高精度廚房秤_應用說明書

HY16F198

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，因協力廠商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　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　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。

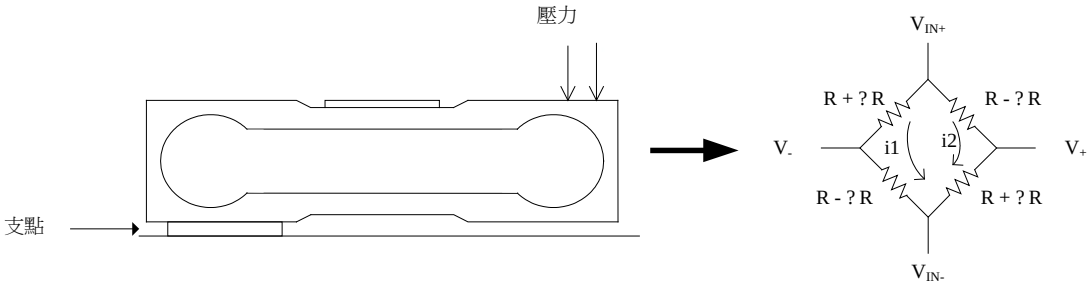
1.內容簡介

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　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HY16F198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　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。

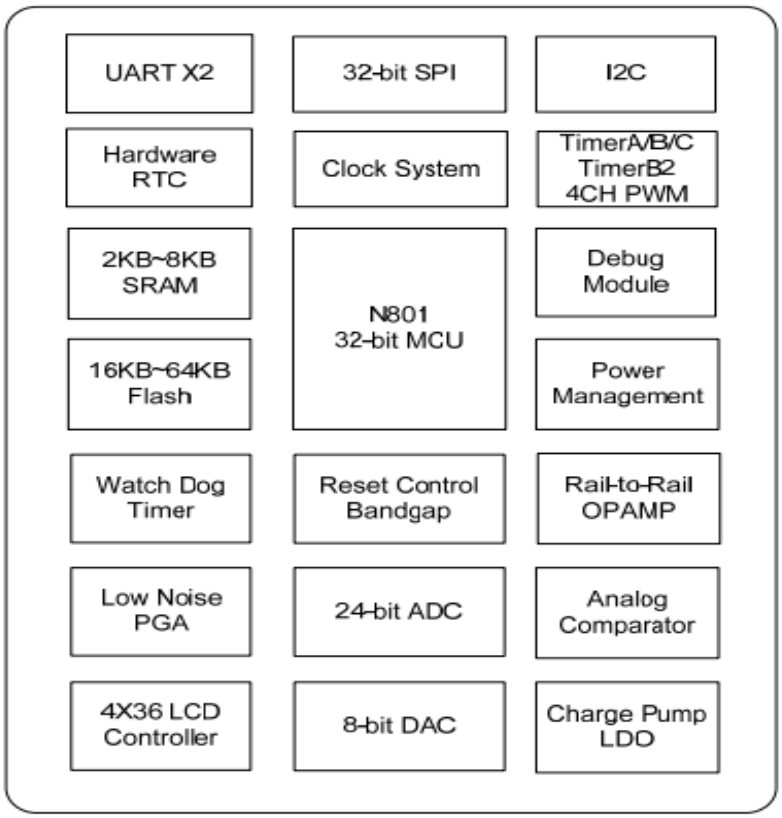
內外解析度之比越小，電子秤精度越高，但內外解析度之比是有限制的。比如 Load Cell 滿量程壓差為 3mV，要做到 3000 Count，內外比為 1：10 的電子秤，如果不經過信號放大，那最小要處理的信號為 3mV/(3000X10)=0.1μV。而 SD24 所能處理的最小信號值大約為 65nV，所以假如內外比再減小的話將產生使 ADC 不能識別的信號。如果使用 OPAMP 的話則會增加成本。所以內外解析度之比要穩定在一定範圍內。

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 HY16F198 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。 HY16F198 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 位元元高性能 Flash 單片機(HY16F198)



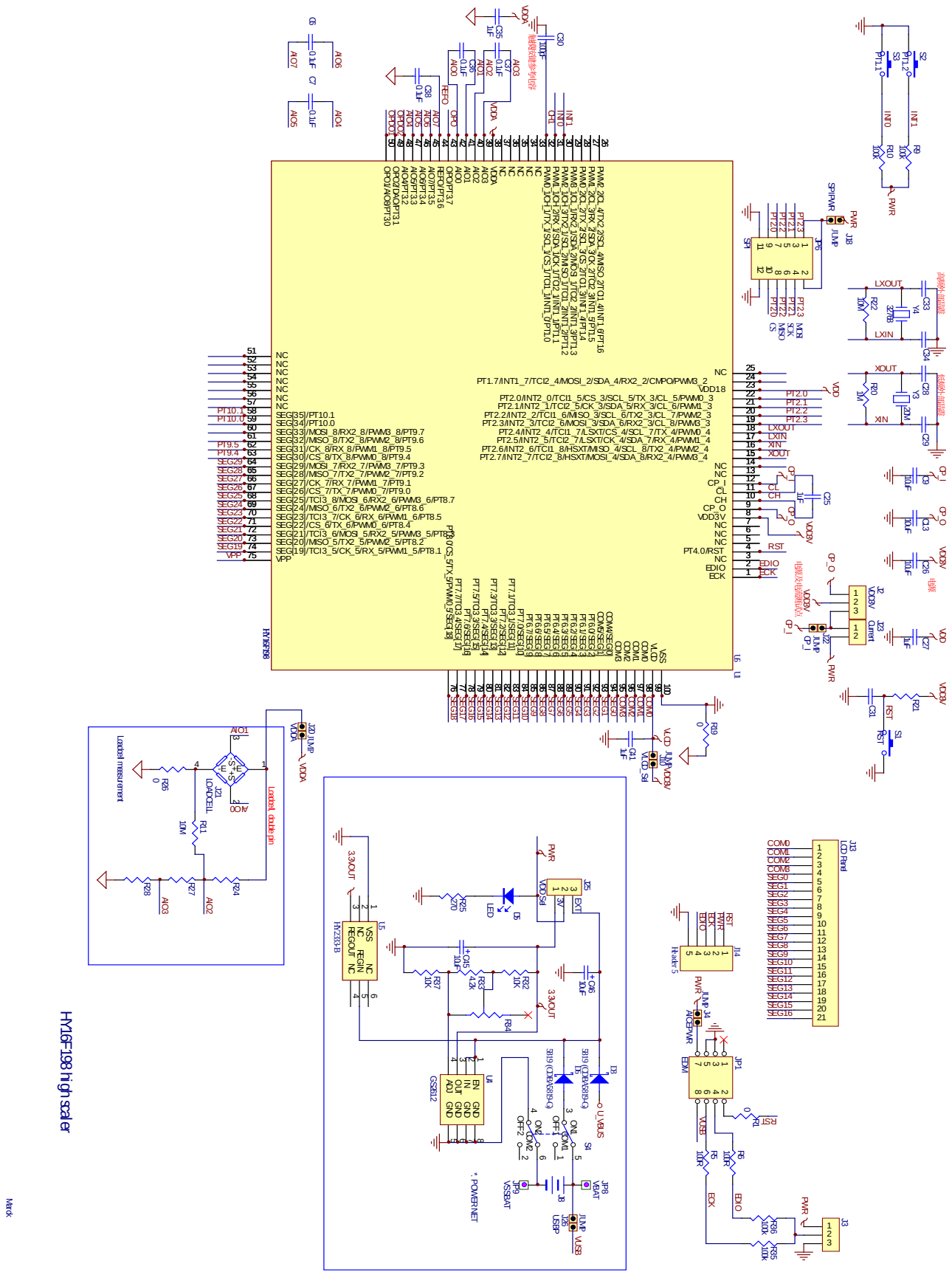
紘康 HY16F 系列 32 位元元高性能 Flash 單片機(HY16F198)

(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 16MHz 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　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/ Timer B2 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/UART2 模組
(15)硬體 RTC 時鐘功能模組
(16)硬體 Touch KEY 功能模組
(17)硬體 LCD Driver 4 X 36, 6X34

