



十速

TM52F5284/88

Mini Development Board

使用說明書

Rev 1.0

tenx reserves the right to change or discontinue the manual and online documentation to this product herein to improve reliability, function or design without further notice. **tenx** does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. **tenx** products are not designed, intended, or authorized for use in life support appliances, devices, or systems. If Buyer purchases or uses **tenx** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **tenx** and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that **tenx** was negligent regarding the design or manufacture of the part.

AMENDMENT HISTORY

Version	Date	Description
V1.0	Sep, 2015	New release.

CONTENTS

AMENDMENT HISTORY	2
第一章、實驗板硬體介紹.....	5
1.1 微控制器介紹.....	5
1.2 實驗板模組介紹.....	8
GPIO 模組：.....	10
4x4 掃描按鍵模組:.....	11
4 位數七段顯示器模組：.....	12
8 顆 LED 模組:.....	13
UART 通訊模組：.....	14
紅外線接收模組：.....	15
可調電阻電壓調變模組：.....	15
Buzzer 驅動模組：.....	16
1602 文字型液晶顯示模組：.....	16
Arduino 腳位擴充模組：.....	17
第二章、軟體開發環境介紹.....	18
2.1 開發軟體 Keil C.....	18
2.2 F51DLL_Setup 安裝-加入十速 MCU 套件.....	21
2.3 建立專案.....	24
2.4 開新檔案開始寫 Code.....	25
第三章、進入 8051 微控制器的世界.....	28
3.1 讓 IC 跑起來（使用 MCU 的基本初始設定）.....	29
3.2 點 LED 燈（IO 控制、debug 模式簡介）.....	31
3.3 點亮七段顯示器（IO 控制、timer 應用）.....	35
3.4 LED 呼吸燈（PWM 應用、timer 應用）.....	38
3.5 掃描按鍵輸入（IO 控制、timer 應用）.....	40
3.6 讓 PC 跟實驗板利用 UART（RS232）通訊（UART 應用）.....	43
3.7 可變電阻模組類比數位轉換（ADC 應用）.....	47
3.8 1602 LCM 文字型液晶模組.....	49



附錄..... 57
 GPIO 連接板..... 57

第一章、實驗板硬體介紹

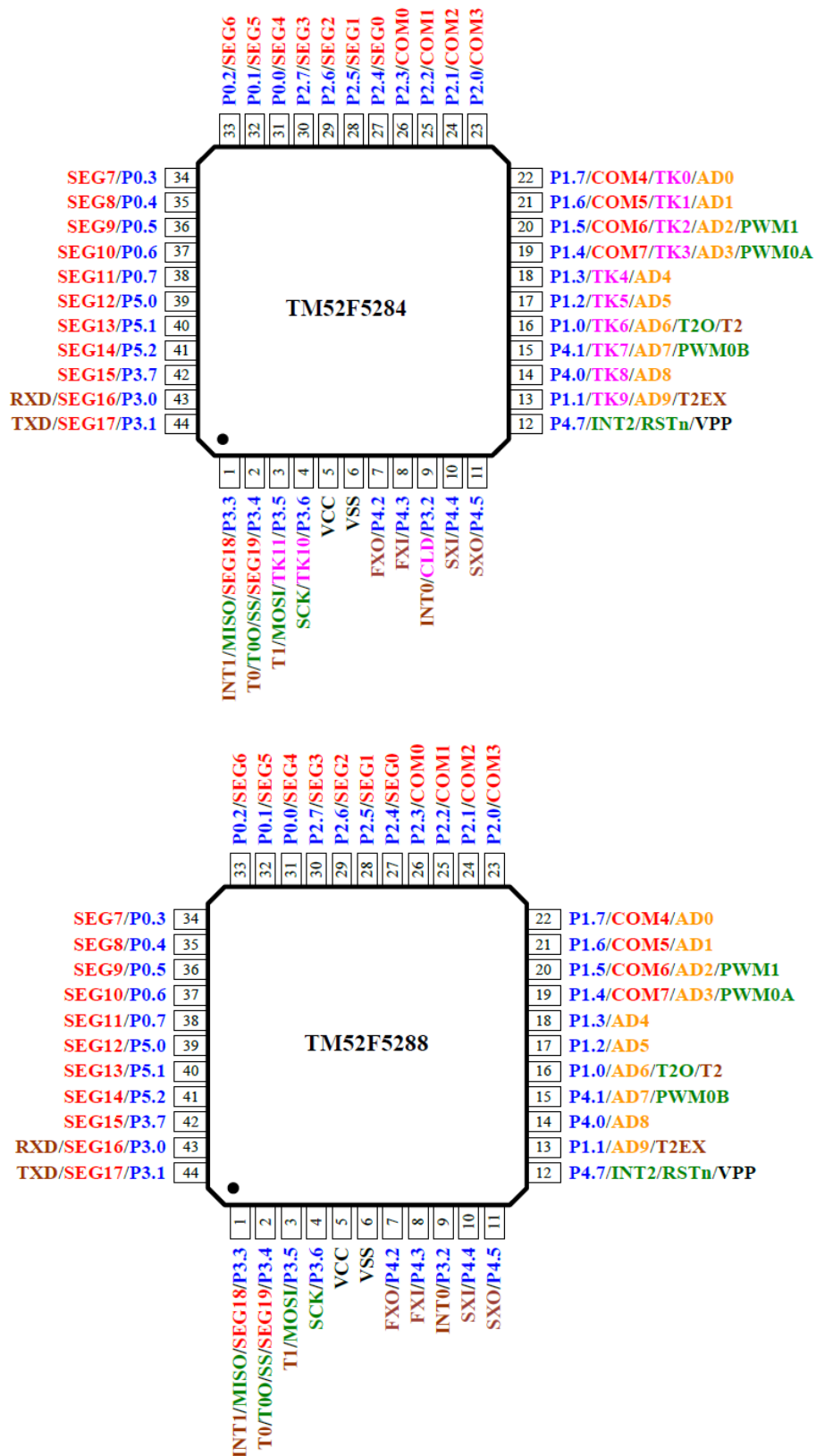
1.1 微控制器介紹

TM52F5284/88 Mini DEV Board 使用十速科技 F51 系列微控制器 TM52F5288/84，主要規格如下（詳細資訊請參考 Datasheet，[於此下載 http://www.tenx.com.tw/product_detail.aspx?ProductID=309](http://www.tenx.com.tw/product_detail.aspx?ProductID=309)）：

- 標準 8051 指令集，快速機器周期
- 16K Bytes Flash 程式記憶體空間
- 512 Bytes SRAM 資料記憶體空間 (IRAM + XRAM)
- 四種運作時脈：
 - Fast clock from 1~6 MHz Crystal
 - Fast clock from Internal RC (7.3728 MHz)
 - Slow clock from 32768 Hz Crystal
 - Slow clock from Internal RC (80 KHz)
 - System clock can be divided by 1/2/4/16 option
- 8051 標準計時器 – Timer0/1/2
- 一個 15-bit Time3
- 一組 8051 標準 UART
- 兩組獨立 "8+2" bits PWMs
- 一組 SPI 介面
- 12-通道 Touch Key(僅 TM52F5284)
- 12-bit ADC 具 10 個外部接腳通道和兩通道內部參考電壓
- LCD Controller/Driver
- LED Controller/Driver
- 11 個中斷源
 - Timer0/Timer1/Timer2/Timer3 Interrupt
 - INT0/INT1 Falling-Edge/Low-Level Interrupt
 - Port1 Pin Change Interrupt
 - UART TX/RX Interrupt
 - P4.7 (INT2) Interrupt
 - ADC/Touch Key Interrupt
 - SPI Interrupt

- 接腳中斷能將停止模式下的 CPU 喚醒
- 最大 42 Programmable I/O pins
- 獨立 RC 震盪看門狗計時器
- 五種 Reset 方式(上電 Reset/可選外部接腳 Reset/看門狗 Reset/軟體 Reset/可選低電壓 Reset)
- 3 級低電壓 Reset(1.9V/2.3V/2.9V)
- 1 級低電壓檢測(2.3V)
- 四種電源運作模式 (Fast/Slow/Idle/Stop Mode)
- On-chip Debug/ICE 介面
 - Use P1.2/P1.3 pin
 - Share with ICP programming pin
- 工作電壓與電流
 - $V_{CC} = 2.9V \sim 5.5V$ @ $F_{SYSCLK} = 7.3728$ MHz
 - $V_{CC} = 1.9V \sim 5.5V$ @ $F_{SYSCLK} = 4$ MHz
 - $I_{CC} = 3.5\mu A$ @Stop mode, LVR enable, MODE3V = 0, PWRSV = 1, $V_{CC} = 5V$
 - $I_{CC} = 1.2\mu A$ @Stop mode, LVR enable, MODE3V = 0, PWRSV = 1, $V_{CC} = 3V$
 - $I_{CC} = 1.2\mu A$ @Stop mode, LVR enable, MODE3V = 1, PWRSV = 1, $V_{CC} = 3V$

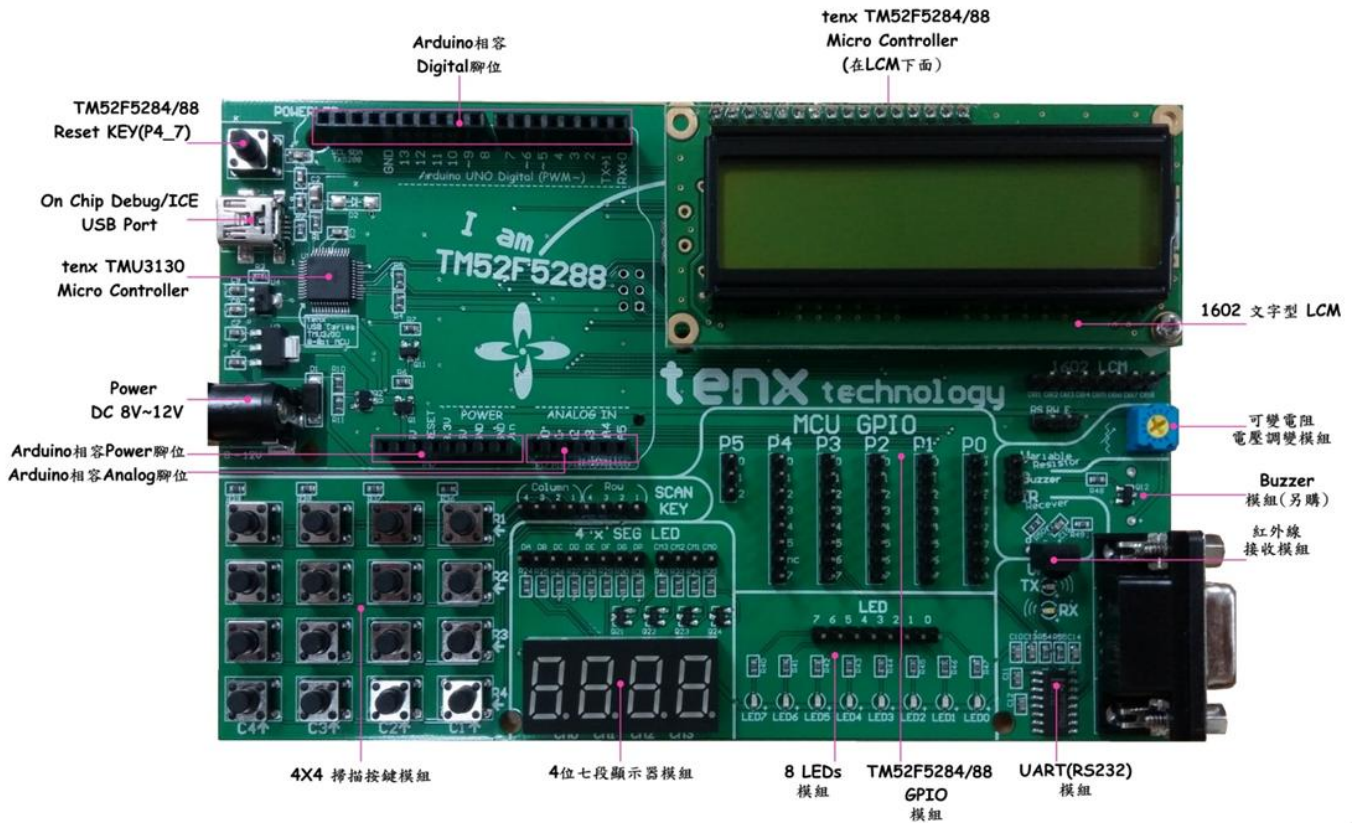
下圖為本實驗板使用之 TM52F5284/88 QFP-44 封裝的腳位圖：



1.2 實驗板模組介紹

本實驗板已將十速 TM52F5288 微控制器及 On Chip Debug/ICE 介面併在板中，只需要插上 USB 線即可使用，在 Keil C 撰寫完程式之後，Keil C 搭配板上的 On Chip Debug / ICE 介面即可使用除錯功能、執行、單步執行、燒錄、監控數值...等功能，另板上亦含有許多常見之周邊電路模組供開發者使用，各模組電路如下：

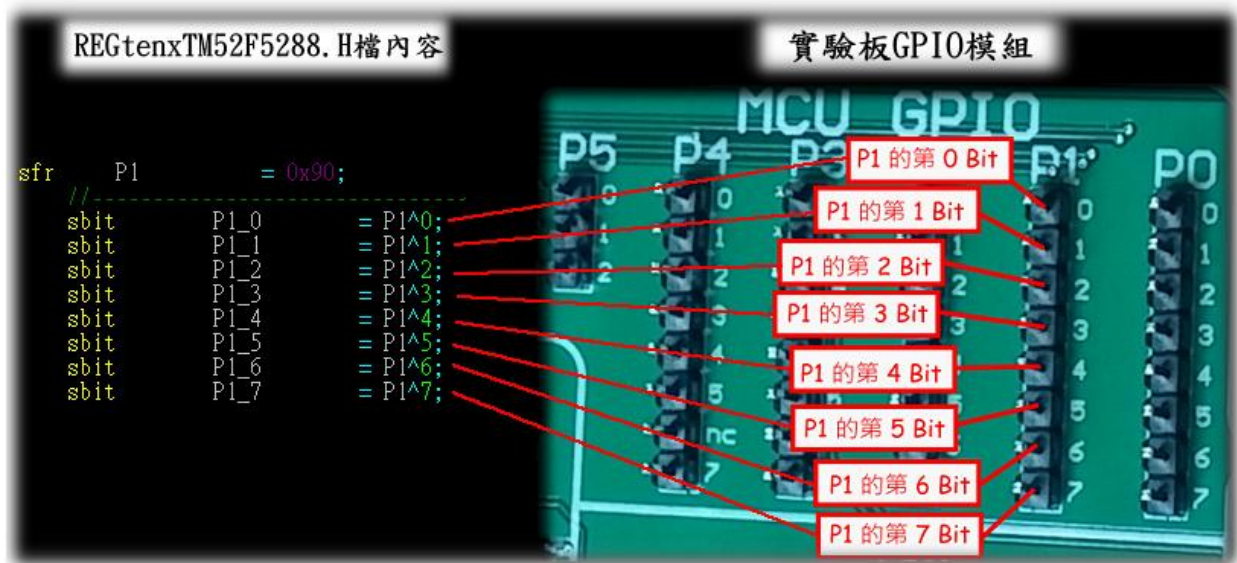
1. [GPIO 模組](#)
2. [4x4 掃描按鍵模組](#)
3. [4 位數七段顯示器模組](#)
4. [8 顆 LED 模組](#)
5. [UART 通訊模組](#)
6. [可調電阻電壓調變模組](#)
7. [Buzzer 驅動模組 \(蜂鳴器另購\)](#)
8. [紅外線接收模組](#)
9. [1602 文字型液晶顯示模組](#)
10. [符合 Arduino 定義腳位之模組，可供外插更多應用子板](#)



GPIO 模組：

GPIO 模組即是將 TM52F5288 上的所有 I/O 拉出按編號排列，方便連接其他實驗模組收發訊號，開發程式時只要 include REGtenxTM52F5288.H 檔，此檔已將各 Port 的腳位定義好名稱，方便程式撰寫。

