

# TM57FLA80

單晶片

教學實驗板實驗手冊

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

<b>Version</b>	<b>Date</b>	<b>Description</b>
V1.0	Aug, 2011	New release
V1.1	Oct, 2011	1. Add Section 2 about instruction set usage. 2. Add Section 3 about the emulator usage. 3. Add experiment board module and I/O Port descriptions.
V1.2	Mar, 2012	1. Add TICE99 figure. 2. Modify the word "ICE59" to "TICE99" 3. Add TM57 experimental board figure 4. Modify the document title to " TM57FLA80 Microcontroller Experiment Guide Application Note"

# CONTENTS

<b>AMENDMENT HISTORY</b> .....	<b>2</b>
<b>1. 簡介</b> .....	<b>5</b>
1-1 韌體電路設計的演進.....	5
1-2 單晶片微處理器.....	5
1-3 十速 TM57FLA80 晶片.....	6
<b>2. 指令介紹與程式書寫方式</b> .....	<b>7</b>
2-1 指令集說明.....	7
2-1-1 F-Plane 位元組資料運算指令.....	8
2-1-2 F-Plane 位元資料運算指令.....	16
2-1-3 R-Plane 資料運算指令.....	18
2-1-4 常數運算指令.....	19
2-1-5 分支指令.....	21
2-1-6 其他指令.....	23
2-1-7 虛指令.....	24
2-2 指令集說明表.....	25
2-3 程式的編輯.....	27
2-4 程式書寫的方式.....	29
<b>3. 模擬器使用說明</b> .....	<b>32</b>
3-1 安裝 TICE99.....	32
3-2 模擬器介面之介紹.....	36
3-3 新增與開啟專案.....	37
3-4 執行與除錯.....	42
3-5 主選單與工具列之功能介紹.....	45
<b>4. 紅綠燈實習</b> .....	<b>51</b>
<b>5. 指撥開關與七段顯示器實習</b> .....	<b>58</b>
<b>6. 計時器實習</b> .....	<b>68</b>
<b>7. 數位電子鐘實習</b> .....	<b>77</b>
<b>8. 4 × 4 鍵盤實習</b> .....	<b>85</b>
<b>9. 數位電子琴實習</b> .....	<b>97</b>
<b>10. 數位電錶類比轉數位實習</b> .....	<b>111</b>
<b>11. 脈波調變控制之電風扇實習</b> .....	<b>128</b>
<b>12. 文字型 LCD 顯示器實習</b> .....	<b>137</b>
<b>13. UART 與 RS232 傳輸實習</b> .....	<b>150</b>

---

14. SPI 串列週邊介面實習 .....	171
15. 密碼鎖實習 .....	191
附錄 1 TM57 教學實驗板使用說明 .....	207
TM57 教學實驗板照片 .....	208
附錄 1-1 TM57 教學實驗板簡介 .....	209
附錄 1-2 模組板使用方式 .....	211
附錄 1-3 模組版電路圖 .....	212
附錄 2 配合模組板使用的程式修改方式.....	225

# 1. 簡介

## 1-1 韌體電路設計的演進

電腦硬體設計通常可分為純硬體電路(hardware)及韌體電路(firmware)兩種方式，但由於純硬體設計通常需要自行設計時序(Timing)及控制單元(Control Unit; CU)，在設計上較為費時與不易，因此有人把較常使用的微運算和時序及控制單元與記憶體及輸入輸出做在一顆晶片上，而成為初期的單晶片微電腦系統。

隨著積體電路設計技術的進步，一顆晶片內能建造更多的電路，於是現今的單晶片微處理器紛紛將許多常用的功能，如計時器、類比數位轉換、非同步串列傳輸裝置、串列週邊介面、...等整合進單一顆晶片內，使得單晶片微處理器的功能大大的提升，所能建造的系統也更加的多樣化。

雖然嵌入式系統及個人電腦的普及率與功能性也一直在提升，但由於單晶片微處理器尺寸小、成本低，且多數應用不需用到大量的運算與儲存空間，因此單晶片微處理器仍是整個微電腦應用市場中，用量最大的微處理器產品，也是值得大家深入了解的產品。

## 1-2 單晶片微處理器

微處理器架構一般可分為三大部分：中央處理單元(含算術邏輯單元與控制單元)、記憶單元、以及輸入/輸出單元，其架構如圖 1-1 所示：

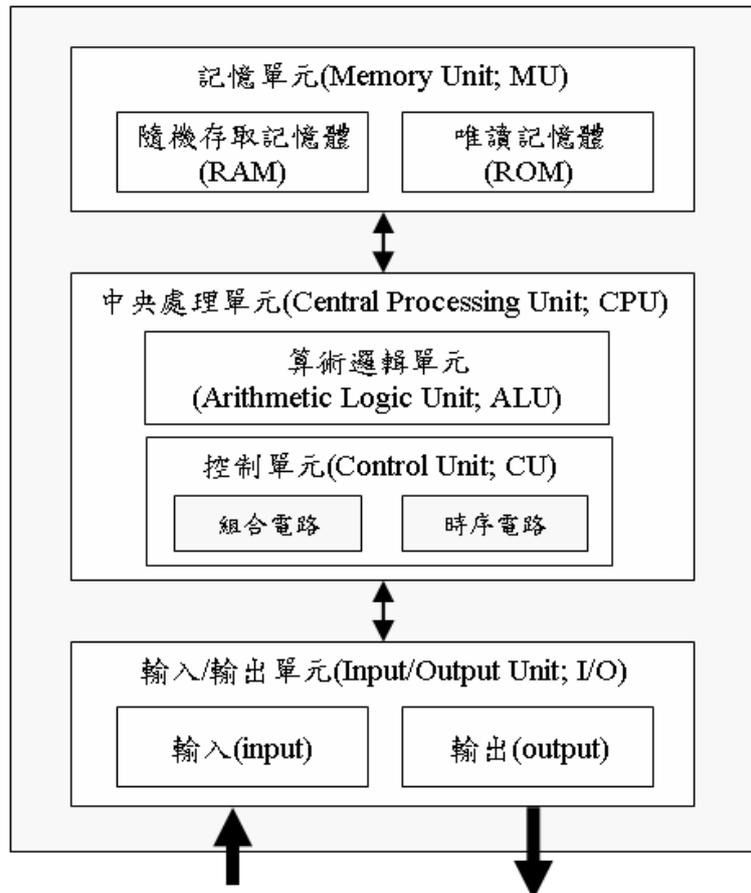


圖 1-1 微處理器架構圖

在微處理器的架構中，中央處理單元是微處理器的心臟，控制著整個微處理器的運作，運作方式是由記憶體中取出指令，再由控制單元的解碼器辨別指令後，啟動相對的運算及控制，完成單一指令的操作，透過完成一連串相關的指令可完成特定目的工作；記憶單元包括唯讀記憶體(Read Only Memory;ROM)與隨機存取記憶體(Random Access Memory;RAM)等兩種，唯讀記憶體通常用來存放控制微處理器動作的程式碼，隨機存取記憶體則用來存放程式執行中需要使用到的臨時變數；輸入/輸出單元(Input/Output; I/O)提供外部與微處理器之間的溝通橋樑，微處理器可透過輸入單元了解外部情況，例如了解外界的溫度、濕度、或按鍵狀態等，微處理器工作過程中或工作處理完畢，則可控制周邊設備採取適當動作，或將結果透過輸出單元傳遞給（例如顯示出來或發出聲音）使用者知道。

為方便使用者能簡單有效的使用微電腦系統，將中央處理單元、記憶體、輸入/輸出、及常用的輸入/輸出介面放入單一顆晶片內，便構成單晶片微處理器。

### 1-3 十速 TM57FLA80 晶片

十速科技成立於 1997 年元月，主要從事消費性微處理器及多媒體應用 IC 產品開發，初期以 4 位元的小型微處理器為主，並持續不斷研發許多新產品。

TM57FLA80 單晶片是十速公司 2010 年推出的微處理器，該處理器整合了許多常用的輸入/輸出介面，例如類比數位轉換(Analog to Digital Converter; ADC)、脈波調變控制功能(Pulse Width Modulation; PWM)、LCD 控制、串列傳輸介面(Universal Asynchronous Receiver/Transmitter; UART)、與串列週邊介面 (Serial Peripheral Interface; SPI) 等，使用 TM57FLA80 單晶片，可大幅度降低系統開發時程，成為單晶片市場一顆閃亮的新星。

TM57FLA80 單晶片功能完善，是極具潛力的一顆晶片，主要功能如下：

- 8 位元的微處理器
- 有 F-plane 與 R-plane 兩塊資料記憶體：F-plane 有 176 Bytes，R-plane 有 192 Bytes
- 8k\*14bits 的內部程式記憶體(ROM)
- 指令集有 37 種指令(instructions)
- 8-level 的堆疊
- 9 個中斷向量 Timer0、Timer1、Timer2、PWM0、WKT、XINTA、XINTB、UART、SPI
- 45 隻 I/O 腳位(pins)：分為 6 組 I/O 埠(port)，PA0~PA6、PB0~PB7、PD0~PD7、PE0~PE7、PF0~PF7、PG0~PG5
- 3 個計時器(Timer)：Timer 0 為 8 位元計時計數器且有 8 位元的預除裝置；Timer 1 為 16 位元計時器；Timer 2 為 15 位元計時器，提供 LCD 模組的時脈
- 2 個脈波寬度調變(PWM)裝置：PWM0、PWM1
- 1 個看門狗(WDT)裝置
- 16 個觸控式輸入腳
- 1 個 LCD 控制器
- 6 通道的類比數位轉換器(ADC)，解析度 12 位元
- 1 個全雙工的通用非同步串列傳輸裝置(UART)，傳輸率由 1200bps 到 38400bps
- 1 個全雙工的串列傳輸介面(SPI)

為了快速了解 TM57FLA80 單晶片的用法，以下各章分別提供不同的實習內容，介紹 TM57FLA80 單晶片所提供各種功能的操作方式。

## 2. 指令介紹與程式書寫方式

本章介紹指令的用法以及程式的撰寫方式。

### 2-1 指令集說明

指令中常用的變數說明表

符號	意義
f	F-Plane 的記憶體位址暫存器
r	R-Plane 的記憶體位址暫存器
b	位元位址
k	常數，或標記
d	儲存目的暫存器之選擇位元 0: 工作暫存器 W ; 1: 原暫存器
W	工作暫存器
Z	零旗標
C	進位旗標
DC	輔助進位旗標
PC	程式計數器
TOS	堆疊頂端
GIE	整體中斷致能旗標
.	位元區
←	指定方向

## 2-1-1 F-Plane 位元組資料運算指令

**ADDWF**


---

指令:	ADDWF f[,d]	
運算元:	f: 00h ~ 7fh, d: 0, 1	
運算:	(目的暫存器) ← W + f	
影響旗標:	C, DC, Z	
運算碼:	00 0111 dfff ffff	
定義:	若是 d = 0, 則目的暫存器為 W, W = W + f 若是 d = 1, 則目的暫存器為 f, f = W + f	
工作週期:	1	
範例:	ADDWF FR, 0	指令執行前 W=21h, FR=10h 指令執行後 W=31h, FR=10h
	ADDWF FR, 1	指令執行前 W=21h, FR=10h 指令執行後 W=21h, FR=31h

**ANDWF**


---

指令:	ANDWF f[,d]	
運算元:	f: 00h ~ 7fh, d: 0, 1	
運算:	(目的暫存器) ← W AND f	
影響旗標:	Z	
運算碼:	00 0101 dfff ffff	
定義:	將 W 和 f 作邏輯 AND 運算 若是 d = 0 則目的暫存器為工作暫存器 W, W = W AND f 若是 d = 1 則目的暫存器為 f, f = W AND f	
工作週期:	1	
範例:	ADDWF FR, 0	指令執行前 W=3ch, FR=17h 指令執行後 W=14h, FR=17h
	ADDWF FR, 1	指令執行前 W=3ch, FR=17h 指令執行後 W=3ch, FR=14h

**CLRF**


---

指令:	CLRF f	
運算元:	f : 00h ~ 7fh	
運算:	(目的暫存器) ← 00h , Z ← 1	
影響旗標:	Z	
運算碼:	00 0001 1fff ffff	
定義:	清除 f 記憶體內容為 0 , 並將零位旗標設為 1	
工作週期:	1	
範例:	CLRF FR	指令執行前 FR=5ah 指令執行後 FR=00h , Z=1

**CLRW**


---

指令:	CLRW	
運算元:	-	
運算:	(W) ← 00h , Z ← 1	
影響旗標:	Z	
運算碼:	00 0001 0100 0000	
定義:	清除工作暫存器 W 內容為 0 , 並將零位旗標設為 1	
工作週期:	1	
範例:	CLRW	指令執行前 W=5ah 指令執行後 W=00h , Z=1

**COMF**


---

指令:	COMF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← $\bar{f}$	
影響旗標:	Z	
運算碼:	00 1001 dfff ffff	
定義:	將 f 作反向運算 若是 d = 0 , 則將 $\bar{f}$ 的值放入工作暫存器 W 若是 d = 1 , 則將 $\bar{f}$ 的值放進 f 中	
工作週期:	1	
範例:	COMF FR , 0	指令執行前 FR=13h , W=22h 指令執行後 FR=13h , W=ech
	COMF FR , 1	指令執行前 FR=13h , W=22h 指令執行後 FR=ech , W=22h

**DECF**


---

指令:	DECF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← f - 1	
影響旗標:	Z	
運算碼:	00 0011 dfff ffff	
定義:	若是 d = 0 , 則將 f-1 的值放入工作暫存器 W 若是 d = 1 , 則將 f-1 的值放進 f 中	
工作週期:	1	
範例:	DECF FR , 0	指令執行前 FR=05h , W=22h 指令執行後 FR=05h , W=04h
	DECF FR , 1	指令執行前 FR=05h , W=22h 指令執行後 FR=04h , W=22h

**DECFSZ**


---

指令:	DECFSZ f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← f - 1 若是 f - 1 的值為 0 則跳過下一個指令, 執行下一個指令	
影響旗標:	-	
運算碼:	00 1011 dfff ffff	
定義:	若 f - 1 的值為 0 , 則跳過下一個指令執行下下一個指令, 指令執行時間為 2 個週期 若 f - 1 的值不為 0 , 則執行下一個指令, 所需時間為 1 個週期 若是 d = 0 , 則將 f - 1 的值放入工作暫存器 W 若是 d = 1 , 則將 f - 1 的值放進原暫存器 f 中	
工作週期:	1 or 2	
範例:	LABEL : DECFSZ FR , 1 GOTO LOOP CONTINUE	指令執行前 PC= LABEL 指令執行後 If FR-1 = 0 , PC = LABEL+2 If FR-1≠0 , PC = LABEL+1

**INCF**


---

指令:	INCF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← f + 1	
影響旗標:	Z	
運算碼:	00 1010 dfff ffff	
定義:	若是 d = 0 , 則將 f + 1 的值放入工作暫存器 W 若是 d = 1 , 則將 f + 1 的值放進原暫存器 f 中	
工作週期:	1	
範例:	INCF FR , 0	指令執行前 FR=05h , W=22h 指令執行後 FR=05h , W=06h
	INCF FR , 1	指令執行前 FR=05h , W=22h 指令執行後 FR=06h , W=22h

**INCFSZ**


---

指令:	INCFSZ f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← f + 1 , 若是 f + 1 為 0 則跳過下一個指令 , 執行下一個指令	
影響旗標:	-	
運算碼:	00 1111 dfff ffff	
定義:	若 f + 1 的值為 0 , 則跳過下一個指令 , 執行下下一個指令 , 指令執行時間為 2 個週期 若 f + 1 的值不為 0 , 則執行下一個指令 , 指令執行時間為 1 個週期 若是 d = 0 , 則將 f + 1 的值放入工作暫存器 W 若是 d = 1 , 則將 f + 1 的值放進原暫存器 f 中	
工作週期:	1 or 2	
範例:	LABEL : INCFSZ FR, 1 GOTO LOOP CONTINUE	指令執行前 PC= LABEL 指令執行後 If FR+1 = 0 , PC = LABEL+2 If FR+1≠0 , PC = LABEL+1

**IORWF**


---

指令:	IORWF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← W OR f	
影響旗標:	Z	
運算碼:	00 0100 dfff ffff	
定義:	將 W 與 f 作邏輯 OR 運算 若是 d = 0 , 則目的暫存器為工作暫存器 W , W = W OR f 若是 d = 1 , 目的暫存器為 f , f = W OR f	
工作週期:	1	
範例:	IORWF FR , 0	指令執行前 W=3ch , FR=17h 指令執行後 W=3fh , FR=17h
	IORWF FR , 1	指令執行前 W=3ch , FR=17h 指令執行後 W=3ch , FR=3fh

**MOVFW**


---

指令:	MOVFW f	
運算元:	f : 00h ~ 7fh	
運算:	W ← f	
影響旗標:	-	
運算碼:	00 1000 0fff ffff	
定義:	將 f 的值移到 W	
工作週期:	1	
範例:	MOVFW FR	指令執行前 W=? , FR=10h 指令執行後 W=10h , FR=10h

**MOVWF**


---

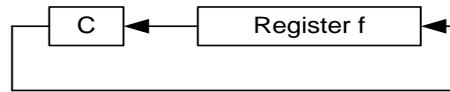
指令:	MOVWF f	
運算元:	f : 00h ~ 7fh	
運算:	f ← W	
影響旗標:	-	
運算碼:	00 0000 1fff ffff	
定義:	將 W 數值移到 f	
工作週期:	1	
範例:	MOVWF FR	指令執行前 W=10h , FR=ffh 指令執行後 W=10h , FR=10h

**RLF**

指令: RLF f [,d]

運算元: f : 00h ~ 7fh , d : 0 , 1

運算:



影響旗標: C

運算碼: 00 1101 dfff ffff

定義: 將 f 內容左移一位, 最左邊位元移入進位旗標 C ,  
並將 C 內容移入 f 最右邊位元

若是 d = 0 , 則將位移結果放入工作暫存器 W

若是 d = 1 , 則將位移結果放入原暫存器 f 中

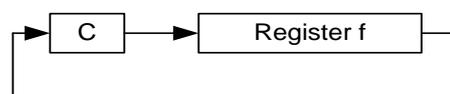
工作週期: 1

範例: RLF FR , 0      指令執行前 FR=9ch , W=22h , C=0  
指令執行後 W =38h , FR=9ch , C=1RLF FR , 1      指令執行前 FR=9ch , W=22h , C=0  
指令執行後 FR=38h , W=22h , C=1**RRF**

指令: RRF f [,d]

運算元: f : 00h ~ 7fh , d : 0 , 1

運算:



影響旗標: C

運算碼: 00 1100 dfff ffff

定義: 將 f 內容右移一位, 最右邊位元移入進位旗標 C ,  
並將 C 內容移入 f 最左邊位元

若是 d = 0 , 則將移位結果放入工作暫存器 W

若是 d = 1 , 則將移位結果放入原暫存器 f 中

工作週期: 1

範例: RRF FR , 0      指令執行前 FR=9ch , W=22h , C=0  
指令執行後 W =4eh , FR=9ch , C=0RRF FR , 1      指令執行前 FR=9ch , W=22h , C=0  
指令執行後 FR=4eh , W=22h , C=0

**SUBWF**


---

指令:	SUBWF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← f - W	
影響旗標:	C , DC , Z	
運算碼:	00 0010 dfff ffff	
定義:	將 f 數值減去 W(使用 2 的補數運算) 若是 d = 0 , 則將 f - W 的值放入工作暫存器 W 若是 d = 1 , 則將 f - W 的值放進原暫存器 f 中	
工作週期:	1	
範例:	SUBWF FR , 1	指令執行前 W=02h , FR=03h 指令執行後 FR=01h , W=02h , C=1, Z=0
	SUBWF FR , 1	指令執行前 W=02h , FR=01h 指令執行後 FR=ffh , C=0 , Z=0

**SWAPF**


---

指令:	SWAPF f [,d]	
運算元:	f : 00h ~ 7fh d : 0 , 1	
運算:	(目的暫存器.7~4) ← (f.3~0) (目的暫存器.3~0) ← (f.7~4)	
影響旗標:	-	
運算碼:	00 1110 dfff ffff	
定義:	將暫存器 f 的(3~0)位元放入目的暫存器的(7~4)位元 將暫存器 f 的(7~4)位元放入目的暫存器的(3~0)位元 若是 d = 0 , 則將高低四位元互換結果放入工作暫存器 W 若是 d = 1 , 則將高低四位元互換結果放入原暫存器 f 中	
工作週期:	1	
範例:	SWAPF FR , 0	指令執行前 FR=17h , W=23h 指令執行後 FR=17h , W=71h
	SWAPF FR , 1	指令執行前 FR=17h , W=23h 指令執行後 FR=71h , W=23h

**TESTZ**


---

指令:	TESTZ f	
運算元:	f : 00h ~ 7fh	
運算:	當 f = 0 , Z ← 1	
影響旗標:	Z	
運算碼:	00 1000 1fff ffff	
定義:	若 f = 0 將零旗標 Z 設為 1 , 否則 Z = 0	
工作週期:	1	
範例:	TESTZ FR	指令執行前 FR=00h , Z=? 指令執行後 FR=00h , Z=1

**XORWF**


---

指令:	XORWF f [,d]	
運算元:	f : 00h ~ 7fh , d : 0 , 1	
運算:	(目的暫存器) ← W XOR f	
影響旗標:	Z	
運算碼:	00 0110 dfff ffff	
定義:	將 W 與 f 作邏輯互斥或 XOR (Exclusive OR) 運算 若是 d = 0 , 則將 W XOR f 的值放入工作暫存器 W 若是 d = 1 , 則將 W XOR f 的值放入原暫存器 f 中	
工作週期:	1	
範例:	IORWF FR , 0	指令執行前 W=b5h , FR=afh 指令執行後 W=1ah , FR=afh
	IORWF FR , 1	指令執行前 W=b5h , FR=afh 指令執行後 W=b5h , FR=1ah

## 2-1-2 F-Plane 位元資料運算指令

**BCF**


---

指令:	BCF f [,b]	
運算元:	f : 00h ~ 7fh , b : 0 ~ 7	
運算:	f.b ← 0	
影響旗標:	-	
運算碼:	01 000b bbff ffff	
定義:	將 f 的第 b 位元清除為 0	
工作週期:	1	
範例:	BCF FR , 0	指令執行前 FR=ffh 指令執行後 FR=feh
	BCF FR , 7	指令執行前 FR=ffh 指令執行後 FR=7fh

**BSF**


---

指令:	BSF f [,b]	
運算元:	f : 00h ~ 7fh , b : 0~7	
運算:	f.b ← 1	
影響旗標:	-	
運算碼:	01 001b bbff ffff	
定義:	將 f 的第 b 位元設定為 1	
工作週期:	1	
範例:	BSF FR , 0	指令執行前 FR=00h 指令執行後 FR=01h
	BSF FR , 7	指令執行前 FR=00h 指令執行後 FR=80h

**BTFSC**

---

指令:	BTFSC f [,b]	
運算元:	f : 00h ~ 7fh , b : 0 ~ 7	
運算:	f.b = 0 則跳過下一個指令，執行下下一個指令	
影響旗標:	-	
運算碼:	01 010b bbff ffff	
定義:	測試 f 的第 b 位元是否為 0，若為 0 則跳過下一個指令，執行下 一個指令，所需時間為 2 個工作週期 若不為 0 則 PC = LABEL + 1，執行下一個指令，所需時間為 1 個工作週期	
工作週期:	1 or 2	
範例:	LABEL : BTFSC FR , 3 GOTO LOOP CONTINUE	指令執行前 PC= LABEL 指令執行後 If FR.3=0 , PC=LABEL+2 If FR.3=1 , PC=LABEL+1

**BTFSS**


---

指令:	BTFSS f [,b]	
運算元:	f : 00h ~ 7fh , b : 0~7	
運算:	f.b = 1 則跳過下一個指令，執行下下一個指令	
影響旗標:	-	
運算碼:	01 011b bbff ffff	
定義:	測試 f 的第 b 位元是否為 1 若為 1 則跳過下一個指令，執行下 一個指令，所需時間為 2 個工作週期 若不為 1 則 PC = LABEL + 1，執行下一個指令，所需時間為 1 個 工作週期	
工作週期:	1 or 2	
範例:	LABEL : BTFSS FR , 3 GOTO LOOP CONTINUE	指令執行前 PC= LABEL 指令執行後 if FR.3=1 , PC=LABEL+2 if FR.3=0 , PC=LABEL+1

## 2-1-3 R-Plane 資料運算指令

**MOVRW**


---

指令:	MOVRW r	
運算元:	r : 00h ~ ffh	
運算:	$W \leftarrow r$	
影響旗標:	-	
運算碼:	01 1111 rrrr rrrr	
定義:	將 r 暫存器的數值移到 W	
工作週期:	1	
範例:	MOVRW RR	指令執行前 W=?, RR=10h 指令執行後 W=10h , RR=10h

**MOVWR**


---

指令:	MOVWR r	
運算元:	r : 00h ~ ffh	
運算:	$r \leftarrow W$	
影響旗標:	-	
運算碼:	01 1110 rrrr rrrr	
定義:	將 W 數值移到暫存器 r 內	
工作週期:	1	
範例:	MOVWR RR	指令執行前 W=10h , RR=ffh 指令執行後 W=10h , RR=10h

## 2-1-4 常數運算指令

**ADDLW**


---

指令:	ADDLW k	
運算元:	k : 00h ~ ffh	
運算:	$W \leftarrow W + k$	
影響旗標:	C, DC, Z	
運算碼:	01 1100 kkkk kkkk	
定義:	將 W 數值加上 k 放入 W	
工作週期:	1	
範例:	ADDLW 10h	指令執行前 W=21h 指令執行後 W=31h

**ANDLW**


---

指令:	ANDLW k	
運算元:	k : 00h ~ ffh	
運算:	$W \leftarrow W \text{ AND } k$	
影響旗標:	Z	
運算碼:	01 1011 kkkk kkkk	
定義:	將 W 和 k 做 AND 運算 結果放入 W	
工作週期:	1	
範例:	ANDLW 1bh	指令執行前 W=37h 指令執行後 W=13h

**XORLW**


---

指令:	XORLW k	
運算元:	k : 00h ~ ffh	
運算:	$W \leftarrow W \text{ XOR } k$	
影響旗標:	Z	
運算碼:	01 1101 kkkk kkkk	
定義:	將 W 和 k 做 XOR(Exclusive OR) 運算 結果放入 W	
工作週期:	1	
範例:	XORLW 1bh	指令執行前 W=37h 指令執行後 W=2ch

**IORLW**

---

指令:	IORLW k	
運算元:	k : 00h ~ ffh	
運算:	$W \leftarrow W \text{ OR } k$	
影響旗標:	Z	
運算碼:	01 1010 kkkk kkkk	
定義:	將 W 和 k 做 OR 運算結果放入 W	
工作週期:	1	
範例:	IORLW 1bh	指令執行前 W=37h 指令執行後 W=3fh

**MOVLW**

---

指令:	MOVLW k	
運算元:	k : 00h ~ ffh	
運算:	$W \leftarrow k$	
影響旗標:	-	
運算碼:	01 1001 kkkk kkkk	
定義:	將 k 數值移到 W	
工作週期:	1	
範例:	MOVLW 3bh	指令執行前 W=00h 指令執行後 W=3bh

## 2-1-5 分支指令

**CALL**


---

指令:	CALL k
運算元:	k : 000h ~ fffh
運算:	TOS $\leftarrow$ (PC) + 1 , PC.11 ~ 0 $\leftarrow$ k
影響旗標:	-
運算碼:	10 kkkk kkkk kkkk
定義:	副程式呼叫，先將 PC + 1 放入堆疊頂端再將呼叫位址 K 放入 PC 中

工作週期:	2
範例:	LABEL1 : CALL SUB1      指令執行前 PC = LABEL1 指令執行後 PC = SUB1 TOS = LABEL1+1

**GOTO**


---

指令:	GOTO k
運算元:	k : 000h ~ fffh
運算:	PC.11~0 $\leftarrow$ k
影響旗標:	-
運算碼:	11 kkkk kkkk kkkk
定義:	跳躍到位址 k 的指令將位址 k 放入 PC 中

工作週期:	2
範例:	LABEL1 : GOTO SUB1      指令執行前 PC = LABEL1 指令執行後 PC = SUB1

**RET**


---

指令:	RET
運算元:	-
運算:	PC $\leftarrow$ TOS
影響旗標:	-
運算碼:	00 0000 0100 0000
定義:	一般副程式返回，PC = 堆疊最頂端之返回位址值
工作週期:	2
範例:	RET      指令執行後 PC = TOS

**RETI**

指令: RETI  
 運算元: -  
 運算:  $PC \leftarrow TOS, GIE \leftarrow 1$   
 影響旗標: -  
 運算碼: 00 0000 0110 0000  
 定義: 中段副程式之返回指令, PC = 堆疊最頂端之返回位址值, 並將中斷致能, 即 GIE = 1  
 工作週期: 2  
 範例: RETI                                    指令執行後 PC = TOS , GIE = 1

**RETLW**

指令: RETLW k  
 運算元: k : 00h ~ ffh  
 運算:  $PC \leftarrow TOS, W \leftarrow k$   
 影響旗標: -  
 運算碼: 01 1000 kkkk kkkk  
 定義: PC = 堆疊最頂端(返回位址)  
       並將常數值 k 放入 W, 用於傳回副程式執行結果或查表  
 工作週期: 2  
 範例:                                    CALL TABLE                                    指令執行前 W = 07h  
   :  
   :  
   :  
   TABLE : ADDWF PC , 1  
   RETLW k1  
   :  
   :  
   RETLW kn

## 2-1-6 其他指令

**CLRWDT**


---

指令:	CLRWDT	
運算元:	-	
運算:	WDT ← 00h	
影響旗標:	-	
運算碼:	01 1110 0000 0100	
定義:	清除 Watchdog Timer	
工作週期:	1	
範例:	CLRWDT	指令執行前 WDT counter=? 指令執行後 WDT counter=00h

**NOP**


---

指令:	NOP
運算元:	-
運算:	不動作
影響旗標:	-
運算碼:	00 0000 0000 0000
定義:	不動作
工作週期:	1
範例:	NOP

**SLEEP**


---

指令:	SLEEP
運算元:	-
運算:	-
影響旗標:	-
運算碼:	01 1110 0000 0011
定義:	進入睡眠狀態
工作週期:	1
範例:	SLEEP

## 2-1-7 虛指令

編譯器中常用的虛指令之用法如下表：

名稱	用法	定義
ORG	ORG 00h	下一指令放入程式記憶體 00h
EQU	P1 EQU 33h	定義常數 P1，其值為 33h
DEFSTR	P1.3 DEFSTR 33h,3	定義字串 P1.3，其值為 33h,3
DB	DB 33h	將目前程式記憶體位址內容設定為 33h

## 2-2 指令集說明表

使用 F-Plane 位元組運作之指令			
撰寫方法	週期	影響旗標	意義
ADDWF f,d	1	C,DC,Z	若 d = 0, $W = W + f$ 若 d = 1, $f = W + f$
ANDWF f,d	1	Z	若 d = 0, $W = W \text{ AND } f$ 若 d = 1, $f = W \text{ AND } f$
CLRF f	1	Z	$f = 0, z = 1$
CLRW	1	Z	$W = 0, z = 1$
COMF f,d	1	Z	若 d = 0, $W = \bar{f}$ 若 d = 1, $f = \bar{f}$
DECF f,d	1	Z	若 d = 0, $W = f - 1$ 若 d = 1, $f = f - 1$
DECFSZ f,d	1or2	-	(目的暫存器) $\leftarrow f - 1$ 若 $f - 1 = 0$ , 跳過下一道指令
INCF f,d	1	Z	若 d = 0, $W = f + 1$ 若 d = 1, $f = f + 1$
INCFSZ f,d	1or2	-	(目的暫存器) $\leftarrow f + 1$ 若 $f + 1 = 0$ , 跳過下一道指令
IORWF f,d	1	Z	若 d = 0, $W = W \text{ OR } f$ 若 d = 1, $f = W \text{ OR } f$
MOVWF f	1	-	將數值從 f 移動到 W
MOVWF f	1	-	將數值從 W 移動到 f
RLF f,d	1	C	將 f 數值與進位旗標做循環左移
RRF f,d	1	C	將 f 數值與進位旗標做循環右移
SUBWF f,d	1	C,DC,Z	若 d = 0, $W = f - W$ 若 d = 1, $f = f - W$
SWAPF f,d	1	-	d = 0, W = 原 f 暫存器之高低四位元互換之結果 d = 1, f = 原 f 暫存器之高低四位元互換之結果
TESTZ f	1	Z	若 $f = 0, Z = 1$ 否則 $Z = 0$
XORWF f,d	1	Z	若 d = 0, $W = W \text{ XOR } f$ 若 d = 1, $f = W \text{ XOR } f$
使用 F-Plane 位元運作之指令			
撰寫方法	週期	影響旗標	意義
BCF f,b	1	-	將 f 的第 b 位元清除為 0
BSF f,b	1	-	將 f 的第 b 位元設定為 1
BTFSC f,b	1or2	-	若 $f.b = 0$ , 跳過下一個指令
BTFSS f,b	1or2	-	若 $f.b = 1$ , 跳過下一個指令

使用 R-Plane 位元運作之指令			
MOVRW r	1	-	將 r 暫存器的數值移到 W
MOVWR r	1	-	將 W 數值移到暫存器 r 內

使用常數運作之指令			
ADDLW k	1	C,DC,Z	$W = W + k$
ANDLW k	1	Z	$W = W \text{ AND } k$
XORLW k	1	Z	$W = W \text{ XOR } k$
IORLW k	1	Z	$W = W \text{ OR } k$
MOVLW k	1	-	$W = k$

分支指令			
撰寫方法	週期	影響旗標	意義
CALL k	2	-	呼叫位址 k 之副程式
GOTO k	2	-	無條件跳躍到位址 k
RET	2	-	一般副程式返回
RETI	2	-	中斷副程式返回
RETLW k	2	-	一般副程式返回，並將常數值 k 放入 W

其他指令			
CLRWDW	1	-	清除 Watchdog Timer
NOP	1	-	不動作
SLEEP	1	-	進入睡眠狀態

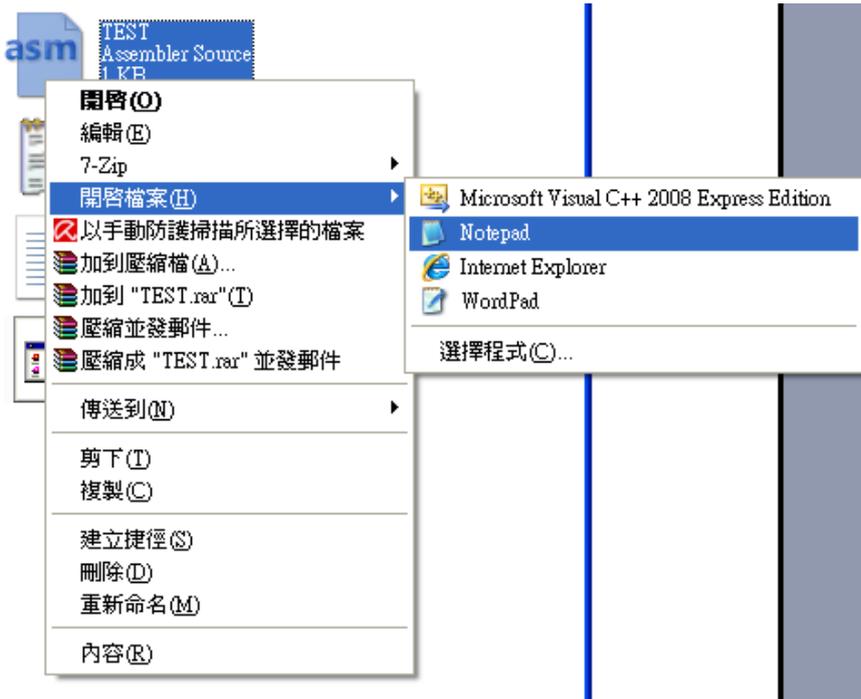
### 2-3 程式的編輯

TM57FLA80 晶片之組合語言程式，其編輯方法可分兩種方式。

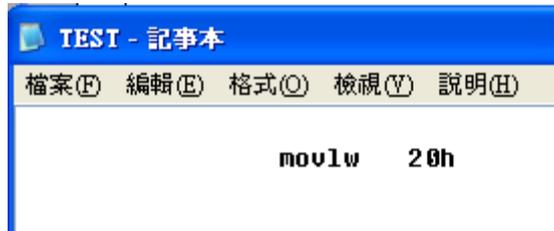
- 一、直接使用模擬器做編輯，詳如第三章模擬器使用說明。
- 二、使用其他編輯軟體來編輯組合語言程式檔案(例如 Notepad)。

使用 Notepad 編輯軟體來輸入組合語言程式檔案之方法如下

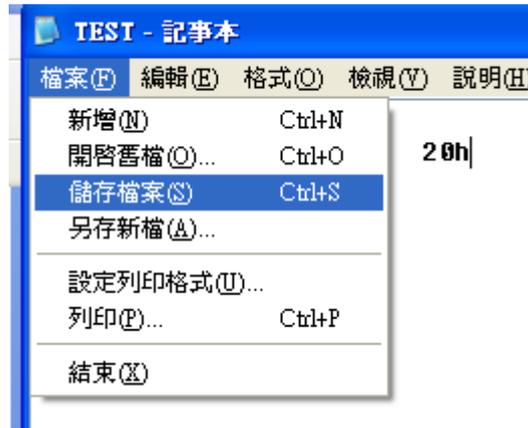
1. 以 Notepad 開啟編輯視窗



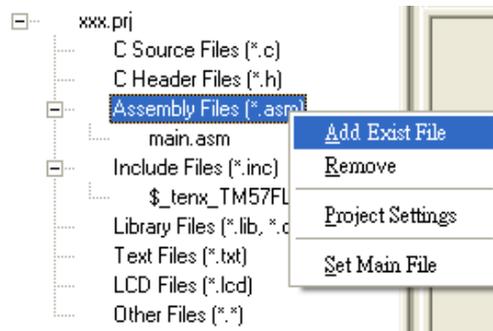
2. 輸入組合語言書寫之程式



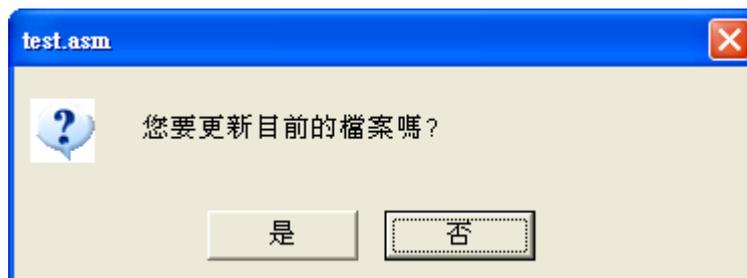
3. 儲存程式，副檔名為.asm；且存為 ANSI 的純文字檔



4. 利用模擬軟體將程式載入到模擬器中



當程式檔案以 Notepad 編輯時，若程式同時在模擬器中開啟，在 Notepad 編輯系統中儲存檔案時，在模擬器的程式視窗會顯示下面訊息：



按是，即可更新模擬器中的程式。

## 2-4 程式書寫的方式

程式中“;”後之文字代表註解不會被翻譯為機械碼，在書寫程式前先利用 EQU 定義會使用到的記憶體位址，如：“變數 EQU 記憶體位址”，接著利用 ORG 指令來定義指令擺放的位址，再依指令執行的先後順序寫入程式中即可完成程式的書寫，含有中斷的程式範例說明如下：

;首先定義會使用到的記憶體位址

```
PC          equ      02h      ;將PC定義為02H之記憶體位址
TM1L       equ      0ah
TM1H       equ      0bh
PED        equ      13h
```

;接著利用ORG指令定義指令擺放位址

```
org        00h      ;定義下一道指令 goto Start 之位址為00h
goto      Start
```

;若是有使用中斷,可利用ORG指令搭配中斷位址，完成含中斷副程式  
;的位址設定，並利用goto指令完成中斷副程式的配置。

```
org        01h      ;01h為Timer0的中斷
goto      Timer0
org        02h      ;02h為Timer1的中斷
goto      Timer1
org        03h      ;03h為Timer2的中斷
goto      Timer2
org        04h      ;04h為PWM0的中斷
goto      PWM0I
org        05h      ;05h為WKT的中斷
goto      WKTl
org        06h      ;06h為XINTA的中斷
goto      XINTAI
org        07h      ;07h為XINTB的中斷
goto      XINTBI
org        08h      ;08h為UART的中斷
goto      UARTI
org        09h      ;09h為SPI的中斷
goto      SPll
```

;主程式由此開始，指令的格式“標記：指令 數值或變數；註解”

```
Start:     movlw 10
           movwf digit_loop
           movlw 0
           movwf TM1H
           :
           :
           :

           bsf      TM1IE
           bcf      STOPTM1

           goto   $
```

;goto \$ 為跳到本行，可讓程式停止  
;副程式寫在主程式之後，如delay為一延遲副程式。

```

delay:    movlw 10
          movwf R1

delay_L1: movlw 200
          movwf R2
          :
          :
          :
          goto    delay_L1
          ret      ;返回主程式
    
```

;一般副程式由 ret 指令結束，返回呼叫點之位址。

;中斷副程式

```

Timer1:  decfsz    count, 1
          goto    NOT_1s

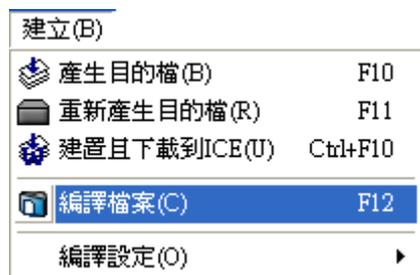
          movlw 40
          :
          :
          :
          :
          clrf     digit

NORMAL:  movfw     digit
          call    Table7S
          movwf  PED

NOT_1s:  bcf      TM1I
          reti
    
```

;中斷副程式返回指令為 reti，

程式撰寫完成後，可利用模擬器的編譯檔案功能進行編譯(Compile)。



編譯後會出現2種可能的情形:

1. 有語法錯誤，會顯示出錯誤地方以及個數，此時須修改程式後再重新編譯，直到所有錯誤更正後才可執行程式。



當有錯誤時，可在輸出窗格中錯誤處(如下圖所示) 雙擊左鍵，找尋錯誤的地方，將錯誤更正，直到所有錯誤均正確為止。



2. 當程式語法正確後，才可執行程式。



### 3. 模擬器使用說明

#### 3-1 安裝 TICE99



圖 TICE99

##### 第一步

將 TICE99 的安裝光碟放入光碟機中，系統會自動執行安裝程式。如果沒有自動運行，則直接執行光碟根目錄下的 `setup_TICE99_BetaVersionV102Build009.exe` 也可以進入安裝程式。

##### 第二步

於圖 3-1 中，點選 Next，執行下一步。

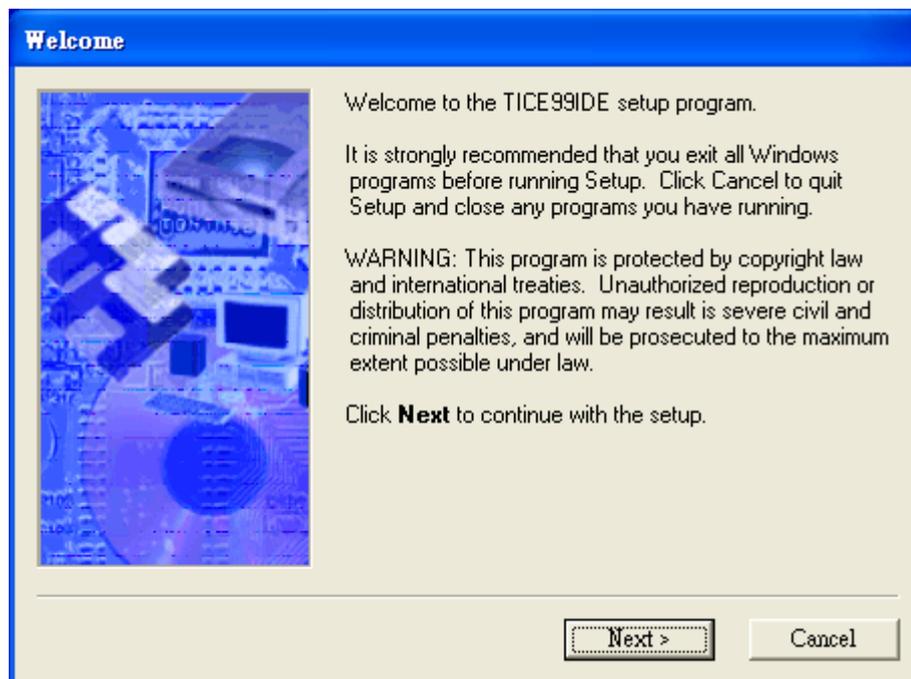


圖 3-1 Welcome 畫面

第三步

於圖 3-2 中，輸入使用者名稱(Name)和公司名稱(Company)，即可點選 Next，執行下一步。

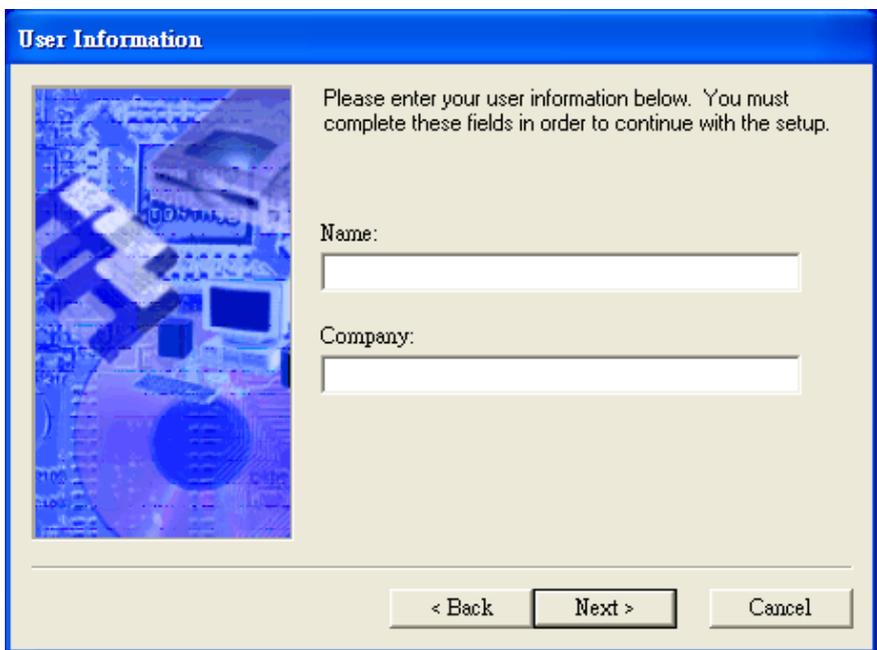


圖 3-2 User Information 畫面

第四步

於圖 3-3 中，選取安裝路徑，可直接使用預設路徑 C:\Program Files\tenx\TICE99IDE Beta Version V1.0.2Build009，或點選 Browse... 自行更換安裝路徑。選取完安裝路徑後，點選 Next，進行下一步。

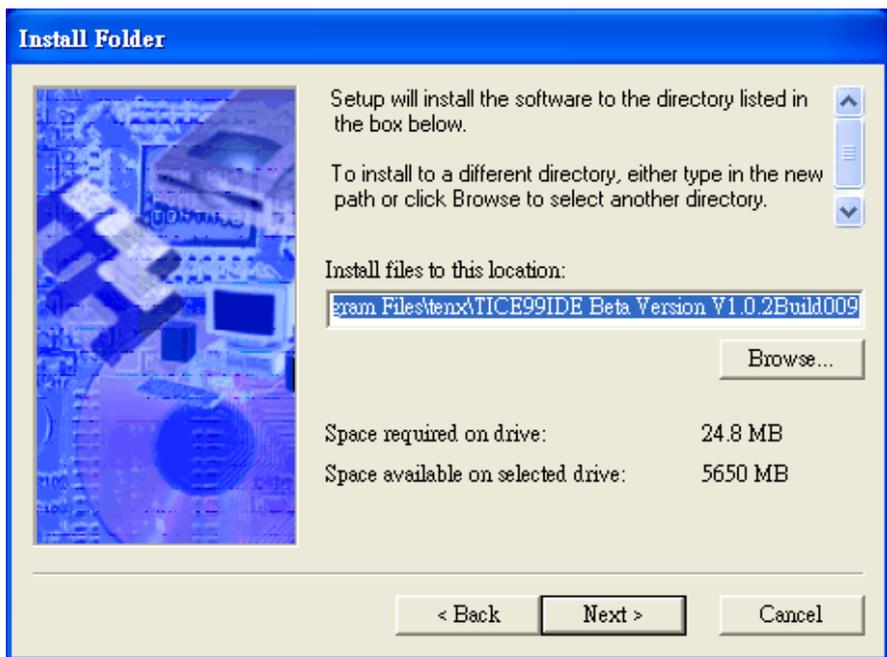


圖 3-3 Install Folder 畫面

第五步

設定「開始」功能表\程式集 裡 Users\「開始」功能表\程式集\tenx\TICE99IDE Beta Version V1.0.2Build009”，也可點選 Browse...自行更換安裝路徑。選取完路徑後，點選圖 3-4 中之 Next，進行下一步。的路徑，如圖 3-4，預設路徑為“C:\Documents and Settings\All

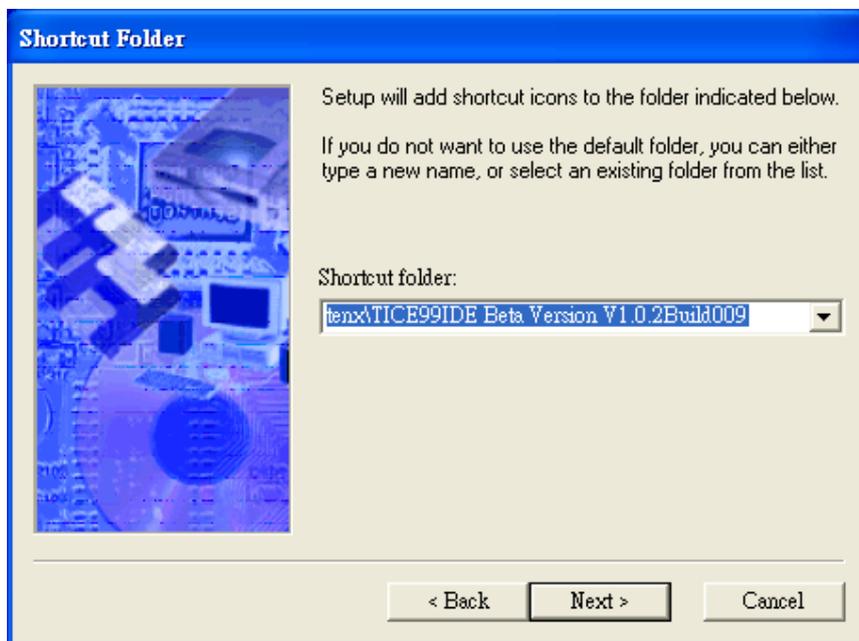


圖 3-4 Shortcut Folder 畫面

第六步

於圖 3-5 中，點選 Install，開始安裝。

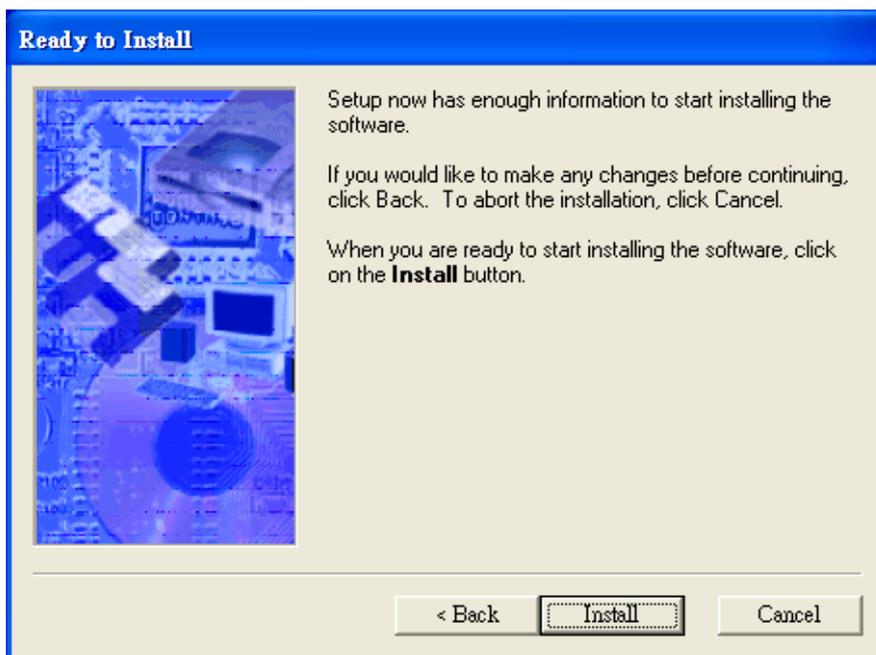


圖 3-5 Ready to Install 畫面

第七步

安裝中，如圖 3-6。



圖 3-6 Installing Files 畫面

安裝結束後，點選圖 3-7 中之 Finish，結束安裝程式。

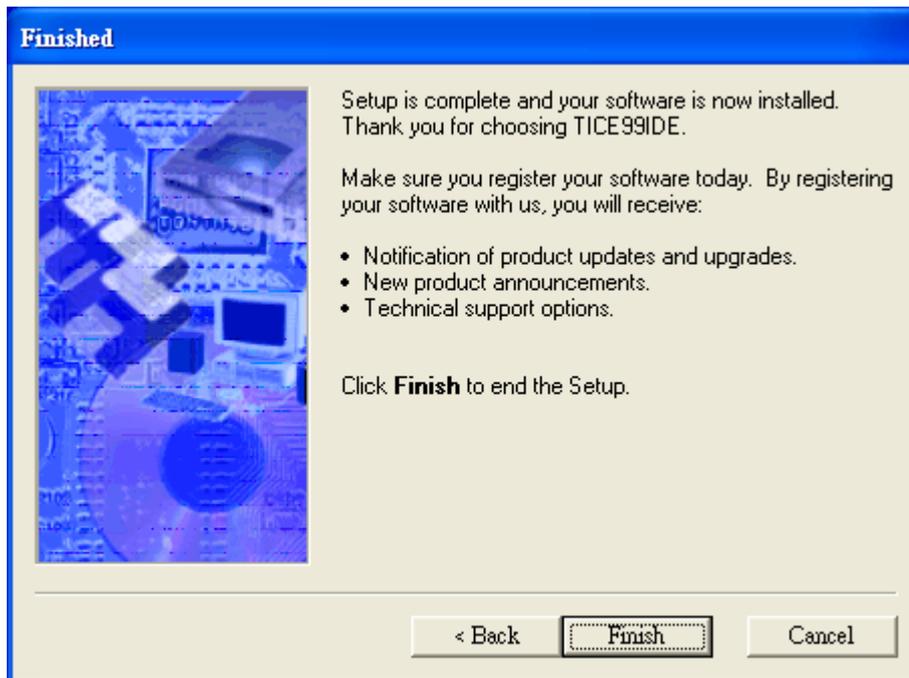


圖 3-7 Finished 畫面

3-2 模擬器介面之介紹

安裝完成後，點選「開始」功能表\所有程式\tenx\TICE99IDE Beta Version V1.0.2Build009\TICE99IDE.exe”或點選桌面上的快捷鍵 TICE99IDE Beta Version V1.0.2Build009.exe，即可進入模擬器執行程式的測試與模擬。進入模擬器後之程式操作介面如圖 3-8 所示。

1	主選單	2	工具列
3	專案總管窗格	4	程式編輯窗格
5	暫存器窗格	6	編譯結果顯示窗格
7	堆疊與程式記憶體顯示窗格	8	變數窗格

圖 3-8 操作介面

3-3 新增與開啟專案

(1) 語系更改

第一次執行程式，系統預設語系為英文。可由功能表列的“Tools\ Language\Traditional Chinese”更改為繁體中文，如圖 3-9 所示。

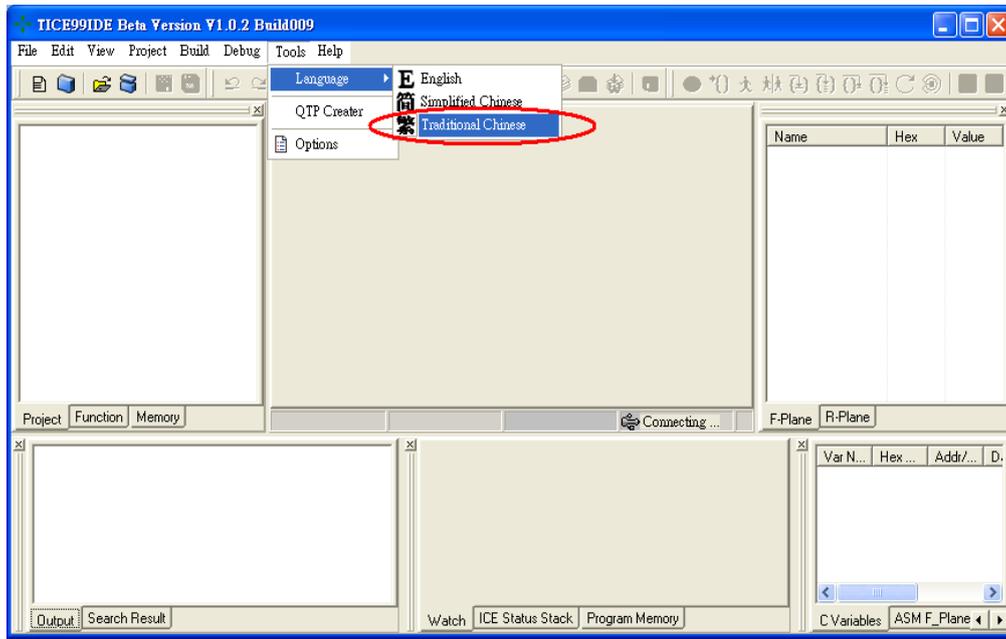


圖 3-9 英文語系介面

語系更改為繁體中文後，如圖 3-10 所示。

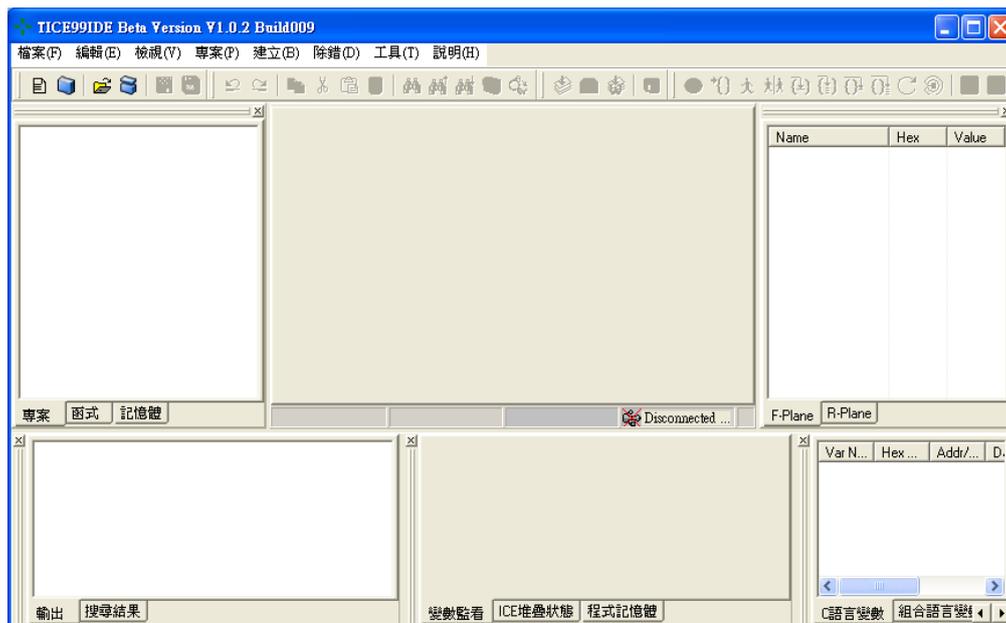


圖 3-10 繁體中文語系介面

(2) 新增專案

可由“檔案\新增\專案”中選取，如圖 3-11 所示。

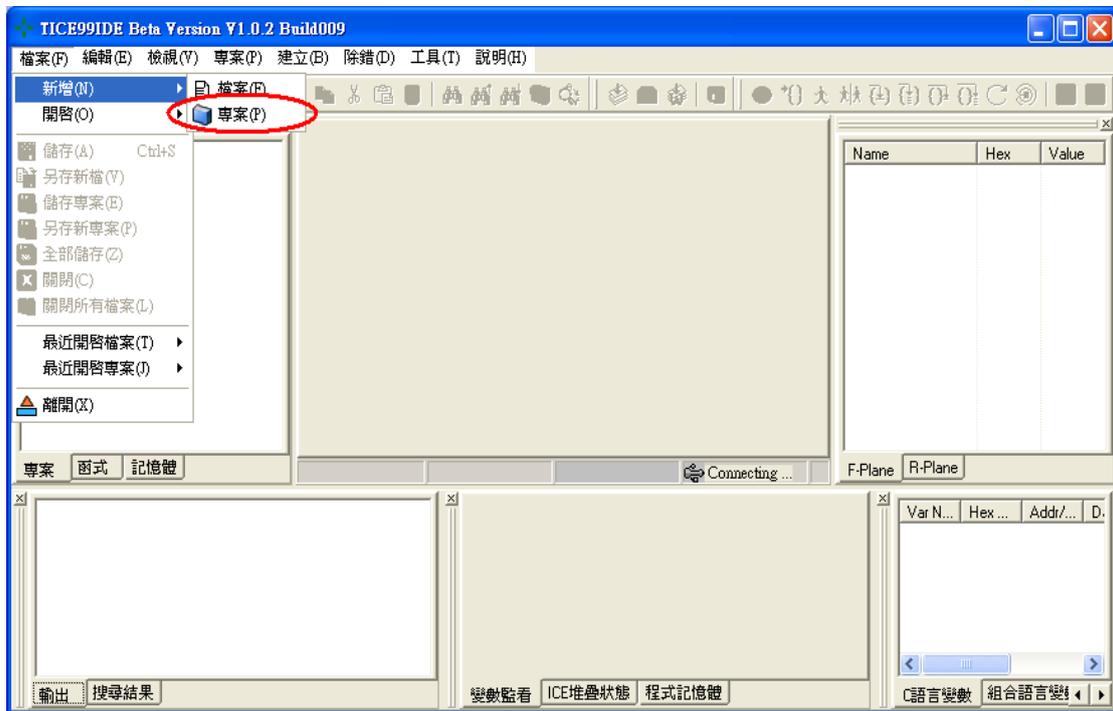


圖 3-11 專案選取(從主選單)