3.系統設計

3.1 硬體說明

HY16F198 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：
HY16F198 (Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + UMC 64K Flash)
(B)電源電路：5.0V 轉 3.3V 電源系統
(C)　比感測模組：內部 ADC
(D)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 平臺 IDE 以及 HYCON　比軟體分析工具與 GUI 等支援。

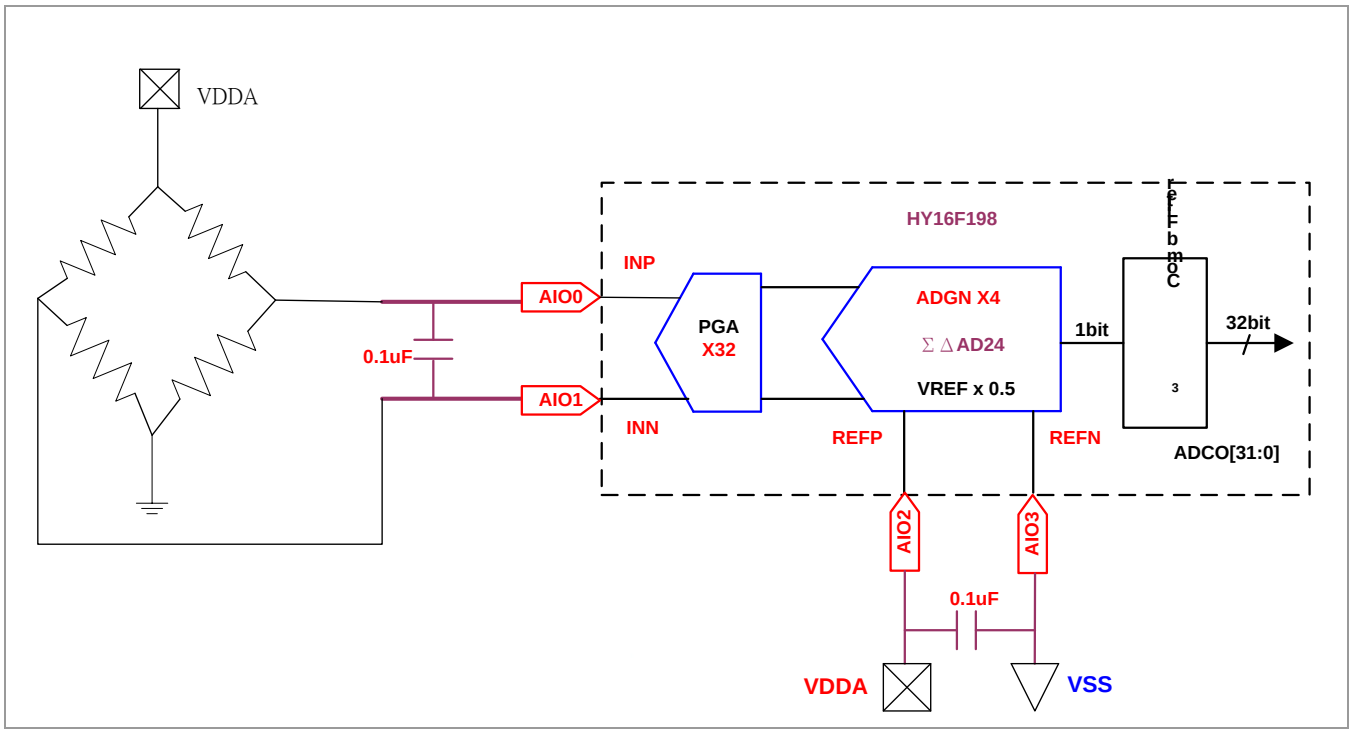


應用電路原理圖

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。

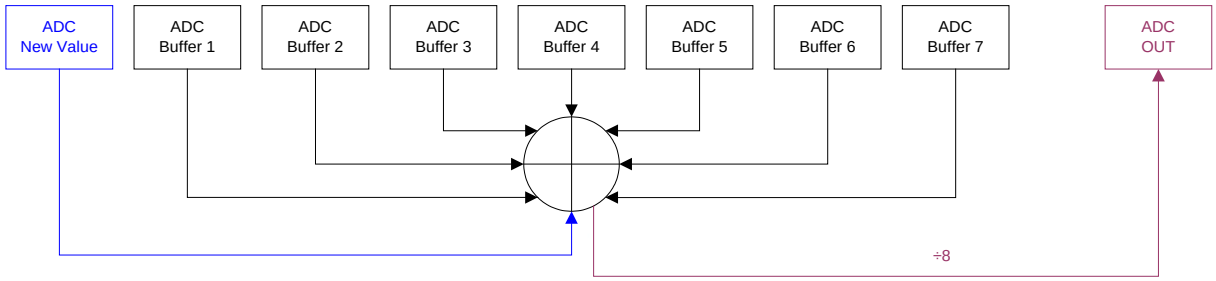


3.2 軟體說明

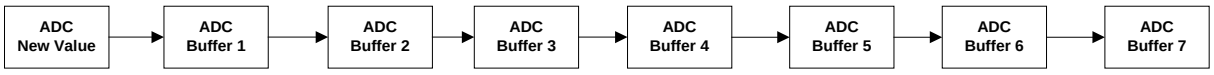
ADC 資料處理

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。



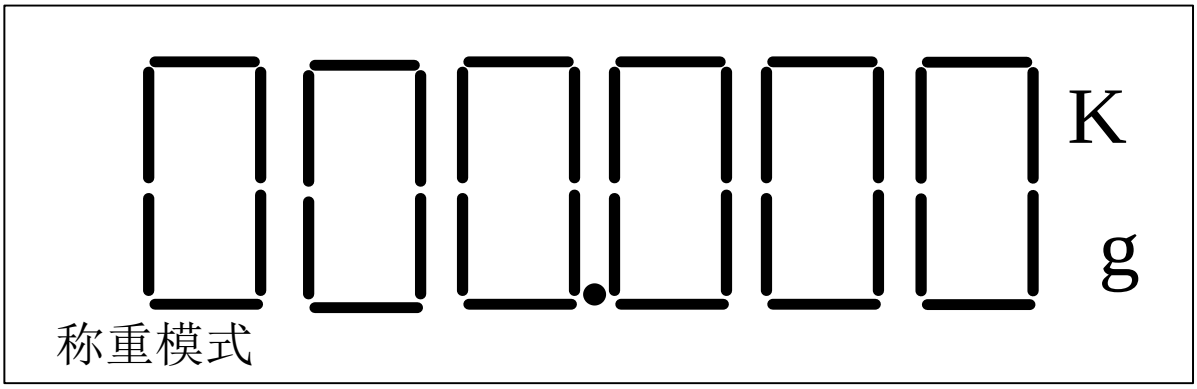
4.操作流程

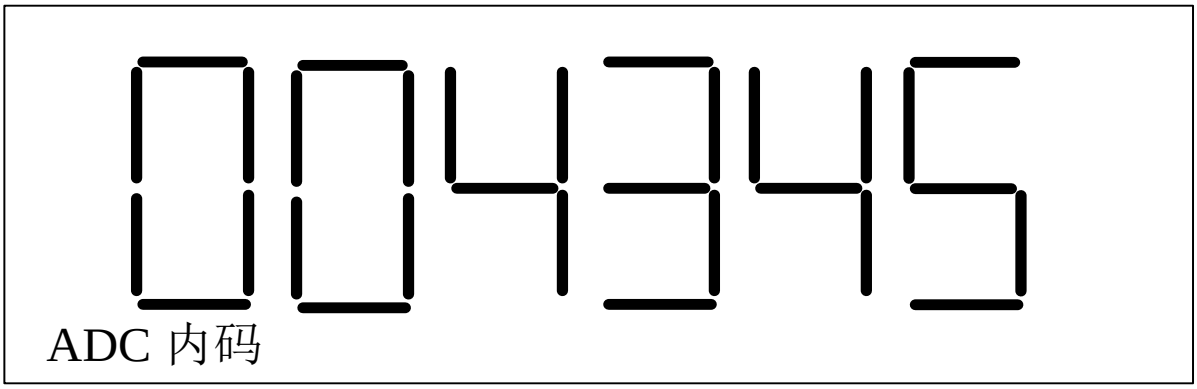
4.1 操作方法

上電啓動後，顯示”Hycon”字樣，然後自動抓取零點 ADC 值，若 Flash ROM 內已有校正值，讀取 Flash 存儲的校正值及 ADC 位數、解析度、最大測量範圍、及校正標準重量值，均完成後自動進入量測模式；否則進入校正模式，開始相關數據校正。

