# 高精度觸控廚房秤_應用說明書

# HY16F188

# Touch Precision Kitchen Scales

# 目　錄

注意：

1、 本說明書中的內容，隨著產品的改進，有可能不經過預告而更改。請客戶及時到本公司網站下載更新 http://www.hycontek.com

2、 本規格書中的圖形、應用電路等，因第三方工業所有權引發的問題，本公司不承擔其責任。

3、 本產品在單獨應用的情況下，本公司保證它的性能、典型應用和功能符合說明書中的條件。當使用在客戶的產品或設備中，以上條件我們不作保證，建議客戶做充分的評估和測試。

4、 請注意輸入電壓、輸出電壓、負載電流的使用條件，使 IC 內的功耗不超過封裝的容許功耗。對於客戶在超出說明書中規定額定值使用產品，即使是瞬間的使用，由此所造成的損失，本公司不承擔任何責任。

5、 本產品雖內置防靜電保護電路，但請不要施加超過保護電路性能的過大靜電。

6、 本規格書中的產品，未經書面許可，不可使用在要求高可靠性的電路中。例如健康醫療器械、防災器械、車輛器械、車載器械及航空器械等對人體產生影響的器械或裝置，不得作為其部件使用。

7、 本公司一直致力於提高產品的品質和可靠度，但所有的半導體產品都有一定的失效概率，這些失效概率可能會導致一些人身事故、火災事故等。當設計產品時，請充分留意冗餘設計並採用安全指標，這樣可以避免事故的發生。

8、 本規格書中內容，未經本公司許可，嚴禁用於其他目的之轉載或複製。

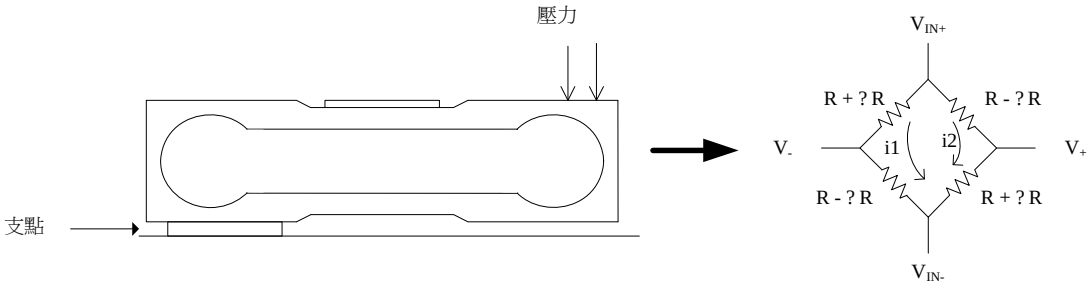
## 1.內容簡介

電子化秤重在生活中，已逐漸取代傳統彈簧、天平等量測工具，例如電子計價秤、電子體重秤等。設計電子秤產品主要的元件有：感測器、ADC和MCU單晶片。本文所設計的電子秤就是利用壓力感測器（Load Cell）將壓力物理量轉換為電壓訊號，再將電壓轉換為數字顯示出來。由於電壓為類比量，所以要用ADC將它轉換為數位信號。此時也需要MCU單晶片來控制電子秤主機板上的訊號處理與顯示功能。

紘康HY16F188控制晶片內建高精密SD 24 Bit ADC、可程式放大PGA和多段式穩壓輸出等功能，可以很大幅簡化PCB周邊線路。具有高解析度、高解析度、低溫漂的SD24 AD轉換器，可以精准完成由類比到數位的轉換。雖然輸出速率不是非常高，但用於像電子秤這種對於轉換速率要求不高的產品，是沒有問題的。

## 2.原理說明

Load Cell 的原理是在鋁制的棒上面貼上一片由橋式電阻所組成的應變儀，即惠斯頓電橋，如圖 2-1 所示。因為電橋上的 4 個電阻(阻值相同)，所以當有電壓施加在 VIN+與 VIN-兩端時 V+ = V-，即電橋達到了平衡。




此 ΔR 的變化量產生在訊號兩端的電壓變化為

$$V_+ - V_- = \left( \frac{(R + \Delta R)}{(R - \Delta R) + (R + \Delta R)} \times (V_{in+} - V_{in-}) \right) - \left( \frac{(R - \Delta R)}{(R - \Delta R) + (R + \Delta R)} \times (V_{in+} - V_{in-}) \right)$$

$$V_+ - V_- = \frac{\Delta R}{R} \times (V_{in+} - V_{in-})$$

解析度分為外部解析度和內部解析度，外部解析度為 Load Cell 滿量程的輸出電壓值與需要識別的最小重量引起的電壓值之比，最小重量可以定義為 1g、0.5g、0.1g 等。

內部解析度是衡量電子秤等級的一個重要指標。一般我們以目視法認定的內部解析度通常是指我們經軟體處理後 LCD 顯示只有 1 格滾動時，此時滿量程的格數就是內部解析度，其 1 格所代表的訊號約為 2~3 倍 RMS Noise。

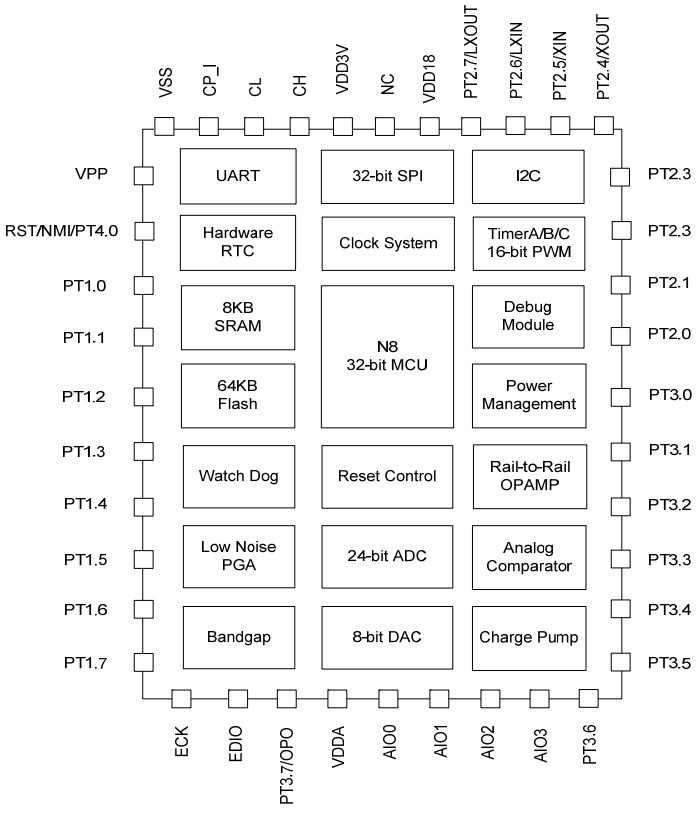
內外解析度之比越小，電子秤精度越高，但內外解析度之比是有限制的。比如 Load Cell 滿量程壓差為 3mV，要做到 3000 Count，內外比為 1：10 的電子秤，如果不經過信號放大，那最小要處理的信號為 3mV/(3000X10)=0.1μV。而 SD24 所能處理的最小信號值大約為 65nV，所以假如內外比再減小的話將產生使 ADC 不能識別的信號。如果使用 OPAMP 的話則會增加成本。所以內外解析度之比要穩定在一定範圍內。

晶片 ADC 性能能否達到規格要求，通常是以 RMS Noise 來推算外部是否穩定內部解析度比值。對於開發電子秤產品而言，使用 HY16F188 晶片其所能達到的最大內部解析度的瓶頸在於 Input RMS Noise 而不在於 ADC 的解析度。 HY16F188 的 ADC 待測信號在由 PGA、AD 倍率調整器的放大後（PGA=32，ADGN=4），經 OSR=32768 每秒輸出 10 筆 ADC 值的條件下，其 Input RMS Noise 約為 65nV，但由於其 Input Noise 主要由 Thermal Noise 組成，所以如果我們透過平均的軟體處理是可以再將 Input Noise 進一步降低。

如果我們使用 8 筆的軟體平均處理其 Input RMS Noise 約為 40nV，3 倍 RMS Noise 代表約 1 格的滾動，即為 120nV。在使用 2.4V Load Cell 驅動電壓，1mV/V 的 Load Cell，滿量程時壓差可達 2.4mV，所以在此情形下我們可以得到 20000 Counts 的內部解析度。

## 2.2 控制晶片

單片機簡介：HY16F 系列 32 位元高性能 Flash 單片機(HY16F188)