以 P1 為例，如在程式中讀寫 P1_0 就代表是讀寫實驗板上 P1 的第 0 Pin 的電壓準位訊號，因為板子是以 5V 運作，所以程式中 0 代表輸出或讀到 0V，1 代表輸出或讀到 5V，下圖為實驗板模組與程式定義檔對照圖：

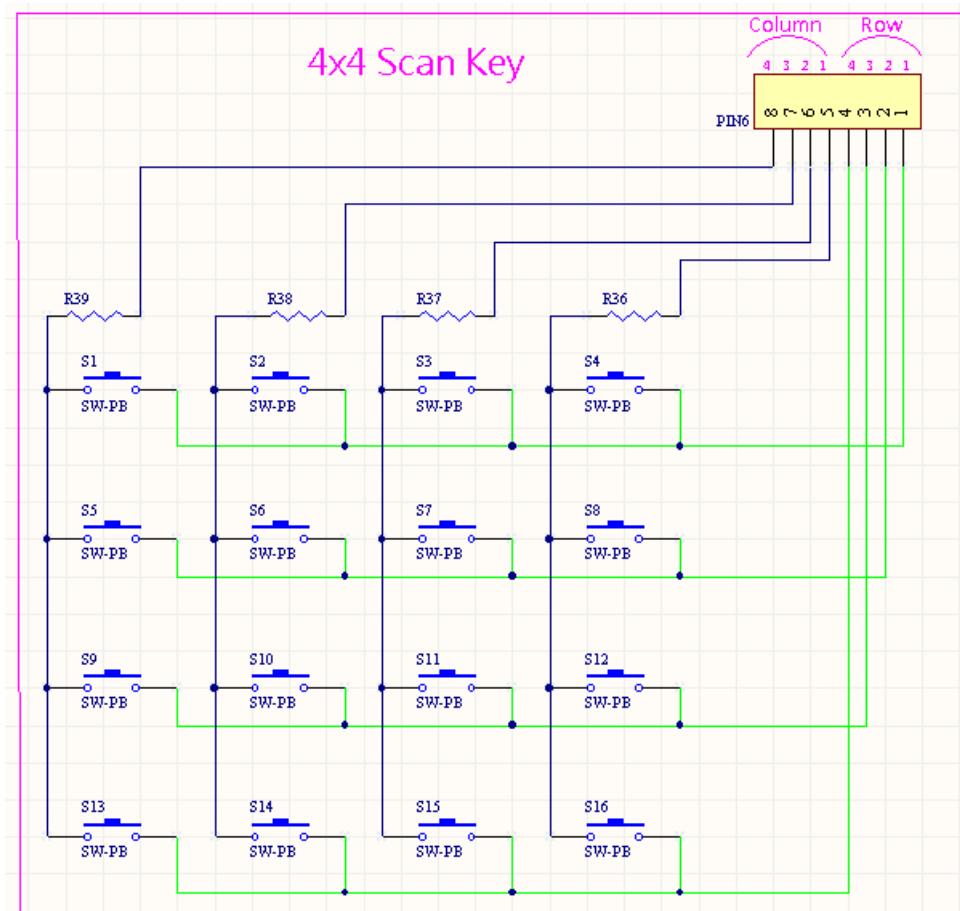


4x4 掃描按鍵模組：

此模組共有 8 個 I/O，在程式中可將 Column1~4 設為 “output 並輸出 HIGH (1)”，Row 設為 “input 並將上拉電阻打開”，這樣只要依序將 Column1~Column4 設為輸出 LOW (0)，並分別去檢查 Row1~Row4 讀進來的值是 HIGH (1) 或 LOW (0) 就可以知道按鍵有沒有被按下。

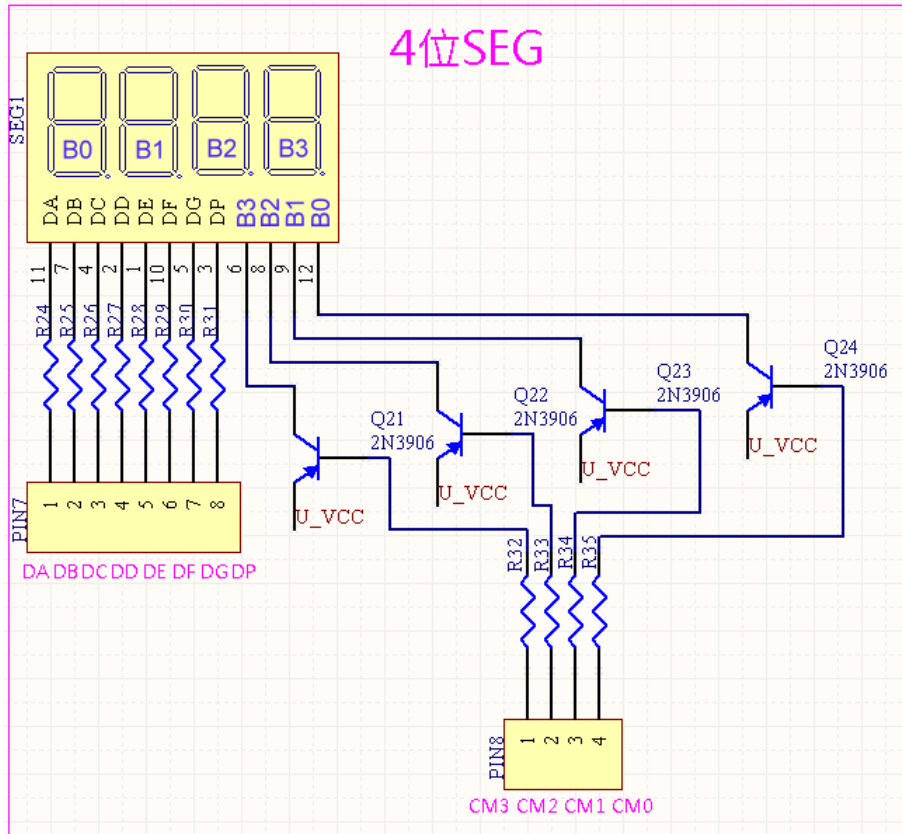
例如：

設定 Column1~3 輸出 HIGH (1)，Column4 輸出 LOW (0)，此時依序讀取 Row1~Row4 狀態檢查是 HIGH (1) 或是 LOW (0)...如果檢查到 ROW2 是 LOW (0)，其他維持 HIGH (1)，那就表示 S5 這顆按鍵被按下了。



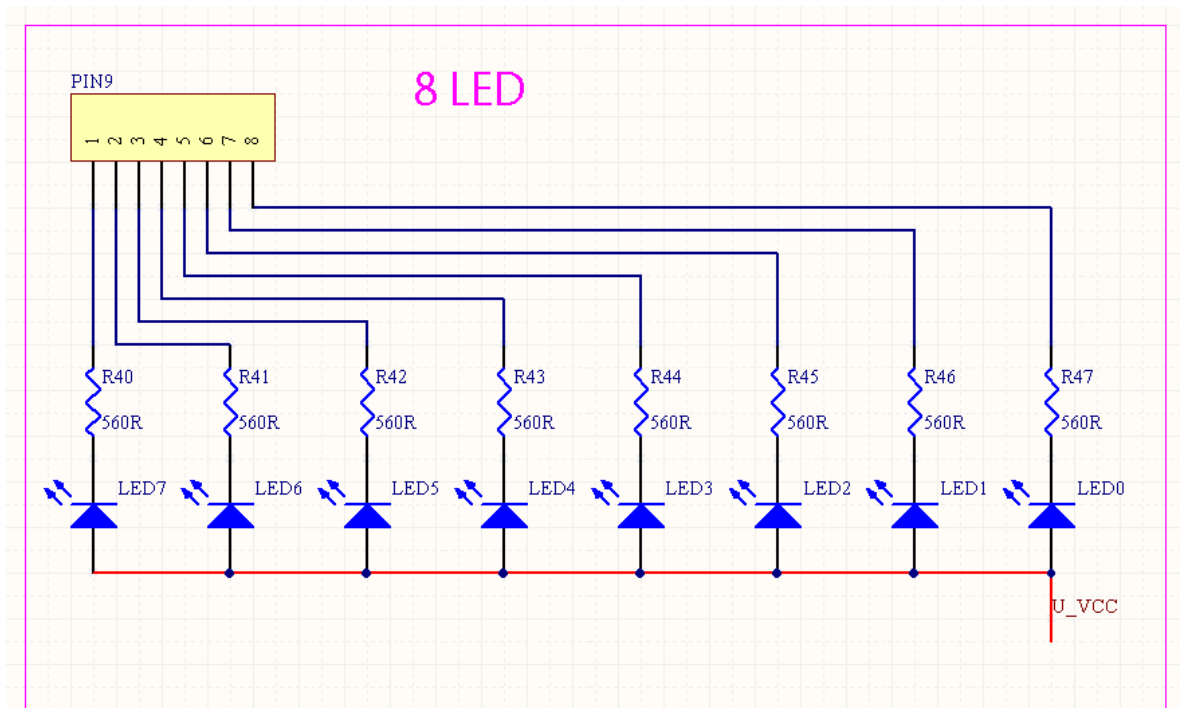
4 位數七段顯示器模組：

本實驗板的七段顯示器是共陽極，故理論上送 HIGH (1) 到 B0~B3，DA~DP 送 LOW，七段顯示器就會亮了，但是此模組沒有直接用 MCU 去驅動七段顯示器的 B0~B3，而是利用 Q21~Q24 這 4 顆 3906 去控制供電給 B0~B3，所以 MCU 送 LOW (0) 會令 3906 導通，送 HIGH (1)，會不導通。



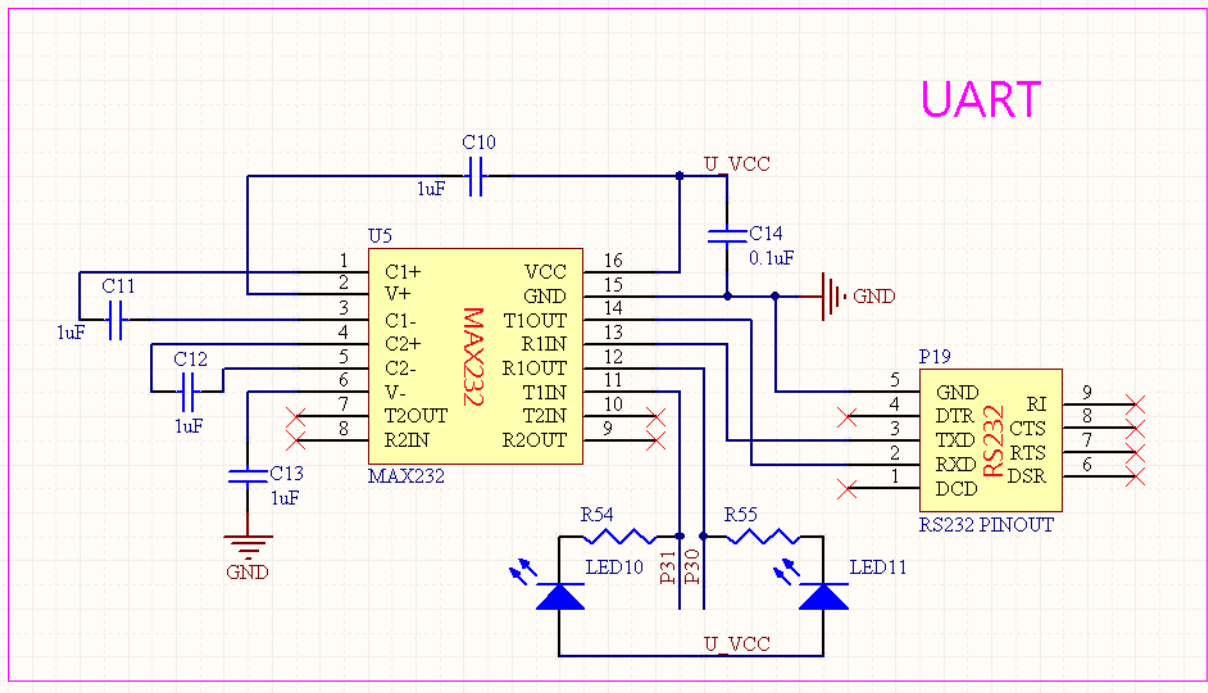
8 顆 LED 模組:

本模組為單純點亮 LED 燈的電路，LED 正端已接 VCC 5V，所以 MCU 的 IO 接到 Pin 腳後，I/O 控制 LED 是輸出 LOW (0) 點亮，輸出 HIGH(1)暗。



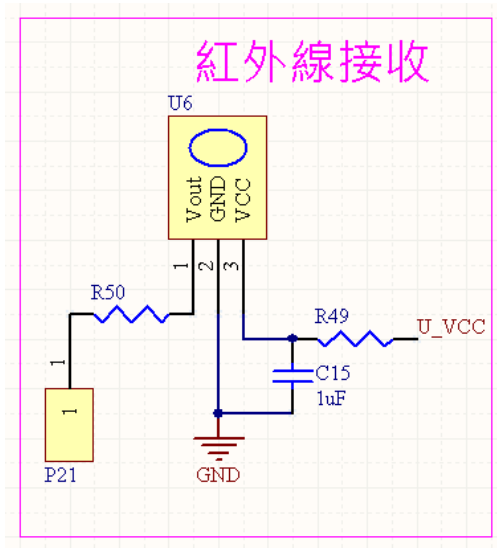
UART 通訊模組：

本模組內電路已直接接到 TM52F5288 的 TX 及 RX 接腳，TM52F5288 是使用 RS232 通訊，只需寫程式把 UART 運作參數設定好就可自動收送資料，並由此模組將電壓轉成標準 UART (RS232) 的電壓，與別的裝置通訊，當傳送或接收資料時，TX 及 RX 的 LED 會亮。



紅外線接收模組：

此模組內含一個紅外線接收器，將 P21 腳接到 MCU 的 I/O，MCU 該 Pin 腳要設為輸入腳，當收到紅外線訊號時，可利用 MCU 外部中斷功能或持續對該腳做輪詢的動作，就可以解析出紅外線的訊號。

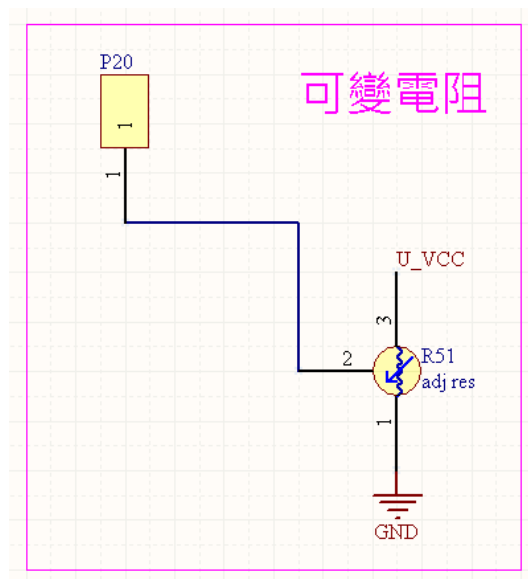


可調電阻電壓調變模組：

此模組的可變電阻兩端分別接 VCC 5V 及 GND，可透過旋轉可變電阻輸出 0~5V 的電壓，將 P20 PIN 腳接到 MCU 有 ADC 功能的腳位上，即可利用 MCU 內的 ADC 將可變電阻目前輸出的電壓轉為數位值，TM52F5288 的 ADC 是 12-bits 的，所以可以將類比輸入值切割成 4096 個階，也就是可以跑出數位數值 0~4095。

