2.hex)。而程式的說明與在 Keil C 的操作，詳閱「[例說 89S51-C 語言版](#)」。

```
/* ch05-4-3.c - 4 個七節顯示器跑馬燈實驗,P1 為掃描信號 P2 接七節顯示器 */
//==宣告區=====
#include <reg51.h>           // 定義 8051 暫存器之標頭檔,P2-17~19
#define SCANP    P1          // 定義掃描碼由 Port 1 輸出
#define SEG7P    P2          // 定義七節顯示碼由 Port 2 輸出
char code TAB[11]={ 0xc0, 0xf9, 0xa4, 0xb0, 0x99,           // 數字 0-4
                   0x92, 0x83, 0xf8, 0x80, 0x98, 0xff };    // 數字 5-9,空白
```



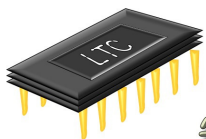
```

#define counts 8           // 宣告字組數量
#define digits 8           // 宣告七節顯示器位數
char disp[2*digits+counts-1]={ 10,10,10,10,10,10,10,10,
                                2,7,5,5,3,3,9,0,
                                10,10,10,10,10,10}; //----27553390---

void delay1ms(int);        // 宣告延遲函數
void scanner(char);        // 掃描函數
//==主程式=====
main()                     // 主程式開始
{   char i;                // 宣告變數 i
    while(1)               // 無窮迴圈,程式一直跑
        for(i=0;i<digits+counts;i++) // 顯示 count 列字組,for 迴圈(字組 i)開始
            scanner(i);    // 掃描第 i 列字組
}                           // 主程式結束
//==副程式=====
/* 延遲函數,延遲約 x*1ms */
void delay1ms(int x)        // 延遲函數開始
{   int i,j;               // 宣告整數變數 i,j
    for (i=0;i<x;i++)      // 計數 x 次,延遲 x*1ms
        for (j=0;j<120;j++); // 計數 120 次,延遲 1ms
}                           // 延遲函數結束
/* 掃描字組函數,顯示第 x 組數字 */
void scanner(char x)        // 掃描字組函數開始
{   char i,j,BCD,scan;     // 宣告變數
    for (i=0;i<30;i++)    // 掃描 30 次 i 迴圈
    {   scan=1;            // 掃描信號初值 0000 0001
        for (j=0;j<digits;j++) // 掃描 4 個數字 j 迴圈
        {   SEG7P=0xff;    // 關閉七段顯示器(防止閃動)
            SCANP=~scan;   // 輸出掃描信號(低 4 位元)
            BCD=disp[x+digits-1-j]; // 讀取第 x 組第 j 個數字之 BCD 碼
            SEG7P=TAB[BCD]; // 輸出至七節顯示器
            delay1ms(16/digits); // 延遲
            scan<=<=1;     // 產生下個掃描信號
        }                 // 結束掃描 4 個數字 j 迴圈
    }                     // 結束掃描 30 次 i 迴圈
}                           // scanner 函數結束

```

產生可執行檔後，將它燒錄到 89S51 線上燒錄實驗板裡的 89S51 晶片。則在 LTC-KDM 實習板上，即以跑馬燈方式顯示「27553390」。



## LTC\_KDM 實務應用三

本應用範例 LTC-KDM 實習板上的 8 位數七節顯示器與 4×4 按鍵組，在 4×4 按鍵組所按的按鍵，隨即在七節顯示器的最右邊顯示，並將原來的數字左移，此設計可對應到「例說 89S51-C 語言版」的第 5 章(5-42 頁，ch05-4-4.c)。同樣地，在此分為兩