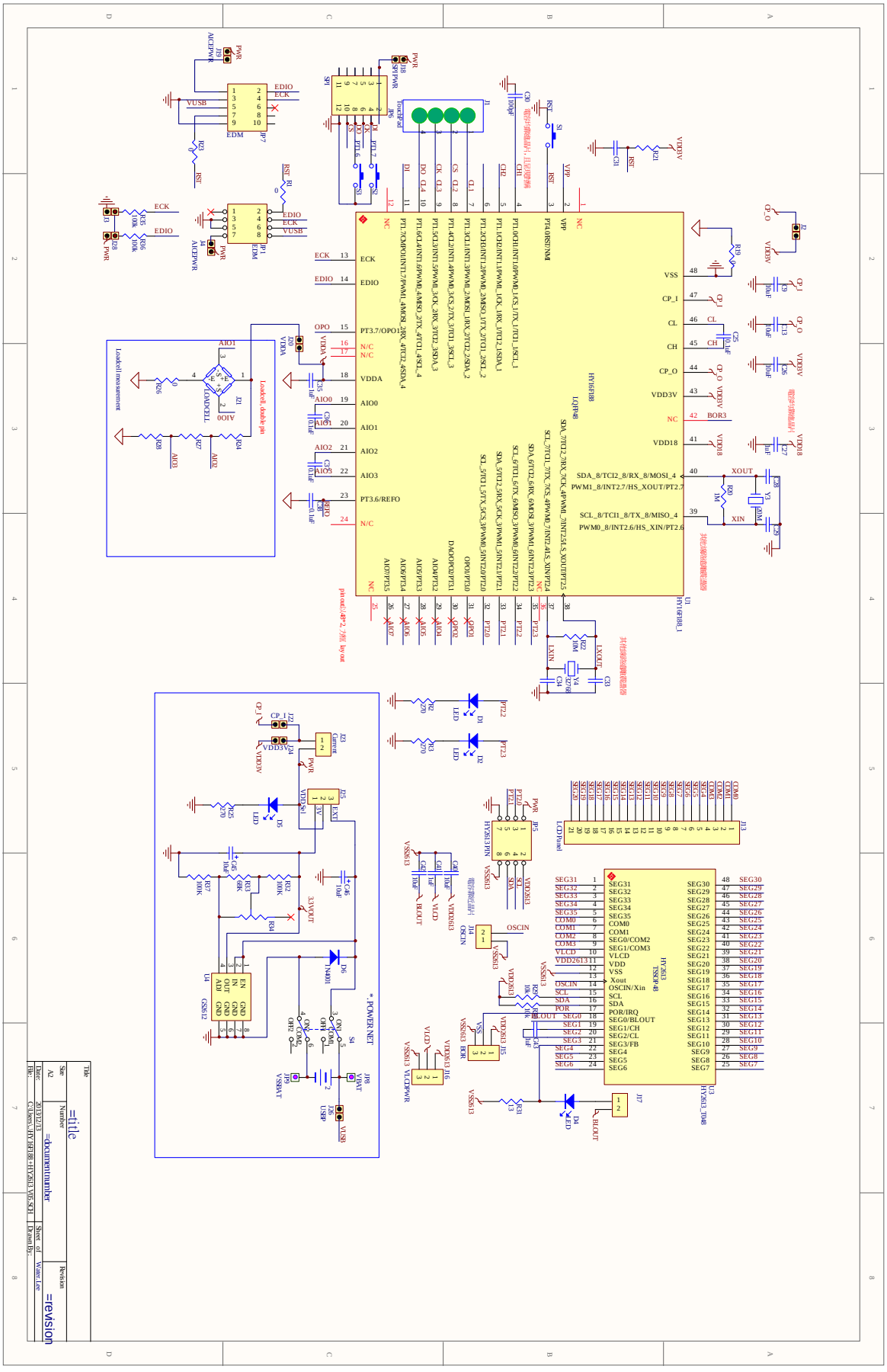

紘康 HY16F 系列 32 位元高性能 Flash 單片機(HY16F188)

(1)採用最新 Andes 32 位元 CPU 核心 N801 處理器。
(2)電壓操作範圍 2.4~3.6V，以及-40℃~85℃工作溫度範圍。
(3)支援外部 20MHz 石英震盪器或內部 20MHz 高精度 RC 震盪器，
擁有多種 CPU 工作時脈切換選擇，可讓使用者達到最佳省電規劃。
(3.1)運行模式 350uA@2MHz/2(3.2)待機模式 10uA@32KHz/2(3.3)休眠模式 2.5uA
(4)程式記憶體 64KBytes Flash ROM
(5)資料記憶體 08KBytes SRAM。
(6)擁有 BOR and WDT 功能，可防止 CPU 死機。
(7)24-bit 高精準度ΣΔADC 類比數位轉換器
(7.1)內置 PGA (Programmable Gain Amplifier)最高可達 128 倍放大。
(7.2)內置溫度感測器 TPS。
(8)超低輸入雜訊運算放大器 OPAMP。
(9)16-bit Timer A
(10)16-bit Timer B 模組具 PWM 波形產生功能
(11)16-bit Timer C 模組具數位 Capture/Compare 功能
(12)硬體串列通訊 SPI 模組
(13)硬體串列通訊 I2C 模組
(14)硬體串列通訊 UART 模組
(15)硬體 RTC 時鐘功能模組
(16)硬體 Touch KEY 功能模組

## 3.系統設計

### 3.1 硬體說明

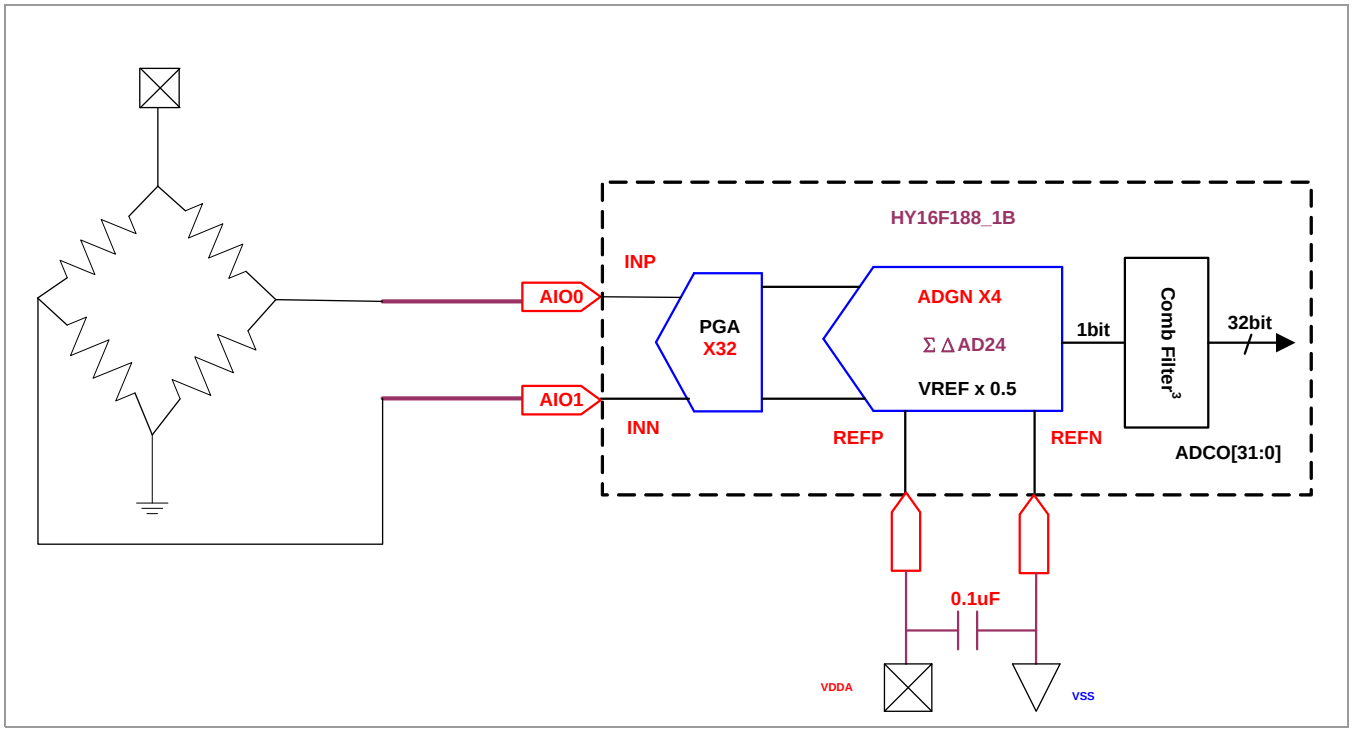
HY16F188 對於高精度廚房秤的應用，整體電路包含 4 個 touch key 部分及 LCD 顯示模組。
(A)中央處理器：
HY16F188 (Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + UMC 64K Flash)
(B)顯示晶片：HY2613 (HYCON LCD Driver LCD Segment 4X36)
(C)電源電路：5.0V 轉 3.3V 電源系統
(D)類比感測模組：內部 ADC
(E)線上燒錄與 ICE 連結電路，透過 EDM 的連接，可支援線上燒錄模擬。
並擁有強大的 C 平台 IDE 以及 HYCON 類比軟體分析工具與 GUI 等支援。