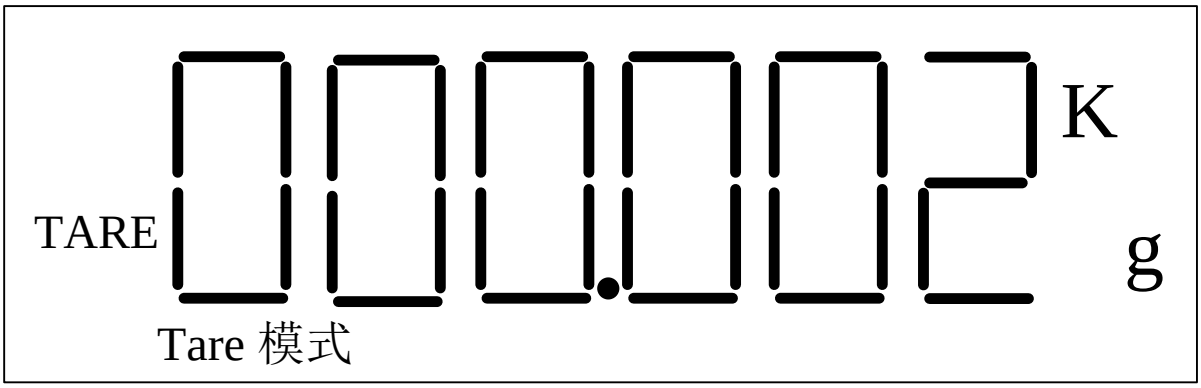
量測模式:

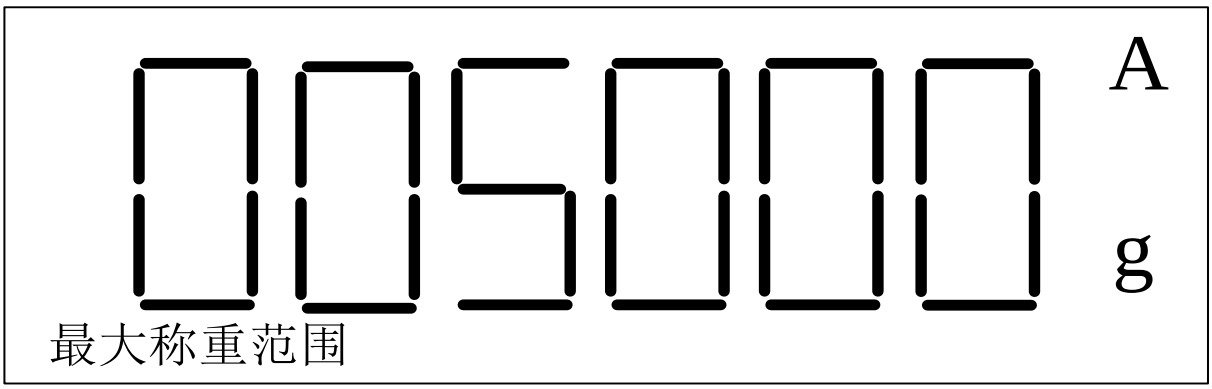
按 PT1.1 按鍵會產生歸零（當前重量小於最大重量的 10%）或去皮（當前重量大於最大重量的 10%）功能；按 PT1.2 按鍵會切換顯示重量或者 ADC 內碼；





啟動去皮功能後，會顯示 TARE 符號；進入校正模式會在右上角顯示 ‘A’字元。

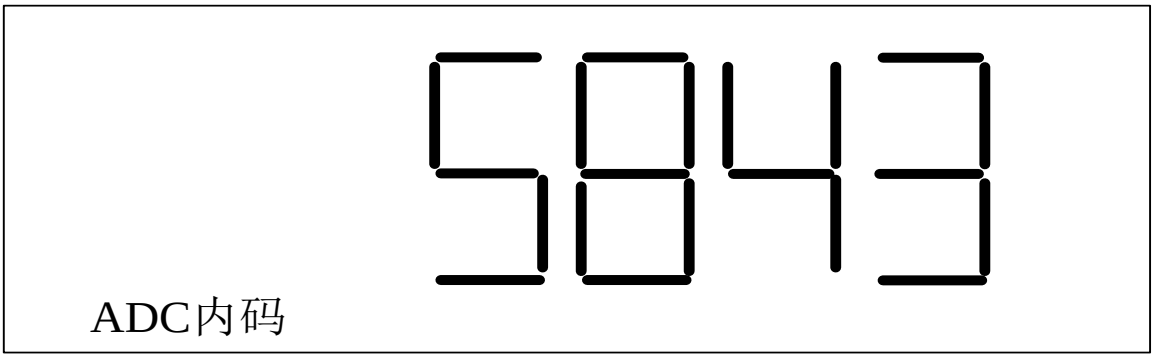




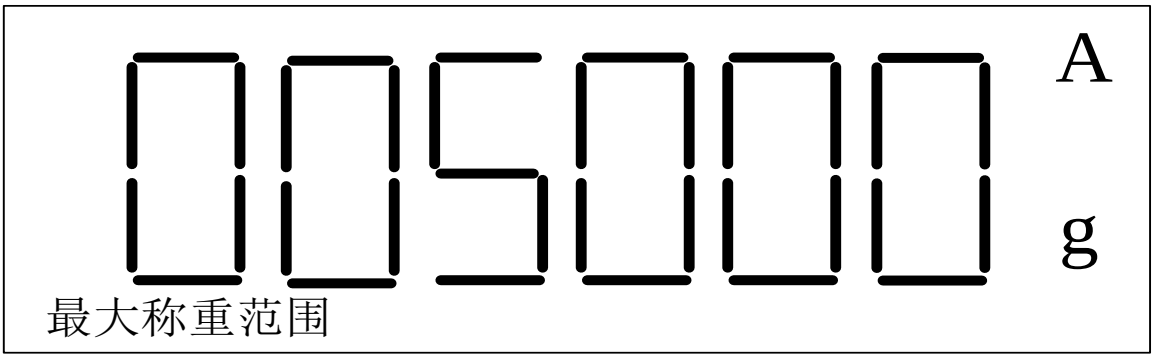
校正模式:

開機上電，判斷 Flash 不存在校正值，則自動進入校正模式；或者上電前，已先按 PT1.2，則上電後，自動進入校正模式。進入校正模式首先顯示的最大測量範圍值；在校正模式下，設置重量只顯示 g 單位，小數點及解析度都顯示相關單位。校正模式下，長按 PT1.1 按鍵切換不同的設置項目，按 PT1.1/PT1.2 按鍵具有遞減/遞增功能。

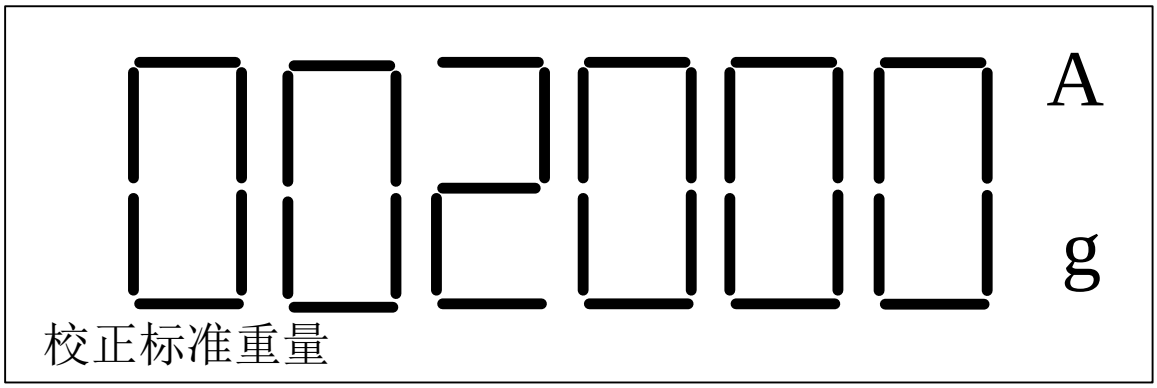
進入校正模式第一項顯示 ADC 內碼，長按 PT1.1 按鍵進入下一個項目設置;