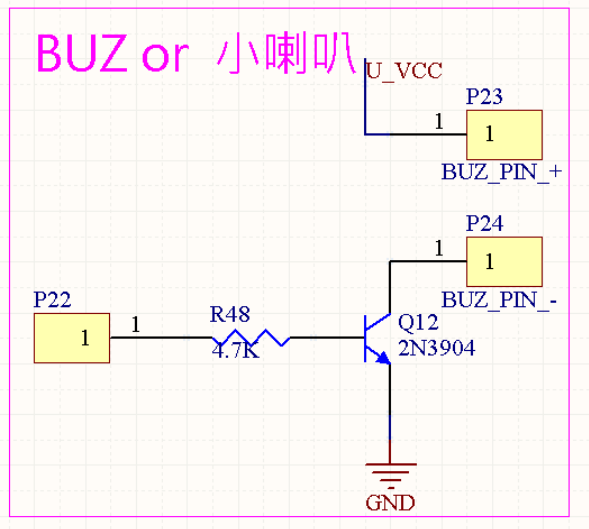
例如：

當鈕轉到中間，輸出 2.5V 電壓時，理論上 MCU 的 ADC 會吐出 2048 這個數位值。



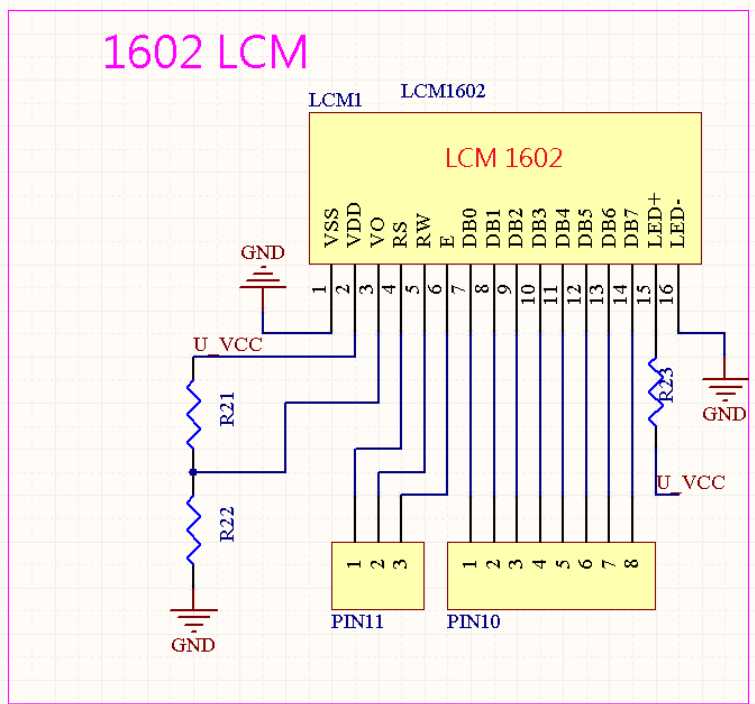
Buzzer 驅動模組：

此模組電路可供使用者驅動另購的喇叭或蜂鳴器，將 P22 PIN 腳接到 MCU 有 PWM 功能的腳位，即可利用 PWM 打出讓 BUZZER 或喇叭發出聲音的頻率訊號。



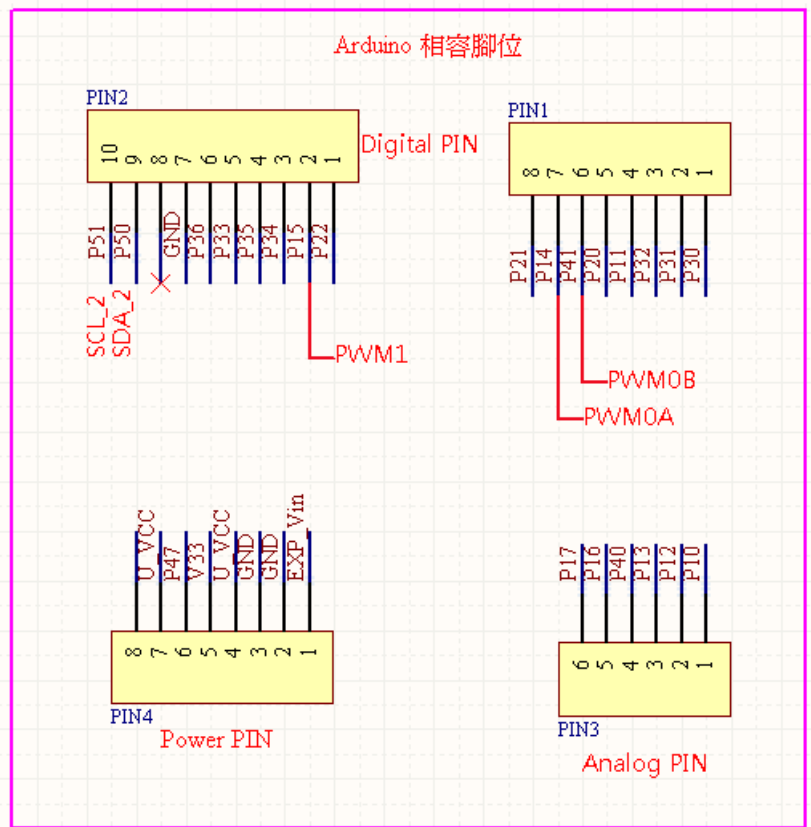
1602 文字型液晶顯示模組：

本模組內含一個裡面有字庫的文字型液晶顯示器，一行 16 個字元，共上下兩行，將 PIN10 及 PIN11 接到 MCU 的 IO 上，由 MCU 從這些 IO 依據液晶顯示器 SPEC 送指令，即可在螢幕上顯示字元。



Arduino 腳位擴充模組：

此模組電路將 TM52F5288 之部分 I/O，依據 Arduino 定義之腳位排列，故可將很多有趣的 Arduino 子板，當擴充板拿來插在本實驗板上，即可結合 Arduino 及 8051 兩者資源開發應用。



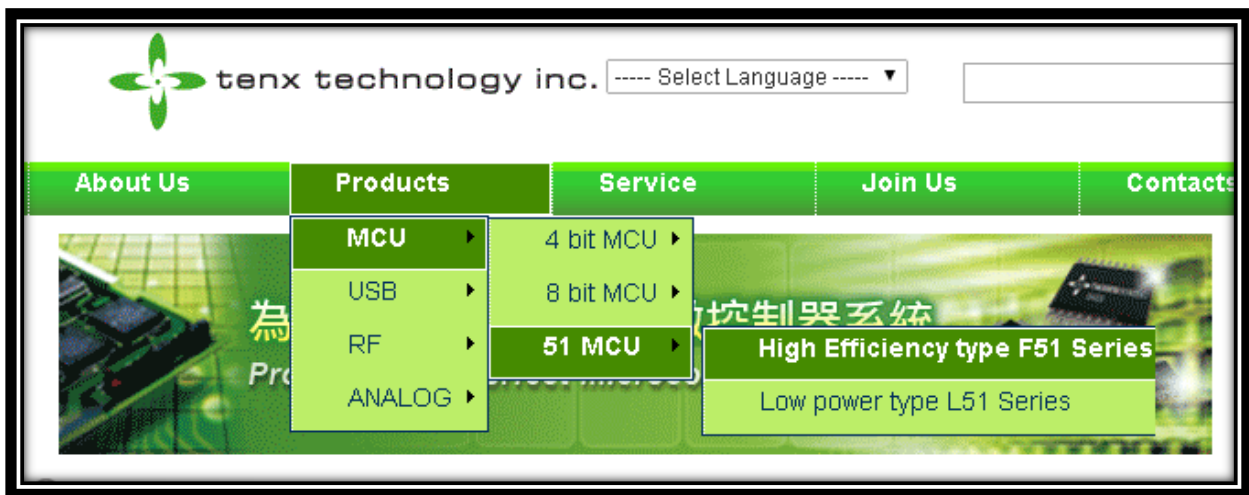
第二章、軟體開發環境介紹

2.1 開發軟體 Keil C

本實驗板之開發環境是使用 Keil C，在安裝完 Keil C 之後需再安裝十速科技 Keil C 擴充套件，並根據 2.2 節說明安裝。擴充套件可至十速科技網站下載


<http://www.tenx.com.tw/>

網頁中，請選擇 Product >> MCU >> 51 MCU >> High Efficiency type F51 Series



點擊 TM52F5288

High Efficiency type F51 Series

 [Parametric Search](#)

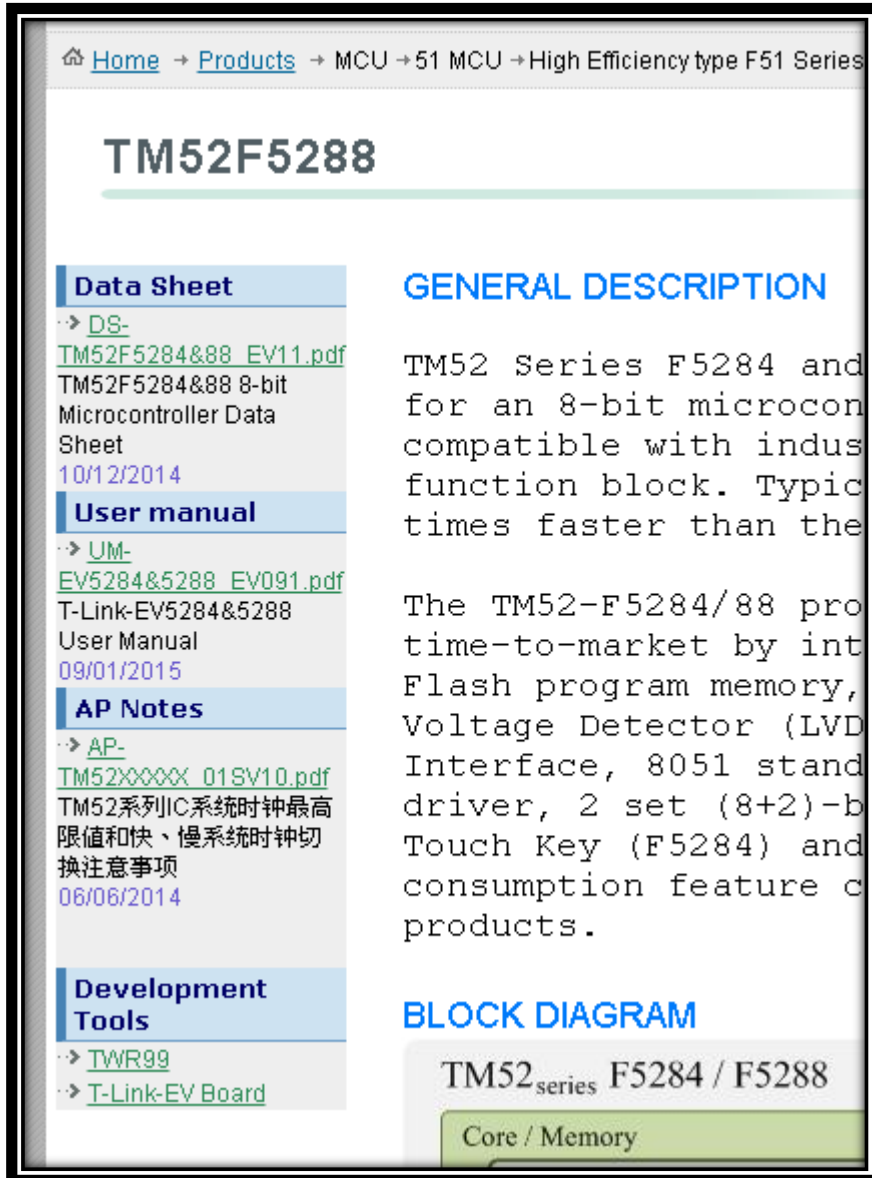
[Show All Spec View](#)

How to Use This Comparison Utility

 To Sort in ascending order
 To Sort in descending order

Product No.	CPU CORE	ROM Size	ROM Type	RAM bytes	GPIO	ADC 12-bit	Touch key	Touch ATK key	Touch TK	LCD max.	LED	Interface	PWM	UART	RTC	Voltage Range	Max. Freq.
TM52F5264	F51	8K	Flash	256	22	12-ch	--	--	--	--	--	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.6~5.5	8 MHz
TM52F5268	F51	8K	Flash	256	22	12-ch	4-ch	14-ch	--	--	--	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.6~5.5	8 MHz
TM52F5274	F51	8K	Flash	256+256	30	12-ch	--	--	--	4*18 dots	4*18 dots	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.6~5.5	8 MHz
TM52F5278	F51	8K	Flash	256+256	30	12-ch	4-ch	14-ch	--	4*18 dots	4*18 dots	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.6~5.5	8 MHz
TM52F5284	F51	16K	Flash	256+256	42	12-ch	--	12ch	--	8*20 dots	8*20 dots	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.9~5.5	6MHz
TM52F5288	F51	16K	Flash	256+256	42	12-ch	--	--	--	8*20 dots	8*20 dots	SPI	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.9~5.5	6MHz
TM52M5254	F51	4K	MTP	256	18	12-ch	--	--	--	--	--	--	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.9~5.5	6MHz
TM52M5258	F51	4K	MTP	256	18	12-ch	--	14ch	--	--	--	--	2-ch	Tx/Rx, 1-Wire	15 bit/T3	1.9~5.5	6MHz

此時出現 TM52F5288 介紹頁，此頁即可取得 TM52F5288 的 datasheet 及一些相關使用手冊，在 Development Tools 部分點擊 “T-Link-EV Board”



The screenshot shows a web page for the TM52F5288 microcontroller. The breadcrumb navigation is: Home → Products → MCU → 51 MCU → High Efficiency type F51 Series. The main heading is TM52F5288. On the left sidebar, there are three sections: Data Sheet, User manual, and AP Notes. The Data Sheet section includes a link to 'DS-TM52F5284&88_EV11.pdf' (TM52F5284&88 8-bit Microcontroller Data Sheet, dated 10/12/2014). The User manual section includes a link to 'UM-EV5284&5288_EV091.pdf' (T-Link-EV5284&5288 User Manual, dated 09/01/2015). The AP Notes section includes a link to 'TM52XXXXX_01SV10.pdf' (TM52系列IC系统时钟最高限值和快、慢系统时钟切换注意事项, dated 06/06/2014). Below these is the Development Tools section with links to 'TWR99' and 'T-Link-EV Board'. The main content area has a 'GENERAL DESCRIPTION' section starting with 'TM52 Series F5284 and for an 8-bit microcon compatible with indus function block. Typic times faster than the'. Below that is a paragraph: 'The TM52-F5284/88 pro time-to-market by int Flash program memory, Voltage Detector (LVD Interface, 8051 stand driver, 2 set (8+2)-b Touch Key (F5284) and consumption feature c products.'. At the bottom right, there is a 'BLOCK DIAGRAM' section with a diagram showing 'TM52_{series} F5284 / F5288' and a box labeled 'Core / Memory'.