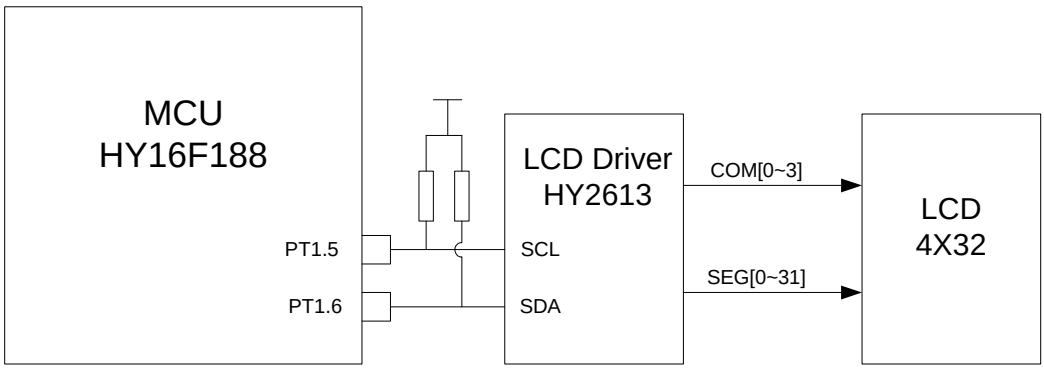
## 3.2 電路說明

ADC 測量電路

ADC 內部的 PGA 放大 32 倍，ADGN 放大 4 倍，
參考電壓由 VDDA –VSS 供給，則ΔVR_I=1.2V。

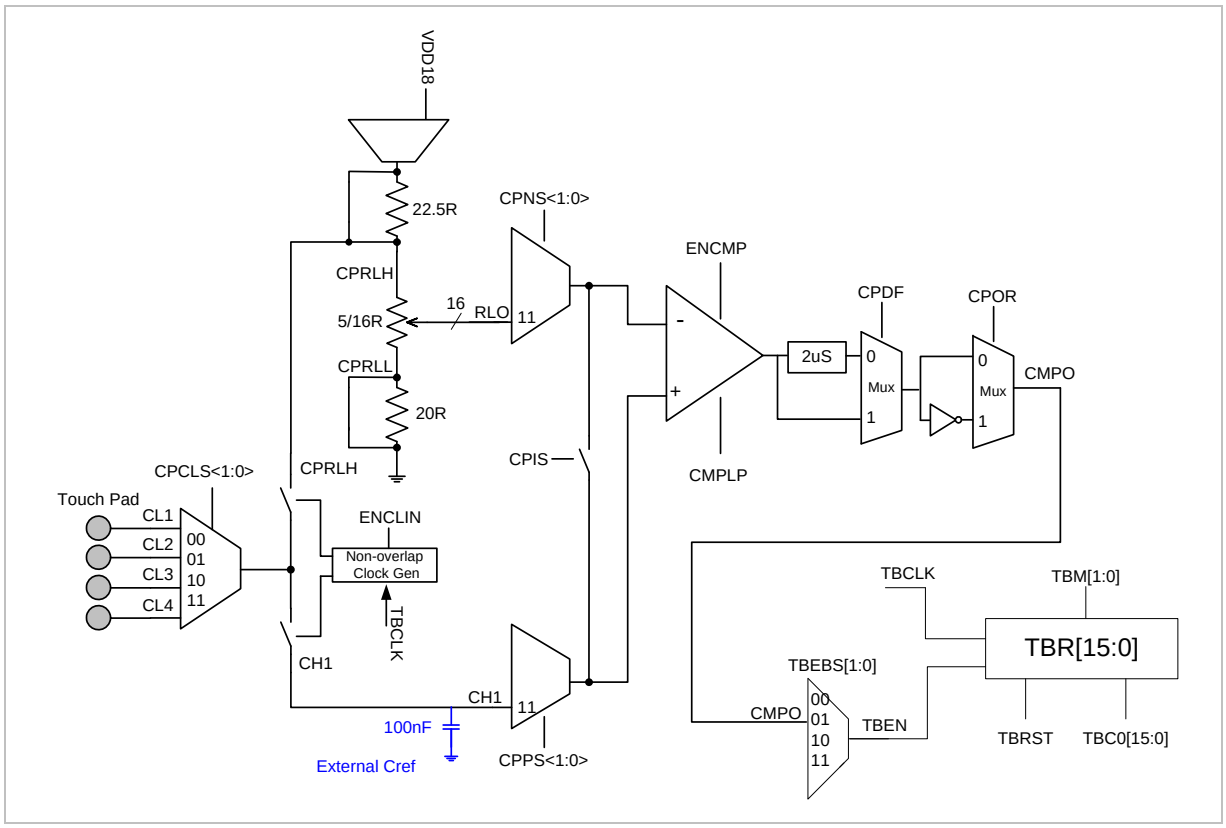



LCD DRIVER 路




MCU 通過 IIC 與 LCD driver 通訊，電路簡單，操作方便，只須將資料發送給 LCD driver HY2613，MCU 就可以處理其他事情，且更新資料方便。

內建硬體觸控模組(使用類比比較器方塊)



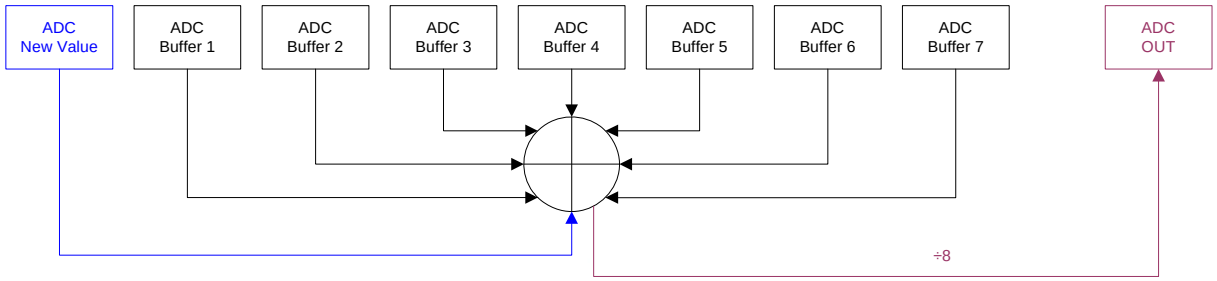

如上圖 所示，TOUCH KEY 週邊電路連接簡單，只需再 CMP 的正輸入端 CH1 端接入一個參考電容 Cref=10nf；CMP 的正輸入端配置為 CH1，與 touch key pad 的 CH1 端連接；負輸入端配置為 RLC，與 NON-OVERLAP 的輸出端 RLO 連接；NON-OVERLAP 的電壓源選擇 VDD18=1.8v，且 CPRLS=1 短路 22.5R 與 20R 電阻，設置 NON-OVERLAP 分壓輸出為 1/16R；啓動 TMB 且計數源為 CMPO。透過設置 CPIS=1，令 CMP 的輸入端短路，將 CH1 上的 Cref 電容上的電量通過 RLO 接到 VSS，進行完全放電；啓動比較器及 TMB 開始計數，啓動 NON-OVERLAP，讓 VDD 對 touch pad 充電，由於 NON-OVERLAP 的開關功能，touch PAD 對 CH1 Cref 充電，使得 CH1 端電壓慢慢上升，當 CH1 端電壓上升到 RLO 電位時，比較器輸出轉態 CMPO=0，產生 CMP 中斷標誌位元，停止 TMB 計數並記錄 TMBR 計數值，與設定的 TOUCH KEY 計數臨界值比較，若小於臨界值，表示有觸摸 Touch Pad，反則，沒有觸摸 Touch Pad。分別對不同的 touch pad 掃描。

## 3.2 軟體說明

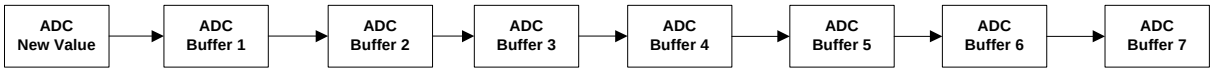
### ADC資料處理

ADC 設置為對輸入信號ΔSI 放大 128 倍，資料輸出率為 ADC-CK/32768，每秒輸出 10 筆資料，最終取有效位元數為 18Bit(使用者可自行調整)。截取原始資料 18Bit，進行平均滑動濾波處理。每 8 筆資料做一次平均值，得到的平均值再作為 ADC 最終轉換值。平均滑動濾波實現如圖所示。

由於小訊號放大到 128 倍，ADC 的輸出 Bit 只能達到 18 Bit，如果使用軟體平均方式可以再將 ADC 的解析度提升 1~2Bit 並使數值更加穩定。將新的 ADC 值與 7 個 ADC Buffer 值相加除以 8 輸出到 ADC OUT 如圖，此目的是將 8 筆 ADC 做平均輸出，這可以將 Noise 平均提高信號輸出的 Bit 數。