第二項目為最大測量範圍:預設為 5000g，右上角顯示 ‘A’字元；按 PT1.1/PT1.2 按鍵可以 500 步長遞減/遞增，循環從 0 至 9500；完成後長按 PT1.1 按鍵進入下一個項目設置。

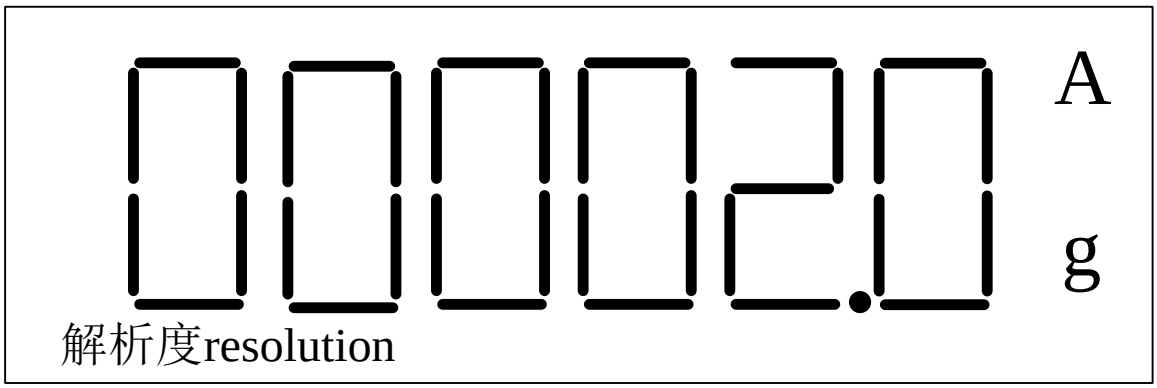


第三項目為校正標準重量:設置校正標準重量值；預設為 2000g，透過 PT1.1/PT1.2 按鍵可以 500 步長逐步遞減/遞增，循環從 0 至 9500；設置完後長按 PT1.1 切換進入下一個設置項目。

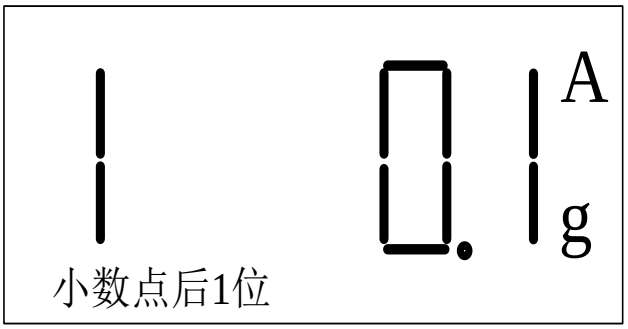


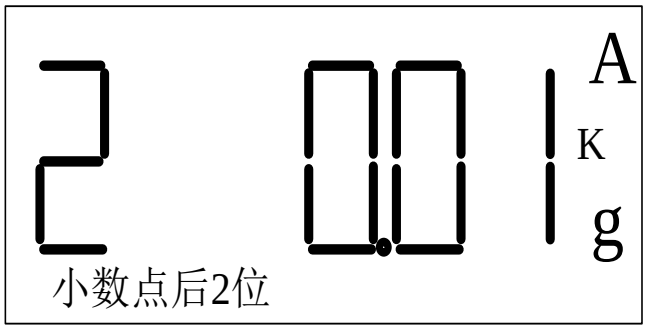
第四個項目為解析度：設置顯示最小刻度 resolution；預設為 1g；透過 PT1.1/PT1.2 按鍵可以步長 0.5g 逐步遞減/遞增，循環從 0 至 2；設置完成後長按 PT1.1 切換進入下一個設

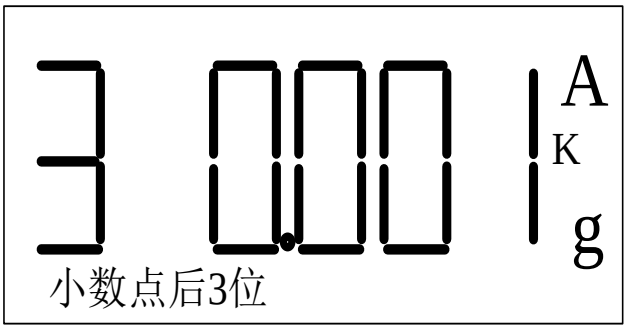
置項目。



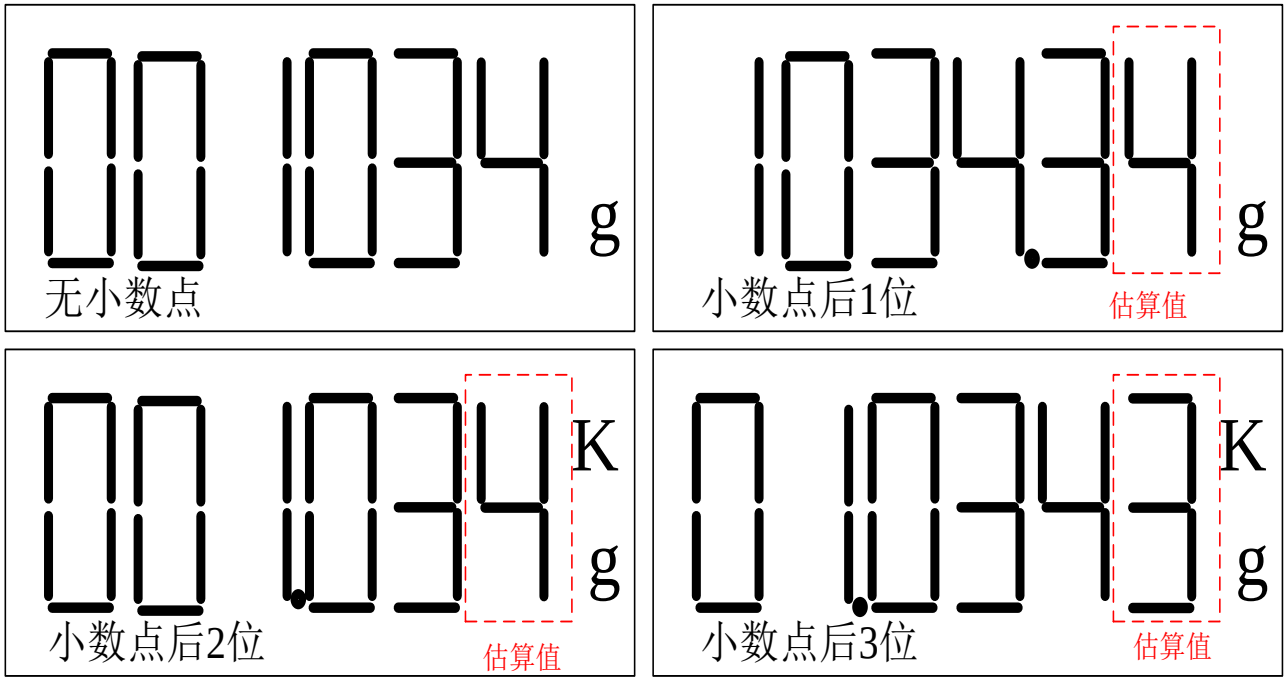
第五個項目為小數點：設置小數點位置，預設為 3；範圍為 0~3，分別表示小數點的位置為無小數點，小數點後 1 位，小數點後 2 位，小數點後 3 位；設置不同的小數點位置，有不同的顯示模式，設置為 1~3 時，稱重時顯示重量會在小數點後多添加一位數；設置為 0 時就沒有。透過按 PT1.1/PT1.2 可以遞減/遞增切換不同的值，選擇不同的小數點位置；設置完成長按 PT1.1 切換至下一個項目。



