HYCON 紘康科技

HY16F198B

計價秤應用說明書

Pricing Scale Application Manual

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。本文所設計的計價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。

而由於計價秤需要做嚴格的認證，包括溫度、精確度與解析度，更需要具有抗干擾能力（ESD 與 EMI），這些除了周邊零件的特性需要達到規格要求之外，更需要量測晶片的規格能符合要求。因此如果能夠簡化周邊的被動元件（電阻），這將能夠降低溫度對量測的影響之外，更能夠節省成本，對抗干擾能力也有明顯的改善。

紘康 HY16F198B 控制晶片集成了高精度、高解析度、低溫漂的 24-bit 調變類比轉換器（ΣΔADC），並具有可編程增益（PGA）的低噪音放大器（和類比數位轉換器一起使用，最大增益為 128 倍放大倍率），而在電源管理方面提供可選擇的類比電路調節電壓，可做為電壓基準源與液晶驅動顯示（LCD）、傳感器的電源驅動等功能，故大幅簡化了 PCB 周邊線路，提高了電子秤應用的性價比。

2. 原理說明

2.1. 量測原理

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片(Strain gauge)，如圖 2-1 所示。

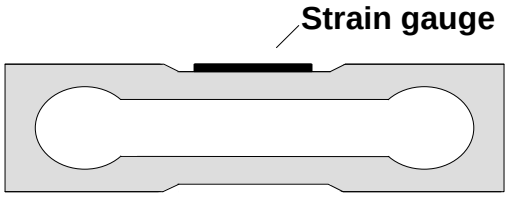


圖 2-1 應變片（Strain gauge）示意圖

因為電橋上的 4 個 R 電阻的阻值皆相同，所以當施加壓力讓鋁棒產生形變時將導致 Strain Gauge 的橋式電阻產生 ΔR 的變化量，如圖 2-2 所示。

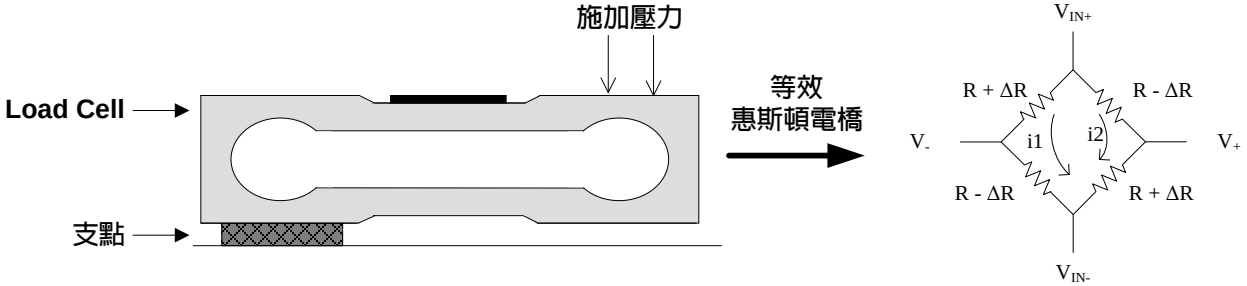


圖 2-2 Load Cell 施壓後產生 ΔR 變化之示意圖

此 ΔR 的變化量產生在訊號兩端的電壓變化為

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

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

因此我們可將此電壓變化的物理量經過 ADC 的轉換成為數位訊號，經由顯示器顯示出來。但是由於此電壓變化大致為 mV 等級的電壓訊號（因為 ΔR 的變化量遠小於 R），要做一個高精度的秤，處理的訊號將接近於 0.1μV，如果 ADC 的性能(Noise 的處理)無法達到要求，勢必要將電壓訊號再經過一級 OP 的放大，以達到精度要求。

經過 OP 放大所要處裡的因數就顯的複雜多了，除了 OP 本身的性能要求要能達到外，還要考慮外圍的電阻元件也要達到溫度變化的要求，此作法的成本相對性的要提高很多。又如果要將 Load Cell 輸出訊號直接轉換成數位訊號，那除了 ADC 本身的分辨率需要達到要求外，最小訊號的處理更要能小於 0.1μV 以下，這樣才能做出一個真正符合要求的秤。

Load Cell 的 R 大約 1KΩ，而 ΔR 的變化量最大也只有 1Ω，如果 VIN+-VIN-的電壓為 3V，輸出訊號 V+ - V-的電壓也只有 3mV；如果要做到 3000 Count，內外比為 1:10 的計價秤，那最小要處理的訊號為 $\frac{3mV}{3000 \times 10} = 0.1\mu V$

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。就一般我們以目視法認定的內部解析度通常是指我們經軟件處理後顯示只有 1 格滾動時，此時滿量程的格數就是其內部解析度，其 1 格所代表的訊號約為 2~3 倍 RMS Noise，而就需要認證的計價秤而言，內外解析度比值至少要達到 1：10。但就不須認證的電子秤通常為了顯示產品的最佳性能，通常也會盡可能提升外部解析度，降低內外解析度比值，但內外解析度比小於 1：3 時如果使用一般常規的軟件處理就不容易做到穩定的外部顯示。

HY16F198B 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(HY16F198B)

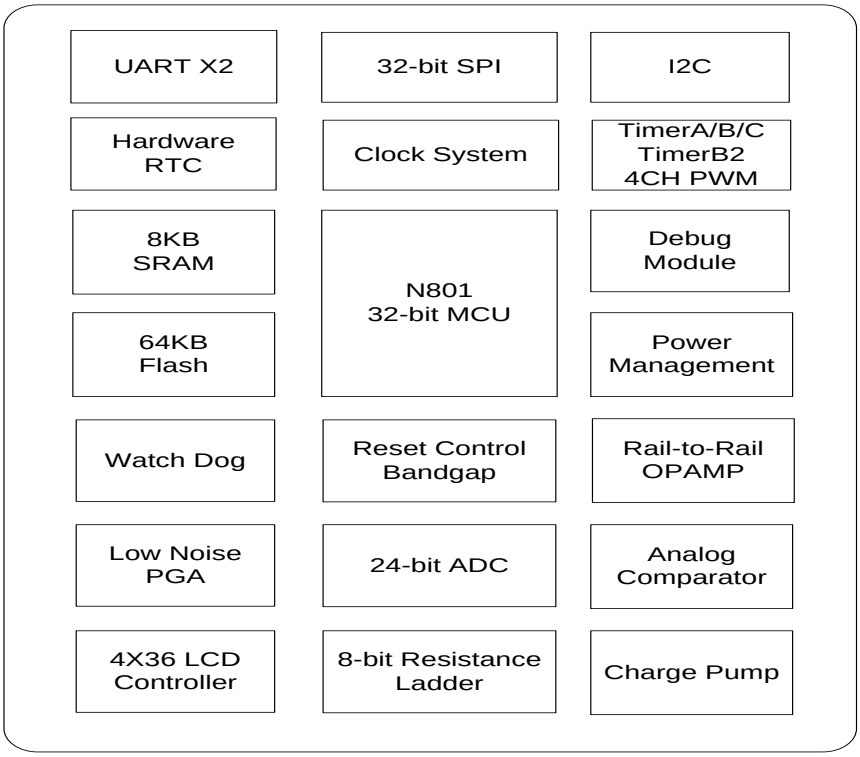


圖 2-3 HY16F198B 晶片功能架構圖

(1) 採用最新 Andes 32 位元 CPU 核心 N801 處理器。
(2) 電壓操作範圍 2.2~3.6V，以及-40℃~85℃工作溫度範圍。
(3) 支援外部 16MHz 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 350μA@2MHz/2
  (3.2) 待機模式 10μA@32KHz/2
  (3.3) 休眠模式 2.5μA(Sleep Mode), 5μA(Idle Mode)
(4) 程式記憶體 64KBytes Flash ROM，資料記憶體 8KBytes SRAM。
(5) 擁有 BOR and WDT 功能，可防止 CPU 死機。
(6) 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。
(7) 超低輸入雜訊運算放大器。
(8) 16-bit Timer A。
(9) 16-bit Timer B 模組具 PWM 波形產生功能。
(10) 16-bit Timer C 模組具 Capture/Compare 功能。
(11) 硬體的串列通訊 SPI 、I2C、UART 模組。
(12) 硬體 RTC 時鐘功能模組。
(13) 硬體 Touch KEY 功能模組。

3. 系統設計

3.1. 硬體說明

Load Cell 輸出的類比信號傳輸至 HY16F198B，MCU 通過本身的 ADC 轉換，採集 AD 信號值，經過運算處理得出對應的重量值，顯示到 LCD 上（共有 3 片 LCD），並可以通過 4X4 矩陣的按鍵輸入進行相關的設定及功能操作（每當按鍵被按下時蜂鳴器會發出一聲嗶聲當提示），而當超過 3 分鐘以上未使用時，可進入 Sleep 模式以減低功耗，整體應用架構如圖 3-1 所示。

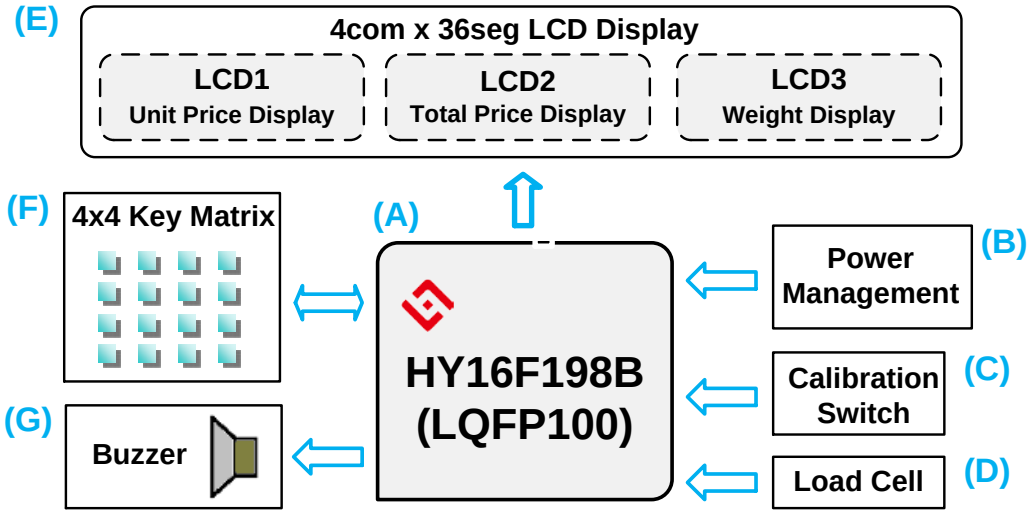


圖 3-1 HY16F198B 計價秤演示板之整體應用架構圖

此計價秤演示板是基於 HY16F198B 開發板來實現的，如圖 3-2 所示，相關架構說明如下：

(A) 中央處理器： HY16F198B，(Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + MCU 64KB Flash + 8 bit DAC + R2R OPAMP + LCD Driver)，功能為量測電信號、控制、運算及 LCD 驅動顯示。