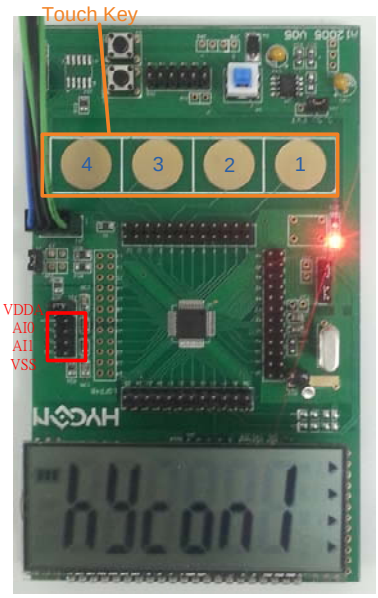

當 ADC 平均輸出後，將新值移到 Buffer 1 ，Buffer 1 移到 Buffer 2…Buffer6 移到 Buffer 7，如下圖。



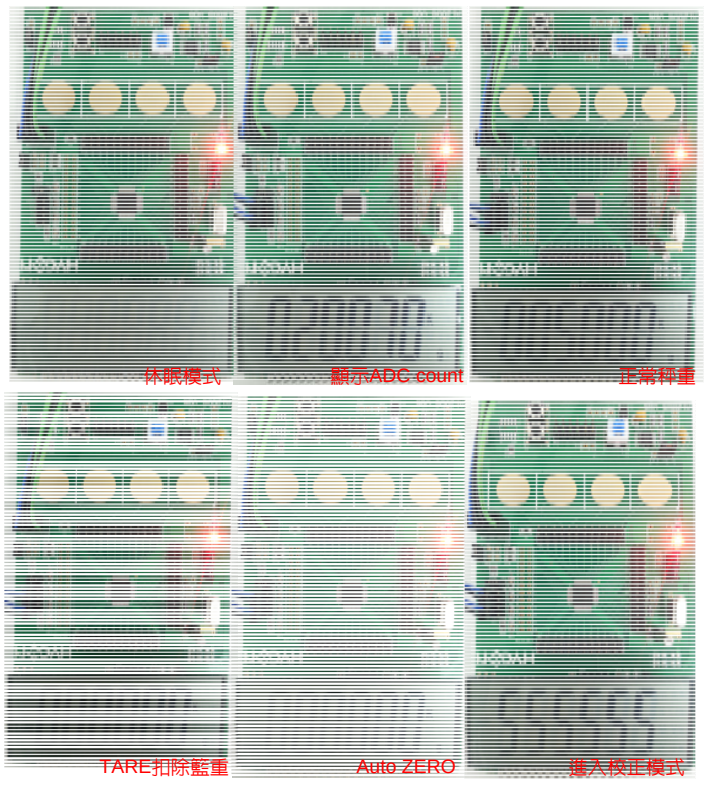
# 4.操作流程

## 4.1 操作方法

啓動後，如 Flash ROM 內已有校正值會顯示 HYCON 字樣，並抓取讀取 FlashROM 校正值及判斷零點是否需要更新，均完成後自動進入量測模式。

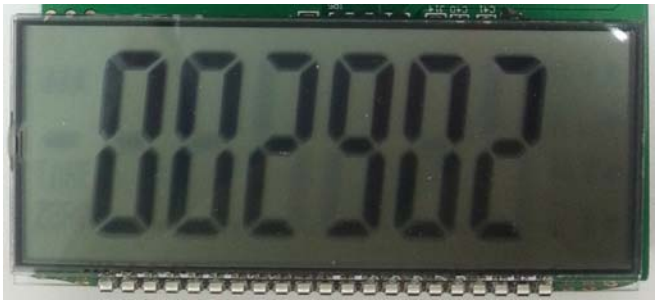



再量測模式下，輕觸 Touch key1 會關閉 LCD 顯示並進入休眠模式；輕觸 Touch key2 則將 LCD 顯示由 g 改為 AD count；輕觸 Touch key3 會扣除籃重；輕觸 Touch key4 會進入校正模式。



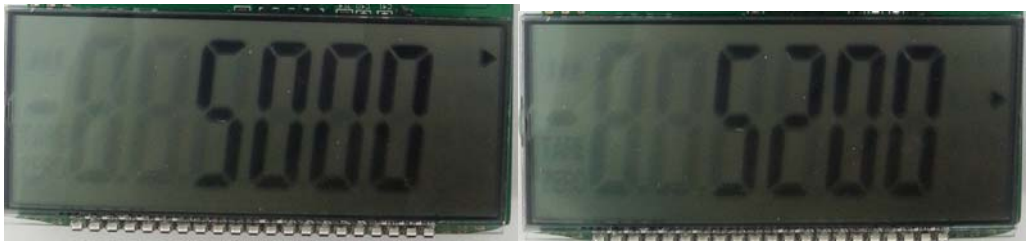
再扣除籃重下，會顯示 TARE；Auto zero 下，會顯示 ZERO；進入校正模式前會有 5.4.3.2.1 的倒數。

校正模式:
在校正模式下，將不會顯示任何單位。
程序一:設定 ADC 採用的 bit 數(24-1)，預設 18bit。平常顯示為當下 AD count。
透過 Touch key1.2 可以改變 bit 數並顯示現在 bit 數。設定完成後觸摸 Touch key4 進入下一個程序。



程序二:設定最大秤重，預設 8000。
程序三:設定校正重量，預設 5000。

程序 2.3 均藉由 Touch key1.2，進行數值的增減。並可透過右邊黑色的小三角形知道現在所設定的位數為哪一位。第一個黑色三角代表千位、第二個黑色三角代表百位，依序下去。
Touch key3 可以改變黑色三角位子；Touch key4 進入下一個程序。



程序四: 零點校正。此時 MCU 會自動抓取 ADC 值並判斷是否穩定。穩定後會自動進入下一個程序。如遲遲無法穩定可以觸碰 Touch key4 強迫抓值，並進入下一個程序。

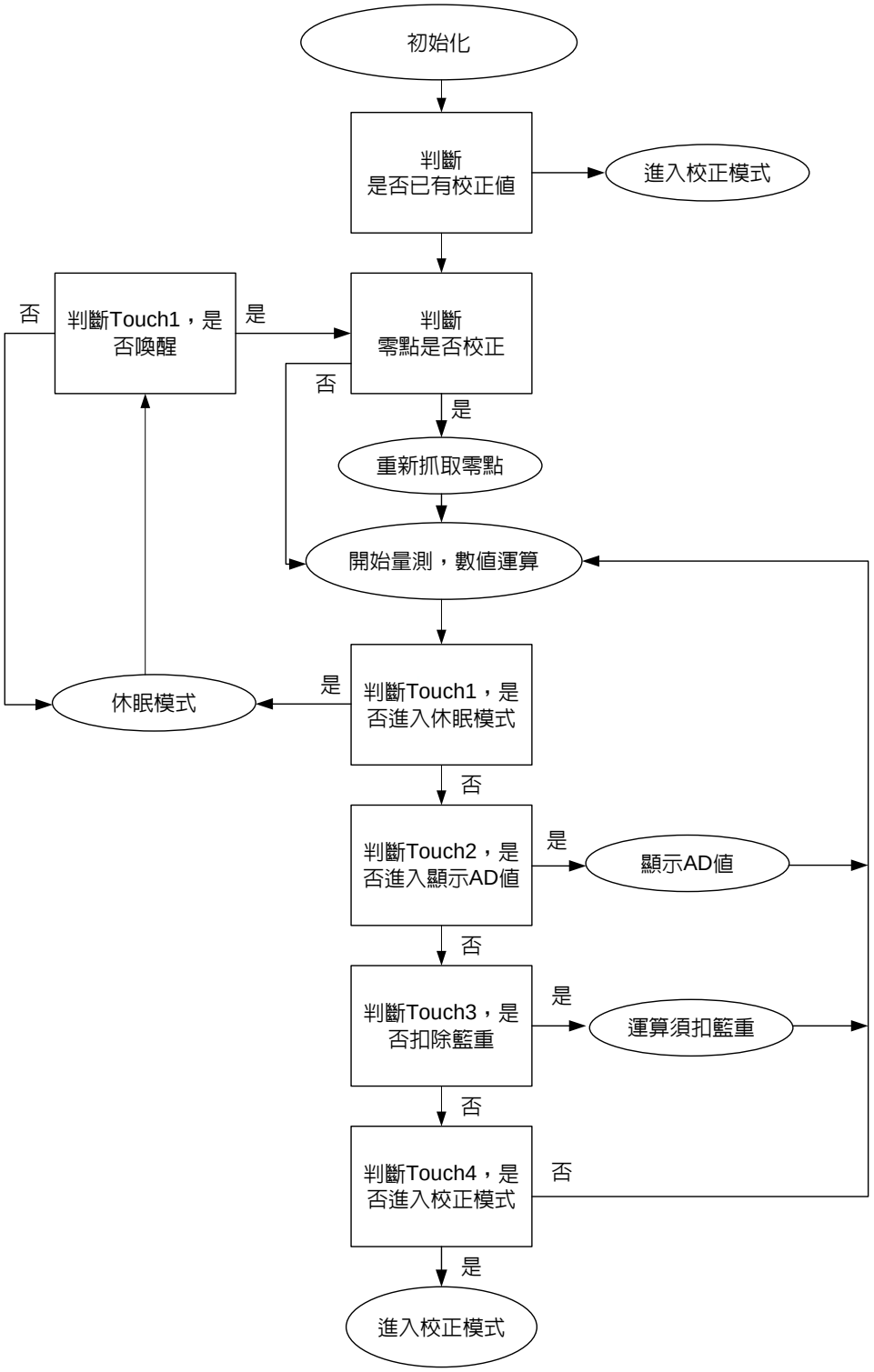
程序五:校正物校正。此時需放上校正物(須大於 50g)於秤台上，方能顯示 ADC 值，待 MCU 判斷為穩定後，自動進入下一個程序。如遲遲無法穩定可以觸碰 Touch key4 強迫抓值，並進入下一個程序。

程序六:會顯示 4 個數字。第一個數字代表精準度；第二個數字代表小數點顯示幾位；三、四則是程序一時，ADC 採用幾個 bit。一、二位數的操作方式與最大秤重設定一樣，藉由 Touch key1.2 進行數值的增減；Touch key3 改變黑色三角位子；Touch key4 進行下一個校正程序。