亦可直接點選工具列的新增專案圖示，如圖 3-12 所示。

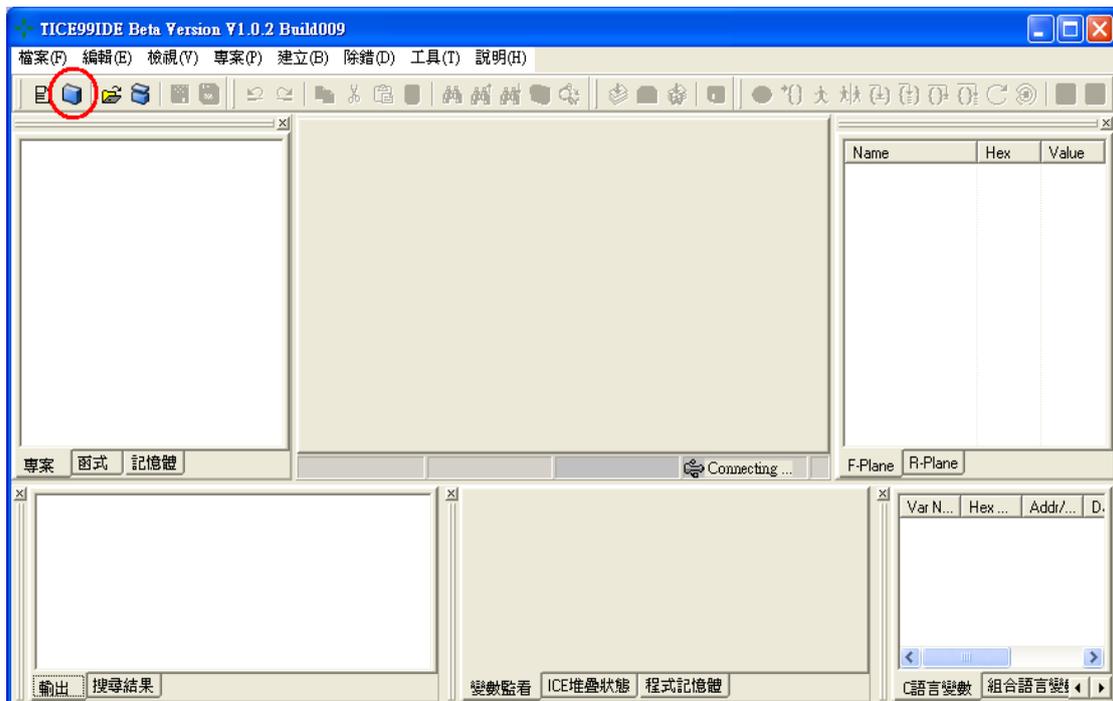


圖 3-12 專案選取(從工具列)

專案之設定如圖 3-13 所示，選擇專案中使用之微處理器晶片之類別、使用語言、專案名稱、專案所在位置後，點選確定。（本書實習皆使用組合語言，故以組合語言為例。）

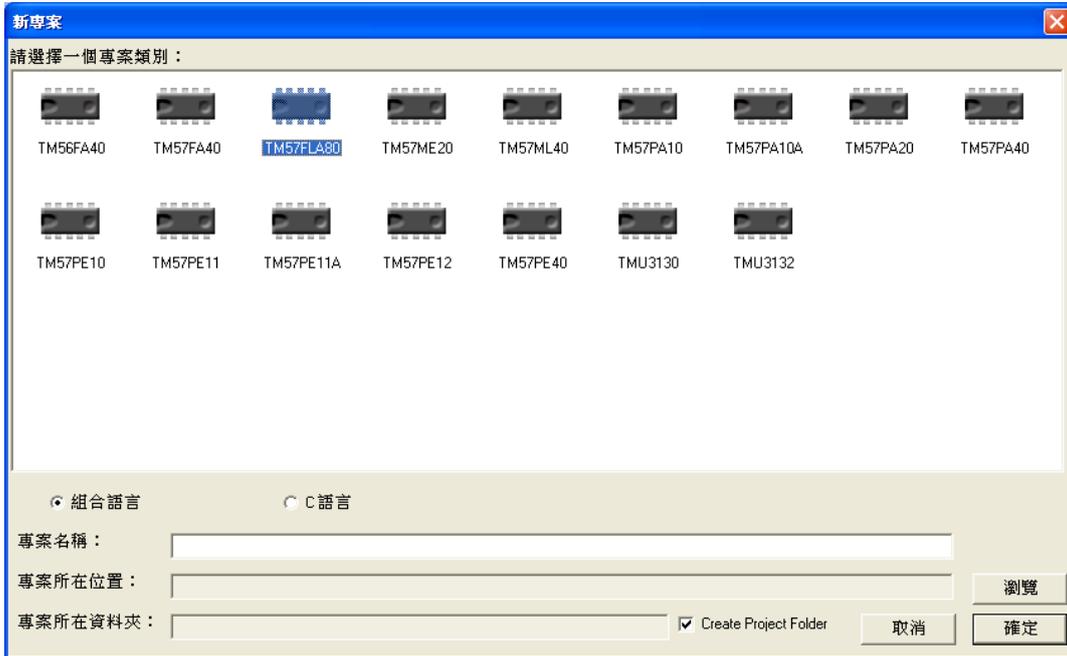


圖 3-13 新專案之設定

設定完成後，新專案如圖 3-14 所示。

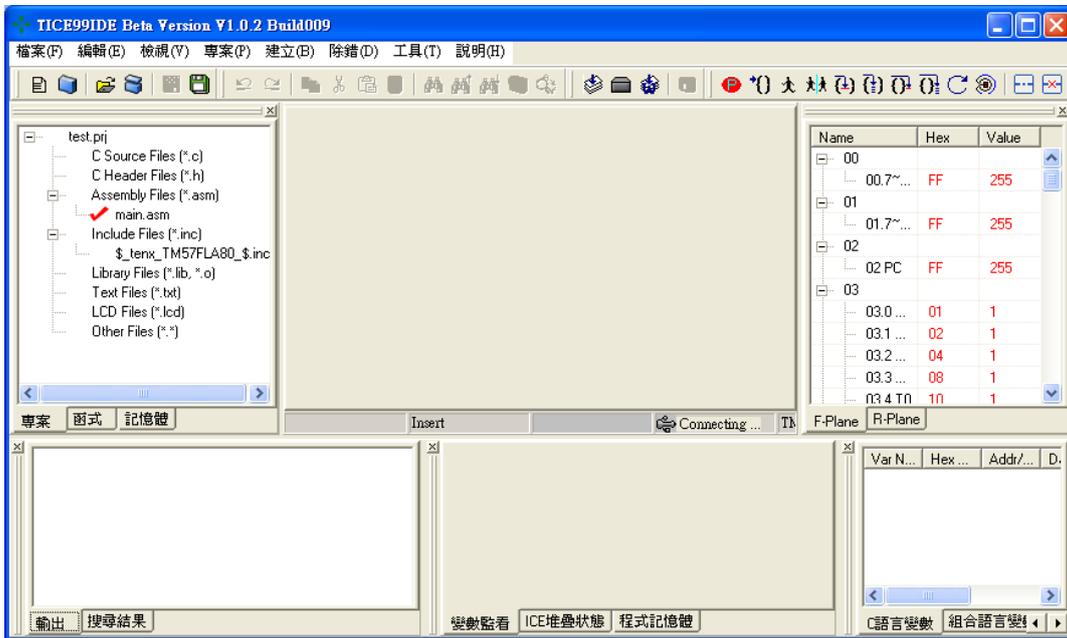


圖 3-14 新專案

若已經有寫好的程式檔想直接使用，可在專案總管中點選右鍵的 Add Exist File。如圖 3-15 所示。

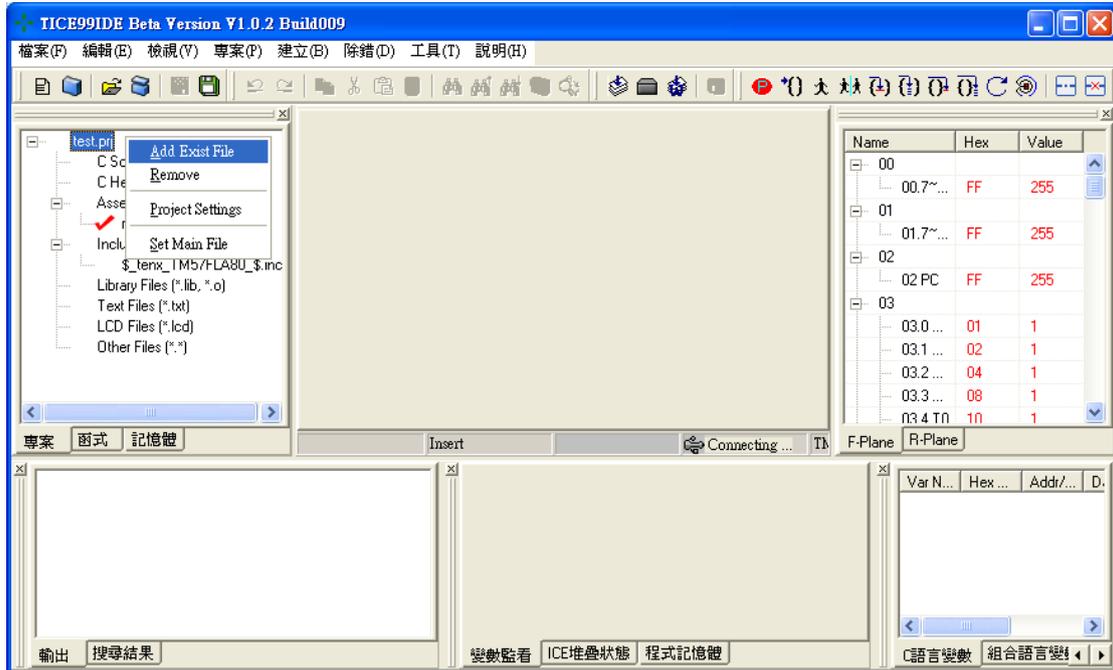


圖 3-15 新增程式檔

加入程式檔之後，記得在程式檔名稱點選右鍵的 Set Main File。將其設置為主程式檔，如圖 3-16 所示。

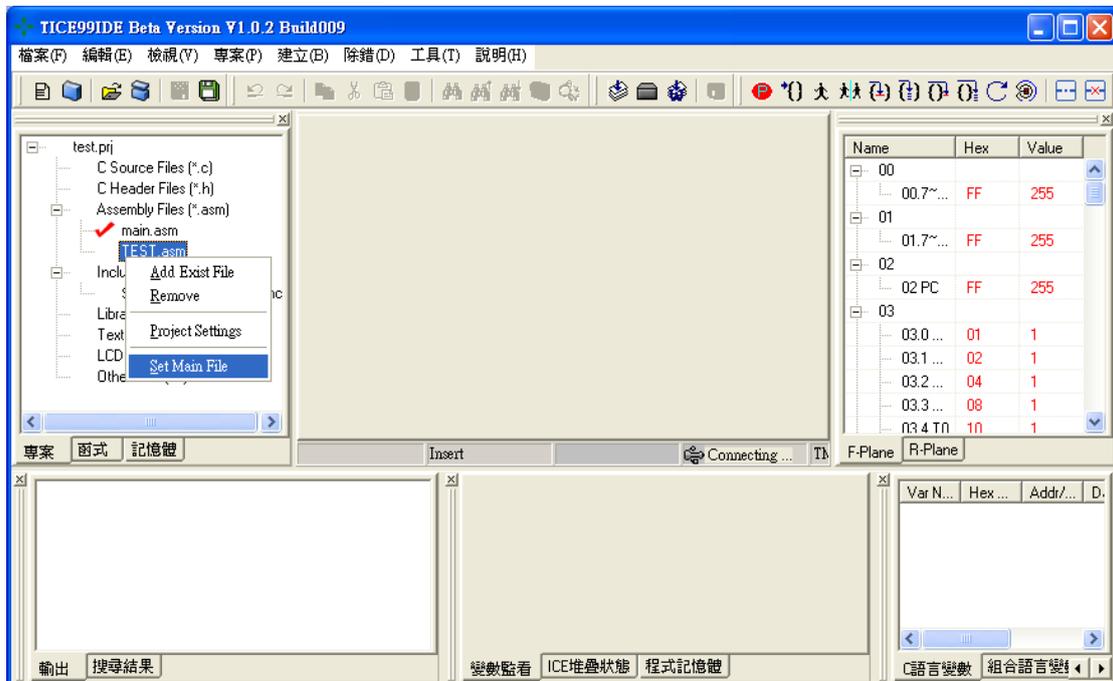


圖 3-16 設定主程式檔

建議將非主程式檔的其他程式檔刪除，就不會造成程式碼衝突而產生執行上的錯誤。如圖 3-17 所示。

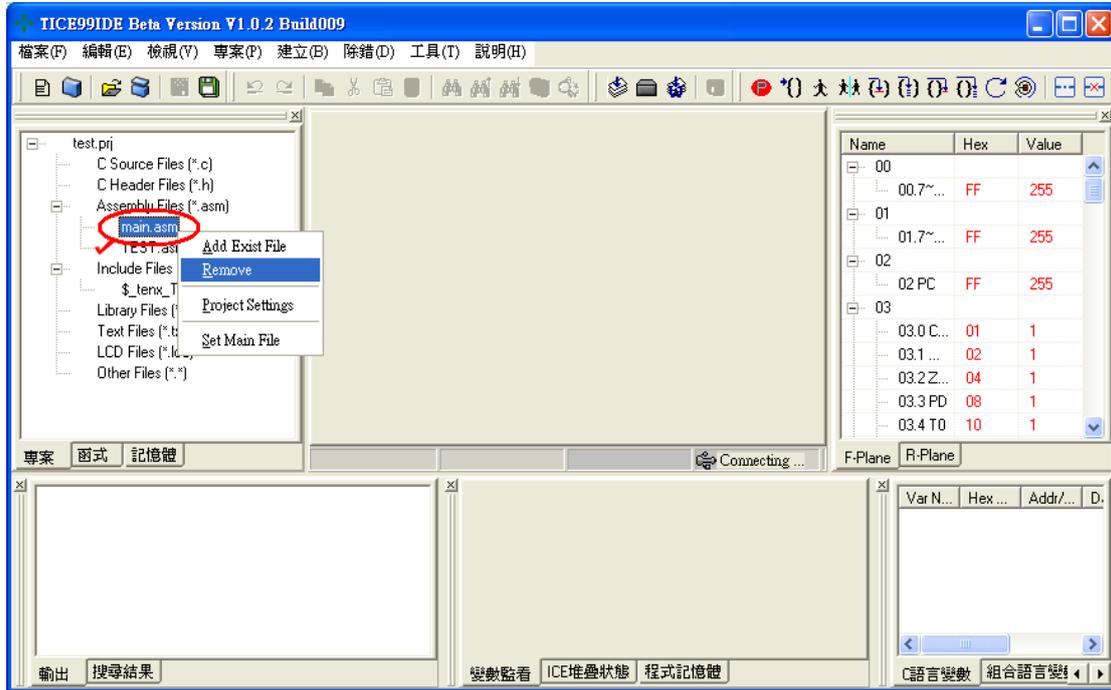


圖 3-17 移除非主程式檔

設定後之結果，如圖 3-18 所示。

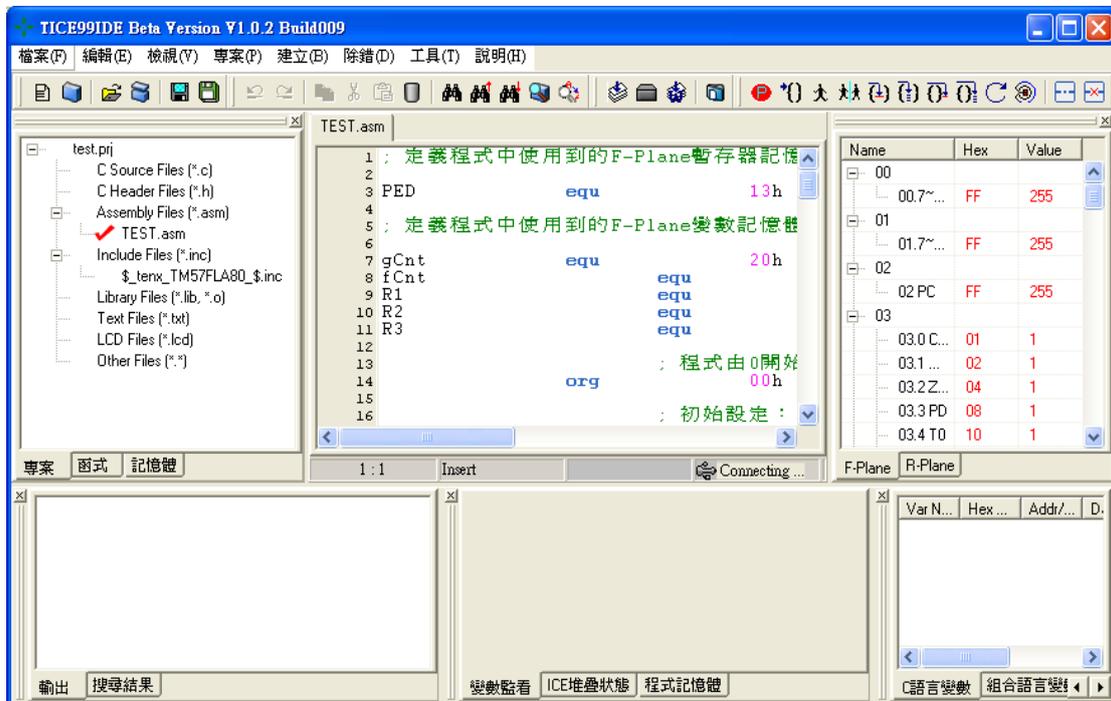


圖 3-18 程式檔設定完成

### 3-4 執行與除錯

點選“建立\產生目的檔”，即可編譯程式並產生目的檔。如圖 3-19 所示。

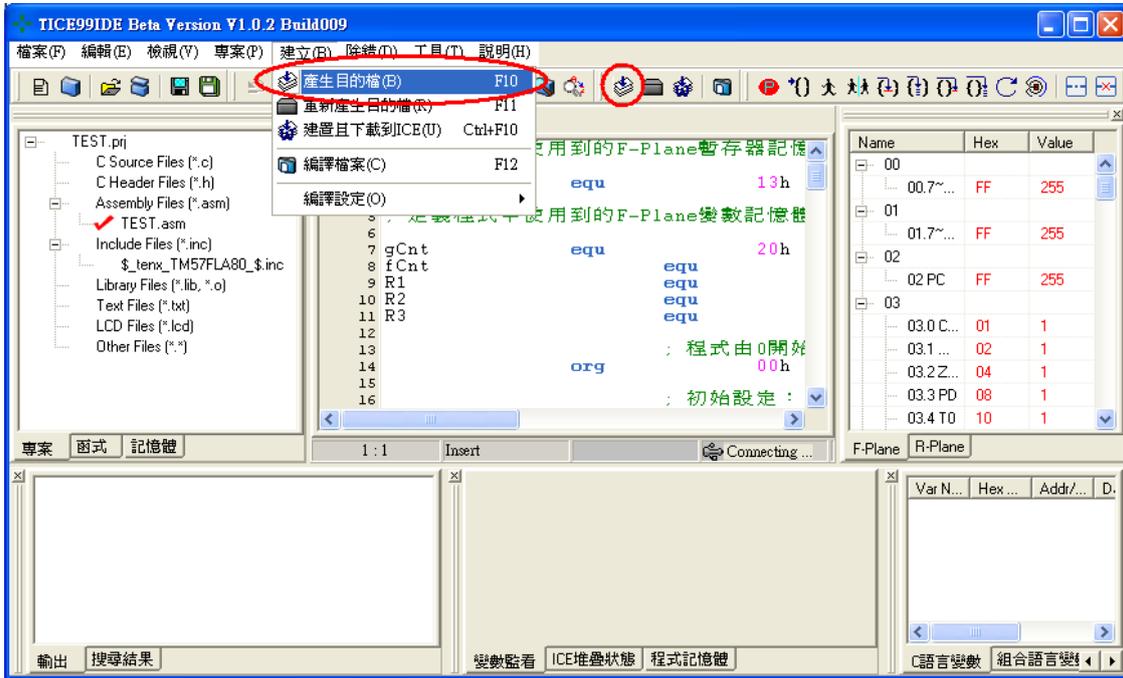


圖 3-19 產生目的檔

編譯結果顯示於左下角的輸出窗格中，程式是否有語法錯誤將顯示於此窗格中。如圖 3-20 所示。

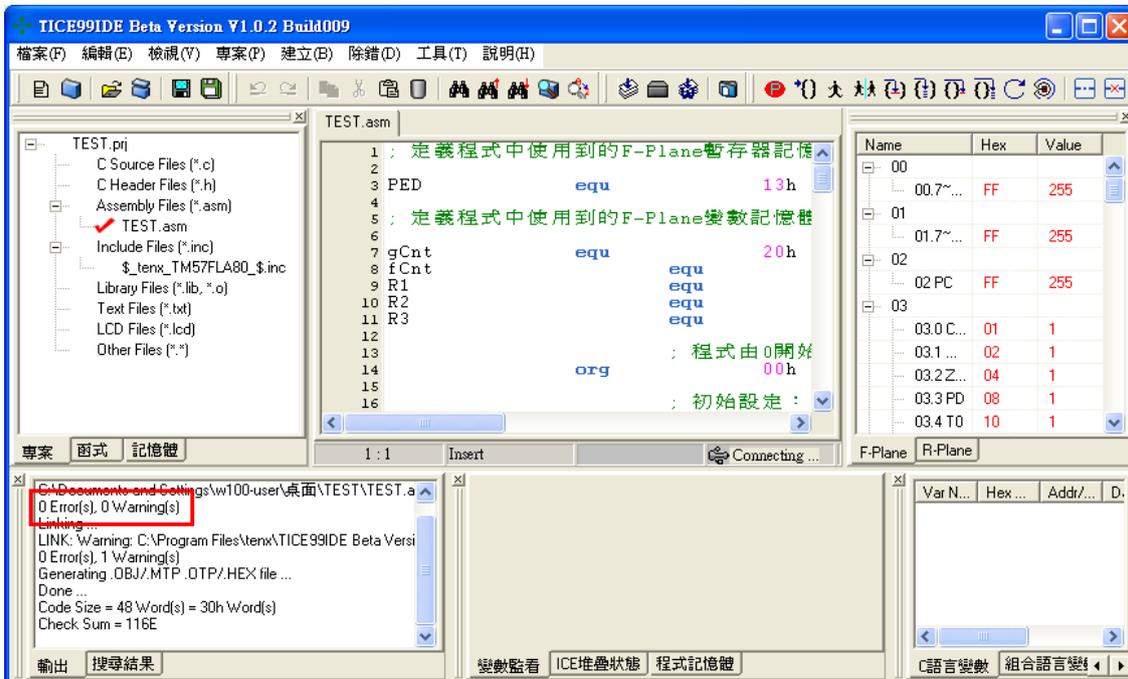


圖 3-20 編譯結果顯示

當程式語法無誤後，可點選“建立\建置且下載到 ICE”，來編譯程式並下載程式到 ICE。如圖 3-21 所示。

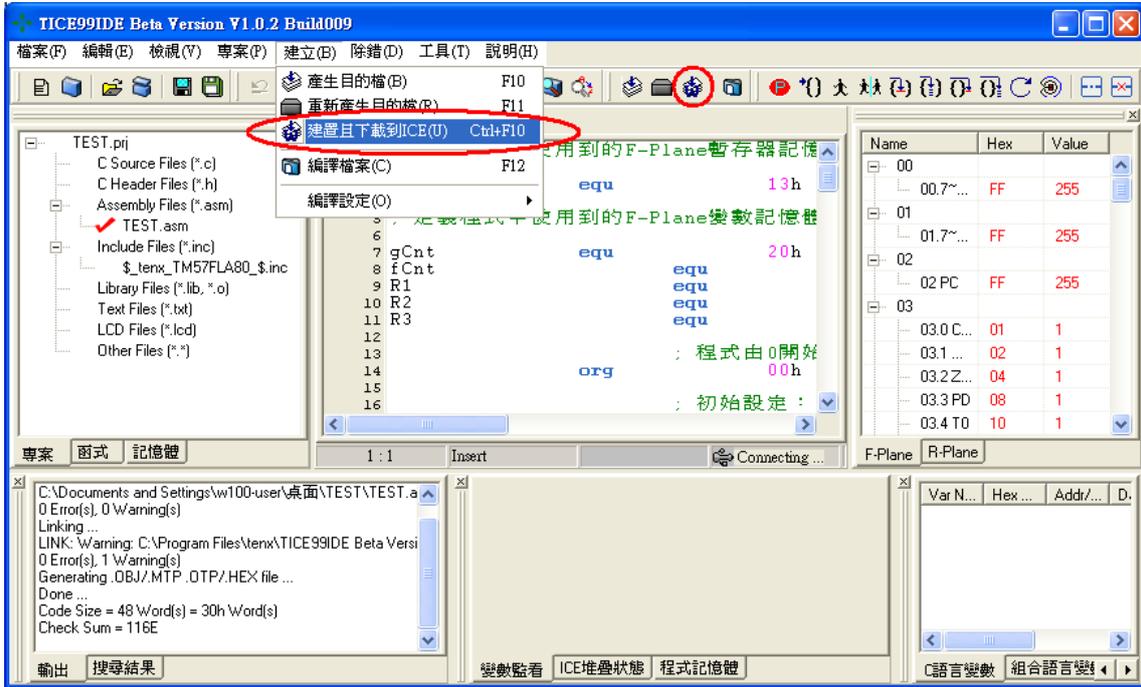


圖 3-21 建置且下載到 ICE

程式下載到 ICE 後，即可點選「除錯」中的指令(如 3-5-1~6 節之說明)，開始執行程式的測試工作。如圖 3-22 所示。

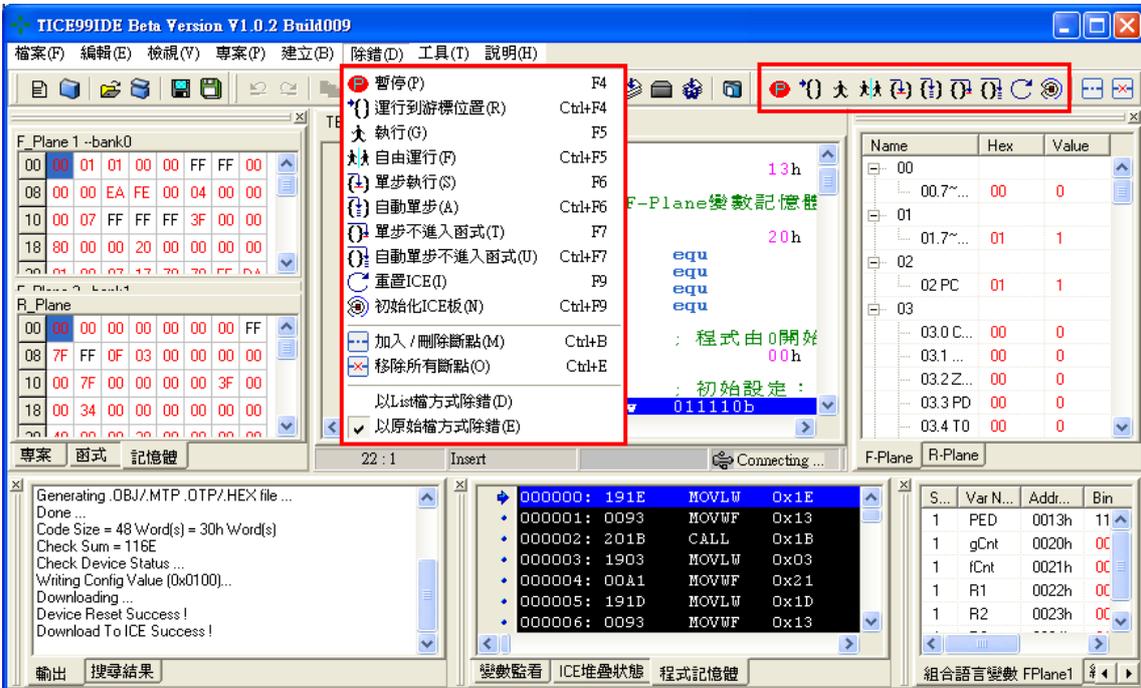


圖 3-22 除錯

斷點的設定方式可以參考下面的方式來設定：  
點選程式窗格內行數旁的點，即可設立斷點。如圖 3-23 所示。

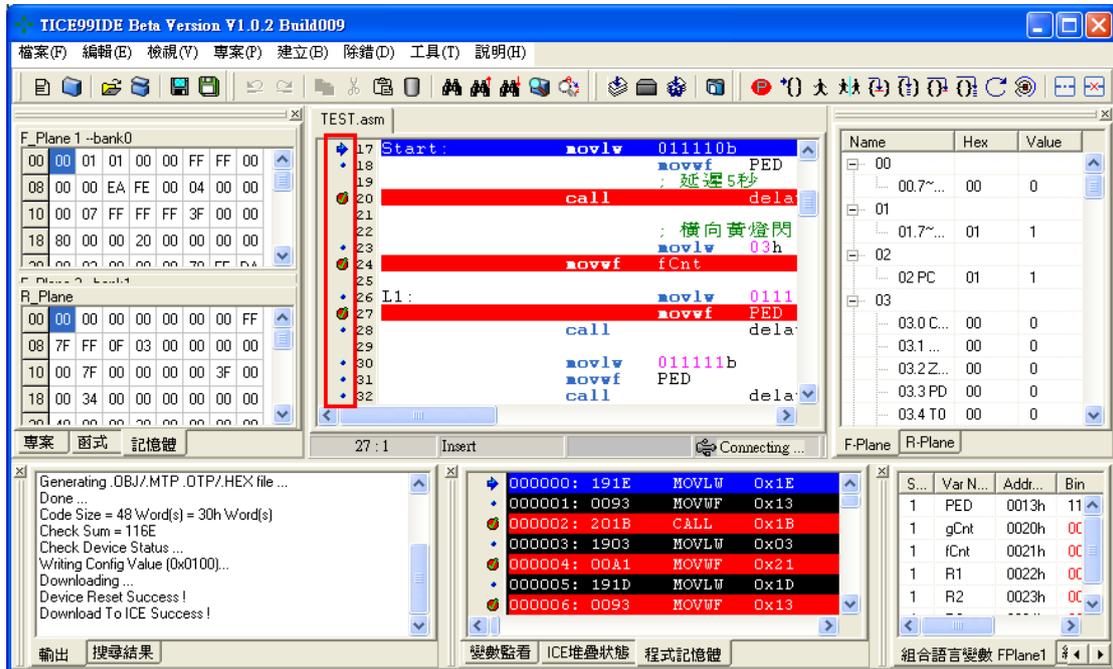


圖 3-23 設立斷點

設立斷點後，執行程式會自動停在設定的斷點的指令上。此時再觀看記憶體內的變化，以方便除錯。如圖 3-24 所示。

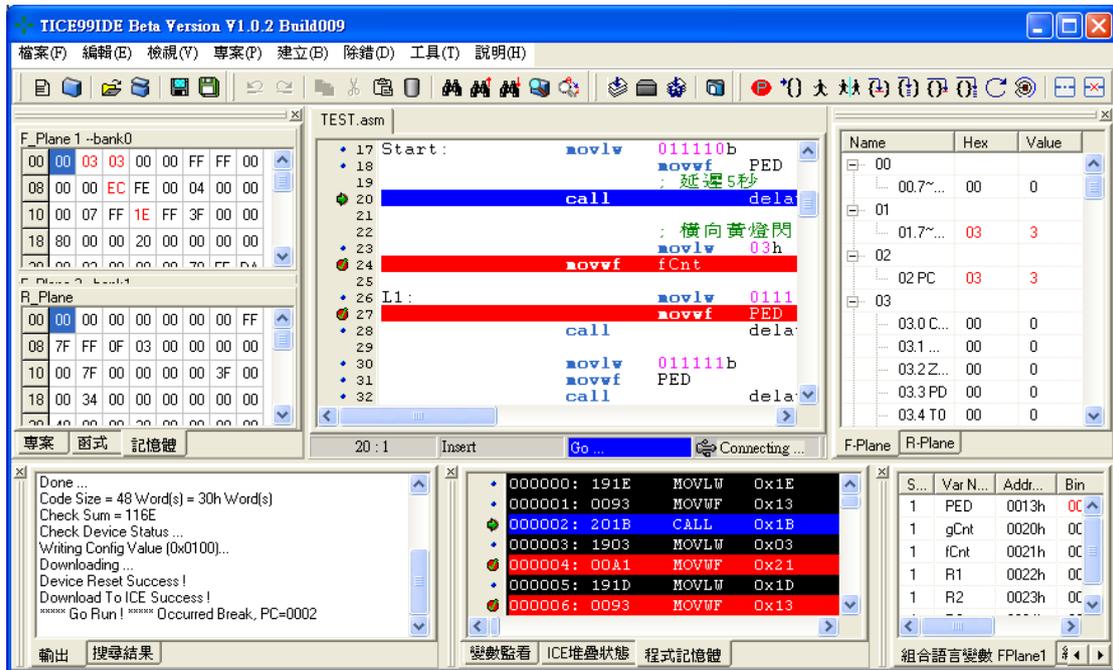


圖 3-24 斷點與除錯

3-5 主選單與工具列之功能介紹

3-5-1 主選單之功能介紹

ICE 的主選單有下列的項目可以設定，

檔案(F) 編輯(E) 檢視(V) 專案(P) 建立(B) 除錯(D) 工具(T) 說明(H)

本節將就其功能做一個簡單的介紹。

3-5-1-1 檔案子選單

檔案子選單之各選項功能說明

	新增	檔案	新增一個檔案
		專案	新增一個專案
	開啟	檔案	選擇路徑開啟原有檔案
		專案	選擇路徑開啟原有專案
	儲存		儲存檔案
	另存新檔		選擇路徑另外儲存檔案
	儲存專案		儲存專案
	另存新專案		選擇路徑另外儲存專案
	全部儲存		儲存現有檔案與專案
	關閉		關閉正在編輯的檔案
	關閉所有檔案		關閉所有檔案
	最近開啟檔案		從選單中開啟最近使用的檔案
	最近開啟專案		從選單中開啟最近使用的專案
	離開		離開 ICE

3-5-1-2 編輯子選單

編輯子選單之各選項功能說明

	返回	回到上一個動作
	重做	回到返回前一個動作
	複製	複製選擇的內容
	剪下	剪下選擇的內容
	貼上	貼上剪下或複製選擇的內容
	刪除	刪除選擇的內容
	選擇全部	選擇全部內容
	尋找	從程式中尋找相同的字串
	找上一個	往上尋找下一個相同字串
	找下一個	往下尋找下一個相同字串
	在專案中尋找	在整個專案中尋找
	尋找並取代	把尋找到的相同字串，由新的字串取代
	到	游標跳到輸入行數

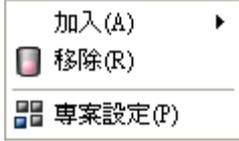
3-5-1-3 檢視子選單

檢視子選單之各選項功能說明

	專案管理員	專案總管顯示專案窗格	
	函式檢視器	專案總管顯示函式窗格	
	記憶體視窗	專案總管顯示記憶體窗格	
	暫存器檢視	暫存器窗格顯示 F-Plane 窗格	
	變數監看	檢視變數	
	ICE 堆疊狀態	檢視堆疊狀態	
	程式記憶體	檢視程式記憶體內容	
	C 語言變數	檢視 C 語言變數	
	組合語言變數 F-Plane1	檢視 F-Plane1 之變數	
	組合語言變數 F-Plane2	檢視 F-Plane2 之變數	
	組合語言變數 R-Plane	檢視 R-Plane 之變數	
	輸出	顯示編譯結果	
	搜尋結果	從編譯結果中搜尋	
	LCD 編譯器	開啟 LCD 編譯器視窗	
	工具列	檔案	開啟/關閉檔案快捷列
		編輯	開啟/關閉編輯快捷列
編譯		開啟/關閉編譯快捷列	
除錯		開啟/關閉除錯快捷列	

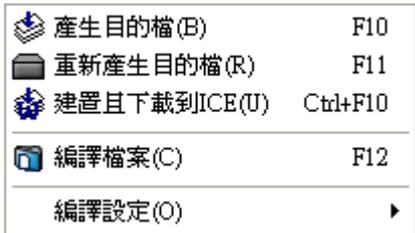
**3-5-1-4 專案子選單**

專案子選單之各選項功能說明

	加入	新檔	開啟新專案
		已存在檔案	開啟已存在的專案
	移除		移除專案裡的檔案
	專案設定		開啟專案設定視窗

**3-5-1-5 建立子選單**

建立子選單之各選項功能說明

	產生目的檔		編譯並產生目的檔
	重新產生目的檔		重新編譯並產生目的檔
	建置且下載到 ICE		編譯並下載到 ICE
	編譯檔案		編譯檔案
	編譯設定	除錯模式	更換除錯模式/發布模式
		發佈模式	

3-5-1-6 除錯子選單

除錯子選單之各選項功能說明

 <p> <input type="radio"/> 暫停(P) F4  <input type="radio"/> 運行到游標位置(R) Ctrl+F4  <input type="radio"/> 執行(G) F5  <input type="radio"/> 自由運行(F) Ctrl+F5  <input type="radio"/> 單步執行(S) F6  <input type="radio"/> 自動單步(A) Ctrl+F6  <input type="radio"/> 單步不進入函式(T) F7  <input type="radio"/> 自動單步不進入函式(U) Ctrl+F7  <input type="radio"/> 重置ICE(I) F9  <input type="radio"/> 初始化ICE板(N) Ctrl+F9  <hr/> <input type="checkbox"/> 加入 / 刪除斷點(M) Ctrl+B  <input type="checkbox"/> 移除所有斷點(O) Ctrl+E  <hr/> <input type="checkbox"/> 以List檔方式除錯(D)  <input checked="" type="checkbox"/> 以原始檔方式除錯(E)         </p>	暫停 (Pause)	將進行中程式暫停
	運行到游標位置 (Run to Cursor)	程式執行到游標位置
	執行 (Go)	全速執行程式，遇暫停或斷點才停止
	自由運行 (Free Run)	與全速執行程式同，但遇斷點不停止
	單步執行 (Single Step)	執行下一行指令，遇副程式會進入副程式
	自動單步 (Auto Single Step)	連續執行下一行指令，遇副程式會進入副程式，遇暫停或斷點才停止
	單步不進入函式 (Step Over)	執行下一行指令，遇副程式將副程式視為單一行指令，不進入副程式
	自動單步不進入函式 (Auto Step Over)	連續執行下一行指令，遇副程式將副程式視為單一行指令，不進入副程式，直到暫停或斷點才停止
	重置 ICE (Reset ICE)	重置 ICE
	初始化 ICE 板(Initiallize ICE Board)	將 ICE 初始化
	加入/刪除斷點	新增斷點與刪除斷點
	移除所有斷點	移除所有斷點
	以 List 檔方式除錯	以列表檔方式除錯
	以原始檔方式除錯	以程式檔方式除錯

3-5-2 工具列之使用說明

工具列選項如下



工具列選項說明如下表：

表 3-1 工具列選項說明表

	新增檔案		新增專案
	開啟檔案		開啟專案
	儲存檔案		全部儲存
	返回		重做
	複製		剪下
	貼上		刪除
	尋找		向上尋找
	向下尋找		在專案中尋找
	尋找並取代		編譯並產生目的檔
	重新產生目的檔		編譯且下載到 ICE
	編譯檔案		暫停
	運行到游標位置		全速執行
	自由運行		單步執行且進入副程式
	自動單步執行且進入副程式		單步執行，跨過副程式，將副程式當成一步
	自動單步執行，跨過副程式，將副程式當成一步		重置 ICE
	初始化 ICE 模擬器		加入/刪除斷點
	移除所有斷點		

#### 4. 紅綠燈實習

##### 實習目的：

本實習主要目的在學習使用單晶片進行基本輸出控制。

##### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

##### 實習材料：

項次	品名	數量
1	74LS245	1
2	紅 LED	2
3	黃 LED	2
4	綠 LED	2
5	10KΩ排阻(6PIN 以上)	1
6	330Ω排阻(6PIN 以上)	1

##### 實習板模組與 I/O Port:

模組	I/O Port
LED 模組	Port E

##### 實習說明：

本實習將使用十速 57FLA80 系列單晶片控制 LED 燈，模擬紅綠燈控制系統的運作，紅綠燈配置如圖 2-1 所示，包括兩組紅綠燈，一組控制縱向車道，另一組控制橫向車道。

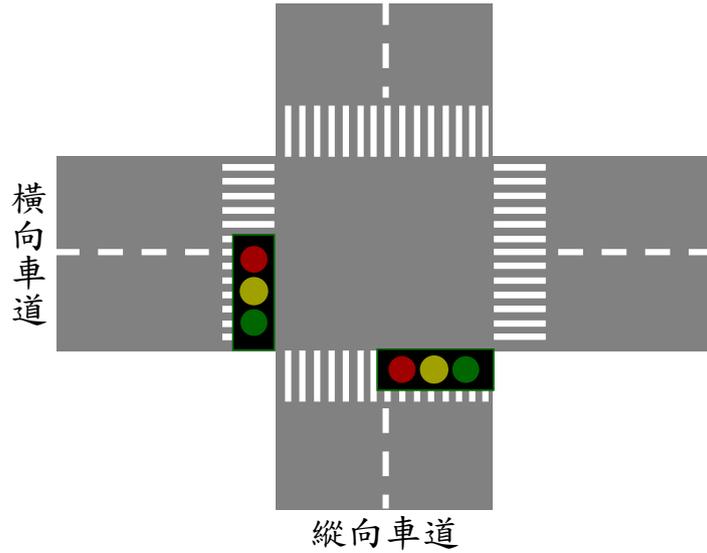


圖 4-1 紅綠燈配置圖

紅綠燈系統的運作方式如圖 2-2 所示，紅綠燈系統一開始的狀態如狀態一所示，縱向紅綠燈是紅燈橫向紅綠燈為綠燈，等待一段時間後，橫向紅綠燈會進入閃黃燈狀態（狀態二），接著才變成紅燈，橫向紅綠燈變成紅燈的同時，縱向紅綠燈會變成綠燈（狀態三）。同樣的，縱向紅綠燈也會經過綠燈變閃黃燈（狀態四）再變紅燈的過程（狀態一）。以上過程會重覆執行，直到系統關機為止。

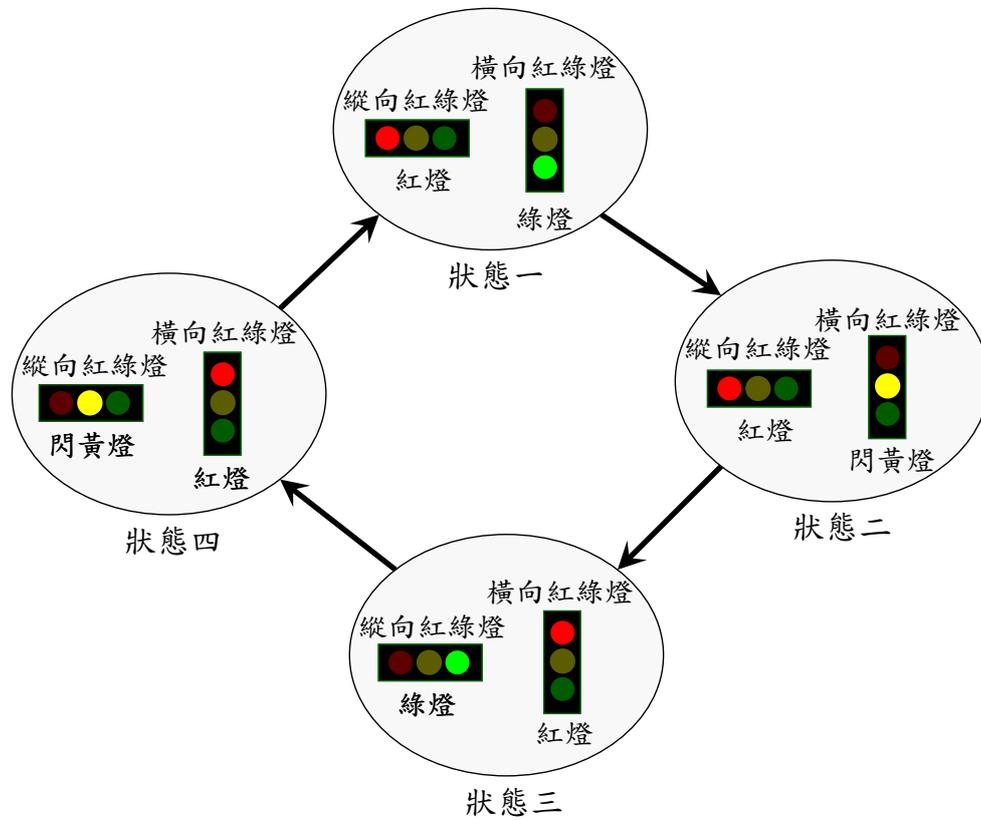


圖 4-2 紅綠燈系統運作方式

硬體線路圖：

紅綠燈模擬系統硬體線路如圖 4-3 所示，本實習分別使用 57FLA80 單晶片 Port E 的 PE5, PE4, PE3, PE2, PE1, PE0 等 6 支腳位來控制縱向紅綠燈的紅黃綠與橫向紅綠燈的紅黃綠等 6 個 LED 的亮滅。為了提供 LED 足夠的電源推力以及保護單晶片，Port E 出來的控制訊號不直接接到 LED，而是先接到 74245 緩衝器，經緩衝器出來的訊號再接到 LED。另外，為確保 LED 的電流量充足，我們將 LED 的正極直接接到電源，由電源負責供電，負極接到緩衝器，由緩衝器的輸出結果控制 LED 的亮滅，緩衝器的輸出為低電位(0)時，可點亮 LED，反之，可關掉 LED。這種作法可確保多顆 LED 同時點亮時，每顆 LED 都會有足夠的電流量可以使用，降低當多顆 LED 同時點亮時，個別 LED 亮度不足情形發生的可能性。

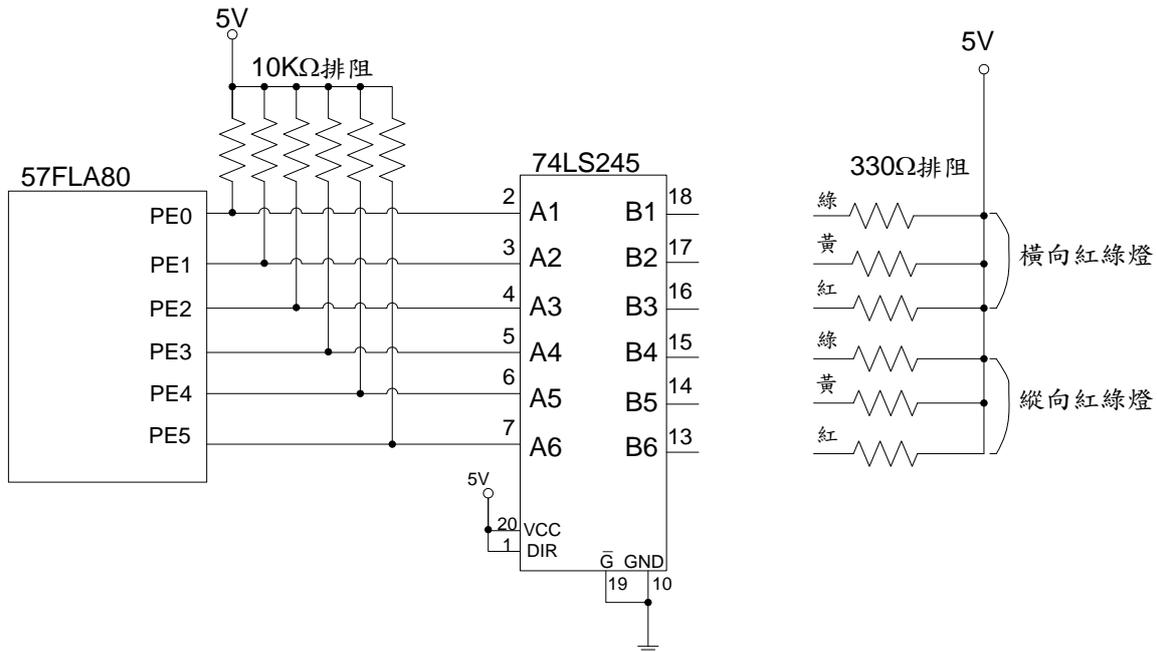


圖 4-3 紅綠燈系統硬體線路圖

程式流程圖：

紅綠燈模擬系統程式流程如圖 2-4 所示，首先設定初始狀態，初始狀態設定會將縱向紅綠燈設為紅燈，橫向紅綠燈設為綠燈；接著必須等待一段時間，讓橫向道路的車輛通過，本實習假設綠燈持續時間為 5 秒；綠燈等待時間過後，橫向紅綠燈必須進入閃黃燈狀態，本實習假設黃燈閃爍三次，燈亮與燈暗持續時間分別為 0.5 秒，共 3 秒；橫向黃燈閃爍完畢，將縱向與橫向紅綠燈變成綠燈與紅燈。同樣的，縱向紅綠燈也會由綠燈變閃黃燈再到紅燈，縱向紅綠燈的等待時間設定跟橫向紅綠燈相同。

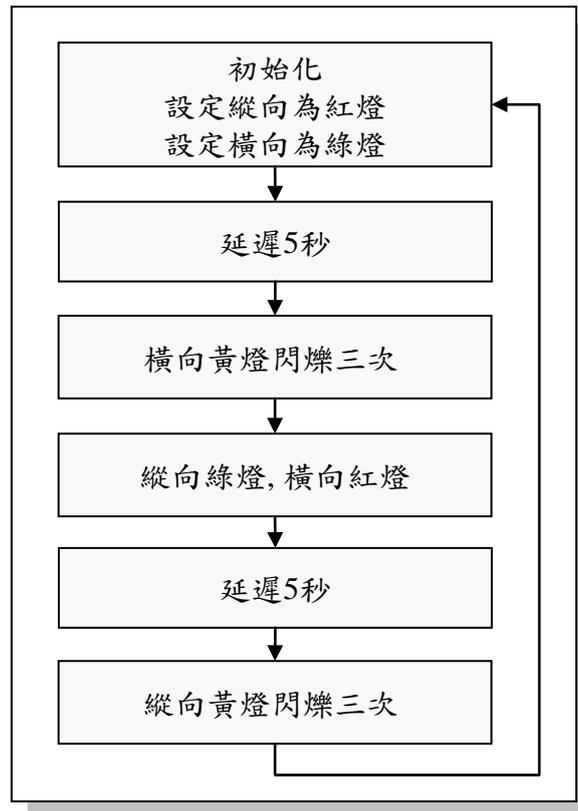


圖 4-4 紅綠燈系統運作方式

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址
PED          equ          13h          ; Port E

; 定義程式中使用到的F-Plane變數記憶體位址
gCnt         equ          20h          ; 綠燈延遲時間(單位：0.5秒)
fCnt         equ          21h          ; 黃燈閃爍次數
R1           equ          22h          ; 供delay函式使用之外層迴圈控制變數
R2           equ          23h          ; 供delay函式使用之中層迴圈控制變數
R3           equ          24h          ; 供delay函式使用之內層迴圈控制變數

; 系統開機進入點
org          00h

; 初始設定：縱向紅燈,橫向綠燈
Start:       movlw        011110b     ; 縱向紅燈, 橫向綠燈為011110b
             movwf        PED         ; 將紅綠燈設定寫入Port E

; 延遲5秒
             call         delay5      ; 呼叫delay5副程式, 延遲5秒

; 橫向黃燈閃爍三次
             movlw        03h         ; 設定黃燈閃爍3次
             movwf        fCnt        ; 將閃爍次數放入fCnt
L1:          movlw        011101b     ; 點亮橫向紅綠燈的黃燈
             movwf        PED         ; 將紅綠燈設定寫入Port E
             call         delay        ; 延遲0.5秒

             movlw        011111b     ; 關掉橫向紅綠燈的黃燈
             movwf        PED         ; 將紅綠燈設定寫入Port E
             call         delay        ; 延遲0.5秒

             decfsz       fCnt, 1     ; 將fCnt減1, 若fCnt=0離開迴圈
             goto         L1          ;

; 縱向綠燈, 橫向紅燈
             m ovlw        110011b    ; 設定縱向綠燈, 橫向紅燈
             movwf        PED         ; 將紅綠燈設定寫入Port E

; 延遲5秒
             call         delay5      ; 呼叫delay5副程式, 延遲5秒

; 縱向黃燈閃爍三次
             movlw        03h         ; 設定黃燈閃爍3次

```

```

                movwf    fCnt        ; 將閃爍次數放入fCnt
L2:            movlw    101011b     ; 點亮縱向紅綠燈的黃燈
                movwf    PED        ; 將紅綠燈設定寫入Port E
                call    delay       ; 延遲0.5秒

                movlw    111011b   ; 關掉縱向紅綠燈的黃燈
                movwf    PED        ; 將紅綠燈設定寫入Port E
                call    delay       ; 延遲0.5秒

                decfsz   fCnt, 1    ; 將fCnt減1，若fCnt=0離開迴圈
                goto    L2          ;
                goto    Start       ; 回到程式開頭重新執行

;
; 延遲5秒副程式      : 重覆呼叫延遲0.5秒副程式(delay副程式)10次
;
delay5:        movlw    10          ; 設定迴圈重覆執行次數為10次
                movwf    gCnt       ;
delay5_L1:     call    delay       ; 呼叫延遲0.5秒副程式

                decfsz   gCnt, 1    ; 將gCnt減1，若gCnt=0離開迴圈
                goto    delay5_L1  ;

                ret                ; 返回呼叫程式

; 延遲0.5秒副程式
; 若clock rate=4MHz, 一個指令週期=2*clock=0.5μs,
; 要延遲0.5秒必須消耗1,000,000指令週期,
; 由於暫存器大小只有一個byte, 要延遲0.5秒需使用多層迴圈
; 本程式使用三層迴圈實作時間延遲功能, 時間延遲程式設計說明如下:
; 最內層迴圈消耗5指令週期, 執行200次, 共消耗1,000個指令週期
; 中間層迴圈執行100次, 共消耗約100,000個指令週期
; 最外層迴圈執行10次, 共消耗約1,000,000個指令週期

delay:         movlw    10          ; 設定外層迴圈執行10次
                movwf    R1        ;
; 外層迴圈
delay_L1:      movlw    100        ; 設定中層迴圈執行100次
                movwf    R2        ;
; 中層迴圈
delay_L2:      movlw    200        ; 設定內層迴圈執行200次
                movwf    R3        ;

```

```
        ;內層迴圈：迴圈跑一次約消耗5個指令週期
delay_L3: nop                ;消耗1個指令週期
        nop                  ;消耗1個指令週期
        decfsz    R3, 1      ;約消耗1個指令週期
        goto     delay_L3    ;消耗2個指令週期

        decfsz    R2, 1      ;將R2減1，若R2=0離開迴圈
        goto     delay_L2    ;

        decfsz    R1, 1      ;將R1減1，若R1=0離開迴圈
        goto     delay_L1    ;

        ret                ;返回呼叫程式
```

## 5. 指撥開關與七段顯示器實習

### 實習目的：

本實習主要目的在學習如何使用單晶片讀取輸入按鍵狀態以及控制七段顯示器顯示數字。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	74LS47 IC	1
2	七段顯示器(共陽)	1
3	按壓式按鍵	1
4	指撥開關 (4 個開關)	1
5	10KΩ排阻(9PIN 以上)	1
6	330Ω電阻	1

### 實習板模組與 I/O Port:

模組	I/O Port
指撥模組	Port D
四顆七段顯示器模組	Port E

**實習說明：**

本實習主要目的在熟悉使用單晶片讀取一般按鍵與指撥開關的狀態以及了解七段顯示器的控制方式。一般按鍵、指撥開關、以及七段顯示器用法說明如下：

- **按壓式按鍵(Push Button)**

指類似電腦鍵盤的按鍵，按壓式按鍵有兩種狀態：即按下與釋放，按鍵按下時，會使得原本沒有導通的電路被導通，改變電路的電位狀態，透過觀察相關電路的電位狀態，可了解按鍵有沒有被按下。圖 3-1 所示為按壓式按鍵的電路設計方式，如圖所示，按鍵沒有被按下時 PA0 會接收到高電位訊號(1)，反之，當按鍵被按下時 PA0 會接收到低電位訊號(0)，透過 PA0 所接收到的電位訊號即可了解按鍵是否被按下。

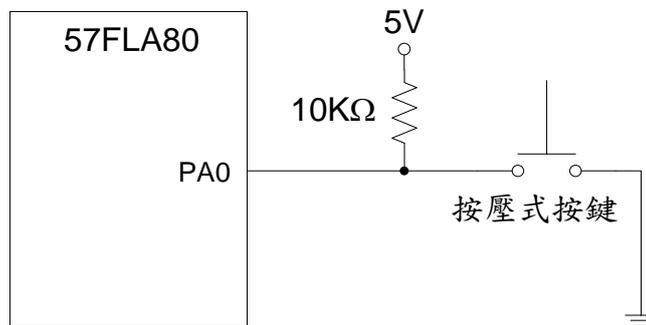


圖 5-1 按壓式按鍵使用範例

按壓式按鍵剛被按下時，可能會產生彈跳現象，以致雖然只有按一次鍵卻可能產生多次按鍵結果。為去除彈跳現象所造成的不良影響，通常的作法是在系統偵測到按鍵被按下後，先等待一段穩定時間，等到該穩定時間過後，再次檢查按鍵是否仍處於被按下的狀態，如果是才確認按鍵確實被按下。

- **指撥開關**

指撥開關是一種類似一般電源開關的裝置，具有"開"與"關"兩種狀態，可供使用者用來設定狀態用。指撥開關的一般用法如圖 3-2 所示，圖中所示指撥開關被切換到"開"的狀態時，電路會被切斷，因此 PA0 會接收到高電位訊號(1)，反之，當指撥開關被切換到"關"的狀態時 PA0 會接收到低電位訊號(0)，透過 PA0 所接收到的電位訊號即可了解指撥開關處於何種狀態。

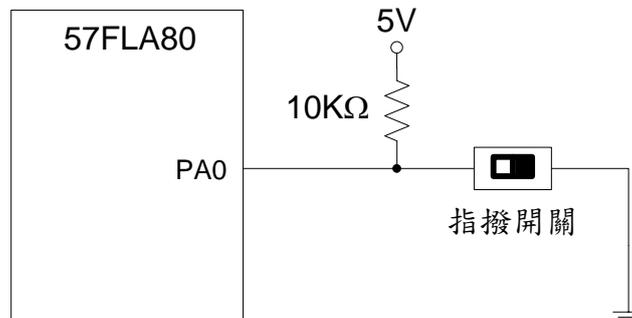


圖 5-2 指撥開關使用範例

● 七段顯示器

七段顯示器是一個可以顯示單一個文數字的電子元件，圖 5-3 所示為一個典型的七段顯示器。七段顯示器內含七個 LED(分別為 a, b, c, d, e, f, g)，每個 LED 由一個腳位(a ~ g)控制其亮暗，利用控制不同 LED 的亮暗，可產生所要的文字，例如：同時點亮 abcdef 等 LED 可產生阿拉伯數字 0，只點亮 bc 則可產生阿拉伯數字 1。

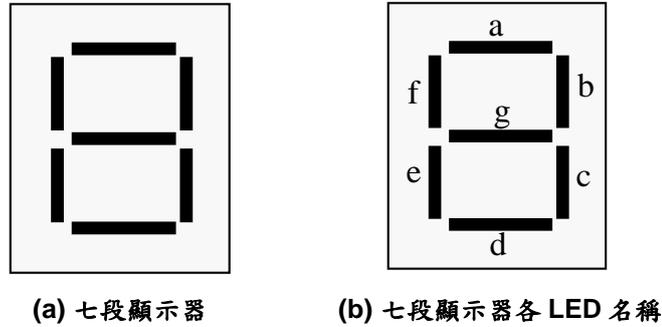


圖 5-3 七段顯示器

七段顯示器的七顆 LED 可分別由七個腳位控制，一顆 LED 有兩端，七段顯示器的七個控制訊號分別接到七顆 LED 的一端，七顆 LED 的另一端則互相接在一起。若互接的那端為 LED 的正極則該七段顯示器稱為共陽極，否則稱為共陰極。圖 5-4 為共陽極與共陰極七段顯示器的示意圖。

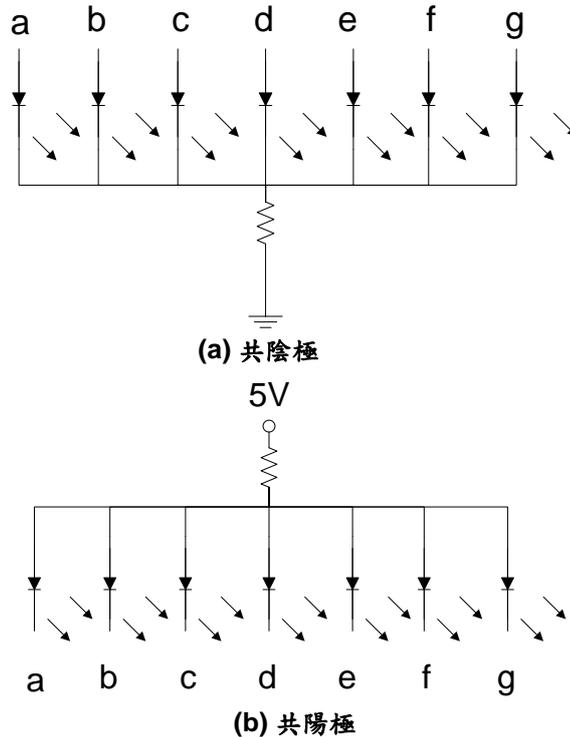


圖 5-4 共陽極與共陰極七段顯示器示意圖

由於七段顯示器通常用來顯示阿拉伯數字，因此有專用的 IC 可以接收 4 個位元的 BCD Code 並產生相對應的七段顯示器控制訊號，下表所示為 BCD Code 與七段顯示器控制訊號的對應表。

表 5-1 : BCD Code 與七段顯示器控制訊號對應表

數字	BCD 碼	共陽極	共陰極	顯示結果	
		abc defg	abc defg		
0	0000	0000001	1111110		
1	0001	1001111	0110000		
2	0010	0010010	1101101		
3	0011	0000110	1111001		
4	0100	1001100	0110011		
5	0101	0100100	1011011		
6	0110	0100000	1011111		
7	0111	0001111	1110000		
8	1000	0000000	1111111		
9	1001	0000100	1111011		

常見的 BCD 碼轉七段顯示器控制訊號的 IC 為 7447，7447 可輸出共陽極的控制訊號，其腳位圖如圖 5-5 所示。

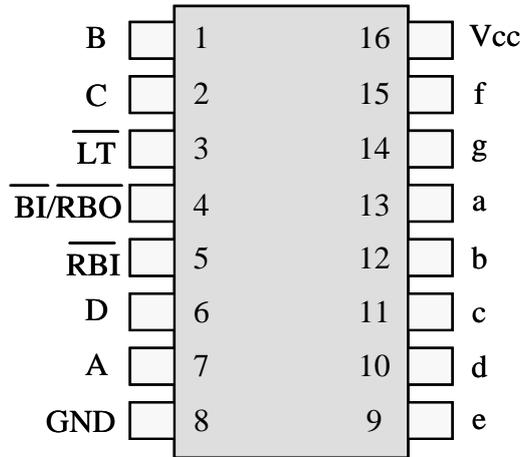


圖 5-5 7447 腳位圖

其中 DCBA (6,2,1,7) 等腳位是用來接收 BCD 碼的腳位；abcdefg (13,12,11,10,9,15,14) 等腳位是用來控制七段顯示器的控制信號；LT 腳位輸入低電位時可點亮七段顯示器的所有 LED，測試七段顯示器是否正常；RBI 與 RBO 則是當多顆 7447 同時使用時，用來串接多顆七段顯示器使用，以便將數字的前導零關閉不顯示。單顆 7447 使用時 LT, RBI, 與 RBO 都必須接高電位，才能使用。7447 的 DCBA 輸入與 abcdefg 輸出及所產生的七段顯示器顯示結果如表 5-2 所示：

表 5-2 : 7447 DCBA 輸入與 abcdefg 輸出對照表

DCBA	abcdefg	顯示結果
0000	0000001	
0001	1001111	
0010	0010010	
0011	0000110	
0100	1001100	
0101	0100100	
0110	0100000	
0111	0001111	
1000	0000000	
1001	0000100	
1010	1110010	
1011	1100110	
1100	1011100	
1101	0110100	
1110	1110000	
1111	1111111	

硬體線路圖：

本實習電路圖如圖 3-6 所示，本實習使用 57FLA80 單晶片 Port D 的 PD4 讀取按鍵狀態，使用 Port D 的 PD0, PD1, PD2, PD3 讀取指撥開關的狀態，以及使用 Port E 的 PE0, PE1, PE2, PE3 傳送 BCD 碼給 7447 IC 用來控制七段顯示器的顯示。使用時，需先用指撥開關設定要顯示在七段顯示器的 BCD 碼，然後按下按壓式按鍵，系統便會將對應於指撥開關所設定之 BCD 碼的數字顯示在七段顯示器上。

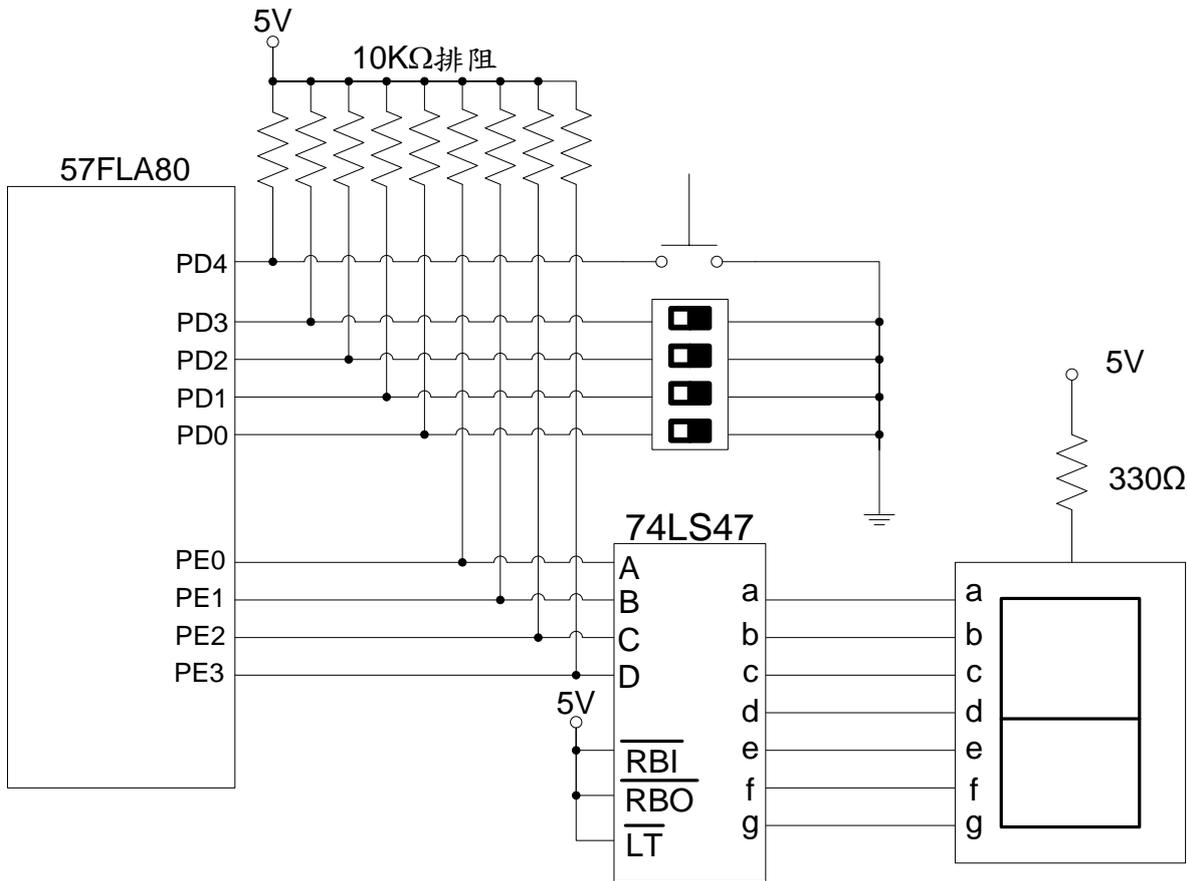


圖 3-6 指撥開關與七段顯示器控制硬體線路圖

程式流程圖：

本實習的程式流程如圖 3-7 所示，首先對七段顯示器進行初始設定，將七段顯示器顯示的內容設定為 0，接著檢查按壓式按鍵是否被按下，如果按鍵被按下，則必須先進行除彈跳動作，除彈跳動作的作法是等待 0.005 秒，等待時間過後，則再次檢查按鍵狀態，如果按鍵仍處於被按下的狀態，則透過 Port D 讀取指撥開關的 BCD 碼設定，然後將所讀取到的 BCD 碼透過 Port E 傳送給 7447，用來控制七段顯示器顯示出適當的數字。

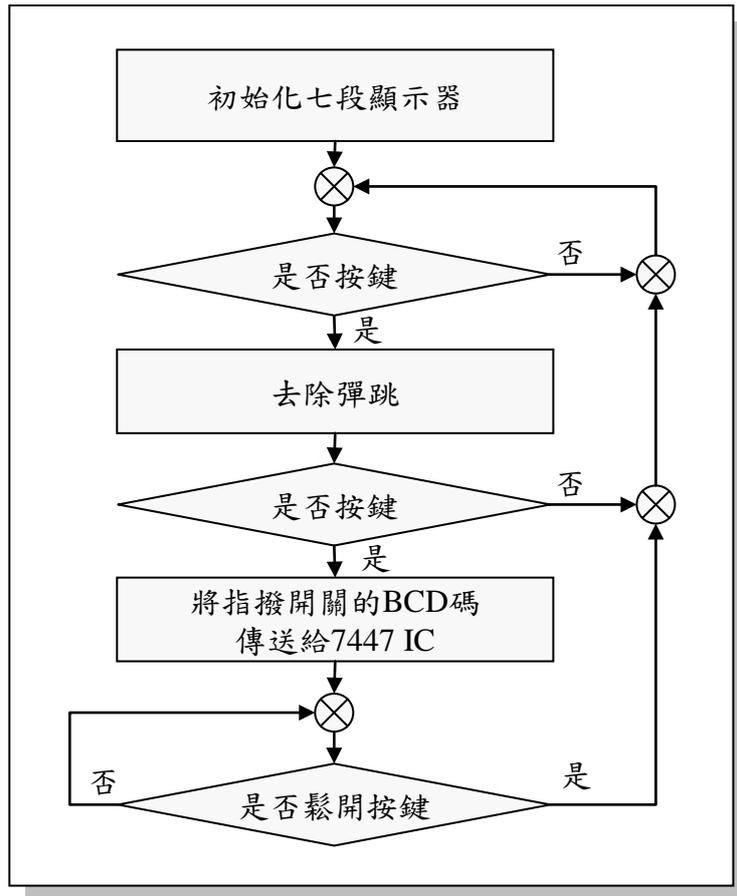


圖 3-7 指撥開關與七段顯示器控制程式流程

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址
PDD      equ      12h      ; Port D
PED      equ      13h      ; Port E

; 定義程式中使用到的F-Plane變數記憶體位址
R1       equ      20h      ; 除彈跳時間之外層迴圈控制變數
R2       equ      21h      ; 除彈跳時間之內層迴圈控制變數

; 系統開機進入點
org      00h

; 初始設定：設定七段顯示器顯示0
movlw   00h
movwf   PED

Start:   btfsc    PDD, 4    ; 檢查按鍵是否按下(檢查PD4是否為0)
         goto    Start    ; 若按鍵未被按下(PD4≠0)，回到Start

; 去除彈跳
         call    delay     ; 等待一段時間
         btfsc   PDD, 4    ; 檢查按鍵是否按下(檢查PD4是否為0)
         goto    Start    ; 若按鍵未被按下(PD4≠0)，回到Start

; 將指撥開關設定的BCD碼傳送給7447 IC
movfw   PDD      ; 讀取指撥開關設定
movwf   PED      ; 將BCD碼傳給7447 IC

; 等待使用者放開按鍵
NotRelease: btfss   PDD, 4    ; 檢查按鍵是否放開(檢查PD4是否為1)
           goto    NotRelease ; PD4≠1
           goto    Start     ; PD4=1

; 延遲0.005秒副程式
; 若clock rate=4MHz, 一個指令週期=2*clock=0.5μs,