(B) 電源電路：9V 轉 3.3V 電源供給系統。

(C) 校正開關：校正模式開關。

(D) 類比感測模組：壓力感測器（Load Cell）。

(E) LCD 模組：顯示操作的訊息及量測的結果，共分三個顯示區域（LCD1 顯示單價、LCD2 顯示總價、LCD3 顯示重量）。

(F) 4x4 矩陣鍵盤：功能操作之輸入按鍵。

(G) 蜂鳴器：用來發出 4KHz 的按鍵提示聲。

(H) EDM 接口：與 HY-Protool 仿真器連接的接口（仿真時才需要用到此接口）。

(I) Writer 接口:與 HY16F00-WK01 燒錄器連接的接口(燒錄 MCU 時才需要用到此接口)。

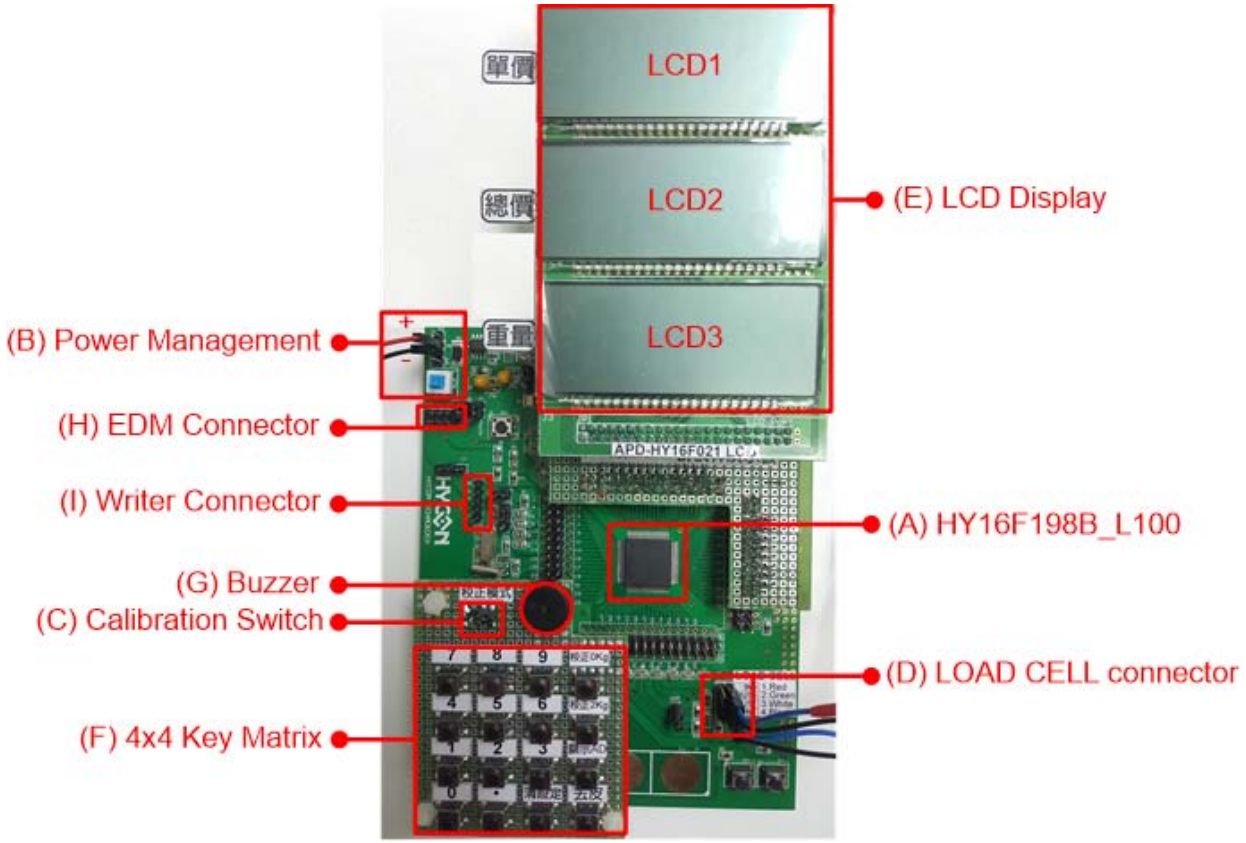


圖 3-2 HY16F198B 計價秤演示板外觀圖

HY16F198B 計價秤硬體連接電路如下:

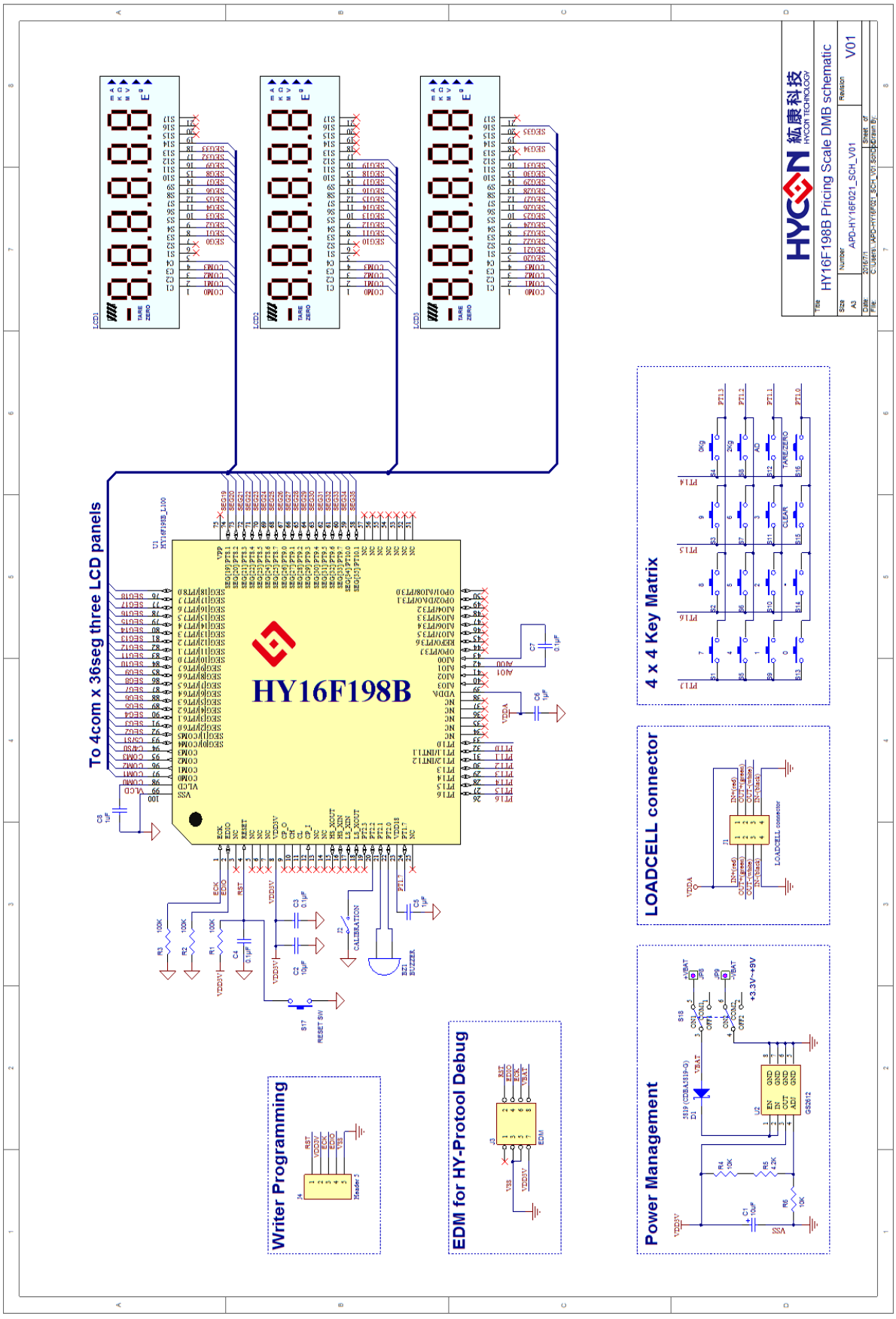


圖 3-3 HY16F198B 計價秤電路圖

3.2. 電路說明

3.2.1 ADC 電路說明

Load Cell 輸入電壓由內置穩壓器 VDDA 2.4V 輸出來提供，ADC 內部的 PGA 放大 32 倍，ADGN Gain 放大 4 倍，ADC 的參考電壓由 VDDA –VSS 供給 1.2V（將 FRb[0]設置[1]，則ΔVR_I=1.2V），由於 SD24 具有良好的溫漂特性，整體的溫度曲線約略為±10PPM，所以只要選擇低溫漂係數 Load Cell，就可以達到溫度漂移的要求。

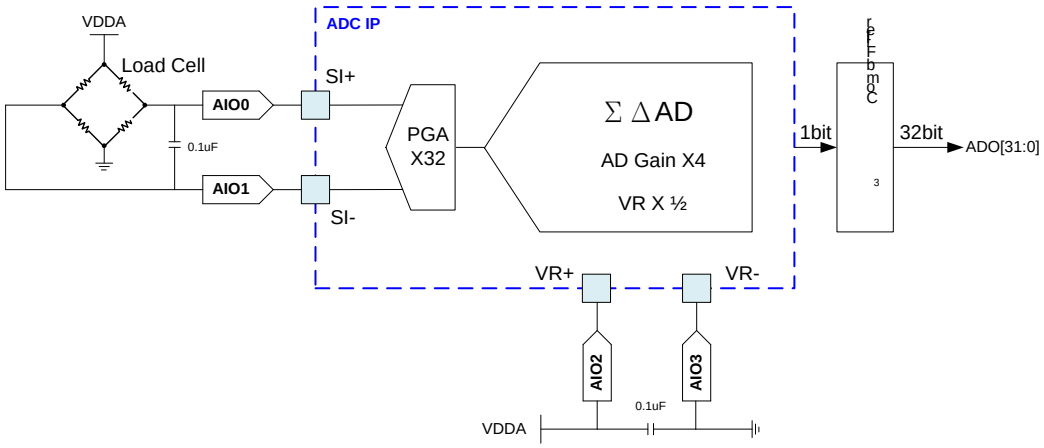


圖 3-4 ADC 電路示意圖

3.2.2 矩陣鍵盤說明

在矩陣式鍵盤中，每條水平線和垂直線在交叉處不直接連通，而是通過一個按鍵加以連接。這樣一個埠（如 PT1 埠共 8PIN）就可以構成 4X4=16 個按鍵，比之直接將埠線用於鍵盤多出了一倍，而且線數越多，區別越明顯，比如再多加一條線就可以構成 20 鍵的鍵盤，而直接用埠線則只能多出 1 鍵（9 鍵）。由此可見，在需要的鍵數比較多時，採用矩陣法來做鍵盤是合理的。

矩陣式結構的鍵盤顯然比直接法要複雜一些，識別也要複雜一些，列線通過 MCU 內部的上拉電阻接正電源，並將行線所接的 MCU 的 I/O 口作為輸出端，而列線所接的 I/O 口則作為輸入。這樣，當按鍵沒有按下時，所有的輸入端都是高電平，代表無鍵按下。行線輸出是低電平，一旦有鍵按下，則輸入線就會被拉低，這樣通過讀入輸入線的狀態就可得知是否有鍵按下。

矩陣按鍵識別方法：

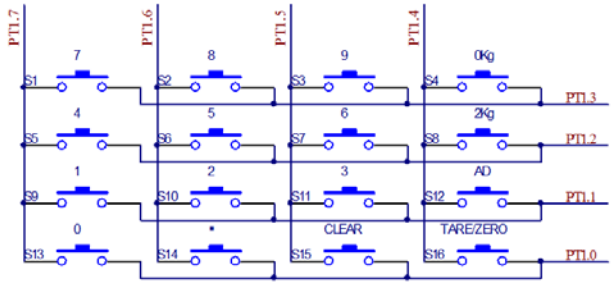


圖 3-5 4x4 矩陣鍵盤連接圖

此演示板是用行掃描法又稱為逐行（或列）掃描查詢法來識別鍵盤的，此法是一種最常用的按鍵識別方法，如上圖 3-5 所示，介紹過程如下：

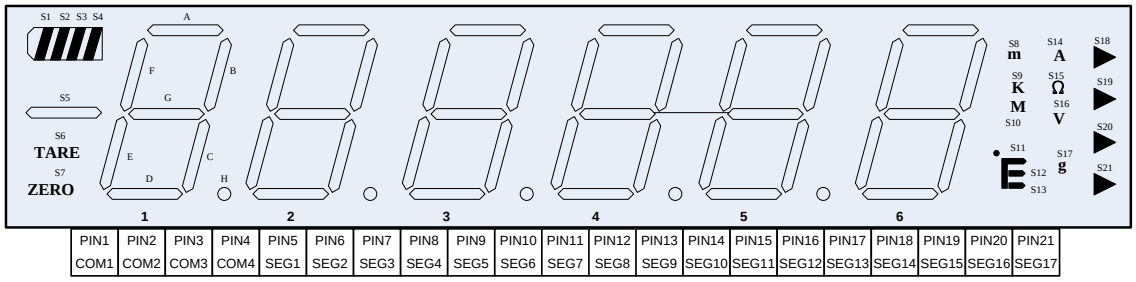
(1) 判斷鍵盤中有無鍵按下將全部列線 PT1[7:4]置低電平，然後檢測行線 PT1[3:0]的狀態。只要有一行的電平為低，則表示鍵盤中有鍵被按下，而且閉合的鍵位於低電平線與 4 根列線相交叉的 4 個按鍵之中。若所有行線 PT1[3:0]均為高電平，則鍵盤中無鍵按下。