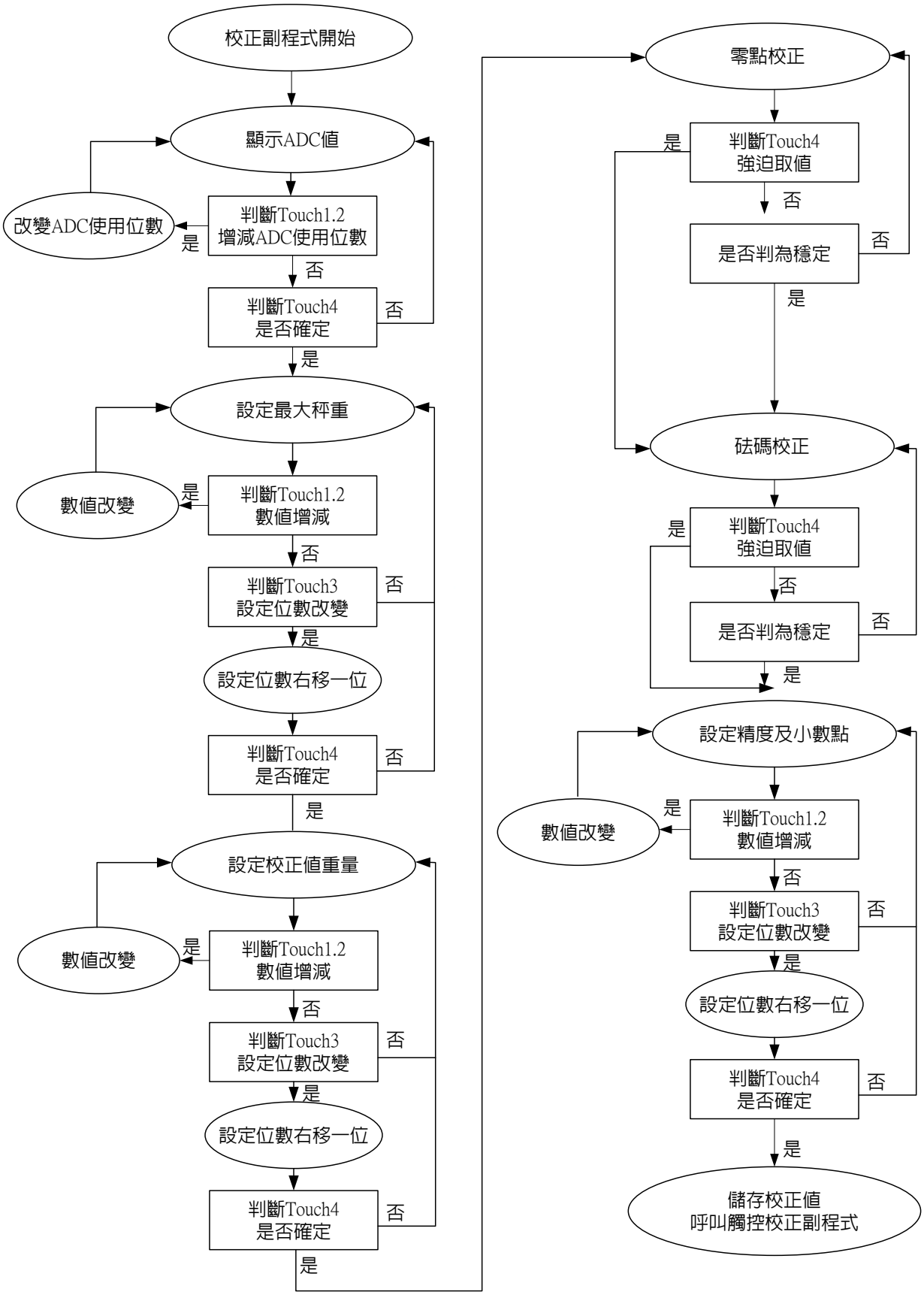
程序七:程序七開始的校正為 Touch key 的校正。一開始會自動抓取 untouch value，此時 LCD 會顯示 888888 與 000000 的交替，此時切勿碰觸 Touch Key，完成會自動進入下一個程序。

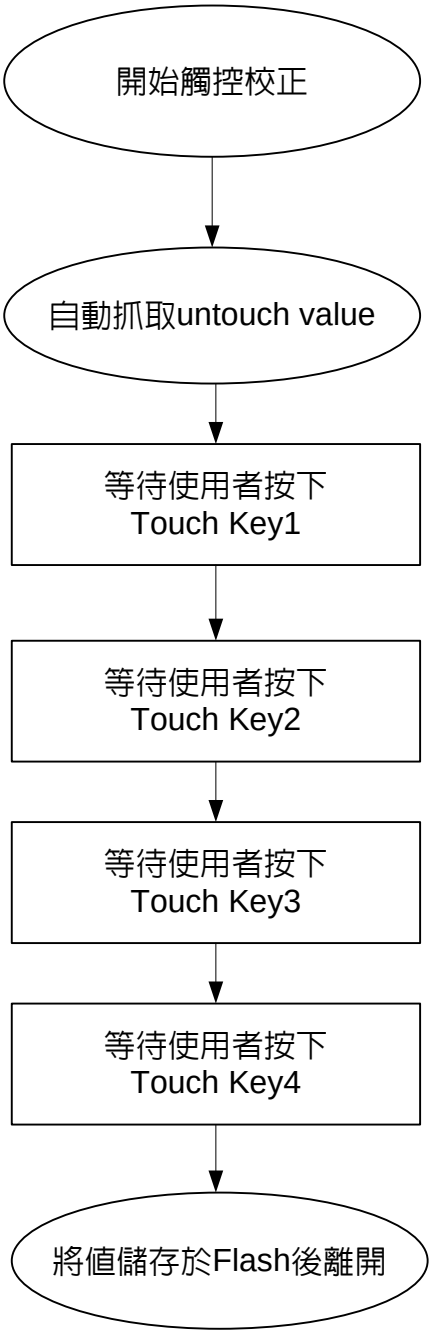
程序八: Touch key 的閥值抓取，LCD 螢幕上會出現 99XXXX 的數字，在 XXXX 等於甚麼數字時碰觸對應的 Touch Key。當 4 個 Touch Key 都碰觸過後，自動結束校正模式回到量測模式。

## 4.2 主程式流程


初始化
判斷
是否已有校正值
進入校正模式
否
判斷Touch1，是
否喚醒
是
判斷
零點是否校正
否
是
重新抓取零點
開始量測，數值運算
休眠模式
是
判斷Touch1，是
否進入休眠模式
否
判斷Touch2，是
否進入顯示AD值
是
顯示AD值
否
判斷Touch3，是
否扣除籃重
是
運算須扣籃重
否
判斷Touch4，是
否進入校正模式
否
是
進入校正模式

## 4.3 校正副程式


校正副程式開始
顯示ADC值
判斷Touch1.2
增減ADC使用位數
是
改變ADC使用位數
否
判斷Touch4
是否確定
否
是
設定最大秤重
判斷Touch1.2
數值增減
是
數值改變
否
判斷Touch3
設定位數改變
否
是
設定位數右移一位
判斷Touch4
是否確定
否
是
設定校正值重量
判斷Touch1.2
數值增減
是
數值改變
否
判斷Touch3
設定位數改變
否
是
設定位數右移一位
判斷Touch4
是否確定
否
是
零點校正
判斷Touch4
強迫取值
是
否
是否判為穩定
否
是
砝碼校正
判斷Touch4
強迫取值
是
否
是否判為穩定
否
是
設定精度及小數點
判斷Touch1.2
數值增減
是
數值改變
否
判斷Touch3
設定位數改變
否
是
設定位數右移一位
判斷Touch4
是否確定
否
是
儲存校正值
呼叫觸控校正副程式



## 5.技術規格

(1)電源接點：3.0V
(2)功耗：工作模式總消耗電流 7.51mA
　　晶片耗耗電流 1.76mA
　　Load Cel 消耗電流 5.75mA(內阻 400Ω)
(3)適用範圍：各種秤
(4)內外碼比： 8:1
(5)解析度： 1/8g
(6)精度： 1g
(7)反應時間：0.1s
(7)工作溫度：-40℃ ～ +85℃；
(8)存貯溫度：-55℃ ～ +125℃；
(9)相對濕度：＜95%（20±5℃條件）

## 6.量測結果

| 砝碼重量 | 測得重量 |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 10 | 10 |
| 100 | 100 |
| 1000 | 1000 |
| 2000 | 2000 |
| 4000 | 4000 |
| 6000 | 6000 |
| 8000 | 8000 |

單位g

## 7.結果總結

以 HY16F188 為主控結合內部高精度、多通道輸入、快速 ADC 的量測。不論廚房秤或是計價秤。都非常適合利用 HY16F188 來進行開發及應用。

## 8.附加檔案


HY16F011V03.zip

## 9.參考文獻

(1)HYCON HY16F188 Series Data Sheet
(2)HYCON HY16F188 Series User’ s Guide

## 10.修訂紀錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

| 版本 | 頁次 | 變更摘要 | 日期 |
|---|---|---|---|
| V1.0 | ALL | 初版發行 | 2013/11/26 |
| V2.0 | ALL | 改為 Touch key | 2013/12/12 |
| V3.0 | ALL | 更新程序 | 2014/12/18 |