在此即可取得需下載來安裝的 keil C 擴充套件，

下圖例即為 “F51DII_setupv1.3.0.0_AP3A2_V3.b7.exe” 這一個檔案。

(擴充套件會持續更新，故此檔案檔名後面的版號不一定與下圖例一樣，請放心下載安裝)

Home → Products → MCU → 51 MCU → High Efficiency type F51 Series → T-Link-EV Board → ToolsDetail

MCU > 51 MCU > High Efficiency type F51 Series > T-Link-EV Board

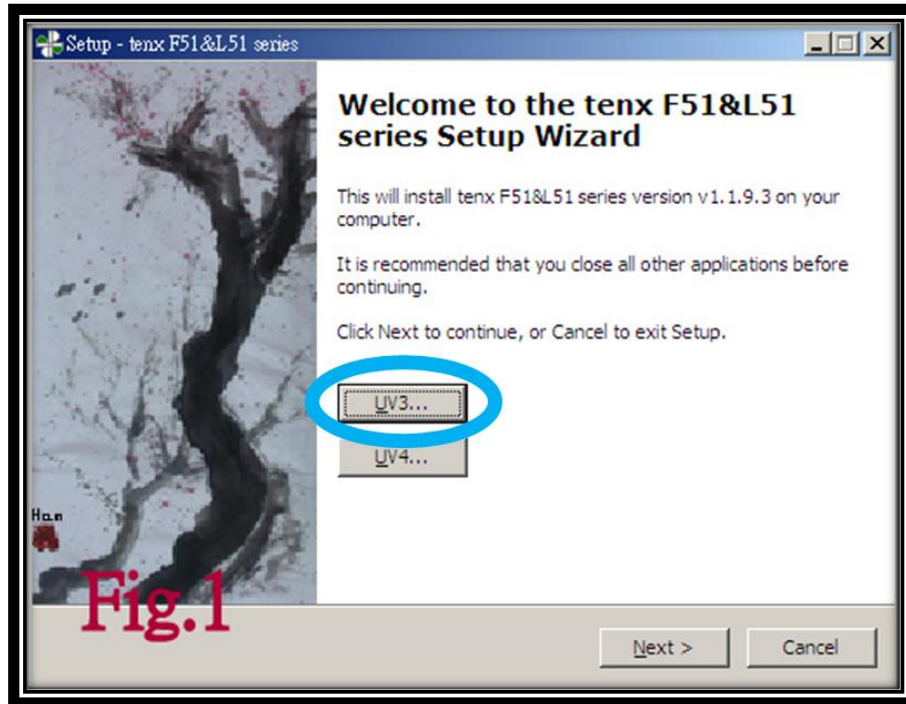
How to Use This Comparison Utility

To Sort in ascending order
 To Sort in descending order

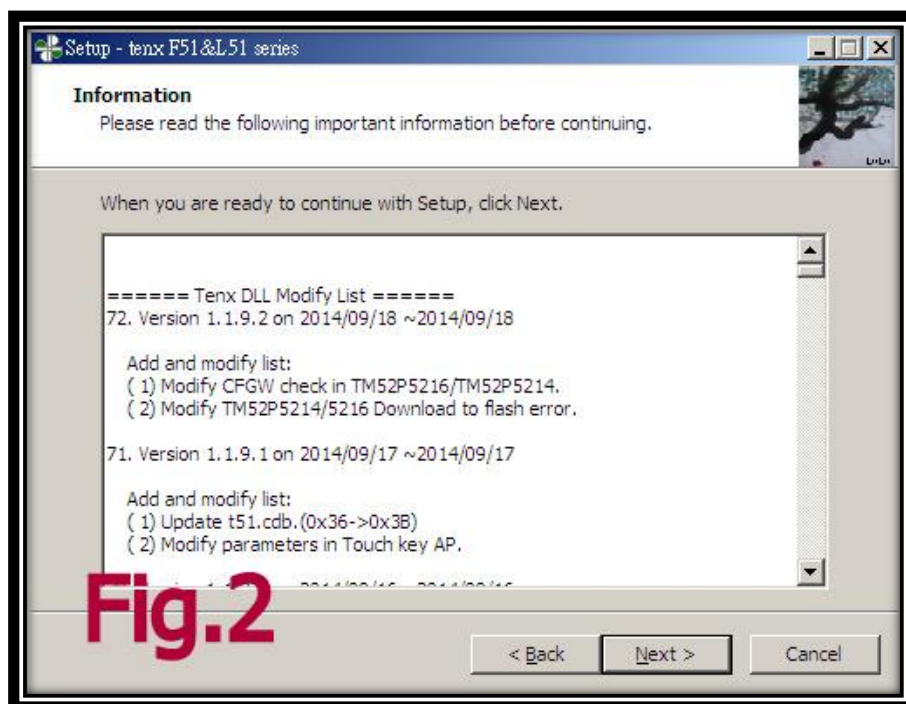
	FileName	VER	DESCRIPTION	UPDATE DATE
■	F51DII_setupv1.3.0.0_AP3A2_V3.b7_Release_Notes_EV.txt	1.3.0.0	F51DII_setupv1.3.0.0_AP3A2_V3.b7_Release_Notes	26/01/2015
■	F51DII_setupv1.3.0.0_AP3A2_V3.b7.exe	1.3.0.0	F51DII_setupv1.3.0.0_AP3A2_V3.b7.exe	26/01/2015

2.2 F51DLL_Setup 安裝-加入十速 MCU 套件

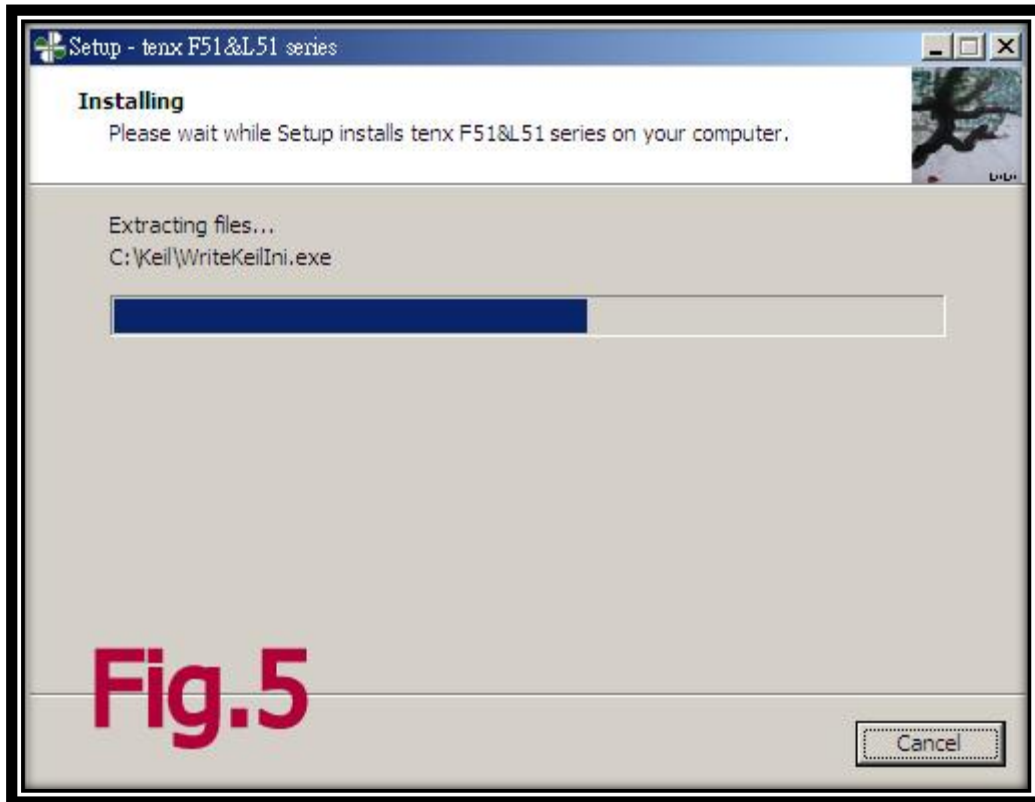
將十速科技的擴充套建安裝至 Keil C 裡，依照下列指示安裝，Fig.1 需依據您 Kiel C 版本來點擊，圖例為 Keil C UV3，如果您是安裝 UV4 版本，請點擊 UV4...選完版本請按 NEXT



後面 Fig2~5 即按照一般軟體安裝步驟下一步到結束。





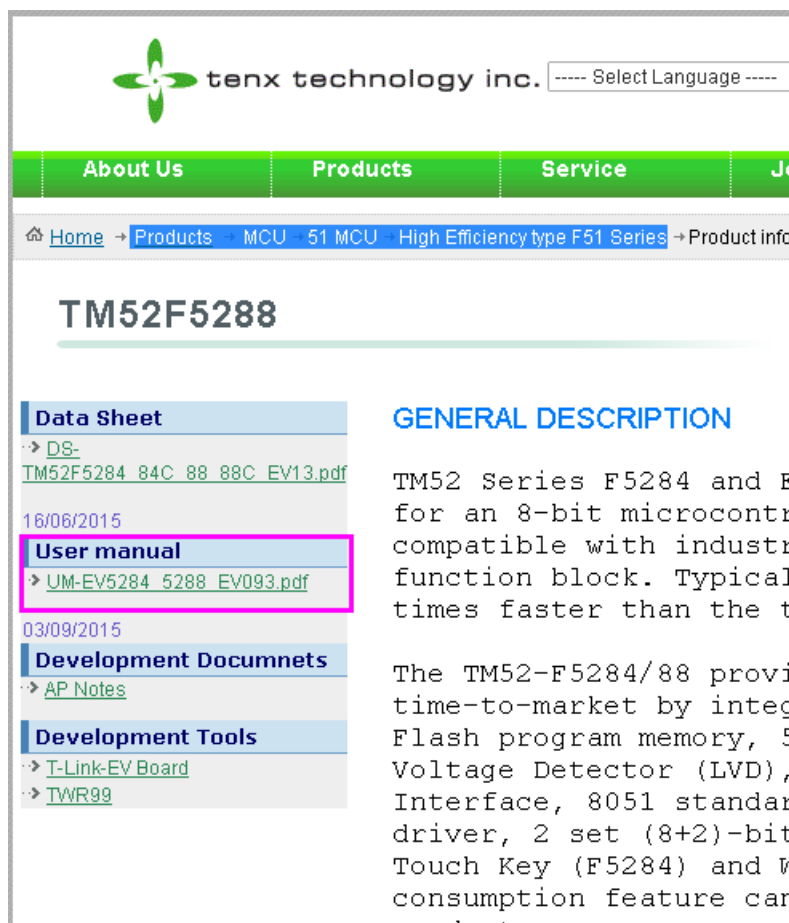


2.3 建立專案

寫程式之前，必須在 Keil C 上新建一個專案，並需針對此 IC 做設定。在 Keil C 上選擇 Project>>NEW>>選 uVision Project 之後的詳細步驟說明及圖例於官網有另一份 User Manual 詳細說明。

User Manual 放於官網 Products>>MCU51 MCU>>High Efficiency type F51 Series>>TM52F5288

網址：http://www.tenx.com.tw/product_detail.aspx?ProductID=309



tenx technology inc. ----- Select Language -----

About Us Products Service Jc

Home → Products → MCU → 51 MCU → High Efficiency type F51 Series → Product info

TM52F5288

Data Sheet
→ [DS-TM52F5284_84C_88_88C_EV13.pdf](#)
16/06/2015

User manual
→ [UM-EV5284_5288_EV093.pdf](#)
03/09/2015

Development Documents
→ [AP Notes](#)

Development Tools
→ [T-Link-EV Board](#)
→ [TWR99](#)

GENERAL DESCRIPTION

TM52 Series F5284 and F5288 for an 8-bit microcontroller compatible with industrial function block. Typical execution times faster than the t

The TM52-F5284/88 provide time-to-market by integrating Flash program memory, 5 Voltage Detector (LVD), Interface, 8051 standard driver, 2 set (8+2)-bit Touch Key (F5284) and low consumption feature can

如上圖框起來此份文件(UM-EV5284_5288)中的 4.9 開始，即有專案詳細設定的說明及圖例。

2.4 開新檔案開始寫 Code

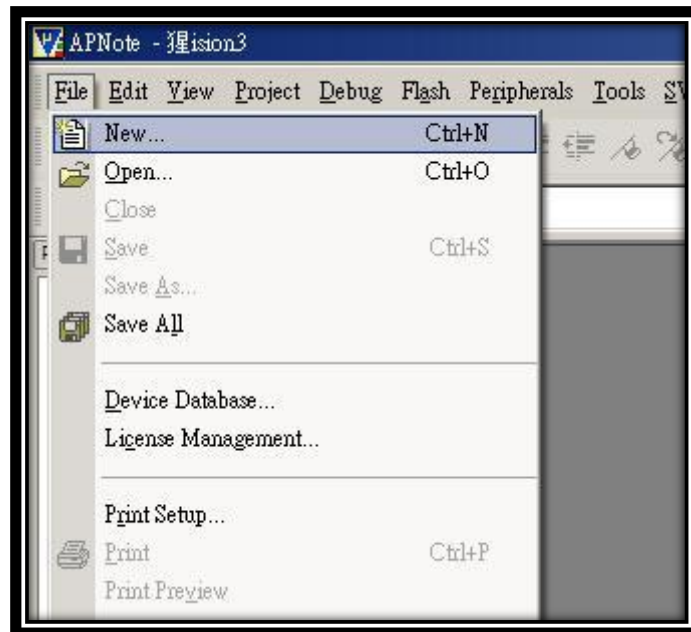


Fig.18 按 File >> New 即可開啟一個新的檔案

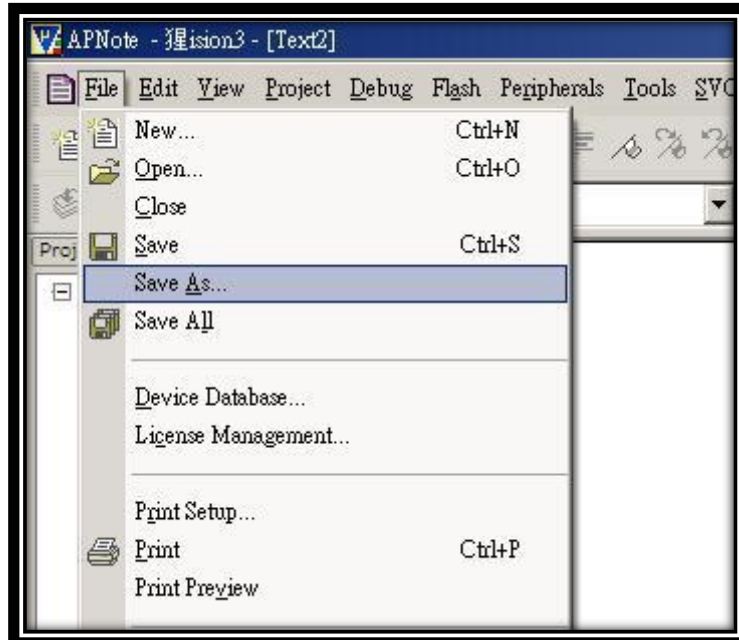


Fig.19 將此新檔案儲存（如存成.asm 為組語使用，存成.c 為 C 語言使用）

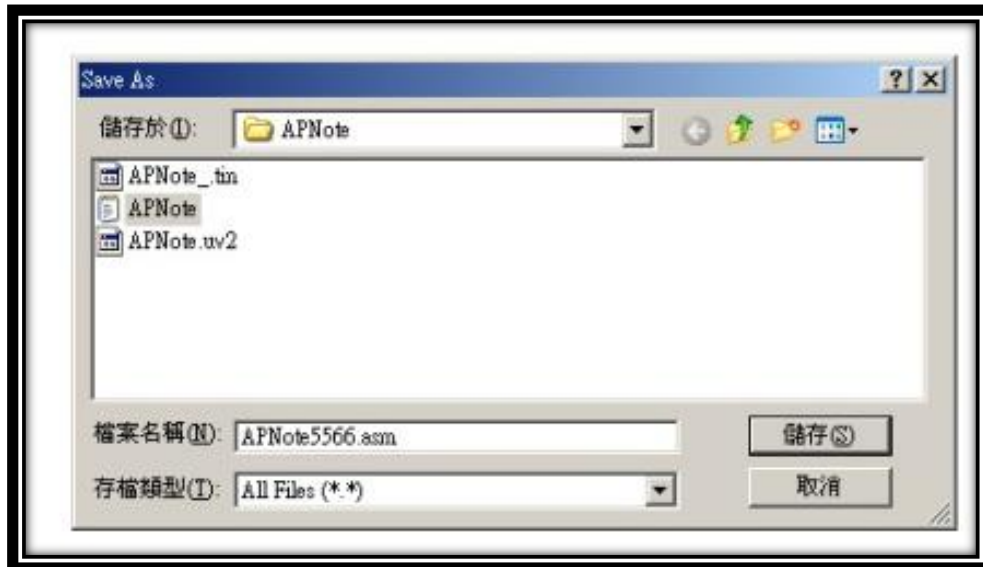


Fig.20 此例存為.asm (表示要用組語寫程式) 如果是使用 C 語言要存成.c

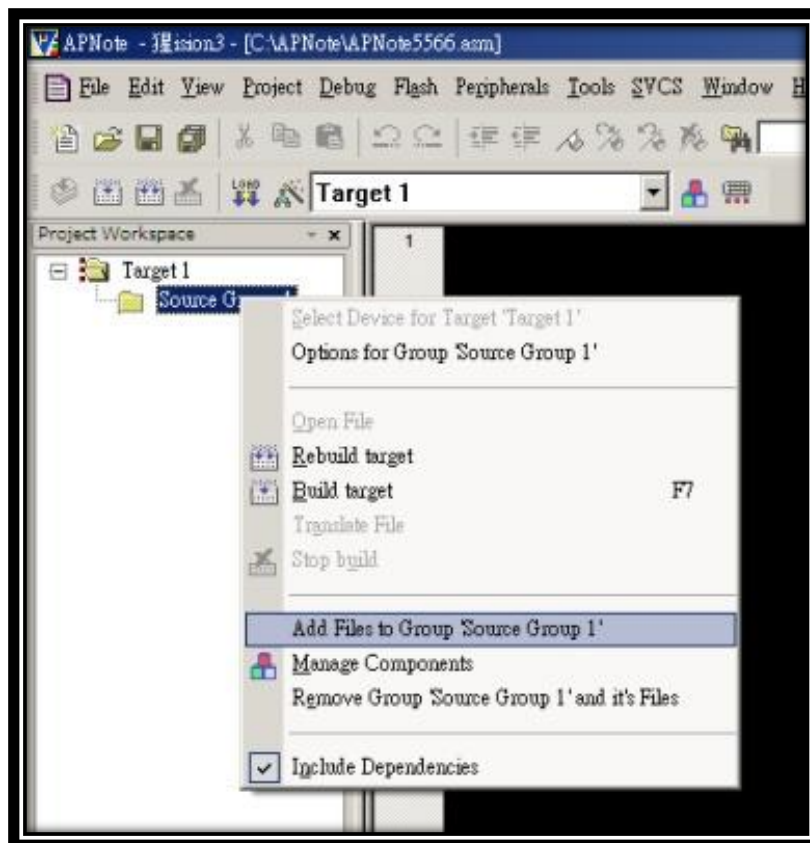


Fig.21 將此新文件加入專案內



Fig.22 選擇剛剛生出來的新文件並點選 Add

到這邊就已經把程式的文件加到專案中，可以開始寫程式囉！！！！

第三章、進入8051微控制器的世界

本章節利用一些簡單的範例程式，讓使用者可以較容易熟悉 Keil C 開發軟體以及 TM52F5288 實驗板的使用，並學會使用 TM52F5288 微控制器的大部分功能