```

```
; 要延遲0.005秒必須消耗10,000指令週期,  
; 本程式使用兩層迴圈實作時間延遲功能，時間延遲說明如下：  
; 內層迴圈消耗5指令週期，執行200次，共1,000指令週期  
; 外層迴圈執行10次，1,000指令週期*10=10,000指令週期  
  
delay:      movlw      10          ; 設定外層迴圈執行10次  
            movwf     R1          ;  
  
            ;外層迴圈  
delay_L1:   movlw     200          ; 設定內層迴圈執行200次  
            movw     fR2          ;  
  
            ;內層迴圈：迴圈跑一次約消耗5個指令週期  
delay_L2:   nop                ; 消耗1個指令週期  
            nop                ; 消耗1個指令週期  
            decfsz    R2, 1      ; 約消耗1個指令週期  
            goto     delay_L2   ; 消耗2個指令週期  
  
            decfsz    R1, 1      ; 將R1減1，若R1=0離開迴圈  
            goto     delay_L1   ;  
  
            ret                ; 返回主程式
```

## 6. 計時器實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片所提供的計時中斷功能。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	74LS245 IC	1
2	七段顯示器(共陽)	1
3	10KΩ排阻(7PIN 以上)	1
4	330Ω電阻	1

### 實習板模組與 I/O Port:

模組	I/O Port
一顆七段顯示器模組	Port E

### 實習說明：

本實習將利用單晶片所提供的計時中斷及一顆七段顯示器，產生一個單一數字的計時器，該計時器每秒鐘更新七段顯示器的內容一次，七段顯示器顯示的內容依序為 0, 1, 2, 3, 4, 5, 6, 7, 8, 9，顯示完一輪後又會從 0 開始顯示，程式會一直執行，直到關機。

定時中斷是一項很重要的功能，可供單晶片用來計時、定期檢查輸入埠狀態、或定期處理特定事項使用。使用定時中斷步驟如下：

1. 提供中斷入口：不同的中斷有不同的中斷入口，要使用中斷必須在所要使用的中斷入口放置一個 goto 指令，跳躍到該中斷的服務程式。
2. 初始化計時器：主要工作包括設定 clock 來源、設定計時中斷時間、以及啟動計時與計時中斷功能。
3. 撰寫中斷服務程式：負責處理計時中斷發生時要進行的工作。

十速 57FLA80 系列的單晶片提供 Timer 0, Timer 1, 與 Timer 2 等三組定時中斷功能，57FLA80 系列單晶片各種中斷的中斷入口如圖 6-1 所示，例如要提供 Timer 0 計時中斷，必須在程式記憶體 0001 的位置放入 goto 指令，跳到 Timer 0 的中斷服務程式；要提供 Timer 1 計時中斷，則必須在程式記憶體 0002 的位置放入 goto 指令，跳到 Timer 1 的中斷服務程式。

程式記憶體		
0000	RESET VECTOR	系統開機進入點
0001	TMR0 Interrupt	Timer 0 中斷進入點
0002	TMR1 Interrupt	Timer 1 中斷進入點
0003	TMR2 Interrupt	Timer 2 中斷進入點
0004	PWM0 Interrupt	PWM 中斷進入點
0005	WKT Interrupt	WKT 中斷進入點
0006	XINTA Interrupt	XINTA 中斷進入點
0007	XINTB Interrupt	XINTB 中斷進入點
0008	UART Interrupt	UART 中斷進入點
0009	SPI Interrupt	SPI 中斷進入點
1FFF		

圖 6-1 中斷入口記憶體對應表

要初始化計時器，必須先了解與計時器控制相關的暫存器內容，本實習使用 Timer 1 計時器功能，以下列出 Timer 1 計時器的功能方塊圖。

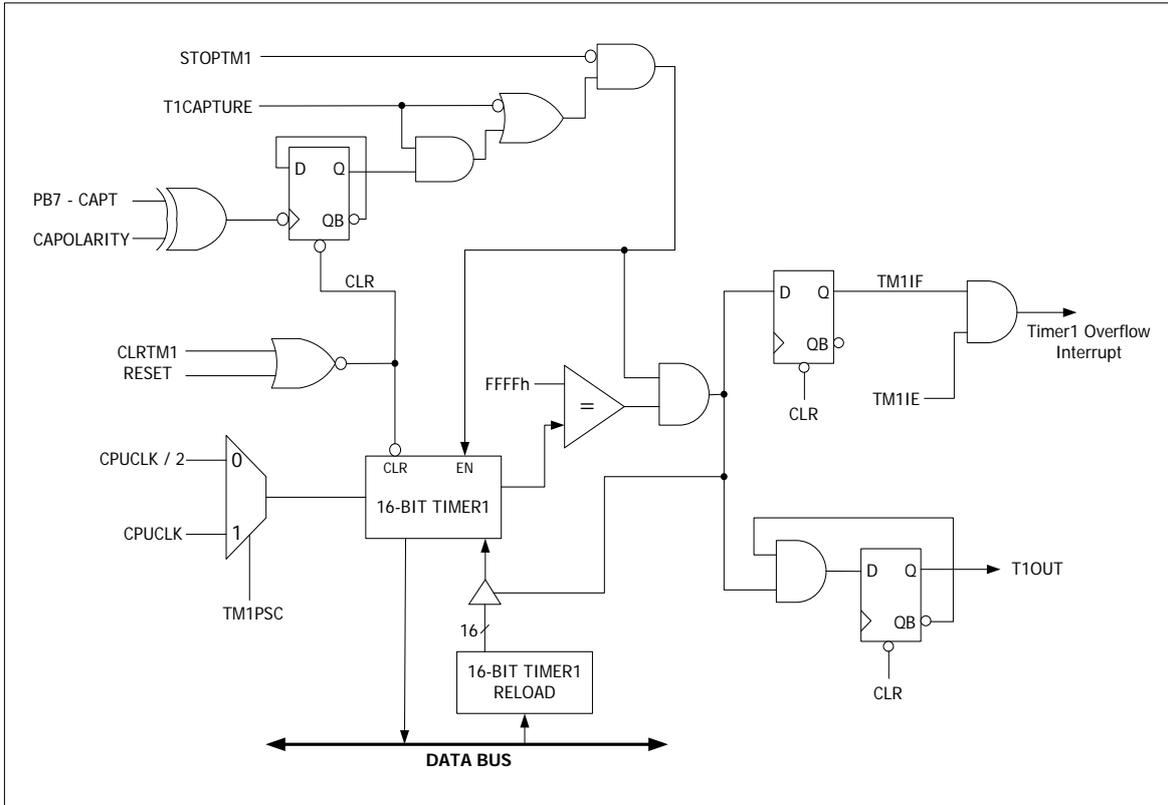


圖 6-2 Timer 1 功能方塊圖

如圖 6-2 所示，要初始化 Timer 1 計時器，計數器的初始值以及 TM1PSC、CLR TM1、STOPTM1 等幾個重要的控制暫存器必須先被初始化。Timer 1 計數器的初始值可透過 TM1\_H 與 TM1\_L 兩個暫存器給值，給定的值會儲存在 TIMER1 RELOAD(位於功能方塊圖下方)內，計時器每次溢位 (Overflow) 時，TIMER1 RELOAD 的值將會自動傳送給計時器，計時器會由該數值開始累加。累加速度由 TM1PSC 決定，有兩種速度可供選擇，一種是使用 CPU 的時脈，另一種則是使用指令時脈 (速度是 CPU 時脈的一半)。CLR TM1 與 STOPTM1 則用來控制計時器的運作狀態，要讓計時器正常運作，CLR TM1 與 STOPTM1 的值必須設定成 0。此外，為了在計時器溢位時產生中斷，必須透過 TM1IE 控制位元開啟計時器的中斷功能。Timer 1 計時器相關控制暫存器整理如下：

表 6-1 Timer 1 計時器控制暫存器

暫存器	說明
TM1IE	此控制位元位於 F-PLANE 位址 08h 暫存器位元 5，當此位元的值為 0 時，計時器雖會正常動作，但是不會對外產生中斷訊號，要讓計時器的中斷訊號正常運作，此位元必須設為 1（預設值為 0）。
TM1_L	位於 F-PLANE 位址 0ah 暫存器，儲存 Timer 1 計數器溢位後重設之計數器初始值的低位元組內容。
TM1_H	位於 F-PLANE 位址 0bh 暫存器，儲存 Timer 1 計數器溢位後重設之計數器初始值的高位元組內容。
TM1PSC	此控制位元位於 R-PLANE 位址 0ch 暫存器位元 0，用來選擇計時器的輸入時脈 (clock)，可供選擇的時脈有單晶片的工作時脈(F <sub>CPUCLK</sub> )與指令時脈(F <sub>CPUCLK</sub> /2)。當此位元的值為 1 時，選擇工作時脈，當此位元的值為 0 時，選擇指令時脈（預設值為 0）。
CLR <sub>TM1</sub>	此控制位元位於 F-PLANE 位址 0dh 暫存器位元 3，當此位元的值為 1 時，計時器處於清除狀態，計時器不會動作，要讓計時器正常動作此位元必須設定為 0（預設值為 0）。
STOPTM1	此控制位元位於 F-PLANE 位址 0dh 暫存器位元 1，當此位元的值為 1 時，計時器沒有辦法動作。要讓計時器動作，此位元必須設為 0（預設值為 0）。

硬體線路圖：

本實習硬體線路如圖 6-3 所示，本實習分別利用 57FLA80 單晶片 Port E 的 PE6 ~ PE0 等腳位直接傳送控制訊號給七段顯示器的 a ~ g 等腳位，用來控制七段顯示器的顯示內容。

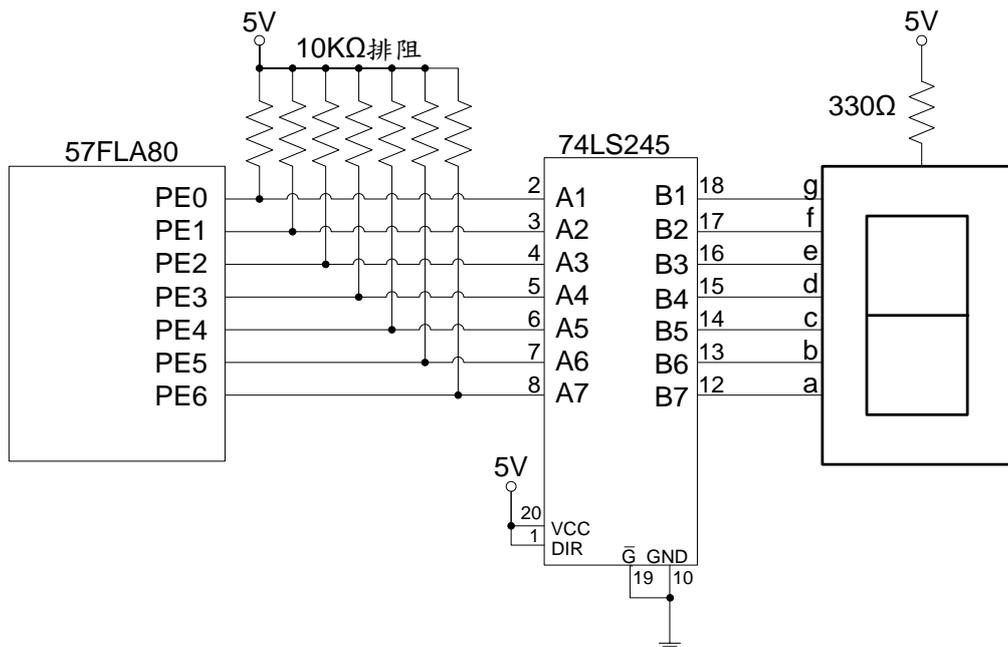


圖 6-3 計時器控制硬體線路圖

**程式流程：**

本實習使用 Timer 1 計時中斷功能達到每秒更新七段顯示器的目的，本實習程式包括主程式與計時器中斷服務程式兩個部份。以下分別說明主程式與計時器中斷服務程式的程式流程。主程式的程式流程如圖 6-4 所示。

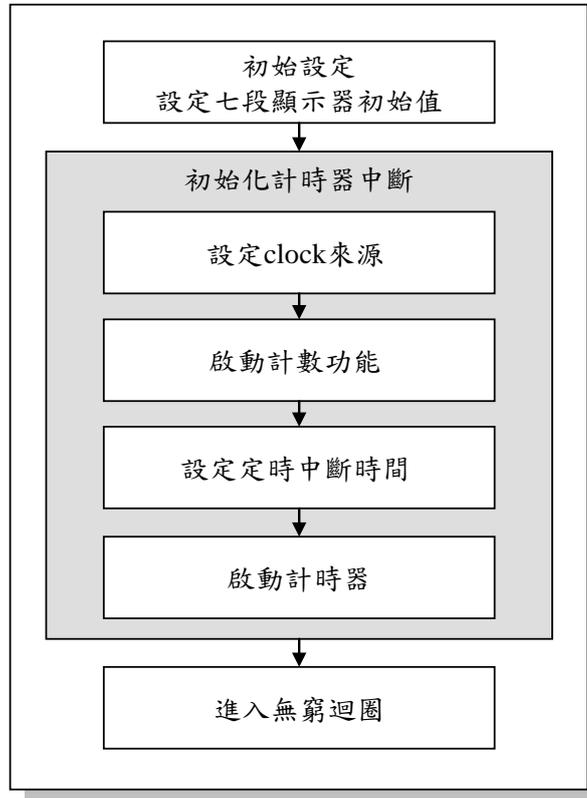


圖 6-4 計時器控制主程式流程圖

如圖 6-4 所示，主程式主要目的在初始化計時器的中斷功能，計時器中斷初始動作包括設定 clock 來源、啟動計數功能、設定定時中斷時間、以及啟動計時器等。以上工作分別說明如下：

- 設定 clock 來源

Timer 1 的 clock 來源有兩種，一種是單晶片的工作時脈，另一種是單晶片的指令時脈(指令時脈=工作時脈/2)，本實習選擇使用指令時脈，因此必須將 TM1PSC 的值設定為 0。本實習使用 4 MHz 的工作時脈，因此指令時脈為 2 MHz，一個指令時脈週期耗時 0.5  $\mu$ s。

- 啟動計數功能

將 Timer 1 的 CLRTM1 位元設為 0。

- 設定定時中斷時間

Timer 1 提供 16 位元的計數器，計數器的初始值需透過 TM1H 與 TM1L 兩個暫存器的內容載入，程式設計者給定計數器的初始值後，計數器會由該數值開始累加，直到計數器累加到 65535 時，便會產生計時中斷。本實習希望程式每秒更新七段顯示器的內容一次，假設一開始計數器為 0，則計數器累加到 65535 產生中斷，會經過 65535 個指令時脈週期，一個指令時脈週期耗時 0.5 $\mu$ s，65535 個指令時脈週期共耗時 32.7675ms。要達到每秒更新七段顯示器的內

容一次，必須等待 30.518 次中斷後，才需更新七段顯示器內容。為了方便計算時間，本程式將計數器的初始值設定為 15535，也就是中斷間隔時間為  $65535-15535=50000$  個指令時脈週期，50000 個指令時脈週期共耗時 25ms。以上可知，要每秒更新七段顯示器內容一次，須每 40 次中斷更改一次七段顯示器內容。要將計數器初始值設定為 15535，要將 15535 的高位元組給 TM1H 低位元組的值給 TM1L，因此必須設定  $TM1H=15535/256=60$ ， $TM1L=15535\%256=175$ （%表示取餘數運算）。

● 啟動計時器

計時器設定完成後，必須啟動計時器，以及啟動計時器的中斷功能。要啟動計時器，必須將 STOPTM1 位元清為 0，啟動計時器的中斷功能，則要將 TM1IE 位元設為 1。

中斷服務程式的程式流程如圖 4-5 所示。主要是用來判斷是否已經過一秒，如果不是，則直接離開中斷服務程式，程式中使用一個 count 變數，count 初始值為 40，每次進入中斷服務程式時減 1，當 counter 減到 0 時，表示已經過一秒。如果 counter 的值已減到 0，則重設 count 的值為 40，並將七段顯示器的值加 1 後顯示出來；若七段顯示器的值等於 10 表示 0 ~ 9 所有數字皆顯示過一次，則必須重設七段顯示器的值為 0 後，再顯示出來。

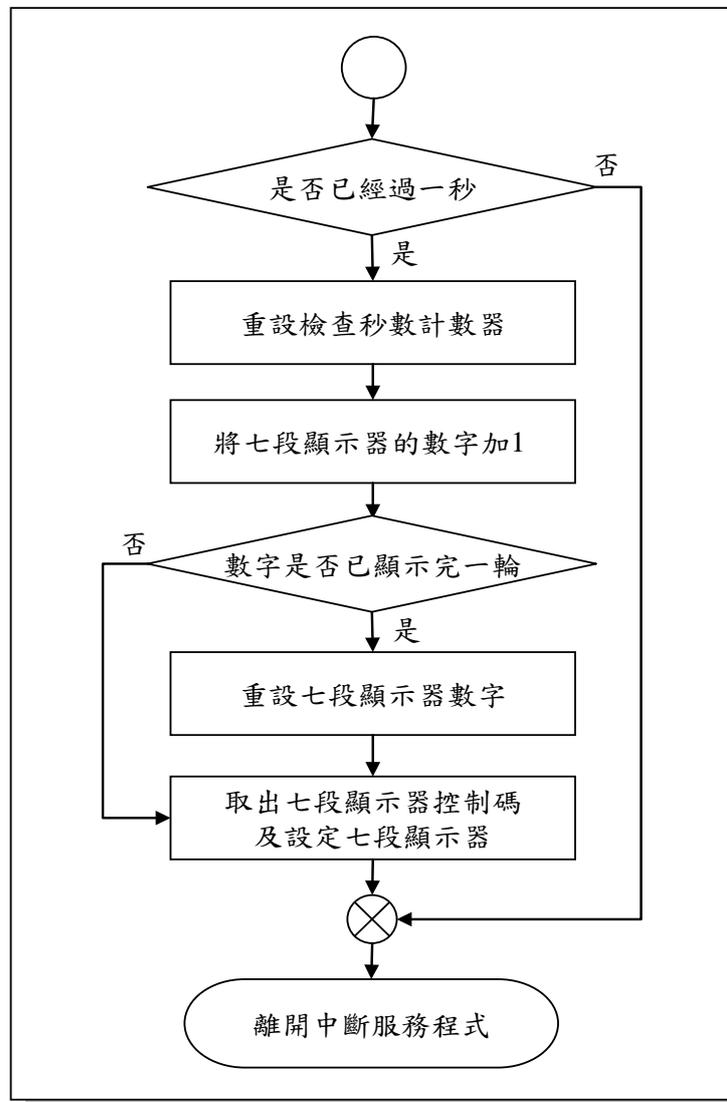


圖 6-5 計時器控制中斷程式流程圖

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址

PC          equ          02h          ; Program Counter
TM1L        equ          0ah          ; Timer 1計數器的低位元組
TM1H        equ          0bh          ; Timer 1計數器的高位元組
PED         equ          13h          ; Port E

; 定義程式中使用到的R-Plane暫存器記憶體位址

M1PSC       equ          0ch

; 定義程式中使用到的F-Plane變數記憶體位址

waitCount   equ          20h          ; 計時中斷重覆次數
digit       equ          21h          ; 七段顯示器要顯示的數字
digitLoop   equ          22h          ; 數字顯示迴圈控制變數

; 定義程式中所使用到的字串

TM1IE       defstr       08h, 5      ; Timer1中斷致能(enable)位元
; 0：不啟動Timer1中斷功能,
; 1：啟動Timer1中斷功能

TM1I        defstr       09h,5      ; Timer1中斷旗號

STOPTM1     defstr 0      dh,1      ; Timer1輸出致能位元
; 0：允許Timer1輸出中斷訊號,
; 1：不允許Timer1輸出中斷訊號

CLR TM1     defstr       0dh,3      ; Timer1計數器清除及啟動控制位元
; 0：啟動Timer1計數器,
; 1：清除Timer1計數器

; 系統開機進入點
org         00h
goto       Start          ; 跳到主程式

; Timer 1中斷進入點
org         02h

```

```

    goto    Timer1    ; 跳到Timer1 中斷服務程式

; 初始設定
Start:    movlw    10    ; 設定要顯示的數字個數
          movwf    digitLoop ; 共有0,1,2,3,4,5,6,7,8,9等十個數字

          movlw    0    ; 設定七段顯示器的初始值為0
          movwf    digit ;
          call    Table7S ; 查表取出七段顯示器控制碼
          movwf    PED   ; 將控制碼寫入Port E

          movlw    40    ; 設定一秒鐘計時中斷會中斷幾次
          movwf    waitCount ; 中斷一次25ms，中斷40次1秒

; 初始化Timer1

; 設定Timer1的clock來源
          movlw    0    ; 選擇Timer1的clock來源=clock/2
          movwr    TM1PSC ; 0表clock/2=4M/2=2M Hz

; 啟動Timer1計數功能
          bcf     CLRTM1

; 設定定時中斷時間為25ms
; 25ms : Timer1計數器須跑50000次，50000*0.5μs = 25ms
; 要跑50000次產生中斷，Timer1初始值需為65535-50000=15535
; 跑50000次後Timer1計數器會加到65535，產生中斷
          movlw    60    ; 15535/256=60
          movwf    TM1H   ; 設定Timer1計時器高位元組
          movlw    175   ; 15535%256=175
          movwf    TM1L   ; 設定Timer1計時器低位元組

; 啟動Timer1
          bsf     TM1IE   ; 開啟Timer1中斷功能
          bcf     STOPTM1 ; 允許Timer1輸出中斷訊號

          goto    $      ; 進入無窮迴圈
                          ; 由Timer1中斷程式處理後續工作

; Timer1 中斷服務程式
Timer1:decfsz    waitCount, 1 ; 將waitCount - 1

```

goto	NOT_1s	; 如果waitCount不是0表示還沒經過1秒
movlw	40	; 重設waitCount的值
movwf	waitCount	;
incf	digit, 1	; 將七段顯示器的數字加1
decfsz	digitLoop, 1	; 將迴圈控制變數減一，若減為0表示
goto	NORMAL	; 數字已顯示完一輪，需重設變數內容
RESET:movlw	10	; 重設digit與digitLoop
movwf	digitLoop	
clrf	digit	; 將digit清為0
NORMAL:movwf	digit	; 取出七段顯示器的數字
call	Table7S	; 查表取出七段顯示器控制碼
movwf	PED	; 將控制碼寫入Port E
NOT_1s: bcf	TM1I	; 清除Timer1中斷旗號，表示處理完畢
reti		; 離開中斷服務程式
		; 七段顯示器控制碼表格
Table7S: addwf	PC, 1	; 依工作暫存器的值傳回適當控制碼
retlw	0000001b	; 七段顯示器控制碼, 0
retlw	1001111b	; 七段顯示器控制碼, 1
retlw	0010010b	; 七段顯示器控制碼, 2
retlw	0000110b	; 七段顯示器控制碼, 3
retlw	1001100b	; 七段顯示器控制碼, 4
retlw	0100100b	; 七段顯示器控制碼, 5
retlw	1100000b	; 七段顯示器控制碼, 6
retlw	0001111b	; 七段顯示器控制碼, 7
retlw	0000000b	; 七段顯示器控制碼, 8
retlw	0000100b	; 七段顯示器控制碼, 9

## 7. 數位電子鐘實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片製作一個小時的數位電子鐘。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	74LS245 IC	1
2	74LS47 IC	1
3	七段顯示器(共陽)	4
4	10KΩ排阻(A-type 9PIN)	1
5	4.7KΩ電阻	4
6	330Ω電阻	4
7	A1015 電晶體	4

### 實習板模組與 I/O Port：

模組	I/O Port
四顆七段顯示器模組	Port E

### 實習說明：

本實習主要目的在利用十速 57FLA80 系列單晶片所提供的 Timer 1 計時中斷以及四顆七段顯示器，實作一個可顯示 00 分 00 秒到 59 分 59 秒的數位電子鐘。Timer 1 計時器的使用如第七章所述，本實習使用 Timer 1 計時器產生時間間隔為 25 ms 的定時中斷，每經過 40 次中斷(一秒)更新一次時間。七段顯示器的數字由 00 分 00 秒開始，每經過 1 秒，七段顯示器上的秒數便會往上加 1，加到 60 秒則秒數歸零，分鐘數加 1，若分鐘數已加到 60，則分鐘數也會歸零。圖 7-1 列出數位電子鐘的顯示方式。

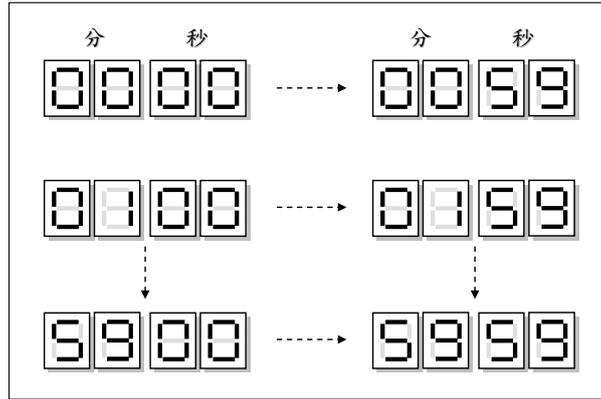


圖 7-1 數位電子鐘顯示方式

硬體線路圖：

本實習電路圖如圖 7-2 所示，本實習使用 57FLA80 單晶片 Port E 的 PE3 ~ PE0 傳送 BCD 碼給 74LS47，用來控制七段顯示器的顯示內容，以及使用 PE4 ~ PE7 等 4 支腳位分別控制數位電子鐘的秒數個位數、秒數十位數、分數個位數、與分數十位數等七段顯示器的亮暗。為了提供足夠的電流驅動七段顯示器，PE4 ~ PE7 的輸出用來作為 PNP 電晶體的開關，當 PE4 ~ PE7 輸出低電位時，電晶體會導通並點亮七段顯示器，否則，電晶體不導通，七段顯示器也不會被點亮。

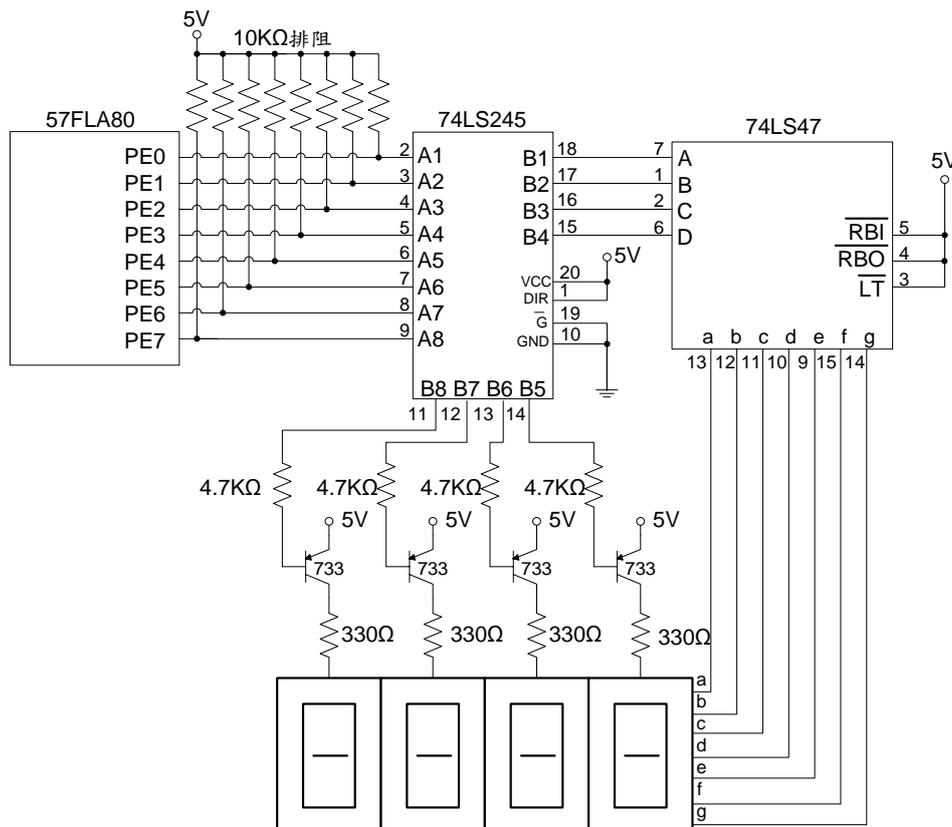


圖 7-2 數位電子鐘硬體線路圖

程式流程：

本實習主要工作包括兩個部份，分別是更新時間以及顯示時間。其中，更新時間部份必須利用計時中斷計算時間，本實習使用 Timer 1 計時中斷，因此在 Timer 1 計時中斷服務程式內更新時間，Timer 1 計時中斷每 25ms 中斷一次，每經過 40 次中斷(一秒)更新一次時間。顯示時間工作在主程式進行，為了節省硬體成本，本實習只用 4 個位元傳送 BCD 碼以及使用一個 74LS47 傳送七段顯示器的控制訊號，要顯示四個七段顯示器的內容，必須使用掃描方式進行，例如，顯示時間時，依序顯示秒數個位數、秒數十位數、分數個位數、與分數十位數，每顯示完一個數字後必須等候一段足夠時間，再顯示下個數字，所有數字顯示完後，則從頭開始顯示。當掃描速度夠快且等候時間足夠時，由於人類的視覺暫留現象，便會覺得四個七段顯示器都有數字顯示出來。

以下分別列出主程式與計時器中斷服務程式的程式流程。主程式的程式流程如圖 7-3 所示，主要工作包括計時器的初始化以及顯示時間，其中計時器初始化動作請參考第六章的說明。

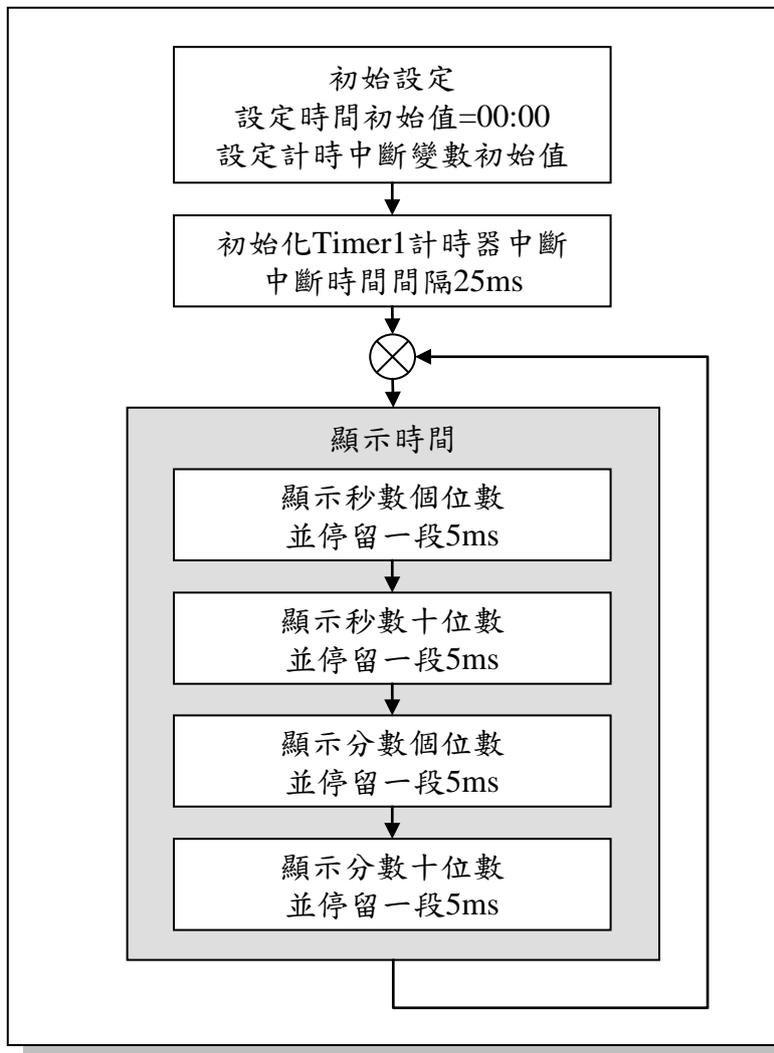


圖 7-3 數位電子鐘主程式流程圖

中斷服務程式的程式流程如圖 7-4 所示。主要是用來更新時間。

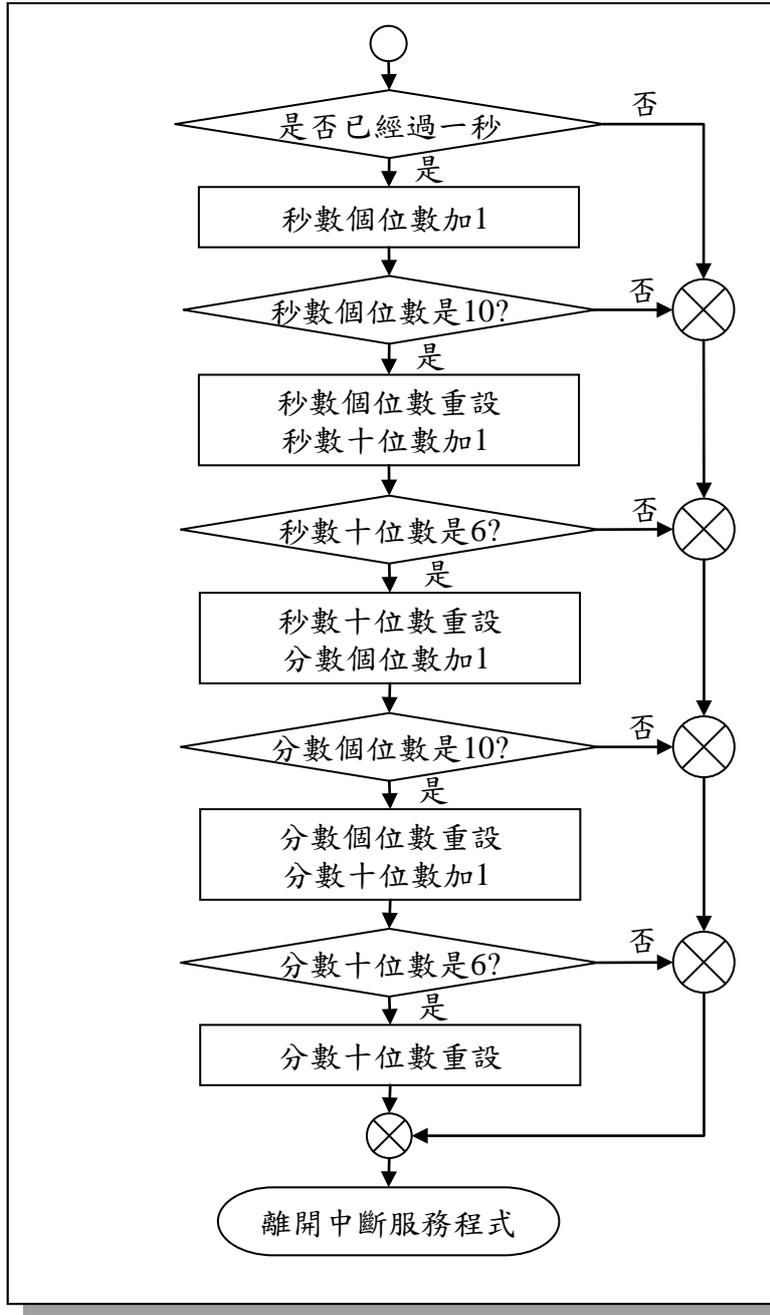


圖 7-4 數位電子鐘控制中斷程式流程圖

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址

PC          equ          02h          ; Program Counter
TM1L        equ          0ah          ; Timer 1計數器的低位元組
TM1H        equ          0bh          ; Timer 1計數器的高位元組
PED         equ          13h          ; Port E

; 定義程式中使用到的R-Plane暫存器記憶體位址

TM1PSC      equ          0ch

; 定義程式中所使用到的字串

TM1IE       defstr       08h, 5      ; Timer1中斷致能(enable)位元
; 0：不啟動Timer1中斷功能,
; 1：啟動Timer1中斷功能

TM1I        defstr       09h,5      ; Timer1中斷旗號

STOPTM1     defstr       0dh,1      ; Timer1輸出致能位元
; 0：允許Timer1輸出中斷訊號,
; 1：不允許Timer1輸出中斷訊號

CLR TM1     defstr       0dh,3      ; Timer1計數器清除及啟動控制位元
; 0：啟動Timer1計數器,
; 1：清除Timer1計數器

; 定義程式中使用到的F-Plane變數記憶體位址

waitCount   equ          20h          ; 計時中斷重覆次數
digitS0     equ          21h          ; 秒數個位數
digitS1     equ          22h          ; 秒數十位數
digitM0     equ          23h          ; 分數個位數
digitM1     equ          24h          ; 分數十位數
R1          equ          25h          ; 時間延遲迴圈控制變數
R2          equ          26h          ; 時間延遲迴圈控制變數

; 系統開機進入點
org         00h
goto       Start          ; 跳到主程式

```

```

; Timer 1 中斷進入點
org      02h
goto    Timer1      ; 跳到Timer1 中斷服務程式

; 初始設定

; 設定時間初始值為00:00
Start:   movlw      0          ; 設定七段顯示器的初始值為0
         movwf     digitS0     ; 設定秒數個位數初始值為為0
         movwf     digitS1     ; 設定秒數十位數初始值為為0
         movwf     digitM0     ; 設定分數個位數初始值為為0
         movwf     digitM1     ; 設定分數十位數初始值為為0

; 設定定時中斷變數初始值
         movlw     00000000b    ; 將七段顯示器內容清為0
         movwf     PED         ;

         movlw     40           ; 設定一秒鐘計時中斷會中斷幾次
         movwf     waitCount   ;

         movlw     250         ; 設定分鐘十位數初始值為-6
         movwf     digitM1     ; 分鐘十位數範圍為(-6)~(-1)
         movlw     246         ; 設定分鐘個位數初始值為-10
         movwf     digitM0     ; 分鐘個位數範圍為(-10)~(-1)
         movlw     250         ; 設定秒數十位數初始值為-6
         movwf     digitS1     ; 分鐘十位數範圍為(-6)~(-1)
         movlw     246         ; 設定秒數個位數初始值為-10
         movwf     digitS0     ; 分鐘個位數範圍為(-10)~(-1)

; 初始化Timer1

; 設定Timer1的clock來源
         movlw     0           ; 選擇Timer1的clock來源=clock/2
         movwr     TM1PSC     ; 0表clock/2=4M/2=2M Hz

; 啟動Timer1計數功能
         bcf       CLRTM1

; 設定定時中斷時間為25ms
         movlw     60          ; 15535/256=60
         movwf     TM1H        ; 設定Timer1計時器高位元組
         movlw     175         ; 15535%256=175
         movwf     TM1L        ; 設定Timer1計時器低位元組

; 啟動Timer1
         bsf       TM1IE       ; 開啟Timer1 中斷功能
         bcf       STOPTM1     ; 允許Timer1輸出中斷訊號

```

```

; 顯示時間
ShowTime: movfw    digitS0    ; digitS0範圍加10可得秒數個位數
          addlw    10          ;
          iorlw    11110000b   ; 將要顯示數字放入Port E的bits 3 ~ bits 0
          movwf   PED         ; 以及關掉所有七段顯示器(bits 7 ~ bits 4)
          bcf     PED,4       ; 點亮秒數個位數的七段顯示器
          call    delay       ; 等待足夠形成視覺暫留的時間

          movfw    digitS1    ; 將digitS1+6可得秒數十位數
          addlw    6          ;
          iorlw    11110000b   ; 將要顯示數字放入Port E的bits 3 ~ bits 0
          movwf   PED         ; 以及關掉所有七段顯示器(bits 7 ~ bits 4)
          bcf     PED,5       ; 點亮秒數十位數的七段顯示器
          call    delay       ; 等待足夠形成視覺暫留的時間

          movfw    digitM0    ; 將digitM0+10可得分鐘個位數
          addlw    10         ;
          iorlw    11110000b   ; 將要顯示數字放入Port E的bits 3 ~ bits 0
          movwf   PED         ; 以及關掉所有七段顯示器(bits 7 ~ bits 4)
          bcf     PED,6       ; 點亮分鐘個位數的七段顯示器
          call    delay       ; 等待足夠形成視覺暫留的時間

          movfw    digitM1    ; 將digitM1+6可得分鐘個位數
          addlw    6          ;
          iorlw    11110000b   ; 將要顯示數字放入Port E的bits 3 ~ bits 0
          movwf   PED         ; 以及關掉所有七段顯示器(bits 7 ~ bits 4)
          bcf     PED,7       ; 點亮分鐘十位數的七段顯示器
          call    delay       ; 等待足夠形成視覺暫留的時間

          goto    ShowTime    ; 重覆顯示時間

; 延遲0.005秒副程式
; 若clock rate=4MHz, 一個指令週期=2*clock=0.5μs,
; 要延遲0.005秒必須消耗10,000指令週期,
; 本程式使用兩層迴圈實作時間延遲功能, 時間延遲說明如下:
; 內層迴圈消耗5指令週期, 執行200次, 共1,000指令週期
; 外層迴圈執行10次, 1,000指令週期*10=10,000指令週期

delay:    movlw    10          ; 設定外層迴圈執行10次
          movwf   R1          ;
          ;外層迴圈
delay_L1: movlw    200        ; 設定內層迴圈執行200次
          movwf   R2          ;
          ;內層迴圈: 迴圈跑一次約消耗5個指令週期
delay_L2: nop              ; 消耗1個指令週期
          nop              ; 消耗1個指令週期
          decfsz  R2, 1      ; 約消耗1個指令週期

```

goto	delay_L2	; 消耗2個指令週期
decfsz	R1,1	; 將R1減1，若R1=0離開迴圈
goto	delay_L1	;
ret		; 返回主程式
; Timer1 中斷服務程式		
Timer1:	decfsz waitCount, 1	; 將waitCount - 1
	goto NOT_1s	; waitCount不是0表示還沒經過1秒
movlw	40	; 重設waitCount的值
movwf	waitCount	;
incfsz	digitS0, 1	; 秒數個位數加1
goto	NOT_1s	; 秒數個位數≠0, 離開中斷
movlw	246	; 重設秒數個位數值為-10
movwf	digitS0	;
incfsz	digitS1, 1	; 秒數十位數加1
goto	NOT_1s	; 秒數十位數≠0, 離開中斷
movlw	250	; 重設秒數十位數值為-6
movwf	digitS1	;
incfsz	digitM0, 1	; 分數個位數加1
goto	NOT_1s	; 分數個位數≠0, 離開中斷
movlw	246	; 重設分數個位數值為-10
movwf	digitM0	;
incfsz	digitM1, 1	; 分數十位數加1
goto	NOT_1s	; 分數十位數≠0, 離開中斷
movlw	250	; 重設分數十位數值為-6
movwf	digitM1	;
NOT_1s:	bcf M1I	; 清除Timer1中斷旗號，表示處理完畢
	reti	; 離開中斷服務程式

**8. 4 × 4 鍵盤實習****實習目的：**

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片讀取 4×4 鍵盤的輸入。

**實習設備：**

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

**實習材料：**

項次	品名	數量
1	74LS245 IC	1
2	74LS47 IC	1
3	4 × 4 鍵盤	1
4	20KΩ 排阻 (A-type 9PIN)	3
5	七段顯示器 (共陽)	2
6	4.7KΩ 電阻	2
7	330Ω 電阻	2
8	A1015 電晶體	2

**實習板模組與 I/O Port：**

模組	I/O Port
4 × 4 鍵盤模組	Port D
四顆七段顯示器模組	Port E

實習說明：

本實習主要目的在利用十速 57FLA80 系列單晶片讀取 4 × 4 鍵盤的輸入，4 × 4 鍵盤外型及內部結構圖如圖 8-1 所示。

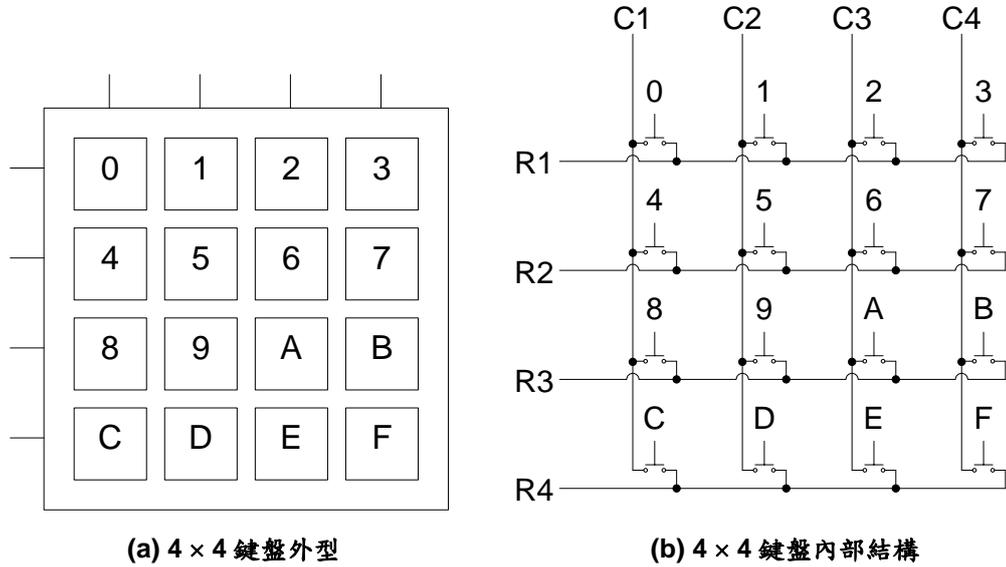


圖 8-1 4 × 4 鍵盤外型及內部結構圖

如圖 8-1 所示，4 × 4 鍵盤共有 16 個按鍵，有八條控制線，包括 4 條列控制線( R1 ~ R4) 與 4 條行控制線( C1 ~ C4 )。使用 4 × 4 鍵盤時，必須將 4 × 4 鍵盤的列控制線與行控制線接到具讀寫功能的 I/O 埠。圖 8-2 所示是將 4 × 4 鍵盤接到 57FLA80 單晶片 Port D 的例子。

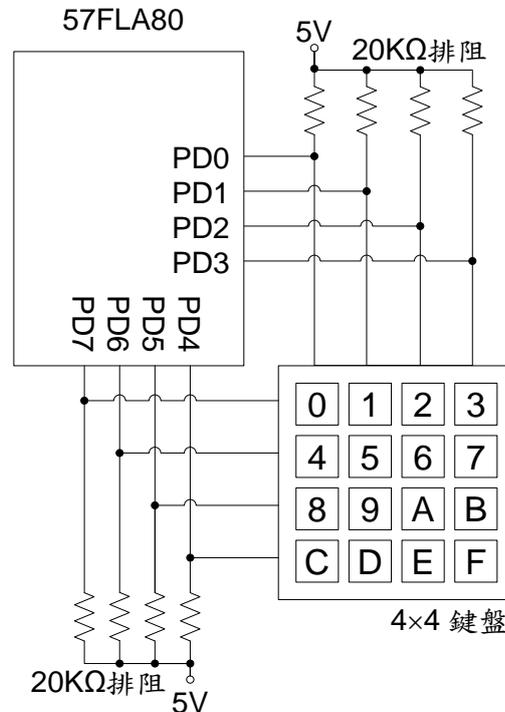


圖 8-2 4 × 4 鍵盤接線圖

要判斷  $4 \times 4$  鍵盤有沒有按鍵被按下，可透過 I/O 埠將列控制線與行控制線設定成不同電位，例如將列控制線設定成低電位(0)以及將行控制線設定成高電位(1)，然後再讀取 I/O 埠的內容，如果所有按鍵都沒有被按下，則列控制線與行控制線沒有接觸，列控制線與行控制線的電位不會改變，若有任何按鍵被按下，高低電位接觸，則原本高電位的控制線會變成低電位，例如，若按鍵 9 被按下，則行控制線 C2 的電位會由高電位變成低電位，藉由檢查 I/O 埠的內容有無被改變，可知道有無按鍵被按下。圖 8-3 所示的程式碼，是以圖 8-2 的接線方式為例，判斷  $4 \times 4$  鍵盤有沒有被按下的程式碼。

```

; 檢查4x4鍵盤有沒有被按下
movlw  ffh          ; 將PD0~PD7設為1, 清除按鍵狀態
movwf  PDD          ;
movlw  0fh          ; 將鍵盤列位址(bits4~7)設成低電位
                    ; 將鍵盤行位址(bits0~3)設成高電位
movwf  PDD          ; 將控制碼寫入Port D
movfw  PDD          ; 讀取鍵盤狀態, 沒按鍵應是0FH
addlw  f0h          ; +F0H
movwf  keyStatus    ; 取得按鍵狀態, 沒按鍵是FFH
incfsz keyStatus, 1 ; 若按鍵狀態+1等於0, 表示沒按鍵
goto   KeyPressed   ; 有按鍵, 到KeyPressed處理按鍵
goto   KeyNotPressed ; 沒按鍵, 到KeyNotPressed

```

圖 8-3 檢查  $4 \times 4$  鍵盤按鍵程式碼

圖 8-3 只能知道  $4 \times 4$  鍵盤有無按鍵被按下，無法判斷哪個按鍵被按下，要判斷哪個按鍵被按下，必須使用掃描方式，一一列檢查。檢查一列時，可將要檢查的列控制線設定成低電位其餘列控制線設定成高電位，所有行控制線也設定成高電位，然後再讀取所有控制線的內容，如果所有控制線的內容有改變，則表示要檢查的列有按鍵被按下，接下來只要判斷哪個行控制線變成低電位即可知道哪個按鍵被按下。以圖 8-2 的接線方式為例，要檢查第一列是否有按鍵被按下，可將  $11101111b=EFH$  寫入 Port D，然後再讀入 Port D 的內容做檢查，此時若按鍵‘2’被按下，則讀入的 Port D 內容會變成  $11101011b$ ，測試 Port D 的 bit 2 是否為 0，便可知道按鍵‘2’被按下。

硬體線路圖：

本實習電路圖如圖 8-4 所示，本實習使用 57FLA80 單晶片 Port E 來控制七段顯示器的顯示，其中 PE3 ~ PE0 用來傳送 BCD 碼，PE4 與 PE5 分別用來控制數字一與數字二的顯示，PE4 與 PE5 的值如果是 0 (低電位)，會點亮相對應的七段顯示器，否則相對應的七段顯示器不會亮。4 × 4 鍵盤的控制則透過 Port D 來達成，其中，PD7 ~ PD4 分別對應到鍵盤的 1 ~ 4 列，PD0 ~ PD3 則分別對應到鍵盤的 1 ~ 4 行。

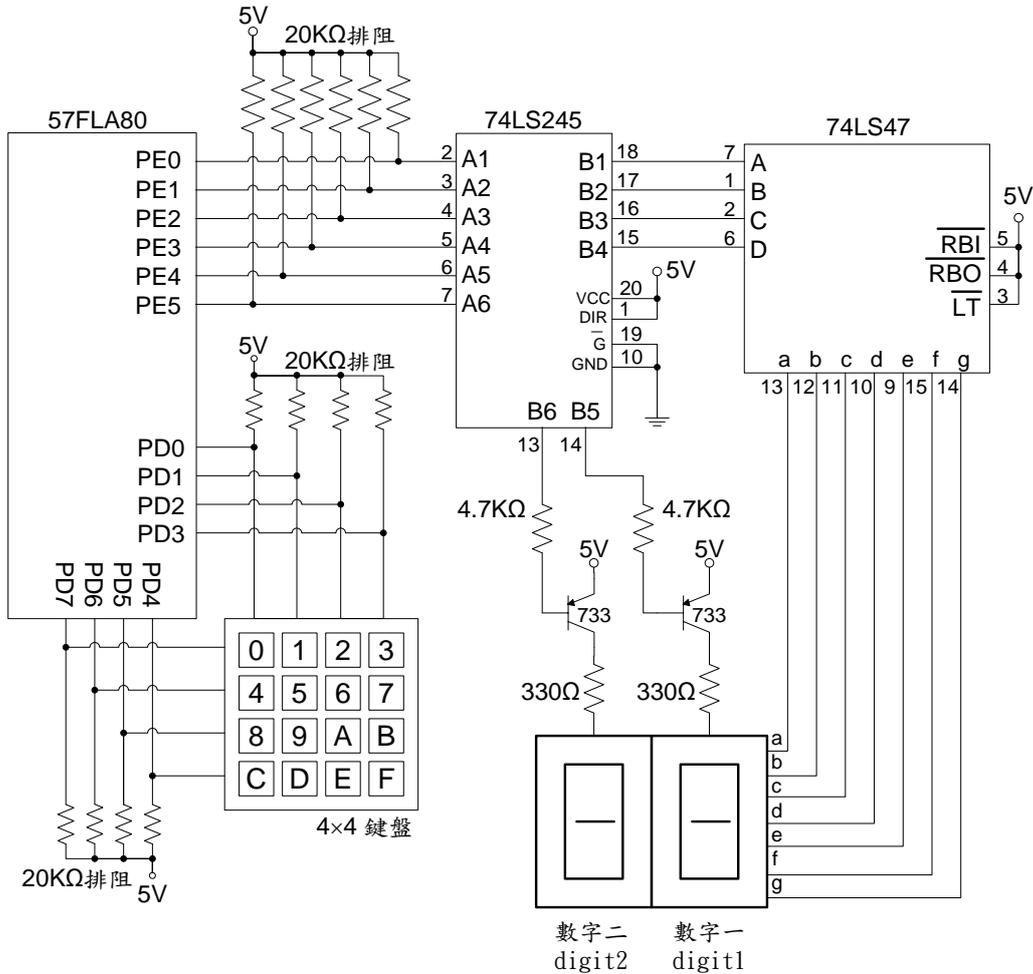


圖 8-4 4 × 4 鍵盤硬體線路圖

程式流程：

本實習主要工作在將按鍵代碼顯示在七段顯示器上，並檢查使用者是否有按下任何按鍵，若使用者按下按鍵則將七段顯示器的內容改成被按下按鍵的代碼，然後等待使用者放開按鍵。主程式流程如圖 8-5 所示。