附件:範例程式

```
/* Includes*/
/*-------------------------------------------------------------------------*/
/**************************************************************************
 *HY16F188
 * main.c
 * Created on:2013/12/08
 * Maintenance:2014/10/01
 * ------------------------------------------------------------------------
 * Release 1.0.0
 * Program Description:
 * --------------------
 *
 * ---------
 *          |                 --------------------
 *    PT2.0 | SDA ---> SDA | LCD Drive HY2613 |
 *    PT2.1 | SCK ---> SCK --------------------
 *     GND  |
 *          |
 * ---------
 **************************************************************************/
/*-------------------------------------------------------------------------*/
/* Includes                                                                */
/*-------------------------------------------------------------------------*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvI2C.h"
#include "DrvHWI2C.h"
#include "DrvCLOCK.h"
#include "DrvTimer.h"
#include "DrvADC.h"
#include "DrvPMU.h"
#include "DrvCMP.h"
#include "HY2613.h"
#include "my define.h"
#include "DrvFlash.h"
```

```
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
typedef union _MCUSTATUS
{
  char  _byte;
  struct
  {
    unsigned b_ADCdone:1;
    unsigned b_TMAdone:1;
    unsigned b_TMBdone:1;
    unsigned b_TMC0done:1;
    unsigned b_TMC1done:1;
    unsigned b_RTCdone:1;
    unsigned b_UART_TxDone:1;
    unsigned b_UART_RxDone:1;
  };
} MCUSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS  MCUSTATUSbits;
extern unsigned char seg[16];
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Ini_Touch(void);
void Initial_ALL(void);
void InitalADC(void);
void scalse_cn(void);
void AD01(void);
```

```
int ScanKey4(unsigned int TCH);
void touch_ct(void);
void g1(void);

void LCD_DATA_DISPLAY(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY2(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY3(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY4(unsigned int LcdBuffer);

/*-------------------------------------------------------------------------*/
/* Main Function                                                           */
/*-------------------------------------------------------------------------*/
int main(void)
{
        Ini_Touch();
        InitalADC();
        Initial_ALL();
        InitalI2C();
        Ini_Display();
        ClearLCDframe();

        MCUSTATUSbits._byte = 0;
        pass=ReadWord(0x801c);              //判斷是否 Flash 內含有校正值
        if(pass==0x11100)
        {
                decimal=ReadWord(0x8004);    //有，讀取各校正值
                ADnum=ReadWord(0x8008);
                resolution=ReadWord(0x800c);
                ADzero=ReadWord(0x8010);
                ADcn=ReadWord(0x8014);
                cn=ReadWord(0x8018);
                gain=ReadWord(0x8030);
                threh0=ReadWord(0x802c);
                threh1=ReadWord(0x8020);
                threh2=ReadWord(0x8024);
                threh3=ReadWord(0x8028);
        }
```

```
        else
        {
                touch_ct();
                   //threh0=238;
                   //threh1=153;
                   //threh2=143;
                   //threh3=111;
                   scalse_cn();
        }

DisplayHYcon();
while(1)
  {

          pmu_00=0xff00ff00;                    //將比較器功能關閉(省電)
          cmp_00=0xff00;
          cmp_04=0xff00ff00;
          asm("NOP");
          clk_08=0xff00ff0d;                    //關閉高速震盪器，使用低速(省電)
          Delay(0x100);
          clk_00=0xff00ff00;
          sys_04=0xff10;
          asm("standby 0");
          asm("NOP");
          DrvADC_Disable();                              //關閉 ADC 功能(省電)

          if(t==1)                          //(t=1 代表 TMA 已進入中斷)
          {
                  t=0;                          //喚醒 CPU
                  pmu_00=0x03031b1b;    //開啟比較器功能(開啟 touch 功能)
                  cmp_00=0x13c3c;
                  cmp_04=0x101d3d3;

                  clk_00=0xff01;                //開啟高速
                  clk_08=0xff4c;
                  TCH=0;ScanKey4(TCH); //掃描 touch1 是否被按下，touch1 被按下 LED
                  if(LED>=1)
                  {
```

```
DrvADC_Enable();      //開啟 ADC 功能
DisplayHYcon();;      //等待 ADC 穩定時顯示 HYCON
//Delay(0xf0000);
LED=0;      //每次啟動，都會在偵測一次零點

AD01();
g=average-ADzero;
if(g<70)    //當 ADD 值與上次零點相差 70 以內自動更新零點
{
        c=10;
        while(c)
        {
              AD01();
              ADzero=average;
              Delay(0xf000);
              AD01();
              g=ADzero-average;
              if(g==0)
              {
                    next++;
              }
              else {next=0;}
              if(next>3) c=0;
        }
        ADzero=average;
}

ClearLCDframe();
while(1)
{
        g1(); //呼叫正亮演算副程式
        if(LED>=10)
//tocuh1 長時間備按住當 LED 加至 15 時，離開溫度顯示，再次進入省電狀態
        {
              LED=0;
              ClearLCDframe();  //清除 LCD
              Delay(0xFF00);
              //LCD_DATA_DISPLAY(ADzero);
```

```
                                        //Delay(0xf1000);
                                        goto bb;     //關閉 LCD 準備進入省電模式
                                }
                                TCH=1;ScanKey4(TCH); //掃描 touch key2
                                if(LED<=-10)
                                {
                                        LED=0;
                                        if(bb==0) bb=1;  //顯示 AD 值旗標=1
                                        else bb=0;    //再按一次則取消
                                }
                                g1();                   //呼叫正亮演算副程式
                                TCH=3;ScanKey4(TCH);//掃描 touch key4
                                if(c<=-10)
                                {
                                        scalse_cn(); //呼叫校正副程式
                                }
                                g1();                   //呼叫正亮演算副程式
                                TCH=2;ScanKey4(TCH);   //掃描 touch key3
                                if(enter>=10)
                                {
                                        enter=0;
                                        if(tare==0) tare=1;
                                        else tare=0;
                                        tareg=g;
                                }
                                Delay(0x2000);
                                TCH=0;ScanKey4(TCH);//掃描 touch key1
                        }
                }
        }
        t=0;
bb:     ClearLCDframe();                //清除 LCD 顯示
  }



while(1);
```

```
  return 0;
}
void g1(void)
{
        unsigned long long buu;
        buu=0;
        buu=buu-1;
          AD01();                                  //呼叫 ADC 副程式

          if(bb==1) LCD_DATA_DISPLAY(average);//顯示 AD 值

          else
          {
                  switch(decimal)      //判斷小數點位數，並給於相對應的倍數值
                  {
                          case 4:{dou=100;}break;
                          case 3:{dou=10;}break;
                          case 2:{dou=1;}break;
                          case 5:{dou=1000;}break;
                          case 6:{dou=10000;}break;
                          default: break;
                  }
                  if(average<ADzero)                    //重量為負時
                  {
                          gg=ADzero-average;            //Autozero 判斷
                          if(gg<=2)
                          {
                                  autozero=1;
                                  g=0;
                          }
                          else
                          {
                                  autozero=0;
                                  nn=1;                          //負號旗標為 1
                                  buu=10*cn/resolution;
                                  buu=buu*(ADzero-average);   //重量演算
                                  buu=buu/dou*100;
                                  buu=buu/gain;
```

```
                gg=buu%10;
                if(gg>5) buu+=10;
                g=buu/10;
                g=g*resolution;
        }
}
else
{
        gg=average-ADzero;          //重量為正
        if(gg<=2)                     //Autozero 判斷
        {
                autozero=1;
                g=0;
        }
        else
        {
                autozero=0;
                nn=0;                 //負號旗標為 0
                buu=10*cn/resolution;
                buu=buu*(average-ADzero);  //重量演算
                buu=buu/dou*100;
                buu=buu/gain;
                gg=buu%10;
                if(gg>5) buu+=10;
                g=buu/10;
                g=g*resolution;
        }
}
if(tare==1)
{
        g=g-tareg;
        if(g<0)
        {
                g=g*-1;
                nn=1;
        }
        else nn=0;
}
```

```
                if(g==0) nn=0;
                LCD_DATA_DISPLAY(g);

        }
}
void scalse_cn(void)                    //溫度校正副程式
{
        DrvADC_Enable();                //開啟 ADC
        nn=0;
        autozero=0;
        tare=0;
        intest=1;
        ClearLCDframe();
        LCD_DATA_DISPLAY(555555);Delay(0x10000);
        LCD_DATA_DISPLAY(444444);Delay(0x10000);
        LCD_DATA_DISPLAY(333333);Delay(0x10000);
        LCD_DATA_DISPLAY(222222);Delay(0x10000);
        LCD_DATA_DISPLAY(111111);Delay(0x10000);
        ADnum=6;
        c=2;
        LED=ADnum;
        int o;
        while(c)
        {
                AD01();