共分為以下幾個小節：

3.0 [讓 IC 跑起來](#)

使用 MCU 的基本初始運作情形及初始設定說明。

3.1 [點 LED 燈](#)

說明 IO 設定及控制，並說明如何使用 Keil C 搭配本實驗板，在開發程式時進入 Debug 模式，做線上模擬及除錯以及注意事項。

3.2 [點亮七段顯示器](#)

說明 IO 設定及控制、timer 應用以及七段顯示器顯示原理。

3.3 [LED 呼吸燈](#)

MCU 上的 PWM 功能應用、Timer 應用以及 LED 調光原理。

3.4 [掃描按鍵輸入](#)

說明 IO 設定及控制、timer 應用及掃描按鍵程式運作方式。

3.5 [讓 PC 跟實驗板利用 UART \(RS232\) 通訊](#)

說明 UART 設置及鮑率計算說明，PC 端收發軟體教學最後建立連線。

3.6 [可變電阻模組類比數位轉換](#)

MCU 上 ADC 功能設定以及應用。

3.7 [1602 LCM 文字型液晶模組](#)

利用控制 IO 發送指令給液晶螢幕模組，顯示我們要的資訊。

3.1 讓 IC 跑起來 (使用 MCU 的基本初始設定)

通常 MCU 一上電，就會從程式記憶體的位址 0 開始跑程式，MCU 會依據控制暫存器裡面的設定值來運作，所以寫程式只要依據 datasheet 內的說明來寫值到控制暫存器裡，就可以令 MCU 做出開發者想要的功能。上電後會先依據控制暫存器的預設值來運作，所以通常在程式一開始，會先寫一些基本設置的程式去設定 MCU，例如運作時脈（**注意：TM52F52885 時脈最快只能跑 6MHz，故如果跑快鐘一定要設定除頻除以 2 以上**）以及 MCU 腳位輸出入設定、中斷設置，其餘功能設置...等，所以如果沒寫程式去改變控制暫存器的值，當然就都是以預設值的設定運作囉！下面一段小程式當範例，程式功能是讀取 Port1 各 Pin 的訊號，再把收到的訊號從 Port0 各 Pin 輸出。

```
#include <REGtenxTM52F5288.H>           //這是 TM52F5288 的定義檔，要 include 這個，Compiler
                                        //才會認得
                                        //程式中 CLKCON、POOE、P1MODH.....這些東西。

void main (void)
{
    //系統時脈配置
    //CLKCON=0x23;           //0010 0011    這是預設值
    CLKCON=0x22;           //0010 0010    改變除頻
    CLKCON=0x26;           //0010 0110    切換成快鐘

    //IO 設定
    //Port 0
    P0OE=0xFF;             //Port0 I/O 預設值是 0 (input)，把全部設成 1 (output)
    P0=0xFF;               //要輸出的 DATA 預設值

    //Port 1
    P1MODH=0x00;          //Port 1 的 7~4 I/O 設定
    P1MODL=0x00;          //Port 1 的 3~0 I/O 設定
    //P1=0xFF;             //因為 Port 1 當輸入，所以可以從 P1 讀到外部輸入進來的訊號



    while (1){
        P0=P1;
    }
}
```

程式一開始的進入點會從 void main (void) 這個 function 的第一行開始跑，我們首先是想要改變 TM52F5288 的運作的速度，在 datasheet 中的“5.Clock Circuitry and Opreation Mode”中有詳細介紹 TM52F5288 的時脈設定，可跑外部快鐘、外部慢鐘、內部快鐘 (7.3728 MHz) 及內部慢鐘 (80 KHz)，並有除頻器，經過除頻後的振盪頻率當作 TM52F5288 的運作時脈。在 datasheet 中找到下面這個表格，可以知道 CLKCON 這個暫存器位址是在 0xD8，對它寫入資料可以設定時脈來源，以及除頻的設定，然後**預設值是 0x23 (內部慢鐘以及除頻除以 1)**

SFR D8h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKCON	SCKTYPE	FCKTYPE	KICKSXT	STPPCK	STPFCK	SELFCK	CLKPSC	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	1	0	0	0	1	1

這個小程式設計想要跑內部快鐘以及除頻除以 2，在 datasheet 內的說明可以知道，想要除頻除以 2 要改變 CLKPSC 設定，要從慢鐘切到快鐘要改變 SELFCK 設定，改變運作時脈必須一次只改一個項目的設定，避免一邊運行一邊改變多個設置可能產生錯誤，所以程式先寫 0x22 到 CLKCON 改變除頻設定，再寫 0x26 到 CLKCON 改變運作來源，從慢鐘切到快鐘，程式只要執行到這邊之後後續程式內容就會以內部快鐘以及除頻除以 2 的速度來執行程式。

再來我們要讓程式把 Port1 收到的值，再從 Port0 送出去，在 datasheet” 7.I/O Port” 有 I/O 設定的詳細說明，要把 Port 0 設成輸出就是把 0xFF 寫到 P0OE 去，要把 Port 1 設成輸入就是把 0x00 寫到 P1MODH 及 P1MODL，就可以把輸出輸入設定好了，最後用 while 把 P0=P1 包起來，程式執行結果就會一直把 P1 收到的東西丟到 P0 輸出了，記得寫程式前要先開啟專案設定好專案設定，並產生寫 Code 的檔案（不知道的看這邊），寫完程式後，要先按 Keil C 開發軟體上工具列的

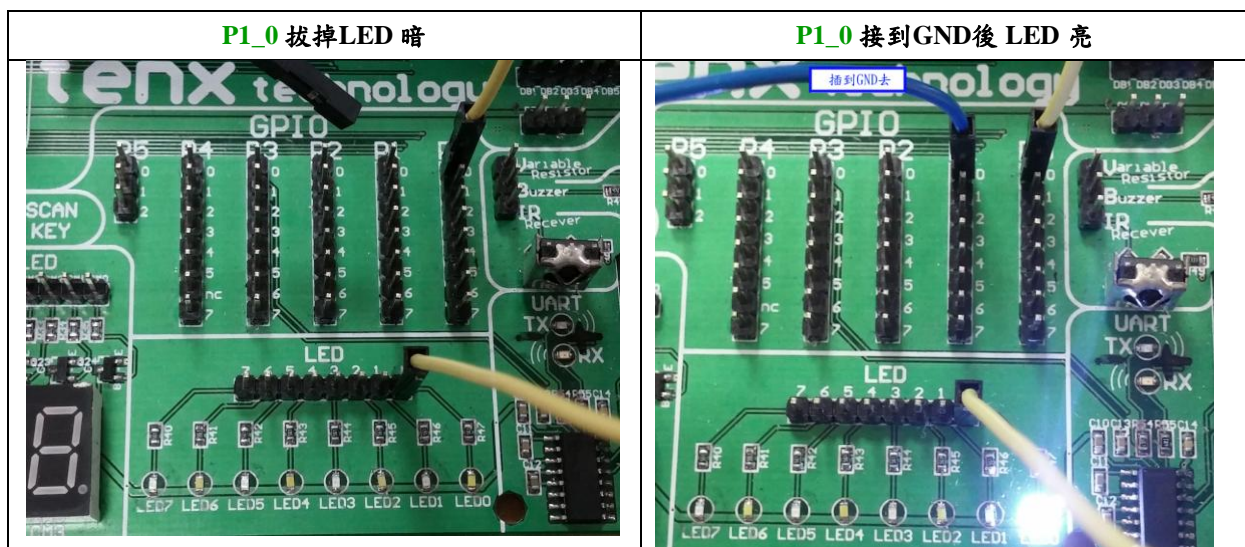
 把程式 Compiler，如果都沒錯誤再按  將程式燒錄到實驗板上。

燒錄完可以用個簡單方法測試一下這個小程式：

1. 拿兩條杜邦線
2. 把 P0 的第 0 PIN (以下簡稱 P0_0) 接到 LED 燈模組的任一 PIN 上
3. 把 P1 的第 0 PIN (以下簡稱 P1_0) 接到板子左上方那邊有 GND 的腳位可以插

結果可以發現，當 P1_0 沒有插到 GND 時，因為程式中有把 P1_0 的上拉電阻功能打開，所以 P1_0 輸入的值會是 1，所以依照程式的動作 P0_0 會把剛剛收到的 1 送出去，所以這時後板上的 P0_0 是輸出 5V (因為我們 IC 是用 5V 運作) ...這時後 LED 燈不會亮；當 P1_0 插到 GND 時，因為 P1_0 被拉到 GND (0V)了，所以 P1_0 輸入的值會變 0，所以依照程式的動作 P0_0 會把剛剛收到的 0 送出去，所以這時後的 P0_0 是輸出 0V...這時後 LED 燈就亮囉!!!

(如果不知道為何 LED 是 1：暗、0：亮...請回頭看本份文件的 1.2 實驗板模組介紹的 LED 模組)



3.2 點 LED 燈 (IO 控制、debug 模式簡介)

經過上一單元的洗禮，這個範例應該很好看懂，這個程式的功能是要讓 LED 模組的 LED 從 LED0 一直亮到 LED7 然後全滅，再回頭從 LED0 一直亮到 LED7 然後又全滅，持續循環。

```
#include <REGtenxTM52F5288.H>
#define LED0 P0_0
#define LED1 P0_1
#define LED2 P0_2
#define LED3 P0_3
#define LED4 P0_4
#define LED5 P0_5
#define LED6 P0_6
#define LED7 P0_7

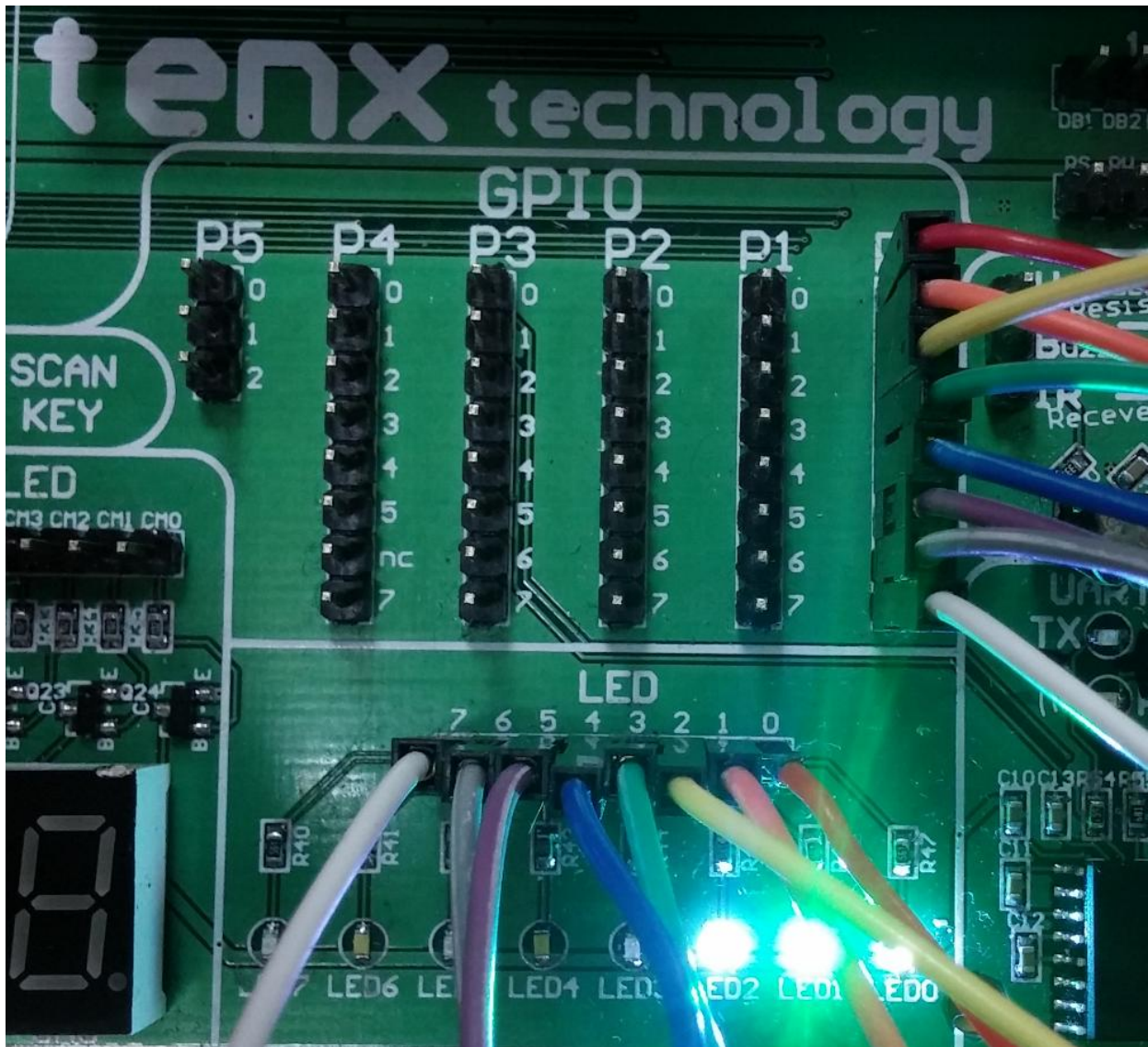
void delay (int count);    //delay 的副程式放在最下面，
                          //但因為 main 裡有用到所以要在這邊先宣告一下

void main (void){
    //IO 設定 Port 0 全部設成輸出
    P0OE=0xFF;
    P0=0xFF;
    //維持慢鐘，改變除頻
    CLKCON=0x20;          //0010 0000    除頻改為除以 16，讓 MCU 慢慢的跑

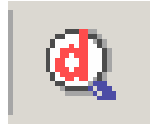
    //主程序循環
    while(1){
        P0= 0xFF;        //Port 0 各 Pin 全輸出 1 (LED0~7 全滅)
        delay (50);     //延遲
        LED0= 0;        //LED0 輸出 0 (LED0 亮)
        delay (50);     //延遲
        LED1= 0;        //LED1 輸出 0 (LED1 亮)
        delay (50);     //延遲
        LED2= 0;        //LED2 輸出 0 (LED2 亮)
        delay (50);     //延遲
        LED3= 0;        //LED3 輸出 0 (LED3 亮)
        delay (50);     //延遲
        LED4= 0;        //LED4 輸出 0 (LED4 亮)
        delay (50);     //延遲
        LED5= 0;        //LED5 輸出 0 (LED5 亮)
        delay (50);     //延遲
        LED6= 0;        //LED6 輸出 0 (LED6 亮)
        delay (50);     //延遲
        LED7= 0;        //LED7 輸出 0 (LED7 亮)
        delay (50);     //延遲
    }
}
//利用數個空指令達成延遲的副程式，依據程式邏輯呼叫此副程式時()內數字越大，for 就跑越多次。
void delay (int count){
    int    i;
    for (i= 0; i<count; i++);
}
```