(2) 判斷閉合鍵所在的位置在確認有鍵按下後，即可進入確定具體閉合鍵的過程。其方法是：依次將列線 PT1[7:4]置為低電平，即在置某根列線為低電平時，其它線為高電平。在確定某根列線位置為低電平後，再逐行檢測各行線 PT1[3:0]的電平狀態。若某行為低，則該行線與置為低電平的列線交叉處的按鍵就是閉合的按鍵。

3.2.3 LCD 顯示說明

HY16F198B 可直接驅動 4COM x 36SEG 共 144 點的 LCD 點數，而這 144 點分別由 LCD1、LCD2 及 LCD3 共 3 片的 HYCON 開發板用的標準 LCD 來均分顯示，而 HYCON 標準 LCD 的外觀及腳位如圖 3-6 所示，而其規格如下：

- ◇ Pin Type：90˚ 針腳
- ◇ Pin Pitch：2.54mm
- ◇ LCD Size：70.0mm(L)x27.0mm(W)
- ◇ VLCD：3.0V
- ◇ LCD Freq.：60Hz
- ◇ View Angle：60˚
- ◇ Bias：1/3bias
- ◇ Duty：1/4 duty



PIN1	PIN2	PIN3	PIN4	PIN5	PIN6	PIN7	PIN8	PIN9	PIN10	PIN11	PIN12	PIN13	PIN14	PIN15	PIN16	PIN17	PIN18	PIN19	PIN20	PIN21
COM1	COM2	COM3	COM4	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	SEG11	SEG12	SEG13	SEG14	SEG15	SEG16	SEG17

	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	SEG11	SEG12	SEG13	SEG14	SEG15	SEG16	SEG17
COM1	1F	1A	2A	2E	3A	3E	4A	4E	5A	5E	6A	6E	S1	S5	S10	S9	S18
COM2	1G	1B	2B	2F	3B	3F	4B	4F	5B	5F	6B	6F	S2	S6	S11	S14	S19
COM3	1E	1C	2C	2G	3C	3G	4C	4G	5C	5G	6C	6G	S3	S7	S12	S15	S20
COM4	1D	1H	2D	2H	3D	3H	4D	4H	5D	5H	6D	S17	S4	S8	S13	S16	S21

圖 3-6 HYCON 開發板用之標準 LCD 外觀及腳位圖

3.2.4 校正開關說明

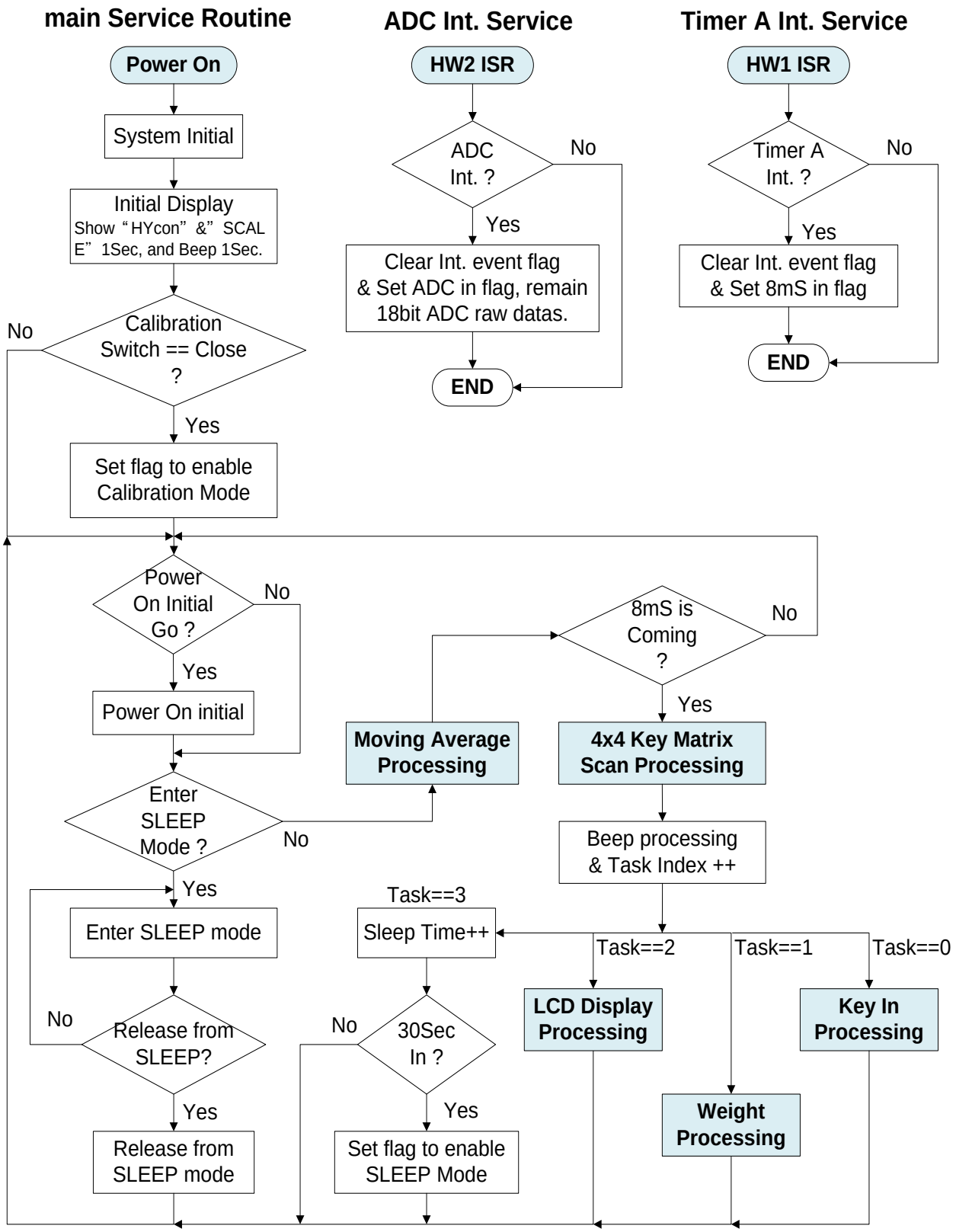
校正開關是用來進入校正模式，以用來記錄 0Kg 及 2Kg 這兩點的 ADC 校正值；當演示板上電前就將此開關關閉後再上電，則就可以進入校正模式，反之，開關如果是打開，則為正常的秤重模示。

3.2.5 蜂鳴器驅動說明

當按鍵輸入確認時蜂鳴器可用來產生提示聲，而 MCU 透過兩根 I/O 腳輸出 4KHz 的正反向 PWM Pulse 就可以直接驅動蜂鳴器來產生此提示聲。

3.3. 軟體說明

程式整體流程如圖 3-7 所示



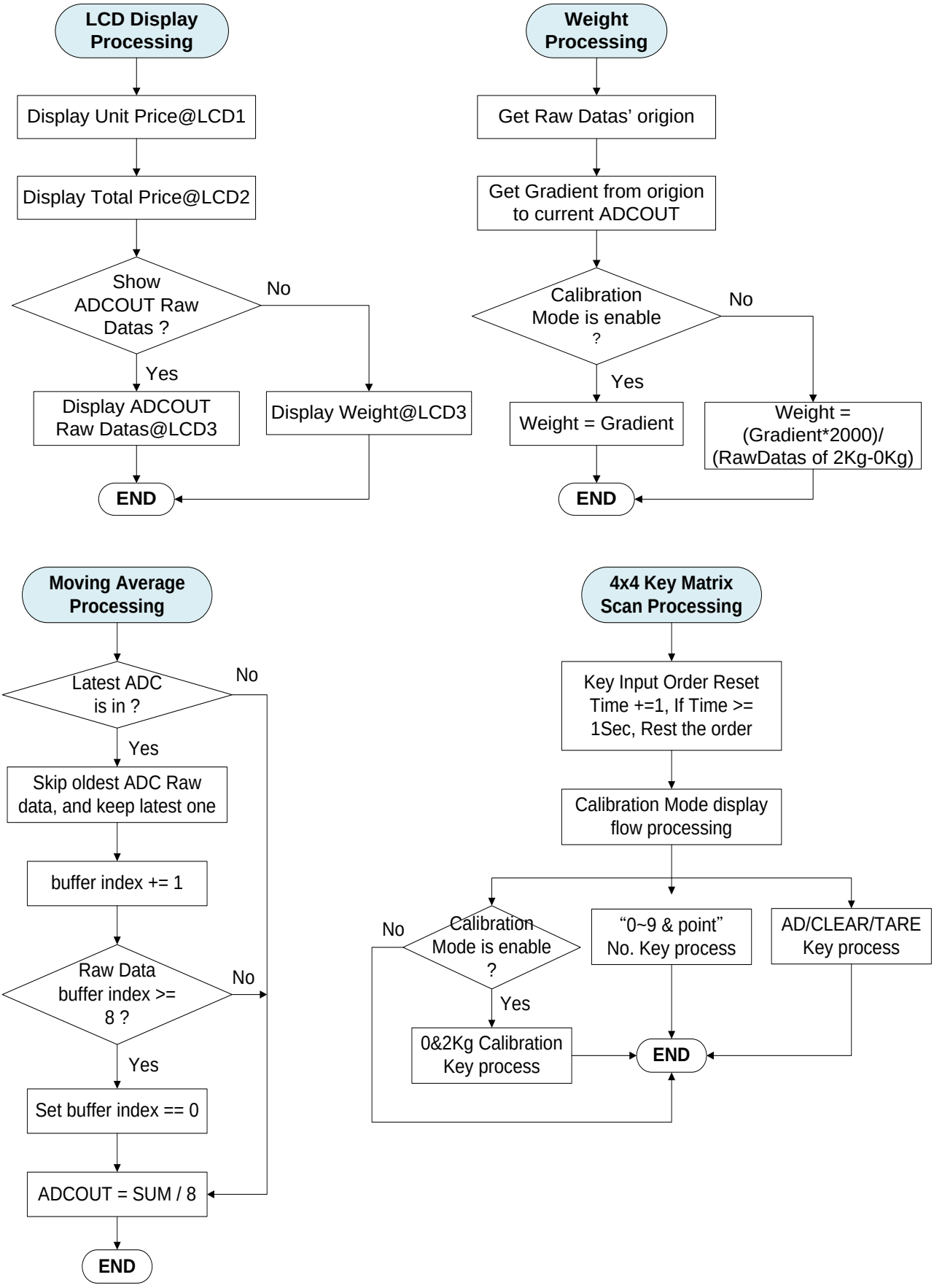

LCD Display Processing
Display Unit Price@LCD1
Display Total Price@LCD2
Show ADCOUT Raw Datas ?
No
Yes
Display ADCOUT Raw Datas@LCD3
Display Weight@LCD3
END
Weight Processing
Get Raw Datas' origion
Get Gradient from origion to current ADCOUT
Calibration Mode is enable ?
No
Yes
Weight = Gradient
Weight = (Gradient*2000)/ (RawDatas of 2Kg-0Kg)
END
Moving Average Processing
Latest ADC is in ?
No
Yes
Skip oldest ADC Raw data, and keep latest one
buffer index += 1
Raw Data buffer index >= 8 ?
No
Yes
Set buffer index == 0
ADCOUT = SUM / 8
END
4x4 Key Matrix Scan Processing
Key Input Order Reset Time +=1, If Time >= 1Sec, Rest the order
Calibration Mode display flow processing
Calibration Mode is enable ?
No
Yes
0&2Kg Calibration Key process
"0~9 & point" No. Key process
AD/CLEAR/TARE Key process
END

圖 3-7 程式整體流程圖

3.3.1 程式功能說明

(1) 上電（Power On）初始化

上電時開始做相關的 MCU 初始化，並對 LCD 初始化，讀取 0Kg 及 2Kg 標準重量校正值，並抓取當前重量的 ADC 值為秤重 0 點。初始化過程 LCD 會顯示『HYcon』及『SCALE』1 秒（此時蜂鳴器也會響），待 1 秒後初始化完成。