個部分，第一部分為線路連接，第二部分為程式設計與燒錄。

## KDM-3

## 電路連接

本範例的電路圖，如圖 7 所示，在此將應用 89S51 線上燒錄實驗板與 LTC-KDM 實習板，以實現其功能。

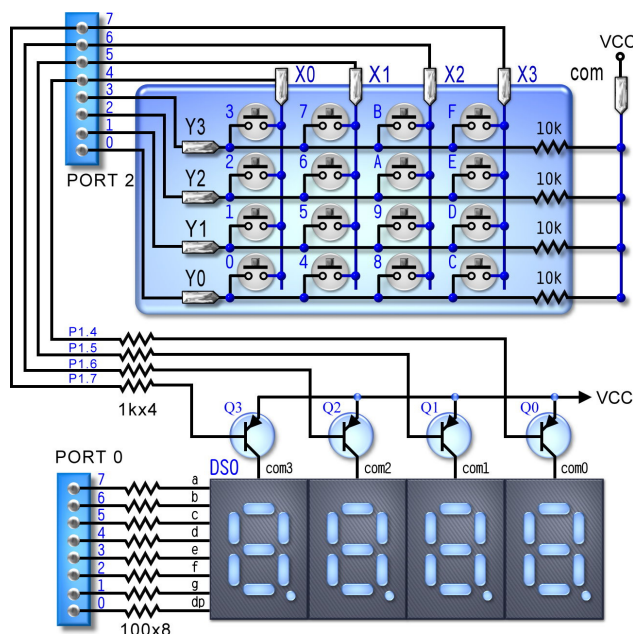


圖7 實習電路

線路連接接步驟如下：

1. 本單元不使用步進馬達，將 **JP2** 的短路環拿掉(除非要使用步進馬達，否則保持開路)。
2. 使用 8pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 Port 2，另一端連接的 **LTC-KDM 實習板**，其中 Port 2.0 連接到 **JP3** 的 a(最右邊)，如 8 圖所示。
3. 使用 4pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 P1.0~P1.3，另一端連接 **LTC-KDM 實習板** 的 **JP8**，其中 P1.0 連接到 **JP8** 的 R0(最上方)、P1.3 連接到 **JP8** 的 R3，如 8 圖所示。
4. 使用 4pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 P1.4~P1.7，另一端連接 **LTC-KDM 實習板** 的 **JP8**，其中 P1.4 連接到 **JP8** 的 C0(最上方)、P1.7 連接到 **JP8** 的 C3，如 8 圖所示。
5. 使用 2pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 **CN2**(下方為+5V、上方為 GND)，另一端連接的 **LTC-KDM 實習板**，其中+5V 連接到 **JP9** 的左排針、GND 連接到 **JP9** 的右排針，如 8 圖所示。