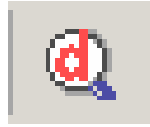
在實驗板上我們將 P0_0 接到 LED 模組的 LED0、P0_1 接到 LED1、P0_2 接到 LED2、...、P0_7 接到 LED7，然後利用 `#define`，把 LED0 這幾個字定義為跟 P0_0 相同...其他 Pin 也一樣定義好，這樣我們寫程式時就會比較直覺，LED0=0，就是板上 LED0 這一顆點亮，不然如果是寫 P0_0=0 程式寫大之後，可能會忘記 P0_0 是什麼東西！至於為什麼要叫 LED0，不叫 Light0...恩恩！！純粹只是因為板子上是印 LED0~LED7，所以定義一樣的字符會比較好辨識！

接線如下圖，用 8 條杜邦線，P0_0 接到 LED0、...、P0_7 接到 LED7

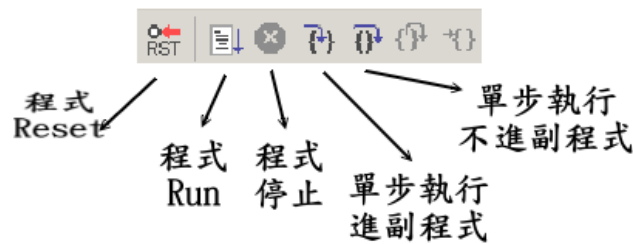


剛才都是直接將 Code 燒到實驗板上面看結果，當在開發程式階段，有時候程式如有寫錯燒到板子上看也看不出個什麼東西，本實驗板可配合 Keil C 做線上模擬及 Debug，當程式 Compiler 完，

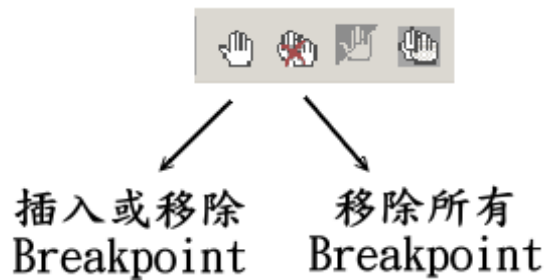


在 Keil C 工具列上按  就會進入 debug 模式（再按一次就退出），此時工具列會出現 debug 模式控制程式運作的按鈕～

注意！因 Debug 模式需佔用 P1_2 及 P1_3 來與 PC 通訊，故程式如有用到 P1_2 及 P1_3，在 Debug 模式會干擾導致出問題。



如上圖所示 Debug 工具可以控制程式要一直跑還是要停止或是單步執行！方便開發者觀看程式運作及檢查程式有沒有錯，另外如下圖所示還可以在程式中設定數個 Breakpoint（進 debug 模式後，程式前面有灰灰方塊的就可以設 Breakpoint）



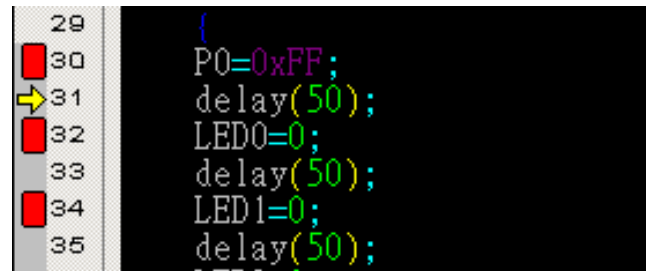
Breakpoint 可讓程式在 Run 的時候，當到跑到有設定 Breakpoint 的那一行會停住！可利用此功能檢查程式運作是否有誤，或是否邏輯有問題沒跑到那一行...等應用。如下圖所示我們在程式第 30，32，34 放 breakpoint，然後按 Run，當程式跑到準備執行第 30 行時會停住，此時表示準備跑 30 這行但是還沒有跑！

```

29  {
30  P0=0xFF;
31  delay(50);
32  LED0=0;
33  delay(50);
34  LED1=0;
35  delay(50);

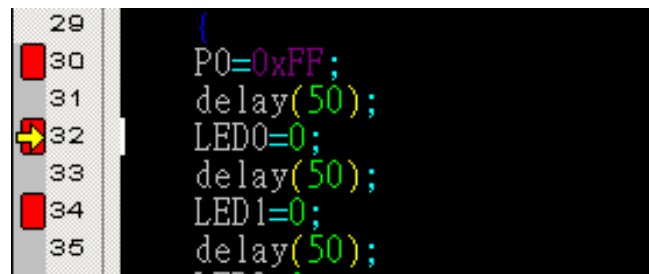
```

這時候我們按一下單步執行，如下圖程式就會跑完 30 換準備執行 31 行，我們就可以去看實驗板上 P0 的是不是都送 HIGH 了。



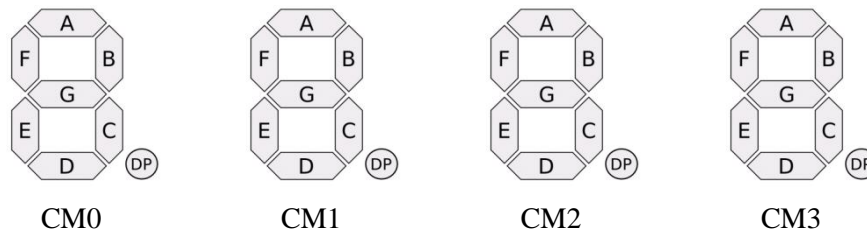
```
29 {
30 P0=0xFF;
31 delay(50);
32 LED0=0;
33 delay(50);
34 LED1=0;
35 delay(50);
```

再按 Run 過一下子程式跑完 delay 跑到 32 行時才會自動停住！！！！

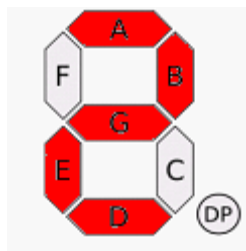


```
29 {
30 P0=0xFF;
31 delay(50);
32 LED0=0;
33 delay(50);
34 LED1=0;
35 delay(50);
```

3.3 點亮七段顯示器（IO 控制、timer 應用）



本實驗利用 GPIO 控制七段顯示器的 DA、DB、DC、...、DG、DP，送出顯示排列的訊號，讓顯示器顯示出數字，也就是說如果我們要顯示 2，就讓 DA.DB.DD.DE.DG 亮，如下圖所示：



又因為實驗板上有 4 顆七段顯示器，DA~DP 是共用線，透過控制 CM0~CM3 就可以指定要亮哪一個位數的七段顯示器！本實驗想要讓數字自動由 0~9 變化，且 1 秒鐘變化一次，所以用到了 Timer2 的 Timer 中斷，Timer2 的詳細使用說明，請看 Datasheet 的“8. Timers”，Timer2 是 16 位元計數器，計數到 65536 溢位產生中斷，每一個指令周期計數加一，且可以設定中斷之後的 reload 值，所以不一定從 0 開始數！這樣我們透過計算並設定 reload 值就可以讓 timer2 剛好 1 秒鐘中斷一次。

這個範例的時脈我們用預設值的慢鐘 80 KHz，除頻設除以 1，TM52F5288 的指令周期是 2 個系統時脈，所以指令周期頻率是 40K，也就是 25us 會讓 timer2 加 1，所以當 timer2 加了 40000 次，就是一秒鐘了...所以要計算 reload 值，就是用 65536-40000 等於 25536，16 進制是 0 x 63C0，這就是程式為何 reload 是設定 RCP2H = 0 x 63；RCP2L = 0 x C0；的由來！

線路請依程式中定義的一樣，把 P1_4 接到 CM0、P1_3 接到 CM1、....共 12 支接腳

```
#include <REGtenxTM52F5288.H>
#define SEG_CM0 P1_4
#define SEG_CM1 P1_5
#define SEG_CM2 P1_6
#define SEG_CM3 P1_7
(下頁續.....)
```



```
#define SEG_BUS P0
#define SEG_DA P0_7
#define SEG_DB P0_6
#define SEG_DC P0_5
#define SEG_DD P0_4
#define SEG_DE P0_3
#define SEG_DF P0_2
#define SEG_DG P0_1
#define SEG_DP P0_0

bit    bf_1_Sec;
unsigned char    LED_LOOP_CNT;
void Seg_Num (unsigned char number);    //宣告七段顯示器顯示設置的副程式

void main (void){
    //IO 設定
    //Port 0 DA~DP
    P0OE= 0xFF;
    P0= 0xFF;
    //Port 1 CM0~CM3
    P1MODH= 0xAA;

    //用預設系統時鐘配置
    //CLKCON= 0x23;    //0010 0011 // SRC FRC 慢鐘 div1

    //timer2 設定
    T2CON= 0x04;    //0000 0100
    TH2= 0x63;
    TL2= 0xC0;
    RCP2H= 0x63;    //reload 值高位元
    RCP2L= 0xC0;    //reload 值低位元
    ET2= 1;    //開 Timer2 中斷
    EA= 1;    //開中斷

    SEG_CM0= 0;    //只要亮 CM0 就好了
    SEG_CM1= 1;
    SEG_CM2= 1;
    SEG_CM3= 1;

    //循環
    while (1){
        if (bf_1_Sec==1){    //1 秒鐘才會進來一次(Timer2 中斷裡會把這個旗標設 1)
            bf_1_Sec= 0;    //進來後把旗標清 0
            LED_LOOP_CNT++;    //LED_LOOP_CNT 用來產生 0~9 的數字
            if(LED_LOOP_CNT>= 10){
                LED_LOOP_CNT= 0;
            }
            Seg_Num (LED_LOOP_CNT);    //呼叫設定七段顯示器的程式顯示
        }
    }
}

LED_LOOP_CNT 數字
}

}

(下頁續.....)
```

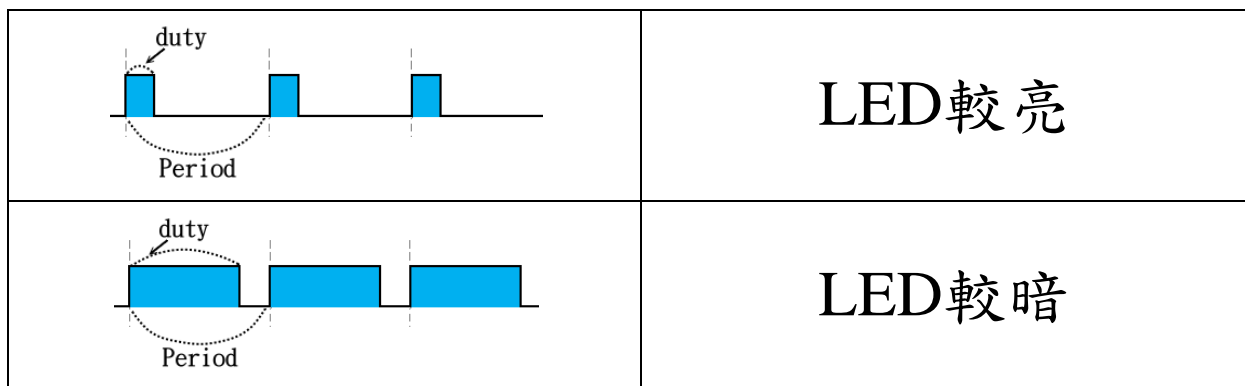


```
//七段顯示器顯示哪幾類的設定副程式
void Seg_Num (unsigned char number){
    switch (number){
        case 0:
            SEG_BUS= 0x03;          //0000 0011
            break;
        case 1:
            SEG_BUS= 0x9F;          //1001 1111
            break;
        case 2:
            SEG_DA= 0;              //也可以一根腳一根腳送，但會浪費程式空間
                                   //及運作時間
            SEG_DB= 0;
            SEG_DC= 1;
            SEG_DD= 0;
            SEG_DE= 0;
            SEG_DF= 1;
            SEG_DG= 0;
            SEG_DP= 1;
            break;
        case 3:
            SEG_BUS= 0x0D;          //0000 1101
            break;
        case 4:
            SEG_BUS= 0x99;          //1001 1001
            break;
        case 5:
            SEG_BUS= 0x49;          //0100 1001
            break;
        case 6:
            SEG_BUS= 0x41;          //0100 0001
            break;
        case 7:
            SEG_BUS= 0x1B;          //0001 1011
            break;
        case 8:
            SEG_BUS= 0x01;          //0000 0001
            break;
        case 9:
            SEG_BUS= 0x09;          //0000 1001
            break;
        default:
            break;
    }
}

//Timer2 中斷副程式，Timer2 是第 5 個中斷源 (Datasheet 有說明)
void TIMER2_int (void) interrupt 5{
    TF2 = 0;                       //中斷旗標歸 0
    bf_1_Sec=1;                     //自設的 1 秒旗標設 1，放在這裡表示一秒鐘會被設成 1 一次
}
}
```

3.4 LED 呼吸燈（PWM 應用、timer 應用）

利用 TM52F5288 的 PWM 功能自動產生調變波形，可以令 LED 因快速閃爍，達成調光的效果，在 datasheet“10. PWMs”中有詳細介紹 PWM 設定及運作情形，下圖為 PWM 會產生之波型與 LED 亮度示意圖～因為 LED 是訊號拉 LOW 時亮，所以當 PWM 的 duty 越小，LED 會越亮，本範例寫一程式讓 PWM 的 Duty 會漸漸變大，再漸漸變小，達成呼吸燈的功能。



本範例電路很簡單，只需要一條線，連接 P1_5 跟 LED 模組上隨便一顆 LED 燈即可，程式設定 PWM 的時脈及 Period 後將 PWM 運作設定開啟，之後利用 Timer 中斷去定期改變 PWM 的 duty 調整亮度，即可達成漸漸亮再漸漸暗的效果！

```

#include <REGtenxTM52F5288.H>

#define LED0 P1_5

bit    bfUP_DOWN;           //程式運作旗標 1:數值遞增 0:數值遞減
bit    bf_100ms;

void main (void)
{
    //IO 設定 PWM1 輸出 PIN 是 P1.5
    //Port 1
    P1MODH= 0x08; //00 00 10 00
    LED0= 1;

    //CLKCON= 0x23;           //0010 0011
    CLKCON= 0x20;           //0010 0000    改變除頻變除 16
    CLKCON= 0x24;           //0010 0100    改快鐘

    //PWM1 設定
    PWM1PRD= 250;
    PWM1DH= 20;
    PWMCON= 0x08;           //PWM1 時脈設定為快鐘/4
    PINMOD= 0x80;           //PWM1 打開，並將訊號輸出至 P1_5
    
```

(下頁續....)

```

bfUP_DOWN= 1;          //程式運作用到的旗標初始值 1

//timer2 設定
T2CON= 0x04; //timer2 開始計數
TH2= 0xA6;
TL2= 0x00;
RCP2H= 0xA6;          //快鐘除以 16，reload 設 0xA600 等於 100ms 中斷一次
RCP2L= 0x00;
ET2= 1;
EA= 1;

//主程序循環
while (1)
{
    if (bf_100ms==1)    //100ms 進來一次
    {
        bf_100ms= 0;
        //*****以下這段 Code 會讓 PWM1DH 數值遞增，超過 230 數值變遞減，持續來回*****
        if (PWM1DH<20)
        {
            bfUP_DOWN= 1;
            PWM1DH= PWM1DH+20;
        }
        else if (PWM1DH>230)
        {
            bfUP_DOWN= 0;
            PWM1DH= PWM1DH-20;
        }
        else
        {
            if (bfUP_DOWN==1){
                PWM1DH= PWM1DH+10;
            }
            else{
                PWM1DH= PWM1DH-10;
            }
        }
        //*****
    }
}

//Timer2 中斷副程式
void TIMER2_int (void) interrupt 5{
    TF2= 0;
    bf_100ms= 1;
}
    
```

執行結果，可以看到 LED 漸漸亮漸漸暗持續循環！



3.5 掃描按鍵輸入 (IO 控制、timer 應用)