(2) 秤重模式

上電初始化完成後，如果校正開關是打開的，則進入正常的秤重模式，此時 LCD1 會顯示『00.00』、LCD2 會顯示『000.00』、LCD3 會顯示『00.000Kg』。

(3) 校正模式

上電初始化完成後，如果校正開關是關閉的，則進入校正模式，LCD1 會顯示『00.00』、LCD2 會顯示『000.00』、LCD3 會顯示『當前重量的 ADC 值』。

(4) 4x4 矩陣鍵盤操作

4x4 矩陣鍵盤的各按鍵功能如下表 3-1 說明，

Key No	Key Name	Function Description
S1	7	單價輸入按鍵：數字 7
S2	8	單價輸入按鍵：數字 8
S3	9	單價輸入按鍵：數字 9
S4	0Kg	0Kg 校正按鍵（校正模式下使用）
S5	4	單價輸入按鍵：數字 4
S6	5	單價輸入按鍵：數字 5
S7	6	單價輸入按鍵：數字 6
S8	2Kg	2Kg 校正按鍵（校正模式下使用）
S9	1	單價輸入按鍵：數字 1
S10	2	單價輸入按鍵：數字 2
S11	3	單價輸入按鍵：數字 3
S12	AD	顯示 ADC 量測值
S13	0	單價輸入按鍵：數字 0
S14	.	單價輸入按鍵：小數點
S15	CLEAR	清除設定
S16	TARE/ZERO	去皮按鍵

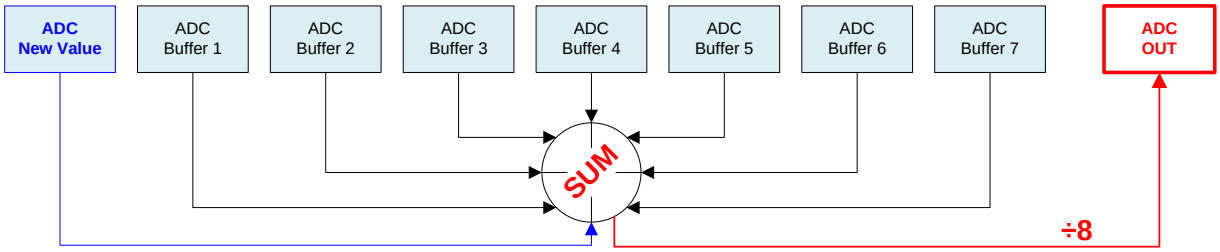
表 3-1 4x4 矩陣鍵盤之按鍵功能說明

3.3.2 ADC 數據處理

ADC 設置為對輸入信號（V+ ~V-）放大 128 倍，資料輸出率 OSR 為 ADC-CK/32768，每秒輸出 10 筆資料，最終取有效位元數為 18Bit。由於小訊號放大到 128 倍，ADC 的輸出 Bit 只能達到±17 Bit，如果使用軟體平均方式可以再將 ADC 的解析度提升 1~2Bit。

在此我們用移動平均濾波法做為軟體的平均方式，方法為將新的 ADC 值與前 7 筆舊的 ADC 值相加後再除以 8 以得到 ADC 最終轉換值（即每 8 筆資料做一次平均值），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。移動平均濾波實現如圖 3-8 所示，其中 ADC OUT 即為 ADC 最終轉換值。

由於平均輸出的反應時間比較慢，當有較大的 ADC 值變化時，需要跳過此平均程式。當 ADC 新值大於 ADC 平均值超過 0X200 時，先記錄此新 ADC 值，但不加入平均值運算，如果下一次的 ADC 值還是超過 0X200，將新值取代所有 ADC 的 Buffer 並輸出；如果下一次的 ADC 值沒有超過，可回到平均流程。



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

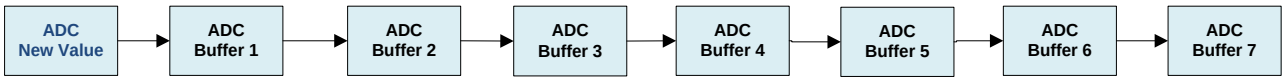


圖 3-8 移動平均濾波示意圖

校正模式程式整體流程如圖 3-7 所示

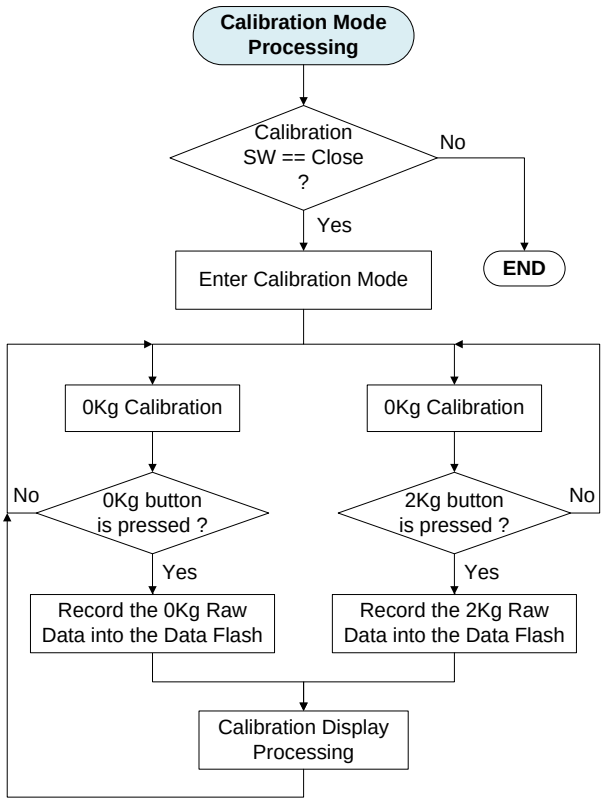
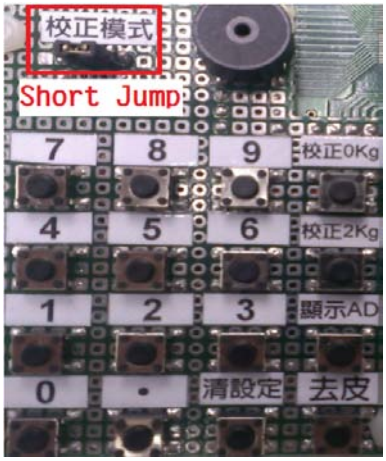
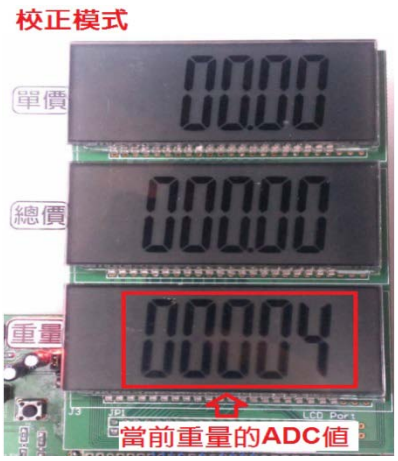


圖 3-9 校正模式程式流程圖

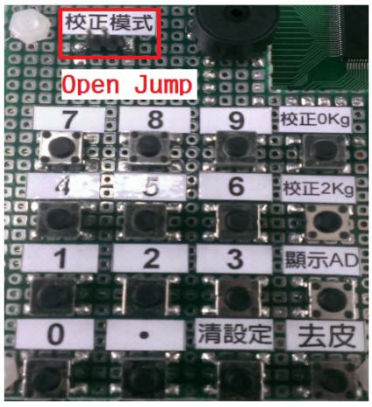
4. 操作流程

4.1. 校正模式操作步驟說明

STEP1:上電前先 Short 校正模式的 Jump 	STEP2:顯示當前 ADC 值 
STEP3:將秤台清空情況下按下校正 0Kg key 	STEP4:將 2kg 的砝碼放在秤台上，並按下校正 2kg key 

- 離開校正模式

Open 校正模式上的 Jump，並按下 Reset Key 離開校正模式.進入秤重模式

STEP1: 離開校正模式,先 Open 校正模式上的 Jump 	STEP2:按下 Reset Key 離開校正模式 

4.2. 秤重模式操作步驟說明

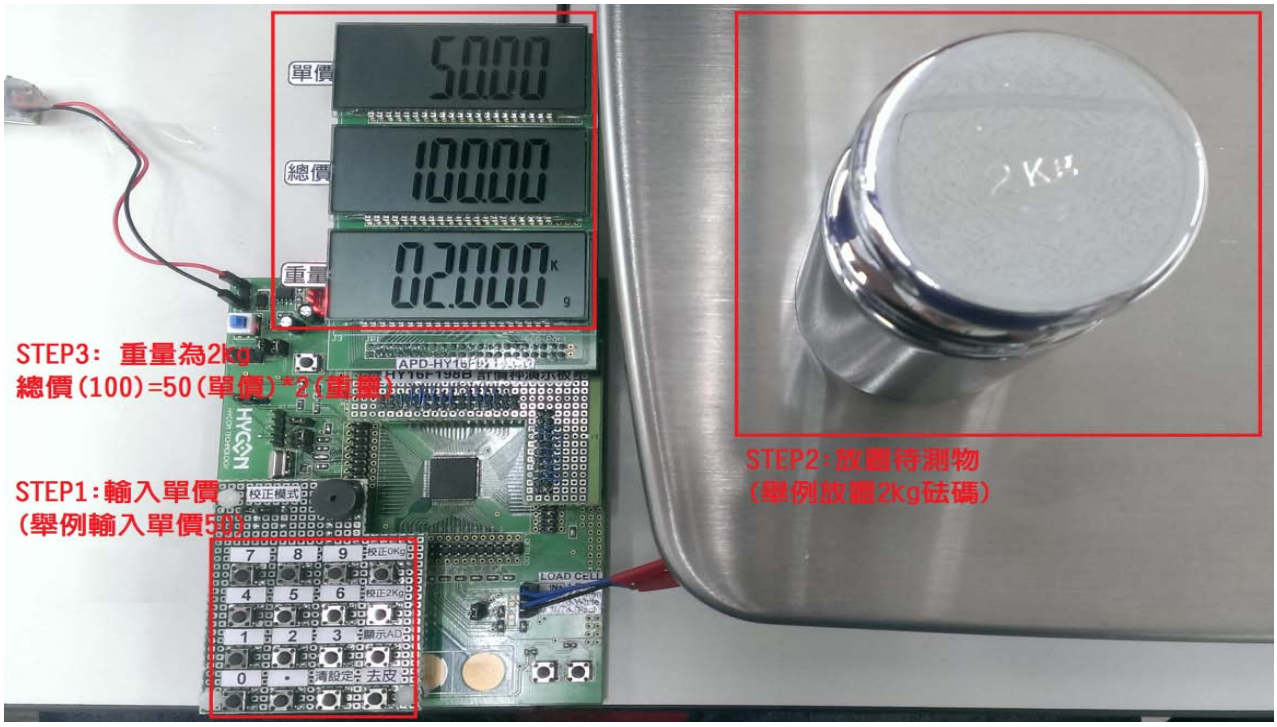
4.2.1. 秤重模式實際操作

STEP1: 輸入單價(舉例輸入單價為 50)

STEP2: 將待測物放入秤台上(放置 2kg 砝碼)

STEP3: 從 LCD3(顯示重量)可看到目前待測物重量

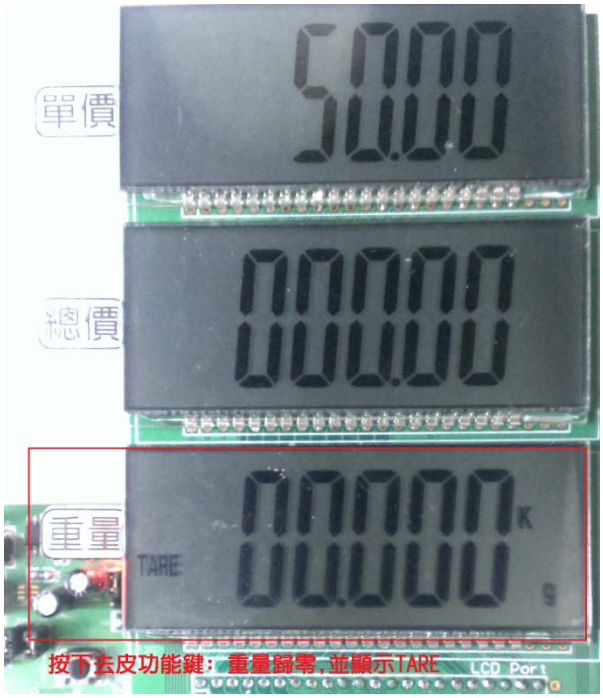
從 LCD2(顯示總價),總價=單價*重量 => 100=50*2



4.2.2. 顯示 AD & 清除設定 & 去皮功能說明

按下顯示 AD 功能鍵: 重量顯示變成當前待測物 ADC 數值	按下清除設定功能鍵: 單價與總價歸零, 重量顯示目前秤台上待測物重量.
	