6. 使用 USB 線，一端連接 89S51 線上燒錄實驗板的 CN1(USB 埠)，另一端連接電腦上的 USB 埠(安裝驅動程式的连接埠)。

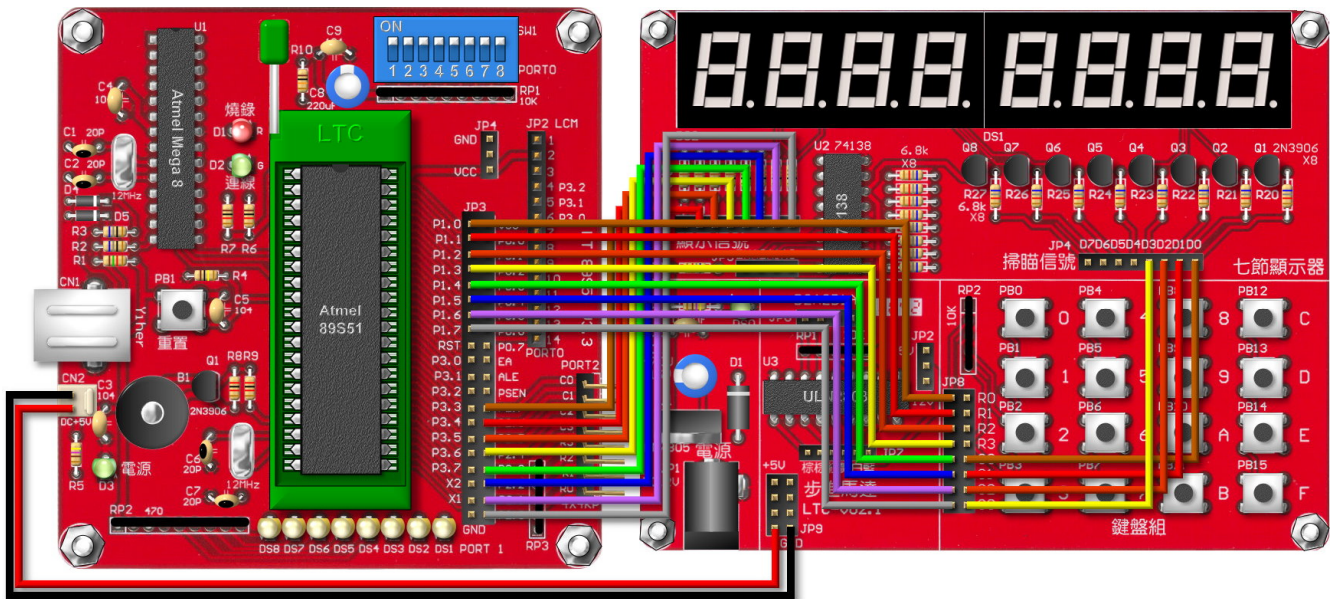


圖8 線路連接圖

## KDM-3

## 程式設計

yiker

本程式如下所示(可在光碟片中找到)，在 Keil C 裡設計完成，並建構產生可執行檔(kdm-3.hex)。而程式的說明與在 Keil C 的操作，詳閱「[例說 89S51-C 語言版](#)」。

```

/* ch05-4-4.c - 4x4 鍵盤與 4 個 7 節顯示器實驗, P1.4~7 為共用掃描信號 */
/* P1.0~3 為鍵盤輸入值, P2 為 7 節顯示器直接輸出 */
//==宣告區=====
#include <reg51.h>    // 定義 8051 暫存器之標頭檔
#define KEYP    P2    // 掃描輸出埠(高位元)及鍵盤輸入埠(低位元)
#define SEG7P    P0    // 7 節顯示器(g~a)輸出埠
unsigned char code TAB[17]=    // 共陽 7 節顯示器(g~a)編碼
{
    0xc0, 0xf9, 0xa4, 0xb0, 0x99,    // 數字 0-4
    0x92, 0x82, 0xf8, 0x80, 0x98,    // 數字 5-9
    0xa0, 0x83, 0xa7, 0xa1, 0x84,    // 字母 a-e(10-14)
    0x8e, 0xbf};    // 字母 F(15), 負號(-)
unsigned char disp[4]={ 0xbf, 0xbf, 0xbf, 0xbf }; // 顯示陣列初值為負號(-)
unsigned char scan[4]={ 0xef, 0xdf, 0xbf, 0x7f }; // 7 顯示器及鍵盤之掃描碼
void delay1ms(int);    // 宣告延遲函數
void scanner(void);    // 宣告掃描函數
//==主程式=====
main()    // 主程式開始
{
    while(1)    // 無窮迴圈, 程式一直跑
        scanner();    // 掃描鍵盤及顯示 7 段顯示器
}    // 主程式結束
// === 延遲函數, 延遲約 x*1ms =====
void delay1ms(int x)    // 防彈跳函數開始

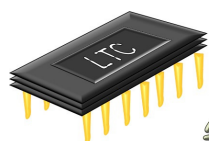
```

```

{   int i,j;                // 宣告整數變數 i
    for(i=0;i<x;i++)        // 計數 x 次,延遲約 *1ms
        for(j=0;j<120;j++); // 計數 120 次,延遲約 1ms
}                                     // 防彈跳函數結束
// ===== 掃描 4*4 鍵盤及 4 個 7 節顯示器函數 =====
void scanner(void)           // 掃描函數開始
{   unsigned char col,row,dig; // 宣告變數(col:行,row:列,dig:顯示位)
    unsigned char rowkey,kcode; // 宣告變數(rowkey:列鍵值,kcode:按鍵碼)
    for(col=0;col<4;col++)    // for 迴圈,掃描第 col 行
    {   KEYP = scan[col];     // 高 4 位輸出掃描信號,低 4 位元輸入列值
        SEG7P = disp[col];    // 輸出第 col 行數字
        rowkey = ~KEYP & 0x0f;
        // 讀入 KEYP 低 4 位,反相再清除高 4 位求出列鍵值
        if(rowkey != 0)      // 若有按鍵
        {   if(rowkey == 0x01) row=0; // 若第 0 列被按下
            else if(rowkey == 0x02) row=1; // 若第 1 列被按下
            else if(rowkey == 0x04) row=2; // 若第 2 列被按下
            else if(rowkey == 0x08) row=3; // 若第 3 列被按下
            kcode = 4 * col + row; // 算出按鍵之號碼
            for(dig = 0; dig < 3 ; dig++) // 顯示陣列之左 3 字
                disp[dig]=disp[dig+1]; // 將右側編碼左移 1 位
            disp[3]=TAB[kcode]; // 鍵值編碼後,寫入最右側
            while(rowkey != 0) // 當按鈕未放開
                rowkey=~KEYP & 0x0f; // 再讀入列鍵值
        } // if 敘述(有按鍵時)結束
        delay1ms(4); // 延遲 4ms
    } // for 迴圈結束(掃描 col 行)
} // 掃描函數 scanner()結束