本範例實做一個掃描按鍵的程式，將程式結果從 P0 送出...可接到 LED 模組觀看結果，掃描按鍵的原理請看本文件 1.2 介紹 ([在這裡](#))，本範例掃描按鍵程式亦含有簡易去彈跳的機制，程式設計每 100 ms 才去掃描偵測一次按鍵狀態，要連續兩次偵測結果都是同一個按鍵被按下，才確定這個按鍵備按下了，實驗板腳位連接請按照程式中的 define 來接，並將 P0 接到 LED 模組。

```
#include <REGtenxTM52F5288.H>

#define R1 P2_0      //定義掃描按鍵腳位
#define R2 P2_1
#define R3 P2_2
#define R4 P2_3
#define C1 P2_4
#define C2 P2_5
#define C3 P2_6
#define C4 P2_7

unsigned char Scankey (void);    //宣告掃描按鍵副程式

unsigned char GET_KEY;          //取得的按鍵
unsigned char GET_KEY_BUF;      //去彈跳程序的按鍵值備份
bit          bf_100ms;

void main (void){
    //Port 0
    P0OE= 0xFF;
    P0= 0xFF;

    //Port 2
    P2OE= 0xF0;
    P2= 0xFF;

    //CLKCON= 0x23;          //預設值
    CLKCON= 0x20;           //0010 0000    改變除頻變除 16
    CLKCON= 0x24;           //0010 0100    改快鐘

    //timer2 設定
    T2CON= 0x04;            //Timer2 開始計數
    TH2= 0xA6;
    TL2= 0x00;
    RCP2H= 0xA6;            //快鐘除以 16，reload 設 0xA600 等於 100ms 中斷一次
    RCP2L= 0x00;
    ET2= 1;
    EA= 1;

    GET_KEY_BUF= 0;
    bf_100ms= 0;
}
```

(下頁續....)



```
//主程序循環
while (1)
{
if (bf_100ms==1){           //以下程式 100ms 執行一次
bf_100ms=0;
GET_KEY=Scankey();        //呼叫掃描按鍵副程式，會返回按鍵值，0 就是沒按
if (GET_KEY==GET_KEY_BUF) //如果上次收到的按鍵值跟這次一樣，
//表示確認按鍵被按了
{
P0= ~GET_KEY;             //把按鍵值從 P0 送出去，P0 可接到 LED 模組觀察按鍵結果
}
else
{
GET_KEY_BUF= GET_KEY;    //如果本次讀到的按鍵值跟上次不一樣，
//那就把本次的新按鍵值備份起來
}
}
}
}

unsigned char Scankey (void){ //掃描按鍵副程式
//*****掃 C1 行*****
C1=0;
if (R1==0){
C1= 1;           //C1R1 被按了
return 4;
}
else if (R2==0){
C1=1;           //C1R2 被按了
return 8;
}
else if (R3==0){
C1= 1;           //C1R3 被按了
return 12;
}
else if (R4==0){
C1= 1;           //C1R4 被按了
return 16;
}
else{
//都沒按
C1= 1;
}

//*****換掃 C2 行*****
C2=0;
if (R1==0){
C2= 1;           //C2R1 被按了
return 3;
}
else if (R2==0){
C2= 1;           //C2R2 被按了
return 7;
}
else if (R3==0){
C2= 1;           //C2R3 被按了
return 11;
}
else if (R4==0){
C2= 1;           //C2R4 被按了
return 15;
}
else{
//都沒按
C2= 1;
}
}
(下頁續....)
```



```

//*****換掃 C3 行*****
C3= 0;
If (R1==0){
    C3= 1;          //C3R1 被按了
    return 2;
}
else if (R2==0){
    C3= 1;          //C3R2 被按了
    return 6;
}
else if (R3==0){
    C3= 1;          //C3R3 被按了
    return 10;
}
else if (R4==0){
    C3= 1;          //C3R4 被按了
    return 14;
}
else{
    C3= 1;
    //都沒按
}

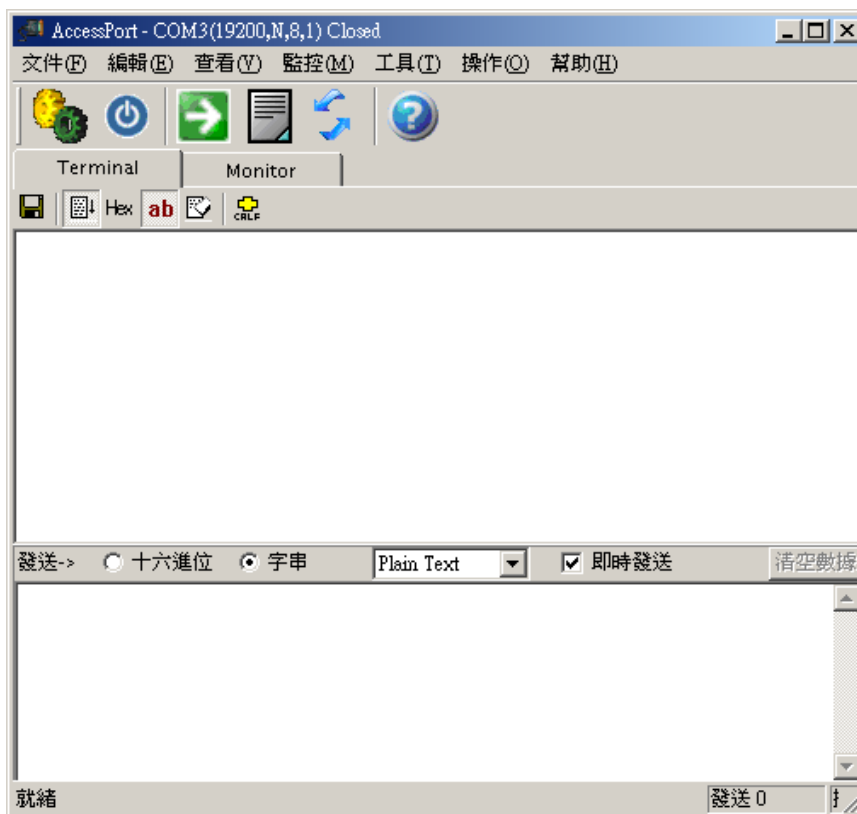
//*****換掃 C4 行*****
C4= 0;
if (R1==0){
    C4= 1;          //C4R1 被按了
    return 1;
}
else if (R2==0){
    C4= 1;          //C4R2 被按了
    return 5;
}
else if (R3==0){
    C4= 1;          //C4R3 被按了
    return 9;
}
else if (R4==0){
    C4= 1;          //C4R5 被按了
    return 13;
}
else{
    //都沒按
    C4= 1;
}
return 0;          //都沒按 return 一個 0
}

//Timer2 中斷副程式
void TIMER2_int (void) interrupt 5{
    TF2= 0;
    bf_100ms= 1;
}

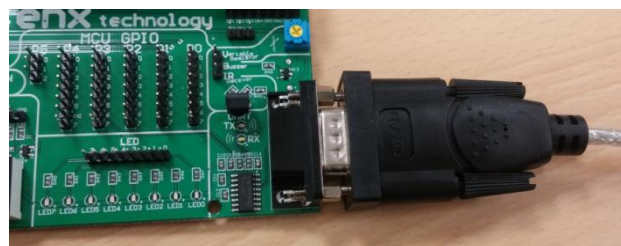
```

3.6 讓 PC 跟實驗板利用 UART (RS232) 通訊 (UART 應用)

本範例做一個簡易的 UART 傳輸程式，讓實驗板可以跟 PC，或兩片實驗板之間互傳資料，PC 端可以安裝超級終端機或 AccessPort 這類可以收發 UART (RS232) 的程式，這邊以 AccessPort 這一個程式為例，在 google 打 AccessPort 免費軟體就可以下載的到，[或是按這裡](#)，下載好之後打開軟體，軟體長的像下圖這樣

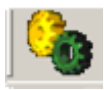


我們先把範例 Code 燒到板子上，然後用 COM Port 的線把實驗板跟電腦連起來！



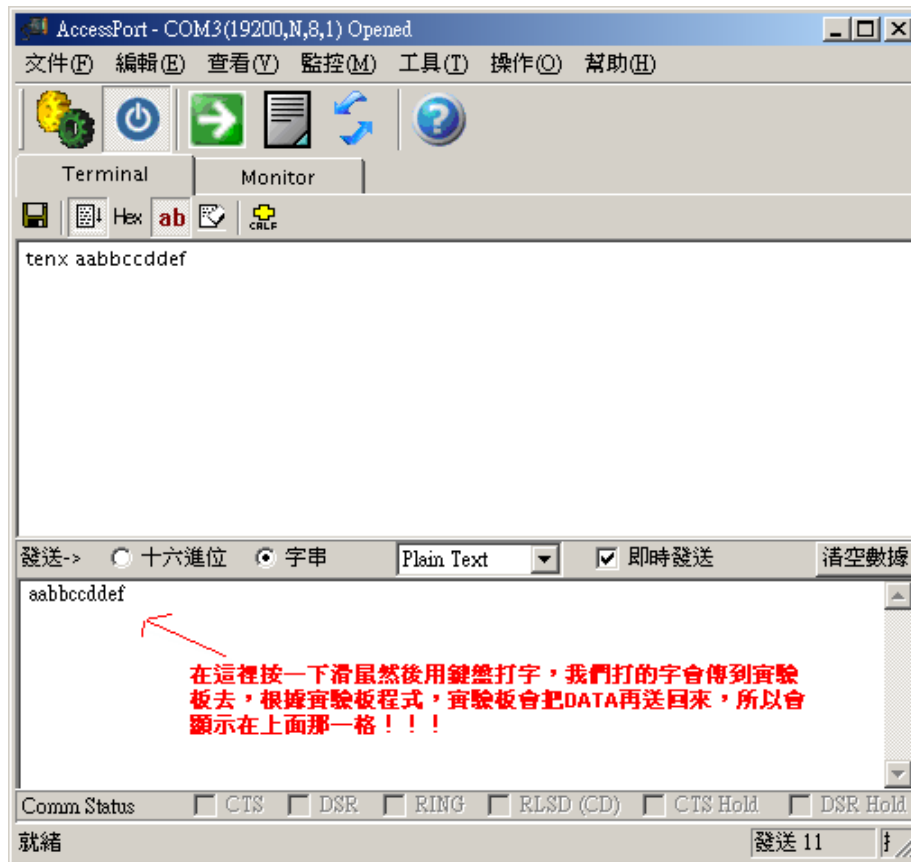
然後到我的電腦中的**裝置管理員**找看看板子是哪一個 **Com** 的 **Port**，本範例是 COM3！可以一直插拔板子，看看哪個 COM 會出現又消失就是那一個啦！！



知道 COM 的編號後，回到 AccessPort 軟體按左上角的齒輪  出現設定的畫面，按照下面設定（COM 的串口編號就用剛剛你們自己的），傳輸速我們範例 Code 是設定為 19200，NONE 校驗位，數據 8 位元，停止位 1...設定完按確定



按完確定回到軟體畫面，把實驗板電源線或 USB 拔掉，重新插上去，這時後軟體就會跑出在實驗板程式裡面寫要送出的那串字‘tenx’



實驗板程式中 UART 的設定及運作模式，在 datasheet 的“9. UART”中有說明，範例程式中亦有註解說明設定內容，比較複雜的鮑率計算，我們範例程是跑內部快鐘，除頻除以 2，所以是 3.6864 MHz，UART 是設定跑 MODE 1（8 bit UART, Baud Rate is variable），用 Timer1 reload 值是 250，SMOD 設 1，所以依據下面鮑率的公式，算出結果

Mode 1, 3: if using Timer1 auto reload mode

$$\text{Baud Rate} = (\text{SMOD} + 1) \times \text{F}_{\text{SYSCLK}} / (32 \times 2 \times (256 - \text{TH1}))$$

計算 $(1+1) * 3686400 / (32*2*(256-250)) = 19200$ 所以電腦端串口速度才會設定 19200

```
#include <REGtenxTM52F5288.H>
unsigned char  Get_DATA;    //存放收到的 DATA
void main (void){

    //I/O 設定
    P3MODL= 0x80;          // P3_1= TX ; P3_0= RX
    P3= 0xFF;

    //時脈設定
    CLKCON= 0x22;         //除 2
    CLKCON= 0x26;         //切快鐘

    //UART 設定
    SCON= 0x50;           //UART 設 MODE 1
    PCON= 0x80;           //開啟 UART

    //利用 Timer1 產生 UART 需要的鮑率
    TMOD= 0x20;           //Timer1 設為 8bits 自動 reload 計數器，當 UART 的鮑率
    TH1= 250;             //reload 值
    TR1= 1;               //timer 1 run

    SBUF= 't';
    while (TI==0);        //等待傳送完成 TI 會變 1
    TI= 0;                 //傳送完要把 TI 清 0

    SBUF= 'e';
    while (TI==0);        //等待傳送完成 TI 會變 1
    TI= 0;                 //傳送完要把 TI 清 0

    SBUF= 'n';
    while (TI==0);        //等待傳送完成 TI 會變 1
    TI= 0;                 //傳送完要把 TI 清 0

    SBUF= 'x';
    while (TI==0);        //等待傳送完成 TI 會變 1
    TI= 0;                 //傳送完要把 TI 清 0
    while (1){

        if (RI==1){
            Get_DATA= SBUF;    //RI 變 1 就是收到 PC 資料了，把值放到 Get_DATA
            RI= 0;             //取回值後把 RI 清為 0
            SBUF= Get_DATA;    //再把 Get_DATA 丟到 SBUF 送回去給 PC，
                               //這樣 PC 螢幕才會顯示剛剛鍵盤按的字
            while (TI==0) ;
            TI= 0;
        }
    }
}
```

3.7 可變電阻模組類比數位轉換 (ADC 應用)

本範例程式利用 TM52F5288 的 ADC，從 MCU 外部讀進可變電阻模組送出的 0V~5V 間之類比電壓值，當電壓約 0V~1V 時亮一顆 LED 燈、約 1V~2V 時亮二顆 LED 燈、約 2V~3V 時亮三顆 LED 燈、約 3V~4V 時亮四顆 LED 燈、約 4V~5V 時亮五顆 LED 燈！

TM52F5288 裡是一 12Bits 的 ADC，所以 ADC 可將 0V~5V 的電壓切成 4096 個階，轉換完成的數值高 8Bits 放在 ADCDH，低 4Bits 放在 ADCDL 這兩個暫存器中，按照此範例的程式需求只需將數值分成 5 個階段，所以可以只讀 ADCDH 這個暫存器就夠了！也就是設定當數值是 0~50 亮 1 顆燈、51~100 亮 2 顆燈、101~150 亮 3 顆燈、151~200 亮 4 顆燈、201~255 亮 5 顆燈。

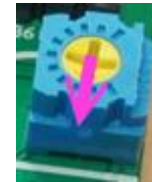
要控制 ADC 只需要設定好 ADC 的運作頻率、設定好 ADC Channel (TM52F5288 有 12 Channel) 然後將 ADSOC 這個 Bit 設 1，ADC 就會開始運作，當運作完成吐出值之後，ADCSOC 會自動被寫回 0，所以當我們把 ADSOC 設 1 讓 ADC 運作之後，我們只需要去檢查 ADSOC 是不是變 0 就知道 ADC 運作完成了沒！本範例使用 ADC Channel 0 也就是 P1_7 這支腳，所以只需要將可變電



阻模組的 PIN 用杜邦線接到 GPIO 的 P1_7 就可以了，然後再將 GPIO P2_0~P2_4 接到 LED 模組的 LED0~LED4 就可以跑本範例程式了！

```
#include <REGtenxTM52F5288.H>
#define LED P2
void main (void)
{
    //時脈改除頻除以 2，然後切到快鐘
    CLKCON= 0x22;
    CLKCON= 0x26;
    //IO 設定
    P1MODH= 0xC0; //P1.7 要當 ADCPIN...要設成 Mode3
    P2OE= 0xFF; //P2 當 LED 訊號腳位
    //ADC 設定
    OPTION= 0x00; //設定 ADC 運作速度
    CHSEL= 0x00;
    while (1)
    {
        ADSOC= 1; //ADC 開始運作
        while (ADSOC==1); //等待 ADC 完成 (ADSOC 完成會變 0)
        if (ADCDH<50){ //判斷 ADC 數值在哪个區間
            LED= ~0x01; //亮一顆
        }
        else if (ADCDH<100){
            LED= ~0x03; //亮二顆
        }
        else if (ADCDH<150){
            LED= ~0x07; //亮三顆
        }
        else if (ADCDH<200){
            LED= ~0x0F; //亮四顆
        }
        else{
            LED= ~0x1F; //亮五顆
        }
    }
}
```