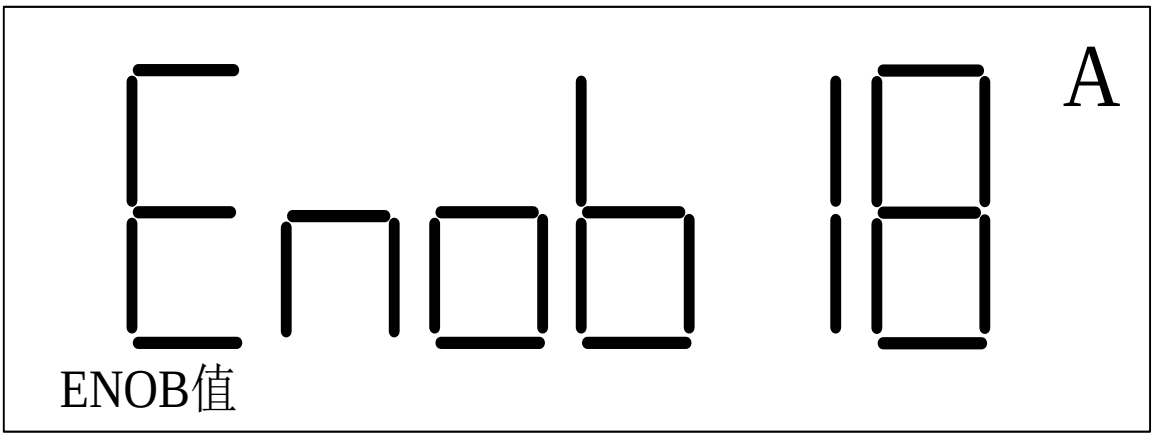




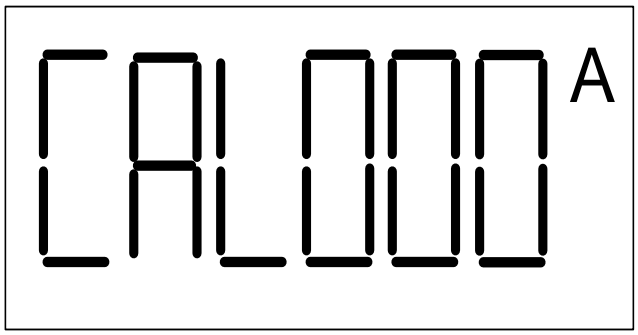
設置不同小數點位置，在稱重模式下，就會有不同的顯示模式：

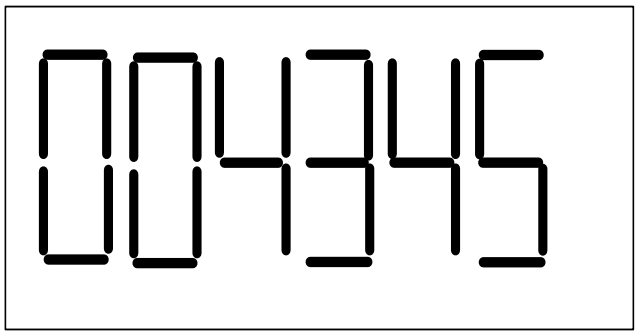


第六個項目為 ENOB：設置保留 ADC 的有效位數（ENOB）值，預設為 18，設置範圍為 1~24；可根據實際需求設置不同 ENOB 值；顯示模式如下圖所示；透過 PT1.1/PT1.2 以步長 1 遞減/遞增；設置完成後長按 PT1.1 切換至下一個項目。

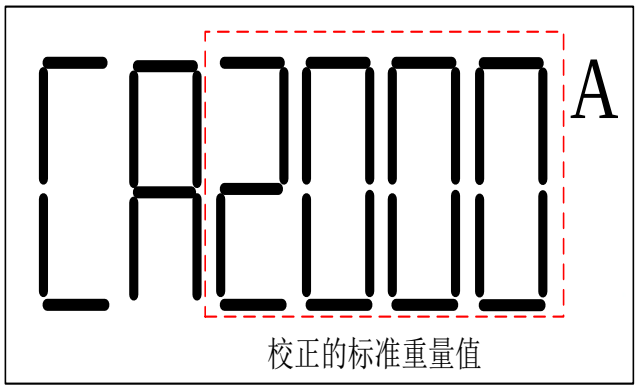


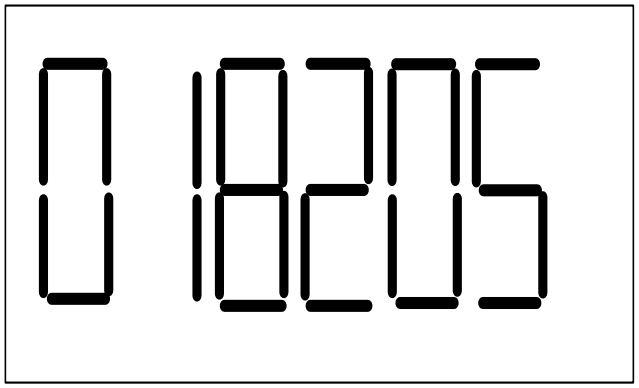
第七個項目校正零點：校正零點 ADC 值；進入到此模式下，顯示為'CAL000'，提示為校正零點值，不能放置重物，以免影響校正。然後長按 PT1.1 就開始進入零點校正，校正過程顯示 ADC 值；校正成功後自動進入標準重量校正模式。