按下去皮功能鍵:
重量顯示歸零,並且顯示 TARE.



省電功能說明:當 30 秒未按任何按鍵,會自動進入睡眠,LCD 會全部熄滅.按任何按鍵可喚醒.

5. 技術規格

(1) Operation voltage : 2.4~3.6V
(2) Operation current : 2.44mA@HSRC= 4MHz (3.0V)
(3) Sleep mode current : 2.5μA @3.0V
(4) ADC configuration : ADCCK=333KHz, OSR=0, ADC Output Rate=333K/32768=10Hz
(5) LCD configuration : LCDCK=434Hz (frame Frq.= 54Hz), VLCD=2.93V, 1/4Duty, 1/3Bias
(6) Scale technical SPEC. : 最大秤量/感量= 2Kg/1g
外部解析度= 2000 counts, 內部解析度= 20000 counts
單價輸入範圍= 0.00 ~ 99.99
總價顯示範圍= 0.00 ~ 999.99
秤重顯示範圍= 0.000Kg ~ 2.000Kg

5.1. 實驗記錄

標準砝碼秤重	演示板的實測結果		誤差 %	
重量 (Kg)	Max. (Kg)	Min. (Kg)	Max. (%)	Min. (%)
0.001	0.001	0.001	0	0
0.002	0.002	0.002	0	0
0.005	0.005	0.005	0	0
0.010	0.010	0.010	0	0
0.020	0.020	0.020	0	0
0.050	0.050	0.050	0	0
0.100	0.100	0.100	0	0
0.200	0.200	0.200	0	0
0.500	0.500	0.500	0	0
1.000	1.000	1.000	0	0
2.000	2.000	2.000	0	0

總結：

使用 HY16F198B 高精准度ΣΔADC,搭配兩點校正(0kg & 2kg)，再加上移動平均濾波法,從上述實驗記錄，可觀察計價秤相當精準.非常適合應用於電子秤這種對轉換速率要求不高的產品.