                LCD_DATA_DISPLAY(average);    //顯示現在 ADC 值
                TCH=3;ScanKey4(TCH);          //掃描 touch key1.2.4
                TCH=0;ScanKey4(TCH);
                TCH=1;ScanKey4(TCH);
                o=ADnum-LED;
                if(o==0)        asm("NOP");
                else
                {
                        ADnum=LED;
                        if(LED>16) LED=0;
                        ADnumber=24-ADnum;    //將右移位數改為 ADC 使用 bit 數
                        LCD_DATA_DISPLAY(ADnumber);   //顯示 bit 數
                        Delay(0x1f700);
```

```
                }
                Delay(0x2000);
        }
        ADnumber=24-ADnum;
        ClearLCDframe();
        Delay(0x1ffff);
        unsigned char lcd1,lcd2,lcd3,lcd4;
        nu=1;
        lcd1=8;
        lcd2=lcd3=lcd4=0;
        c=2;
        first=1;
        while(c)
        {
                for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}     //掃描全部 youch key
                LCD_DATA_DISPLAY1(lcd1);
                LCD_DATA_DISPLAY2(lcd2);
                LCD_DATA_DISPLAY3(lcd3);
                LCD_DATA_DISPLAY4(lcd4);
                switch(nu)
                {
                        case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;//令欲改變值+1
                        case
2:{if(first==1){first=0;LED=lcd2;}if(LED>10)LED=0;lcd2=LED;}break;//令欲改變值-1
                        case
3:{if(first==1){first=0;LED=lcd3;}if(LED>10)LED=0;lcd3=LED;}break;//改變欲改變值
                        case
4:{if(first==1){first=0;LED=lcd4;}if(LED>10)LED=0;lcd4=LED;}break;   //確認
                }
                if(enter>=1)
                {
                        enter=0;
                        nu++;                   //改變欲改變值
                        first=1;
                        if(nu>=5) nu=1;
                }
                Delay(0x7f00);
```

```
        }
        max=(lcd1*1000)+(lcd2*100)+(lcd3*10)+lcd4; //加總換算，存入最大秤重暫存器
        ClearLCDframe();
        Delay(0x1f000);
        lcd1=5;
        lcd2=lcd3=lcd4=0;
        c=2;
        nu=1;
        first=1;
        while(c)
        {
                for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}
                LCD_DATA_DISPLAY1(lcd1);
                LCD_DATA_DISPLAY2(lcd2);
                LCD_DATA_DISPLAY3(lcd3);
                LCD_DATA_DISPLAY4(lcd4);
                switch(nu)
                {
                        case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;//令欲改變值+1
                        case
2:{if(first==1){first=0;LED=lcd2;}if(LED>10)LED=0;lcd2=LED;}break;//令欲改變值-1
                        case
3:{if(first==1){first=0;LED=lcd3;}if(LED>10)LED=0;lcd3=LED;}break;//改變欲改變值
                        case
4:{if(first==1){first=0;LED=lcd4;}if(LED>10)LED=0;lcd4=LED;}break; //確認
                }
                if(enter>=1)
                {
                        first=1;
                        enter=0;
                        nu++;                   //改變欲改變值
                        if(nu>=5) nu=1;
                }
                Delay(0xff00);
        }
        cn=(lcd1*1000)+(lcd2*100)+(lcd3*10)+lcd4;    //加總運算，存入法麻校正值
        ClearLCDframe();
```

```
Delay(0x1f000);
next=0;
c=2;
int z;
while(c)
{
        TCH=3;ScanKey4(TCH);            //觸摸 touch key4 能強迫取值
        AD01();
        ADzero=average;
        LCD_DATA_DISPLAY(ADzero);
        Delay(0xf000);
        AD01();
        z=ADzero-average;
        if(z==0)
        {
                next++;
        }
        else {next=0;}
        LCD_DATA_DISPLAY(ADzero);
        if(next>3) c=0;
}
next=0;
c=2;
ClearLCDframe();
Delay(0x1f000);
while(c)
{
        AD01();
        if(average>(ADzero*115/100))
        {
                TCH=3;ScanKey4(TCH);
                AD01();
                ADcn=average;
                LCD_DATA_DISPLAY(ADcn);
                Delay(0xf000);
                AD01();
                z=ADcn-average;
                if(z==0)
```

```
                        {
                                next++;
                        }
                        else {next=0;}
                        LCD_DATA_DISPLAY(ADcn);
                        if(next>3) c=0;
                }
        }
        ClearLCDframe();
        Delay(0x1f000);
        lcd1=1;
        lcd2=3;
        lcd3=ADnumber/10;
        lcd4=ADnumber%10;
        nu=1;
        c=2;
        first=1;
        while(c)
        {
                for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}
                LCD_DATA_DISPLAY1(lcd1);
                LCD_DATA_DISPLAY2(lcd2);
                LCD_DATA_DISPLAY3(lcd3);
                LCD_DATA_DISPLAY4(lcd4);
                switch(nu)
                {
                        case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;
                        case
2:{if(first==1){first=0;LED=lcd2;}if(LED>=6)LED=0;lcd2=LED;}break;
                }
                if(enter>=1)
                {
                        enter=0;
                        nu++;
                        first=1;
                        if(nu>=3) nu=1;
                }
```

```
                Delay(0xff00);
        }
        resolution=lcd1;
        decimal=7-lcd2;
        gain=ADcn-ADzero;
        DrvADC_DisableInt();                              //關閉 ADC 中斷
        DrvFlash_Burn_Word(0x8004,0x600,decimal);      //將校正值燒入 Flash
        DrvFlash_Burn_Word(0x8008,0x600,ADnum);
        DrvFlash_Burn_Word(0x800c,0x600,resolution);
        DrvFlash_Burn_Word(0x8010,0x600,ADzero);
        DrvFlash_Burn_Word(0x8014,0x600,ADcn);
        DrvFlash_Burn_Word(0x8018,0x600,cn);
        DrvFlash_Burn_Word(0x8030,0x600,gain);
        DrvADC_EnableInt();                  //開起 ADC 中斷
        ClearLCDframe();
        Delay(0xffff);
        touch_ct();

}
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Hardware Communication Interrupt                                          */
/* UART/SPI/I2C Interrupt Service Routines                                   */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
  unsigned char I2C_Status;

  if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT)  // Get I2C Interrupt Flag
  {
    I2C_Status=DrvI2C_GetStatusFlag();  // Get I2C Status Flag
    switch(I2C_Status)
    {
      case 0x90: //MACTFlag+RWFlag
          { /* START has been transmitted */
            DrvI2C_WriteData(I2C_TARGET);  //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
            break;
```

```
    };
case 0x84: //MACTFlag+ACKFlag
    { /* Slave A + W has been transmitted. ACK has been received. */
      DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      break;
    };
case 0x80: //MACTFlag
    { /* Slave A + W has been transmitted. ACK has been received. */
      DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      break;
    };
case 0x30:
    {
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      EndFlag=1;
      break;
    };
case 0x8C: // MACTFlag+DFFlag+ACKFlag
    { /* DATA has been transmitted and ACK has been received */
      if(DataTxIndex<DataTxLen)
      {
        DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
        DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      }
      else
      {
        if(I2C_RW == WRITE)
        {
          DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
          EndFlag=1;
        }
        else if(I2C_RW == READ)
          DrvI2C_Ctrl(1,0,0,0);  // I2C as master sends START signal
        DataTxIndex=0;
      }
      break;
```

```
    };
case 0x88: //MACTFlag+DFFlag
    { /* DATA has been transmitted and NACK has been received */
      DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
      DataTxIndex=0;
      EndFlag=1;
      break;
    };
case 0xB0:
    { /* A repeated START has been transmitted. */
      DrvI2C_WriteData(I2C_TARGET | READ);  //Send Slave Address & R/W Bit
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      break;
    }
case 0x94: //MACTFlag+RWFlag
    { /* Slave A + R has been transmitted. ACK has been received. */
      DrvI2C_Ctrl(0,0,0,1);  // Set ACK bit
      break;
    };
case 0x9C: //MACTFlag+RWFlag+DFFlag+ACKFlag
    { /* Data byte has been received. ACK has been transmitted. */
      if(DataRxLen>DataRxIndex)
      {
        Recbuf[DataRxIndex++]=DrvI2C_ReadData();
        DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      }
      else
      {
        Recbuf[DataRxIndex++]=DrvI2C_ReadData();
        DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
        EndFlag=1;
      }
      break;
    };
case 0x98: //MACTFlag+RWFlag+DFFlag
    { /* Data byte has been received. NACK has been transmitted. */
      DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
      EndFlag=1;
```

```
            break;
          };
      default:
          {
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
            EndFlag=1;
            break;
          };
    }
    DrvI2C_ClearEIRQ();
    DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CIF)
    SYS_EnableGIE(7,0x3F);       // Enable GIE(Global Interrupt)
  }
  if(DrvI2C_ReadIntFlag()==E_DRVI2C_ERROR_INT)  //Get I2C Error Interrupt Flag
  {
    EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,1,0,0);   // I2C as master sends STOP signal
    SYS_EnableGIE(7,0x3F);       // Enable GIE(Global Interrupt)
  }
}

/*-------------------------------------------------------------------------*/
/* WDT & RTC & Timer A/B/C Interrupt Service Routines
*/
