HY16F198B

觸控溫度計應用說明書

TPS Touch Thermometer

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。

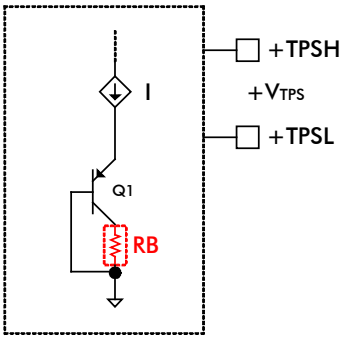
1. 內容簡介

溫度的量測應用非常的廣泛，從農業上的氣溫觀測，及日常防疫的體溫量測至工業上的半導體製程，溫度都是相當重要的一個指標及依據。本文主要是介紹 HYCON HY16F Series 晶片在溫度量測應用，並透過 Touch Key 的介面進行操作。由於 HY16F198B 晶片內部集成高精度∑ΔADC，且 ADC 輸出頻率最快可以到達 10kHz，並搭配內部 LCD 驅動完成顯示。HY16F198B 用於溫度量測，不需外接的感測元件，達到周邊電路簡單且省電的應用.

2. 原理說明

2.1. 量測原理

本應用的溫度量測元件是採用，IC 內部的絕對溫度感測器 TPS，
絕對溫度感測器由二極體(BJT)組成，其電壓信號對溫度的變化為一通過 0°K 曲線，
其具以下特色溫度感測器在環境溫度為 0°K時期輸出的電壓值$V_{TPS@0°K}$ =0V
透過測量方式可使得類比數位轉換器ADC的偏移電壓($V_{ADC-OFFSET}$)與BJT之不對稱性(I_{S1}≠I_{S2})自動抵銷。校正溫度僅需單點校正。



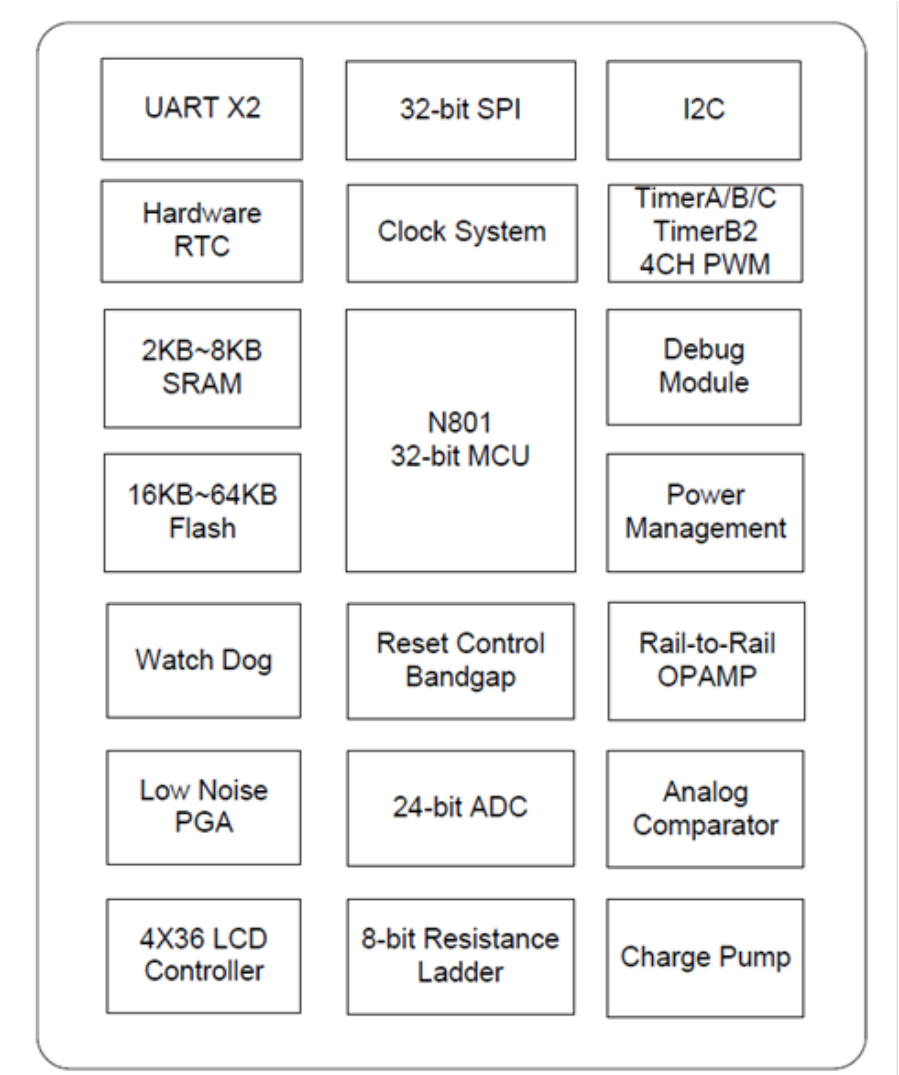
HY16F 啟用時，TPS 的功能隨即被自動啟用。
在同一溫度 TA(℃)下，量測到 VTPS0 與 VTPS1 的數值後，將兩數相加並取平均值即可求得在溫度 TA 下測得 TPS 相對應的電壓值 VTPS@TA。
TPS 的輸出電壓 VTPS 對溫度變化為一線性曲線，故可推倒得出其增益值 GTPS(或稱斜率)

TPS 增益公式

$$G_{TPS}=\left(\frac{V_{TPS@TA}}{273+T_{Offset}+T_A}\right)=\left(\frac{V_{TPS@TA}}{289+T_A}\right)$$

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)



HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)

特點說明:

(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.