6. 附件

6.1. 範例程式

```
/*******************************************************************************
 *
 * Copyright (c) 2016-2026 HYCON Technology, Inc.
 * All rights reserved.
 * HYCON Technology <www.hycontek.com>
 *
 * HYCON reserves the right to amend this code without notice at any time.
 * HYCON assumes no responsibility for any errors appeared in the code,
 * and HYCON disclaims any express or implied warranty, relating to sale
 * and/or use of this code including liability or warranties relating
 * to fitness for a particular purpose, or infringement of any patent,
 * copyright or other intellectual property right.
 *
 * ----------------------------------------------------------------------------
 * Project Name : HY16F021V01 ( Pricing Scale )
 * IDE tooling    : AndeSight C/C++ IDE, version: 2.0.1 Build ID : 201303201736
 * Device Ver.    : V0.9 crt0.o for HY16F19xx & HY16F18xx MCU.
 * Library Ver. : 1.1
 * MCU Device     : HY16F198B-L100 @3.0V
 * Description    : This file implements the customer's function.
 * Created Date : 2016/06/29
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Header file include                                                       */
/*---------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "System.h"
#include "ModuleID.h"
#include "SpecialMacro.h"
#include "Sysinfra.h"
#include "DrvClock.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvREG32.h"
#include "DrvADC.h"
#include "DrvTimer.h"
#include "DrvUART.h"
#include "user_DEF.h"

/*---------------------------------------------------------------------------*/
/* Structure definition                                                      */
/*---------------------------------------------------------------------------*/
volatile 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;

/*---------------------------------------------------------------------------*/
/* Global variable definition                                                */
/*---------------------------------------------------------------------------*/
MCUSTATUS   MCUSTATUSbits;

/*---------------------------------------------------------------------------*/
/* MAIN   function                                                           */
/*---------------------------------------------------------------------------*/
int main(void)
{
    unsigned char scan_idx = 0;
```

```
    User_Sample_Init();

    //Enter Main program loop
    while(1)
    {
        //User task initial going ?
        if ( UserFlag0.b_InitGo_Flag == TRUE )
            PowerOn_Init_Go();

        //System enter SLEEP mode ?
        if ( UserFlag0.b_EntSLP_Flag == TRUE )
        {
            Enter_SLEEP_Mode();
            SYS_LowPower(0);                                    //休眠模式（Sleep mode）
            Release_SLEEP_Mode();
        }

        ADC_RawData_Avg_Process();

        //each task processing period= (base time) * 4tasks = 8mS * 4 = 32mS.
        //
        if( UserFlag0.b_TB8mS_Flag == TRUE )
        {
            UserFlag0.b_TB8mS_Flag = FALSE;
            Key_Scan_Process();                                 //4x4 key matrix scanning.
            Beep_Tone_Process();                                //Beep tone process !

            scan_idx = (scan_idx+1) & 0x03;                     //keep bit[1:0].
            switch( scan_idx )
            {
                //Key matrix scanning.
                case 0:
                    Key_In_Process();                           //Key in processing.
                  // Check_Cal_Mode_SW();                       //Cal. SW scanning!
                    break;
                //Key matrix scanning.
                case 1:
                    Get_Weight_Nums();
                    break;
                //LCD display !
                case 2:
                    LCD_Display_Process();
                    break;
                //Sleep Time Update counting.
                case 3:
                {
                    if ( 2 >= Sleep_Time_Cnt )
                        UserFlag0.b_EntSLP_Flag = TRUE;
                    else
                        Sleep_Time_Cnt--;
                    break;
                }
                default:
                    break;
            }
            DrvWDT_ClearWDT();                                  //Clear WDT interrupt flag
        }
    }
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Exception Service Routines                                                */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
    //procedure define by customer.
    asm("nop");
    asm("nop");
}

/*****************************************************************************/
/* File Name       : user_ISR.c                                              */
/* Description     : This file implements the customer's ISR function.       */
/*****************************************************************************/
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

//------TIMER A Int ?----------------------------
    if( DrvTIMER_GetIntFlag( E_TMA ) )
    {
        DrvTIMER_ClearIntFlag( E_TMA );          //Clear TMA interrupt flag
        UserFlag0.b_TB8mS_Flag = TRUE;
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{
    if( DrvADC_ReadIntFlag() )
    {
        ADC_RawData = DrvADC_GetConversionData();       //get ADC convert data
        ADC_RawData = ADC_RawData >> 14;                //remain 18bit ADC raw data.
        DrvADC_ClearIntFlag();                          // clear ADC interrupt flag
        UserFlag1.b_ADC_In_Flag = TRUE;
    }
}


/*****************************************************************************
 * File Name       : user_sample.c
 * IDE tooling     : AndeSight C/C++ IDE, version: 2.0.1 Build ID : 201303201736
 * Device Ver.     : V0.9 crt0.o for HY16F19xx & HY16F18xx MCU.
 * Library Ver. : 1.1
 * MCU Device      : HY16F198B-L100 @3.0V
 * Description     : This file implements the customer's function.
 * Created Date : 2016/06/29
 *****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Header file include                                                       */
/*---------------------------------------------------------------------------*/
//#include "stdint.h"
#include "HY16F198.h"
#include "System.h"
#include "DrvADC.h"
#include "DrvClock.h"
#include "DrvPMU.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvREG32.h"
#include "DrvRTC.h"
#include "DrvTimer.h"
#include "DrvUART.h"
#include "DrvFlash.h"
#include "ModuleID.h"
#include "SpecialMacro.h"
#include "Sysinfra.h"
#include "user_DEF.h"
#include "user_LCD.h"

/*---------------------------------------------------------------------------*/
/* Global variable definition                                                */
/*---------------------------------------------------------------------------*/
unsigned int   Sleep_Time_Cnt;
unsigned char DisplayBuffer[18];
unsigned char Beep_Tone_Cnt;
unsigned char Key_Code_Buf;
unsigned char Key_Code_No;
unsigned char Key_DB_Cnt;
unsigned int   ADCbuffer[16];
unsigned int   CAL_0Kg_Buf;
unsigned int   CAL_2Kg_Buf;
unsigned int   GradientBuf;
unsigned int   DecorticateBuf;
unsigned int   Total_Price;
unsigned int   Unit_Price;
unsigned int   Weight_Nums;
```

```
unsigned int   ADC_RawData;
unsigned int   RawData_Temp;
unsigned int   ADC_Avg_RawData;
unsigned int   ADC_Zero;
unsigned int   ADC_nums;
unsigned int   FIR_SumBuf;
unsigned char FIR_Buf_Idx;
unsigned int   NoKey_Reset_Cnt;
unsigned int   NoKeyBuf[4];
unsigned char NoBuf_Idx;
unsigned int   CAL_Step_Idx;

/*---------------------------------------------------------------------------*/
/* Constant value definition                                                 */
/*---------------------------------------------------------------------------*/
const unsigned char code_seg[] =
{
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f,          //00, char "0"
    seg_b+seg_c,                                  //01, char "1"
    seg_a+seg_b+seg_d+seg_e+seg_g,                //02, char "2"
    seg_a+seg_b+seg_c+seg_d+seg_g,                //03, char "3"
    seg_b+seg_c+seg_f+seg_g,                      //04, char "4"
    seg_a+seg_c+seg_d+seg_f+seg_g,                //05, char "5"
    seg_a+seg_c+seg_d+seg_e+seg_f+seg_g,          //06, char "6"
    seg_a+seg_b+seg_c,                            //07, char "7"
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f+seg_g,    //08, char "8"
    seg_a+seg_b+seg_c+seg_d+seg_f+seg_g,          //09, char "9"
    seg_a+seg_b+seg_c+seg_e+seg_f+seg_g,          //0A, char "A"
    seg_c+seg_d+seg_e+seg_f+seg_g,                //0B, char "b"
    seg_a+seg_d+seg_e+seg_f,                      //0C, char "C"
    seg_b+seg_c+seg_d+seg_e+seg_g,                //0D, char "d"
    seg_a+seg_d+seg_e+seg_f+seg_g,                //0E, char "E"
    seg_a+seg_e+seg_f+seg_g                       //0F, char "F"
};

/*---------------------------------------------------------------------------*/
/* Function prototype declaration                                            */
/*---------------------------------------------------------------------------*/
//------MISC Function declaration----------------
void User_Sample_Init(void);
void Delay(unsigned int num);
void Beep_Tone_Start(void);
void Beep_Tone_Stop(void);
void Key_Scan_Process(void);
void Key_In_Process(void);
void Key_In_Process(void);
int DrvFlash_WriteWord(unsigned int data_addr, unsigned int data_buf);
void Beep_Tone_Process(void);
void Beep_Tone_Process(void);
void PowerOn_Init_Go(void);
void Enter_SLEEP_Mode(void);
void Release_SLEEP_Mode(void);
void Get_Weight_Nums(void);
unsigned int Check_Origin(void);
//------ADC Function declaration----------------
void ADC_Init(void);
void ADC_RawData_Avg_Process(void);
//------GPIO Function declaration---------------
void GPIO_Init(void);
//------TIMER Function declaration--------------
void TIMER_Init(void);
void WDT_Init(void);
//------LCD Function declaration----------------
void LCD_Init(void);
void LCD_Display_Process(void);
void LCD_RAM_Clear(unsigned char SegBufSize);
void LCD_Data_Display(unsigned char lcd_no, unsigned int LcdBuffer);
void LCD2_Display_HYcon(void);
void LCD3_Display_SCALE(void);
void LCD3_Display_ADC(unsigned int adc_data);
void LCD3_Display_PASS(void);
void LCD3_Display_CAL(unsigned char kg_num);
void RAM2LCD( unsigned char *Buffer_Adr );

/*---------------------------------------------------------------------------*/
/* Function Name: User_Sample_Init                                           */
/*---------------------------------------------------------------------------*/
```

```
void User_Sample_Init(void)
{
    unsigned int InitGo_cnt;

    SYS_DisableGIE();                                   //Disable Global Interrupt.

//------Sys. clock setting----------------------
    DrvCLOCK_SelectIHOSC( TRIM_HAO4MHZ );               //Select HAO= 4MHz.
    DrvCLOCK_EnableHighOSC(E_INTERNAL, 60);             //Enable HAO.
    DrvCLOCK_SelectMCUClock(0, 0);                      //Select MCUCKS= HS_CK/1.

//------MISC. system initial--------------------
    GPIO_Init();
    LCD_Init();
    ADC_Init();
    TIMER_Init();
    UserFlag0._byte = 0;
    SYS_EnableGIE( 4, 0x06 );                           //Enable HW1 & HW2 Int. only.

//------Power on initial going------------------
//Beep tone on & LCD display HYcon SCALE about 1Sec.
//
    Beep_Tone_Start();
    LCD2_Display_HYcon();
    LCD3_Display_SCALE();
    RAM2LCD( DisplayBuffer );                           //Update LCD display RAM.
    for ( InitGo_cnt= 0; InitGo_cnt < INIT_GO_TIME; )
    {
        if ( UserFlag0.b_TB8mS_Flag == TRUE )
        {
            UserFlag0.b_TB8mS_Flag = FALSE;
            InitGo_cnt++;
        }
    }
//  Beep_Tone_Stop();
    LCD_RAM_Clear( SEG_BUF_SIZE );
    UserFlag0.b_InitGo_Flag = TRUE;
}

/*---------------------------------------------------------------------------*/
/* Function Name: PowerOn_Init_Go                                            */
/*---------------------------------------------------------------------------*/
void PowerOn_Init_Go(void)
{
    unsigned int i, j;
    SYS_DisableGIE();                                   //Disable Global Interrupt.
//------MISC data initial-----------------------
    GPIO_Init();
    ADC_Init();
    LCD_Init();
    TIMER_Init();

    UserFlag1._byte = 0;
    Total_Price = 0;
    DecorticateBuf = 0;
    NoKey_Reset_Cnt = 0;
    NoBuf_Idx = 2;
    NoKeyBuf[0] = 0;
    NoKeyBuf[1] = 0;
    NoKeyBuf[2] = 0;
    NoKeyBuf[3] = 0;

    CAL_0Kg_Buf = ReadWord( CAL_0KG_BUF_ADDR );
    CAL_2Kg_Buf = ReadWord( CAL_2KG_BUF_ADDR );
//------Enter Calibration mode ?----------------
    for ( j= 1000, i=0; i < 500; i++ )
    {
        Delay( 10 );                                    //delay 18uS @4MHz.
        if ( DrvGPIO_GetPortBits( E_PT2 ) & 0x04 )
            j--;
        else
            j++;
    }
    if ( j > 1400 ) { UserFlag1.b_CalEI_Flag = TRUE;}
    DrvGPIO_PT2_DisableINPUT( 0x04 );                   //Disable PT2.2 IE.
    DrvGPIO_PT2_DisablePullHigh( 0x04 );                //Disable PT2.2 pull high.
```

```
    if( (CAL_0Kg_Buf == 0) || (CAL_2Kg_Buf == 0) || ( CAL_0Kg_Buf > CAL_2Kg_Buf) || ( CAL_0Kg_Buf == CAL_2Kg_Buf) )
    {
        UserFlag1.b_CalEI_Flag = TRUE;              //enable Calibration mode.
    }

//------System initial go !---------------------
    SYS_EnableGIE( 4, 0x06 );                       //Enable HW1 & HW2 Int. only.

    if ( UserFlag0.b_PowOn_Flag == FALSE )
    {
        //enable Comb Filter.
        DrvADC_CombFilter( 0 );                     //reset !
        DrvADC_CombFilter( 1 );                     //enable !

        for(i=0; i<16; i++)
        {
            ADCbuffer[i] = 0;
        }

        FIR_SumBuf = 0;
        FIR_Buf_Idx = 0;
        Unit_Price = 0;
        Weight_Nums = 0;

        //initial FIR buffer.
        while( UserFlag1.b_ADC_In_Flag == FALSE );
        UserFlag1.b_ADC_In_Flag = FALSE;
        for( i=0; i<8; i++)
        {
            ADCbuffer[i]= ADC_RawData;
            FIR_SumBuf += ADCbuffer[i];
            ADCbuffer[i]= ADC_RawData;
        }
        ADC_Avg_RawData = ADC_RawData;
        ADC_Zero = ADC_Avg_RawData;
    }
    //fast update ADC data, if ADC Raw data is changed more than 400 counts.
    else if ( ( UserFlag0.b_PowOn_Flag == TRUE ) && ( UserFlag0.b_CalGo_Flag == FALSE ) )
    {
        for ( i=0, j=0; i<4; i++ )
        {
            while( UserFlag1.b_ADC_In_Flag == FALSE );
            UserFlag1.b_ADC_In_Flag = FALSE;
            if ( ( ADC_RawData > RawData_Temp+30 ) || ( ADC_RawData < RawData_Temp+30 ) )
                j++;
            else { j = 0; }