```

產生可執行檔後，將它燒錄到 89S51 線上燒錄實驗板裡的 89S51 晶片。則在 LTC-KDM 實習板上，將顯示「----」；再按按鍵，則隨即在七節顯示器上顯示其鍵值。



## LTC\_KDM 實務應用四

本應用範例 LTC-KDM 實習板上的步進馬達驅動電路，以驅動 LTC\_SM12 步進馬達，此設計可對應到「例說 89S51-C 語言版」的第 10 章(10-13 頁，ch10-3-1.c)。同樣地，在此分為兩個部分，第一部分為線路連接，第二部分為程式設計與燒錄。

### KDM-4

### 電路連接

yfher

本範例的電路圖，如圖 7 所示，在此將應用 89S51 線上燒錄實驗板與 LTC-KDM 實習板，以實現其功能。



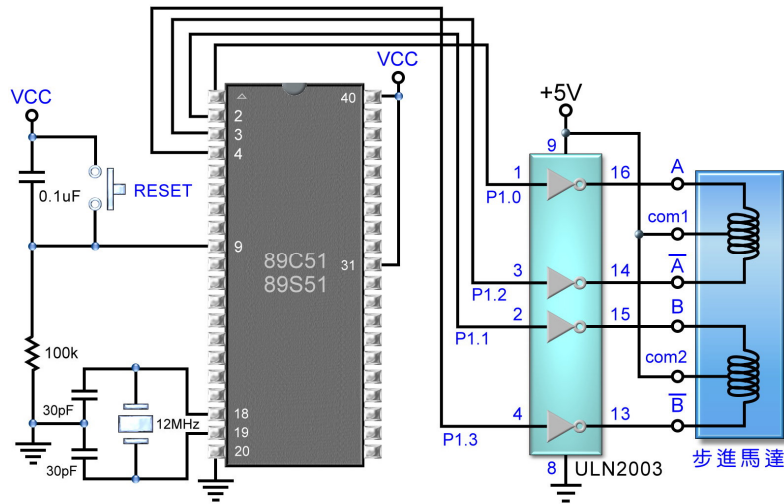


圖9 實習電路

線路連接接步驟如下：

1. 將 DC12V 電源供應器插入 AC110V 市電插座，其輸出(內正外負)插入 **LTC-KDM 實習板** 的 **JP1**。若要使用 12V 驅動步進馬達，則以短路環將 **JP2** 的 12V 端針腳與中間針腳短路。
2. 使用 4pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 P1.0~P1.3，另一端連接的，其中 P1.0 連接 A1(最右邊)，如 10 圖所示。
3. 將 **LTC-SM12** 步進馬達的連接線，按線的顏色插到 **LTC-KDM 實習板** 的 **JP7**，如 10 圖所示。
4. 使用 2pin 的杜邦線，一端連接 **89S51 線上燒錄實驗板** 的 **CN2**(下方為+5V、上方為 GND)，另一端連接的 **LTC-KDM 實習板**，其中+5V 連接到 **JP9** 的左排針、GND 連接到 **JP9** 的右排針，如 10 圖所示。
5. 使用 USB 線，一端連接 **89S51 線上燒錄實驗板** 的 **CN1**(USB 埠)，另一端連接電腦上的 USB 埠(安裝驅動程式的連接埠)。