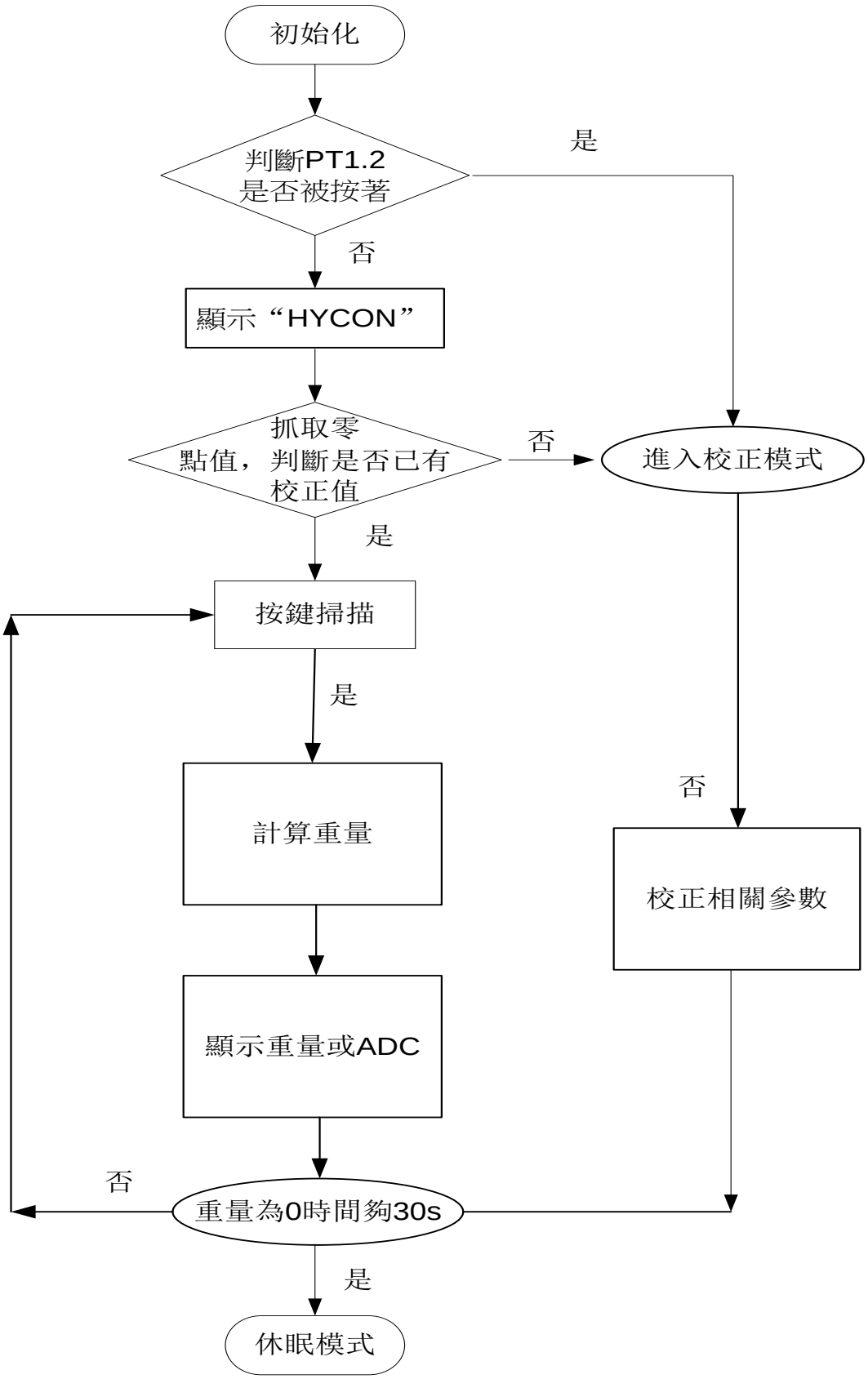


第八個項目校正標準重量：校正標準重量 ADC 值；進入此模式，顯示為 ‘CAXXXX’，’XXXX’表示校正重量；放置待校正的標準重量後，長按 PT1.1 就會開啟標準重量校正;校正過程顯示 ADC 值； 校正成功後，自動重跑程式，倒數 9~1 顯示，再顯示 HYCON 之後進入秤重模式。

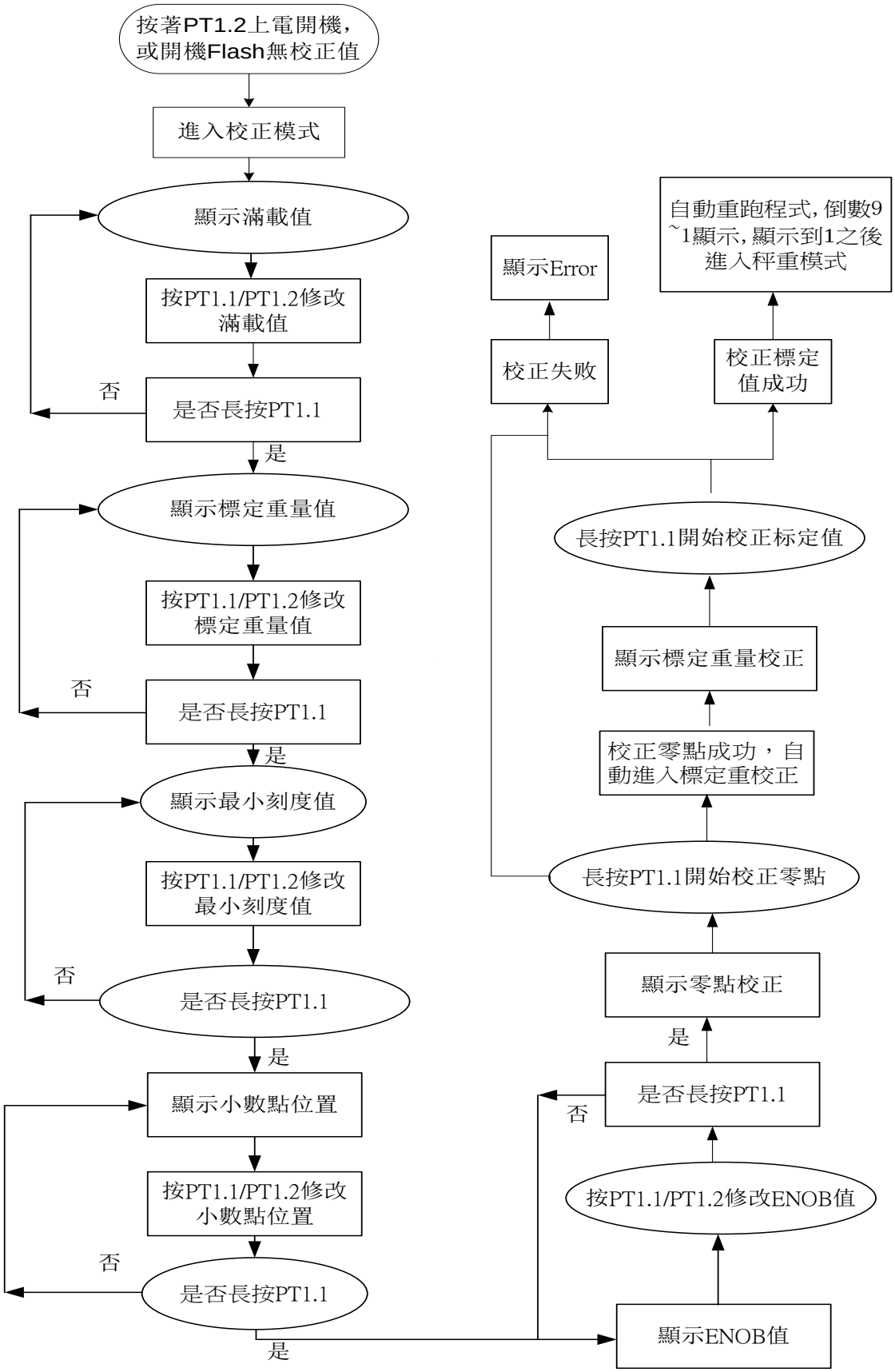




4.2 主程式流程


初始化
判斷PT1.2
是否被按著
是
否
顯示“HYCON”
抓取零
點值，判斷是否已有
校正值
否
進入校正模式
是
按鍵掃描
是
計算重量
否
校正相關參數
顯示重量或ADC
否
重量為0時間夠30s
是
休眠模式

4.3 校正副程式流程


按著PT1.2上電開機，
或開機Flash無校正值
進入校正模式
顯示滿載值
按PT1.1/PT1.2修改
滿載值
否
是否長按PT1.1
是
顯示標定重量值
按PT1.1/PT1.2修改
標定重量值
否
是否長按PT1.1
是
顯示最小刻度值
按PT1.1/PT1.2修改
最小刻度值
否
是否長按PT1.1
是
顯示小數點位置
按PT1.1/PT1.2修改
小數點位置
否
是否長按PT1.1
是
顯示ENOB值
按PT1.1/PT1.2修改ENOB值
是否長按PT1.1
否
是
顯示零點校正
長按PT1.1開始校正零點
校正零點成功，自
動進入標定重校正
顯示標定重量校正
長按PT1.1開始校正标定值
校正失敗
校正標定
值成功
顯示Error
自動重跑程式，倒數9
~1顯示，顯示到1之後
進入秤重模式

5.技術規格

(1)電源接點：3.0V
(2)功耗：工作模式總消耗電流 4.31mA
晶片耗耗電流 2.13mA
Load Cel 消耗電流 2.18mA(內阻 1000Ω)
(3)適用範圍：各種秤
(4)內外碼比：6.8:1
ADC有效位數為18bit，1000g對應6890count ADC，因此6890/1000=6.8count/1g=6.8:1；
(5)解析度： 0.2g
ADC有效位數為18bit，6.8count跳1g，因為最小刻度為1g/6.8=0.15，做0.2g是比較穩定。
(6)精密度： 1/25000
ADC有效位數為18bit，解析度為0.2g，滿量程為5000g，精密度=0.2/5000=1/25000
(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/單位g	測得重量/單位g	ADC內碼（18bit）/count
0	0	4424
1	1	4431
2	2	4438
3	3	4446
4	4	4452
5	5	4459
10	10	4493
100	100	5113
1000	1000	11314
2000	2000	18205
3000	3000	25096
4000	4000	31988
5000	5000	38877

7.結果總結

以 HY16F198 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 HY16F198 來進行開發及應用。

8. 加檔案


HY16F198_Kitchen_Scales_V03.zip

9.參考文獻

(1)HYCON HY16F198 Series Data Sheet
(2)HYCON HY16F198 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
V01	ALL	初版發行	2015/1/16
V02	ALL	更改 PT1.1/PT1.2 定義，更新 DEMO 程式；更新相關圖片；	2015/6/11
V03	ALL	更改無校驗值時顯示內碼，按鍵進入校正；更正校正解析度 2.0 不顯示的 Bug；更改稱重不穩定	2016/2/2