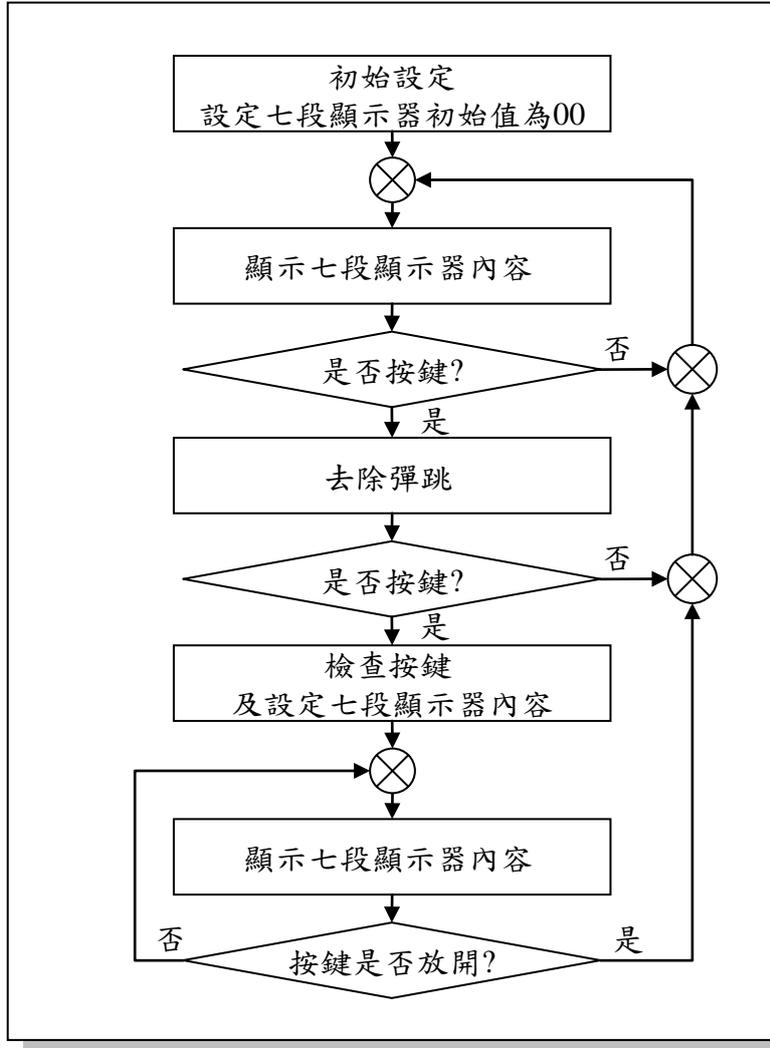


圖 8-5 4 × 4 鍵盤主程式流程圖

主程式流程中，七段顯示器內容的顯示方式使用如第五章所示的掃描方式，去除彈跳方式則使用如第三章所示的一般按鍵去彈跳方式。流程圖中"檢查按鍵及設定七段顯示器內容"方塊在逐列檢查使用者按了什麼鍵，並將使用者所按的鍵的代碼記錄起來，供後續顯示在七段顯示器上，其詳細流程如圖 8-6 所示。

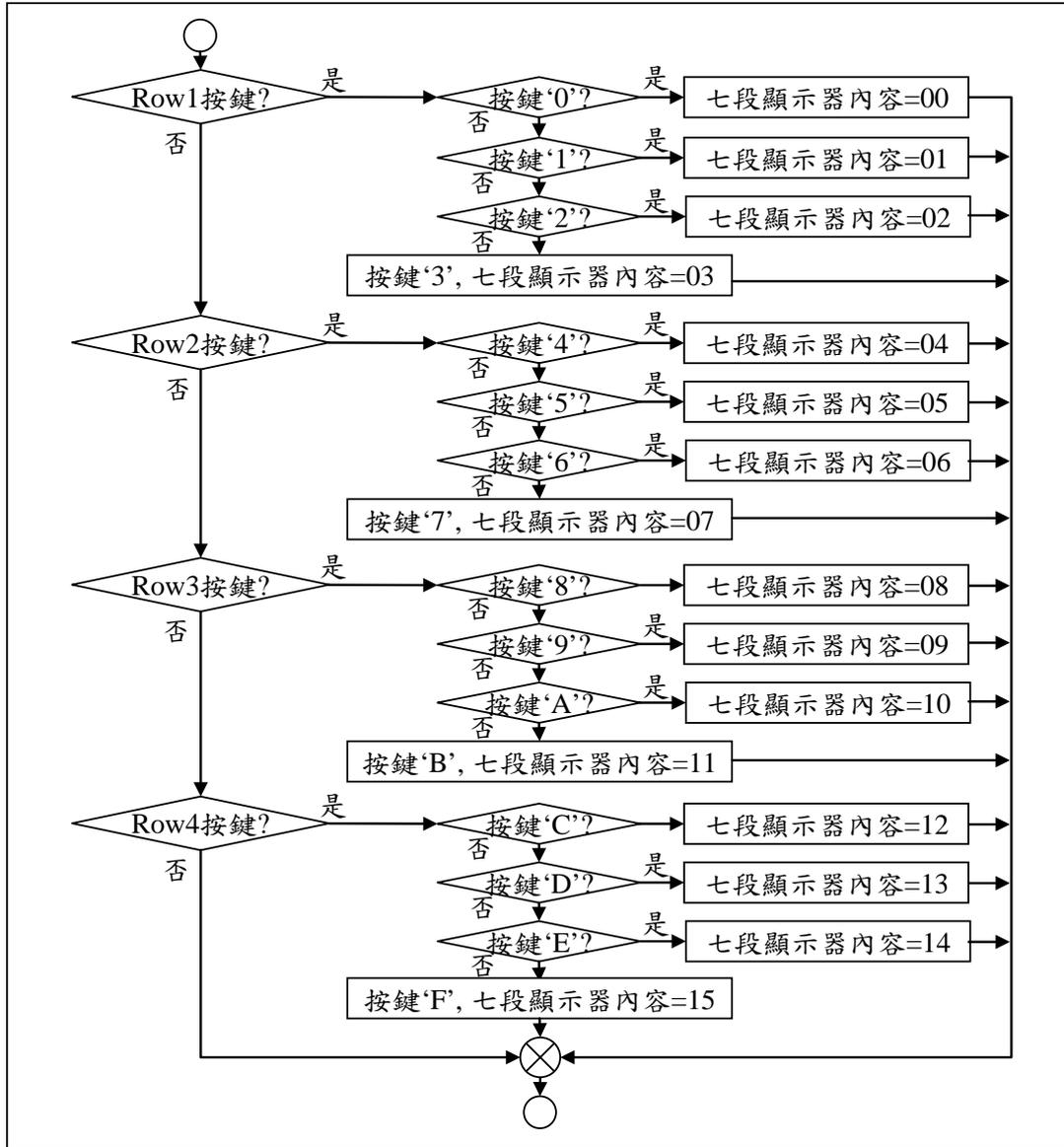


圖 8-6 按鍵檢查程式流程圖

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址
PDD      equ    12h      ; Port D
PED      equ    13h      ; Port E

; 定義程式中使用到的R-Plane暫存器記憶體位址
PDE      equ    12h      ; Port D Push-Pull enable

; 定義程式中使用到的F-Plane變數記憶體位址
digit1   equ    20h      ; 七段顯示器數字一
digit2   equ    21h      ; 七段顯示器數字二
keyStatus equ    22h      ; 按鍵狀態
R1       equ    23h      ; 時間延遲迴圈控制變數
R2       equ    24h      ; 時間延遲迴圈控制變數

; 系統開機進入點
org      00h

; 設定七段顯示器初始值
movlw   f0h      ; 關掉七段顯示器並設定BCD碼為0
movwf   digit1   ; 設定七段顯示器數字一初始值為0
movwf   digit2   ; 設定七段顯示器數字二初始值為0

; 致能鍵盤輸入腳位Push-pull功能
movlw   ffh
movwrr PDE

; 顯示七段顯示器內容
Start:   call    ShowDigits

; 檢查是否有按鍵被按下
CheckKey: movlw   ffh      ;
          movwf   PDD      ; 清除按鍵狀態
          movlw   0fh      ; 將鍵盤列位址(bits4~7)變成低電位
          movwf   PDD      ; 將鍵盤行位址(bits0~3)變成高電位
          movfw   PDD      ; 讀取鍵盤狀態，沒按鍵應是0FH
          addlw   f0h      ; +F0H
          movwf   keyStatus ; 取得按鍵狀態，沒按鍵是0FH+F0H=FFH
          incfsz  keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
          goto   DeNoise   ; 有按鍵，跳到DeNoise去除彈跳
          goto   Start     ; 沒按鍵，回到Start

; 去除彈跳
DeNoise: call    delay     ; 等待5ms

```

	<pre> ; 檢查是否有按鍵被按下 movlw ffh          ; movwf PDD          ; 清除按鍵狀態 movlw 0fh          ; 將鍵盤列位址(bits4~7)變成低電位 movwf PDD          ; 將鍵盤行位址(bits0~3)變成高電位 movwf PDD          ; 讀取鍵盤狀態，沒按鍵應是0FH addlw f0h          ; +F0H movwf keyStatus    ; 取得按鍵狀態，沒按鍵是FFH incfsz keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵 goto Row1          ; 有按鍵，到Row1檢查按了什麼鍵 goto Start         ; 沒按鍵，回到Start </pre>
Row1:	<pre> ; 檢查第一列是否有按鍵被按下 movlw ffh          ; movwf PDD          ; 清除按鍵狀態 movlw efh          ; 將鍵盤第一列(bit 4)變成低電位 movwf PDD          ; movwf PDD          ; 讀取鍵盤狀態，沒按鍵應是efh addlw 10h          ; +10h movwf keyStatus    ; 取得按鍵狀態，沒按鍵應是ffh incfsz keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵 goto Key0          ; 有按鍵，檢查是否按了'0' goto Row2          ; 沒按鍵，檢查第二列 </pre>
Key0:	<pre> btfsc PDD, 0       ; 檢查'0'是否被按下 goto Key1          ; '0'沒被按下，檢查是否按了'1' movlw f0h          ; '0'被按下，設定七段顯示器為00 movwf digit1       ; 設定七段顯示器的第一個數字為0 movwf digit2       ; 設定七段顯示器的第二個數字為0 goto WaitRelease   ; 等待按鍵放開 </pre>
Key1:	<pre> btfsc PDD, 1       ; 檢查'1'是否被按下 goto Key2          ; '1'沒被按下，檢查是否按了'2' movlw f1h          ; '1'被按下，設定七段顯示器為01 movwf digit1       ; 設定七段顯示器的第一個數字為1 movlw f0h          ; movwf digit2       ; 設定七段顯示器的第二個數字為0 goto WaitRelease   ; 等待按鍵放開 </pre>
Key2:	<pre> btfsc PDD, 2       ; 檢查'2'是否被按下 goto Key3          ; '2'沒被按下，所以是'3'被按下 movlw f2h          ; '2'被按下，設定七段顯示器為02 movwf digit1       ; 設定七段顯示器的第一個數字為2 movlw f0h          ; movwf digit2       ; 設定七段顯示器的第二個數字為0 goto WaitRelease   ; 等待按鍵放開 </pre>

Key3:	movlw f3h ; '3'被按下，設定七段顯示器為03
	movwf digit1 ; 設定七段顯示器的第一個數字為3
	movlw f0h ;
	movwf digit2 ; 設定七段顯示器的第二個數字為0
	goto WaitRelease ; 等待按鍵放開
Row2:	; 檢查第二列是否有按鍵被按下
	movlw ffh ;
	movwf PDD ; 清除按鍵狀態
	movlw dfh ; 將鍵盤第二列(bit 5)變成低電位
	movwf PDD ;
	movwf PDD ; 讀取鍵盤狀態，沒按鍵應是dfh
	addlw 20h ; +20h
	movwf keyStatus ; 取得按鍵狀態，沒按鍵應是ffh
	incfsz keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
	goto Key4 ; 有按鍵，檢查是否按了'4'
	goto Row3 ; 沒按鍵，檢查第三列
Key4:	btfsz PDD, 0 ; 檢查'4'是否被按下
	goto Key5 ; '4'沒被按下，檢查是否按了'5'
	movlw f4h ; '4'被按下，設定七段顯示器為04
	movwf digit1 ; 設定七段顯示器的第一個數字為4
	movlw f0h ;
	movwf digit2 ; 設定七段顯示器的第二個數字為0
	goto WaitRelease ; 等待按鍵放開
Key5:	btfsz PDD, 1 ; 檢查'5'是否被按下
	goto Key6 ; '5'沒被按下，檢查是否按了'6'
	movlw f5h ; '5'被按下，設定七段顯示器為05
	movwf digit1 ; 設定七段顯示器的第一個數字為5
	movlw f0h ;
	movwf digit2 ; 設定七段顯示器的第二個數字為0
	goto WaitRelease ; 等待按鍵放開
Key6:	btfsz PDD, 2 ; 檢查'6'是否被按下
	goto Key7 ; '6'沒被按下，所以是'7'被按下
	movlw f6h ; '6'被按下，設定七段顯示器為06
	movwf digit1 ; 設定七段顯示器的第一個數字為6
	movlw f0h ;
	movwf digit2 ; 設定七段顯示器的第二個數字為0
	goto WaitRelease ; 等待按鍵放開
Key7:	movlw f7h ; '7'被按下，設定七段顯示器為07
	movwf digit1 ; 設定七段顯示器的第一個數字為7
	movlw f0h ;
	movwf digit2 ; 設定七段顯示器的第二個數字為0
	goto WaitRelease ; 等待按鍵放開

Row3:	;	檢查第三列是否有按鍵被按下	
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	bfh	; 將鍵盤第三列(bit 6)變成低電位
	movwf	PDD	;
	movfw	PDD	; 讀取鍵盤狀態，沒按鍵應是bfh
	addlw	40h	; +40h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	Key8	; 有按鍵，檢查是否按了'8'
	goto	Row4	; 沒按鍵，檢查第四列
Key8:	btfsf	PDD, 0	; 檢查'8'是否被按下
	goto	Key9	; '8'沒被按下，檢查是否按了'9'
	movlw	f8h	; '8'被按下，設定七段顯示器為08
	movwf	digit1	; 設定七段顯示器的第一個數字為8
	movlw	f0h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為0
	goto	WaitRelease	; 等待按鍵放開
Key9:	btfsf	PDD, 1	; 檢查'9'是否被按下
	goto	KeyA	; '9'沒被按下，檢查是否按了'A'
	movlw	f9h	; '9'被按下，設定七段顯示器為09
	movwf	digit1	; 設定七段顯示器的第一個數字為9
	movlw	f0h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為0
	goto	WaitRelease	; 等待按鍵放開
KeyA:	btfsf	PDD, 2	; 檢查'A'是否被按下
	goto	KeyB	; 'A'沒被按下，所以是'B'被按下
	movlw	f0h	; 'A'被按下，設定七段顯示器為10
	movwf	digit1	; 設定七段顯示器的第一個數字為0
	movlw	f1h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為1
	goto	WaitRelease	; 等待按鍵放開
KeyB:	movlw	f1h	; 'B'被按下，設定七段顯示器為11
	movwf	digit1	; 設定七段顯示器的第一個數字為1
	movwf	digit2	; 設定七段顯示器的第二個數字為1
	goto	WaitRelease	; 等待按鍵放開
Row4:	;	檢查第四列哪個按鍵被按下	
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	7fh	; 將鍵盤第四列(bit 7)變成低電位
	movwf	PDD	;
	movfw	PDD	; 讀取鍵盤狀態，沒按鍵應是7fh

	addlw	80h	; +80h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	KeyC	; 有按鍵，檢查是否按了'C'
	goto	WaitRelease	; 等待按鍵放開
KeyC:	btfsz	PDD, 0	; 檢查'C'是否被按下
	goto	KeyD	; 'C'沒被按下，檢查是否按了'D'
	movlw	f2h	; 'C'被按下，設定七段顯示器為12
	movwf	digit1	; 設定七段顯示器的第一個數字為2
	movlw	f1h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為1
	goto	WaitRelease	; 等待按鍵放開
KeyD:	btfsz	PDD, 1	; 檢查'D'是否被按下
	goto	KeyE	; 'D'沒被按下，檢查是否按了'E'
	movlw	f3h	; 'D'被按下，設定七段顯示器為13
	movwf	digit1	; 設定七段顯示器的第一個數字為3
	movlw	f1h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為1
	goto	WaitRelease	; 等待按鍵放開
KeyE:	btfsz	PDD, 2	; 檢查'E'是否被按下
	goto	KeyF	; 'E'沒被按下，所以是'F'被按下
	movlw	f4h	; 'E'被按下，設定七段顯示器為14
	movwf	digit1	; 設定七段顯示器的第一個數字為4
	movlw	f1h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為1
	goto	WaitRelease	; 等待按鍵放開
KeyF:	movlw	f5h	; 'F'被按下，設定七段顯示器為15
	movwf	digit1	; 設定七段顯示器的第一個數字為5
	movlw	f1h	;
	movwf	digit2	; 設定七段顯示器的第二個數字為1
			; 等待使用者放開按鍵
WaitRelease:	call	ShowDigits	; 顯示七段顯示器內容
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	0fh	; 將0fh寫入PDD
	movwf	PDD	;
	movwf	PDD	; 讀取鍵盤狀態，若放開按鍵應是0fh
	addlw	f0h	; +f0h
	movwf	keyStatus	; 取得按鍵狀態，若放開按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示放開按鍵
	goto	WaitRelease	; 沒放開按鍵，繼續等待
	goto	Start	; 回到Start

```

; 顯示七段顯示器內容副程式

ShowDigits:  ; 顯示七段顯示器第一個數字
             movfw digit1      ; 輸出第一個數字的內容
             movwf PED         ;
             bcf PED, 4        ; 將第一個七段顯示器點亮
             call delay        ; 等待5ms
             bsf PED, 4        ; 將第一個七段顯示器關掉

             ; 顯示七段顯示器第二個數字
             movfw digit2      ; 輸出第二個數字的內容
             movwf PED         ;
             bcf PED, 5        ; 將第二個七段顯示器點亮
             call delay        ; 等待5ms
             bsf PED, 5        ; 將第二個七段顯示器關掉

             ret                ; 返回主程式

; 延遲0.005秒副程式
; 若clock rate=4MHz, 一個指令週期=2*clock=0.5μs,
; 要延遲0.005秒必須消耗10,000指令週期,
; 本程式使用兩層迴圈實作時間延遲功能, 時間延遲說明如下:
; 內層迴圈消耗5指令週期, 執行200次, 共1,000指令週期
; 外層迴圈執行10次, 1,000指令週期*10=10,000指令週期

delay:       movlw 10          ; 設定外層迴圈執行10次
             movwf R1          ;

             ;外層迴圈
delay_L1:    movlw 200         ; 設定內層迴圈執行200次
             movwf R2          ;

             ;內層迴圈: 迴圈跑一次約消耗5個指令週期
delay_L2:    nop              ; 消耗1個指令週期
             nop              ; 消耗1個指令週期
             decfsz R2, 1      ; 約消耗1個指令週期
             goto delay_L2    ; 消耗2個指令週期

             decfsz R1, 1      ; 將R1減1, 若R1=0離開迴圈
             goto delay_L1     ;

             ret                ; 返回主程式

```

**9. 數位電子琴實習****實習目的：**

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片實作簡單的數位電子琴。

**實習設備：**

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

**實習材料：**

項次	品名	數量
1	4 × 4 鍵盤	1
2	喇叭	1
3	20KΩ排阻(A-type 9PIN)	2
4	10KΩ電阻	2
5	10μf 電容	1
6	A1015 電晶體	2

**實習板模組與 I/O Port：**

模組	I/O Port
喇叭模組	Port B
4 × 4 鍵盤模組	Port D

**實習說明：**

本實習主要目的在利用十速 57FLA80 系列單晶片讀取 4 × 4 鍵盤的輸入，並依據使用者按下的按鍵播放不同聲音，4 × 4 鍵盤及其相關說明請參考第八章，按鍵與播放聲音的對照表如下所示：

**表 9-1 按鍵與聲音對照表**

按鍵	播放聲音	播放聲音頻率
0	低音 Do	131 Hz
1	低音 Re	147 Hz
2	低音 Mi	165 Hz
3	低音 Fa	175 Hz
4	低音 Sol	196 Hz
5	低音 La	220 Hz
6	低音 Si	247 Hz
7	中音 Do	262 Hz
8	中音 Re	294 Hz
9	中音 Mi	330 Hz
A	中音 Fa	349 Hz
B	中音 Sol	392 Hz
C	中音 La	440 Hz
D	中音 Si	494 Hz
E	高音 Do	522 Hz
F	高音 Re	587 Hz

要播放聲音，可使用喇叭(Speaker)元件，一般喇叭元件底部有一永久磁鐵，永久磁鐵上方有一薄膜，薄膜上會黏附線圈，線圈通電後產生磁場，帶動薄膜移動，線圈斷電後，薄膜則會回到原來的位置。通斷電喇叭一次會產生一次振動，透過控制每秒鐘喇叭通斷電次數，可產生不同頻率的聲音。

一般喇叭元件的硬體接線方式如圖 9-1 所示，其中使用兩顆電晶體是希望提供較大的電流以便推動喇叭元件。圖 9-1 所示，使用 57FLA80 單晶片 Port B 的 PB0 接腳控制喇叭元件的通斷電，當 PB0 接腳的腳位為低電位時(0)，電晶體會被導通，喇叭元件通電，當 PB0 接腳的腳位為高電位時(1)，電晶體不導通，喇叭元件斷電，喇叭元件不發聲時，應設置在斷電狀態，以減少耗電量。

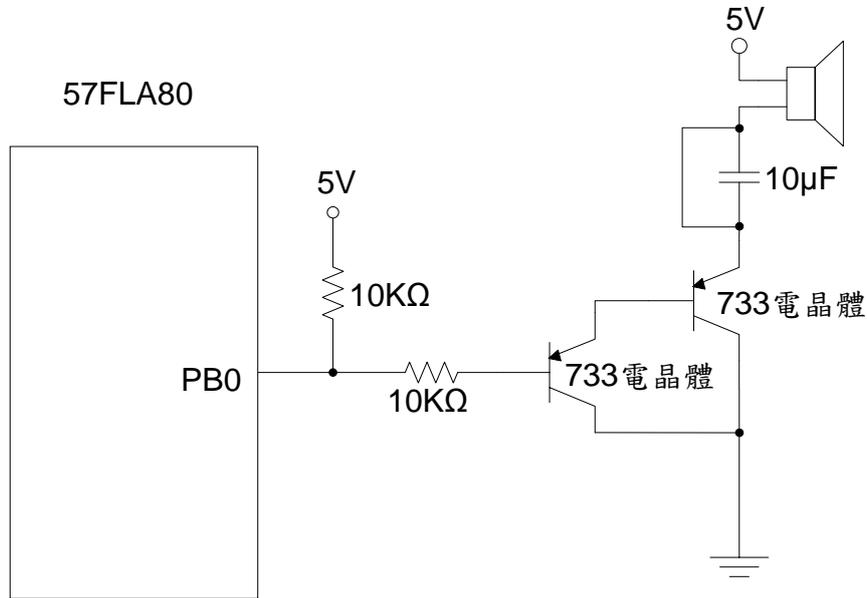


圖 9-1 喇叭元件硬體接線圖

要控制喇叭播放本實習所需要的聲音，必須計算喇叭薄膜振動的時間間隔，假設要播放的聲音頻率為  $F$ ，則喇叭薄膜振動時間間隔  $T$  的計算公式如下：

$$T = \frac{1}{F} \tag{9-1}$$

例如，由表 9-1 可知低音 Do 的頻率為 131 HZ，由公式(9-1)得知，要產生低音 Do 的聲音，必須每間隔  $1/131=0.007634$  秒= $7.634$  ms= $7634$   $\mu$ s 就要產生一次薄膜振動。產生一次薄膜振動包括通電與斷電兩個步驟，一般作法是將時間間隔分成兩半，通電與斷電各佔用一半時間。以產生低音 Do 的聲音為例，薄膜振動的時間間隔為  $7634$   $\mu$ s，因此，要產生一次薄膜振動，必須先將喇叭通電，然後等待  $7634$   $\mu$ s/ $2=3817$   $\mu$ s，接著再將喇叭斷電並等候  $3817$   $\mu$ s，整個過程合計約  $7634$   $\mu$ s。表 9-2 列出本實習所要播放聲音的頻率、薄膜振動時間間隔、通電、以及斷電時間。

表 9-2 各種聲音的頻率、薄膜振動時間、以及通斷電時間

播放聲音	頻率	薄膜振動時間	通電時間	斷電時間
低音 Do	131 Hz	7634 $\mu$ s	3817 $\mu$ s	3817 $\mu$ s
低音 Re	147 Hz	6803 $\mu$ s	3401 $\mu$ s	3401 $\mu$ s
低音 Mi	165 Hz	6061 $\mu$ s	3030 $\mu$ s	3030 $\mu$ s
低音 Fa	175 Hz	5714 $\mu$ s	2857 $\mu$ s	2857 $\mu$ s
低音 Sol	196 Hz	5102 $\mu$ s	2551 $\mu$ s	2551 $\mu$ s
低音 La	220 Hz	4545 $\mu$ s	2273 $\mu$ s	2273 $\mu$ s
低音 Si	247 Hz	4049 $\mu$ s	2024 $\mu$ s	2024 $\mu$ s
中音 Do	262 Hz	3817 $\mu$ s	1908 $\mu$ s	1908 $\mu$ s
中音 Re	294 Hz	3401 $\mu$ s	1701 $\mu$ s	1701 $\mu$ s
中音 Mi	330 Hz	3030 $\mu$ s	1515 $\mu$ s	1515 $\mu$ s
中音 Fa	349 Hz	2865 $\mu$ s	1433 $\mu$ s	1433 $\mu$ s
中音 Sol	392 Hz	2551 $\mu$ s	1276 $\mu$ s	1276 $\mu$ s
中音 La	440 Hz	2273 $\mu$ s	1136 $\mu$ s	1136 $\mu$ s
中音 Si	494 Hz	2024 $\mu$ s	1012 $\mu$ s	1012 $\mu$ s
高音 Do	522 Hz	1916 $\mu$ s	958 $\mu$ s	958 $\mu$ s
高音 Re	587 Hz	1704 $\mu$ s	852 $\mu$ s	852 $\mu$ s

硬體線路圖：

本實習電路圖如圖 9-2 所示，本實習使用 57FLA80 單晶片 Port D 來控制 4 × 4 鍵盤，並且使用 Port B 的 PB0 來控制喇叭的開關，PB0 的值為 0 (低電位) 表示將喇叭通電，PB0 的值為 1 (高電位) 表示將喇叭斷電。透過控制一秒鐘開關喇叭的次數，可發出不同頻率的聲音。

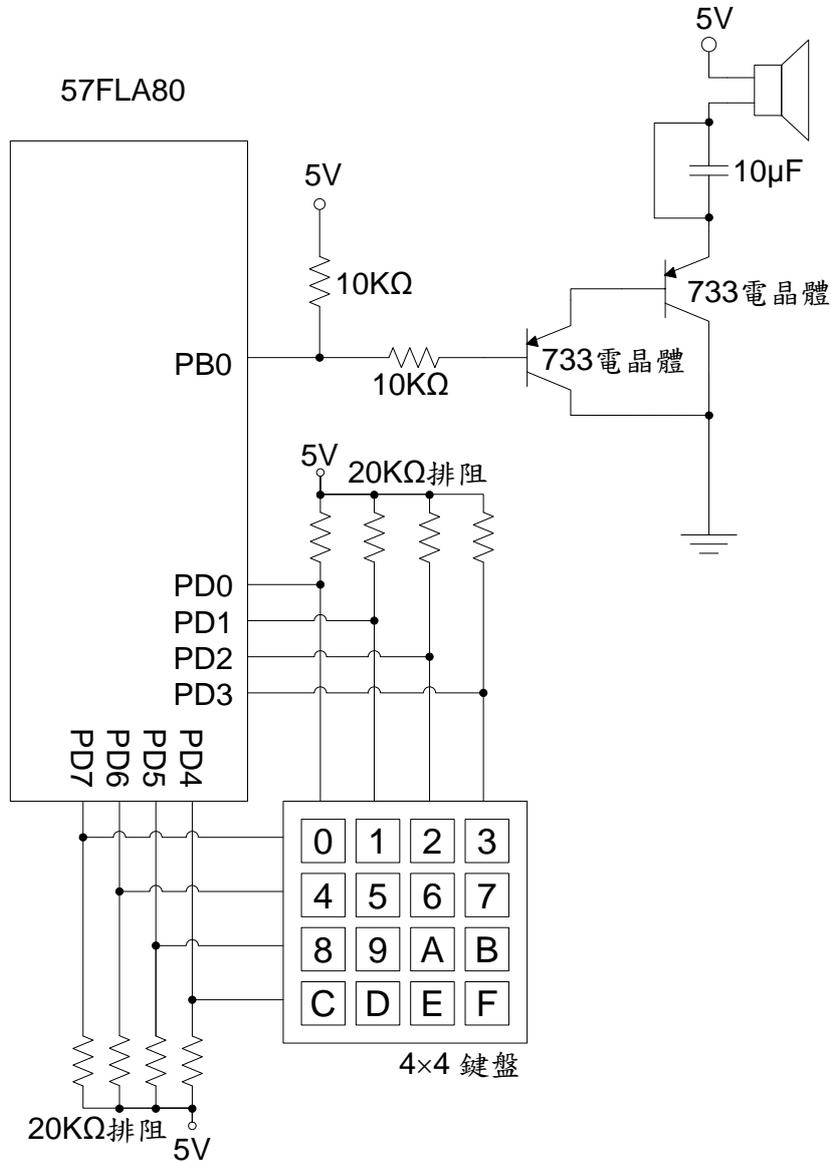


圖 9-2 數位電子琴硬體線路圖

程式流程：

本實習主要工作在檢查使用者是否按下按鍵，如果使用者按下按鍵，則依據使用者所按的鍵設定聲音延遲參數，然後播放聲音，直到使用者放開按鍵為止。主程式流程如圖 9-3 所示。

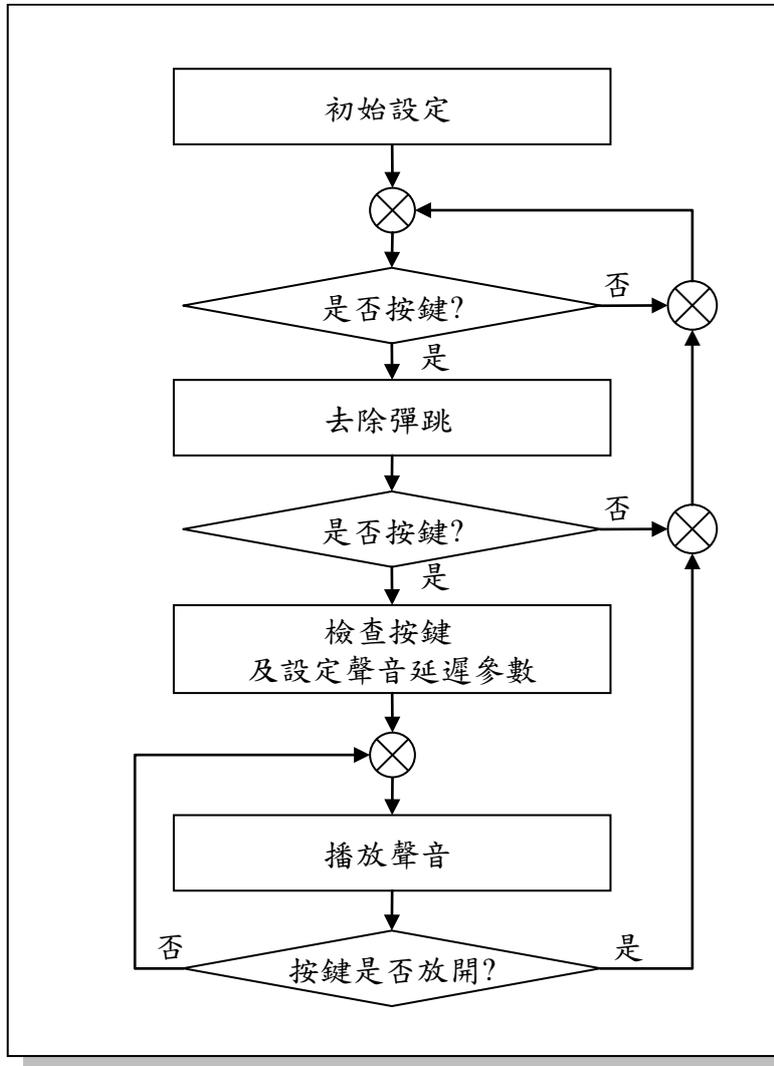


圖 9-3 數位電子琴主程式流程圖

主程式流程中，檢查按鍵及設定聲音延遲參數流程如下圖所示。

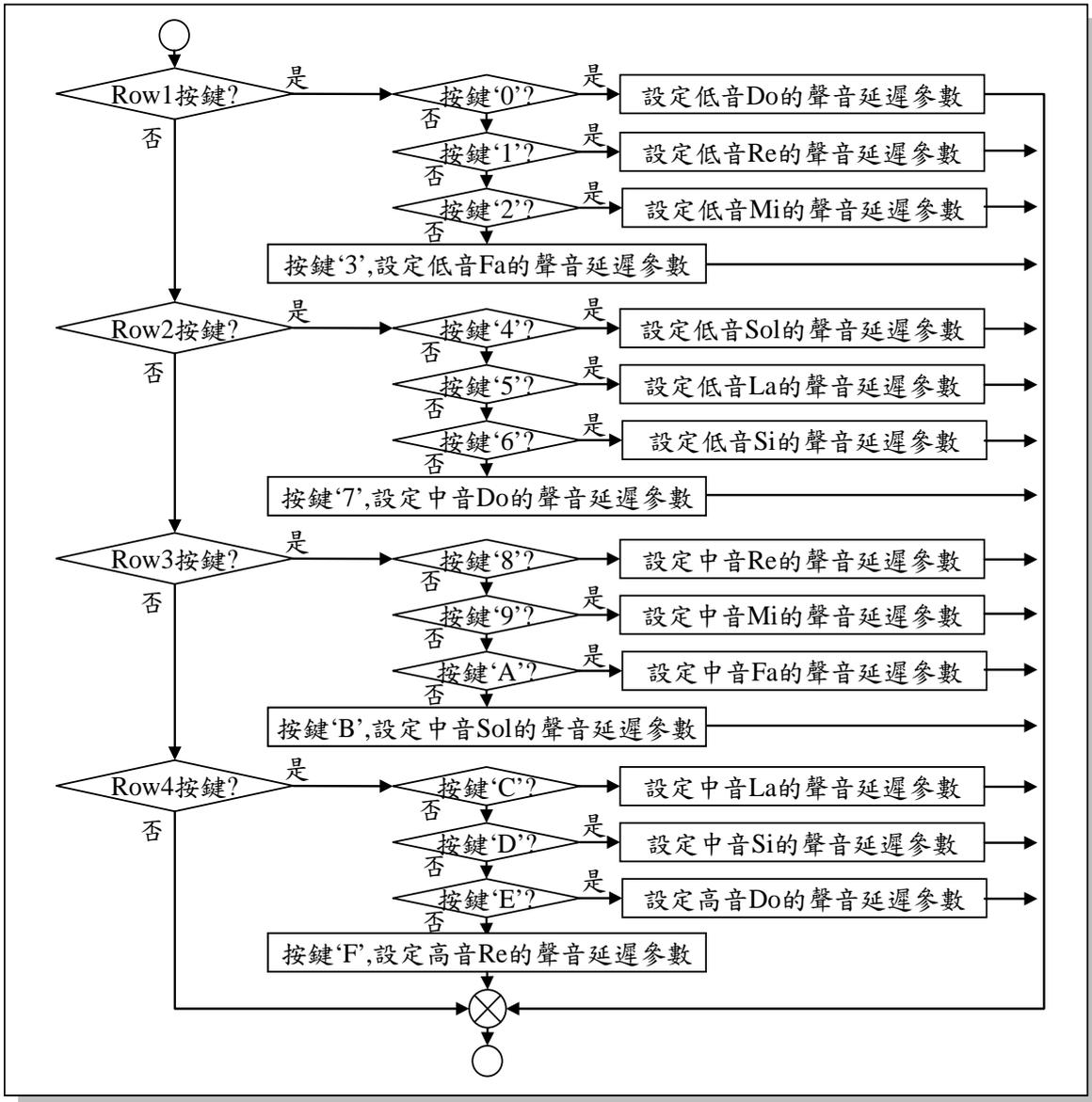


圖 9-4 按鍵檢查程式流程圖

聲音播放方式為將喇叭通電，等待通電時間完成，將喇叭斷電，再等待斷電時間完成。由於每個聲音通斷電時間不同，為避免針對每個聲音撰寫不同的通斷電時間等待程式，本實習提供一個可延遲 15 μs 的時間延遲函式，播放聲音時，可依需求呼叫該函式數次，產生近似所需要的延遲時間。表 9-3 列出不同聲音的通斷電時間以及時間延遲函式的呼叫次數。

表 9-3 聲音通斷電時間、延遲函式呼叫次數、以及實際延遲時間

播放聲音	播放聲音的通斷電時間	延遲函式呼叫次數	實際延遲時間
低音 Do	3817 $\mu$ s	254	3810 $\mu$ s
低音 Re	3401 $\mu$ s	227	3405 $\mu$ s
低音 Mi	3030 $\mu$ s	202	3030 $\mu$ s
低音 Fa	2857 $\mu$ s	190	2850 $\mu$ s
低音 Sol	2551 $\mu$ s	170	2550 $\mu$ s
低音 La	2273 $\mu$ s	152	2280 $\mu$ s
低音 Si	2024 $\mu$ s	135	2025 $\mu$ s
中音 Do	1908 $\mu$ s	127	1905 $\mu$ s
中音 Re	1701 $\mu$ s	113	1695 $\mu$ s
中音 Mi	1515 $\mu$ s	101	1515 $\mu$ s
中音 Fa	1433 $\mu$ s	96	1440 $\mu$ s
中音 Sol	1276 $\mu$ s	85	1275 $\mu$ s
中音 La	1136 $\mu$ s	76	1140 $\mu$ s
中音 Si	1012 $\mu$ s	67	1005 $\mu$ s
高音 Do	958 $\mu$ s	64	960 $\mu$ s
高音 Re	852 $\mu$ s	57	855 $\mu$ s

## 程式碼及程式說明：

；定義程式中使用到的F-Plane暫存器記憶體位址

```
PBD      equ      06h      ; Port B
PDD      equ      12h      ; Port D
```

；定義程式中使用到的F-Plane變數記憶體位址

```
keyStatus equ      20h      ; 按鍵狀態
soundData equ      21h      ; 聲音延遲參數
times15us equ      22h      ; 延遲時間, 要延遲幾個15µs
R1        equ      23h      ; 時間延遲迴圈控制變數
```

；定義程式中使用到的R-Plane變數記憶體位址

```
PEE      equ      13h      ; Port E Push-Pull enable
```

；定義程式中使用到之各種聲音的聲音延遲參數(15µs時間延遲函式呼叫次數)

```
L_Do     equ      254      ; 低音Do
L_Re     equ      227      ; 低音Re
L_Mi     equ      202      ; 低音Mi
L_Fa     equ      190      ; 低音Fa
L_Sol    equ      170      ; 低音Sol
L_La     equ      152      ; 低音La
L_Si     equ      135      ; 低音Si
```

```
M_Do     equ      127      ; 中音Do
M_Re     equ      113      ; 中音Re
M_Mi     equ      101      ; 中音Mi
M_Fa     equ      96       ; 中音Fa
M_Sol    equ      85       ; 中音Sol
M_La     equ      76       ; 中音La
M_Si     equ      67       ; 中音Si
```

```
H_Do     equ      64       ; 高音Do
H_Re     equ      57       ; 高音Re
```

；系統開機進入點

```
org      00h
```

；初始設定

```
bsf     PBD, 0      ; 設定PB0的值為1，喇叭斷電
```

```

; 致能鍵盤輸入腳位Push-pull功能
movlw    ffh
movwr    PEE

; 檢查是否有按鍵被按下
Start:   movlw    ffh          ;
        movwf    PDD          ; 清除按鍵狀態
        movlw    0fh          ; 將鍵盤列位址(bits4~7)變成低電位
        movwf    PDD          ; 將鍵盤行位址(bits0~3)變成高電位
        movfw    PDD          ; 讀取鍵盤狀態，沒按鍵應是0FH
        addlw    f0h          ; +F0H
        movwf    keyStatus    ; 取得按鍵狀態，沒按鍵是0FH+F0H=FFH
        incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
        goto     DeNoise      ; 有按鍵，跳到DeNoise去除彈跳
        goto     Start        ; 沒按鍵，回到Start

; 去除彈跳
DeNoise: movlw    0          ;
        movwf    times15us   ;
        call     Delay        ; 等待256x15us

; 檢查是否有按鍵被按下
        movlw    ffh          ;
        movwf    PDD          ; 清除按鍵狀態
        movlw    0fh          ; 將鍵盤列位址(bits4~7)變成低電位
        movwf    PDD          ; 將鍵盤行位址(bits0~3)變成高電位
        movfw    PDD          ; 讀取鍵盤狀態，沒按鍵應是0FH
        addlw    f0h          ; +F0H
        movwf    keyStatus    ; 取得按鍵狀態，沒按鍵是FFH
        incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
        goto     Row1         ; 有按鍵，到Row1檢查按了什麼鍵
        goto     Start        ; 沒按鍵，回到Start

Row1:    ; 檢查第一列是否有按鍵被按下
        movlw    ffh          ;
        movwf    PDD          ; 清除按鍵狀態
        movlw    efh          ; 將鍵盤第一列(bit 4)變成低電位
        movwf    PDD          ;
        movfw    PDD          ; 讀取鍵盤狀態，沒按鍵應是efh
        addlw    10h          ; +10h
        movwf    keyStatus    ; 取得按鍵狀態，沒按鍵應是ffh
        incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
        goto     Key0         ; 有按鍵，檢查是否按了'0'
        goto     Row2        ; 沒按鍵，檢查第二列

```

Key0:	btfsc	PDD, 0	; 檢查'0'是否被按下
	goto	Key1	; '0'沒被按下，檢查是否按了'1'
	movlw	L_Do	; '0'被按下，設定聲音延遲參數為L_Do
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
Key1:	btfsc	PDD, 1	; 檢查'1'是否被按下
	goto	Key2	; '1'沒被按下，檢查是否按了'2'
	movlw	L_Re	; '1'被按下，設定聲音延遲參數為L_Re
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
Key2:	btfsc	PDD, 2	; 檢查'2'是否被按下
	goto	Key3	; '2'沒被按下，所以是'3'被按下
	movlw	L_Mi	; '2'被按下，設定聲音延遲參數為L_Mi
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
Key3:	movlw	L_Fa	; '3'被按下，設定聲音延遲參數為L_Fa
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
Row2:	; 檢查第二列是否有按鍵被按下		
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	dfh	; 將鍵盤第二列(bit 5)變成低電位
	movwf	PDD	;
	movfw	PDD	; 讀取鍵盤狀態，沒按鍵應是dfh
	addlw	20h	; +20h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	Key4	; 有按鍵，檢查是否按了'4'
	goto	Row3	; 沒按鍵，檢查第三列
Key4:	btfsc	PDD, 0	; 檢查'4'是否被按下
	goto	Key5	; '4'沒被按下，檢查是否按了'5'
	movlw	L_Sol	; '4'被按下，設定聲音延遲參數為L_Sol
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
Key5:	btfsc	PDD, 1	; 檢查'5'是否被按下
	goto	Key6	; '5'沒被按下，檢查是否按了'6'
	movlw	L_La	; '5'被按下，設定聲音延遲參數為L_La
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開

Key6:	<pre> btfsc      PDD, 2      ; 檢查'6'是否被按下 goto      Key7        ; '6'沒被按下，所以是'7'被按下 movlw     L_Si        ; '6'被按下，設定聲音延遲參數為L_Si movwf     soundData   ; goto      WaitRelease ; 等待按鍵放開 </pre>
Key7:	<pre> movlw     M_Do        ; '7'被按下，設定聲音延遲參數為M_Do movwf     soundData   ; goto      WaitRelease ; 等待按鍵放開 </pre>
Row3:	<pre> ; 檢查第三列是否有按鍵被按下 movlw     ffh         ; movwf     PDD         ; 清除按鍵狀態 movlw     bfh         ; 將鍵盤第三列(bit 6)變成低電位 movwf     PDD         ; movfw     PDD         ; 讀取鍵盤狀態，沒按鍵應是bfh addlw     40h         ; +40h movwf     keyStatus   ; 取得按鍵狀態，沒按鍵應是ffh incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵 goto     Key8         ; 有按鍵，檢查是否按了'8' goto     Row4         ; 沒按鍵，檢查第四列 </pre>
Key8:	<pre> btfsc     PDD, 0      ; 檢查'8'是否被按下 goto     Key9        ; '8'沒被按下，檢查是否按了'9' movlw     M_Re        ; '8'被按下，設定聲音延遲參數為M_Re movwf     soundData   ; goto     WaitRelease ; 等待按鍵放開 </pre>
Key9:	<pre> btfsc     PDD, 1      ; 檢查'9'是否被按下 goto     KeyA        ; '9'沒被按下，檢查是否按了'A' movlw     M_Mi        ; '9'被按下，設定聲音延遲參數為M_Mi movwf     soundData   ; goto     WaitRelease ; 等待按鍵放開 </pre>
KeyA:	<pre> btfsc     PDD, 2      ; 檢查'A'是否被按下 goto     KeyB        ; 'A'沒被按下，所以是'B'被按下 movlw     M_Fa        ; 'A'被按下，設定聲音延遲參數為M_Fa movwf     soundData   ; goto     WaitRelease ; 等待按鍵放開 </pre>
KeyB:	<pre> movlw     M_Sol       ; 'B'被按下，設定聲音延遲參數為M_Sol movwf     soundData   ; goto     WaitRelease ; 等待按鍵放開 </pre>

Row4:			; 檢查第四列哪個按鍵被按下
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	7fh	; 將鍵盤第四列(bit 7)變成低電位
	movwf	PDD	;
	movfw	PDD	; 讀取鍵盤狀態，沒按鍵應是7fh
	addlw	80h	; +80h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	KeyC	; 有按鍵，檢查是否按了'C'
	goto	WaitRelease	; 等待按鍵放開
KeyC:	btfsc	PDD, 0	; 檢查'C'是否被按下
	goto	KeyD	; 'C'沒被按下，檢查是否按了'D'
	movlw	M_La	; 'C'被按下，設定聲音延遲參數為M_La
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
KeyD:	btfsc	PDD, 1	; 檢查'D'是否被按下
	goto	KeyE	; 'D'沒被按下，檢查是否按了'E'
	movlw	M_Si	; 'D'被按下，設定聲音延遲參數為M_Si
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
KeyE:	btfsc	PDD, 2	; 檢查'E'是否被按下
	goto	KeyF	; 'E'沒被按下，所以是'F'被按下
	movlw	H_Do	; 'E'被按下，設定聲音延遲參數為H_Do
	movwf	soundData	;
	goto	WaitRelease	; 等待按鍵放開
KeyF:	movlw	H_Re	; 'F'被按下，設定聲音延遲參數為H_Re
	movwf	soundData	;
			; 等待使用者放開按鍵
WaitRelease:	call	PlaySound	; 播放聲音
	movlw	ffh	;
	movwf	PDD	; 清除按鍵狀態
	movlw	0fh	; 將0fh寫入PDD
	movwf	PDD	;
	movfw	PDD	; 讀取鍵盤狀態，若放開按鍵應是0fh
	addlw	f0h	; +f0h
	movwf	keyStatus	; 取得按鍵狀態，若放開按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示放開按鍵
	goto	WaitRelease	; 沒放開按鍵，繼續等待
	goto	Start	; 回到Start

```

; 聲音播放函式
; 喇叭通電，等候延遲時間，喇叭斷電，等候延遲時間
PlaySound:   bcf           PBD,0           ; 設定PB0的值為0，喇叭通電

              movfw        soundData      ; 載入延遲時間
              movwf        times15us     ; 設定延遲時間
              call         Delay          ; 延遲 times15us × 15µs

              bsf          PBD, 0        ; 設定PB0的值為1，喇叭斷電

              movfw        soundData      ; 載入延遲時間
              movwf        times15us     ; 設定延遲時間
              call         Delay          ; 延遲 times15us × 15µs
              ret

; 延遲times15us×15us副程式
; 若clock rate=4MHz, 一個指令週期=2*clock=0.5µs,
; 本程式使用兩層迴圈實作時間延遲功能，時間延遲說明如下：
; 外層迴圈執行times15us次，每次延遲15µs，消耗30指令週期
; 1+1+4*(1+1+1+2)+(1+1+2)+1+1+2=30指令週期
Delay:        ; 外層迴圈
              movlw        5              ; 消耗1個指令週期
              movwf        R1             ; 消耗1個指令週期

              ; 內層迴圈
Delay_L1:     nop                       ; 消耗1個指令週期
              nop                       ; 消耗1個指令週期
              decfsz       R1, 1         ; 消耗1個指令週期
              goto         Delay_L1      ; 消耗2個指令週期

              nop                       ; 消耗1個指令週期
              decfsz       times15us, 1 ; 消耗1個指令週期
              goto         Delay         ; 消耗2個指令週期
              ret                       ; 返回主程式

```

**10. 數位電錶類比轉數位實習****實習目的：**

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片的類比數位轉換(Analog to Digital Convert)功能實作數位電錶。

**實習設備：**

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

**實習材料：**

項次	品名	數量
1	10KΩ排阻(A-type 9PIN)	3
2	4051 IC	1
3	74LS245IC	1
4	74LS47IC	2
5	741 運算放大器	3
6	可變電阻	1
7	330Ω電阻	3
8	1KΩ電阻	1
9	4KΩ電阻	1
10	4.7KΩ電阻	2
11	七段顯示器(共陽)	3
12	指撥開關(4 個開關)	1

**實習板模組與 I/O Port：**

模組	I/O Port
數位類比轉換模組	Port B
四顆七段顯示器模組	Port D
四顆七段顯示器模組	Port E

**實習說明：**

本實習將實作一個數位電錶，該電錶提供表 10-1 所示三種量測模式，使用者可根據所要量測的電壓範圍使用指撥開關選擇適當的量測模式，再使用量測端子量測電壓。外部電壓值可透過十速 57FLA80 系列單晶片內部提供的類比數位轉換功能轉成數位訊號，再透過七段顯示器顯示出來。

**表 10-1 數位電錶量測模式說明**

量測模式	可量測電壓範圍
一	0V ~ 10V
二	0V ~ 5V
三	0V ~ 1V

本實習使用三顆七段顯示器顯示轉換後的結果，表 10-2 列出三種模式下部份輸入電壓與七段顯示器的顯示內容。

表 10-2 各種量測模式下的七段顯示器顯示方式

量測模式一		量測模式二		量測模式三	
電壓	顯示結果	電壓	顯示結果	電壓	顯示結果
0.0V		0.0V		0.00V	
1.0V		0.5V		0.10V	
2.0V		1.0V		0.20V	
3.0V		1.5V		0.30V	
4.0V		2.0V		0.40V	
5.0V		2.5V		0.50V	
6.0V		3.0V		0.60V	
7.0V		3.5V		0.70V	
8.0V		4.0V		0.80V	
9.0V		4.5V		0.90V	
10.0V		5.0V		1.00V	

使用數位電錶量測物體的電壓必須將量測端子接到要量測的點上，為避免數位電錶訊號輸入腳位的輸入阻抗(Input Impedance)影響到待測物的電壓值，可使用運算放大器(Operational Amplifier)緩衝電路隔離數位電錶對輸入訊號的影響。圖 10-1 所示為運算放大器的緩衝電路。輸入訊號經過運算放大器緩衝電路後，會產生電壓值跟輸入訊號相同的輸出訊號供使用。

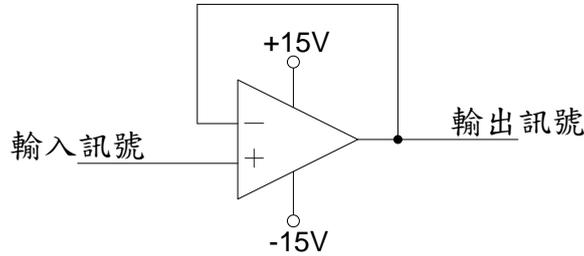


圖 10-1 運算放大器緩衝電路

十速 57FLA80 系列單晶片內部提供的類比轉數位的電壓範圍是 0V ~ 5V，本實習提供三種模式，如果輸入訊號的電壓範圍不是 0V ~ 5V，則必須將輸入訊號的電壓範圍調整成 0V ~ 5V。圖 10-2 與圖 10-3 分別是將電壓範圍 0V ~ 10V 與 0V ~ 1V 調整成 0V ~ 5V 的電路圖。

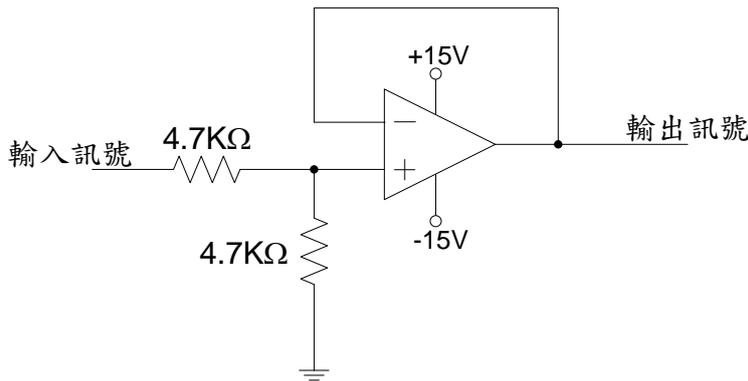


圖 10-2 電壓範圍 0V ~ 10V 調整至 0V ~ 5V 電路圖

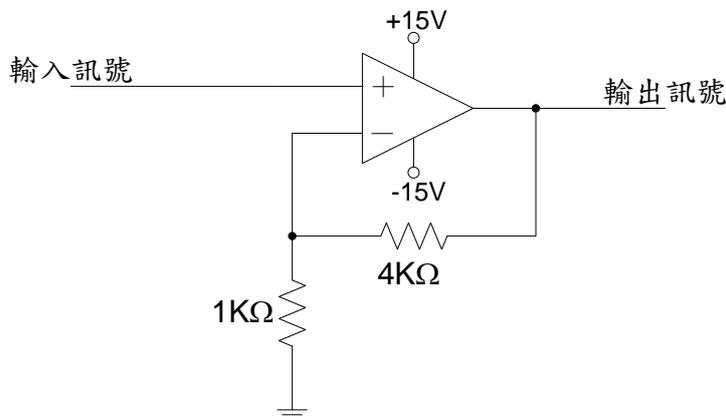


圖 10-3 電壓範圍 0V ~ 1V 調整至 0V ~ 5V 電路圖

本實習提供三種模式供使用者選擇，輸入訊號的電壓範圍可能是 0V ~ 1V、0V ~ 5V、與 0V ~ 10V 其中一種。由於三種模式必須有不同的電壓調整需求，為此，本實習使用類比多工器將輸入訊號進行 1 對 3 的多工處理，並將三個多工輸出分別接到三個不同的電壓調整電路。4051 是一顆 1 對 8 的類比多工器 IC，4051 的腳位圖如圖 10-4 所示，4051 的腳位說明則如表 10-3 所列：

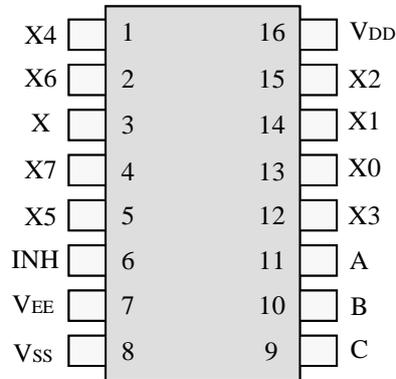


圖 10-4 4051 腳位圖

表 10-3 4051 腳位說明

腳位	腳位說明
X	輸入訊號
X0~X7	訊號輸出腳位
INH	1: 關掉輸入訊號 0: 允許輸入訊號
C, B, A	000: 輸入訊號輸出至 X0 001: 輸入訊號輸出至 X1 010: 輸入訊號輸出至 X2 011: 輸入訊號輸出至 X3 100: 輸入訊號輸出至 X4 101: 輸入訊號輸出至 X5 110: 輸入訊號輸出至 X6 111: 輸入訊號輸出至 X7
V <sub>DD</sub>	正供應電壓 $V_{DD} \geq V_{SS}$
V <sub>EE</sub>	負供應電壓， $V_{EE} \leq V_{SS}$
V <sub>SS</sub>	接地

本實習使用 4051 對輸入訊號進行多工處理，並提供使用者利用指撥開關選擇量測模式，指撥開關也直接接到 4051 多工器的 CBA 腳位，用來選擇 4051 多工器的 X1、X2、與 X4 等輸出腳位，這三個腳位分別用來處理電壓範圍 0V~10V、0V~5V、與 0V~1V 等情況。圖 10-5 列出 4051 控制線路以及電壓範圍調整等相關電路。

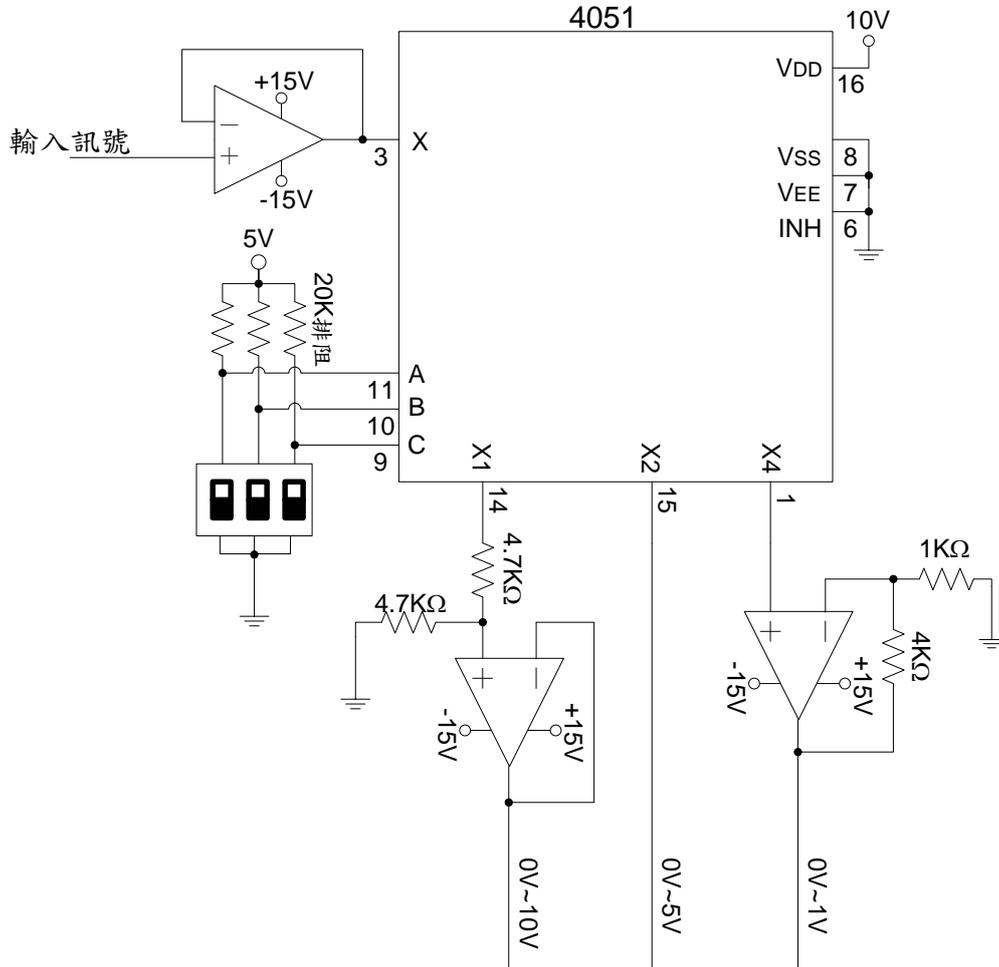


圖 10-5 4051 控制線路以及電壓範圍調整電路

取得符合電壓範圍(0V~5V)的輸入訊號後，接下來便可將輸入訊號接到十速 57FLA80 系列單晶片的類比輸入腳位，進行類比數位轉換，57FLA80 系列單晶片的類比數位轉換功能方塊圖如圖 10-6 所示：

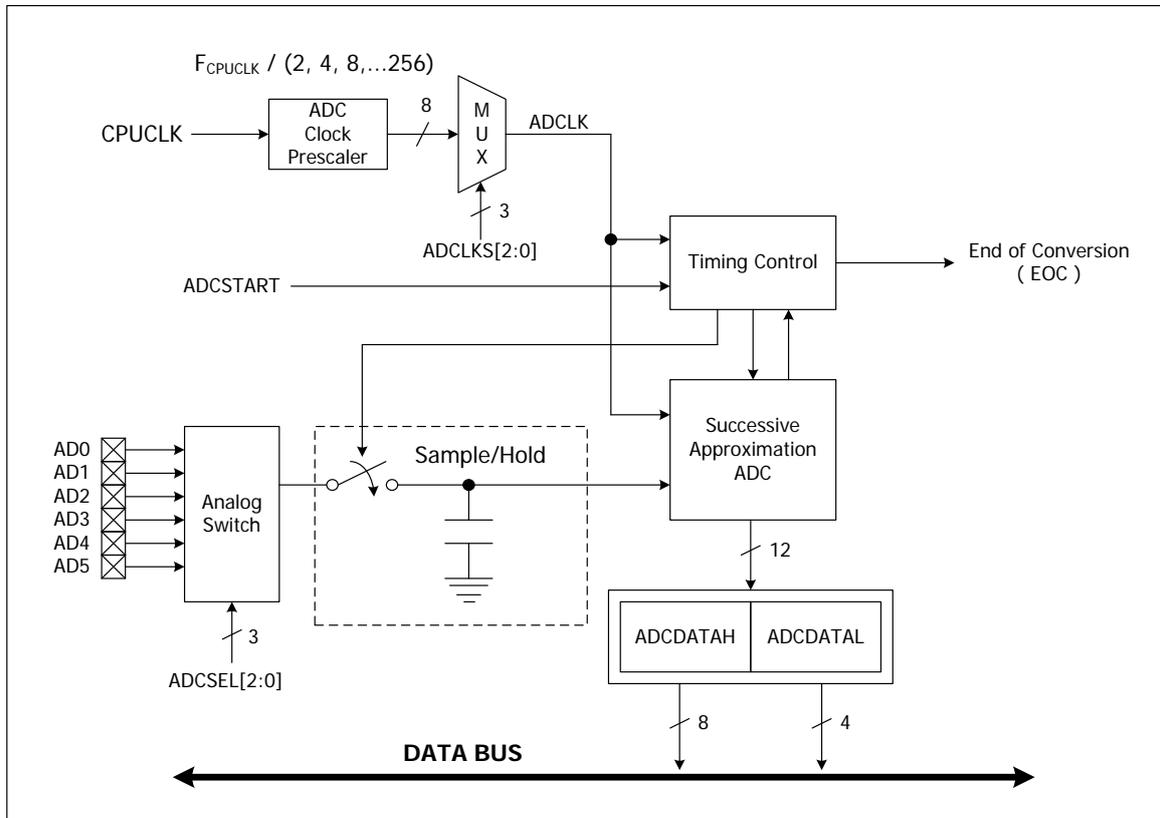


圖 10-6 57FLA80 系列單晶片之類比數位轉換功能方塊圖

57FLA80 系列單晶片提供 6 個類比輸入接腳 AD0~AD5 供進行 12 位元的類比數位轉換，同一時間只能進行一個類比數位轉換，類比輸入腳位 (AD0~AD5) 可透過設定 ADCSEL 控制線的內容來選擇。除了選擇類比輸入腳位外，還必須修改 ADCPIN 控制暫存器將對應類比輸入腳位的位元設定為 0，才能將該輸入腳位設定成類比輸入模式（一般腳位預設功能是作為 I/O 埠使用），例如：PB5/AD1 腳位可作為 PB5(I/O 埠)或是 AD1(類比輸入) 使用，其預設功能是作為 PB5 使用，要做為類比輸入使用，必須將 ADCPIN 控制暫存器位元 1 設定為 0。

進行類比數位轉換前，必須先選擇 ADC Clock，57FLA80 提供八種 ADC Clock 供選擇，分別是 CPU 時脈的 1/2、1/4、1/8、1/16、1/32、1/64、1/128、與 1/256 等，當 CPU 時脈為 4 MHz 時，可用的 ADC Clock 分別為 2M、1M、512K、256K、128K、64K、32K、與 16 KHz。ADC Clock 的選用可透過 ADCLKS 暫存器來設定。

選定類比訊號輸入腳位以及 ADC Clock 後，可利用 ADCSTART 控制位元啟動轉換功能，轉換完成後，ADCSTART 控制位元會被清為 0，轉換結果則會在放在 ADCDATAH (8 位元)與 ADCDATA L (4 位元)暫存器內，共 12 個位元。

圖 10-7 所示為類比數位轉換功能的時序圖，由時序圖可以看到 57FLA80 單晶片需要花費 50 個 ADC Clock 進行一次類比數位轉換，若使用 4 MHz 的 CPU 時脈，可供使用的取樣頻率分別為 40K、20K、10K、5K、2.5K、1250、750、與 375 Hz。

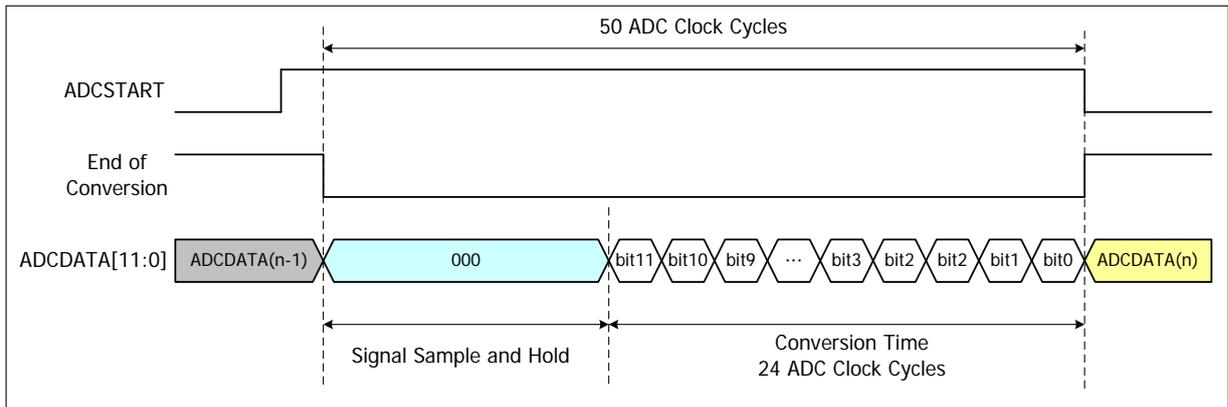


圖 10-7 類比數位轉換功能時序圖

57FLA80 中類比數位轉換相關的控制暫存器整理如下表所示：

表 10-4 類比數位轉換相關控制暫存器

暫存器	說明
ADCLKS	<p>位於 R-PLANE 位址 0ch 暫存器的位元 6~4，用來選擇 ADC Clock。 ADCLKS 內容與 ADC Clock 關係如下，其中位元排列順序由左而右分別是位元 6~4，<math>F_{CPUCLK}</math> 表單晶片的 CPU Clock。</p> <p>000: <math>F_{CPUCLK}/256</math> 001: <math>F_{CPUCLK}/128</math> 010: <math>F_{CPUCLK}/64</math> 011: <math>F_{CPUCLK}/32</math> 100: <math>F_{CPUCLK}/16</math> 101: <math>F_{CPUCLK}/8</math> 110: <math>F_{CPUCLK}/4</math> 111: <math>F_{CPUCLK}/2</math></p>
ADCSEL	<p>位於 F-PLANE 位址 11h 暫存器的位元 2~0，用來選擇類比輸入。 ADCSEL 內容與選用的類比輸入關係如下，其中位元排列順序由左而右分別是位元 2~0。</p> <p>000: AD0 001: AD1 010: AD2 011: AD3 100: AD4 101: AD5</p>
ADCPIN	<p>位於 R-PLANE 位址 16h 暫存器的位元 0~5 分別用來設定 AD0~AD5 腳位的用法，可供選用的用法有類比輸入與 I/O 埠兩種。位元的值預設值是 1，表 I/O 埠，位元的值為 0 表類比輸入。</p>
ADCSTART	<p>位於 F-PLANE 位址 11h 暫存器的位元 3，設定此位元的值為 1，可啟動類比數位轉換，轉換完成，位元內容會被清為 0。</p>
ADCDATA	<p>內含類比數位轉換結果，包括 ADCDATAH 與 ADCDATAL 兩部份， ADCDATAH 大小 8 個位元，位於 F-PLANE 位址 10h， ADCDATAL 大小 4 個位元，位於 F-PLANE 位址 11h 位元 7~4。</p>

硬體線路圖：

本實習完整硬體線路圖如圖 10-8 所示，為了方便測試量測結果的正確性，本實習使用  $V_{in}$  加可變電阻模擬待測物，調整可變電阻的電阻值可改變輸入電壓值， $V_{in}$  可接 10V、5V、或 1V。

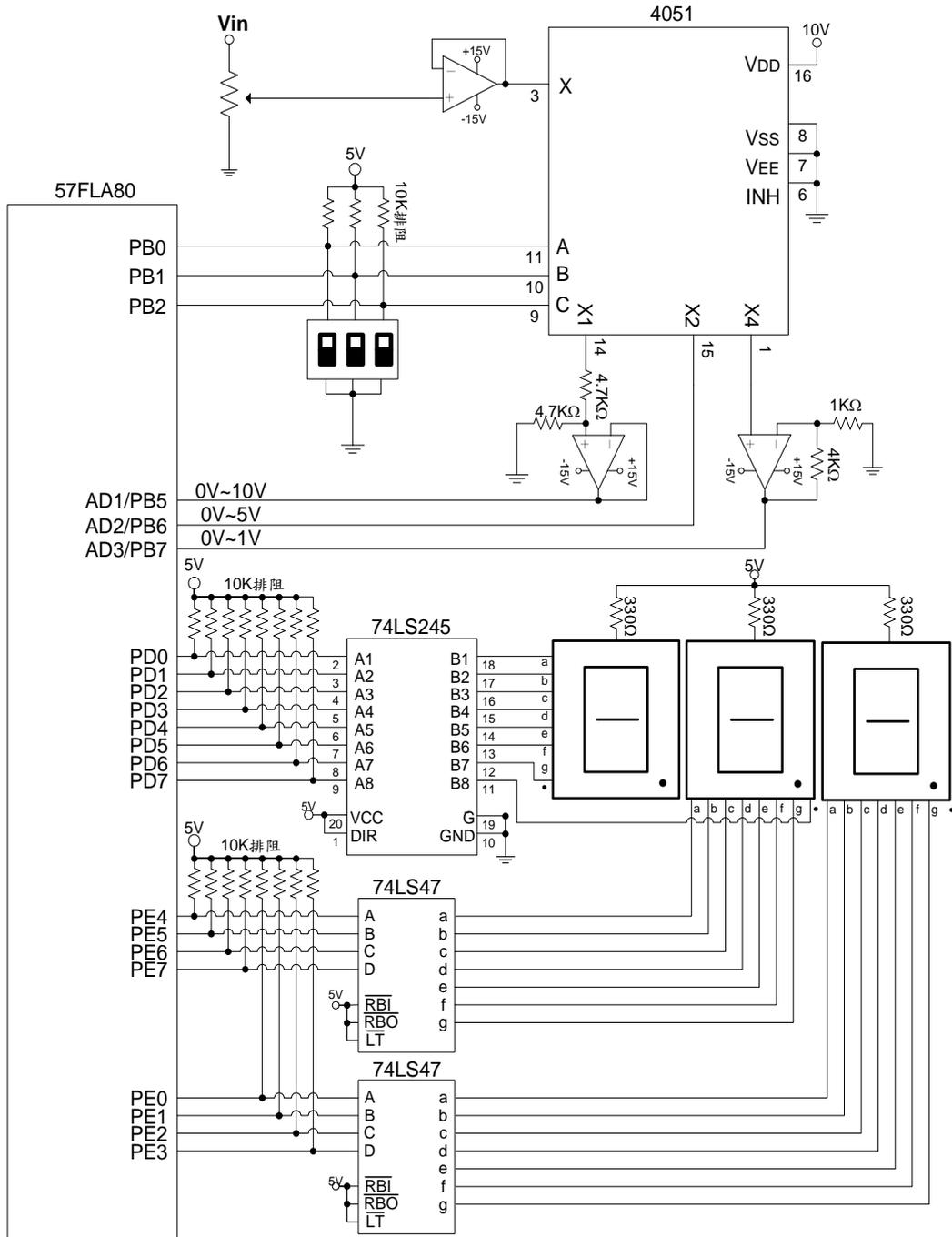


圖 10-8 數位電錶硬體線路圖

本實習使用三個 IO 埠，Port B 的 PB2 ~ PB0 用來讀取目前數位電錶的運作模式（PB2 ~ PB0 的內容為 001 表模式一、010 表模式二、100 表模式三），Port B 的 PB5 ~ PB7 作為類比輸入用(AD1 ~ AD3)，分別用來連接模式一、模式二、與模式三的類比輸入；Port D 與 Port E 用來顯示七段顯示器的內容，其中，Port D 的 PD0 ~ PD6 用來控制最左邊七段顯示器的顯示，由於最左邊七段顯示器只會顯示 0 與 1 兩種數字以及小數點，g 位置的 LED 不會有機會點亮，因此最左邊七段顯示器的 g 腳位不接，Port D 的 PD7 用來控制中間七段顯示器的小數點，Port E 的 PE4 ~ PE7 用來傳送中間七段顯示器的 BCD 碼，Port E 的 PE0 ~ PE3 則用來傳送右邊七段顯示器的 BCD 碼。

程式流程：

本實習程式流程如圖 10-9 所示，首先設定 ADC Clock 及關掉所有七段顯示器，由於本實習只是要量測電壓值，不需太高的取樣頻率，因此只要選擇最慢的 ADC Clock 即可；接著檢查指撥開關看使用者選擇何種量測模式，若使用者選擇任何一種量測模式，程式便會針對使用者所選擇的模式，啟用適當的類比輸入，進行類比數位轉換，以及顯示轉換後的結果。

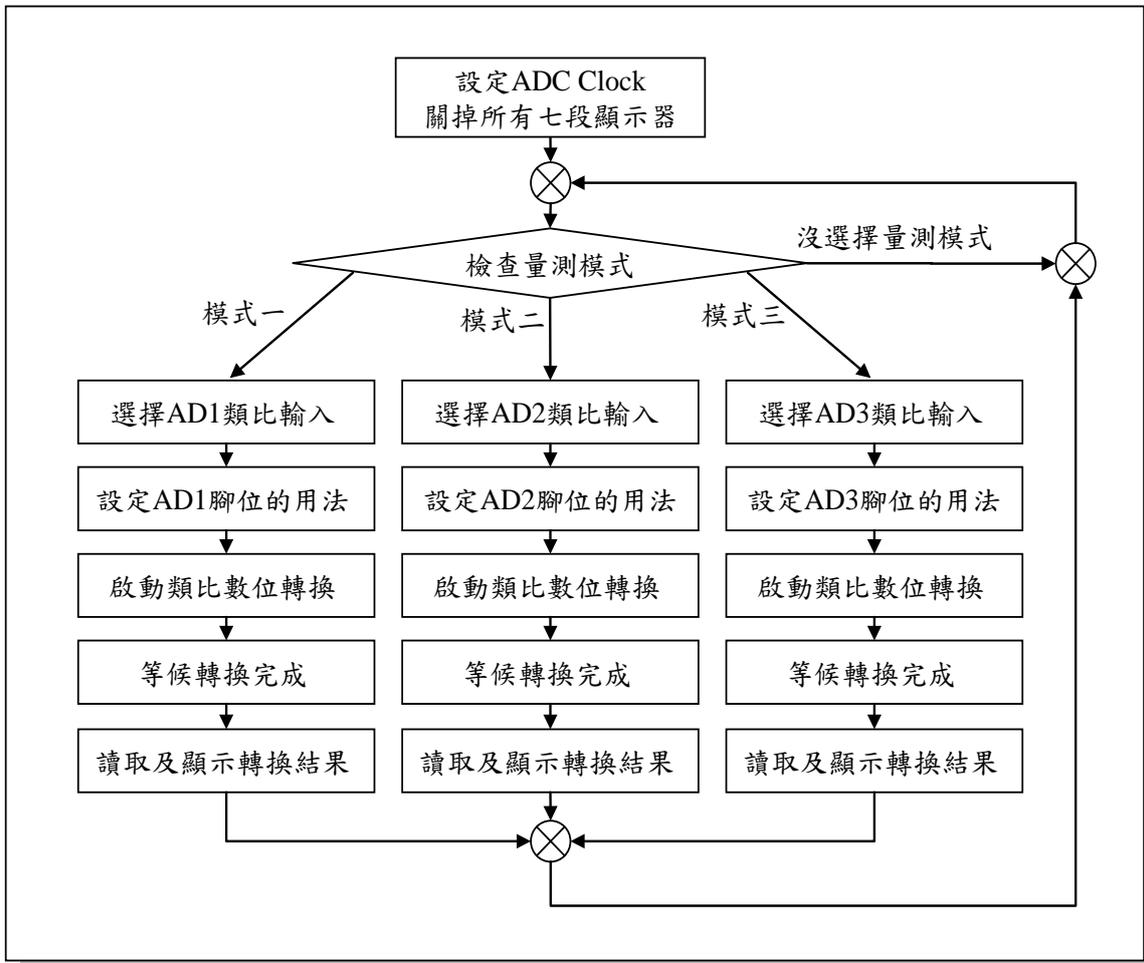


圖 10-9 數位電錶主程式流程圖

## 程式碼及程式說明：

；定義程式中使用到的F-Plane暫存器記憶體位址