        }

        if ( j == 4 )
        {
            FIR_SumBuf = 0;
            FIR_Buf_Idx = 0;
            for( i=0; i<8; i++)
            {
                ADCbuffer[i]= ADC_RawData;
                FIR_SumBuf += ADCbuffer[i];
                ADCbuffer[i]= ADC_RawData;
            }
            ADC_Avg_RawData = ADC_RawData;
            ADC_nums = ADC_RawData;
        }
        Beep_Tone_Start();
    }

    UserFlag0.b_PowOn_Flag = TRUE;
    UserFlag0.b_CalGo_Flag = FALSE;
    UserFlag0.b_InitGo_Flag = FALSE;
    Sleep_Time_Cnt = SLEEP_TIME;
    LCD_Data_Display( LCD_PANEL1, Unit_Price );
    LCD_Data_Display( LCD_PANEL2, Total_Price );
    LCD_Data_Display( LCD_PANEL3, Weight_Nums );
    WDT_Init();
}

/*---------------------------------------------------------------------------*/
/* Function Name: Get_Weight_Nums                                            */
/*---------------------------------------------------------------------------*/
```

```
void Get_Weight_Nums(void)
{
//------Renew Origin---------------------------
    if( ADC_Avg_RawData < ADC_Zero )
    {
        ADC_Zero = ADC_Avg_RawData;
    }
    else
    {
        if( ADC_Avg_RawData < (ADC_Zero + 3) )
        {
            ADC_Avg_RawData = ADC_Zero;
        }
    }

//------Get Gradient (means delta-ADC)----------
    if(ADC_nums > ADC_Avg_RawData)
    {
        if( ADC_nums > (ADC_Avg_RawData + 3) )
        {
            ADC_nums = ADC_Avg_RawData;
        }
    }
    else
    {
        if( ADC_Avg_RawData > (ADC_nums + 3) )
        {
            ADC_nums = ADC_Avg_RawData;
        }
    }

    if ( ADC_nums < ADC_Zero ) { ADC_nums = ADC_Zero; }
    GradientBuf = ADC_nums - ADC_Zero;

//------Get current Weight----------------------
    if ( UserFlag1.b_CalEI_Flag == FALSE )
    {
        Weight_Nums = 0;
        if( (CAL_0Kg_Buf != 0) && (CAL_2Kg_Buf != 0) && (CAL_2Kg_Buf > CAL_0Kg_Buf+500) && (CAL_2Kg_Buf !=
CAL_0Kg_Buf) )
            Weight_Nums = (GradientBuf*2000) / (CAL_2Kg_Buf - CAL_0Kg_Buf);                    //計算總量，單位g
    }
    else
    {
        Weight_Nums = GradientBuf;                                  //只顯示ADC內部數值
    }

    if( Weight_Nums < DecorticateBuf )
    {
        Weight_Nums = 0;
    }
    else
    {
        Weight_Nums = Weight_Nums - DecorticateBuf;
    }

    //Reset Sleep Counter?
    if ( ( RawData_Temp > ADC_Avg_RawData + 20 ) || ( RawData_Temp + 20 < ADC_Avg_RawData ) )
        Sleep_Time_Cnt = SLEEP_TIME;
    RawData_Temp = ADC_Avg_RawData;
}

/*---------------------------------------------------------------------------*/
/* Function Name: Check_Origin                                               */
/*---------------------------------------------------------------------------*/
unsigned int Check_Origin(void)
{
    unsigned int checknum = 0, checkbuf = 0;
    while( checknum < 100 )
    {
        ADC_RawData_Avg_Process();
        if( -5 < (int)checkbuf-(int)ADC_Avg_RawData || (int)checkbuf-(int)ADC_Avg_RawData < 5)
        { checknum += 1; }
        else
            checknum = 0;
        checkbuf = ADC_Avg_RawData;
```

```
    }
    return checkbuf;
}

/*---------------------------------------------------------------------------*/
/* Function Name: Key_Scan_Process                                           */
/*---------------------------------------------------------------------------*/
void Key_Scan_Process(void)
{
    unsigned char key_code, temp, s_bit;
    unsigned int    r_data, r_bit;

    key_code = NULL_KEY_NO;
    temp = NULL_KEY_NO;
//------Matrix scaning-------------------------
    for ( s_bit= 0x80; s_bit !=0x08; s_bit = s_bit >> 1 )
    {
        DrvGPIO_PT1_DisableOUTPUT( ~s_bit );
        DrvGPIO_PT1_EnableOUTPUT( s_bit );
        Delay( 10 );                                    //delay 18uS @4MHz.
        r_data = DrvGPIO_GetPortBits( E_PT1 );
        for ( r_bit= 0x08; r_bit > 0; r_bit = r_bit >> 1 )
        {
            temp++;
            if ( !(r_data & r_bit) )
                key_code = temp;
        }
    }

//------Key in debounce------------------------
    Key_DB_Cnt++;
    if ( key_code != Key_Code_Buf )
    {
        Key_Code_Buf = key_code;
        Key_DB_Cnt = 0;
    }
    else
    {
        if (Key_DB_Cnt >= KEY_DB_TIME)
        {
            Key_DB_Cnt = 0;
            if ( Key_Code_Buf == NULL_KEY_NO )
            {
                UserFlag0.b_KeyOn_Flag = FALSE;
            }
            else
            {
                if (UserFlag0.b_KeyOn_Flag == FALSE)
                {
                    UserFlag0.b_KeyOn_Flag = TRUE;
                    UserFlag0.b_KInPro_Flag = TRUE;
                    Key_Code_No = Key_Code_Buf;
                }
                Sleep_Time_Cnt = SLEEP_TIME;
            }
        }
    }
    DrvGPIO_PT1_DisableOUTPUT( 0xFF );              //close PT1[7:0] OE.
    DrvGPIO_PT1_ClrPortBits(0xF0);                  //PT1[7:4] output low.
}

/*---------------------------------------------------------------------------*/
/* Function Name: Key_In_Process                                             */
/*---------------------------------------------------------------------------*/
void Key_In_Process(void)
{
    unsigned char no = 0xFF;

//------Rest for newly No. key in---------------
    NoKey_Reset_Cnt++;
    if ( NoKey_Reset_Cnt > NO_KEY_RESET_TIME )
    {
        UserFlag1.b_DotKey_Flag = FALSE;
        NoKey_Reset_Cnt = 0;
        NoBuf_Idx = 2;
        NoKeyBuf[0] = 0;
        NoKeyBuf[1] = 0;
```

```
            NoKeyBuf[2] = 0;
            NoKeyBuf[3] = 0;
        }

//------Next Cal. Step display ?----------------
//step1: just delay 1500mS
//step2: show "PASS" at LCD3 about 1000mS.
//step3: release Cal. mode.
//
        if( UserFlag0.b_CalGo_Flag == TRUE )
        {
            CAL_Step_Idx += 1;

            if( CAL_Step_Idx == (1500u/32u) )
                LCD3_Display_PASS();
            else if( CAL_Step_Idx > (2500u/32u) )
                UserFlag0.b_InitGo_Flag = TRUE;

            UserFlag0.b_KInPro_Flag = FALSE;
            return;
        }

//------Key In process going--------------------
        if( UserFlag0.b_KInPro_Flag != TRUE )
            return;
        UserFlag0.b_KInPro_Flag = FALSE;

        switch( Key_Code_No )
        {
            //No. 7
            case S1_KEY_NO:
                no = 7;
                break;
            //No. 8
            case S2_KEY_NO:
                no = 8;
                break;
            //No. 9
            case S3_KEY_NO:
                no = 9;
                break;
            //CAL. 0Kg.
            case S4_KEY_NO:
            {
                if ( UserFlag1.b_CalEI_Flag == TRUE )
                {
                    UserFlag0.b_CalGo_Flag = TRUE;
                    CAL_Step_Idx = 0;
                    LCD3_Display_CAL( 0 );
                    CAL_0Kg_Buf = Check_Origin();
                    ADC_Zero = ADC_Avg_RawData;
                    SYS_DisableGIE();                                   //Disable Global Interrupt.
                    DrvFlash_WriteWord( CAL_0KG_BUF_ADDR, CAL_0Kg_Buf );
                    SYS_EnableGIE( 4, 0x06 );                           //Enable HW1 & HW2 Int. only.
                }
                break;
            //No. 4
            case S5_KEY_NO:
                no = 4;
                break;
            //No. 5
            case S6_KEY_NO:
                no = 5;
                break;
            //No. 6
            case S7_KEY_NO:
                no = 6;
                break;
            //CAL. 2Kg.
            case S8_KEY_NO:
            {
                if ( UserFlag1.b_CalEI_Flag == TRUE )
                {
                    UserFlag0.b_CalGo_Flag = TRUE;
                    CAL_Step_Idx = 0;
                    LCD3_Display_CAL( 2 );
                    CAL_2Kg_Buf = Check_Origin();
```

```
                    ADC_nums = ADC_Avg_RawData;
                    SYS_DisableGIE();                                   //Disable Global Interrupt.
                    DrvFlash_WriteWord( CAL_2KG_BUF_ADDR, CAL_2Kg_Buf );
                    SYS_EnableGIE( 4, 0x06 );
                }
                break;          }
            //No. 1
            case S9_KEY_NO:
                no = 1;
                break;
            //No. 2
            case S10_KEY_NO:
                no = 2;
                break;
            case S11_KEY_NO:
                no = 3;
                break;
            //Show ADC raw data.
            case S12_KEY_NO:
                UserFlag1.b_ShowAD_Flag = TRUE;
                break;
            //No. 0
            case S13_KEY_NO:
                no = 0;
                break;
            //"."
            case S14_KEY_NO:
            {
                UserFlag1.b_DotKey_Flag = TRUE;
                NoKey_Reset_Cnt = 0;
                break;
            }
            //Clear Setting.
            case S15_KEY_NO:
            {
                UserFlag1.b_ShowAD_Flag = FALSE;
                UserFlag1.b_DotKey_Flag = FALSE;
                NoBuf_Idx = 2;
                NoKeyBuf[0] = 0;
                NoKeyBuf[1] = 0;
                NoKeyBuf[2] = 0;
                NoKeyBuf[3] = 0;
                Unit_Price = 0;
                DecorticateBuf = 0;
                UserFlag1.b_TareEI_Flag = FALSE;
                break;
            }
            //Decorticate (Zero).
            case S16_KEY_NO:
                DecorticateBuf = Weight_Nums + DecorticateBuf;

                UserFlag1.b_TareEI_Flag = TRUE;
                break;
            default:
                break;
        }

        //No. 0~9 key in !
        if ( no < 10 )
        {
            NoKey_Reset_Cnt = 0;
            if ( UserFlag1.b_DotKey_Flag != TRUE )
            {
                NoKeyBuf[NoBuf_Idx+1] = NoKeyBuf[NoBuf_Idx];
                NoKeyBuf[NoBuf_Idx] = no;
            }
            else if ( NoBuf_Idx != 0 )
            {
                NoBuf_Idx--;
                NoKeyBuf[NoBuf_Idx] = no;
            }
            Unit_Price = NoKeyBuf[3]*1000 + NoKeyBuf[2]*100 + NoKeyBuf[1]*10 + NoKeyBuf[0];        //計算單價
        }
        Beep_Tone_Start();
}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: DrvFlash_WriteWord                                         */
/*---------------------------------------------------------------------------*/
int DrvFlash_WriteWord(unsigned int data_addr, unsigned int data_buf)
{
    unsigned char i;

    for ( i=0; i<BURN_VERIFY_TIMES; i++ )
    {
        DrvFlash_Burn_Word( data_addr, 0x2000, data_buf );
        //Verify.
        if ( data_buf == ReadWord( data_addr ) )
            return TRUE;
    }
    return FALSE;
}

/*---------------------------------------------------------------------------*/
/* Function Name: Enter_SLEEP_Mode                                           */
/*---------------------------------------------------------------------------*/
void Enter_SLEEP_Mode(void)
{
    SYS_DisableGIE();                           //Disable Global Interrupt.
    RawData_Temp = ADC_Avg_RawData;             //backup last ADC_Avg_RawData
//------Close all LCD IP-----------------------
    LCD_RAM_Clear( SEG_BUF_SIZE );
    RAM2LCD( DisplayBuffer );                   //Update LCD display RAM.
    DrvLCD_DisplayMode( E_LCD_PIXELOFF );       //All LCD off.
    DrvLCD_VLCDMode( E_VLCD_DISABLE );          //close VLCD.
    while( (inw(0x41B00) & (1<<IDF) ) == 0);    //Wait LCD Idle, IDF= 20.
//------Close all ADC IP-----------------------
    DrvADC_Disable();                           //disable ADC.
    DrvADC_ClkDisable();
    DrvADC_DisableInt();                        //Disable ADC interrupt.
    DrvADC_ClearIntFlag();                      //clear ADC interrupt flag.
    DrvPMU_VDDA_LDO_Ctrl( 0 );                  //Disable VDDA output.
//------Close all Timer IP---------------------
    DrvTMA_Close ();
    DrvTIMER_DisableInt( E_TMA );
    DrvTIMER_ClearIntFlag( E_TMA );
    DrvTMB_Close();
//------Initial GPIO for SLEEP wake up---------
    DrvGPIO_PT1_EnableOUTPUT( 0xF0 );           //Enable PT1[7:4] OE.
    DrvGPIO_PT1_ClearIntFlag( 0xFF );           //clear PT1[7:0] interrupt flag.
    DrvGPIO_PT1_EnableINT( 0x0F );              //PT1[3:0] interrupt enable.

    DrvPMU_LDO_LowPower(1);                     //SET low power mode
    DrvWDT_ClearWDT();                          //Clear WDT interrupt flag
}

/*---------------------------------------------------------------------------*/
/* Function Name: Release_SLEEP_Mode                                         */
/*---------------------------------------------------------------------------*/
void Release_SLEEP_Mode(void)
{
    unsigned char hi_status= 0, i;
    unsigned int   r_data;

    //Debounce for PT1[3:0] wake up.
    for ( i=0; i < 250; i++ )
    {
        Delay( 100 );                           //delay 40uS @4MHz.
        r_data = DrvGPIO_GetPortBits( E_PT1 );
        if ( (r_data & 0x0F) != 0x0F )
            hi_status++;
        else
            hi_status = 0;
    }
    if ( hi_status >= 200 )
    {
        UserFlag0.b_EntSLP_Flag = FALSE;
        UserFlag0.b_InitGo_Flag = TRUE;
        UserFlag0.b_KeyOn_Flag   = TRUE;
    }
}

/*---------------------------------------------------------------------------*/
```

```
/* Function Name: Beep_Tone_Process                                          */
/*---------------------------------------------------------------------------*/
void Beep_Tone_Process(void)
{
    if (Beep_Tone_Cnt != 0)
    {
        Beep_Tone_Cnt--;
        if (Beep_Tone_Cnt <= 0)
            Beep_Tone_Stop();
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: Beep_Tone_Start                                            */
/*---------------------------------------------------------------------------*/
void Beep_Tone_Start(void)
{
#if ( BEEP_TONE_ENABLE )
    Beep_Tone_Cnt = BEEP_TONE_TIME;
    DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );    //set TMB working on normal mode & TMB clock int
                                                                //period= PWM_4KHZ_TIME/HS_CK= 250uS.
    DrvPWM0_Open( 0, 1, 2 );                                    //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO0 working on PWMA
                                                                //mdoe with normal pulse output.
    DrvPWM1_Open( 0, 0, 2 );                                    //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO1 working on PWMA
                                                                //mdoe with reverse pulse output.
#endif
}

/*---------------------------------------------------------------------------*/
/* Function Name: Beep_Tone_Stop                                             */