件:範常程式

```
/*******************************************************************************
 * HY198_HIGH_SCALER
 * ------------------------------------------------------------------------
 * function:  高精度   房秤，可以   置 ADC 的有效位   、   重精度、校正重量。
 * author:              MARCK
 * create date:     2014-12-09
 * Library release V0.4
 * Copyright 2014 HYCON Technology
 * http://www.hycontek.com/
 * note: key:PT1.1/PT1.2; ADC INPUT: AI0/AI1     VREF: AI2/AI3
 * modify:chang PT1.1/PT1.2 for the calibrate methods
 *
 *******************************************************************************/
/*-------------------------------------------------------------------------*/
/* Includes                                                                */
/*-------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvCLOCK.h"
#include "DrvPMU.h"
#include "DrvADC.h"
#include "DrvTIMER.h"
#include "specialmacro.h"
#include "drvflash.h"

#include "my define.h"
#include "Display.h"
#include "key.h"
#include "adc_deal.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 int adc_value;
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void data_initial(void);
void sys_initial(void);
void scale_sleep(void);
void Displaycal(void);
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
 // unsigned int m=0;
 // unsigned int z=0;
   sys_initial();
   data_initial();
   DisplayInit();
   ClearLCDframe();

 // cal_stander_adc=20000;
 // cal_zero_adc=16;

//read calibration information
   SYS_DisableGIE();       //DISABLE GOLABLE INTERRUPT
   cal_summer=ReadWord(save_addr);
   if((cal_summer!=0)&&(cal_summer!=0xffffffff))    //read information from the flash; not the first time;
   {
                              cal_summer=ReadWord(save_addr);                    //read PT1.2
                              if((cal_summer!=0)&&(cal_summer!=0xffffffff))
        {
                              //get the 'resolution'
                              resolution=ReadWord(save_addr+0x04);
                              //get the 'adc_enob'
                              adc_enob=ReadWord(save_addr+0x8);
                              //get the 'decimal_piont'
                              decimal_piont=ReadWord(save_addr+0xc);
                              //get the 'full_range'
                              full_range=ReadWord(save_addr+0x10);
                              //get the "stander_wg'
                              stander_wg=ReadWord(save_addr+0x14);
                              //get the 'cal_stander_adc'
                              cal_stander_adc=ReadWord(save_addr+0x18);//*/
                              //get the 'cal_zero_adc"
                              cal_zero_adc=ReadWord(save_addr+0x1c);
        }
                           }
  else
  {
                              cal_flag=0x55;              //get into the calibration mode;
                              cal_summer=0;
                              cal_zero_ok=0;                    //zero  校正    志
                              cal_stander_ok=0;
```

```
                        adstbcnt=8;
                        adstbok=0;
  }
  SYS_EnableGIE(0X7,0X1FF);
  adc_zero=cal_zero_adc;
  TraceWeight=(float)cal_stander_adc/(stander_wg);
  Tracezero= TraceWeight/2;

  if(DrvGPIO_GetBit(E_PT1,2)==0)              //press pt1.2 begin into calibrate
                        {
                         //      error_num=0;
                          adstbcnt=16;
                          adstbok=0;
                             cal_zero_ok=0;              //zero 校正   志
                             cal_stander_ok=0;
                          if(cal_flag==0)
                                cal_flag=0x55;
                          else
                                cal_flag=0;
                          Displaycal();
                          while(DrvGPIO_GetBit(E_PT1,2)==0);
                          while(cal_flag!=0)
                          {
                                key_reaction();
                                adc_calibrate();
                                LCD_Cal_Display();
                          }
                        }
  adc_zero=cal_zero_adc;
  TraceWeight=(float)cal_stander_adc/(stander_wg);
  Tracezero= TraceWeight/2;

  DisplayHYcon();                   //显示“HYCON”
  Delay(50000);
  //==============取零   ====================
                        adstbcnt=16;
                        adstbok=0;
                          while(adstbok==0)
                          {
                            int j=0;
                            while(ad_bok==0);
                            adc_smooth_filter();
                            while(j<30000)
                                {
                                      j++;
                                }
                                j=0;
                          }
                          adc_zero=adstable;
                          zerobuf=adc_zero;
                        adstbcnt=8;
                        adstbok=0;
                        adsum=0;

//*****************************************************************************
  time_counter=0;
  SYS_EnableGIE(0X7,0X1FF);       //ENABLE THE GOLABLE INTERRUPT
```

```
  while(1)
  {
                        key_reaction();
                        adc_smooth_filter();
                        if(cal_flag==0)              //normal measure weight mode
                        {
                          calculate_wg();
                          if(sys_flag0&0x8)
                          {
                                LCD_DATA_DISPLAY(disweight);
                          }
                          else
                          {
                                time_counter=0;
                                LCD_ADC_Display(adcout);
                          }

                        }

                        else if(cal_flag!=0)    //get into calibration
                        {
                          adc_calibrate();
                          LCD_Cal_Display();
                        }

                        if(time_counter>7500)
                        {//get into the low power mode
                          ClearLCDframe();
                          scale_sleep();
                             adstbcnt=16;
                             adstbok=0;
                                while(adstbok==0)
                                {
                                      int j=0;
                                      while(ad_bok==0);
                                      adc_smooth_filter();
                                      while(j<30000)
                                          {
                                                j++;
                                          }
                                          j=0;
                                }
                                adc_zero=adstable;
                                zerobuf=adc_zero;
                             adstbcnt=8;
                             adstbok=0;
                             adsum=0;
                          time_counter=0;
                        }
  }

  while(1);
  return 0;
}

/*---------------------------------------------------------------------------*/
/* Hardware Communication Interrupt                                          */
```

```
/* UART/SPI/I2C Interrupt Service Routines                                    */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* WDT & RTC & Timer A/B/C Interrupt Service Routines                         */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
    if(DrvTIMER_GetIntFlag(E_TMA))
    {
                DrvTIMER_ClearIntFlag(E_TMA);    //clear timer A interrupt flag
                key_delay_time++;
                long_key++;
                time_counter++;
    }
}

/*---------------------------------------------------------------------------*/
/* HW2 ADC Interrupt Subroutines                                             */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{
                if(DrvADC_ReadIntFlag())                    //check ADC interrupt flag
                {
                  DrvADC_ClearIntFlag();          //clear the ADC interrupt flag
                  adc_value=DrvADC_GetConversionData()>>8;    //get ADC convert data from
32bit to 24bit
                  sys_flag0=sys_flag0^0x02;                   //set the ADC convert
success flag
                  ad_bok=1;
                  adsum++;
                }
}

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

}

/*---------------------------------------------------------------------------*/
/* PT1 Interrupt Service Routines                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{
                DrvGPIO_ClearIntFlag(E_PT1,0X06);          //clear the interrupt flag;
}

/*---------------------------------------------------------------------------*/
/* PT2 Interrupt Service Routines                                            */
/*---------------------------------------------------------------------------*/
```

```
void HW5_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
      asm("NOP");
}
/*****************************************************************************