```
PC          equ          02h          ; Program Counter
PBD         equ          06h          ; Port B
PDD         equ          12h          ; Port D
PED         equ          13h          ; Port E

ADCDATAH   equ          10h          ; 類比數位轉換結果前8個位元
ADCSEL     equ          11h          ; 選擇類比輸入用
```

；定義程式中使用到的R-Plane暫存器記憶體位址

```
ADCLKS     equ          0ch          ; 取樣頻率設定用
ADPIN      equ          16h          ; 類比輸入腳位用法設定用
```

；定義程式中使用到的F-Plane變數記憶體位址

```
digitalData equ          20h          ; 儲存轉換後的數位資料
                                         ; 轉換後的數位資料有12bits
                                         ; 量測模式一與三取前7個bits使用
                                         ; 量測模式二取前6個bits使用

RRTimes    equ          21h          ; 存放ADCH的右移次數
                                         ; 右移1位，取ADCDATAH前7個bits用
                                         ; 右移2位，取ADCDATAH前6個bits用
```

；定義程式中所使用到的字串

```
CF          defstr       03h, 0      ; 進位旗號

ADCSTART    defstr       11h, 3      ; 類比數位轉換啟用位元

; 系統開機進入點
org         00h

; 初始設定
Start:      movlw        00h          ; 設定ADC Clock為FCPUCLK/256 = 16K Hz
            movwr        ADCLKS      ;

            movlw        ffh         ; 關掉所有七段顯示器
            movwf        PDD         ;
            movwf        PED         ;

SelMode:    btfsc        PBD, 0      ; 檢查量測模式
```

	goto	Mode1	; PB0=1表選擇Mode 1
	bfsc	PBD, 1	;
	goto	Mode2	; PB1=1表選擇Mode 2
	bfsc	PBD, 2	;
	goto	Mode3	; PB2=1表選擇Mode 3
	goto	SelMode	; 沒選擇量測模式，回SelMode重新檢查
; 量測模式一：0V ~10V類比數位轉換			
Mode1:	movlw	001b	;
	movwf	ADCSEL	; 選擇AD1類比輸入
	movlw	11111101b	; 設定AD1腳位用法為類比輸入
	movwr	ADPIN	;
	bsf	ADCSTART	; 啟動類比數位轉換
	call	WaitData	; 等候類比數位轉換完成
	movlw	ffh	; 將左邊七段顯示器及所有小數點關掉
	movwf	PDD	;
	movlw	01h	;
	movwf	RRTimes	; 設定將ADCDATAH的值右移1次
	call	ReadData	; 讀取及顯示轉換結果
	bcf	PDD, 7	; 點亮中間七段顯示器的小數點
	goto	SelMode	; 跳回SelMode
; 量測模式二：0V ~5V類比數位轉換			
Mode2:	movlw	010b	;
	movwf	ADCSEL	; 選擇AD2類比輸入
	movlw	11111011b	; 設定AD2/PB6腳位用法為類比輸入
	movwr	ADPIN	;
	bsf	ADCSTART	; 啟動類比數位轉換
	call	WaitData	; 等候轉換完成及讀取轉換結果
	movlw	ffh	; 將左邊的七段顯示器關掉
	movwf	PDD	;
	movlw	02h	;
	movwf	RRTimes	; 設定將ADCDATAH的值右移2次
	call	ReadData	; 讀取及顯示轉換結果
	bcf	PDD, 7	; 點亮中間七段顯示器的小數點

	goto	SelMode	; 跳回SelMode
; 量測模式三：0V ~ 1V類比數位轉換			
Mode3:	movlw	011b	;
	movwf	ADCSEL	; 選擇AD3作為類比輸入
	movlw	11110111b	; 設定AD3/PB7腳位用法為類比輸入
	movwf	ADPIN	;
	bsf	ADCSTART	; 啟動類比數位轉換
	call	WaitData	; 等候轉換完成及讀取轉換結果
	movlw	11000000b	; 左邊的七段顯示器顯示0
	movwf	PDD	;
	movlw	01h	;
	movwf	RRTimes	; 設定將ADCDATAH的值右移1次
	call	ReadData	; 讀取及顯示轉換結果
	bcf	PDD, 6	; 點亮左邊七段顯示器的小數點
	goto	SelMode	; 跳回SelMode
; 等待類比數位轉換完成			
WaitData:	btfsc	ADCSTART	; 檢查轉換是否完成
	goto	Wait Data	; 轉換尚未完成繼續等
	ret		; 返回呼叫函式
; 讀取轉換結果，將結果轉成BCD碼，以及顯示結果			
ReadData:	; 讀取轉換結果前八個位元		
	movfw	ADCDATAH	;
	movwf	digitalData	;
	; 取出想要的位元(前6個或前7個位元)		
rerrf:	bcf	CF	; 清除進位旗號
	rrf	digitalData	; 右移1位
	decfsz	RRTimes	; RRTimes-1
	goto	rerrf	; RRTimes ≠ 0，繼續右移
; 將結果轉成兩位數BCD碼			
	movfw	digitalData	; 載入要顯示的數位資料
	call	BinToBCD	; 將數位資料轉成兩位數的BCD碼

```

; 顯示結果
movwf    PED    ; 將結果顯示在中間及右邊的七段顯示器
ret      ; 返回呼叫函式

; 將二進位的數位資料轉成兩位數的BCD碼
; 數位資料 0 ~ 63 對應到 00h ~ 50h
; 數位資料 64 ~ 126 對應到 51h ~ 99h
; 數位資料 127 對應到 00h

BinToBCD:  addwf PC,1    ; 以下註解部份為二進位碼
           retlw 00h    ; 0
           retlw 01h    ; 1
           retlw 02h    ; 2
           retlw 02h    ; 3
           retlw 03h    ; 4
           retlw 04h    ; 5
           retlw 05h    ; 6
           retlw 06h    ; 7
           retlw 06h    ; 8
           retlw 07h    ; 9
           retlw 08h    ; 10
           retlw 09h    ; 11
           retlw 09h    ; 12
           retlw 10h    ; 13
           retlw 11h    ; 14
           retlw 12h    ; 15
           retlw 13h    ; 16
           retlw 13h    ; 17
           retlw 14h    ; 18
           retlw 15h    ; 19
           retlw 16h    ; 20
           retlw 17h    ; 21
           retlw 17h    ; 22
           retlw 18h    ; 23
           retlw 19h    ; 24
           retlw 20h    ; 25
           retlw 20h    ; 26
           retlw 21h    ; 27
           retlw 22h    ; 28
           retlw 23h    ; 29
           retlw 24h    ; 30
           retlw 24h    ; 31
           retlw 25h    ; 32
           retlw 26h    ; 33
           retlw 27h    ; 34
           retlw 28h    ; 35
           retlw 28h    ; 36
           retlw 29h    ; 37
           retlw 30h    ; 38
           retlw 31h    ; 39
           retlw 31h    ; 40
    
```

retlw 32h	:	41
retlw 33h	:	42
retlw 34h	:	43
retlw 35h	:	44
retlw 35h	:	45
retlw 36h	:	46
retlw 37h	:	47
retlw 38h	:	48
retlw 39h	:	49
retlw 39h	:	50
retlw 40h	:	51
retlw 41h	:	52
retlw 42h	:	53
retlw 43h	:	54
retlw 43h	:	55
retlw 44h	:	56
retlw 45h	:	57
retlw 46h	:	58
retlw 46h	:	59
retlw 47h	:	60
retlw 48h	:	61
retlw 49h	:	62
retlw 50h	:	63
retlw 50h	:	64
retlw 51h	:	65
retlw 52h	:	66
retlw 53h	:	67
retlw 54h	:	68
retlw 54h	:	69
retlw 55h	:	70
retlw 56h	:	71
retlw 57h	:	72
retlw 57h	:	73
retlw 58h	:	74
retlw 59h	:	75
retlw 60h	:	76
retlw 61h	:	77
retlw 61h	:	78
retlw 62h	:	79
retlw 63h	:	80
retlw 64h	:	81
retlw 65h	:	82
retlw 65h	:	83
retlw 66h	:	84
retlw 67h	:	85
retlw 68h	:	86
retlw 69h	:	87
retlw 69h	:	88
retlw 70h	:	89
retlw 71h	:	90
retlw 72h	:	91
retlw 72h	:	92

```

retlw 73h      ; 93
retlw 74h      ; 94
retlw 75h      ; 95
retlw 76h      ; 96
retlw 76h      ; 97
retlw 77h      ; 98
retlw 78h      ; 99
retlw 79h      ; 100
retlw 80h      ; 101
retlw 80h      ; 102
retlw 81h      ; 103
retlw 82h      ; 104
retlw 83h      ; 105
retlw 83h      ; 106
retlw 84h      ; 107
retlw 85h      ; 108
retlw 86h      ; 109
retlw 87h      ; 110
retlw 87h      ; 111
retlw 88h      ; 112
retlw 89h      ; 113
retlw 90h      ; 114
retlw 91h      ; 115
retlw 91h      ; 116
retlw 92h      ; 117
retlw 93h      ; 118
retlw 94h      ; 119
retlw 94h      ; 120
retlw 95h      ; 121
retlw 96h      ; 122
retlw 97h      ; 123
retlw 98h      ; 124
retlw 98h      ; 125
retlw 99h      ; 126
movlw 39h      ; 127
movwf PDD      ; 將最左邊七段顯示器內容設為1
retlw 00h      ; 並設定中間及右邊七段顯示器內容為0
    
```

## 11. 脈波調變控制之電風扇實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片的脈波調變控制功能(Pulse Width Modulation; PWM)實作電風扇。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	10KΩ排阻(A-type 9PIN)	1
2	10KΩ電阻	1
3	BU806	1
4	直流馬達(5V)	1
5	按壓式按鍵	4

### 實習板模組與 I/O Port：

模組	I/O Port
馬達模組	Port B
指撥模組	Port D

實習說明：

本實習將使用 57FLA80 單晶片所提供的脈波調變控制(PWM)功能實作具三種不同轉速的電風扇。57FLA80 單晶片提供 PWM0P、PWM0N、與 PWM1 三個不同的脈波調變控制功能供使用，脈波調變控制功能可供程式控制調整工作週期(Duty Cycle)，符合本實習需求。本實習使用 PWM0P 來控制電風扇的轉速，作法是利用 PWM0P 輸出接腳(PB1)控制流經電風扇馬達的電流量，給定 PWM0P 較長的工作週期，流經電風扇的電流量大，轉速快，反之，流經電風扇的電流量小，轉速慢。PWM0 的功能方塊圖如圖 11-1 所示：

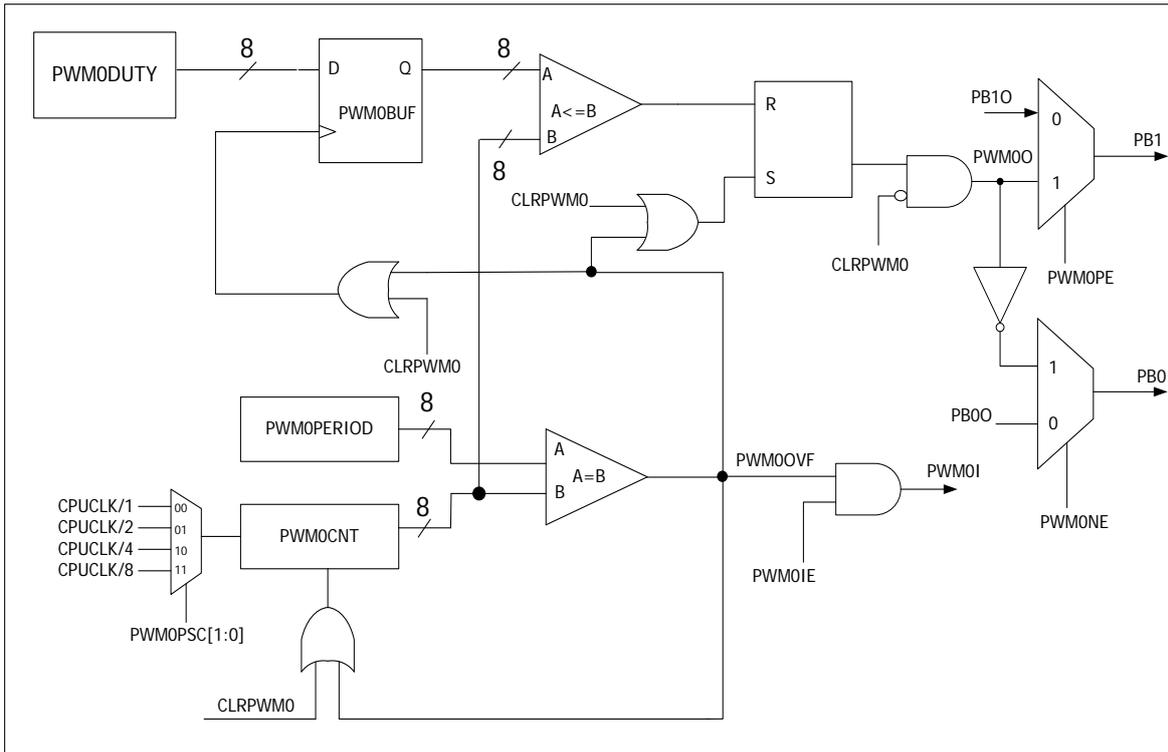


圖 11-1 PWM0 的功能方塊圖

PWM0 功能方塊圖中與 PWM0P 工作週期調整有關的部份主要位於上半部，由 PWM0PE、CLR PWM0、PWM0PSC、PWM0CNT、以及 PWM0DUTY 等控制線控制。

PWM0PE 用來控制 PWM0P/PB1 腳位的用途，當 PWM0PE 的值為 0 時，該腳位作為 PB1 使用，當 PWM0PE 的值為 1 時，該腳位作為 PWM0P 使用；CLR PWM0 用來清除 PWM0 內部狀態以及停住 PWM0；PWM0PSC 用來選擇 PWM0 內部使用的 clock；PWM0CNT 是一個 8 位元的內部暫存器，每經過一個 clock，PWM0CNT 的值會自動加 1，PWM0CNT 的值會隨時跟 PWM0DUTY (8 位元暫存器) 的值比較，比較結果用來控制 RS 正反器，PWM0CNT 的值等於 0 時，正反器的內容會被設定為 1，PWM0CNT 的值大於 PWM0DUTY 時，正反器的內容會被清為 0，PWM0P 的時序圖如圖 11-2 所示。

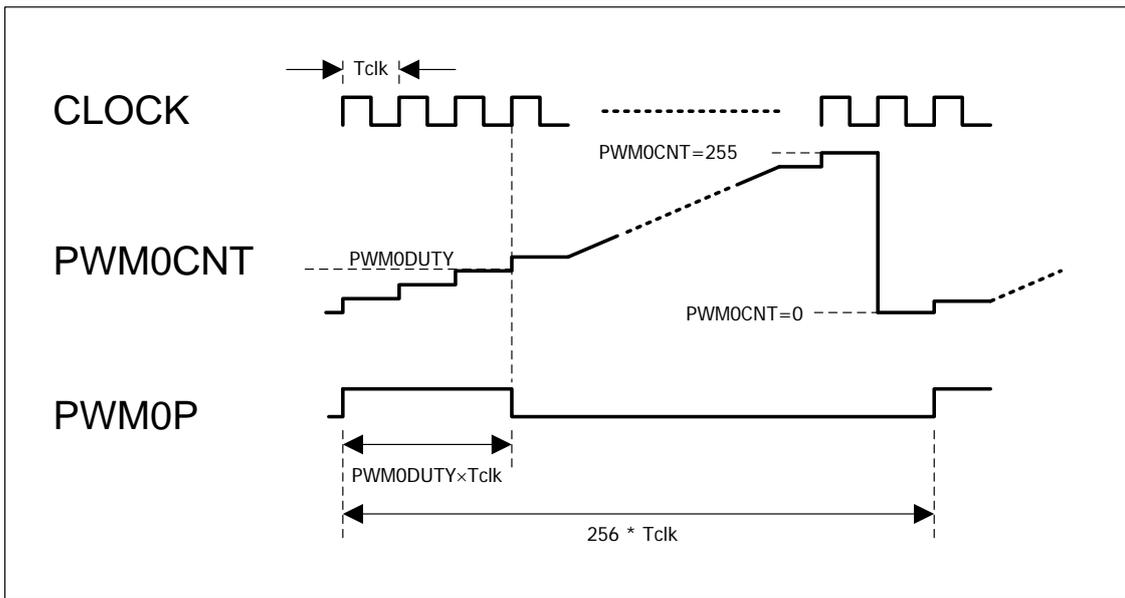


圖 11-2 PWM0P 時序圖

由圖 11-2 可見，PWM0P 輸出訊號的週期時間(cycle time)為  $256 \times T_{clk}$ ，PWM0P 的工作週期則為  $PWM0DUTY \times T_{clk}$ 。PWM0CNT 的值不可由外部設定，要修改 PWM0P 工作週期只能透過設定 PWM0DUTY 的值來達成。57FLA80 單晶片與 PWM0P 控制相關的暫存器整理如下

:

表 11-1 PWM0P 控制相關暫存器

暫存器	說明
PWM0DUTY	位於 F-PLANE 位址 0ch 暫存器，用來設定 PWM0P 輸出訊號的工作週期。PWM0P 工作週期為 $PWM0DUTY \times T_{clk}$ 。PWM0DUTY 預設值為 0。
CLR PWM0	位於 F-PLANE 位址 0dh 暫存器的位元 2，此位元的值為 1 時會清除及停止 PWM0 的功能，PWM0 要正常動作，此位元的值需為 0（預設值為 1）。
PWM0PE	位於 R-PLANE 位址 0bh 暫存器的位元 6，用來設定 PWM0P/PB1 腳位的用途，此位元的值為 1 表 PB1 作為 PWM0P 使用，此位元的值為 0 表 PB1 作為 IO 埠使用（預設值為 0）。
PWM0PSC	位於 R-PLANE 位址 0ch 暫存器的位元 3 ~ 2，用來設定 PWM0 內部使用的 clock。PWM0PSC 設定跟所選用的 clock 關係如下，其中位元排列順序由左而右分別是位元 3 ~ 2， $F_{CPUCLK}$ 表單晶片的 CPU Clock。（預設值為 00） 00: $F_{CPUCLK}$ 01: $F_{CPUCLK} / 2$ 10: $F_{CPUCLK} / 4$ 11: $F_{CPUCLK} / 8$

硬體線路圖：

本實習完整硬體線路如圖 11-3 所示，分別使用 Port D 的 PD0 ~ PD3 用來讀取高速、中速、低速、與停止等按鍵，以及使用 PWM0P/PB1 腳位作為脈波調變控制輸出，PWM0P 的輸出接到 BU806，控制流經電風扇的電流量。其中 BU806 是一個高電壓的高速達靈頓 NPN 電晶體，Vce 電壓範圍為 1.5V~330V，切換時間小於 1 微秒( $\mu s$ )，BU806 等效電路如圖 11-4 所示，

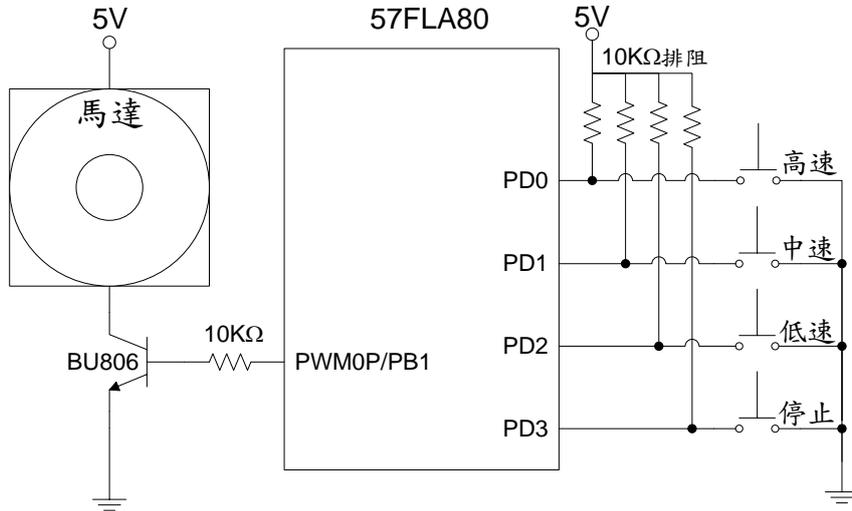


圖 11-3 電風扇硬體線路圖

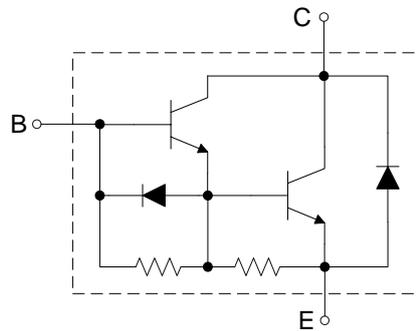


圖 11-4 BU806 等效電路

程式流程：

本實習程式流程如圖 11-5 所示，首先進行初始設定，啟用 PWM0P 以及設定 PWM0 的 clock，接著檢查按鍵是否被按，若有按鍵被按則依被按下的按鍵調整 PWM0 設定並等待按鍵放開。

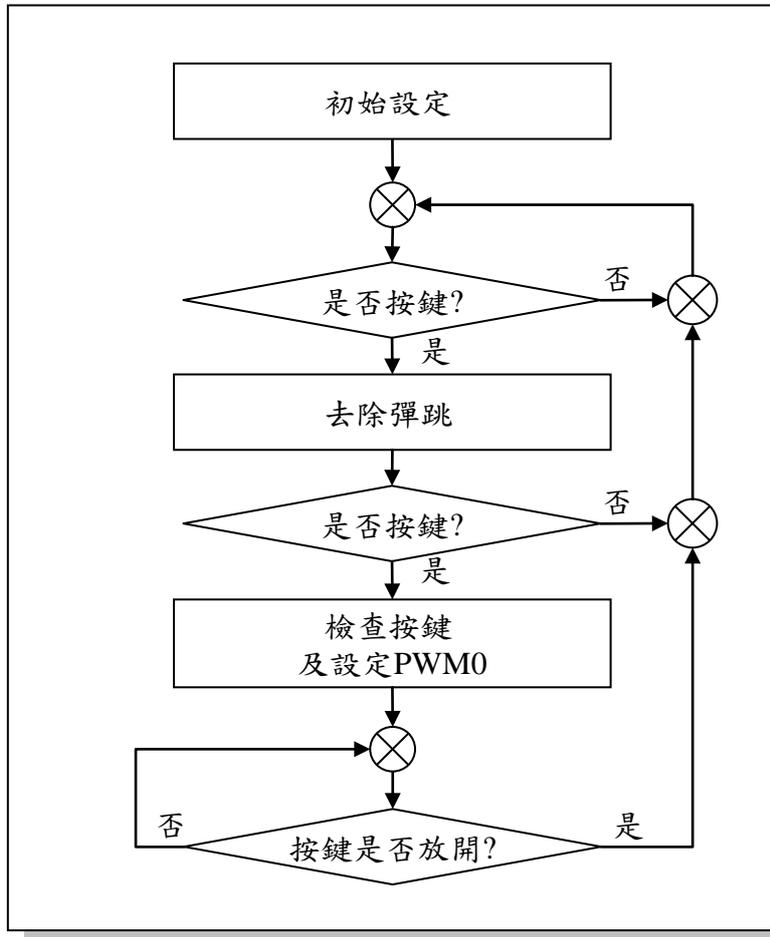


圖 11-5 電風扇主程式流程圖

## 程式碼及程式說明：

; 定義程式中使用到的F-Plane暫存器記憶體位址

```
PDD          equ          12h          ; Port D
PWM0DUTY     equ          0ch          ; 設定PWM0P輸出訊號的工作週期用
```

; 定義程式中使用到的R-Plane暫存器記憶體位址

```
PBE          equ          06h          ; 設定Port B各腳位的輸出方式
                                           ; bits 0~7分別代表PB0~PB7腳位
                                           ; bit值為0: 使用Open-Drain輸出方式
                                           ; bit值為1: 使用CMOS Push-Pull輸出方式
```

```
PWM0PE       equ          0bh          ; PWM0PE所在的暫存器位址
PWM0PSC      equ          0ch          ; 選擇PWM0 clock用
```

; 定義程式中使用到的F-Plane變數記憶體位址

```
keyStatus    equ          20h          ; 按鍵狀態
R1           equ          21h          ; 時間延遲迴圈控制變數
R2           equ          22h          ; 時間延遲迴圈控制變數
```

; 定義程式中所使用到的常數

```
H_SPEED      equ          c0h          ; 工作週期佔75%
M_SPEED      equ          a0h          ; 工作週期佔63%
L_SPEED      equ          80h          ; 工作週期佔50%
```

; 定義程式中所使用到的字串

```
CLRPWM0      .defstr        0dh, 2    ; PWM0清除位元
```

; 系統開機進入點

```
org          00h
```

; 初始設定

```
Start:       movlw         02h          ;
              movwr        PBE          ; 將PB1設為CMOS push-pull output
              movlw         43h          ;
              movwr        PWM0PE       ; 將PWM0PE設為1，其餘位元用預設值
              movlw         00h          ; 選擇CPU clock作為PWM0的clock
              movwr        PWM0PSC      ; 選擇PWM0的clock
```

```

; 檢查是否有按鍵被按下
CheckKey: movlw 0fh ;
movwf PDD ; 將1寫入PD0~PD3, 清除按鍵狀態
movfw PDD ; 讀取鍵盤狀態, 沒按鍵應是0FH
addlw f0h ; +F0H
movwf keyStatus ; 取得按鍵狀態, 沒按鍵是0FH+F0H=FFH
incfsz keyStatus, 1 ; 若按鍵狀態+1等於0, 表示沒按鍵
goto DeNoise ; 有按鍵, 去除彈跳
goto CheckKey ; 沒按鍵, 到CheckKey檢查是否有按鍵

Key1: btfsc PDD, 0 ; 檢查高速鍵是否被按下
goto Key2 ; 高速鍵沒被按下, 檢查其它按鍵
movlw H_SPEED ; 設定高速運轉工作周期, 出力75%
movwf PWMODUTY ; 將工作週期寫入PWMODUTY
bcf CLRPWM0 ; 啟動PWM0
goto WaitRelease ; 等待按鍵放開

Key2: btfsc PDD, 1 ; 檢查中速鍵是否被按下
goto Key3 ; 中速鍵沒被按下, 檢查其它按鍵
movlw M_SPEED ; 設定中速運轉工作周期, 出力63%
movwf PWMODUTY ; 將工作週期寫入PWMODUTY
bcf CLRPWM0 ; 啟動PWM0
goto WaitRelease ; 等待按鍵放開

Key3: btfsc PDD, 2 ; 檢查低速鍵是否被按下
goto Key4 ; 低速鍵沒被按下, 停止鍵被按下
movlw L_SPEED ; 設定低速運轉工作周期, 出力50%
movwf PWMODUTY ; 將工作週期寫入PWMODUTY
bcf CLRPWM0 ; 啟動PWM0
goto WaitRelease ; 等待按鍵放開

Key4: bsf CLRPWM0 ; 停止鍵被按下, 關閉PWM0

; 等待使用者放開按鍵
WaitRelease: movlw 0fh ; 將0fh寫入PDD
movwf PDD ;
movfw PDD ; 讀取鍵盤狀態, 若放開按鍵應是0fh
addlw f0h ; +f0h
movwf keyStatus ; 取得按鍵狀態, 若放開按鍵應是ffh
incfsz keyStatus, 1 ; 若按鍵狀態+1等於0, 表示放開按鍵
goto WaitRelease ; 沒放開按鍵, 繼續等待
goto CheckKey ; 沒按鍵, 到CheckKey檢查是否有按鍵

; 去除彈跳
    
```

```

DeNoise:    call    delay    ; 等待5ms
            movlw   0fh      ; 將1寫入PD0~PD3, 清除按鍵狀態
            movwf  PDD      ;
            movfw  PDD      ; 讀取鍵盤狀態, 沒按鍵應是0FH
            addlw  f0h      ; +F0H
            movwf  keyStatus ; 取得按鍵狀態, 沒按鍵是FFH
            incfsz keyStatus, 1 ; 若按鍵狀態+1等於0, 表示沒按鍵
            goto   Key1     ; 有按鍵, 檢查按了什麼鍵
            goto   CheckKey ; 沒按鍵, 到CheckKey檢查是否有按鍵

; 延遲0.005秒副程式

delay:      movlw   10      ; 設定外層迴圈執行10次
            movwf  R1      ;

            ; 外層迴圈
delay_L1:   movlw   200     ; 設定內層迴圈執行200次
            movwf  R2      ;

            ; 內層迴圈: 迴圈跑一次約消耗5個指令週期
delay_L2:   nop           ; 消耗1個指令週期
            nop           ; 消耗1個指令週期
            decfsz  R2, 1  ; 約消耗1個指令週期
            goto   delay_L2 ; 消耗2個指令週期

            decfsz  R1, 1  ; 將R1減1, 若R1=0離開迴圈
            goto   delay_L1 ;

            ret          ; 返回主程式

```

## 12. 文字型 LCD 顯示器實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片控制文字型 LCD 顯示器。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	10KΩ排阻(A-type 9PIN)	2
2	HD44780 相容文字型 LCM (可顯示兩列文字，一列 20 字)	1

### 實習板模組與 I/O Port：

模組	I/O Port
文字型 LCD 顯示模組	Port B
文字型 LCD 顯示模組	Port D

**實習說明：**

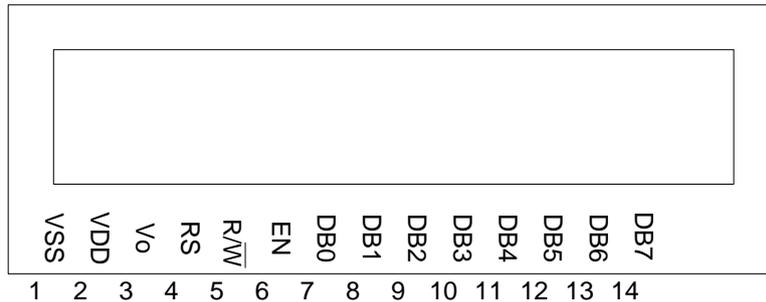
本實習將實作一個文字型 LCD（液晶顯示器）顯示器，該顯示器可顯示兩列英文字串。HD44780 是日立(Hitachi)公司開發的一顆標準文字型 LCD 控制 IC，該 IC 提供內建的 ASCII 字型，放在其字元產生器(Character Generator ROM；CG ROM)唯讀記憶體內，使用時只要將 ASCII Code 寫入 LCD 控制 IC 的顯示資料記憶體(Display Data RAM；DD RAM)內，該 IC 便會將字型顯示在 LCD 上。此外，該 IC 也提供使用者定義幾個自己的字型放在 CG RAM 中。本實習僅使用其內建字型，因此不介紹自行定義字型的作法。

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表 12-1 所列：

**表 12-1 LCD 顯示位置與 DD RAM 位址對映關係**

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

許多 LCM(液晶顯示模組)具備 HD44780 相容功能，提供相似的控制介面，圖 12-1 所示為一 HD44780 相容的 LCM 腳位圖。



**圖 12-1 HD44780 相容 LCM 腳位圖**

HD44780 相容 LCM 腳位說明如表 12-2 所示。

表 12-2 HD44780 相容 LCM 腳位說明

腳位	腳位說明
V <sub>SS</sub>	接地
V <sub>DD</sub>	電源(+5V)
V <sub>0</sub>	明暗控制，V <sub>0</sub> =V <sub>SS</sub> 對比最大，V <sub>0</sub> =V <sub>DD</sub> 對比最小
R/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

為了跟單晶片溝通，HD44780 相容 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單晶片送來的命令，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 相容 IC 可接受的命令共有 11 種，分別說明如下，傳送以下命令給 HD44780 相容 IC 時 EN 腳位必須設定為 1，命令傳送完成，EN 腳位必須設定為 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	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	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 為資料讀寫方向，S 表顯示內容是否位移，I/D 與 S 的值與相關作用說明如下：

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

控制顯示器(D)，游標(C)，與游標字元閃爍(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	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	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	R/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	R/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	$\overline{R/W}$	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	$\overline{R/W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

HD44780 相容 IC 所有命令整理如下：

表 12-3 HD44780 相容 LCM 命令列表

命令	命令內容									
清除顯示內容	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	0	1
返回左上角	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	0	1	x
設定輸入模式	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	0	1	I/D	S
顯示器開關控制	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	0	1	D	C	B
游標或顯示內容位移	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	0	1	S/C	R/L	X	X
功能設定	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	0	1	DL	N	F	X	X
設定 CG RAM 位址	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	0	0	1	A5	A4	A3	A2	A1	A0
設定 DD RAM 位址	RS	R $\bar{W}$	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	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	0	1	BF	A6	A5	A4	A3	A2	A1	A0
寫入 CG 或 DD RAM	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	1	0	D7	D6	D5	D4	D3	D2	D1	D0
讀取 CG 或 DD RAM	RS	R $\bar{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	1	1	D7	D6	D5	D4	D3	D2	D1	D0

硬體線路圖：

本實習硬體線路圖如圖 12-2 所示，本實習分別使用 Port D 的 PD5、PD6、與 PD7 控制 LCM 模組的 RS、 $\overline{R/W}$ 、與 EN 等腳位，以及使用 Port B 來存取 LCM 的命令與資料。

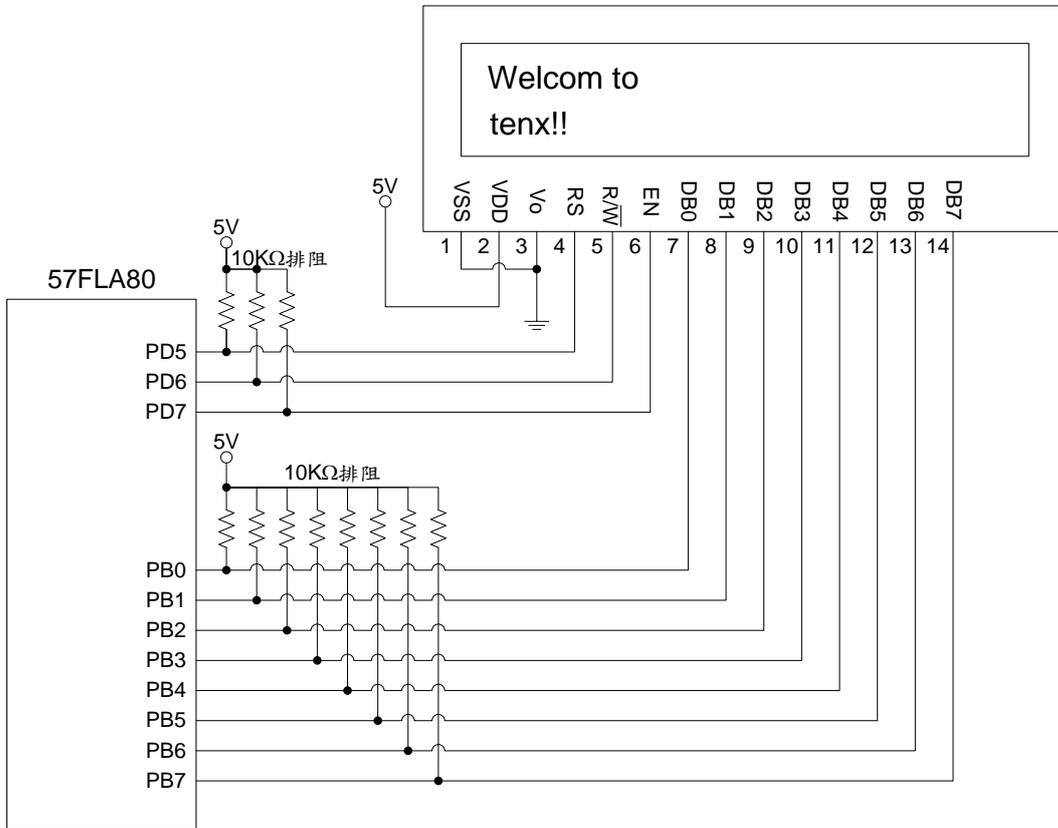
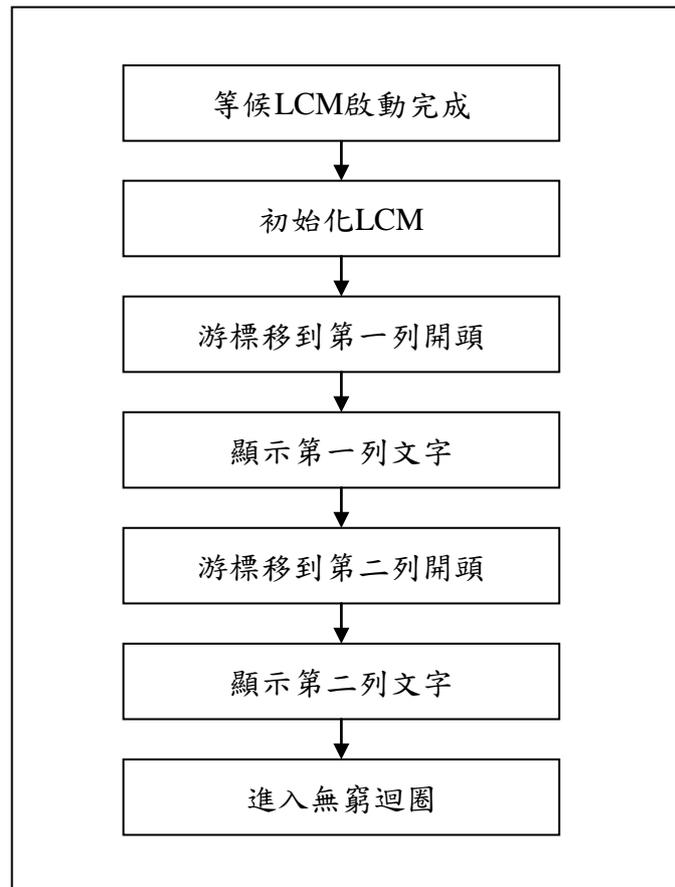


圖 12-2 文字型 LCD 顯示器硬體線路圖

**程式流程：**

本實習的程式很簡單，只是要在 LCD 上顯示兩行文字，然後停止程式，程式流程如圖 12-3 所示，首先等候 LCM 完成開機動作，然後對 LCM 進行初始化動作，接著將兩個預設的字串分別寫入 LCD 的第一列與第二列後，程式便進入無窮迴圈。其中，要特別注意的是，在進行 LCM 初始化設定時，為了能夠穩定完成設定，最好先進行三次或三次以上的命令寫入動作以及等候適當的時間延遲，然後再開始真正的 LCM 初始化動作。

**圖 12-3 文字型 LCD 顯示器**

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址
PC          equ          02h          ; Program Counter
PBD         equ          06h          ; Port B

; 定義程式中使用到的F-Plane變數記憶體位址
data        equ          20h          ; 儲存要寫入LCM的資料
strIndex    equ          21h          ; 記錄字元在字串中的位置資訊
R1          equ          22h          ; 迴圈控制變數
R2          equ          23h          ; 迴圈控制變數

; 定義程式中所使用到的字串
RS_LCM     .defstr       12h, 5      ; LCM RS控制位元
RW_LCM     .defstr       12h, 6      ; LCM RW控制位元
EN_LCM     .defstr       12h, 7      ; LCM EN控制位元

; 系統開機進入點
org        00h

; 主程式
Start:     call          Delay        ; 等候LCM啟動

          call          LCMInitialize ; 進行LCM初始設定

          call          MoveToRow1    ; 將游標移至LCD第一行開頭
          call          DisplayRow1   ; 顯示第一行字串
          call          MoveToRow2    ; 將游標移至LCD第二行開頭
          call          DisplayRow2   ; 顯示第二行字串

          goto         $              ; 程式進入無窮迴圈

; LCM初始設定副程式
LCMInitialize:
; 進行三次命令寫入動作及等候適當時間延遲，讓系統穩定
movlw     30h          ; 功能設定：設定DL=N=F=0
movwf    data          ; 載入命令碼
call     WriteIR       ; 將命令碼寫到LCM

movlw     30h          ; 功能設定：設定DL=N=F=0
movwf    data          ; 載入命令碼
call     WriteIR       ; 將命令碼寫到LCM

movlw     30h          ; 功能設定：設定DL=N=F=0
movwf    data          ; 載入命令碼

```

	call	WriteIR	; 將命令碼寫到LCM
	movlw	38h	; 功能設定：設定DL=N=1, F=0
	movwf	data	; 載入命令碼
	call	WriteIR	; 將命令碼寫到LCM
	movlw	08h	; 顯示器控制：D=C=B=0
	movwf	data	; 載入命令碼
	call	WriteIR	; 將命令碼寫到LCM
	movlw	01h	; 清除顯示
	movwf	data	; 載入命令碼
	call	WriteIR	; 將命令碼寫到LCM
	movlw	06h	; 模式設定：I/D=1, S=0
	movwf	data	; 載入命令碼
	call	WriteIR	; 將命令碼寫到LCM
	movlw	0eh	; 顯示器控制：D=C=1, B=0
	movwf	data	; 載入命令碼
	call	WriteIR	; 將命令碼寫到LCM
	ret		
;將命令碼寫入LCM指令暫存器副程式			
WriteIR:	bcf	RS_LCM	; 選擇指令暫存器
	bcf	RW_LCM	; 選擇進行寫入動作
	bsf	EN_LCM	; 開啟LCM讀寫功能
	movfw	data	; 載入命令碼
	movwf	PBD	; 送出命令碼
	call	Delay	; 等候一段時間，讓資料完成寫入動作
	bcf	EN_LCM	; 關閉LCM讀寫功能
	ret		; 返回呼叫函式
;將資料寫入LCM記憶體副程式			
WriteDR:	bsf	RS_LCM	; 選擇記憶體
	bcf	RW_LCM	; 選擇進行寫入動作
	bsf	EN_LCM	; 開啟LCM讀寫功能
	movfw	data	; 載入資料
	movwf	PBD	; 送出資料
	call	Delay	; 等候一段時間，讓資料完成寫入動作
	bcf	EN_LCM	; 關閉LCM讀寫功能
	ret		; 返回呼叫函式
; 將游標位置移到第一列開頭副程式			
MoveToRow1:	movlw	80h	; 設定DD RAM位址為00H
	movwf	data	; 將命令碼寫入data變數
	call	WriteIR	; 將命令碼寫到LCM
	ret		; 返回呼叫函式

```

; 顯示第一行字串副程式副程式
DisplayRow1: movlw      00h          ; 由字串第0個字元開始處理
              movwf     strIndex    ;
              ;

DR1_Next:    movfw     strIndex      ; 取出字元的位置
              call      String1     ; 取出字元的ASCII Code
              movwf     data        ; 將字元的ASCII Code寫入data變數
              incfsz    data, 0     ; 若字元的ASCII Code為ffh結束函式
              goto      DR1_Show    ; 將字元的ASCII Code寫入LCM記憶體
              goto      DR1_End     ; 結束函式
DR1_Show:    call      WriteDR      ; 將字元的ASCII Code寫入LCM記憶體
              incf      strIndex    ; 將字元位置+1
              goto      DR1_Next    ; 處理下個字元
DR1_End:     ret                    ; 返回呼叫函式

; 將游標位置移到第二列開頭副程式
MoveToRow2: movlw      c0h          ; 設定DD RAM位址為40H
              movwf     data        ; 將命令碼寫入data變數
              call      WriteIR     ; 將命令碼寫到LCM
              ret                    ; 返回呼叫函式

; 顯示第二行字串副程式
DisplayRow2: movlw      00h          ; 由字串第0個字元開始處理
              movwf     strIndex    ;
              ;

DR2_Next:    movfw     strIndex      ; 取出字元的位置
              call      String2     ; 取出字元的ASCII Code
              movwf     data        ; 將字元的ASCII Code寫入data變數
              incfsz    data, 0     ; 若字元的ASCII Code為ffh結束函式
              goto      DR2_Show    ; 將字元的ASCII Code寫入LCM記憶體
              goto      DR2_End     ; 結束函式
DR2_Show:    call      WriteDR      ; 將字元的ASCII Code寫入LCM記憶體
              incf      strIndex    ; 將字元位置+1
              goto      DR2_Next    ; 處理下個字元
DR2_End:     ret                    ; 返回呼叫函式

; 第一個字串
String1:     addwfw    PC, 1         ; 跳到第i行，i=working register的內容
              retlw     57h         ; W
              retlw     65h         ; e
              retlw     7ch         ; l
              retlw     63h         ; c
              retlw     6fh         ; o
              retlw     6dh         ; m

```

```

        retlw    65h        ; e
        retlw    20h        ;
        retlw    74h        ; t
        retlw    6fh        ; o
        retlw    ffh        ; 字串結尾

; 第二個字串
String2:  addwf    PC, 1      ; 跳到第i行，i=working register的內容
        retlw    74h        ; t
        retlw    65h        ; e
        retlw    6eh        ; n
        retlw    78h        ; x
        retlw    21h        ; !
        retlw    21h        ; !
        retlw    ffh        ; 字串結尾

; 延遲0.015秒副程式
Delay:    movlw    30        ; 設定外層迴圈執行30次
        movwf    R1        ;

        ; 外層迴圈
delay_L1: movlw    200       ; 設定內層迴圈執行200次
        movwf    R2        ;

        ; 內層迴圈：迴圈跑一次約消耗5個指令週期
delay_L2: nop            ; 消耗1個指令週期
        nop            ; 消耗1個指令週期
        decfsz    R2, 1    ; 約消耗1個指令週期
        goto     delay_L2 ; 消耗2個指令週期

        decfsz    R1, 1    ; 將R1減1，若R1=0離開迴圈
        goto     delay_L1 ;
        ret

```

### 13. UART 與 RS232 傳輸實習

#### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片所提供的 UART 介面進行 RS232 傳輸。

#### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

#### 實習材料：

項次	品名	數量
1	10KΩ排阻(A-type 9PIN)	2
2	20KΩ排阻(A-type 9PIN)	2
3	22μF 電容	4
4	文字型 LCM	1
5	4 × 4 鍵盤	1
6	ICL232 IC	1
7	RS232 連接器(9PIN 母頭)	1
8	RS232 連線(9PIN 公頭)	1

#### 實習板模組與 I/O Port：

模組	I/O Port
文字型 LCD 顯示模組	Port B
文字型 LCD 顯示模組	Port D
4 × 4 鍵盤模組	Port E
RS232 串列傳輸模組	Port G

**實習說明：**

本實習將利用 RS232 介面進行個人電腦與單晶片之間的資料傳輸。以下將介紹 RS232 以及 57FLA80 單晶片所提供的 RS232 功能。

RS232 是美國電子工業聯盟 (Electronic Industry Association ; EIA) 所提出的一個建議標準(Recommended standard)，該標準制定了序列資料傳輸介面，內容包括電氣特性、通訊協定、纜線、以及連接頭等規範。一般個人電腦上所提供的 COM port 即支援 RS232 標準，提供電腦與其它電腦或周邊裝置之間的資料傳輸能力。

RS232 提供許多可用的連接器，最常用的是九針(PIN)的 D 型連接器，九針 D 型連接器外型及接腳如圖 13-1 所示：

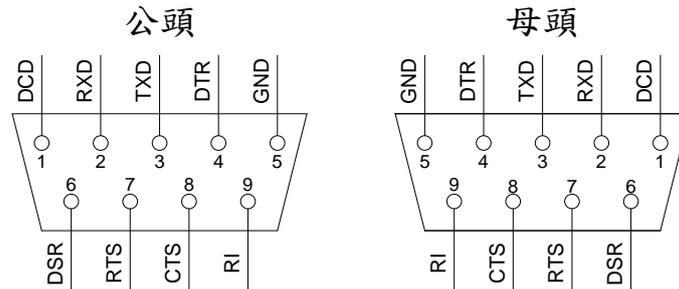


圖 13-1 九針 RS232 D 型連接器

九針 D 型連接器各腳位功能說明如下：

表 13-1 九針 D 型連接器各腳位功能說明

腳位	腳位說明
1	資料載波偵測(Data Carrier Detection ; DCD)
2	接收資料(Receive Data; RxD)
3	傳送資料(Transmit Data; TxD)
4	資料終端備妥(Data Terminal Ready ; DTR)
5	接地(Ground ; GND)
6	資料備妥(Data Set Ready ; DSR)
7	要求發送資料(Request To Send ; RTS)
8	清除發送資料(Clear To Send ; CTS)
9	響鈴指示(Ring Indicator ; RI)

由以上接腳資料得知，假設裝置一與裝置二要使用 RS232 互相傳送資料，則裝置一的 1, 5, 9 等腳位分別要接到裝置二的 1, 5, 9 等腳位，裝置一與裝置二的腳位 2, 3、腳位 4, 6、與腳位 7, 8 要跳接，也就是裝置一的第 2, 3, 4, 6, 7, 8 腳位要分別跟裝置二的第 3, 2, 6, 4, 8, 7 腳位互接。

一般使用 RS232 傳送資料，通常不需要使用到全部腳位，最簡單的用法只需用到 RxD、TxD、與 GND 等 3 個腳位，用法如圖 13-2 所示，這種用法稱為 null Modem without hand shaking 方法，是一般最常見的用法，也是本實習使用的方法。