程式運作結果如下圖，可用十字起子轉動可變電阻！電壓會從 0V~5V 變化~



3.8 1602 LCM 文字型液晶模組

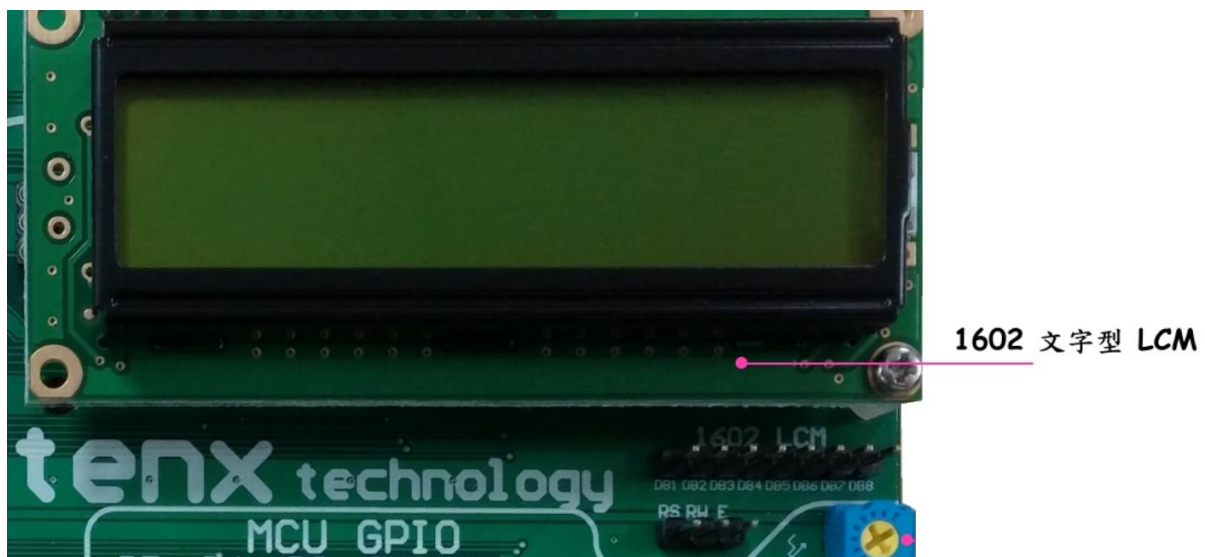
本範例實作一個文字型 LCM 控制程式，該顯示器可顯示兩列英文字串。此模組上的 HD44780 是日立 (Hitachi) 公司開發的一顆標準文字型 LCD 控制 IC，該 IC 提供內建的 ASCII 字型，放在其字元產生器 (Character Generator ROM；CG ROM) 唯讀記憶體內，使用時只要將 ASCII Code 寫入 LCD 模組上之控制 IC 的顯示資料記憶體 (Display Data RAM；DD RAM) 內，該 IC 便會將字型顯示在 LCD 上。此外該 IC 也提供使用者定義幾個自己的字型放在 CG RAM 中。HD44780 IC 最多可顯示兩列，一列最多 40 個字共 80 個字，LCD 上每個位置與 DD RAM 記憶體位址對映關係如下表所示：

顯示位置	1	2	3	~	38	39	40
第 1 列	00H	01H	02H	~	25H	26H	27H
第 2 列	40H	41H	42H	~	65H	66H	67H

本實驗板之 LCM 模組已將控制腳位拉出，各腳位說明如下表

腳位	腳位說明
R/ \bar{W}	0：資料寫入 LCM，1：讀取 LCM 資料
EN	0：啟用讀寫 LCM 功能，1：關閉讀寫 LCM 功能
RS	0：選擇指令暫存器，1：選擇資料暫存器
DB0~DB7	資料匯流排

下圖為實驗板上 LCM 模組腳位在模組的右下角，可與上表對照觀察～



LCM 模組要跟 TM52F5288 溝通 HD44780 IC 內部提供兩個 8 位元暫存器，兩個暫存器分別是指令暫存器 (Instruction Register ; IR) 與資料暫存器 (Data Register ; DR)。指令暫存器用來儲存 TM52F5288 送來的命令，並進行各種相關設定，資料暫存器則用來儲存要顯示的 ASCII Code 或字型資料，並將資料送到適當的記憶體位址。

HD44780 IC 可接受的命令共有 11 個分別說明如下，只要由 MCU 傳送以下命令就可以讓 LCM 做出相對應的設定或動作，傳送命令給 HD44780 IC 時 EN 腳位必須設定為 1，命令傳送完成 EN 腳位必須再恢復為 0。

- **清除顯示內容 (Clear Display) :** 將 DD RAM 的內容全部清為空白 (ASCII Code = 20h)，游標返回螢幕左上角 (第 1 列第 1 個位置)，位址計數器 (Address Counter ; AC) 歸零，其中，AC 是當有資料要寫入 DD RAM 時，用來指定資料要寫入的位置，資料寫入 DD RAM 後，AC 值會自動調整。清除顯示內容命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

- **返回左上角 (Return Home) :** DD RAM 內容不變，游標返回左上角，AC 歸零。返回左上角命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	x

- **設定輸入模式 (Entry Mode Set) :** 設定讀寫 DD RAM 時，顯示內容、游標、與 AC 值的改變方式。設定輸入模式命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

其中，I/D 及 S 設定之內容如下表

I/D	S	作用
0	0	顯示內容不動，游標左移，AC 減 1
1	0	顯示內容不動，游標右移，AC 加 1
0	1	顯示內容右移，游標不變，AC 不變
1	1	顯示內容左移，游標不變，AC 不變

- **顯示器開關控制 (Display on/off control) :** 控制顯示器 (D)，游標 (C)，與游標字元閃爍 (B) 功能的開關。顯示器開關控制命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

功能開關說明如下：

D	0：關閉顯示器，1：開啟顯示器
C	0：不顯示游標，1：顯示游標
B	0：字元不閃爍，1：字元閃爍

- **游標或顯示內容位移 (Cursor or display shift)：**在不改變 DD RAM 內容情況下，移動游標或顯示內容。游標或顯示內容位移命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

S/C 與 R/L 用途說明如下：

S/C	R/L	作用
0	0	游標左移，AC 減 1
0	1	游標右移，AC 加 1
1	0	顯示內容左移
1	1	顯示內容右移

- **功能設定 (Function Set)：**設定資料匯流排寬度 (DL)，顯示列數 (N)，或字型 (F)。功能設定命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

DL、N、與 F 用途說明如下：

DL	0：4 bits，只用 DB7~DB4 1：8 bits
N	0：1 列 1：2 列
F	0：字型寬×高=5×8 1：字型寬×高=5×10

- **設定 CG RAM 位址 (Set CG RAM address)：**設定接下來要讀寫的 CG RAM 位址，位址線有 6 個位元 (A5~A0)，可定址範圍 00H~3FH。設定 CG RAM 位址命令如下：

RS	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A5	A4	A3	A2	A1	A0

- **設定 DD RAM 位址 (Set DD RAM address) :** 設定接下來要讀寫的 DD RAM 位址，位址線有 7 個位元 (A5~A0)，可定址範圍 00H~7FH。設定 DD RAM 位址命令如下：

RS	$\overline{R/\overline{W}}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A6	A5	A4	A3	A2	A1	A0

- **讀取忙碌旗標與位址 (Read busy flag and address) :** 讀取忙碌旗標 (BF) 與 DD RAM 位址 (AC 的值)。讀取忙碌旗標與位址命令如下：

RS	$\overline{R/\overline{W}}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	A6	A5	A4	A3	A2	A1	A0

忙碌旗標說明如下：

BF	0：可接受外部命令 1：忙碌中
----	--------------------

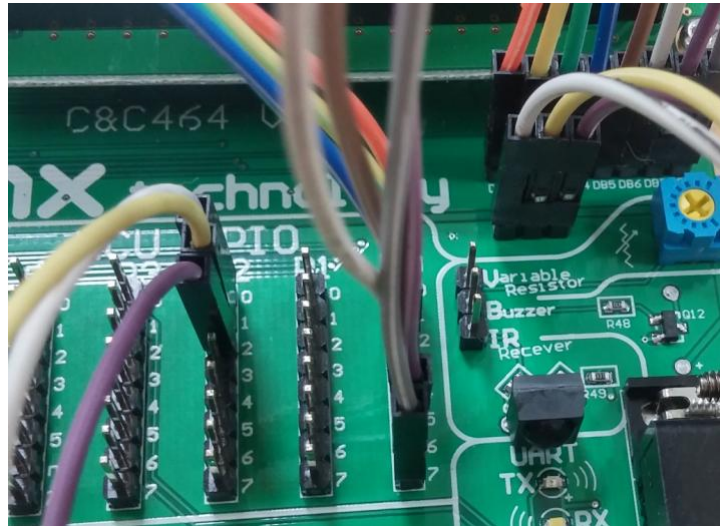
- **寫入 CG 或 DD RAM (Write data to CG or DD RAM) :** 若最近曾設定 CG RAM 位址，則資料會寫入 CG RAM，最近曾設定 DD RAM 位址，則資料會寫入 DD RAM。寫入 CG 或 DD RAM 命令如下：

RS	$\overline{R/\overline{W}}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

- **讀取 CG 或 DD RAM (Read data from CG or DD RAM) :** 若最近曾設定 CG RAM 位址，則會讀取 CG RAM 資料，最近曾設定 DD RAM 位址，則會讀取 DD RAM 資料。讀取 CG 或 DD RAM 命令如下：

RS	$\overline{R/\overline{W}}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

本範例程式要在 LCM 上顯示兩行文字然後就停止程式，要控制 LCM 首先需等候 LCM 完成開機動作，通常 LCM 模組的 Datasheet 都會寫 LCM 上電後需要延遲多少時間，然後才開始對 LCM 進行初始化設定的動作，完成初始化後接著將兩個預設的字串分別寫入 LCD 的第一列與第二列及完成。依照範例程式定義 LCM 模組的 DB1~DB8 接到 P0_0~P0_7，RS 接 P2_0，RW 接 P2_1，EN 接 P2_2。



首先我們先建立 LCM 傳送控制碼以及傳送 DATA 的兩個副程式！

下面是傳送 Command 的副程式

```
void LCM_Write_CTRL (unsigned char CMD)
{
    LCM_RS= 0;           //設定成寫指令 0: 指令 1: 資料
    LCM_RW= 0;           //設定讀寫      0: 寫入 1: 讀取

    LCM_DATA_BUS= CMD;   //把 Command 送到 LCM_DATA_BUS

    LCM_EN= 1;           //LCM 開始寫入
    Delay (100);         //延遲一段時間
    LCM_EN= 0;           //LCM 寫入結束
}
```

下面是傳送 DATA 的副程式

```
void LCM_Write_DATA (unsigned char DATA)
{
    LCM_RS= 1;           //設定成寫資料 0: 指令 1: 資料
    LCM_RW= 0;           //設定讀寫      0: 寫入 1: 讀取

    LCM_DATA_BUS= DATA; //把 DATA 送到 LCM_DATA_BUS

    LCM_EN= 1;           //LCM 開始寫入
    Delay (100);         //延遲一段時間
    LCM_EN= 0;           //LCM 寫入結束
}
```

LCM 初始化設定的副程式

```
void LCM_initial (void)
{
    int i;
    for (i= 0;i<100;i++)           //延遲一段時間
    {
        Delay (100);
    }
    LCM_Write_CTRL (0x30);        //用送 Command 的副程式送出 0x30，datasheet 建議送 3 次
    LCM_Write_CTRL (0x30);        //確保 LCM 有正確設定 (0x30 是什麼請看這裡)
    LCM_Write_CTRL (0x30);

    for (i= 0; i<100; i++)        //延遲一段時間
    {
        Delay (100);
    }

    LCM_Write_CTRL (0x38);        //用送 Command 的副程式送出 0x38 (0x38 是什麼請看這裡)
    Delay (100);
    LCM_Write_CTRL (0x08);        //用送 Command 的副程式送出 0x08 (0x08 是什麼請看這裡)
    Delay (100);
    LCM_Write_CTRL (0x01);        //用送 Command 的副程式送出 0x01 (0x01 是什麼請看這裡)
    Delay (100);
    LCM_Write_CTRL (0x06);        //用送 Command 的副程式送出 0x06 (0x06 是什麼請看這裡)
    Delay (100);
    LCM_Write_CTRL (0x0E);        //用送 Command 的副程式送出 0x0E (0x0E 是什麼請看這裡)
    Delay (100);
}
}
```

建立好以上三個副程式後，就可以開始輕鬆的控制 LCM 了，其餘程式如下

```
#include <REGtenxTM52F5288.H>
#define LCM_DATA_BUS P0
#define LCM_RS      P2_0
#define LCM_RW      P2_1
#define LCM_EN      P2_2

void LCM_initial (void);
void LCM_Write_CTRL (unsigned char CMD);
void LCM_Write_DATA (unsigned char DATA);
void Delay (int count);

void main (void)
{
    //時脈改除頻除以 2，然後切到快鐘
    //CLKCON= 0x23;
    CLKCON= 0x22;
    CLKCON= 0x26;

    //IO 設定
    P0OE= 0xFF;
    P2OE= 0x07;

    LCM_initial();           //呼叫 LCM 初始設定副程式，啟動 LCM

    LCM_Write_CTRL (0x80);   //設定顯示位址(0x80 從哪裡來看這裡)
    LCM_Write_DATA ('H');   //丟資料到 LCM，LCM 會自動顯示並遞增顯示位置
    LCM_Write_DATA ('e');
    LCM_Write_DATA ('l');
    LCM_Write_DATA ('l');
    LCM_Write_DATA ('o');
    LCM_Write_DATA ('!');
    LCM_Write_DATA (' ');

    LCM_Write_CTRL (0xC0);  //設定顯示位址(0xC0 從哪裡來看這裡)
    LCM_Write_DATA (' ');  //丟資料到 LCM，LCM 會自動顯示並遞增顯示位置
    LCM_Write_DATA (' ');
    LCM_Write_DATA (' ');
    LCM_Write_DATA ('t');
    LCM_Write_DATA ('e');
    LCM_Write_DATA ('n');
    LCM_Write_DATA ('x');
    While (1);             //停
}

void Delay (int count) //延遲副程式
{
    unsigned char i;
    int j;
    for (j= 0; j<count; j++){
        for (i= 0; i<10; i++);
    }
}
```

程式執行結果如下圖～



程式中可發現很多指令送出後，都會插一個 Delay 副程式延遲一段時間！這延遲時間的長短，不同廠商的 LCM 都會一些不一樣，下列提供本實驗板之 LCM 指令延遲參數以供參考。

Instruction	Instruction Code										Description	ExecutionTime(fosc=270kHz)	
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM set DDRAM address to "00H" from AC	1.52ms	
Return Home	0	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed	1.52ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display	38 μ s	
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D) cursor(C) and blinking of cursor(B) on/off	38 μ s	
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data	38 μ s	
Function Set	0	0	0	0	1	DL	N	F	-	-	Set interface data length of display line (N: 2line/1line)and, display font type F:5X11dots/5X8dots	38 μ s	
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	38 μ s	
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	38 μ s	
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF The contents of address counter of address counter can also be read	0 μ s	
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM)	38 μ s	
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM)	38 μ s	

附錄

GPIO 連接板

在各範例裡需要將程式中有使用到的腳位，用杜邦線拉到各實驗模組的電路去！需要另購很多杜邦線，好處是可以很靈活的調配 GPIO 設定；如果使用本連接板就只能依照板上畫好的腳位來使用 GPIO，但優點就是可以省去插很多條杜邦線的步驟，在此連接板上亦已標示各模組的 IO 是連接到 GPIO 的哪一個腳位，寫 Code 時只要依照板上標示的腳位設定即可。



插上實驗板示意圖