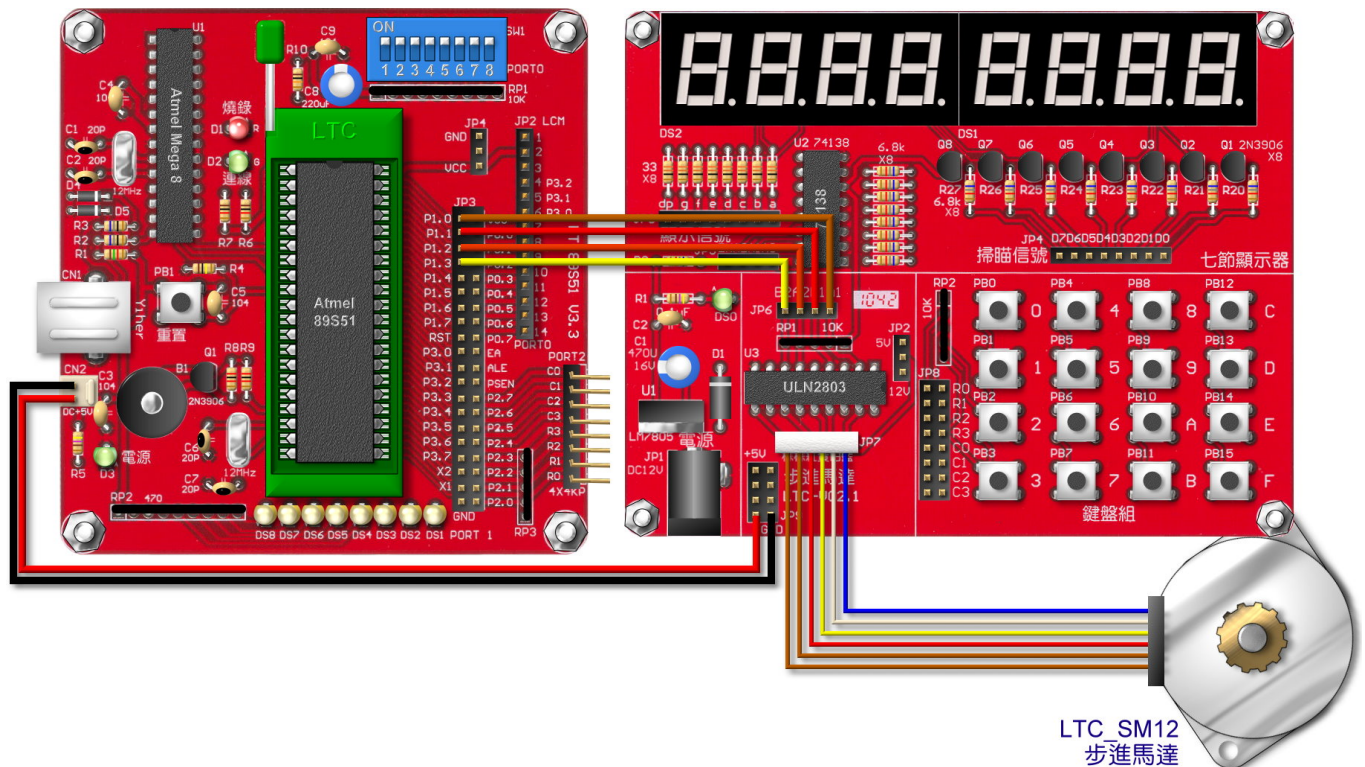


圖 10 線路連接圖

## KDM-4

## 程式設計

yifer

本程式如下所示(可在光碟片中找到)，在 Keil C 裡設計完成，並建構產生可執行檔(kdm-4.hex)。而程式的說明與在 Keil C 的操作，詳閱「[例說 89S51-C 語言版](#)」。

```

/* ch10-3-1.c - 1 相驅動實驗 */
//利用 delay5mDELAY 副程式(5ms×times)，產生驅動信號
//由 P1 輸出 速度為 1/(5ms×timers) 步/秒
#include <reg51.h> // 包含 reg51.h 檔
#define OUTPUT P1 // 定義輸出埠為 P1
unsigned int times=50; // 宣告延遲時間變數(×5ms)
unsigned char excite; // 宣告激磁變數
void step_rst(void); // 宣告定位函數
void delay5ms(int); // 宣告延遲函數
//=====主程式=====
main() // 主程式開始
{ OUTPUT=0; // 關閉輸出
  step_rst(); // 定位
  while (1) // while 迴圈
    step_rst(); // 運轉
} // 結束主程式
//=====定位函數=====
void step_rst(void) // 定位函數開始

```

```
{ char i; // 宣告變數
  excite=1; // 激磁初值
  for(i=0;i<4;i++) // 輸出四個信號
  { OUTPUT=excite; // 輸出激磁信號
    delay5ms(times); // 延遲 0.25 秒
    excite<<=1; // 左移，下一個激磁
  } // 結束
}
//=====延遲函數=====
void delay5ms(int x) // 延遲函數開始
{ int i,j; // 宣告變數
  for(i=0;i<x;i++) // i 迴圈
    for(j=0;j<600;j++); } // j 迴圈
```

產生可執行檔後，將它燒錄到 [89S51 線上燒錄實驗板](#)裡的 89S51 晶片，則 [LTC-SM12](#) 步進馬達將持續旋轉。

*yiher.chang@msa.hinet.net*