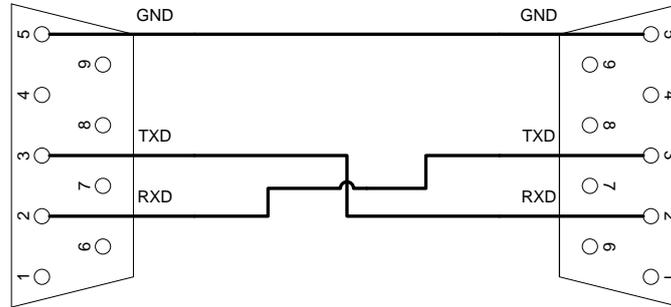


圖 13-2 Null Modem without hand shaking

傳送資料的兩端使用圖 13-2 的方法連接後，便可使用 RS232 序列傳輸方式互傳資料，RS232 序列傳輸方式傳輸速度以每秒可傳送多少個 Baud 作為傳輸速度的單位，一個 Baud 代表一個符號或一個脈衝，由於 RS232 傳輸的是二進位的訊號，因此一個 Baud 等於一個位元，一秒可傳送  $N$  個位元，Baud Rate 為  $N$  bps (bits per second)。一般常用的 Baud Rate 有：1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 等幾種，可依系統需求選擇合適的傳輸速度。

使用 RS232 傳送資料，資料必須封裝成如圖 13-3 所示的資料封包(frame)傳送：

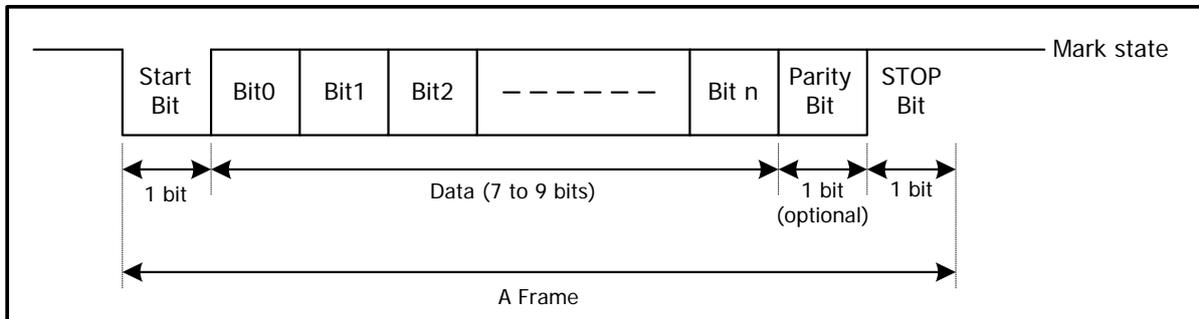


圖 13-3 RS232 資料封裝格式

RS232 的資料封包(frame)內含 Start bit、Data、Parity Bit、與 Stop Bit 等欄位，這些欄位功能說明如下：

表 13-2 RS232 資料封包欄位功能說明

欄位	欄位說明
Start Bit	表示要開始傳送資料，RS232 的 TXD 腳位不傳送資料時，會維持在一定電位（一般是低電位），傳送資料前改變 TXD 電位，用來通知對方要開始傳送資料。
Data	用來放置要傳送的資料，RS232 一個封包可傳送 7 ~ 9 個位元。
Parity Bit	此位元用來傳送 Data 中 1 的個數資訊，可供用來檢查接收到的資料是否正確。此位元非必要，系統設計者可依需求決定是否使用此位元。 同位檢查有奇同位與偶同位兩種模式。在奇同位模式下，當 Data 中 1 的個數是奇數時，同位位元為 0，否則為 1；在偶同位模式下，當 Data 中 1 的個數是奇數時，同位位元為 1，否則為 0。
Stop Bit	資料傳送完畢，將 TXD 腳位回覆到不傳送資料的狀態。

57FLA80 單晶片內建萬用序列非同步傳輸(UART)介面，支援 RS232、RS422、RS423、RS449、與 RS485 等多種序列傳輸標準，功能方塊圖如下所示，其中上半部負責資料傳送，下半部負責資料接收。

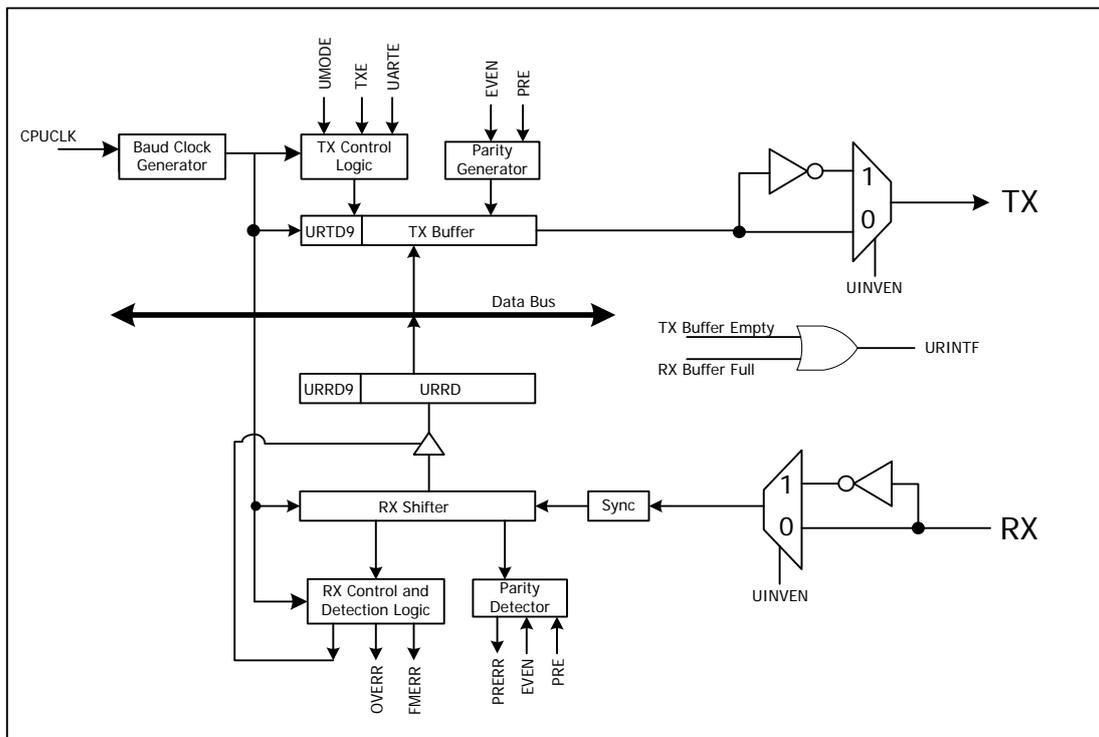


圖 13-4 57FLA80 單晶片 UART 功能方塊圖

使用 UART 收送資料前，必須先將 UARTE 設定為 1，啟用 UART 功能，再設定 Baud Rate 與封包格式，57FLA80 單晶片 Baud Rate (單位：bps)與 Baud Clock Generator 產生的 Baud clock 有關，Baud Clock 可透過 UBAUD 暫存器控制，UBAUD 內容與 Baud Rate 關係如下：

$$\text{Baud Rate} = \frac{F_{\text{CPUCLK}}}{2 * 16 * \text{UBAUD}} \quad (13-1)$$

其中，F<sub>CPUCLK</sub> 是 CPU 的 clock。當使用 4 MHz 的 CPU clock 時，常用的 Baud Rate 以及相關 UBAND 設定如下表所示。

表 13-3 常用的 Baud Rate 以及 UBAND 設定

Baud Rate (bps)	UBAUD 內容
1200	104
2400	52
4800	26
9600	13

Baud clock 產生出來後，會傳送給 TX 與 RX 控制電路以及移位緩衝器(shifter buffers)作為資料收送同步訊號。

在封包格式方面，57FLA80 單晶片提供 5 種不同的封包格式供選用，封包格式的選用可透過設定 UMODE 與 PRE 的內容來達成，UMODE 與 PRE 內容跟封包格式對應關係如圖 13-5 所示。

UMODE[1:0]	PRE	S   1   2   3   4   5   6   7   8   9   10
0 0	0	7-bit data STOP
0 0	1	7-bit data Pty STOP
0 1	0	8-bit data STOP
0 1	1	8-bit data Pty STOP
1 0	X	9-bit data STOP

圖 13-5 封包格式指定方式

同位模式可用 EVEN 位元指定，當 EVEN 位元為 1 時，表示使用偶同位，否則，使用奇同位。此外，如果要輸出反相訊號（如圖 13-6 所示）的話，可將 UINVEN 設為 1，否則將 UINVEN 設為 0。

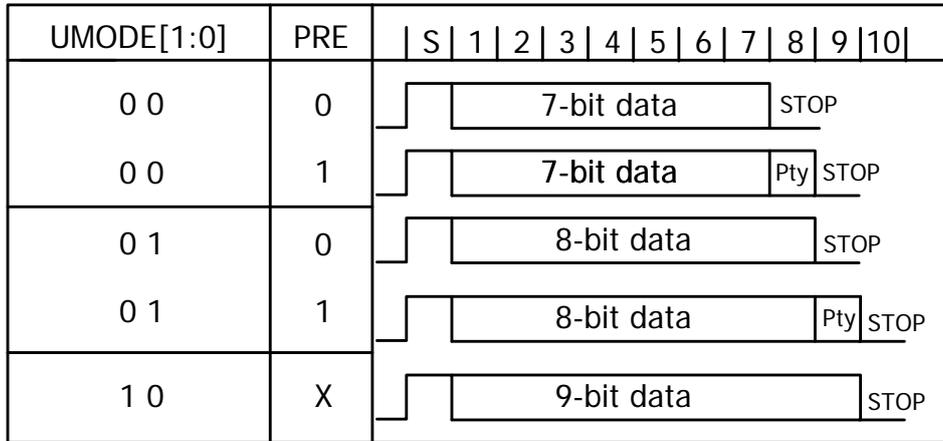


圖 13-6 反相輸出結果

完成 Baud Rate 跟封包格式設定後，TDX/PG4 與 RXD/PG5 接腳便可用來傳送及接收序列資料，要傳送資料，必須將 TXE 設定為 1，接著只要將要傳送的位元組寫入 UR9TD (要傳送的第 9 個位元) 與 URDATA 暫存器，硬體便會自動將資料傳送出去，資料傳送過程中，可透過 UTBE 的值檢查資料是否傳送完成，傳送未完成 UTBE 為 0，傳送完成 UTBE 為 1。

要接收資料，必須將 RXE 設定為 1，資料傳送過來後會儲存在 UR9RD (接收到的第 9 個位元) 與 URDATA 暫存器內，而且會將 URBF 的值設為 1，URDATA 的值被讀取後，URBF 的值會自動被清為 0，透過 URBF 的值，可以了解是否有資料尚未被讀取。另外，為了解接收到的資料是否正確，硬體會自動檢查資料接收過程中是否有錯誤發生，並設定適當的錯誤位元。錯誤位元有 FMERR、PRERR、與 OVERR 等三種，FMERR 表示沒有收到封包的 Stop Bit，PRERR 表同位位元錯誤，OVERR 表示之前收到的資料還沒被讀取即被新進資料覆蓋掉的錯誤。

資料傳送接收過程中，如果 URTIE 位元設定為 1，則資料收送完成，會自動發出中斷，中斷發生後，URTIF 旗號會設定為 1，中斷處理完成，可透過韌體將 URTIF 清為 0。

57FLA80 單晶片 UART 相關暫存器整理如下：

表 13-4 UART 相關暫存器

暫存器	說明
URTIE	位於 F-PLANE 位址 0eh 暫存器位元 6，用來設定 UART 收送完成是否產生中斷，1：啟用，0：關閉(預設值)。
URTIF	位於 F-PLANE 位址 0fh 暫存器位元 6，UART 中斷旗號，收送完成，硬體自動將 URTIF 設為 1。
UINVEN	位於 F-PLANE 位址 17h 暫存器位元 0，設定收送訊號是否反相，1：反相，0：不反相(預設值)。
UARTE	位於 F-PLANE 位址 17h 暫存器位元 1，用來設定是否啟用 UART 功能，1：啟用，0：關閉(預設值)。
RXE	位於 F-PLANE 位址 17h 暫存器位元 2，用來設定是否啟用 UART 接收功能，1：啟用，0：關閉(預設值)。
TXE	位於 F-PLANE 位址 17h 暫存器位元 3，用來設定是否啟用 UART 傳送功能，1：啟用，0：關閉(預設值)。
UMODE	位於 F-PLANE 位址 17h 暫存器位元 6~5，用來選擇 UART 模式。00：7 bits(預設值)，01：8 bits，10：9 bits。
URTD9	位於 F-PLANE 位址 17h 暫存器位元 7，存放要傳送資料的第 9 個位元。
URBF	位於 F-PLANE 位址 18h 暫存器位元 0，資料接收完畢自動設為 1，讀取 URDATA 自動清為 0。
FMERR	位於 F-PLANE 位址 18h 暫存器位元 1，封包錯誤硬體自動設為 1，需靠韌體清為 0。
OVERR	位於 F-PLANE 位址 18h 暫存器位元 2，接收資料覆蓋到前一筆資料硬體自動設為 1，需靠韌體清為 0。
PRERR	位於 F-PLANE 位址 18h 暫存器位元 3，同位位元錯誤自動設為 1，需靠韌體清為 0。
PRE	位於 F-PLANE 位址 18h 暫存器位元 4，用來設定是否使用同位位元，1：啟用，0：關閉(預設值)。
EVEN	位於 F-PLANE 位址 18h 暫存器位元 5，用來設定是否使用偶同位，1：偶同位，0：奇同位(預設值)。
URRD9	位於 F-PLANE 位址 18h 暫存器位元 6，存放接收到資料的第 9 個位元。
UTBE	位於 F-PLANE 位址 18h 暫存器位元 7，用來表示傳送是否完成，1：傳送完成(預設值)，0：資料傳送中。
URDATA	位於 F-PLANE 位址 19h 暫存器，用來存放要傳送或接收到的資料。
UBAUD	位於 R-PLANE 位址 17h 暫存器，用來設定 UART baud rate 用。Baud rate 設定公式： $\text{Baud Rate} = \frac{F_{\text{CPUCLK}}}{2 * 16 * \text{UBAUD}}$

RS232 與單晶片使用不同的電壓表示高低電位 (1 與 0 的電壓)，RS232 使用+12V 與-12V，單晶片則使用+5V 與 0V，因此，單晶片的資料無法直接透過 RS232 介面傳輸。要將單晶片的資料透過 RS232 傳輸，在傳送資料前，必須先將單晶片輸出資料的高低電位調整成 RS232 需要的高低電位，同樣的，接收到透過 RS232 介面傳送過來的資料後，也必須將 RS232 接收到資料的高低電位調整成單晶片適合的高低電位。

目前市面上有許多 IC 可提供以上所述的電壓轉換功能，ICL232 是 InterSil 公司所推出的 RS232 傳送接收 IC，內建兩組 RS232 傳送接收介面。圖 13-7 所示為 ICL232 IC 的腳位圖。

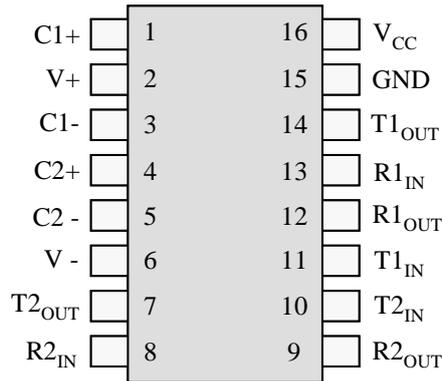


圖 13-7 ICL232 IC 腳位圖

ICL232 IC 腳位說明如表 13-5 所示。

表 13-5 ICL232 IC 腳位說明

腳位	腳位說明
C1+	電壓加倍用外部電容“+”極。
V+	內部產生的+10V。
C1-	電壓加倍用外部電容“-”極。
C2+	電壓改變(inverter)用外部電容“+”極。
C2-	電壓改變(inverter)用外部電容“-”極。
V-	內部產生的-10V。
T2OUT	第 2 組 RS-232，用來將資料傳送到 RS-232，需連接至 RS232 連接器的 TXD，輸出電壓±10V。
R2IN	第 2 組 RS-232，用來接收 RS-232 的資料，需連接至 RS232 連接器的 RXD。
R2out	第 2 組 RS-232，用來將 RS232 收到的資料輸出至單晶片，需連接至單晶片的 RXD，輸出電壓 5V 與 0V。
T2IN	第 2 組 RS-232，用來接收單晶片的資料，需連接至單晶片的 TXD。
T1IN	第 1 組 RS-232，用來接收單晶片的資料，需連接至單晶片的 TXD。
R1OUT	第 1 組 RS-232，用來將 RS232 收到的資料輸出至單晶片，需連接至單晶片的 RXD，輸出電壓 5V 與 0V。
R1IN	第 1 組 RS-232，用來接收 RS-232 的資料，需連接至 RS232 連接器的 RXD。
T1OUT	第 1 組 RS-232，用來將資料傳送到 RS-232，需連接至 RS232 連接器的 TXD，輸出電壓±10V。
GND	接地，0V。
VCC	接電源，+5V。

硬體線路圖：

本實習硬體線路圖如圖 13-8 所示，本實習分別使用 PD5、PD6、與 PD7 控制 LCM 模組的 RS、 $\overline{R/W}$ 、與 EN 等腳位，使用 Port B 存取 LCM 的命令與資料，使用 Port E 讀取 4x4 按鍵，以及使用 RXD/PG5 與 TXD/PG4 進行序列資料傳輸。

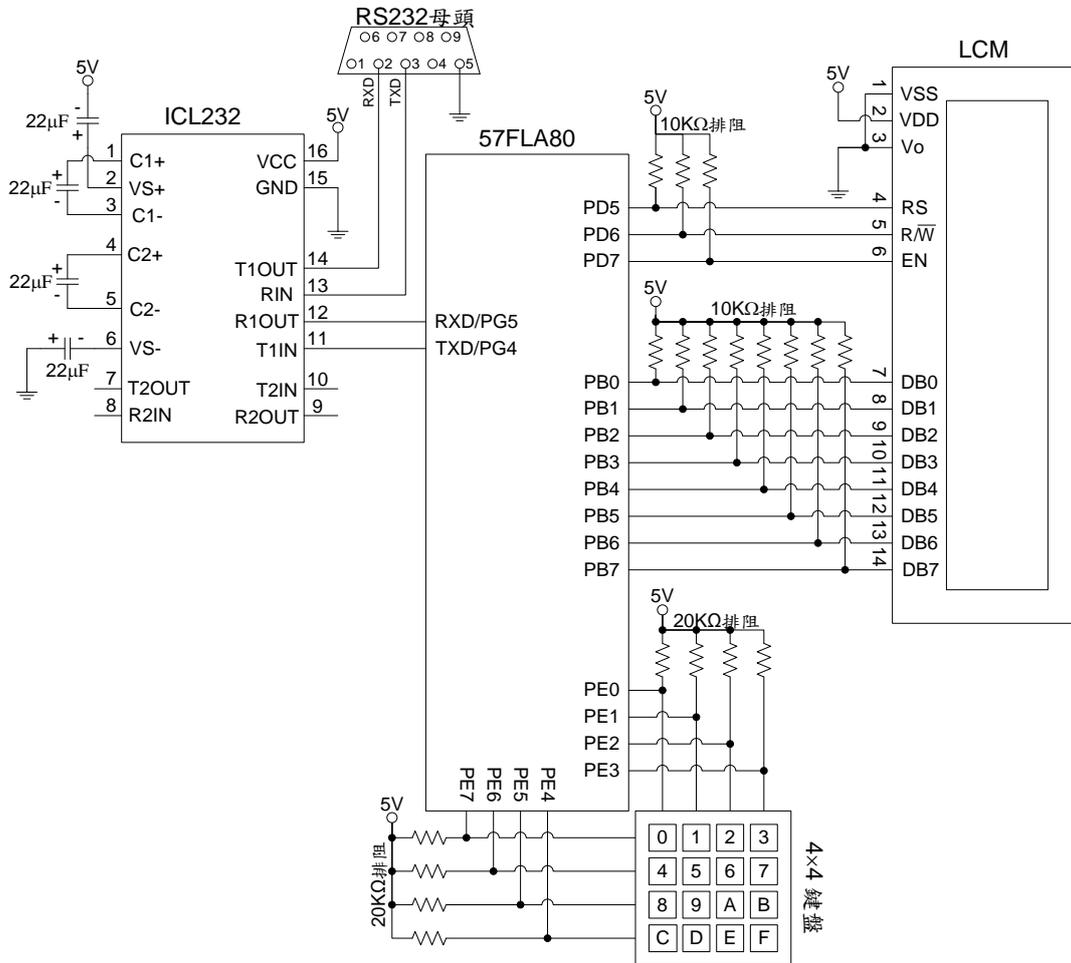


圖 13-8 RS232 傳輸硬體線路圖

**程式流程：**

本實習將利用 RS-232 介面進行個人電腦與單晶片之間的資料傳輸，要進行個人電腦與單晶片之間的資料傳輸，必須在個人電腦端與單晶片端各提供一個具備 RS232 傳輸功能的程式。為了方便測試，本書提供一個 RS232.EXE 程式，該程式可在個人電腦上執行，提供 RS-232 參數設定，以及透過 RS232 傳輸資料等功能。

執行本書提供的 RS232 工具程式，一開始會進入參數設定頁面，參數設定頁面如圖 13-9 所示，使用者可選擇要使用的個人電腦 Com Port、Baud Rate、Data Size、與 Stop Bit。

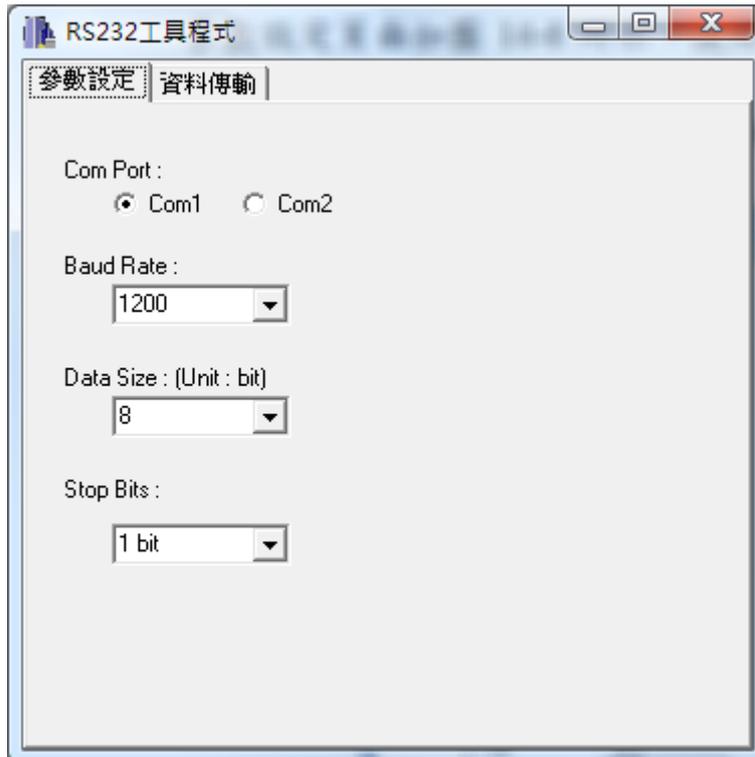


圖 13-9 RS232 工具程式—參數設定頁

設定好想要的 RS232 參數後，便可點進資料傳輸頁面進行資料傳輸工作，資料傳輸頁面如圖 13-10 所示。

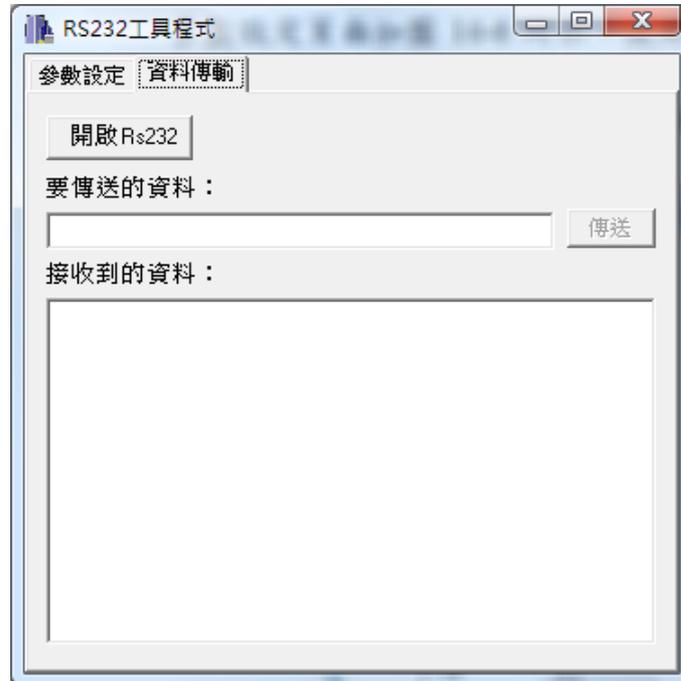


圖 13-10 RS232 工具程式－資料傳輸頁

在資料傳輸頁傳送資料前，需先點選“開啟 RS232”按鈕，才可開始收送資料，點選“開啟 RS232”按鈕後的畫面如圖 13-11 所示：



圖 13-11 RS232 工具程式－開啟 RS232 後

開啟 RS232 後，要傳送資料，只要在“要傳送的資料”下方文字輸入方塊內輸入字串並按下“傳送”按鈕，便可將資料透過 RS232 傳送出去。此外，RS232 工具程式也會隨時監看是否有資料透過 RS232 傳送進來，當有資料傳送進來時，RS232 工具程式會將收到的資料顯示在“接收到的資料”方塊內，方便使用者檢測資料傳輸結果是否正確。

單晶片端的程式流程如圖 13-12 所示，首先對 LCM 以及 UART 進行初始化動作，接著檢查有無資料由個人電腦端傳入，如果有，則將接收到的資料顯示在 LCD 上，接著，再檢查看看使用者有無按鍵，如果有按鍵，則將按鍵碼透過 RS232 傳送給個人電腦。

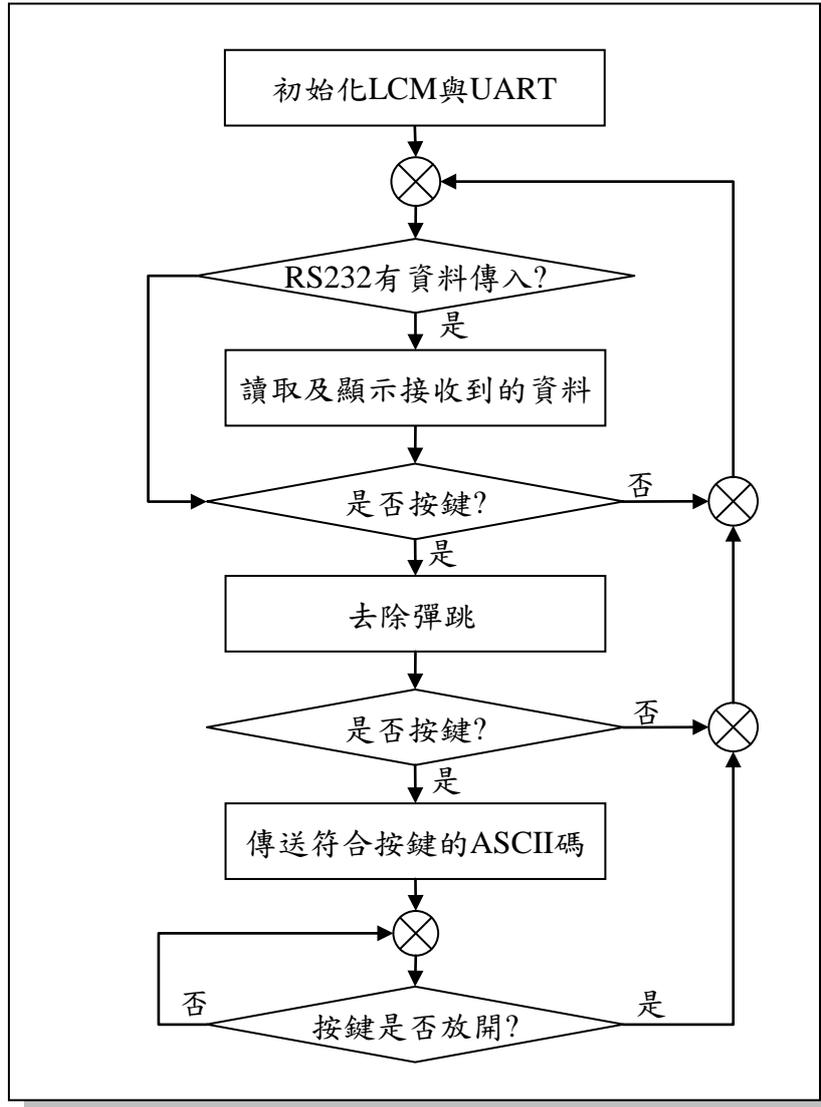


圖 13-12 單晶片端 RS232 傳輸程式流程

## 程式碼及程式說明：

```

; 定義程式中使用到的F-Plane暫存器記憶體位址
PBD      equ      06h      ; Port B
PED      equ      13h      ; Port E

UARTC    equ      17h      ; UART control register
URDATA   equ      19h      ; UART data buffer

; 定義程式中使用到的R-Plane暫存器記憶體位址
PEE      equ      13h      ; Port E Push-Pull enable
UBAUD    equ      17h      ; UART control register

; 定義程式中使用到的F-Plane變數記憶體位址
data     equ      20h      ; 儲存要傳送的資料
keyStatus equ      21h      ; 按鍵狀態
R1       equ      22h      ; 迴圈控制變數
R2       equ      23h      ; 迴圈控制變數

; 定義程式中所使用到的字串
RS_LCM   .defstr    12h, 5 ; LCM RS控制位元
RW_LCM   .defstr    12h, 6 ; LCM RW控制位元
EN_LCM   .defstr    12h, 7 ; LCM EN控制位元

URBF     .defstr    18h, 0 ; UART資料接收狀態
PRE      .defstr    18h, 4 ; UART同位位元設定
UTBE     .defstr    18h, 7 ; UART資料傳送狀態

; 系統開機進入點
org      00h

; LCM初始設定
Start:   call      Delay      ; 等候LCM啟動完成
         call      LCMInitialize ; 進行LCM初始設定

; UART初始設定
         movlw     00101110b ; UMODE=01, DATA=8位元
         ; TXE=1, 啟用傳送功能
         ; RXE=1, 啟用接收功能
         ; UARTE=1, 啟用UART功能
         ; UINVEN=0, 訊號不反相
         movwf     UARTC      ; 寫入UART控制暫存器

```

```

        bcf          PRE          ; 設定PRE=0，封包不合同位位元

        movlw       104          ;
        movwr       UBAUD        ; 設定UBAUD=104，使用鮑率1200bps

        ; 致能鍵盤輸入腳位Push-pull功能
        movlw       ffh
        movwr       PEE

        ; 檢查是否有資料傳送進來
MainLoop: btfsc          URBF          ; 檢查輸入緩衝是否有資料
        call        ReadUR      ; 有資料，讀取及顯示接收到的資料

        ; 檢查是否有按鍵
        movlw       ffh          ;
        movwf       PED          ; 清除按鍵狀態
        movlw       0fh          ; 將鍵盤列位址(bits4~7)變成低電位
        movwf       PED          ; 將鍵盤行位址(bits0~3)變成高電位
        movfw       PED          ; 讀取鍵盤狀態，沒按鍵應是0FH
        addlw       f0h          ; +F0H
        movwf       keyStatus    ; 取得按鍵狀態，沒按鍵是
0FH+F0H=FFH
        incfsz      keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
        goto        DeNoise      ; 有按鍵，跳到DeNoise去除彈跳
        goto        MainLoop     ; 沒按鍵，回到MainLoop

DeNoise: ; 去除彈跳
        call        Delay        ; 等待15ms

        ; 檢查是否有按鍵被按下
        movlw       ffh          ;
        movwf       PED          ; 清除按鍵狀態
        movlw       0fh          ; 將鍵盤列位址(bits4~7)變成低電位
        movwf       PED          ; 將鍵盤行位址(bits0~3)變成高電位
        movfw       PED          ; 讀取鍵盤狀態，沒按鍵應是0FH
        addlw       f0h          ; +F0H
        movwf       keyStatus    ; 取得按鍵狀態，沒按鍵是FFH
        incfsz      keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵
        goto        Row1        ; 有按鍵，到Row1檢查按了什麼鍵
        goto        MainLoop     ; 沒按鍵，回到MainLoop

Row1:    ; 檢查第一列是否有按鍵被按下
        movlw       ffh          ;
        movwf       PED          ; 清除按鍵狀態
        movlw       efh          ; 將鍵盤第一列(bit 4)變成低電位
        movwf       PED          ;

```

	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是efh
	addlw	10h	; +10h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	Key0	; 有按鍵，檢查是否按了'0'
	goto	Row2	; 沒按鍵，檢查第二列
Key0:	btfscl	PED, 0	; 檢查'0'是否被按下
	goto	Key1	; '0'沒被按下，檢查是否按了'1'
	movlw	'0'	; '0'被按下，將'0'的ASCII碼放入
	movwf	data	; data變數中，等待傳送給對方
	goto	WaitRelease	; 等待按鍵放開
Key1:	btfscl	PED, 1	; 檢查'1'是否被按下
	goto	Key2	; '1'沒被按下，檢查是否按了'2'
	movlw	'1'	; '1'被按下，將'1'的ASCII碼放入
	movwf	data	; data變數中，等待傳送給對方
	goto	WaitRelease	; 等待按鍵放開
Key2:	btfscl	PED, 2	; 檢查'2'是否被按下
	goto	Key3	; '2'沒被按下，所以是'3'被按下
	movlw	'2'	; '2'被按下，將'2'的ASCII碼放入
	movwf	data	; data變數中，等待傳送給對方
	goto	WaitRelease	; 等待按鍵放開
Key3:	movlw	'3'	; '3'被按下，先將'3'的ASCII碼放入
	movwf	data	; data變數中，等待傳送給對方
	goto	WaitRelease	; 等待按鍵放開
Row2:			; 檢查第二列是否有按鍵被按下
	movlw	ffh	;
	movwf	PED	; 清除按鍵狀態
	movlw	dfh	; 將鍵盤第二列(bit 5)變成低電位
	movwf	PED	;
	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是dfh
	addlw	20h	; +20h
	movwf	keyStatus	; 取得按鍵狀態，沒按鍵應是ffh
	incfsz	keyStatus, 1	; 若按鍵狀態+1等於0，表示沒按鍵
	goto	Key4	; 有按鍵，檢查是否按了'4'
	goto	Row3	; 沒按鍵，檢查第三列
Key4:	btfscl	PED, 0	; 檢查'4'是否被按下
	goto	Key5	; '4'沒被按下，檢查是否按了'5'
	movlw	'4'	; '4'被按下，將'4'的ASCII碼放入
	movwf	data	; data變數中，等待傳送給對方
	goto	WaitRelease	; 等待按鍵放開

Key5:	<pre> btfsc    PED, 1    ; 檢查'5'是否被按下 goto     Key6     ; '5'沒被按下，檢查是否按了'6' movlw    '5'      ; '5'被按下，將'5'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>
Key6:	<pre> btfsc    PED, 2    ; 檢查'6'是否被按下 goto     Key7     ; '6'沒被按下，所以是'7'被按下 movlw    '6'      ; '6'被按下，將'6'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>
Key7:	<pre> movlw    '7'      ; '7'被按下，將'7'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>
Row3:	<pre> ; 檢查第三列是否有按鍵被按下 movlw    ffh      ; movwf    PED      ; 清除按鍵狀態 movlw    bfh      ; 將鍵盤第三列(bit 6)變成低電位 movwf    PED      ; movwf    PED      ; 讀取鍵盤狀態，沒按鍵應是bfh addlw    40h      ; +40h movwf    keyStatus ; 取得按鍵狀態，沒按鍵應是ffh incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示沒按鍵 goto     Key8     ; 有按鍵，檢查是否按了'8' goto     Row4     ; 沒按鍵，檢查第四列 </pre>
Key8:	<pre> btfsc    PED, 0    ; 檢查'8'是否被按下 goto     Key9     ; '8'沒被按下，檢查是否按了'9' movlw    '8'      ; '8'被按下，將'8'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>
Key9:	<pre> btfsc    PED, 1    ; 檢查'9'是否被按下 goto     KeyA     ; '9'沒被按下，檢查是否按了'A' movlw    '9'      ; '9'被按下，將'9'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>
KeyA:	<pre> btfsc    PED, 2    ; 檢查'A'是否被按下 goto     KeyB     ; 'A'沒被按下，所以是'B'被按下 movlw    'A'      ; 'A'被按下，將'A'的ASCII碼放入 movwf    data     ; data變數中，等待傳送給對方 goto     WaitRelease ; 等待按鍵放開 </pre>

KeyB:	movlw movwf goto	'B' data WaitRelease	; 'B'被按下，將'B'的ASCII碼放入 ; data變數中，等待傳送給對方 ; 等待按鍵放開
Row4:			; 檢查第四列哪個按鍵被按下
	movlw movwf movlw movwf movfw addlw movwf incfsz goto goto	ffh PED 7fh PED PED 80h keyStatus keyStatus, 1 KeyC WaitRelease	; ; 清除按鍵狀態 ; 將鍵盤第四列(bit 7)變成低電位 ; ; 讀取鍵盤狀態，沒按鍵應是7fh ; +80h ; 取得按鍵狀態，沒按鍵應是ffh ; 若按鍵狀態+1等於0，表示沒按鍵 ; 有按鍵，檢查是否按了'C' ; 等待按鍵放開
KeyC:	btfsz goto movlw movwf goto	PED, 0 KeyD 'C' data WaitRelease	; 檢查'C'是否被按下 ; 'C'沒被按下，檢查是否按了'D' ; 'C'被按下，將'C'的ASCII碼放入 ; data變數中，等待傳送給對方 ; 等待按鍵放開
KeyD:	btfsz goto movlw movwf goto	PED, 1 KeyE 'D' data WaitRelease	; 檢查'D'是否被按下 ; 'D'沒被按下，檢查是否按了'E' ; 'D'被按下，將'D'的ASCII碼放入 ; data變數中，等待傳送給對方 ; 等待按鍵放開
KeyE:	btfsz goto movlw movwf goto	PED, 2; KeyF 'E' data WaitRelease	; 檢查'E'是否被按下 ; 'E'沒被按下，所以是'F'被按下 ; 'E'被按下，將'E'的ASCII碼放入 ; data變數中，等待傳送給對方 ; 等待按鍵放開
KeyF:	movlw movwf	'F' data	; 'F'被按下，將'F'的ASCII碼放入 ; data變數中，等待傳送給對方
			; 等待使用者放開按鍵
WaitRelease:	movfw movwf	data URDATA	; 取出要傳送的資料 ; 將資料傳送出去
ChkTx:	btfsz goto	UTBE ChkTx	; 查看是否傳送完成 ; 還沒傳完，回到ChkTx
			; 讀取按鍵狀態

```

movlw    ffh          ;
movwfw   PED         ; 清除按鍵狀態
movlw    0fh         ; 將0fh寫入PED
movwfw   PED         ;
movwfw   PED         ; 讀取鍵盤狀態，若放開按鍵應是0fh
addlw    f0h         ; +f0h
movwfw   keyStatus   ; 取得按鍵狀態，若放開按鍵應是ffh
incfsz   keyStatus, 1 ; 若按鍵狀態+1等於0，表示放開按鍵
goto     ChkTx       ; 沒放開按鍵，繼續等待
goto     MainLoop    ; 回到MainLoop

; LCM初始設定副程式
LCMInitialize:
    ; 進行三次命令寫入動作及等候適當時間延遲，讓系統穩定
movlw    30h         ; 功能設定：設定DL=N=F=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    30h         ; 功能設定：設定DL=N=F=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    30h         ; 功能設定：設定DL=N=F=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    38h         ; 功能設定：設定DL=N=1, F=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    08h         ; 顯示器控制：D=C=B=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    01h         ; 清除顯示
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    06h         ; 模式設定：I/D=1, S=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM

movlw    0eh         ; 顯示器控制：D=C=1, B=0
movwfw   data        ; 載入命令碼
call     WriteIR     ; 將命令碼寫到LCM
ret

```

```

;將命令碼寫入LCM指令暫存器副程式
WriteIR:    bcf      RS_LCM    ;選擇指令暫存器
            bcf      RW_LCM    ;選擇進行寫入動作
            bsf      EN_LCM    ;開啟LCM讀寫功能
            movfw   data      ;載入命令碼
            movwf   PBD      ;送出命令碼
            call    Delay     ;等候一段時間，讓資料完成寫入動作
            bcf      EN_LCM    ;關閉LCM讀寫功能
            ret           ;返回呼叫函式

;將資料寫入LCM記憶體副程式
WriteDR:    bsf      RS_LCM    ;選擇記憶體
            bcf      RW_LCM    ;選擇進行寫入動作
            bsf      EN_LCM    ;開啟LCM讀寫功能
            movfw   data      ;載入資料
            movwf   PBD      ;送出資料
            call    Delay     ;等候一段時間，讓資料完成寫入動作
            bcf      EN_LCM    ;關閉LCM讀寫功能
            ret           ;返回呼叫函式

;讀取及顯示接收到資料之副程式
ReadUR:     movfw   URDATA    ;取得UART收到的值
            movwf   data      ;
            call    WriteDR   ;將UART收到的值顯示到LCD上
            ret

;延遲0.015秒副程式
Delay:      movlw   30        ;設定外層迴圈執行30次
            movwf   R1        ;

            ;外層迴圈
delay_L1:   movlw   200       ;設定內層迴圈執行200次
            movwf   R2        ;

            ;內層迴圈：迴圈跑一次約消耗5個指令週期
delay_L2:   nop              ;消耗1個指令週期
            nop              ;消耗1個指令週期
            decfsz   R2, 1    ;約消耗1個指令週期
            goto    delay_L2 ;消耗2個指令週期

            decfsz   R1, 1    ;將R1減1，若R1=0離開迴圈
            goto    delay_L1 ;
            ret

```

## 14. SPI 串列週邊介面實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片所提供的串列週邊介面 (Serial Peripheral Interface ; SPI) 。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	10KΩ排阻 (A-type 9PIN)	3
2	20KΩ排阻 (A-type 9PIN)	2
3	10KΩ電阻	1
4	330Ω電阻	4
5	七段顯示器(共陽)	4
6	ICE25P05 快閃記憶體	1
7	7417 IC	1
8	7447 IC	4
9	4 × 4 鍵盤	1

### 實習板模組與 I/O Port：

模組	I/O Port
四顆七段顯示器模組	Port B
4 × 4 鍵盤模組	Port E
FLASHROM 模組	Port F
FLASHROM 模組	Port G

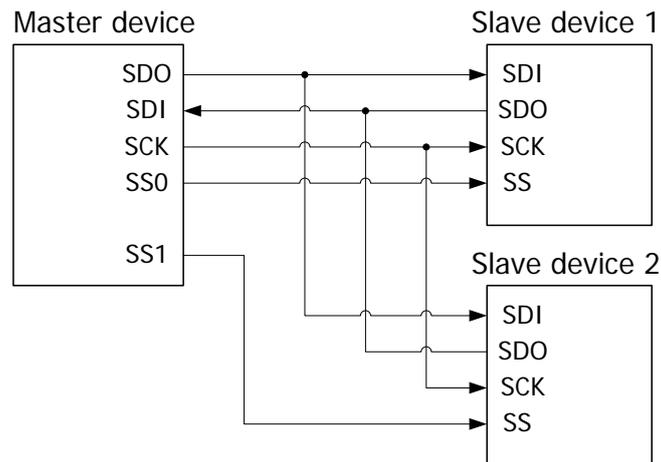
**實習說明：**

本實習使用十速 57FLA80 單晶片所提供的 SPI 串列週邊介面 (Serial Peripheral Interface) 讀寫外部的快閃記憶體(Flash Memory)。運作方式是等待使用者由 4x4 鍵盤輸入 4 個數字，然後將數字寫入快閃記憶體後再讀出來顯示於 4 顆七段顯示器上，檢查七段顯示器顯示的數字，可了解快閃記憶體是否可正常讀寫。以下依序介紹 SPI、57FLA80 的 SPI 支援、ICE25P05 快閃記憶體、以及 7417 IC。

**● SPI 簡介**

SPI 是一種具全雙工與同步傳輸能力的串列傳輸介面，可供單晶片跟週邊裝置 (如其它單晶片、週邊 IC、快閃記憶體等) 傳輸資料使用。其中，單晶片稱為主裝置(Master Device)，週邊裝置稱為從裝置(Slave Device)，在使用 SPI 傳輸資料的過程中，主裝置只能有一個，從裝置則可以有很多個，所有資料傳輸動作都必須由主裝置發動及掌控，從裝置則被動接受主裝置的讀寫要求。

使用 SPI 傳輸資料，每個從裝置必須提供 SDI, SDO, SCK, 與 SS 等 4 個接腳，其中 SDI, SDO, 與 SCK 用來傳輸資料，而 SS 則是主裝置用來選取要動作的從裝置用的，主裝置與從裝置之間的接線方式如圖 14-1 所示：

**圖 14-1 主從裝置的 SPI 接法**

如圖 14-1 所示，主裝置要使用 SPI 介面傳送資料給從裝置時，可利用 SS0 與 SS1 腳位選擇要動作的從裝置，然後再利用 SDI, SDO, 與 SCK 等腳位控制資料的傳輸。SDI, SDO, SCK, 與 SS 等腳位詳細說明如下：

表 14-1 SPI 控制訊號線

控制訊號線	說明
SDI (Serial Data In) 串列資料輸入	用來接收對方傳送過來的資料
SDO(Serial Data Out) 串列資料輸出	用來傳送資料給對方
SCLK(Serial Clock) 串列時脈	主裝置透過此腳位提供時脈訊號給從裝置
SS(Slave Select) 從裝置選取	主裝置用來選取要動作的從裝置，每個從裝置都必須提供一個 SS 腳位。

● 57FLA80 的 SPI 支援

57FLA80 單晶片支援 SPI 傳輸介面，可作為主裝置或從裝置使用，57FLA80 單晶片作為主裝置或從裝置使用的接線方式如圖 14-2 所示：

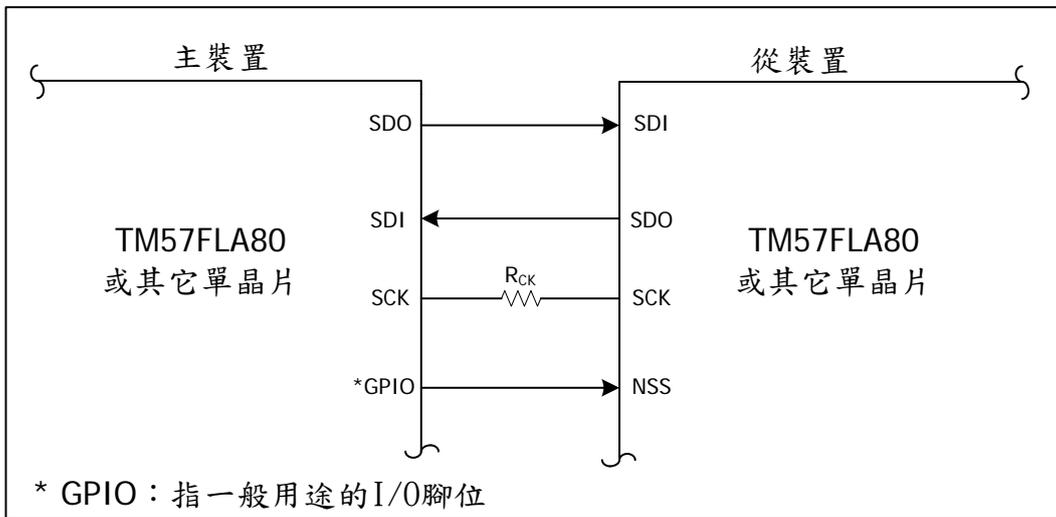


圖 14-2 57FLA80 單晶片作為主裝置或從裝置的接線方式

57FLA80 單晶片不論當作主裝置或從裝置使用，都會使用 SDI、SDO、與 SCK 等三個腳位（這三個腳位分別跟 PG0, PG1, 與 PG2 使用相同腳位）。當 57FLA80 作為主裝置使用時，不需提供 SS 腳位，要選擇從裝置則可使用一般用途的 I/O 腳位(例如 PA0~6, PB0~7, ...等)；當 57FLA80 作為從裝置使用時，則必須使用 NSS 腳位（跟 PG3 使用相同腳位），該腳位可接受主裝置控制，決定是否接受 SDI、SDO、與 SCK 等控制訊號。

57FLA80 單晶片的 SPI 功能方塊圖如圖 14-3 所示：

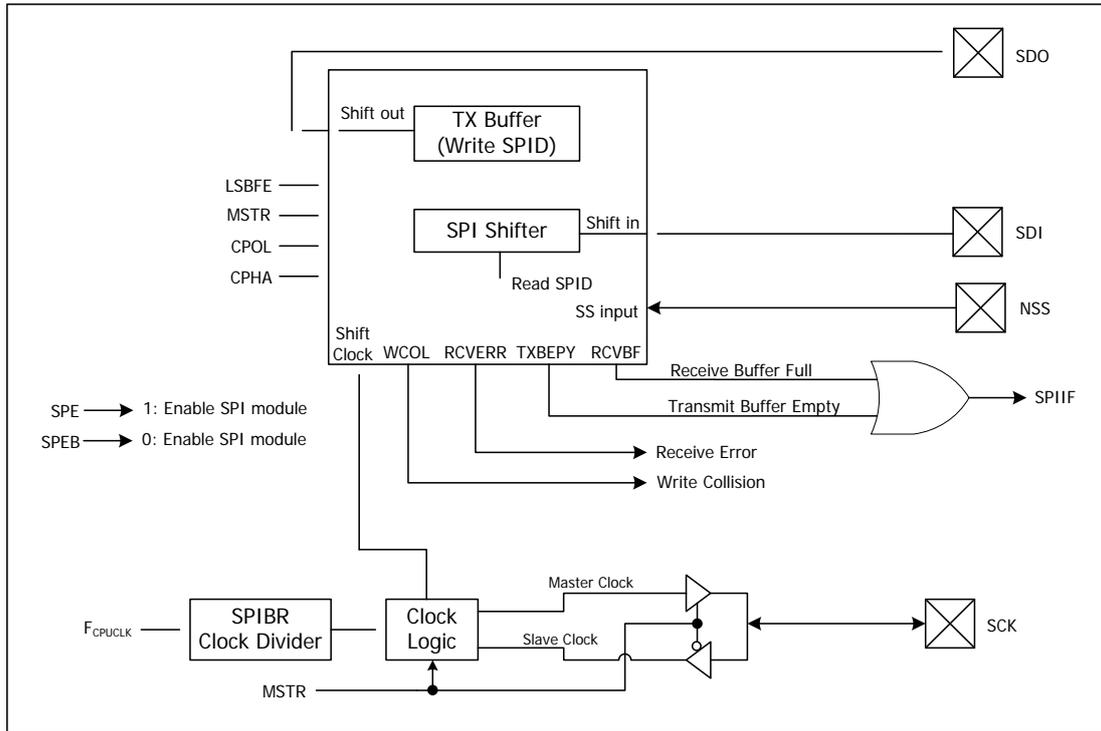


圖 14-3 57FLA80 單晶片 SPI 功能方塊圖

使用 57FLA80 的 SPI 功能前，必須先將 SPE 位元設定為 1 以及將 SPEB 位元設定為 0，才能啟用 SPI 功能，接著利用 MSTR 選擇要當主裝置還是從裝置。如果要當從裝置使用，則時脈訊號由外部提供並透過 SCK 接腳傳入，如果當作主裝置使用，則必須自行產生時脈訊號，自行產生之時脈訊號的 Baud Rate 由 SPIBR 暫存器的內容決定，可用的 Baud Rate 範圍由  $F_{CPUCLK}/2$  到  $F_{CPUCLK}/2048$ 。SPIBR 暫存器內含 SPPR 與 SPR 兩個 3 位元大小的參數。SPI 的 Baud Rate =  $F_{CPUCLK} / SPPR$  指定的除數 / SPR 指定的除數。SPPR 與 SPR 的除數指定方式分別如表 14-2 與表 14-3 所示：

表 14-2 SPPR 的除數指定方式

SPPR 內容	000b	001b	010b	011b	100b	101b	110b	111b
除數	1	2	3	4	5	6	7	8

表 14-3 SPR 的除數指定方式

SPR	000b	001b	010b	011b	100b	101b	110b	111b
除數	2	4	8	16	32	64	128	256

除了 Baud Rate 外，還有 CPOL 與 CPHA 兩個時脈相關的設定，CPOL 用來設定時脈的極性(Clock Polarity)，當 CPOL 的值為 1 時，時脈為 Active Low，當 CPOL 的值為 0 時，時脈為 Active High。CPHA 用來設定時脈的相位(Clock Phase)，當 CPHA 的值為 1 時，時脈的邊緣出現在資料位元的開始處，當 CPHA 的值為 0 時，時脈的邊緣出現在資料位元的中間。

除了時脈的產生方式之外，使用 SPI 傳輸資料，還要指定位元的傳送順序，在 57FLA80 單晶片內，位元的傳送順序可透過 LSBFE 控制位元指定，當 LSBFE 的值為 1 時，資料傳送方式是由 LSB (Least Significant Bit) 先傳，否則由 MSB (Most Significant Bit) 先傳。

時脈及位元傳送順序確定後，便可開始傳送資料，由於 57FLA80 可作為主裝置或從裝置使用，因此以下分別說明 57FLA80 作為主裝置與從裝置時的資料傳輸方式。

若 57FLA80 作為主裝置使用，要傳送資料時，需先透過一般用途的 IO 接腳選擇從裝置，然後再將要傳送的位元組寫入 SPID 暫存器，資料寫入 SPID 暫存器後，硬體會自動產生 SDO 與 CLK 訊號，將資料傳送出去，資料傳送過程中，可透過 TXBEPY 的值檢查資料是否傳送完成，傳送未完成 TXBEPY 為 0，傳送完成 TXBEPY 為 1。

若單晶片作為從裝置使用，要傳送資料，只要將要傳送的位元組寫入 SPID 暫存器，然後等候 TXBEPY 的值變成 1 即可完成資料傳送過程。

若單晶片作為主裝置使用，要接收資料，需先選用從裝置，然後對 SPID 暫存器進行寫入動作，啟動 SPI 資料接收程序，資料接收程序啟動後，硬體會自動產生 SDI 與 CLK 訊號，向從裝置讀取資料，並將收到的資料放在 SPID 暫存器內以及將 RCVBF 的值設為 1，其中，RCVBF 的值會在 SPID 暫存器內的值被讀取時自動清為 0。另外，在資料接收過程中，硬體會自動檢查是否有錯誤發生，並設定適當的錯誤位元。錯誤位元有 RCVERR 跟 WCOL 兩種，RCVERR 表示資料接收過程中發生錯誤，WCOL 表示 SPID 的內容還沒被讀取就被新接收到的資料覆蓋掉的錯誤。

若單晶片作為從裝置使用，要接收主裝置送來的資料，則可檢查 RCVBF 的值是否為 1，若 RCVBF 的值為 1，表示資料已接收完成，可從 SPID 暫存器取值並進行適當動作。

資料傳送接收過程中，如果 SPIIE 位元設定為 1，則資料收送完成，會自動發出中斷，中斷發生後，SPIIF 旗號會設定為 1，中斷處理完成，可透過韌體將 SPIIF 清為 0。

圖 14-4 是 57FLA80 作為主裝置使用時，將資料透過 SPI 寫到從裝置的相關訊號時序圖。

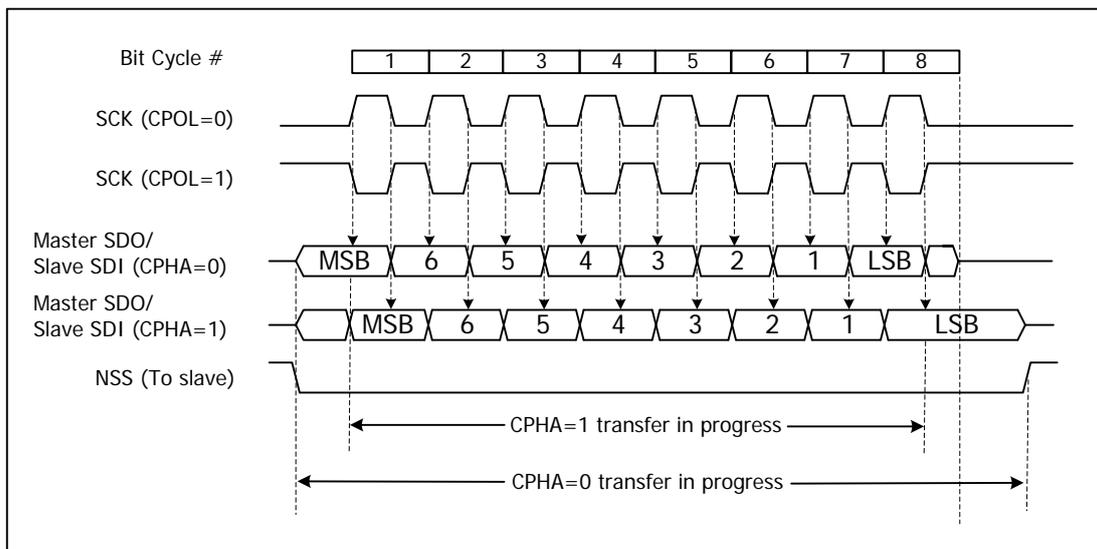


圖 14-4 資料寫入從裝置的相關訊號時序圖

57FLA80 單晶片中與 SPI 控制相關的暫存器整理如下：

表 14-4 SPI 控制相關暫存器

暫存器	說明																		
SPIIE	位於 F-PLANE 位址 0eh 暫存器位元 5，用來設定 SPI 收送完成是否產生中斷，1：啟用，0：關閉（預設值為 0）。																		
SPIIF	位於 F-PLANE 位址 0fh 暫存器位元 5，SPI 中斷旗號，SPI 收送完成，硬體自動將 SPIIF 設為 1（預設值為 0）。																		
SPID	位於 F-PLANE 位址 1ah 暫存器，SPI 傳送接收資料暫存器。																		
RCVERR	位於 F-PLANE 位址 1bh 暫存器位元 7，SPI 接收資料過程發生錯誤自動設為 1，需靠韌體清為 0（預設值為 0）。																		
RCVBF	位於 F-PLANE 位址 1bh 暫存器位元 6，SPI 資料接收完畢自動設為 1，讀取 SPID 自動清為 0（預設值為 0）。																		
TXBEPY	位於 F-PLANE 位址 1bh 暫存器位元 5，用來表示傳送是否完成，1: 傳送完成，0: 資料傳送中（預設值為 1）。																		
WCOL	位於 F-PLANE 位址 1bh 暫存器位元 4，接收資料覆蓋到前一筆資料硬體自動設為 1，需靠韌體清為 0（預設值為 0）。																		
SPPR	位於 R-PLANE 位址 18h 暫存器位元 6~4，SPI Baud Rate 設定用（預設值為 0）。 <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>SPPR 內容</th> <th>000b</th> <th>001b</th> <th>010b</th> <th>011b</th> <th>100b</th> <th>101b</th> <th>110b</th> <th>111b</th> </tr> </thead> <tbody> <tr> <td>除數</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> </tr> </tbody> </table>	SPPR 內容	000b	001b	010b	011b	100b	101b	110b	111b	除數	1	2	3	4	5	6	7	8
SPPR 內容	000b	001b	010b	011b	100b	101b	110b	111b											
除數	1	2	3	4	5	6	7	8											
SPR	位於 R-PLANE 位址 18h 暫存器位元 2~0，SPI Baud Rate 設定用（預設值為 0），SPI Baud Rate 的值为 $F_{CPUCLK}/SPPR$ 指定的除數/SPR 指定的除數。 <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>SPR</th> <th>000b</th> <th>001b</th> <th>010b</th> <th>011b</th> <th>100b</th> <th>101b</th> <th>110b</th> <th>111b</th> </tr> </thead> <tbody> <tr> <td>除數</td> <td>2</td> <td>4</td> <td>8</td> <td>16</td> <td>32</td> <td>64</td> <td>128</td> <td>256</td> </tr> </tbody> </table>	SPR	000b	001b	010b	011b	100b	101b	110b	111b	除數	2	4	8	16	32	64	128	256
SPR	000b	001b	010b	011b	100b	101b	110b	111b											
除數	2	4	8	16	32	64	128	256											
MSTR	位於 R-PLANE 位址 19h 暫存器位元 5，用來設定要當作主裝置或從裝置使用，1: 主裝置，0: 從裝置（預設值為 1）。																		
CPOL	位於 R-PLANE 位址 19h 暫存器位元 4，用來設定 SPI Clock 的極性，1: Active-Low，0: Active High（預設值為 0）。																		
CPHA	位於 R-PLANE 位址 19h 暫存器位元 3，用來設定 SPI Clock 的相位，1: 時脈邊緣出現在資料位元開始處，0: 時脈邊緣出現在資料位元中間（預設值為 0）。																		
SPEB	位於 R-PLANE 位址 19h 暫存器位元 2，用來設定是否關閉 SPI 功能，1: 關閉，0: 啟用（預設值為 1）。																		
LSBFE	位於 R-PLANE 位址 19h 暫存器位元 1，用來設定先送 LSB 或 MSB，1: LSB 先送，0: MSB 先送（預設值為 0）。																		
SPE	位於 R-PLANE 位址 19h 暫存器位元 0，用來設定是否啟用 SPI 功能，1：啟用，0：關閉（預設值為 0）。																		

- ICE25P05 是一顆具備 SPI 傳輸能力，容量 512K 位元的串列式快閃記憶體。ICE25P05 內部記憶體分成兩個資料區(Sector)，每個資料區又有 256 個分頁(Page)，每個分頁內含 128 個位元組。圖 14-5 與表 14-5 分別列出 ICE25P05 快閃記憶體的接腳圖與腳位說明：

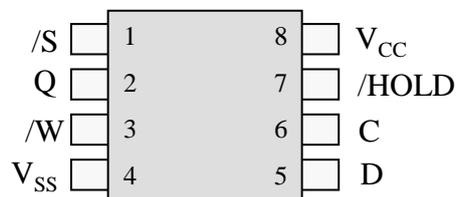


圖 14-5 ICE25P05 快閃記憶體的接腳圖

表 14-5 ICE25P05 快閃記憶體腳位說明

腳位	腳位說明
/S	晶片選擇腳位，此腳位相當於 SPI 的 SS。 0：啟用讀寫功能，1：關閉讀寫功能。
Q	串列資料輸出腳位，此腳位相當於 SPI 的 SDO。
/W	寫入保護腳位，用來凍結保護區域的寫入動作。 保護區域的位置跟狀態暫存器的設定有關。 0：寫入保護，1：不寫入保護
V <sub>SS</sub>	接地，0V。
D	串列資料輸入腳位，此腳位相當於 SPI 的 SDI。
C	時脈輸入腳位，此腳位相當於 SPI 的 SCK。
/HOLD	暫停串列傳輸功能，0：暫停，1：不暫停。
V <sub>CC</sub>	接電源，2.7V ~ 3.6V。

ICE25P05 提供兩種時脈模式，一種是 CPOL=CPHA=0，另一種是 CPOL=CPHA=1，兩種都是在時脈的上昇緣讀寫資料，資料傳送順序是先送 MSB。