 * name    : sys_initial()
 * func    : initial the system
 * input : no
 * output: no
 * date    : 2014-12-11
 * modify: no
 * ***************************************************************************/
void sys_initial(void)
{
                        //setting IC CPU clock
                        DrvCLOCK_EnableHighOSC(E_INTERNAL,50);          // enable HAO
                        DrvCLOCK_SelectIHOSC(E_2MHZ);                   // Select HAO 2MHz
                        DrvCLOCK_SelectMCUClock(0,0);
                        DrvCLOCK_TrimHAO(0x79);                         //calibration the HAO for
4MHZ;

    //setting the    IO port
                        DrvGPIO_ClkGenerator(1,1);
                        DrvGPIO_Close(E_PT1,0X06,E_IO_OUTPUT);    //DISABLE PT1.1 PT1.2
OUTPUT MODE
                        DrvGPIO_Open(E_PT1,0X06,E_IO_INPUT);     //SET PT1.1 PT1.2 AS INPUT
FOR KEY
                        DrvGPIO_Open(E_PT1,0X06,E_IO_PullHigh); //enable PT1.1 PT1.2 PULL
HIGH;

                        //set power manage
                        DrvPMU_VDDA_LDO_Ctrl(E_LDO);              //choose VDDA source for LDO
mode
                        DrvPMU_VDDA_Voltage(E_VDDA2_4);           //set VDDA=2.4V
                        DrvPMU_LDO_LowPower(1);                   //VDDA LDO WORK FOR LOW
POWER
                        DrvPMU_BandgapEnable();                   //ENABLE BANDGAP VOLTAGE
                        DrvPMU_AnalogGround(1);                   //SET ANALOG GROUND FOR
INERNAL

                        Delay(100);

                        //setting ADC
                        DrvADC_ClkEnable(0,1);              //set ADC clock source HS_CK/6, ADC
Clock rising Edge at clock high;
                        DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);    //set
SIGNAL input: AI0--- AI1
                        DrvADC_RefVoltage(AIO2,AIO3);     //VREF INPUT AIO2--AIO3
```

```
                    DrvADC_FullRefRange(1);             //VREF/2
                    DrvADC_Gain(7,3);                   //PGA=32,ADGN=4;
                    DrvADC_OSR(0);                       //OSR=0/32768,10SPS
                    DrvADC_Enable();                    //enable ADC
                    DrvADC_CombFilter(1);               //setting ADC comb filter auto_drop    the
first 3 data
                    DrvADC_CombFilter(0);
                    DrvADC_CombFilter(1);

                    //setting timer A
                    DrvTMA_Open(7,E_HS_CK);          //set timer A clock source and over counter
                    DrvTMA_ClearTMA();                //clear the timer A counter
                    DrvTIMER_ClearIntFlag(E_TMA);    //clear the Timer A interrupt flag
                    DrvTIMER_EnableInt(E_TMA);        //enable the Timer A interrupt

                    //setting all interrupt
                    DrvADC_ClearIntFlag();              //clear the ADC interrupt flag
                    DrvADC_EnableInt();                // enable the ADC interrupt

                    SYS_EnableGIE(7,0X3F);     //enable all interrupt
}
void data_initial(void)
{
                    sys_flag0=0X08;
                    cal_flag=0;
                    cal_summer=0;
                    time_counter=0;
                      Trg=0;
                      Cont=0;
                      key_counter=0;
                      long_key=0;
                      key_delay_time=0;
                      adsum=0;
                      ad_bok=0;
                      cal_zero_ok=0;                    //zero  校正    志
                      cal_stander_ok=0;          //stander  校正    志

                      adc_value=0;
                      adstable=0;
                      adcout=0;
                      TraceADCbuf=0;
                      adc_avg=0;
                      adc_zero=0;

                      adc_avg=0;            //ADC average
                      adc_zero=4200;          //zero ADC value
                      adstbcnt=8;           //判     AD     定的次
                      adsign=0;              //符    判
                      adsignb=0;             //符    判
                      ADOK=0;                //AD     理完成    志      0：AD 未完成，    1：完成

                     cal_stander_adc=13474;
                     cal_zero_adc=10;
                     full_range=5000;
                     stander_wg=2000;

                     zero_num=0;    //weight is 0 counter
```

```
                        def_num=0;
                        adc_enob=18;    //ENOB

                        weight=0;

                        tare_wg=0;
                        resolution=10; //the reslution of weight,   置重量最小刻度，如 2d，表示重量
化大于等于 2g 才跳   。
                        decimal_piont=0;
}

/************************************************************
 * name        : void scale_sleep(void)
 * function : system get into low power,wake up by the pt1.1/pt1.2
 * input       : no
 * output     : no
 * note        : no
 ************************************************************/
void scale_sleep(void)
{
                        unsigned int IO_buf;
                        //setting IO
                        DrvGPIO_Open(E_PT1,0Xf9,E_IO_INPUT);     //SET PT1.1 PT1.2 AS INPUT
FOR KEY
                        DrvGPIO_Open(E_PT1,0Xf9,E_IO_PullHigh); //enable PT1.1 PT1.2 PULL
HIGH;

                        DrvGPIO_Open(E_PT2,0Xff,E_IO_INPUT);     //SET PT1.1 PT1.2 AS INPUT
FOR KEY
                        DrvGPIO_Open(E_PT2,0Xff,E_IO_PullHigh); //enable PT1.1 PT1.2 PULL
HIGH;

                        DrvGPIO_Open(E_PT3,0Xff,E_IO_INPUT);     //SET PT1.1 PT1.2 AS INPUT
FOR KEY
                        DrvGPIO_Open(E_PT3,0Xff,E_IO_PullHigh); //enable PT1.1 PT1.2 PULL
HIGH;
    //setting LCD
                        DrvLCD_VLCDMode(0);
                        DrvLCD_LCDBuffer(0);
                        clk_10=0xFF00;    //LCD CLK=LS_CK
                        //setting timer A
                        DrvTIMER_DisableInt(E_TMA);         //disable the Timer A interrupt
                        DrvTMA_Close();                          //close timer A
                        DrvTMA_ClearTMA();                   //clear the timer A counter
                        DrvTIMER_ClearIntFlag(E_TMA);    //clear the Timer A interrupt flag

                        //disable ADC
                        DrvADC_ClkDisable();//disable ADC clock;
                        DrvADC_Disable();// disable ADC;
                        //disable power
                        DrvPMU_VDDA_LDO_Ctrl(E_HighZ);
                        DrvPMU_BandgapDisable();
                        //enable IO interrupt
                        DrvGPIO_IntTrigger(E_PT1,0X06,E_N_Edge);    //setting edge trigger
                        DrvGPIO_ClearIntFlag(E_PT1,0X06);              //clear the interrupt flag;
                        DrvGPIO_Open(E_PT1,0X06,E_IO_IntEnable);    // enable the IO interrupt
                        //change the IC clock
```

```
                        DrvCLOCK_EnableLowOSC(E_INTERNAL,50);
                        DrvCLOCK_SelectMCUClock(1,0);
                        DrvCLOCK_CloseIHOSC();
                        //get into low power mode
    asm("nop");
    asm("nop");
    SYS_LowPower (SYS_SleepMode);//(SYS_WaitMode);
    asm("nop");
    asm("nop");
    IO_buf=DrvGPIO_GetPortBits(E_PT1)&0X06;
    while(IO_buf^0x06)    //wait for free up the key
    {
                        IO_buf=DrvGPIO_GetPortBits(E_PT1)&0X06;
    }
                        //setting IC CPU clock
                        DrvCLOCK_EnableHighOSC(E_INTERNAL,1);           // enable HAO
                        DrvCLOCK_SelectMCUClock(0,0);
                        //disable IO interrupt
                        DrvGPIO_Close(E_PT1,0X06,E_IO_IntEnable);    // disable the IO interrupt
                        DrvGPIO_ClearIntFlag(E_PT1,0X06);             //clear the interrupt flag;
                        DrvGPIO_IntTrigger(E_PT1,0X06,00);    //setting edge trigger

    //setting power
                        DrvPMU_VDDA_LDO_Ctrl(E_LDO);                //choose VDDA source for LDO
mode
                        DrvPMU_VDDA_Voltage(E_VDDA2_4);           //set VDDA=2.4V
                        DrvPMU_BandgapEnable();                    //ENABLE BANDGAP VOLTAGE

                        //enable the ADC
                        DrvADC_ClkEnable(0,1);              //set ADC clock source HS_CK/12, ADC
Clock rising Edge at clock high;
                        DrvADC_Enable();                    //enable ADC

                        //enable the timer A
                        DrvTMA_Open(7,E_HS_CK);          //set timer A clock source and over counter
                        DrvTMA_ClearTMA();                //clear the timer A counter
                        DrvTIMER_ClearIntFlag(E_TMA);   //clear the Timer A interrupt flag
                        DrvTIMER_EnableInt(E_TMA);       //enable the Timer A interrupt

                        //setting LCD
    clk_10=0xFF16;   //LCD CLK=LS_CK
                        DrvLCD_VLCDMode(E_VLCD30);
                        DrvLCD_LCDBuffer(ENABLE);

                        SYS_EnableGIE(0X7,0X1FF);
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```