/*---------------------------------------------------------------------------*/
void Beep_Tone_Stop(void)
{
    Beep_Tone_Cnt = 0;
    DrvTMB_Close();
    DrvPWM0_Close();
    DrvPWM1_Close();
    DrvGPIO_ClrPortBits( E_PT2, 0x03 );                         //set PT2[1:0] as output low.
}

/*---------------------------------------------------------------------------*/
/* Function Name: Delay()                                                    */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for( ; num >0; num-- )
        asm( "NOP" );
}


/*****************************************************************************/
/* File Name       : user_GPIO.c                                             */
/*****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Function Name: GPIO_Init                                                  */
/*---------------------------------------------------------------------------*/
void GPIO_Init(void)
{
//------PT1 GPIO initial------------------------
    DrvGPIO_ClkGenerator( E_HS_CK, 3 );                         //Set IO sampling clock is HS_CK, and IOCLK is HS_CK/4.
    DrvGPIO_PT1_IntTriggerPorts( 0x0F, E_N_Edge );              //PT1[3:0] interrupt trigger method is negative edge.
    DrvGPIO_PT1_DisableINT( 0xFF );                             //PT1[7:0] interrupt disable.
    DrvGPIO_PT1_ClearIntFlag( 0xFF );                           //clear PT1[7:0] interrupt flag.
    DrvGPIO_PT1_EnablePullHigh( 0x0F );                         //enable PT1[3:0] pull high 75K ohm.
    DrvGPIO_PT1_EnableINPUT( 0x0F );                            //set PT1[3:0] as input port.
    DrvGPIO_PT1_DisableOUTPUT( 0xFF );                          //close PT1[7:0] OE.

//------PT2 GPIO initial------------------------
    //PWM function initial
    DrvGPIO_Open( E_PT2, 0x03, E_IO_OUTPUT );                   //set PT2[1:0] as output port.
    DrvPWM_CountCondition( PWM_4KHZ_TIME/2, PWM_4KHZ_TIME/2 );  //Set TBC1 as 50% duty, TBC2 as 50% duty.
    DrvGPIO_ClrPortBits( E_PT2, 0x03 );                         //set PT2[1:0] as output low.

    //PT2.2 Cal. sw initial
    DrvGPIO_PT2_DisableOUTPUT( 0x04 );                          //close PT2.2 OE.
    DrvGPIO_PT2_EnablePullHigh( 0x04 );                         //Enable PT2.2 pull high 75K ohm.
    DrvGPIO_PT2_EnableINPUT( 0x04 );                            //Enable PT2.2 IE.
```

```
}


/*****************************************************************************/
/* File Name       : user_ADC.c                                              */
/*****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Function Name: ADC_Init                                                   */
/*---------------------------------------------------------------------------*/
void ADC_Init(void)
{
    //set ADC input pin
    DrvADC_PInputChannel( ADC_Input_AIO0 );       //set ADC positive input pin is AIO0.
    DrvADC_NInputChannel( ADC_Input_AIO1 );       //set ADC negative input pin is AIO1.
    DrvADC_InputSwitch( OPEN );                   //set ADC input short-switch is open
    DrvADC_RefInputShort( OPEN );                 //set VREF input short-switch is open
    DrvADC_Gain( ADC_PGA_32, ADC_ADGN_4 );    //set ADC    GAIN is PGA * ADGN = 32x4 = 128.

    //set ADC DC offset
    DrvADC_DCoffset( 0 );                         //set ADC DC input offset is 0 VREF

    //set ADC input reference voltage
    DrvADC_RefVoltage ( VDDA, VSSA );             //set VREF positive input(REFP) is VDDA, negative input(REFN) is VSSA.
    DrvADC_FullRefRange( 1 );                    //set VREF gain = 1/2 VREF= 2.4V/2= 1.2V

    //set ADC comb filter
    DrvADC_OSR( 0 );                             //set OSR= ADCK/32768 = (4MHz/12)/32768 = 10 sps.
    DrvADC_CombFilter( 1 );                       //enable Comb filter.

    //set ADC clock
    DrvADC_ClkEnable( 1,1 );                     //set ADCK = HS_CK/12 = 4MHz/12 = 333KHz    & rising edge is high.

    //set VDDA voltage
    DrvPMU_VDDA_LDO_Ctrl( 3 );                    //enable adjustable LDO mode.
    DrvPMU_VDDA_Voltage( 0 );                     //set VDDA = 2.4V

    //set VREF.
    DrvPMU_AnalogGround( 1 );                     //set analog enable buffer and use internal source
    Delay( 0x1000 );

    //set ADC interrupt
    DrvADC_ClearIntFlag();                       //clear ADC interrupt flag.
    DrvADC_EnableInt();                           //enable ADC interrupt.
    DrvADC_Enable();                             //enable ADC and start ADC.
}

/*---------------------------------------------------------------------------*/
/* Function Name: ADC_RawData_Avg_Process                                    */
/*---------------------------------------------------------------------------*/
void ADC_RawData_Avg_Process(void)
{
    if( UserFlag1.b_ADC_In_Flag == TRUE )
    {
        UserFlag1.b_ADC_In_Flag = FALSE;
        FIR_SumBuf -= ADCbuffer[FIR_Buf_Idx];
        ADCbuffer[FIR_Buf_Idx] = ADC_RawData;
        FIR_SumBuf += ADCbuffer[FIR_Buf_Idx];
        FIR_Buf_Idx += 1;
        if( FIR_Buf_Idx > (8-1) ) { FIR_Buf_Idx = 0; }
        ADC_Avg_RawData = (FIR_SumBuf >> 3);              //FIR_SumBuf/8
        asm("NOP");
    }
}


/*****************************************************************************/
/* File Name       : user_TIMER.c                                            */
/*****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Function Name: TIMER_Init                                                 */
/*---------------------------------------------------------------------------*/
void TIMER_Init(void)
{

//------Initial TIMER A for Time Base-----------
    DrvTMA_Open( 9,0 );                          //Set TimerA int period= HS_CK/32/1024= 4MHz/32768= 8mS.
    DrvTIMER_ClearIntFlag( E_TMA );               //Clear Timer A interrupt flag
```

```
    DrvTIMER_EnableInt( E_TMA );                    //Timer A interrupt enable

//------Initial TIMER B for PWM-----------------
    DrvTMBC_Clk_Source( E_HS_CK, 0 );               //set TMB Clock from HS_CK/1.
    DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );    //set TMB working on normal mode & TMB clock int
                                                                //period= PWM_4KHZ_TIME/HS_CK= 250uS.
}

/*---------------------------------------------------------------------------*/
/* Function Name: WDT_Init                                                   */
/*---------------------------------------------------------------------------*/
void WDT_Init(void)
{
#if ( WDT_RESET_ENABLE )
    DrvWDT_Open( E_NMI, E_PRE_SCALER_D512 );    //Enable WDT and set WDT time out period = (LSRC/512)/256= 0.267Hz.
    DrvWDT_ClearWDT();                          //Clear WDT interrupt flag
#endif
}


/*****************************************************************************/
/* File Name      : user_LCD.c                                               */
/*****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Function Name: LCD_Init                                                   */
/*---------------------------------------------------------------------------*/
void LCD_Init(void)
{
    clk_10 = 0x40404B4B;                        //Set LCK= (HS_CK/64)/9/16= (HAO/64)/9/16= (4MHz/64)/144= 434Hz &
                                                //charge pump clock = HS_CK/16.
    DrvLCD_VLCDMode( E_VLCD27 );
    DrvLCD_VLCDTrim( 3 );                       //Trim VLCD as 2.93V.
    DrvLCD_LcdDuty ( E_LCD_DUTY4 );             //enable LCD@1/4 duty.

    DrvLCD_IOMode( 5, 0xFF );                   //Set SEG1/0 as LCD mode.
    DrvLCD_IOMode( 0, 0xFF );                   //Set PT6(SEG2 ~SEG9 ) as LCD mode.
    DrvLCD_IOMode( 1, 0xFF );                   //Set PT7(SEG10~SEG17) as LCD mode.
    DrvLCD_IOMode( 2, 0xFF );                   //Set PT8(SEG18~SEG25) as LCD mode.
    DrvLCD_IOMode( 3, 0xFF );                   //Set PT9(SEG26~SEG33) as LCD mode.
    DrvLCD_IOMode( 4, 0xFF );                   //Set PT10(SEG34~SEG35) as LCD mode.
    DrvLCD_LCDBuffer( ENABLE );                 //enable LCD working.
    DrvLCD_DisplayMode( E_LCD_NORMAL );         //LCD at normal mode.
    LCD_RAM_Clear( SEG_BUF_SIZE );
}

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_Display_Process                                        */
/*---------------------------------------------------------------------------*/
void LCD_Display_Process(void)
{
    LCD_Data_Display( LCD_PANEL1, Unit_Price );                 //顯示單價
    Total_Price = (Unit_Price * Weight_Nums) / 1000;            //計算總金額
    LCD_Data_Display( LCD_PANEL2, (unsigned int)Total_Price );  //顯示總金額

    if( UserFlag0.b_CalGo_Flag != TRUE )
    {
        if( UserFlag1.b_ShowAD_Flag == FALSE )
        {
            LCD_Data_Display( LCD_PANEL3, Weight_Nums );        //顯示總重量  g.
        }
        else if( UserFlag1.b_ShowAD_Flag == TRUE )
        {
            LCD_Data_Display( LCD_PANEL3, ADC_Avg_RawData );    //顯示  Avg's ADc Raw Data.
        }
    }

    RAM2LCD( DisplayBuffer );
}

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_RAM_Clear                                              */
/*---------------------------------------------------------------------------*/
void LCD_RAM_Clear(unsigned char SegBufSize)
{
    int i=0;
    for(i=0; i<SegBufSize; i++)
```

```
    {
        DisplayBuffer[i] = 0x00;
    }
}

/*---------------------------------------------------------------------*/
/* Function Name: LCD2_Display_HYcon                                   */
/*---------------------------------------------------------------------*/
void LCD2_Display_HYcon(void)
{
    DisplayBuffer[5] = Char_H;
    DisplayBuffer[6] = Char_Y;
    DisplayBuffer[7] = Char_c;
    DisplayBuffer[8] = Char_o;
    DisplayBuffer[9] = Char_n;
}

/*---------------------------------------------------------------------*/
/* Function Name: LCD3_Display_SCALE                                   */
/*---------------------------------------------------------------------*/
void LCD3_Display_SCALE(void)
{
    DisplayBuffer[11] = Char_S;
    DisplayBuffer[12] = Char_C;
    DisplayBuffer[13] = Char_A;
    DisplayBuffer[14] = Char_L;
    DisplayBuffer[15] = Char_E;
}

/*---------------------------------------------------------------------*/
/* Function Name: LCD3_Display_ADC                                     */
/*---------------------------------------------------------------------*/
void LCD3_Display_ADC(unsigned int adc_data)
{
    unsigned char i;

    for(i=0; i<LCD3_DIG_SIZE; i++)
    {
        DisplayBuffer[LCD3_BUF_START-i] = code_seg[adc_data%10];
        adc_data = adc_data/10;
    }

    //Over Range checking
    if ( adc_data )
    {
        for(i=0; i<LCD3_DIG_SIZE; i++)
            DisplayBuffer[LCD3_BUF_START-i] = Char_9;
    }

    DisplayBuffer[17] = 0;                          //clear sign "K".
}

/*---------------------------------------------------------------------*/
/* Function Name: LCD3_Display_PASS                                    */
/*---------------------------------------------------------------------*/
void LCD3_Display_PASS(void)
{
    DisplayBuffer[11] = Char_Blank;
    DisplayBuffer[12] = Char_P;
    DisplayBuffer[13] = Char_A;
    DisplayBuffer[14] = Char_S;
    DisplayBuffer[15] = Char_S;
    DisplayBuffer[17] = 0;                          //clear sign "K".
}

/*---------------------------------------------------------------------*/
/* Function Name: LCD3_Display_CAL                                     */
/*---------------------------------------------------------------------*/
void LCD3_Display_CAL(unsigned char kg_num)
{
    DisplayBuffer[11] = Char_C;
    DisplayBuffer[12] = Char_A;
    DisplayBuffer[13] = Char_L;
    DisplayBuffer[14] = Char_Blank;
    DisplayBuffer[15] = code_seg[ kg_num ];
    DisplayBuffer[17] |= S_K;
    DisplayBuffer[15] |= S_g;
```

```
}

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_Data_Display                                           */
/*---------------------------------------------------------------------------*/
void LCD_Data_Display(unsigned char lcd_no, unsigned int LcdBuffer)
{
    unsigned char Buf_Start, Dig_Size, Point_BIdx, i;

    DisplayBuffer[17] = 0;                          //preclear LCD3's "K" & "TARE" sign bit.
    switch( lcd_no )
    {
        //LCD1
        case LCD_PANEL1:
            Buf_Start = LCD1_BUF_START;
            Dig_Size = LCD1_DIG_SIZE;
            Point_BIdx = LCD1_POINT_BUF;
            break;
        //LCD2
        case LCD_PANEL2:
            Buf_Start = LCD2_BUF_START;
            Dig_Size = LCD2_DIG_SIZE;
            Point_BIdx = LCD2_POINT_BUF;
            break;
        //LCD3
        case LCD_PANEL3:
            Buf_Start = LCD3_BUF_START;
            Dig_Size = LCD3_DIG_SIZE;
            Point_BIdx = LCD3_POINT_BUF;
            break;
        default:
            break;
    }

    for(i=0; i<Dig_Size; i++)
    {
        DisplayBuffer[Buf_Start-i] = code_seg[LcdBuffer%10];
        LcdBuffer = LcdBuffer/10;
    }

    //Over Range checking
    if ( LcdBuffer )
    {
        for(i=0; i<Dig_Size; i++)
            DisplayBuffer[Buf_Start-i] = Char_9;
    }
    //display "g".
    if ( (lcd_no == LCD_PANEL3 ) && ( UserFlag1.b_CalEI_Flag == FALSE ) && ( UserFlag1.b_ShowAD_Flag == FALSE ) )
    {
        DisplayBuffer[17] |= S_K;
        DisplayBuffer[15] |= S_g;
        DisplayBuffer[Point_BIdx] |= seg_h;             //point display.
        if ( UserFlag1.b_TareEI_Flag == TRUE )
            DisplayBuffer[17] |= S_Tare;                //Sign "TARE".
    }
    if ( lcd_no != LCD_PANEL3 )
        DisplayBuffer[Point_BIdx] |= seg_h;             //point display.
}

/*---------------------------------------------------------------------------*/
/* Function Name: RAM2LCD                                                    */
/*---------------------------------------------------------------------------*/
void RAM2LCD( unsigned char *Buffer_Adr )
{
    unsigned char buf_idx;

    for(buf_idx=0; buf_idx<SEG_BUF_SIZE; buf_idx++)
    {
        DrvLCD_WriteData ( buf_idx, *Buffer_Adr );
        Buffer_Adr++;
    }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

6.2. 附加檔案


APD-HY16F021_DemoCode_V01.zip


APD-HY16F021_SCH_V01.pdf

7. 參考文獻

[1] http://www.hycontek.com/attachments/MSP/APD-HY16F007_SC.pdf
紘康科技HY16F188 高精度計價秤
[2] http://www.hycontek.com/attachments/MSP/DS-HY16F198_TC.pdf
紘康科技HY16F198 Datasheet.
[3] http://www.hycontek.com/attachments/MSP/UG-HY16F198_TC.pdf
紘康科技 HY16F198 User Guide.

8. 修訂記錄

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。

日期	文件版次	頁次	摘要
2016/07/27	V01	All	1.初版發行