要存取 ICE25P05，必須透過 ICE25P05 所提供的指令，ICE25P05 所提供的指令集如表 14-6 所示：

表 14-6 ICE25P05 指令集

指令	指令碼	說明
WREN	0000 0110b	允許寫入
WRDI	0000 0100b	不允許寫入
RDSR	0000 0101b	讀取狀態暫存器
WRSR	0000 0001b	寫入狀態暫存器
READ	0000 0011b	讀取資料位元組
PP	0000 0010b	寫入分頁資料
SE	1101 1000b	清除資料區
BE	1100 0111b	清除全部資料
RDID	0001 0101b	讀取裝置識別碼

ICE25P05 所提供的各項指令及用法說明如下：

- ✓ WREN 指令：允許對 ICE25P05 進行寫入動作。

在進行任何寫入 ICE25P05 的指令（包括 WRSR、PP、SE、BE 等指令）前，必須先送出 WREN 指令給 ICE25P05，設定 ICE25P05 進入允許寫入狀態。要傳送 WREN 指令，需先將/S 腳位設定成低電位，送出 WREN 指令碼，再將/S 腳位設定成高電位。

- ✓ WRDI 指令：不允許對 ICE25P05 進行寫入動作。

設定 ICE25P05 進入不允許寫入狀態，在某些情況下 ICE25P05 會自動進入不允許寫入狀態，這些情況包括：開機、完成 WRSR、SE、與 BE 等指令。要傳送 WRDI 指令，需先將/S 腳位設定成低電位，送出 WRDI 指令碼，再將/S 腳位設定成高電位。

- ✓ RDSR 指令：讀取 ICE25P05 內部狀態暫存器內容。

ICE25P05 內部提供一個位元組大小的狀態暫存器，RDSR 指令用來讀取狀態暫存器的內容，要讀取 ICE25P05 內部狀態暫存器內容，需先將/S 腳位設定成低電位，送出 RDSR 指令碼，然後讀取狀態暫存器，最後再將/S 腳位設定成高電位。ICE25P05 內部狀態暫存器格式如下：

表 14-7 ICE25P05 狀態暫存器

位元	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
用途	SRWD	0	0	0	BP1	BP0	WEL	WIP

狀態暫存器各位元說明如表 14-8 所示：

表 14-8 狀態暫存器位元說明

狀態	說明
SRWD	狀態暫存器是否可供寫入的狀態，1：不能寫入，0：可以寫入。
BP1&BP0	標示寫入保護範圍，當 BP1=BP0=1 時，整個記憶體都被保護，不能寫入。當 BP1=BP0=0 時，BE 指令才能被執行。
WEL	ICE25P05 是否可供寫入的狀態，1：可以寫入，0：不能寫入。這個位元的值會受 WREN 與 WEDI 影響。
WIP	寫入動作是否還在進行的狀態，1：寫入動作進行中，0：寫入動作完成。

- ✓ WRSR 指令：寫入 ICE25P05 內部狀態暫存器。

ICE25P05 內部狀態暫存器只有 SRWD、BP1、BP0 可供寫入。要寫入 ICE25P05 內部狀態暫存器，需將/S 腳位設定成低電位，接著送出 WRSR 指令碼，再送出要寫入狀態暫存器的內容，最後再將/S 腳位設定成高電位。

- ✓ READ 指令：讀取 ICE25P05 內部資料位元組。

要讀取 ICE25P05 內部資料，必須先將/S 腳位變為低電位，送出 READ 指令，再送出要讀取的位址，ICE25P05 使用 24 位元的記憶體位址，位址範圍為 000000H ~ 00FFFFH。ICE25P05 收到位址後，便會由指定的位址開始一個位元組接著一個位元組送出，資料傳送過程會在傳送完最後一個位元組 (00FFFFh 的資料)後，或是/S 腳位變為高電位時結束。

- ✓ PP 指令：寫入分頁資料。

要將資料寫入 ICE25P05，必須先將/S 腳位變為低電位，送出 PP 指令碼，再送出要寫入的位址(24 位元)，接著將要寫入的資料一個一個寫出，一個 PP 指令最多只能寫出一個分頁(128 個位元組)的資料，寫完資料後將/S 腳位變為高電位。PP 指令最多只能一次傳送一個分頁的內容，如果要寫入多個分頁，在傳完一個分頁後，必須等待 ICE25P05 完成該分頁的寫入動作後，才能繼續使用 PP 指令寫入下個分頁。

- ✓ SE 指令：清除指定的資料區內容。

SE 指令可將所指定的資料區內容清為 FFH，要清除資料區內容，必須先將/S 腳位變為低電位，送出 SE 指令碼，再送出要清除的資料區位址(24 位元)，接著將/S 腳位變為高電位。資料區 0 的合法位址範圍為 000000H~007FFFH，資料區 1 的合法位址範圍為 008000H~00FFFFH。

- ✓ BE 指令：清除所有資料內容。

要清除所有資料，先將/S 腳位變為低電位，送出 BE 指令碼，接著將/S 腳位變為高電位即可。

● 7417 緩衝器

ICE25P05 的操作電壓介於 2.7V 到 3.6V，跟單晶片使用的電壓不同，因此本實習使用 7417 緩衝器進行電壓轉換。7417 內部提供六組緩衝器，每組緩衝器提供 TTL/CMOS 輸入 (0V~5V)，以及開集極輸出，其緩衝器線路如圖 14-6 所示，其中輸出腳位 Y 沒有接到電源，使用時可自行提供電源，方便改變高電位的準位，例如將輸出腳位 Y 接一電阻至+3V，則 Y 的高電位準位為+3V，將 Y 接一電阻至+5V，則 Y 的高電位準位為 5V。

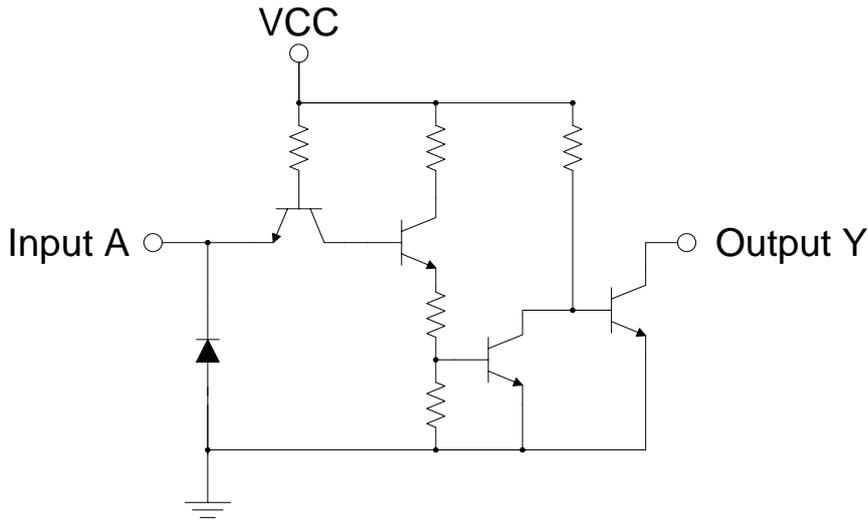


圖 14-6 7417 緩衝器線路圖

7417 緩衝器 IC 腳位圖如圖 14-7 所示，其中共有六組緩衝器，每組緩衝器的 A 是輸入 (0V~5V)，Y 是輸出，V<sub>CC</sub> 必須給+5V。

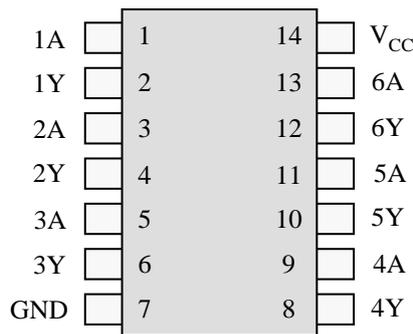


圖 14-7 7417 緩衝器 IC 腳位圖

硬體線路圖：

本實習硬體線路圖如圖 14-8 所示，分別使用 Port B 與 Port D 控制四顆七段顯示器的顯示，使用 Port E 讀取 4x4 鍵盤，以及使用 PG0~PG2 (SDI、SDO、SCK) 與 PF0 ~ PF2 (/HOLD、/W、/S) 進行快閃記憶體的讀寫控制。

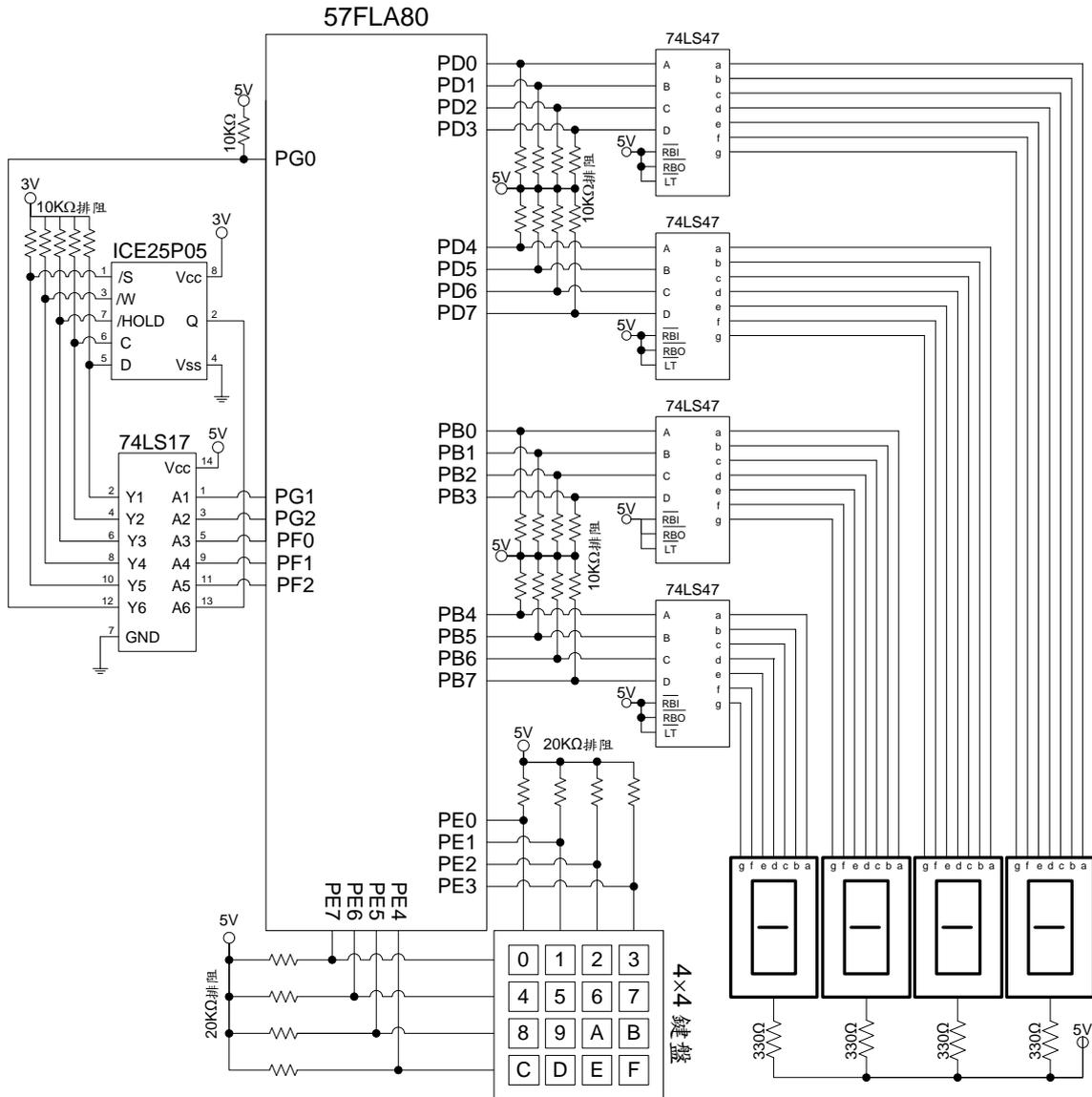
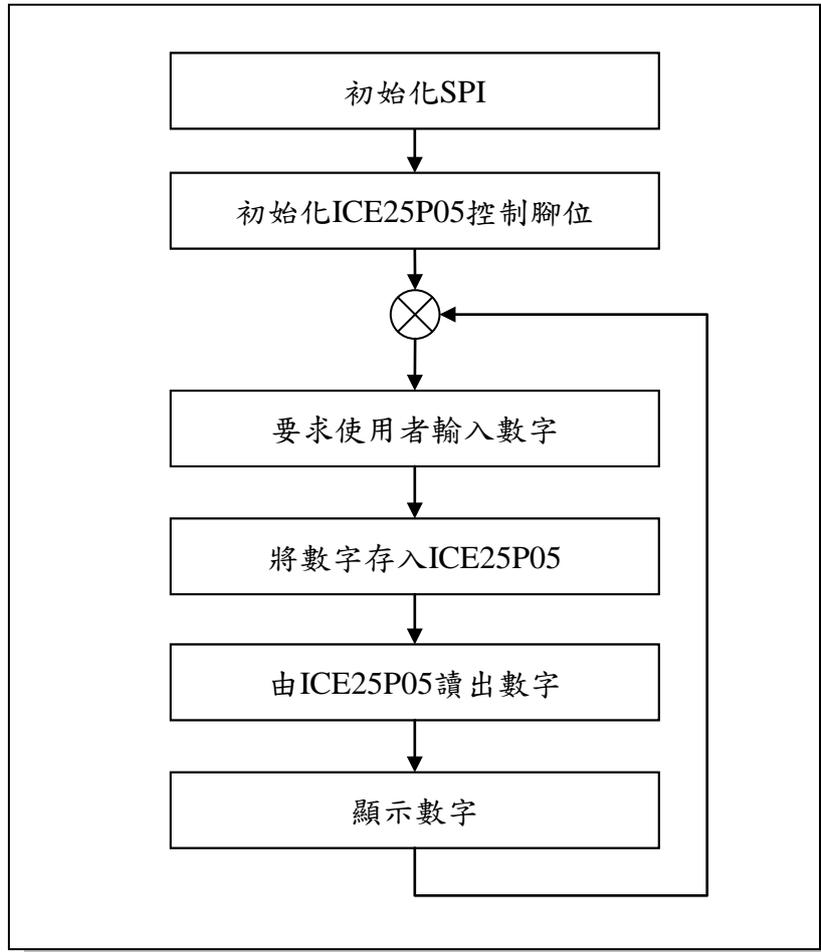


圖 14-8 SPI 傳輸硬體線路圖

**程式流程：**

本實習使用 57FLA80 的 SPI 實作 ICE25P05 快閃記憶體讀寫測試程式，程式流程如圖 14-9 所示，首先對單晶片的 SPI 與 ICE25P05 控制腳位進行初始設定，然後等待使用者由 4×4 鍵盤輸入 4 個數字並將數字寫入快閃記憶體，接著再由快閃記憶體取出先前寫入的 4 個數字並顯示在 4 顆七段顯示器上，方便使用者檢查快閃記憶體的運作是否正常。

**圖 14-9 SPI 傳輸程式流程**

## 程式碼及程式說明：

；定義程式中使用到的F-Plane暫存器記憶體位址

```
PBD      equ      06h      ; Port B
PDD      equ      12h      ; Port D
PED      equ      13h      ; Port E
PFD      equ      14h      ; Port F

SPID     .defstr   1ah      ; SPI資料暫存器
```

；定義程式中使用到的R-Plane暫存器記憶體位址

```
PEE      equ      13h      ; Port E Push-Pull enable

SPIBR    equ      18h      ; SPI Baud Rate設定暫存器
          ; 位元6 ~ 4 : SPPR
          ; 位元2 ~ 0 : SPR

SPICR    equ      19h      ; SPI 控制暫存器
          ; 位元5 : MSTR, SPI主/從裝置設定
          ; 位元4 : CPOL, SPI Clock極性設定
          ; 位元3 : CPHA, SPI Clock相位設定
          ; 位元2 : SPEB, SPI除能設定
          ; 位元1 : LSBFE, SPI資料傳送順序設定
          ; 位元0 : SPE, SPI致能設定
```

；定義程式中使用到的F-Plane變數記憶體位址

```
data     equ      20h      ; 儲存要傳送的資料
R1       equ      21h      ; 迴圈控制變數
R2       equ      22h      ; 迴圈控制變數

NUM      equ      30h      ; 儲存使用者輸入的數字
NUMS1    equ      31h      ; 儲存使用者輸入的第1個數字
NUMS2    equ      32h      ; 儲存使用者輸入的第2個數字
NUMS3    equ      33h      ; 儲存使用者輸入的第3個數字
NUMS4    equ      34h      ; 儲存使用者輸入的第4個數字
NUML1    equ      35h      ; 儲存由快閃記憶體讀出的第1個數字
NUML2    equ      36h      ; 儲存由快閃記憶體讀出的第2個數字
NUML3    equ      37h      ; 儲存由快閃記憶體讀出的第3個數字
NUML4    equ      38h      ; 儲存由快閃記憶體讀出的第4個數字
```

；定義程式中使用到的字串

```
ZF       .defstr   03h, 2   ; Zero旗號

RCVBF    .defstr   1bh, 6   ; SPI資料接收狀態旗號
```

TXBEPY	.defstr	1bh, 5	; SPI資料傳送狀態旗號
; 定義ICE25P05快閃記憶體的指令碼			
OP_WREN	equ	06H	; 允許寫入
OP_WRDI	equ	04H	; 不允許寫入
OP_RDSR	equ	05H	; 讀取狀態暫存器
OP_WRSR	equ	01H	; 寫入狀態暫存器
OP_READ	equ	03H	; 讀取資料位元組
OP_PP	equ	02H	; 寫入分頁資料
OP_SE	equ	D8H	; 清除資料區
OP_BE	equ	C7H	; 清除全部資料
OP_RDID	equ	15H	; 讀取裝置識別碼
; 系統開機進入點			
	org	00h	
; 初始化SPI			
Start:	movlw	100101b	; 設定SPI運作模式MSTR=1, CPOL=0,
	movwr	SPICR	; CPHA=0,SPEB=1, LSBFE=0, SPE=1
	movlw	0001001b	; 設定SPI Baud Rate=4M/2/4=512K bps
	movwr	SPIBR	; SPPR=SPR=1
; 初始化ICE25P05控制腳位			
	bsf	PFD, 0	; 將/hold設為1, 啟用ICE25P05
	bsf	PFD, 1	; 將/w設為1, 關閉寫入保護
	bsf	PFD, 2	; 將/s設為1, 關閉讀寫ICE25P05功能
; 致能鍵盤輸入腳位Push-pull功能			
	movlw	ffh	
	movwr	PEE	
; 測試快閃記憶體			
TestFlash:	call	GetNum	; 等待使用者輸入一個0~9的數字
	movfw	NUM	; 取出數字
	movwf	NUMS1	; 將數字存入NUMS1
	call	GetNum	; 等待使用者輸入一個0~9的數字
	movfw	NUM	; 取出數字
	movwf	NUMS2	; 將數字存入NUMS2
	call	GetNum	; 等待使用者輸入一個0~9的數字
	movfw	NUM	; 取出數字
	movwf	NUMS3	; 將數字存入NUMS3
	call	GetNum	; 等待使用者輸入一個0~9的數字
	movfw	NUM	; 取出數字

	movwf	NUMS4	; 將數字存入NUMS4
	call	SaveNums	; 將所有數字存入Flash Memory
	call	ReadNums	; 將數字由Flash Memory讀出
	call	ShowNums	; 將數字顯示在七段顯示器上
	goto	TestFlash	; 回TestFlash重新測試
; 等待使用者輸入數字副程式，使用者要按下'0' ~ '9'按鍵才會離開此副程式			
; 使用者輸入數字的BCD碼放在NUM內			
GetNum:	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	movlw	0fh	; 將鍵盤列位址(bits4~7)變成低電位
	movwf	PED	; 將鍵盤行位址(bits0~3)變成高電位
	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是0FH
	xorlw	0fh	; 沒按鍵xor結果為0, ZF=1
	btss	ZF	; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
	goto	DeNoise	; 有按鍵，跳到DeNoise去除彈跳
	goto	GetNum	; 沒按鍵，回到GetNum等待按鍵
DeNoise:	call	Delay	; 去除彈跳
; 檢查是否有按鍵被按下			
	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	movlw	0fh	; 將鍵盤列位址(bits4~7)變成低電位
	movwf	PED	; 將鍵盤行位址(bits0~3)變成高電位
	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是0FH
	xorlw	0fh	; 沒按鍵xor結果為0, ZF=1
	btss	ZF	; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
	goto	Row1	; 有按鍵，到Row1檢查按了什麼鍵
	goto	GetNum	; 沒按鍵，回到GetNum等待按鍵
Row1:	; 檢查第一列是否有按鍵被按下		
	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	movlw	efh	; 將鍵盤第一列(bit 4)變成低電位
	movwf	PED	;
	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是efh
	xorlw	efh	; 沒按鍵xor結果為0, ZF=1
	btss	ZF	; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
	goto	Key0	; 有按鍵，檢查是否按了'0'
	goto	Row2	; 沒按鍵，檢查第二列
Key0:	btsc	PED, 0	; 檢查'0'是否被按下
	goto	Key1	; '0'沒被按下，檢查是否按了'1'
	movlw	00h	; '0'被按下，記下'0'的BCD碼
	goto	WaitRelease	; 等待按鍵放開

Key1:	<pre>                     btfsc    PED, 1    ; 檢查'1'是否被按下                     goto     Key2     ; '1'沒被按下，檢查是否按了'2'                     movlw   01h      ; '1'被按下，記下'1'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Key2:	<pre>                     btfsc    PED, 2    ; 檢查'2'是否被按下                     goto     Key3     ; '2'沒被按下，所以是'3'被按下                     movlw   02h      ; '2'被按下，記下'2'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Key3:	<pre>                     movlw   03h      ; '3'被按下，記下'3'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Row2:	<pre>                     ; 檢查第二列是否有按鍵被按下                     movlw   ffh      ; 清除按鍵狀態                     movwf   PED      ;                     movlw   dfh      ; 將鍵盤第二列(bit 5)變成低電位                     movwf   PED      ;                     movfw   PED      ; 讀取鍵盤狀態，沒按鍵應是dfh                     xorlw   dfh      ; 沒按鍵xor結果為0, ZF=1                     btfss   ZF      ; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵                     goto     Key4     ; 有按鍵，檢查是否按了'4'                     goto     Row3     ; 沒按鍵，檢查第三列                 </pre>
Key4:	<pre>                     btfsc    PED, 0    ; 檢查'4'是否被按下                     goto     Key5     ; '4'沒被按下，檢查是否按了'5'                     movlw   04h      ; '4'被按下，記下'4'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Key5:	<pre>                     btfsc    PED, 1    ; 檢查'5'是否被按下                     goto     Key6     ; '5'沒被按下，檢查是否按了'6'                     movlw   05h      ; '5'被按下，記下'5'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Key6:	<pre>                     btfsc    PED, 2    ; 檢查'6'是否被按下                     goto     Key7     ; '6'沒被按下，所以是'7'被按下                     movlw   06h      ; '6'被按下，記下'6'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Key7:	<pre>                     movlw   07h      ; '7'被按下，記下'7'的BCD碼                     goto     WaitRelease ; 等待按鍵放開                 </pre>
Row3:	<pre>                     ; 檢查第三列是否有按鍵被按下                     movlw   ffh      ; 清除按鍵狀態                     movwf   PED      ;                     movlw   bfh      ; 將鍵盤第三列(bit 6)變成低電位                     movwf   PED      ;                 </pre>

	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是bfh
	xorlw	bfh	; 沒按鍵xor結果為0, ZF=1
	btffss	ZF	; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
	goto	Key8	; 有按鍵，檢查是否按了'8'
	goto	GetNum	; 不是'0' ~ '9'被按下，回到GetNum
Key8:	btffsc	PED, 0	; 檢查'8'是否被按下
	goto	Key9	; '8'沒被按下，檢查是否按了'9'
	movlw	08h	; '8'被按下，記下'8'的BCD碼
	goto	WaitRelease	; 等待按鍵放開
Key9:	btffsc	PED, 1	; 檢查'9'是否被按下
	goto	GetNum	; 不是'0' ~ '9'被按下，回到GetNum
	movlw	09h	; '9'被按下，記下'9'的BCD碼
	goto	WaitRelease	; 等待按鍵放開
			; 等待使用者放開按鍵
WaitRelease:	movwf	NUM	; 記下按鍵'的BCD碼
WR_Loop:	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	call	Delay	; 時間延遲
	movlw	0fh	; 將鍵盤列位址(bits4~7)變成低電位
	movwf	PED	; 將鍵盤行位址(bits0~3)變成高電位
	movfw	PED	; 讀取鍵盤狀態，若放開按鍵應是0fh
	xorlw	0fh	; 沒按鍵xor結果為0, ZF=1
	btffss	ZF	; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
	goto	WR_Loop	; 沒放開按鍵，繼續等待
	ret		; 回到呼叫程式
			; 將數字寫入快閃記憶體副程式
			; 要寫入快閃記憶體的位元組放在NUMS1, NUMS2, NUMS3, NUMS4內
SaveNums:			; 寫出WREN 指令，允許寫入ICE25P05
	bcf	PFD, 2	; 設定/s=0，開啟讀寫ICE25P05功能
	movlw	OP_WREN	; 載入WREN 指令碼
	call	WriteSPI	; 將WREN指令傳送出去
	bsf	PFD, 2	; 設定/s=1，關閉讀寫ICE25P05功能
			; 寫出WRSR 指令，設定ICE25P05狀態
	bcf	PFD, 2	; 設定/s=0，開啟讀寫ICE25P05功能
	movlw	OP_WRSR	; 載入WRSR指令碼
	call	WriteSPI	; 將WRSR指令傳送出去
	movlw	00h	; 設定狀態暫存器
	call	WriteSPI	; 將狀態暫存器內容傳送出去
	bsf	PFD, 2	; 設定/s=1，關閉讀寫ICE25P05功能
			; 寫出WREN 指令，允許寫入ICE25P05
	bcf	PFD, 2	; 設定/s=0，開啟讀寫ICE25P05功能
	movlw	OP_WREN	; 載入WREN 指令碼

	call	WriteSPI	; 將WREN指令傳送出去
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出SE 指令, 清除資料區		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_SE	; 載入SE指令碼
	call	WriteSPI	; 將SE指令傳送出去
	movlw	00h	; 設定位址第一個位元組為00
	call	WriteSPI	; 傳送位址的第一個位元組
	movlw	00h	; 設定位址第二個位元組為00
	call	WriteSPI	; 傳送位址的第二個位元組
	movlw	00h	; 設定位址第三個位元組為00
	call	WriteSPI	; 傳送位址的第三個位元組
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出RDSR 指令, 讀取狀態暫存器內容, 等待flash完成動作		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_RDSR	; 載入RDSR指令碼
tsWIP:	call	WriteSPI	; 將RDSR指令傳送出去
	movlw	ffh	; 載入ffh, 要收SPI資料, 必須寫入SPID
	call	WriteSPI	; 接收狀態暫存器內容
	movwf	data	; 將狀態暫存器內容放入data
	btfs	data, 0	; 測試WIP是否為0(WIP=0表示不忙碌)
	goto	tsWIP	; 忙碌中, 繼續檢查, 直到不忙碌
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出WREN 指令, 允許寫入ICE25P05		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_WREN	; 載入WREN 指令碼
	call	WriteSPI	; 將WREN指令傳送出去
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出PP指令, 要求寫入資料		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_PP	; 載入PP指令碼
	call	WriteSPI	; 將PP指令傳送出去
	movlw	00h	; 設定位址第一個位元組為00
	call	WriteSPI	; 傳送位址的第一個位元組
	movlw	00h	; 設定位址第二個位元組為00
	call	WriteSPI	; 傳送位址的第二個位元組
	movlw	00h	; 設定位址第三個位元組為00
	call	WriteSPI	; 傳送位址的第三個位元組
	movfw	NUMS1	; 載入第一個數字
	call	WriteSPI	; 將第一個數字寫入Flash
	movfw	NUMS2	; 載入第二個數字

```

call      WriteSPI      ; 將第二個數字寫入Flash
movfw    NUMS3          ; 載入第三個數字
call     WriteSPI      ; 將第三個數字寫入Flash
movfw    NUMS4          ; 載入第四個數字
call     WriteSPI      ; 將第四個數字寫入Flash
bsf      PFD, 2         ; 結束資料寫入動作

; 寫出RDSR 指令，讀取狀態暫存器內容，等待flash完成動作
bcf      PFD, 2         ; 設定/s=0，開啟讀寫ICE25P05功能
movlw    OP_RDSR       ; 載入RDSR指令碼
call     WriteSPI      ; 將RDSR指令傳送出去
tsWIP2:  movlw    ffh    ; 載入ffh，要收SPI資料，必須寫入SPID
call     WriteSPI      ; 接收狀態暫存器內容
movwf    data          ; 將狀態暫存器內容放入data
btfsc   data, 0        ; 測試WIP是否為0(WIP=0表示不忙碌)
goto    tsWIP2         ; 忙碌中，繼續檢查，直到不忙碌
bsf      PFD, 2         ; 設定/s=1，關閉讀寫ICE25P05功能
ret

; 由快閃記憶體讀出數字副程式
; 讀到的數字放在NUML1, NUML2, NUML3, NUML4內
ReadNums: bcf      PFD, 2         ; 設定/s=0，開啟讀寫ICE25P05功能
movlw    OP_READ      ; 載入READ指令碼
call     WriteSPI     ; 將RDSR指令傳送出去

movlw    00h          ; 設定位址第一個位元組為00
call     WriteSPI     ; 傳送位址的第一個位元組
movlw    00h          ; 設定位址第二個位元組為00
call     WriteSPI     ; 傳送位址的第二個位元組
movlw    00h          ; 設定位址第三個位元組為00
call     WriteSPI     ; 傳送位址的第三個位元組
movlw    ffh          ; 載入ffh，要收SPI資料，必須寫入SPID
call     WriteSPI     ; 讀入第一個數字
movwf    NUML1        ; 將數字放入NUML1

movlw    ffh          ; 載入ffh，要收SPI資料，必須寫入SPID
call     WriteSPI     ; 讀入第二個數字
movwf    NUML2        ; 將數字放入NUML2

movlw    ffh          ; 載入ffh，要收SPI資料，必須寫入SPID
call     WriteSPI     ; 讀入第三個數字
movwf    NUML3        ; 將數字放入NUML3

movlw    ffh          ; 載入ffh，要收SPI資料，必須寫入SPID
call     WriteSPI     ; 讀入第四個數字
movwf    NUML4        ; 將數字放入NUML4

bsf      PFD, 2         ; 結束資料讀取動作

```

```

ret                                ; 返回呼叫函式

; SPI傳送接收資料副程式
; 輸入參數：要傳送的資料放在working register內
; 輸出參數：由SPI讀到的資料放在working register內
WriteSPI:  btfss      TXBEPY      ; 檢查SPI目前是否可傳送資料
           goto      WriteSPI     ; 不能傳，重新檢查，直到可傳
           movwf     SPID         ; 將要傳送的資料寫入SPID暫存器
chkRcv:   btfss      RCVBF       ; 檢查是否收到資料
           goto      chkRcv      ; 沒收到資料，等待資料進來
           movfw     SPID        ; 取出收到的資料
           ret          ; 返回呼叫程式

; 顯示數字副程式
; 將NUML1~NUML4數字顯示到四顆七段顯示器
ShowNums: swapf      NUML1       ; 對調NUML1高低半位元組(Nibble)
           movfw     NUML1       ; 載入NUML1
           lorwf     NUML2,0     ; 將NUML1高半位元組併入NUML2
           movwf     PBD        ; 顯示前兩個數字

           swapf    NUML3       ; 對調NUML3高低半位元組(Nibble)
           movfw     NUML3       ; 載入NUML3
           lorwf     NUML4,0     ; 將NUML3高半位元組併入NUML4
           movwf     PDD        ; 顯示後兩個數字
           ret          ; 返回呼叫程式

; 延遲0.005秒副程式
Delay:    movlw     10           ; 設定外層迴圈執行10次
           movwf     R1          ;

           ; 外層迴圈
Delay_L1: movlw     200         ; 設定內層迴圈執行200次
           movwf     R2          ;

           ; 內層迴圈：迴圈跑一次約消耗5個指令週期
Delay_L2: nop                ; 消耗1個指令週期
           nop                ; 消耗1個指令週期
           decfsz    R2, 1     ; 約消耗1個指令週期
           goto     Delay_L2   ; 消耗2個指令週期

           decfsz    R1,1     ; 將R1減1，若R1=0離開迴圈
           goto     Delay_L1   ;
           ret          ; 返回呼叫程式

```

## 15. 密碼鎖實習

### 實習目的：

本實習主要目的在學習如何使用十速 57FLA80 系列單晶片實作密碼鎖系統。

### 實習設備：

項次	品名	數量
1	十速 TICE99 模擬器	1
2	電源供應器	1
3	麵包板	1

### 實習材料：

項次	品名	數量
1	10KΩ排阻(A-type 9PIN)	3
2	20KΩ排阻(A-type 9PIN)	2
3	10KΩ電阻	1
4	ICE25P05 快閃記憶體	1
5	7417 IC	1
6	HD44780 相容文字型 LCM (可顯示兩列文字，一列 20 字)	1
7	4 × 4 鍵盤	1

### 實習板模組與 I/O Port：

模組	I/O Port
文字型 LCD 顯示模組	Port B
文字型 LCD 顯示模組	Port D
4 × 4 鍵盤模組	Port E
FLASHROM 模組	Port F
FLASHROM 模組	Port G

**實習說明：**

本實習將實作一個密碼鎖系統，此系統首先會在 LCD 螢幕顯示"Enter Password:"字串，等待使用者按下密碼，使用者可以利用 4x4 鍵盤輸入 4 個數字的密碼，密碼輸入錯誤可按'A'鍵重新輸入密碼。當使用者輸入完 4 個數字後，系統會取出儲存在快閃記憶體內的密碼跟使用者輸入的數字作比對，密碼比對正確，顯示"Welcome to Tenx!!"，否則顯示錯誤訊息"Error!"，密碼比對完畢，使用者可按下'A'鍵要求使用者重新輸入密碼。任何時候，只要使用者按下按鍵'B'，便可重新設置密碼。當使用者按下'B'時，LCD 畫面會顯示"Please Set Password:"訊息，要求使用者輸入新的 4 位數密碼，並將新的密碼儲存在快閃記憶體內。

**硬體線路圖：**

本實習硬體線路圖如圖 15-1 所示，分別使用 Port B 與 Port D 控制 LCM 的顯示，使用 Port E 讀取 4x4 鍵盤，以及使用 PG0 ~ PG2 (SDI、SDO、SCK) 與 PF0 ~ PF2 (/HOLD、/W、/S) 進行快閃記憶體的讀寫控制。

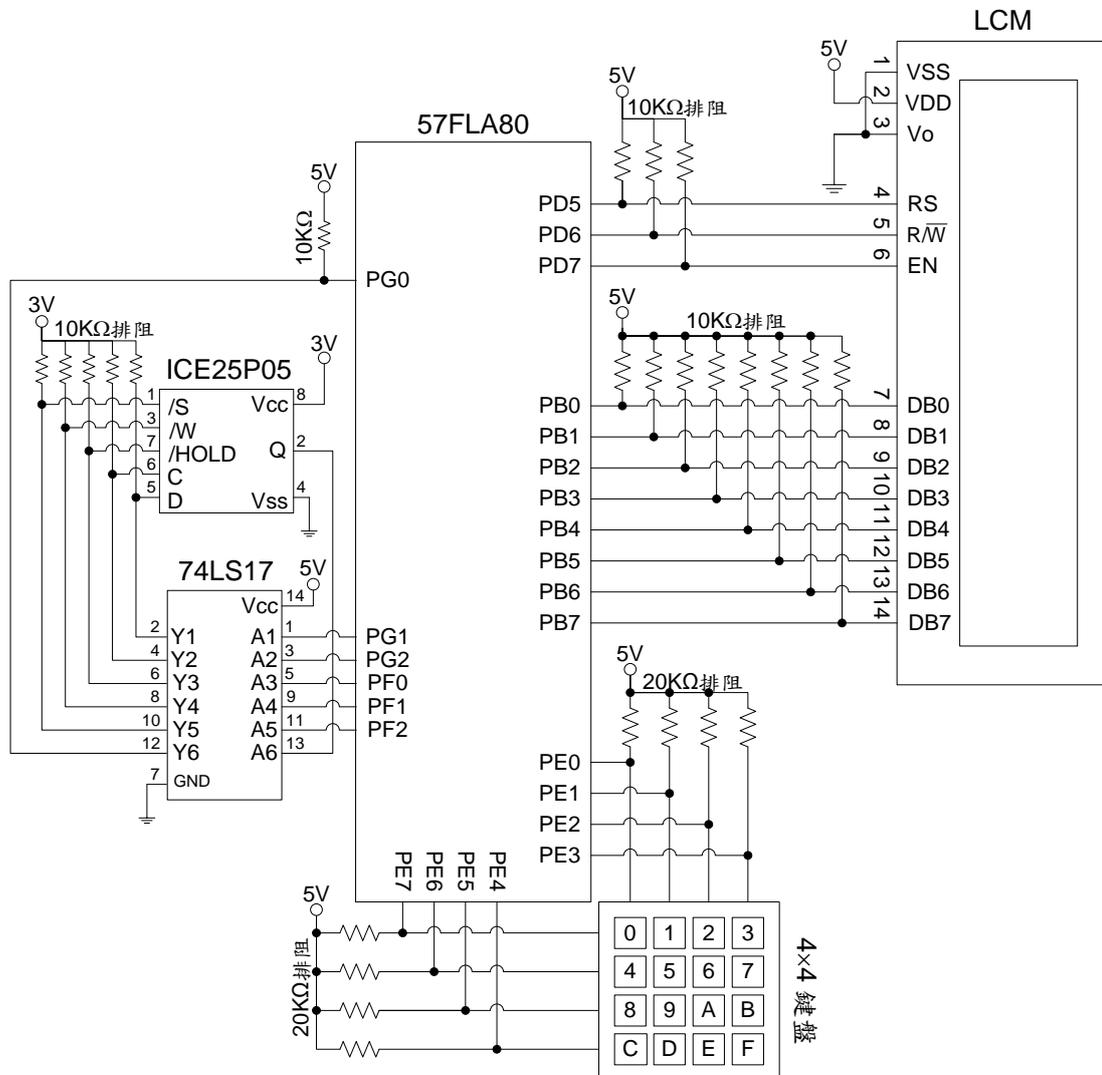


圖 15-1 密碼鎖硬體線路圖

程式流程：

本實習程式流程如圖 15-2 所示，首先進行 SPI、ICE25P05 控制腳位、以及 LCM 初始設定，然後顯示提示訊息等待使用者輸入密碼，使用者可按數字鍵輸入密碼，輸入錯誤可按'A'鍵重新輸入，或按'B'鍵設定密碼。使用者輸入完 4 個數字，程式取出快閃記憶體內的密碼，跟使用者輸入的數字作比對以及顯示比對結果。

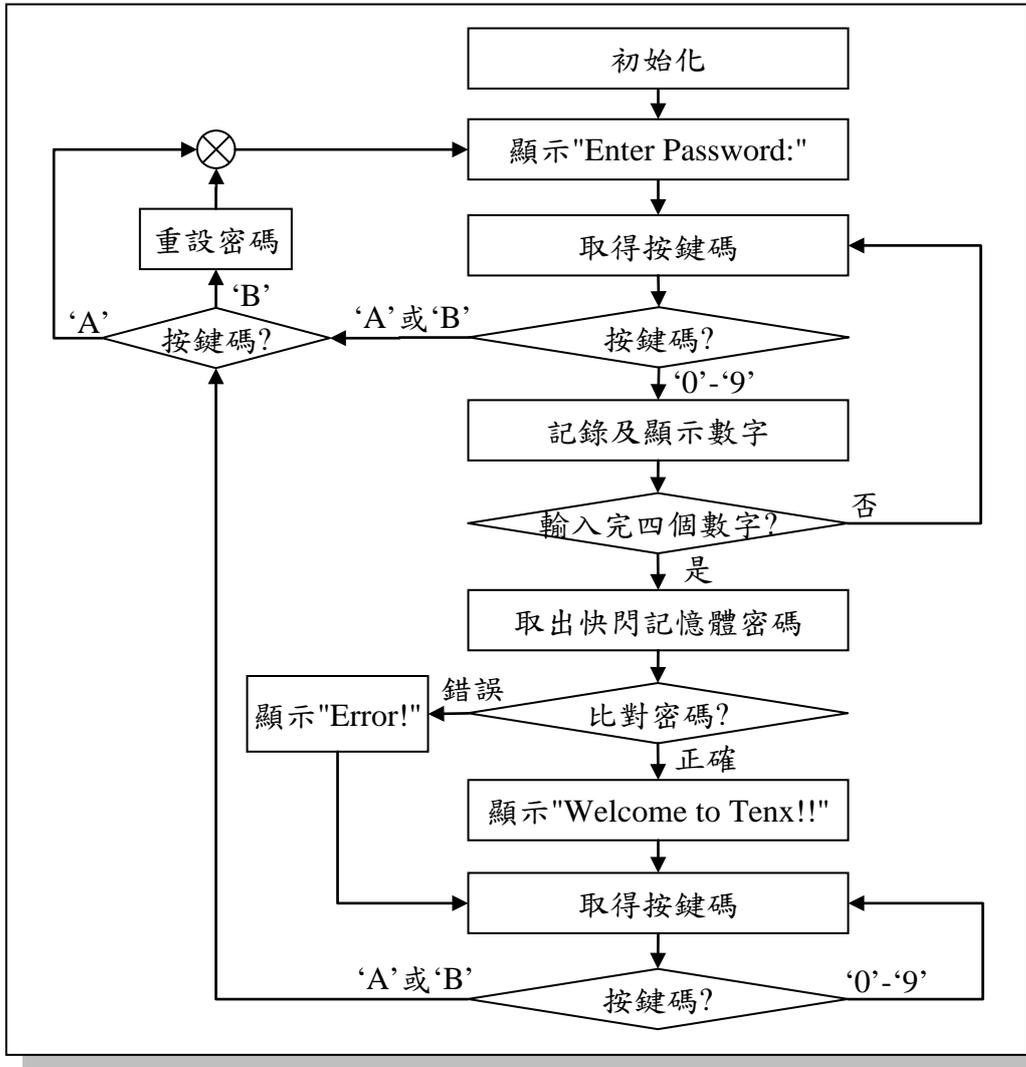


圖 15-2 密碼鎖程式流程

## 程式碼及程式說明：

; 定義程式中使用到的F-Plane暫存器記憶體位址			
PC	equ	02h	; Program Counter
PBD	equ	06h	; Port B
PDD	equ	12h	; Port D
PED	equ	13h	; Port E
PFD	equ	14h	; Port F
SPID	.defstr	1ah	; SPI資料暫存器
; 定義程式中使用到的R-Plane暫存器記憶體位址			
PEE	equ	13h	; Port E Push-Pull enable
SPIBR	equ	18h	; SPI Baud Rate設定暫存器 ; 位元6 ~ 4 : SPPR ; 位元2 ~ 0 : SPR
SPICR	equ	19h	; SPI 控制暫存器 ; 位元5 : MSTR, SPI主/從裝置設定 ; 位元4 : CPOL, SPI Clock極性設定 ; 位元3 : CPHA, SPI Clock相位設定 ; 位元2 : SPEB, SPI除能設定 ; 位元1 : LSBFE, SPI資料傳送順序設定 ; 位元0 : SPE, SPI致能設定
; 定義程式中使用到的F-Plane變數記憶體位址			
data	equ	20h	; 儲存要傳送的資料
strID	equ	21h	; 儲存要顯示字串的代碼 ; 0 : "Enter Password:" ; 1 : "Welcome to Tenx!!" ; 2 : "Please Set Password:" ; 3 : "Error!"
strIndex	equ	22h	; 記錄字元在字串中的位置資訊
R1	equ	23h	; 迴圈控制變數
R2	equ	24h	; 迴圈控制變數
KEY	equ	30h	; 儲存使用者輸入的按鍵碼
N1	equ	31h	; 儲存使用者輸入的第1個數字
N2	equ	32h	; 儲存使用者輸入的第2個數字
N3	equ	33h	; 儲存使用者輸入的第3個數字
N4	equ	34h	; 儲存使用者輸入的第4個數字
PW1	equ	35h	; 儲存由快閃記憶體讀出的第1個數字

PW2	equ	36h	; 儲存由快閃記憶體讀出的第2個數字
PW3	equ	37h	; 儲存由快閃記憶體讀出的第3個數字
PW4	equ	38h	; 儲存由快閃記憶體讀出的第4個數字
; 定義程式中使用到的字串			
CF	.defstr	03h, 0	; 進位旗號
ZF	.defstr	03h, 2	; Zero旗號
RS_LCM	.defstr	12h, 5	; LCM RS控制位元
RW_LCM	.defstr	12h, 6	; LCM RW控制位元
EN_LCM	.defstr	12h, 7	; LCM EN控制位元
RCVBF	.defstr	1bh, 6	; SPI資料接收狀態旗號
TXBEPY	.defstr	1bh, 5	; SPI資料傳送狀態旗號
; 定義ICE25P05快閃記憶體的指令碼			
OP_WREN	equ	06H	; 允許寫入
OP_WRDI	equ	04H	; 不允許寫入
OP_RDSR	equ	05H	; 讀取狀態暫存器
OP_WRSR	equ	01H	; 寫入狀態暫存器
OP_READ	equ	03H	; 讀取資料位元組
OP_PP	equ	02H	; 寫入分頁資料
OP_SE	equ	D8H	; 清除資料區
OP_BE	equ	C7H	; 清除全部資料
OP_RDID	equ	15H	; 讀取裝置識別碼
; 系統開機進入點			
	org	00h	
; 初始化SPI			
Start:	movlw	00100101b	; 設定SPI運作模式MSTR=1, CPOL=0,
	movwr	SPICR	; CPHA=0, SPEB=1, LSBFE=0, SPE=1
	movlw	0001001b	; 設定SPI Baud Rate=4M/2/4=512K bps
	movwr	SPIBR	; SPPR=SPR=1
; 初始化ICE25P05控制腳位			
	bsf	PFD, 0	; 將/hold設為1, 啟用ICE25P05
	bsf	PFD, 1	; 將/w設為1, 關閉寫入保護
	bsf	PFD, 2	; 將/s設為1, 關閉讀寫ICE25P05功能
; 致能鍵盤輸入腳位Push-pull功能			
	movlw	ffh	
	movwr	PEE	
; 初始化LCM			
	call	Delay	; 等候LCM啟動

	call	LCMInitialize	; 進行LCM初始設定
Reset:	call	ClearScreen	; 清除LCD螢幕
	call	MoveToRow1	; 將游標移至LCD第一行開頭
	movlw	00h	;
	movwf	strID	; 選擇第0個字串"Enter Password"
	call	ShowStr	; 將字串顯示到LCD
ReTry:	call	MoveToRow2	; 將游標移至LCD第二行開頭
	call	GetKey	; 呼叫按鍵副程式
	btfsc	KEY, 5	; 檢查按鍵代碼是不是41H或42H
	goto	SetNum1	; 不是41H或42H, 儲存第1個數字
	goto	ProcessCMD	; 檢查及執行命令
SetNum1:	movfw	KEY	; 載入讀到的數字
	movwf N1		; 儲存使用者輸入的第1個數字
	movlw	'*	; 載入 '*'
	call	LcmWriteDR	; 將 '*' 顯示到LCD
		call	GetKey
SetNum2:	btfsc	KEY, 5	; 檢查按鍵代碼是不是41H或42H
	goto	SetNum2	; 不是41H或42H, 儲存第2個數字
	goto	ProcessCMD	; 檢查及執行命令
	movfw	KEY	; 載入讀到的數字
	movwf N2		; 儲存使用者輸入的第2個數字
	movlw	'*	; 載入 '*'
	call	LcmWriteDR	; 將 '*' 顯示到LCD
SetNum3:	call	GetKey	; 呼叫按鍵副程式
	btfsc	KEY, 5	; 檢查按鍵代碼是不是41H或42H
	goto	SetNum3	; 不是41H或42H, 儲存第3個數字
	goto	ProcessCMD	; 檢查及執行命令
	movfw	KEY	; 載入讀到的數字
	movwf N3		; 儲存使用者輸入的第3個數字
	movlw	'*	; 載入 '*'
	call	LcmWriteDR	; 將 '*' 顯示到LCD
SetNum4:	call	GetKey	; 呼叫按鍵副程式
	btfsc	KEY, 5	; 檢查按鍵代碼是不是41H或42H
	goto	SetNum4	; 不是41H或42H, 儲存第4個數字
	goto	ProcessCMD	; 檢查及執行命令
	movfw	KEY	; 載入讀到的數字
	movwf N4		; 儲存使用者輸入的第4個數字
	movlw	'*	; 載入 '*'
	call	LcmWriteDR	; 將 '*' 顯示到LCD
	call	ReadPW	; 讀出存在快閃記憶體內的密碼
	call	CheckPW	; 檢查密碼是否正確
	btfss	ZF	; ZF=1, 密碼正確, ZF=0, 密碼錯誤

	goto	ShowErrMsg	; ZF=0, 密碼錯誤, 顯示錯誤訊息
			; ZF=1, 密碼正確, 顯示密碼正確訊息
	call	ClearScreen	; 清除LCD螢幕
	call	MoveToRow1	; 將游標移至LCD第一行開頭
	movlw	01h	;
	movwf	strID	; 選擇第1個字串"Welcome to Tenx!!"
	call	ShowStr	; 將字串顯示到LCD
			; 等待使用者按下'A'或'B'
WaitCMD:	call	GetKey	; 呼叫按鍵副程式, 等待使用者下達命令
	btfs	KEY, 5	; 檢查按鍵代碼是不是41H或42H
	goto	WaitCMD	; 不是'A'或'B', 重新等待按鍵
	goto	ProcessCMD	; 檢查及執行命令
			; 顯示錯誤訊息
ShowErrMsg:	call	ClearScreen	; 清除LCD螢幕
	call	MoveToRow1	; 將游標移至LCD第一行開頭
	movlw	03h	;
	movwf	strID	; 選擇第4個字串"Error!"
	call	ShowStr	; 將字串顯示到LCD
	goto	WaitCMD	; 等待命令
ProcessCMD:	movfw	KEY	; 載入按鍵代碼
	xorlw	41h	; 檢查是否按下'A'
	btfs	ZF	; 按下'A' ZF=1, 按下'B' ZF=0
	goto	SetPW	; 按下'B', 重設密碼
	goto	Reset	; 請使用者重新輸入密碼
			; 檢查輸入密碼是否正確副程式
			; 輸入參數: NUMS1~NUMS4: 使用者輸入數字, NUML1~NUML4: 密碼
			; 輸出參數: ZF=1表正確, ZF=0表錯誤
CheckPW:	movfw	N1	; 載入密碼第1個數字
	xorwf	PW1, 0	; 若NUMS1=NUML1, XOR結果為0
	btfs	ZF	; ZF=1密碼正確, ZF=0密碼錯誤
	ret		; ZF=0密碼錯誤, 結束密碼檢查
	movfw	N2	; 載入密碼第2個數字
	xorwf	PW2, 0	; 若NUMS2=NUML2, XOR結果為0
	btfs	ZF	; ZF=1密碼正確, ZF=0密碼錯誤
	ret		; ZF=0密碼錯誤, 結束密碼檢查
	movfw	N3	; 載入密碼第3個數字
	xorwf	PW3, 0	; 若NUMS3=NUML3, XOR結果為0
	btfs	ZF	; ZF=1密碼正確, ZF=0密碼錯誤
	ret		; ZF=0密碼錯誤, 結束密碼檢查
	movfw	N4	; 載入密碼第4個數字

```

xorwf    PW4, 0    ; 若NUMS4=NUML4，XOR結果為0
ret      ; 結束密碼檢查

; 設定密碼副程式
SetPW:   call      ClearScreen ; 清除LCD螢幕
         call      MoveToRow1 ; 將游標移至LCD第一行開頭
         movlw     02h        ;
         movwf     strlD      ; 選擇第2個字串" Please Set Password:"
         call      ShowStr    ; 將字串顯示到LCD

         call      MoveToRow2 ; 將游標移至LCD第二行開頭

         call      GetKey     ; 呼叫按鍵副程式
         btfsc    KEY, 5     ; 檢查按鍵代碼是不是41H或42H
         goto     SetPW1     ; 不是41H或42H, 儲存第1個數字
         goto     ProcessCMD ; 檢查及執行命令
SetPW1:  movfw     KEY        ; 載入讀到的數字
         movwf     PW1        ; 儲存密碼第1個數字
         call      LcmWriteDR ; 將輸入的數字顯示到LCD

         call      GetKey     ; 呼叫按鍵副程式
         btfsc    KEY, 5     ; 檢查按鍵代碼是不是41H或42H
         goto     SetPW2     ; 不是41H或42H, 儲存第2個數字
         goto     ProcessCMD ; 檢查及執行命令
SetPW2:  movfw     KEY        ; 載入讀到的數字
         movwf     PW2        ; 儲存密碼第2個數字
         call      LcmWriteDR ; 將輸入的數字顯示到LCD

         call      GetKey     ; 呼叫按鍵副程式
         btfsc    KEY, 5     ; 檢查按鍵代碼是不是41H或42H
         goto     SetPW3     ; 不是41H或42H, 儲存第3個數字
         goto     ProcessCMD ; 檢查及執行命令
SetPW3:  movfw     KEY        ; 載入讀到的數字
         movwf     PW3        ; 儲存密碼第3個數字
         call      LcmWriteDR ; 將輸入的數字顯示到LCD

         call      GetKey     ; 呼叫按鍵副程式
         btfsc    KEY, 5     ; 檢查按鍵代碼是不是41H或42H
         goto     SetPW4     ; 不是41H或42H, 儲存第4個數字
         goto     ProcessCMD ; 檢查及執行命令
SetPW4:  movfw     KEY        ; 載入讀到的數字
         movwf     PW4        ; 儲存密碼第4個數字
         call      LcmWriteDR ; 將輸入的數字顯示到LCD

         call      SavePW     ; 把輸入密碼存入快閃記憶體
         ret      ; 回到主程式

; ***** 4x4 鍵盤相關副程式 *****

```

```

; 等待使用者輸入數字或命令副程式，
; 使用者要按下'0' ~ '9', 'A', 或'B'按鍵才會離開此副程式
; 輸出參數：KEY記憶體存放按鍵的ASCII碼
GetKey:    movlw    ffh          ; 清除按鍵狀態
           movwf   PED          ;
           movlw   0fh          ; 將鍵盤列位址(bits4~7)變成低電位
           movwf   PED          ; 將鍵盤行位址(bits0~3)變成高電位
           movfw   PED          ; 讀取鍵盤狀態，沒按鍵應是0FH
           xorlw   0fh          ; 沒按鍵xor結果為0, ZF=1
           btfss  ZF          ; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
           goto   DeNoise      ; 有按鍵，跳到DeNoise去除彈跳
           goto   GetKey       ; 沒按鍵，回到GetKey等待按鍵

DeNoise:   call    Delay        ; 去除彈跳，等待5ms
           movlw   ffh          ; 清除按鍵狀態
           movwf   PED          ;
           movlw   0fh          ; 將鍵盤列位址(bits4~7)變成低電位
           movwf   PED          ; 將鍵盤行位址(bits0~3)變成高電位
           movfw   PED          ; 讀取鍵盤狀態，沒按鍵應是0FH
           xorlw   0fh          ; 沒按鍵xor結果為0, ZF=1
           btfss  ZF          ; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
           goto   Row1         ; 有按鍵，到Row1檢查按了什麼鍵
           goto   GetKey       ; 沒按鍵，回到GetKey等待按鍵

Row1:      ; 檢查第一列是否有按鍵被按下
           movlw   ffh          ; 清除按鍵狀態
           movwf   PED          ;
           movlw   efh          ; 將鍵盤第一列(bit 4)變成低電位
           movwf   PED          ;
           movfw   PED          ; 讀取鍵盤狀態，沒按鍵應是efh
           xorlw   efh          ; 沒按鍵xor結果為0, ZF=1
           btfss  ZF          ; 檢查ZF，ZF=1沒按鍵，ZF=0有按鍵
           goto   Key0         ; 有按鍵，檢查是否按了'0'
           goto   Row2        ; 沒按鍵，檢查第二列

Key0:      btfsc   PED, 0       ; 檢查'0'是否被按下
           goto   Key1         ; '0'沒被按下，檢查是否按了'1'
           movlw   30h          ; '0'被按下
           goto   WaitRelease  ; 等待按鍵放開

Key1:      btfsc   PED, 1       ; 檢查'1'是否被按下
           goto   Key2         ; '1'沒被按下，檢查是否按了'2'
           movlw   31h          ; '1'被按下
           goto   WaitRelease  ; 等待按鍵放開

Key2:      btfsc   PED, 2       ; 檢查'2'是否被按下
           goto   Key3         ; '2'沒被按下，所以是'3'被按下
           movlw   32h          ; '2'被按下

```

	goto	WaitRelease	; 等待按鍵放開
Key3:	movlw	33h	; '3'被按下
	goto	WaitRelease	; 等待按鍵放開
Row2:	; 檢查第二列是否有按鍵被按下		
	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	movlw	dfh	; 將鍵盤第二列(bit 5)變成低電位
	movwf	PED	;
	movfw	PED	; 讀取鍵盤狀態，沒按鍵應是dfh
	xorlw	dfh	; 沒按鍵xor結果為0, ZF=1
	btss	ZF	; 檢查ZF, ZF=1沒按鍵, ZF=0有按鍵
	goto	Key4	; 有按鍵, 檢查是否按了'4'
	goto	Row3	; 沒按鍵, 檢查第三列
Key4:	btsc	PED, 0	; 檢查'4'是否被按下
	goto	Key5	; '4'沒被按下, 檢查是否按了'5'
	movlw	34h	; '4'被按下
	goto	WaitRelease	; 等待按鍵放開
Key5:	btsc	PED, 1	; 檢查'5'是否被按下
	goto	Key6	; '5'沒被按下, 檢查是否按了'6'
	movlw	35h	; '5'被按下
	goto	WaitRelease	; 等待按鍵放開
Key6:	btsc	PED, 2	; 檢查'6'是否被按下
	goto	Key7	; '6'沒被按下, 所以是'7'被按下
	movlw	36h	; '6'被按下
	goto	WaitRelease	; 等待按鍵放開
Key7:	movlw	37h	; '7'被按下
	goto	WaitRelease	; 等待按鍵放開
Row3:	; 檢查第三列是否有按鍵被按下		
	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	movlw	bffh	; 將鍵盤第三列(bit 6)變成低電位
	movwf	PED	;
	movfw	PED	; 讀取鍵盤狀態, 沒按鍵應是bffh
	xorlw	bffh	; 沒按鍵xor結果為0, ZF=1
	btss	ZF	; 檢查ZF, ZF=1沒按鍵, ZF=0有按鍵
	goto	Key8	; 有按鍵, 檢查是否按了'8'
	goto	GetKey	; 不是'0'~'9', 'A', 'B'被按下, 回到GetKey
Key8:	btsc	PED, 0	; 檢查'8'是否被按下
	goto	Key9	; '8'沒被按下, 檢查是否按了'9'
	movlw	38h	; '8'被按下

	goto	WaitRelease	; 等待按鍵放開
Key9:	btfsc	PED, 1	; 檢查'9'是否被按下
	goto	Key10	; '9'沒被按下, 檢查是否按了'A'
	movlw	39h	; '9'被按下
	goto	WaitRelease	; 等待按鍵放開
Key10:	btfsc	PED, 2	; 檢查'A'是否被按下
	goto	Key11	; 'A'沒被按下, 'B'被按下
	movlw	41h	; 'A'被按下
	goto	WaitRelease	; 等待按鍵放開
Key11:	movlw	42h	; 'B'被按下
	goto	WaitRelease	; 等待按鍵放開
			; 等待使用者放開按鍵
WaitRelease:	movwf	KEY	; 記下按鍵的ASCII碼
WR_LOOP:	movlw	ffh	; 清除按鍵狀態
	movwf	PED	;
	call	Delay	; 時間延遲
	movlw	0fh	; 將鍵盤列位址(bits4~7)變成低電位
	movwf	PED	; 將鍵盤行位址(bits0~3)變成高電位
	movfw	PED	; 讀取鍵盤狀態, 若放開按鍵應是0fh
	xorlw	0fh	; 沒按鍵xor結果為0, ZF=1
	btfss	ZF	; 檢查ZF, ZF=1沒按鍵, ZF=0有按鍵
	goto	WR_LOOP	; 有按鍵, 繼續等待
	ret		; 沒按鍵, 回到呼叫程式
			; ***** LCM 相關副程式 *****
			; LCM初始設定副程式
LCMInitialize:	movlw	30h	; 功能設定: 設定DL=N=F=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	30h	; 功能設定: 設定DL=N=F=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	30h	; 功能設定: 設定DL=N=F=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	38h	; 功能設定: 設定DL=N=1, F=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	08h	; 顯示器控制: D=C=B=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	01h	; 清除顯示
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	06h	; 模式設定: I/D=1, S=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	movlw	0eh	; 顯示器控制: D=C=1, B=0
	call	LcmWriteIR	; 將命令碼寫到LCM
	ret		

```

; 將命令碼寫入LCM指令暫存器副程式
; 輸入參數：要傳送的資料放在working register內
LcmWriteIR:  bcf      RS_LCM    ; 選擇指令暫存器
              bcf      RW_LCM  ; 選擇進行寫入動作
              bsf      EN_LCM  ; 開啟LCM讀寫功能
              movwf   PBD      ; 送出命令碼
              call    Delay    ; 等候一段時間，讓資料完成寫入動作
              bcf      EN_LCM  ; 關閉LCM讀寫功能
              ret          ; 返回呼叫函式

; 將資料寫入LCM記憶體副程式
; 輸入參數：要傳送的資料放在working register內
LcmWriteDR:  bsf      RS_LCM    ; 選擇記憶體
              bcf      RW_LCM  ; 選擇進行寫入動作
              bsf      EN_LCM  ; 開啟LCM讀寫功能
              movwf   PBD      ; 送出資料
              call    Delay    ; 等候一段時間，讓資料完成寫入動作
              bcf      EN_LCM  ; 關閉LCM讀寫功能
              ret          ; 返回呼叫函式

; 清除LCD顯示內容副程式
ClearScreen: movlw    01h      ; 載入清除顯示內容命令
              call    LcmWriteIR ; 將命令碼寫到LCM
              ret          ; 返回呼叫函式

; 將游標位置移到第一列開頭副程式
MoveToRow1: movlw    80h      ; 設定DD RAM位址為00H
              call    LcmWriteIR ; 將命令碼寫到LCM
              ret          ; 返回呼叫函式

; 將游標位置移到第二列開頭副程式
MoveToRow2: movlw    c0h      ; 設定DD RAM位址為40H
              call    LcmWriteIR ; 將命令碼寫到LCM
              ret          ; 返回呼叫函式

; 將字串顯示到LCD副程式
; 輸入參數：strID放字串編號
ShowStr:     movlw    00h      ; 由字串第0個字元開始處理
              movwf   strIndex ;
SS_Next:     movfw    strIndex ; 取出目前的位置
              call    GetStrChar ; 由strID指定的字串取出第strIndex字元
              movwf   data     ; 將字元的ASCII Code寫入data變數
              incfsz  data, 0   ; 若字元的ASCII Code為ffh結束函式
              goto    SS_ShowCh ; 顯示字元
              goto    SS_End   ; 結束函式
SS_ShowCh:   movfw    data     ; 載入字元ASCII Code