/*-------------------------------------------------------------------------*/
void HW1_ISR(void)
{
          DrvTIMER_ClearIntFlag(E_TMA);        //Clear TMA interrupt flag
          t=1;
          asm volatile("sethi $r0, 0xc0000"); //預防中斷不可預期的關閉
          asm volatile("ori  $r0, $r0, 0x003f");
          asm volatile("mtsr $r0, $INT_MASK");
          asm volatile("movi $r0, 0x70009");
          asm volatile("mtsr $r0, $PSW");
}
```

```
/*-------------------------------------------------------------------------*/
/* HW2 ADC Interrupt Subroutines                                           */
/*-------------------------------------------------------------------------*/
void HW2_ISR(void)
{
          DrvADC_ClearIntFlag();                    //清除 AD 中斷旗標
          ADCData=adc_08;
          ADCData=ADCData>>8;
          ok=1;
          asm volatile("sethi $r0, 0xc0000");     //預防中斷不可預期的關閉
          asm volatile("ori  $r0, $r0, 0x003f");
          asm volatile("mtsr $r0, $INT_MASK");
          asm volatile("movi $r0, 0x70009");
          asm volatile("mtsr $r0, $PSW");
}

/*-------------------------------------------------------------------------*/
/* CMP/OPA Interrupt Service Routines
*/
/*-------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* PT1 Interrupt Service Routines                                          */
/*-------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* PT2 Interrupt Service Routines                                          */
/*-------------------------------------------------------------------------*/
void HW5_ISR(void)
{
```

```
}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
  for(;num>0;num--)
    asm("NOP");
}
void AD01(void)
        //ADC 副程式
{
        unsigned char t;
      while(ok)                         //ADC 中斷後 OK=1
        {
                ok=0;
                ADtemp10=(ADCData>>ADnum);

                value_buf[8]=value_buf[7];          //平均滑動濾波處理
                value_buf[7]=value_buf[6];
                value_buf[6]=value_buf[5];
                value_buf[5]=value_buf[4];
                value_buf[4]=value_buf[3];
                value_buf[3]=value_buf[2];
                value_buf[2]=value_buf[1];
                value_buf[1]=ADtemp10;
                sum=0;
                for(t=4;t>0;t--)
                {
                        sum+=value_buf[t];
                }
                average=sum>>2;
        }
}
int ScanKey4(unsigned int TCH)                //touch key 副程式
{
```

```
  CL=TCH;pio_00=0x0100;                          //電容放電
  DrvTMB_ClearTMB();
        //TMB 計時歸零
  DrvCMP_EnableNonOverlap(CL);                   //選擇充電通道
  cmpresult=DrvCMP_ReadData();                   //讀取比較器輸出
  while(cmpresult==1){cmpresult=DrvCMP_ReadData();}   //確定輸出反向
  tmrbcnt=DrvTMB_CounterRead();//TMB //判斷充電時間是否低於閥值
  switch(CL)
  {
    case 0 : if(tmrbcnt<=threh0){LED++;}break;
    case 1 : if(tmrbcnt<=threh1){LED--;}break;
    case 2 : if(tmrbcnt<=threh2){enter++;}break;
    case 3 : if(tmrbcnt<=threh3){c--;}break;
    default: break;
  }
  pio_00=0x0101;                          //電容充電
  pio_04=0x0100;
  return 0;
}
void touch_ct(void)
        //touch key 校正副程式
{
        int z,y;

                LCD_DATA_DISPLAY(888888);
                Delay(0x8000);
                unth0=0;                 //抓取 touch1 untouch 值(8 筆取平均)
                for(z=0;z<8;z++)
                {
                        TCH=0;ScanKey4(TCH);
                        unth0+=tmrbcnt;
                        Delay(0xFFF);
                }
                unth0=unth0>>3;
                LCD_DATA_DISPLAY(000000);
                unth1=0;                    //抓取 touch2 untouch 值(8 筆取平均)
                for(z=0;z<8;z++)
                {
```

```
        TCH=1;ScanKey4(TCH);
        unth1+=tmrbcnt;
        Delay(0xFFF);
}
unth1=unth1>>3;
LCD_DATA_DISPLAY(888888);
unth2=0;                    //抓取 touch3 untouch 值(8 筆取平均)
for(z=0;z<8;z++)
{
        TCH=2;ScanKey4(TCH);
        unth2+=tmrbcnt;
        Delay(0xFFF);
}
unth2=unth2>>3;
LCD_DATA_DISPLAY(000000);
unth3=0;                    //抓取 touch4 untouch 值(8 筆取平均)
for(z=0;z<8;z++)
{
        TCH=3;ScanKey4(TCH);
        unth3+=tmrbcnt;
        Delay(0xfff);
}
unth3=unth3>>3;
LCD_DATA_DISPLAY(991111);
th0=0;                      //抓取 touch1 touch 值(8 筆取平均)
y=8;
while(y)
{
        TCH=0;ScanKey4(TCH);
        if(tmrbcnt<435)
       {
                y--;
                th0+=tmrbcnt;
       }
}
th0=th0>>3;
LCD_DATA_DISPLAY(992222);
th1=0;                  //抓取 touch2 touch 值(8 筆取平均)
```

```
y=8;
while(y)
{
        TCH=1;ScanKey4(TCH);
        if(tmrbcnt<220)
        {
                y--;
                th1+=tmrbcnt;
        }
}
th1=th1>>3;
LCD_DATA_DISPLAY(993333);
th2=0;        //抓取 touch3 touch 值(8 筆取平均)
y=8;
while(y)
{
        TCH=2;ScanKey4(TCH);
        if(tmrbcnt<180)
        {
                y--;
                th2+=tmrbcnt;
        }
}
th2=th2>>3;
LCD_DATA_DISPLAY(994444);
th3=0;        //抓取 touch4 touch 值(8 筆取平均)
y=8;
while(y)
{
        TCH=3;ScanKey4(TCH);
        if(tmrbcnt<180)
        {
                y--;
                th3+=tmrbcnt;
        }
}
th3=th3>>3;
threh0=unth0-(((unth0-th0)*3)/4);        //計算校正值
```

```
                threh1=unth1-(((unth1-th1)*3)/4);
                threh2=unth2-(((unth2-th2)*3)/4);
                threh3=unth3-(((unth3-th3)*3)/4);
                LED=0;
                enter=0;
                DrvADC_DisableInt();                //關閉 ADC 中斷
                DrvTIMER_DisableInt(E_TMA);   //關閉 TMA 中斷
        DrvFlash_Burn_Word(0x801c,0x600,0x11100;//將判斷是否校正過用數值燒入 Flash
                DrvFlash_Burn_Word(0x802c,0x600,threh0); //將校正值存入 Flash
                DrvFlash_Burn_Word(0x8020,0x600,threh1);
                DrvFlash_Burn_Word(0x8024,0x600,threh2);
                DrvFlash_Burn_Word(0x8028,0x600,threh3);
                Delay(0x8000);
                DrvTIMER_EnableInt(E_TMA);   //開啟 TMA 中斷
                DrvADC_EnableInt();   //開啟 ADC 中斷
                intest=0;
}
void Ini_Touch(void)
{
  DrvCMP_Enable();                                  //CMP enable
  DrvCMP_Open(1,1,1);                          //CMP open
  DrvCMP_PInput(0);                           //Comparator positive input  CH1
  DrvCMP_NInput(3);                          //Comparator negative input  RLO
  DrvCMP_RLO_refV(2,1);
  DrvCMP_DisableNonOverlap();
  DrvCMP_RLO_Ctrl(0,1);
  DrvCMP_InputSwitch(1);
  DrvCMP_InputSwitch(0);
  DrvCMP_RLO_Ctrl(1,0);
  DrvTMB_ClearTMB();
  DrvCLOCK_SelectIHOSC(0x00);
  DrvCLOCK_SelectMCUClock(0,0);   //select MCU clock as LS_CK,div1
  DrvTMBC_Clk_Source(1,3);////TimerB Clock enable pre_scale 1
//TimerB overflow 0xffff->PWM period 0xffff
  DrvTMB_Open(E_TMB_MODE0,E_TMB_CMP_HIGH,0xffff);
  DrvTMB_ClearTMB();
  //Set VDDA voltage
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
```

```
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_BandgapEnable();
  DrvPMU_REFO_Enable();
  DrvPMU_AnalogGround(Enable);          //ADC analog ground source selection.
          //1 : Enable buffer and use internal source(need to work with ADC)
DrvPMU_LDO_LowPower(Enable);//VDD LDO with low power control.0 : Normal;1 : Low power
  Delay(0x1000);
}
/*-------------------------------------------------------------------------------*/
void InitalADC(void)
{
  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AI00,ADC_Input_AI01);//ADC Input
  DrvADC_InputSwitch(OPEN);
  DrvADC_RefInputShort(OPEN);
  DrvADC_Gain(7,3);           //Set the ADC Gain
  DrvADC_DCoffset(0);         //DC offset input voltage selection
  DrvADC_RefVoltage(VDDA,0); //Set the ADC reference voltage.
  DrvADC_FullRefRange(1);     //Set the ADC reference range select.
                              //0:Full reference range input
                              //1:1/2 reference range input
  DrvADC_OSR(0);              //0:OSR=32768
  DrvADC_CombFilter(Enable);
  DrvADC_ClkEnable(0,1);     //Setting ADC CLOCK ADCK=HS_CK/6
  //Set VDDA voltage
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_BandgapEnable();
  DrvPMU_REFO_Enable();
  DrvPMU_AnalogGround(Enable);//ADC analog ground source selection.
                              //1:Enable buffer and use internal source
  DrvPMU_LDO_LowPower(0);     //VDD LDO with low power control.
                              //0 : Normal;1 : Low power
  Delay(0x1000);
  //Set ADC interrupt
  DrvADC_EnableInt();
  DrvADC_ClearIntFlag();
  DrvADC_Enable();
```

```
}
void Initial_ALL(void)
{
        pio_00=0xff00ff00;                              //確保沒用的腳位關閉
        pio_04=0xff00ff00;
        pio_08=0x00;
        pio_10=0xff00ff03;
        pio_14=0xff00ff03;

        autozero=0;                                     //設定旗標
        LED=0;
        t=0;
        TCH=0;
        c=0;
        enter=0;
        intest=0;


        DrvTMA_Open(9,1);                    //TimerA Overflow
                                            //9:taclk/1024/32;TMRDV=÷32
                                             //00:HS_CK
        DrvTIMER_ClearIntFlag(E_TMA);        //Clear Timer A interrupt flag
        DrvTIMER_EnableInt(E_TMA);           //Timer A interrupt enable

        DrvCLOCK_SelectIHOSC(0);
        DrvCLOCK_EnableHighOSC(E_INTERNAL,50);        // Select HAO 4MHz
        SYS_EnableGIE(7,0x3F);  // Enable GIE(Global Interrupt)
}
/*-------------------------------------------------------------------------*/
/* End Of File*/
/*-------------------------------------------------------------------------*/
```