```

```

        call      LcmWriteDR ; 將字元的ASCII Code寫入LCM記憶體
        incf     strIndex   ; 將字元位置+1
        goto    SS_Next    ; 處理下個字元
SS_End:  ret          ; 返回呼叫函式

; 由指定字串取出字元副程式
; 輸入參數：strID放字串編號，strIndex放要取出之字元的位置
; 輸出參數：working register放字元的ASCII Code
GetStrChar: bcf      CF          ; 清除進位旗號
            rlf      strID, 0    ; 左移1位, 結果放在working register
            addwf   PC, 1       ; 往下跳至第strID×2個指令
            movfw   strIndex     ; 取出字元位置
            goto    String1     ; 取出字串1內容
            movfw   strIndex     ; 取出字元位置
            goto    String2     ; 取出字串2內容
            movfw   strIndex     ; 取出字元位置
            goto    String3     ; 取出字串3內容
            movfw   strIndex     ; 取出字元位置
            goto    String4     ; 取出字串4內容

String1:   addwf   PC, 1       ; 跳到第i行，i=working register的內容
            retlw   45h        ; E
            retlw   6eh        ; n
            retlw   74h        ; t
            retlw   65h        ; e
            retlw   72h        ; r
            retlw   20h        ;
            retlw   50h        ; P
            retlw   61h        ; a
            retlw   73h        ; s
            retlw   73h        ; s
            retlw   77h        ; w
            retlw   6fh        ; o
            retlw   72h        ; r
            retlw   64h        ; d
            retlw   3ah        ; :
            retlw   ffh        ; 字串結尾

String2:   addwf   PC, 1       ; 跳到第i行，i=working register的內容
            retlw   57h        ; W
            retlw   65h        ; e
            retlw   7ch        ; l
            retlw   63h        ; c
            retlw   6fh        ; o
            retlw   6dh        ; m
            retlw   65h        ; e
            retlw   20h        ;
            retlw   74h        ; t
            retlw   6fh        ; o
            retlw   20h        ;
    
```

```

retlw    74h    ;T
retlw    65h    ;e
retlw    6eh    ;n
retlw    78h    ;x
retlw    21h    ;!
retlw    21h    ;!
retlw    ffh    ;字串結尾

String3:  addwf   PC, 1    ;跳到第i行，i=working register的內容
retlw    50h    ;P
retlw    7ch    ;l
retlw    65h    ;e
retlw    61h    ;a
retlw    73h    ;s
retlw    65h    ;e
retlw    20h    ;
retlw    53h    ;S
retlw    65h    ;e
retlw    74h    ;t
retlw    20h    ;
retlw    50h    ;P
retlw    61h    ;a
retlw    73h    ;s
retlw    73h    ;s
retlw    77h    ;w
retlw    6fh    ;o
retlw    72h    ;r
retlw    64h    ;d
retlw    3ah    ;:
retlw    ffh    ;字串結尾

String4:  addwf   PC, 1    ;跳到第i行，i=working register的內容
retlw    45h    ;E
retlw    72h    ;r
retlw    72h    ;r
retlw    6fh    ;o
retlw    72h    ;r
retlw    21h    ;!
retlw    ffh    ;字串結尾

; ***** SPI相關副程式 *****
; 由快閃記憶體讀取密碼副程式
; 讀到的密碼放在PW1, PW2, PW3, PW4內
ReadPW:  bcf     PFD, 2    ;設定/s=0，開啟讀寫ICE25P05功能
         movlw  OP_READ   ;載入READ指令碼
         call   WriteSPI  ;將RDSR指令傳送出去
         movlw  00h       ;設定位址第一個位元組為00
    
```

```

call      WriteSPI    ; 傳送位址的第一個位元組
movlw    00h          ; 設定位址第二個位元組為00
call     WriteSPI    ; 傳送位址的第二個位元組
movlw    00h          ; 設定位址第三個位元組為00
call     WriteSPI    ; 傳送位址的第三個位元組
movlw    ffh          ; 載入ffh, 要收SPI資料, 必須寫入SPID
call     WriteSPI    ; 讀入第一個數字
movwf    PW1          ; 將數字放入PW1
movlw    ffh          ; 載入ffh, 要收SPI資料, 必須寫入SPID
call     WriteSPI    ; 讀入第二個數字
movwf    PW2          ; 將數字放入PW2
movlw    ffh          ; 載入ffh, 要收SPI資料, 必須寫入SPID
call     WriteSPI    ; 讀入第三個數字
movwf    PW3          ; 將數字放入PW3
movlw    ffh          ; 載入ffh, 要收SPI資料, 必須寫入SPID
call     WriteSPI    ; 讀入第四個數字
movwf    PW4          ; 將數字放入PW4
bsf      PFD, 2       ; 結束資料讀取動作
ret

```

; 將密碼寫入快閃記憶體副程式

; 要寫入快閃記憶體的密碼放在PW1 ~ PW4

SavePW: ; 寫出WREN 指令, 允許寫入ICE25P05

```

bcf      PFD, 2       ; 設定/s=0, 開啟讀寫ICE25P05功能
movlw    OP_WREN     ; 載入WREN 指令碼
call     WriteSPI    ; 將WREN指令傳送出去
bsf      PFD, 2       ; 設定/s=1, 關閉讀寫ICE25P05功能

```

; 寫出WRSR 指令, 設定ICE25P05狀態

```

bcf      PFD, 2       ; 設定/s=0, 開啟讀寫ICE25P05功能
movlw    OP_WRSR     ; 載入WRSR指令碼
call     WriteSPI    ; 將WRSR指令傳送出去
movlw    00h         ; 設定狀態暫存器SRWD=BP1=BP0=0
call     WriteSPI    ; 將狀態暫存器內容傳送出去
bsf      PFD, 2       ; 設定/s=1, 關閉讀寫ICE25P05功能

```

; 寫出WREN 指令, 允許寫入ICE25P05

```

bcf      PFD, 2       ; 設定/s=0, 開啟讀寫ICE25P05功能
movlw    OP_WREN     ; 載入WREN 指令碼
call     WriteSPI    ; 將WREN指令傳送出去
bsf      PFD, 2       ; 設定/s=1, 關閉讀寫ICE25P05功能

```

; 寫出SE 指令, 清除資料區

```

bcf      PFD, 2       ; 設定/s=0, 開啟讀寫ICE25P05功能
movlw    OP_SE       ; 載入SE指令碼
call     WriteSPI    ; 將SE指令傳送出去
movlw    00h         ; 設定位址第一個位元組為00

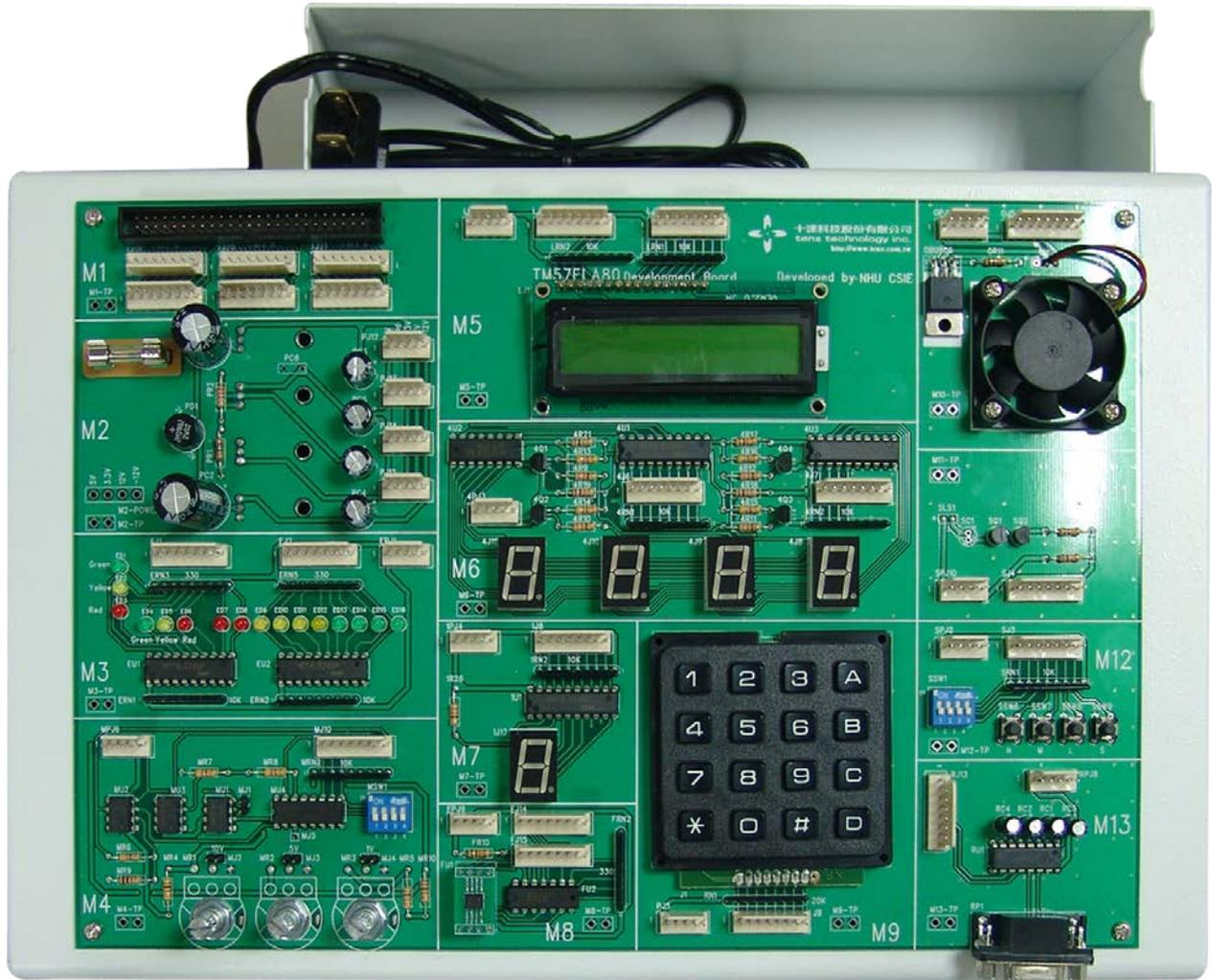
```

	call	WriteSPI	; 傳送位址的第一個位元組
	movlw	00h	; 設定位址第二個位元組為00
	call	WriteSPI	; 傳送位址的第二個位元組
	movlw	00h	; 設定位址第三個位元組為00
	call	WriteSPI	; 傳送位址的第三個位元組
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出RDSR 指令, 讀取狀態暫存器內容, 等待flash完成動作		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_RDSR	; 載入RDSR指令碼
	call	WriteSPI	; 將RDSR指令傳送出去
tsWIP:	movlw	ffh	; 載入ffh, 要收SPI資料, 必須寫入SPID
	call	WriteSPI	; 接收狀態暫存器內容
	movwf	data	; 將狀態暫存器內容放入data
	btfs	data, 0	; 測試WIP是否為0(WIP=0表示不忙碌)
	goto	tsWIP	; 忙碌中, 繼續檢查, 直到不忙碌
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出WREN 指令, 允許寫入ICE25P05		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_WREN	; 載入WREN 指令碼
	call	WriteSPI	; 將WREN指令傳送出去
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	; 寫出PP指令, 要求寫入資料		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_PP	; 載入PP指令碼
	call	WriteSPI	; 將PP指令傳送出去
	movlw	00h	; 設定位址第一個位元組為00
	call	WriteSPI	; 傳送位址的第一個位元組
	movlw	00h	; 設定位址第二個位元組為00
	call	WriteSPI	; 傳送位址的第二個位元組
	movlw	00h	; 設定位址第三個位元組為00
	call	WriteSPI	; 傳送位址的第三個位元組
	movfw	PW1	; 載入第一個數字
	call	WriteSPI	; 將第一個數字寫入Flash
	movfw	PW2	; 載入第二個數字
	call	WriteSPI	; 將第二個數字寫入Flash
	movfw	PW3	; 載入第三個數字
	call	WriteSPI	; 將第三個數字寫入Flash
	movfw	PW4	; 載入第四個數字
	call	WriteSPI	; 將第四個數字寫入Flash
	bsf	PFD, 2	; 結束資料寫入動作
	; 寫出RDSR 指令, 讀取狀態暫存器內容, 等待flash完成動作		
	bcf	PFD, 2	; 設定/s=0, 開啟讀寫ICE25P05功能
	movlw	OP_RDSR	; 載入RDSR指令碼

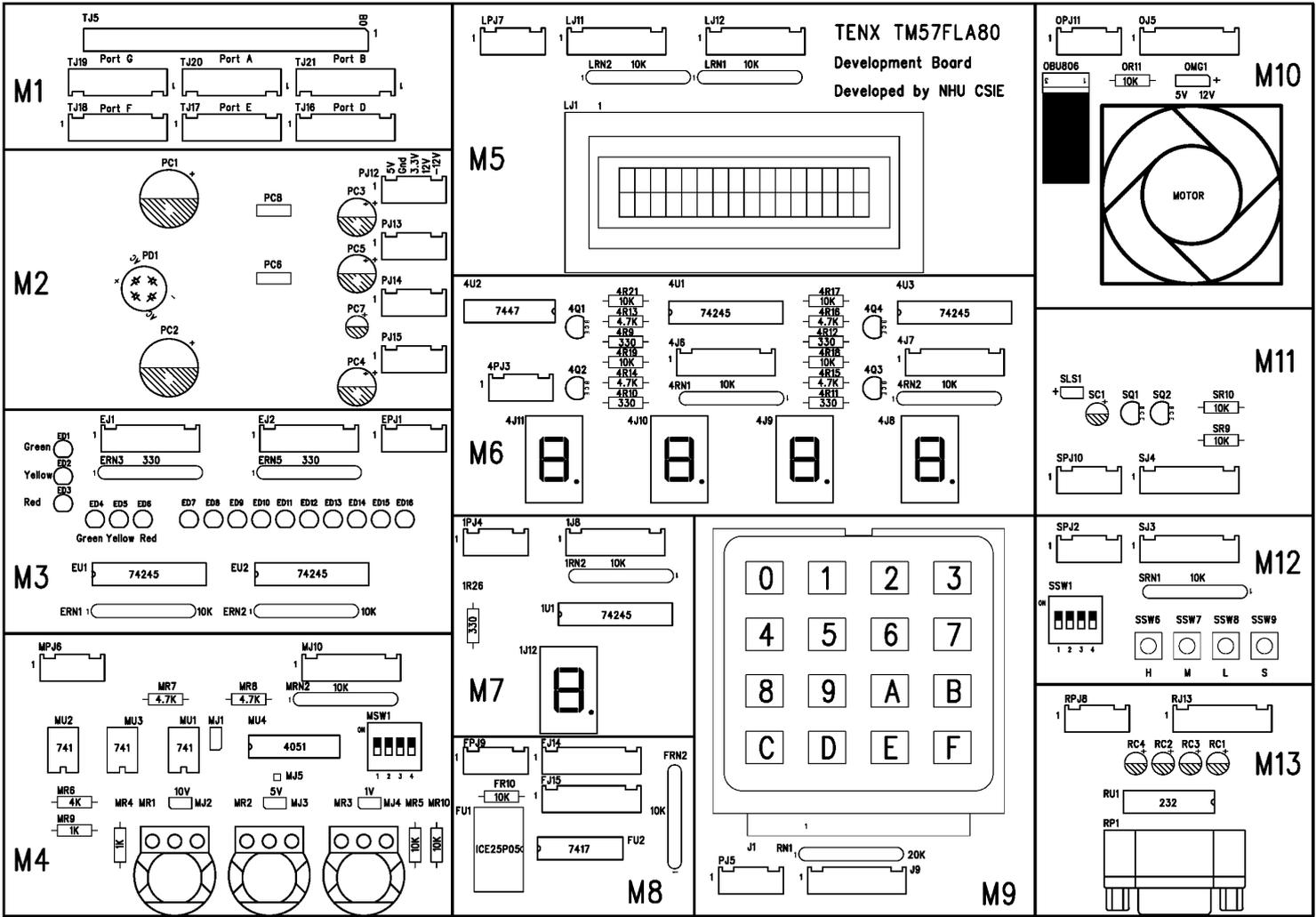
tsWIP2:	call	WriteSPI	; 將RDSR指令傳送出去
	movlw	ffh	; 載入ffh, 要收SPI資料, 必須寫入SPID
	call	WriteSPI	; 接收狀態暫存器內容
	movwf	data	; 將狀態暫存器內容放入data
	btfsc	data, 0	; 測試WIP是否為0(WIP=0表示不忙碌)
	goto	tsWIP2	; 忙碌中, 繼續檢查, 直到不忙碌
	bsf	PFD, 2	; 設定/s=1, 關閉讀寫ICE25P05功能
	ret		
; SPI傳送接收資料副程式			
; 輸入參數: 要傳送的資料放在working register內			
; 輸出參數: 由SPI讀到的資料放在working register內			
WriteSPI:	btfss	TXBEPY	; 檢查SPI目前是否可傳送資料
	goto	WriteSPI	; 不能傳, 重新檢查, 直到可傳
	movwf	SPID	; 將要傳送的資料寫入SPID暫存器
chkRcv:	btfss	RCVBF	; 檢查是否收到資料
	goto	chkRcv	; 沒收到資料, 等待資料進來
	movfw	SPID	; 取出收到的資料
	ret		; 返回呼叫程式
; ***** 其它副程式 *****			
; 延遲0.015秒副程式			
Delay:	movlw	30	; 設定外層迴圈執行30次
	Movwf	R1	;
	; 外層迴圈		
delay_L1:	movlw	200	; 設定內層迴圈執行200次
	movwf	R2	;
	; 內層迴圈: 迴圈跑一次約消耗5個指令週期		
delay_L2:	nop		; 消耗1個指令週期
	nop		; 消耗1個指令週期
	decfsz	R2, 1	; 約消耗1個指令週期
	goto	delay_L2	; 消耗2個指令週期
	decfsz	R1, 1	; 將R1減1, 若R1=0離開迴圈
	goto	delay_L1	;
	ret		

## 附錄1 TM57教學實驗板使用說明

TM57教學實驗板照片



附錄 1-1 TM57 教學實驗板簡介



圖附錄 1-1 模組板配置圖

表附錄 1-1 模組清單

模組編號	模組名稱
M1	介面轉換模組
M2	電源模組
M3	LED 模組
M4	數位類比轉換模組
M5	文字型 LCD 顯示模組
M6	四顆七段顯示器模組
M7	一顆七段顯示器模組
M8	FLASHROM 模組
M9	4X4 鍵盤模組
M10	馬達模組
M11	喇叭模組
M12	指撥模組
M13	RS232 串列傳輸模組

表附錄 1-2 實習與模組對照表

	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
紅綠燈	V										
指撥開關與七段顯示器				V						V	
計時器					V						
數位電子鐘				V							
4X4 鍵盤				V			V				
數位電子琴							V		V		
數位電錶類比轉數位		V		V							
脈波調變控制之電風扇								V		V	
文字型 LCD 顯示器			V								
UART 與 RS232 傳輸			V				V				V
SPI 串列週邊介面				V		V	V				
密碼鎖			V			V	V				

## 附錄 1-2 模組板使用方式

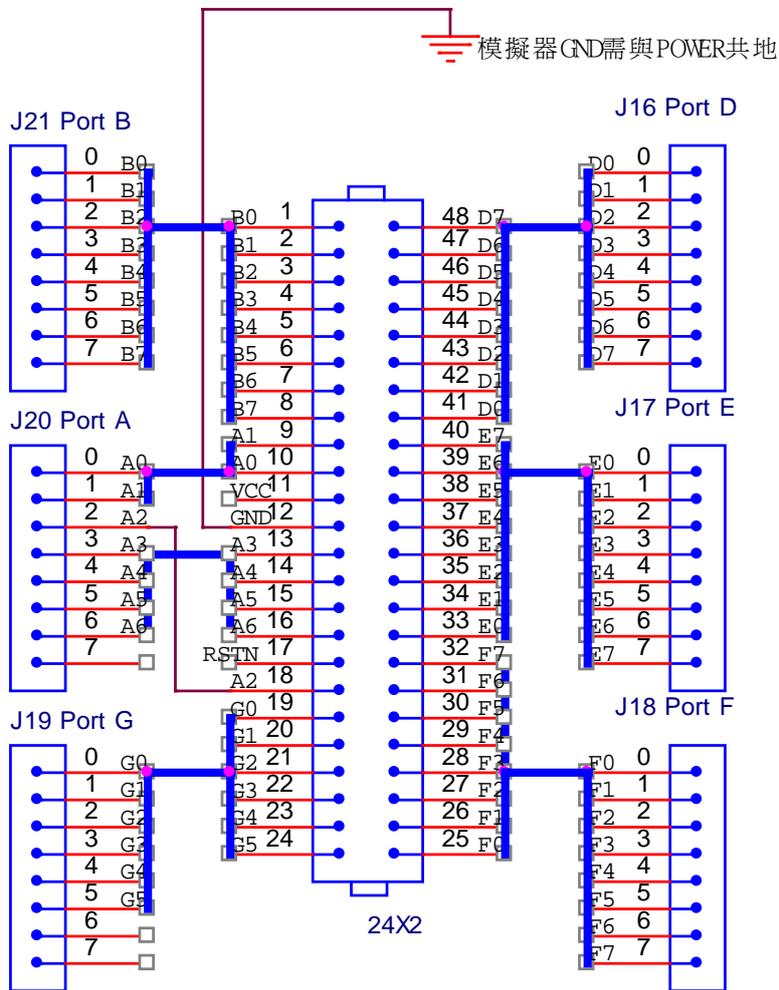
使用模組板時須先將要使用之模組的電源接上，再將各實習所使用到之 I/O Port 接到適當模組的連接埠上。例如：進行紅綠燈實習時，需將 M1 的 Port E 接到模組板 M3 的 J1 連接埠。表附錄 1-3 列出各實習使用的 I/O Port，以及對應的模組板連接埠接線方式。

表附錄 1-3 各實習與模組板接線方式對照表

實習名稱	使用 Port	模組板	連接埠
紅綠燈	Port E	M3	J1
指撥開關	Port D	M12	J3
	Port E	M6	J6
計時器	Port E	M7	J8
電子鐘	Port E	M6	J6
4X4 鍵盤	Port D	M9	J9
	Port E	M6	J6
電子琴	Port B	M11	J4
	Port D	M9	J9
數位電錶	Port B	M4	J10
	Port D	M6	J7
	Port E	M6	J6
電風扇	Port B	M10	J5
	Port D	M12	J3
LCD 顯示	Port B	M5	J12
	Port D	M5	J11
RS232 傳輸	Port B	M5	J12
	Port D	M5	J11
	Port E	M9	J9
	Port G	M13	J13
SPI 實習	Port B	M6	J6
	Port E	M9	J9
	Port F	M8	J14
	Port G	M8	J15
密碼鎖	Port B	M5	J12
	Port D	M5	J11
	Port E	M9	J9
	Port F	M8	J14
	Port G	M8	J15

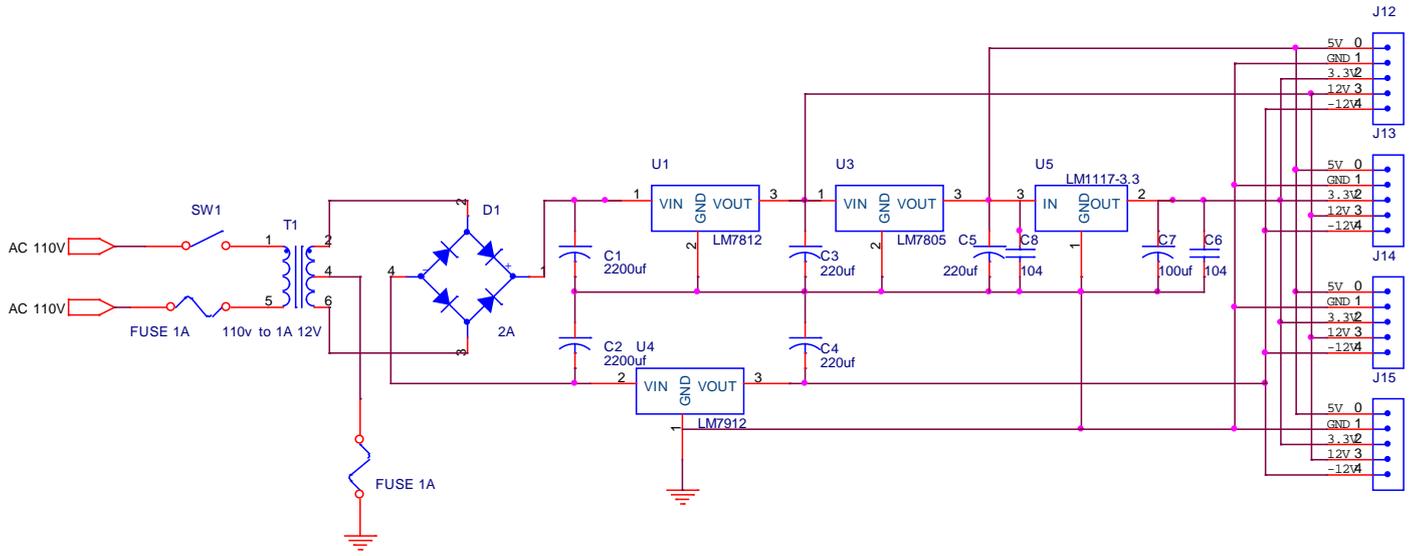
附錄 1-3 模組版電路圖

M1 - 介面轉換模組



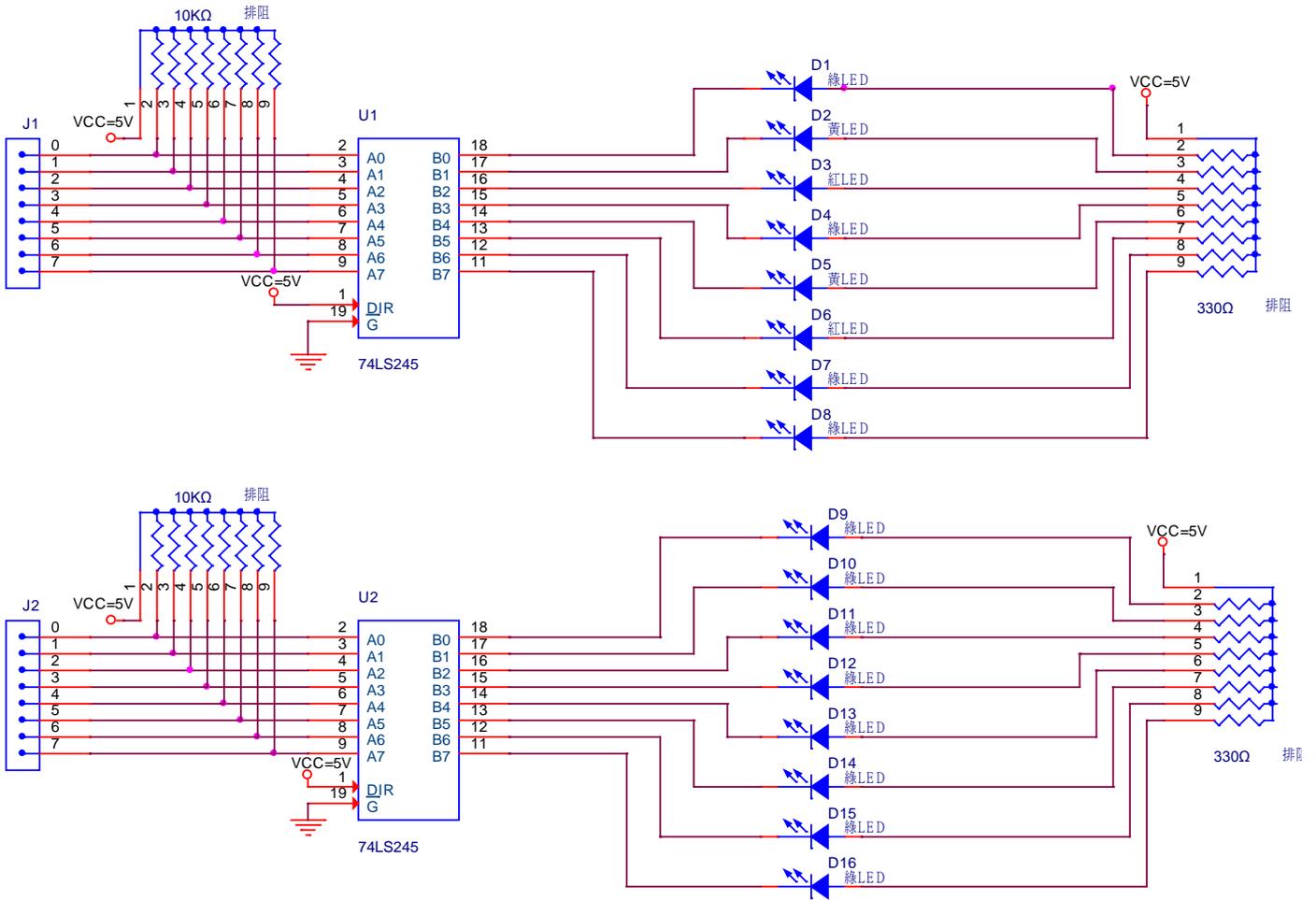
圖附錄 1-2 介面轉換模組

M2 - 電源模組



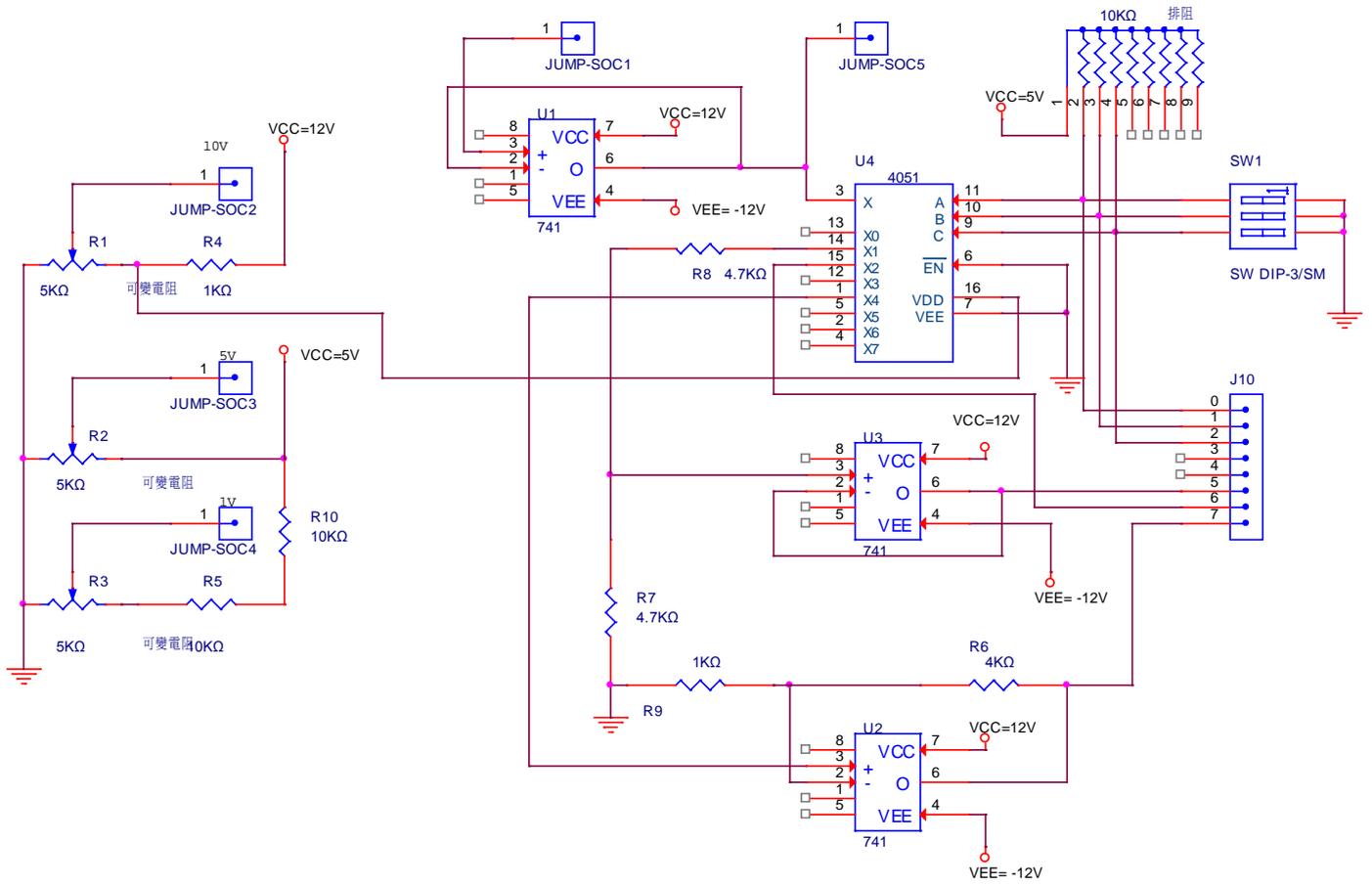
圖附錄 1-3 電源模組

M3 - LED 模組



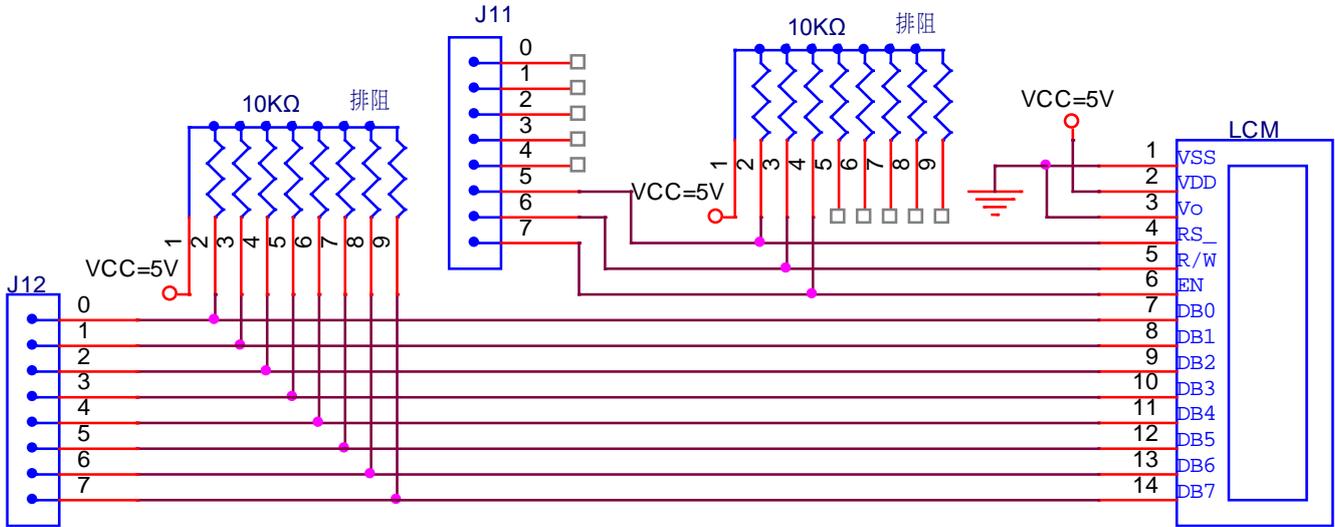
圖附錄 1-4 LED 模組

M4 - 數位類比轉換模組



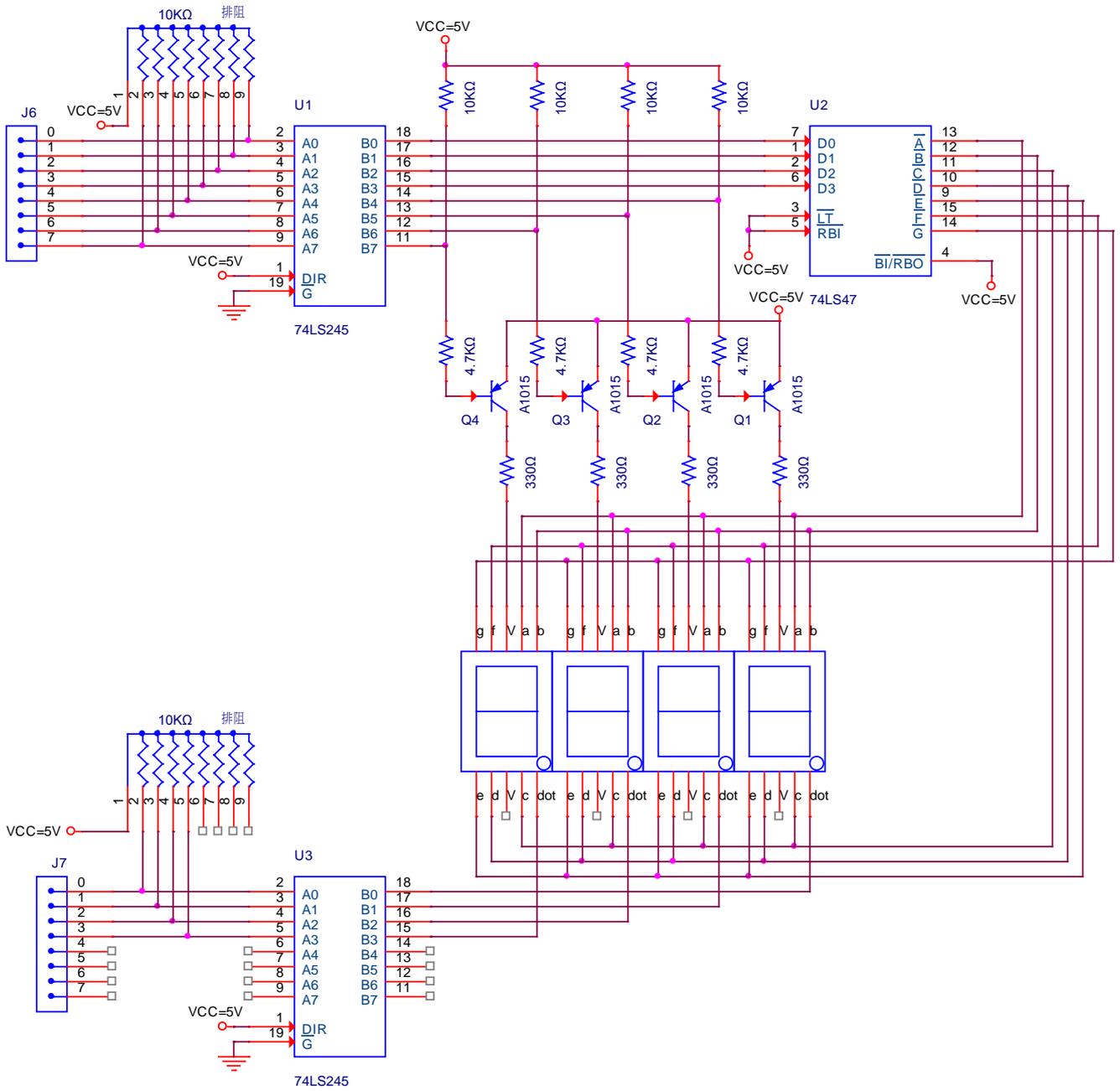
圖附錄 1-5 數位類比轉換模組

M5 - 文字型 LCD 顯示模組



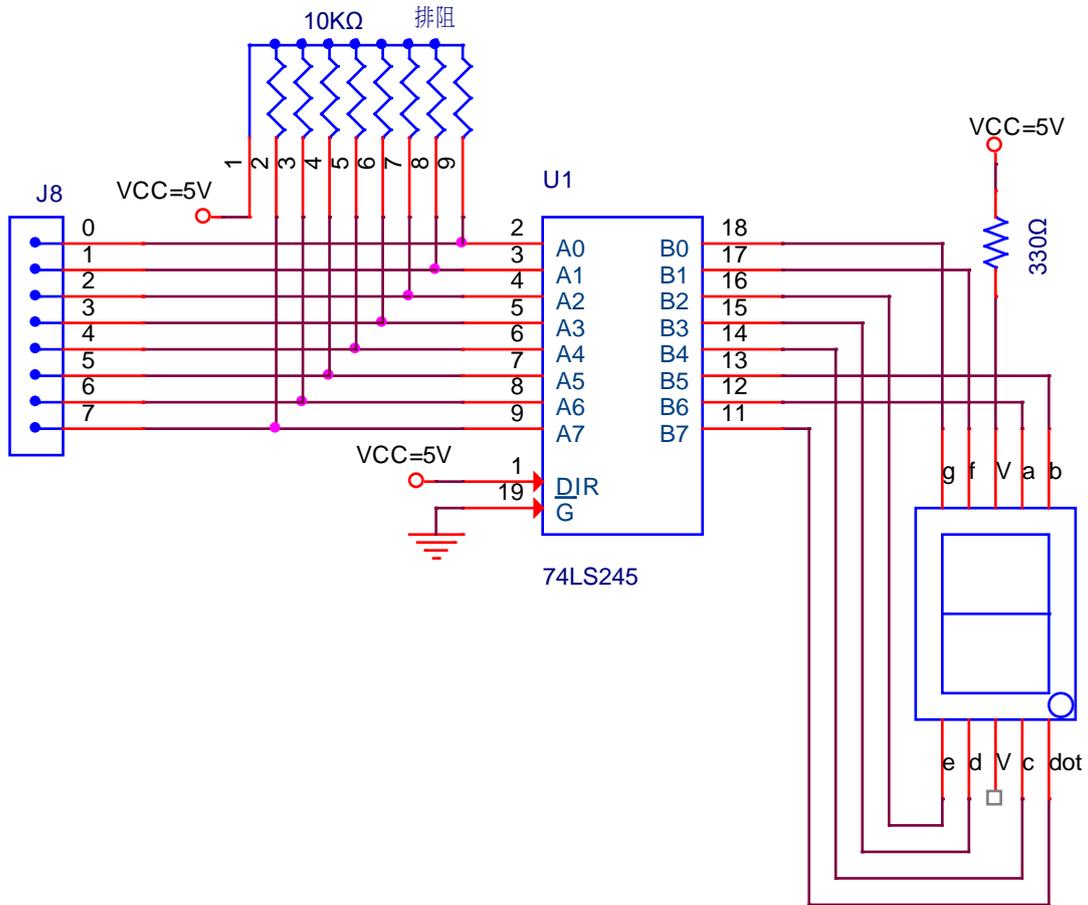
圖附錄 1-6 文字型 LCD 模組

M6 - 四顆七段顯示器模組



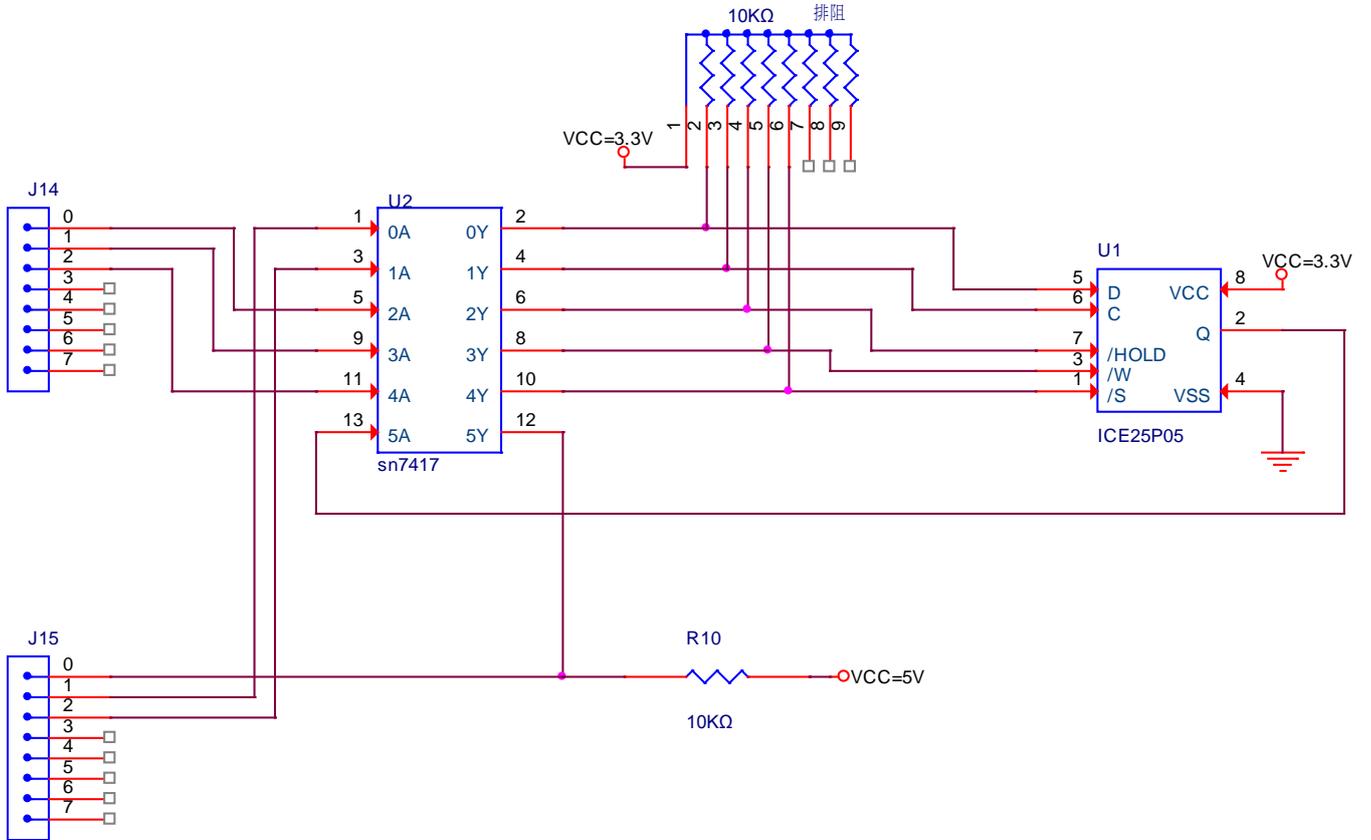
圖附錄 1-7 四顆七段顯示模組

M7 - 一顆七段顯示器模組



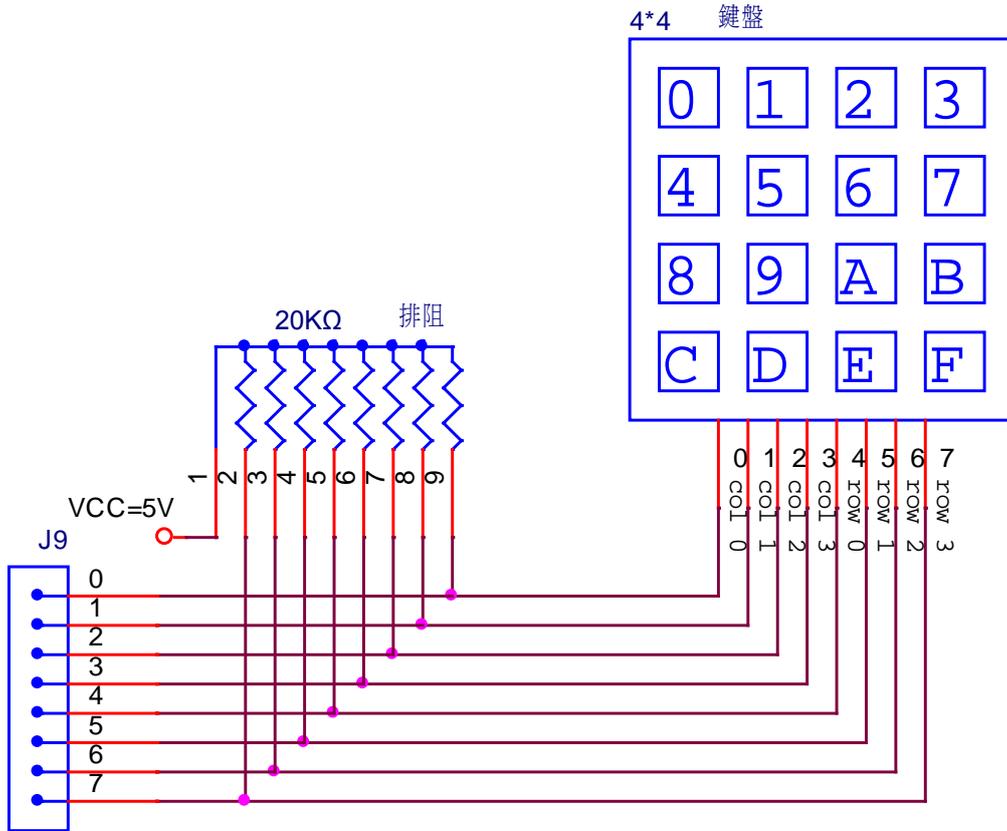
圖附錄 1-8 一顆七段顯示器模組

M8 - FLASHROM 模組



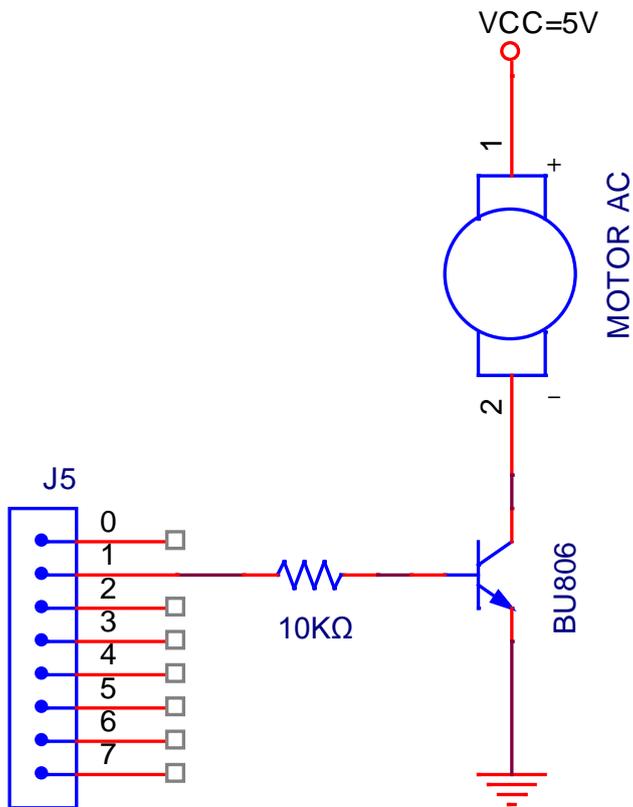
圖附錄 1-9 FLASHROM 模組

M9 - 4X4 鍵盤模組



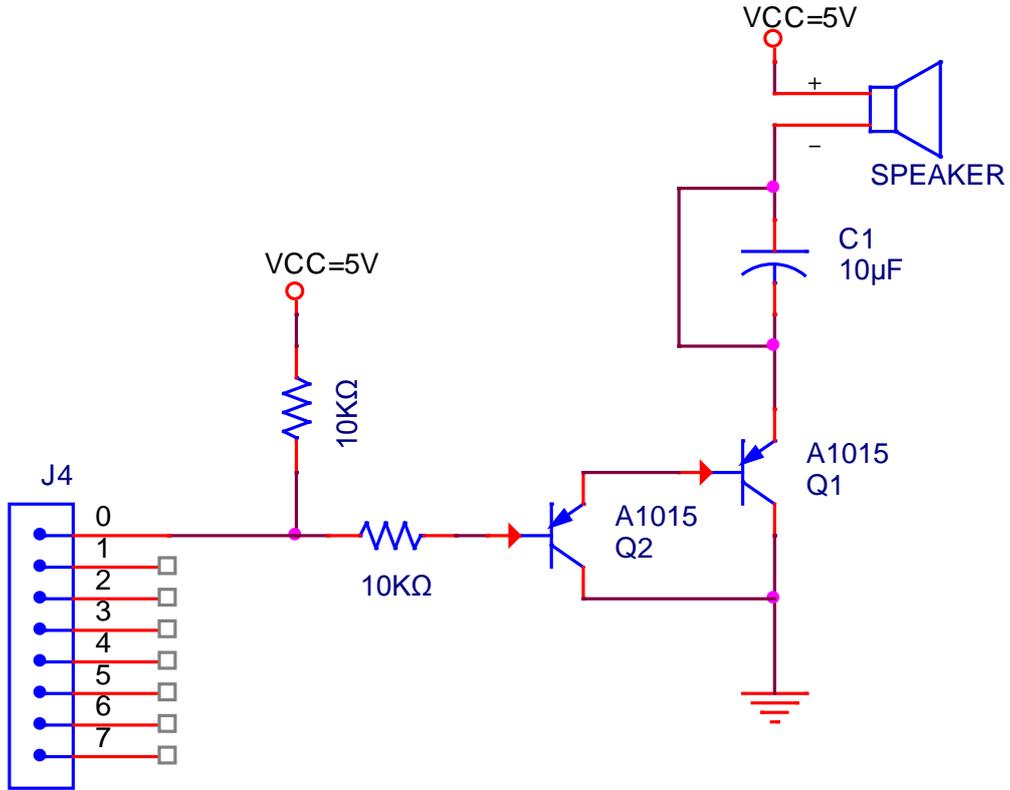
圖附錄 1-10 4X4 鍵盤模組

M10 - 馬達模組



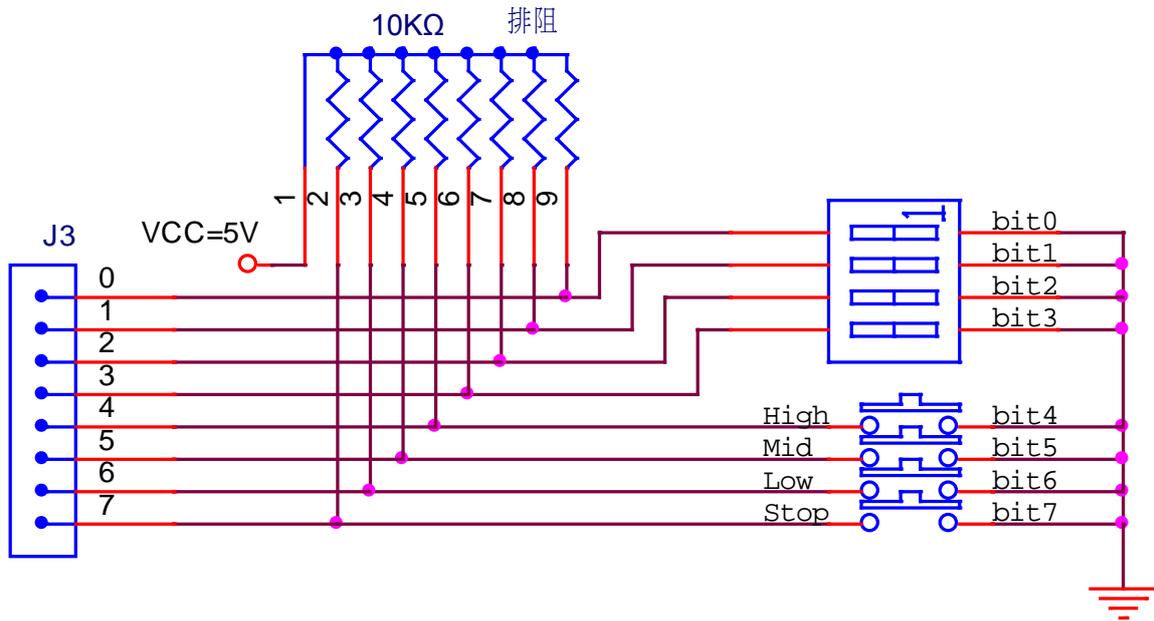
圖附錄 1-11 馬達模組

M11 - 喇叭模組



圖附錄 1-12 喇叭模組

M12 - 指撥模組



圖附錄 1-13 指撥模組



附錄2 配合模組板使用的程式修改方式

表附錄 2-1 紅綠燈實習

修改前	修改後
<pre> Start:  movlw 011110b         movwf PED         ;         延遲5秒         call    delay5          ;橫向黃燈閃爍三次         movlw  03h         movwf  fCnt  L1:     movlw 011101b         movwf  PED         call   delay          movlw  011111b         movwf  PED         call   delay          decfsz fCnt, 1         goto  L1         ;縱向綠燈，橫向紅燈         movlw  110011b         movwf  PED         ;延遲5秒         call   delay5         ;縱向黃燈閃爍三次         movlw  03h         movwf  fCnt  L2:     movlw  101011b         movwf  PED         call   delay          movlw 111011b         movwf  PED         call   delay          decfsz fCnt, 1         goto  L2         goto  Start                     </pre>	<pre> Start:  movlw <b>11</b>011110b         movwf PED         ;         延遲5秒         call    delay5          ;橫向黃燈閃爍三次         movlw  03h         movwf  fCnt  L1:     movlw  <b>11</b>011101b         movwf  PED         call   delay          movlw  <b>11</b>011111b         movwf  PED         call   delay          decfsz fCnt, 1         goto  L1         ;縱向綠燈，橫向紅燈         movlw  <b>11</b>110011b         movwf  PED         ;延遲5秒         call   delay5         ;縱向黃燈閃爍三次         movlw  03h         movwf  fCnt  L2:     movlw  <b>11</b>101011b         movwf  PED         call   delay          movlw <b>11</b>111011b         movwf  PED         call   delay          decfsz fCnt, 1         goto  L2         goto  Start                     </pre>

說明：模組板比實習電路多了2顆LED，故將Port D的最高兩位元設定為1，以關閉LED燈。

表附錄 2-2 指撥開關與七段顯示實習

修改前	修改後
<pre> Start:  btfsc  PDD, 4      goto         Start          ; 去除彈跳         call    delay         btfsc  PDD, 4      goto         Start         ; 將指撥開關設定的         BCD 碼傳送給 7447 IC         movfw  PDD         movwf  PED                     </pre>	<pre> Start:  btfsc  PDD, 4      goto Start         Start          ; 去除彈跳         call    delay         btfsc  PDD, 4         goto  Start         ; 將指撥開關設定的         BCD 碼傳送給 7447 IC         movfw  PDD         iorlw  11100000b         andlw  11101111b         movwf  PED                     </pre>
<p>說明：在指撥開關實習電路中，僅使用一顆七段顯示器，在模組板上使用到的是四顆七段顯示器，故在程式上我們必須將 Port D 的腳位 5~7 位元設定為 1，僅讓 Port D 的第 4 位元為 0，做單一顆七段顯示器的顯示工作。</p>	

表附錄 2-3 計時器實習

修改前	修改後
<pre> Table7S:  addwf  PC, 1           retlw  0000001b           retlw  1001111b           retlw  0010010b           retlw  0000110b           retlw  1001100b           retlw  0100100b           retlw  1100000b           retlw  0001111b           retlw  0000000b           retlw  0000100b                     </pre>	<pre> Table7S:  addwf  PC, 1           retlw  10000001b           retlw  11001111b           retlw  10010010b           retlw  10000110b           retlw  11001100b           retlw  10100100b           retlw  11100000b           retlw  10001111b           retlw  10000000b           retlw  10000100b                     </pre>
<p>說明：由於模組板比實習電路多控制小數點，而實習並未使用到，故給予其值 1 使之不發亮</p>	

表附錄 2-4 數位電表類比轉數位實習

修改前			修改後		
Mode1:	movlw	001b	Mode1:	<b>movlw</b>	<b>10111111b</b>
	movwf	ADCSEL		<b>movwf</b>	<b>SHOWNUM3</b>
	movlw	11111101b	movlw	001b	
	movwr	ADPIN	movwf	ADCSEL	
	bsf	ADCSTART		movlw	11111101b
	call	WaitData		movwr	ADPIN
	movlw	01h		bsf	ADCSTART
	movwf	RRTimes		call	WaitData
	call	ReadData		movlw	01h
	bcf	PDD, 7		movwf	RRTimes
goto	SelMode			call	ReadData
				<b>movlw</b>	<b>11111101b</b>
				<b>movwf</b>	<b>PDD</b>
				<b>movfw</b>	<b>SHOWNUM3</b>
				<b>movwf</b>	<b>PED</b>
				<b>call</b>	<b>Delay</b>
				<b>goto</b>	<b>SelMode</b>
Mode2:	movlw	010b	Mode2:	movlw	010b
	movwf	ADCSEL		movwf	ADCSEL
	movlw	11111011b		movlw	11111011b
	movwr	ADPIN		movwr	ADPIN
	bsf	ADCSTART		bsf	ADCSTART
	call	WaitData		call	WaitData
	movlw	02h		movlw	02h
	movwf	RRTimes		movwf	RRTimes
	call	ReadData		call	ReadData
	bcf	PDD, 7		<b>movlw</b>	<b>11111101b</b>
goto	SelMode			<b>movwf</b>	<b>PDD</b>
				<b>goto</b>	<b>SelMode</b>
Mode3:	movlw	011b	Mode3:	<b>movlw</b>	<b>10110000b</b>
	movwf	ADCSEL		<b>movwf</b>	<b>SHOWNUM3</b>
	movlw	11110111b		movlw	011b
	movwr	ADPIN		movwf	ADCSEL
	bsf	ADCSTART		movlw	11110111b
	call	WaitData		movwr	ADPIN
	movlw	01h		bsf	ADCSTART
	movwf	RRTimes		call	WaitData
	call	ReadData		movlw	01h
				movwf	RRTimes

<pre> bcf          PDD, 6 goto        SelMode         </pre>	<pre> call    ReadData movlw  11111011b movwf  PDD movwf  SHOWNUM3 movwf  PED call   Delay goto   SelMode         </pre>
<pre> rerrf: bcf          CF         rrf          digitalData         decfsz RRTimes      goto         rerrf ; 將結果轉成兩位數 BCD movfw   digitalData call    BinToBCD ; 顯示結果 movwf  PED ret         </pre>	<pre> rerrf: bcf          CF         rrf          digitalData         decfsz RRTimes         goto        rerrf ; 將結果轉成兩位數 BCD movfw   digitalData call    BinToBCD ; 顯示結果 movwf  SHOWNUM movwf  SHOWNUM1 movlw  f0h iorwf  SHOWNUM1 bcf    SHOWNUM1,4 movfw  SHOWNUM1 movwf  PED call   Delay swapf  SHOWNUM movfw  SHOWNUM movwf  SHOWNUM2 movlw  f0h iorwf  SHOWNUM2 bcf    SHOWNUM2,5 movfw  SHOWNUM2 movwf  PED call   Delay ret         </pre>
<pre> movlw  39h movwf  PDD retlw  00h ; 延遲 0.0075 秒副程式 Delay: movlw  30         </pre>	<pre> movlw  10110001b movwf  SHOWNUM3 retlw  00h ; 延遲 0.0075 秒副程式 Delay: movlw  30         </pre>
<p>說明：在實習電路中原使用三顆七段顯示器，為共用模組電路，故將顯示部分改為顯示於四顆七段顯示器上，其控制顯示取代實習電路上直接給予數值顯示之方式改為四顆七段顯示器掃描方式顯示。</p>	

表附錄 2-5 SPI 串列周邊介面實習

修改前	修改後
<pre>ShowNums:  swapf  NUML1               movfw  NUML1               iorwf  NUML2,0               movwf  PBD                swapf  NUML3               movfw  NUML3               iorwf  NUML4,0                movwf  PDD               ret</pre>	<pre>ShowNums:  bcf     NUML1,7               movfw  NUML1               movwf  PBD               call   Delay               bcf     NUML2,6                movfw  NUML2               movwf  PBD               call   Delay               bcf     NUML3,5               movfw  NUML3               movwf  PBD               call   Delay               bcf     NUML4,4                movfw  NUML4               movwf  PBD               call   Delay               ret</pre>
<p>說明：因將 SPI 的內容改由模組板的四顆七段顯示器顯示，故其顯示方式改為使用四顆七段顯示器掃描方式顯示，取代原實習電路上直接給予數值之顯示方式。</p>	