(3.1)運行模式 0.6mA@2MHz/2

(3.2)待機模式 5uA@ LSRC=34KHz+IDLE Mode

(3.3)休眠模式 2.5uA

(4)程式記憶體 64KB Flash ROM

(5)資料記憶體 8KB SRAM

(6)擁有 BOR and WDT 功能，可防止 CPU 死機。

(7)24-bit 高精準度ΣΔADC 類比數位轉換器

(7.1)內置 PGA (Programmable Gain Amplifier)最高可達 128 倍放大。

(7.2)內置溫度感測器 TPS。

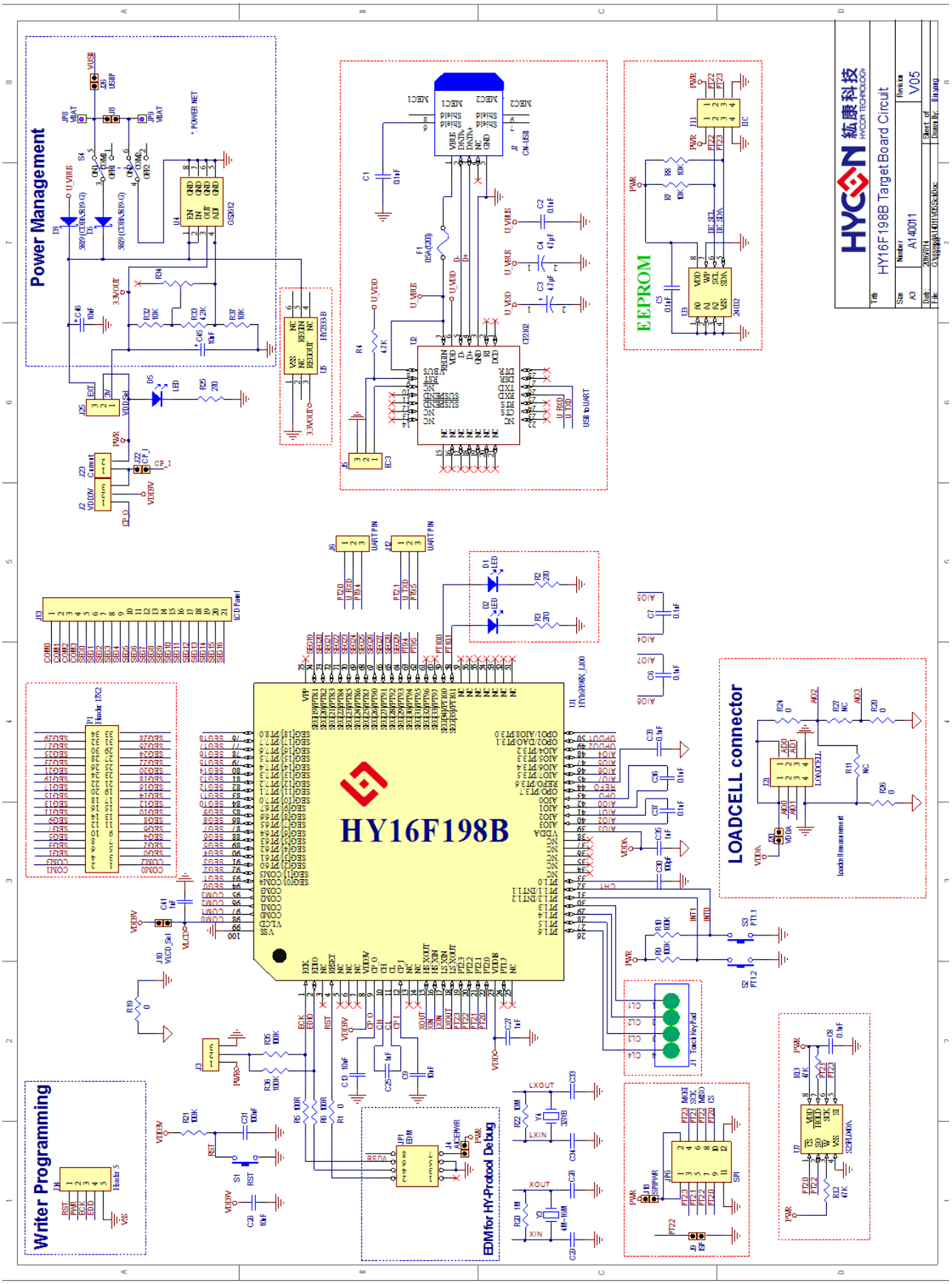
(8)超低輸入雜訊運算放大器 OPAMP。
(9)16-bit Timer A
(10)16-bit Timer B 模組俱 PWM 波形產生功能
(11)16-bit Timer C 模組俱數位 Capture/Compare 功能
(12)硬體串列通訊 SPI 模組
(13)硬體串列通訊 I2C 模組
(14)硬體串列通訊 UART 模組
(15)硬體 RTC 時鐘功能模組
(16)硬體 Touch KEY 功能模組
(17)硬體 LCD Driver 4x36,6x34

3. 系統設計

3.1. 硬體說明

使用HY16F198B對於觸控溫度計的應用，整體電路就只需HY16F開發板上之Touch Key及LCD顯示溫度.

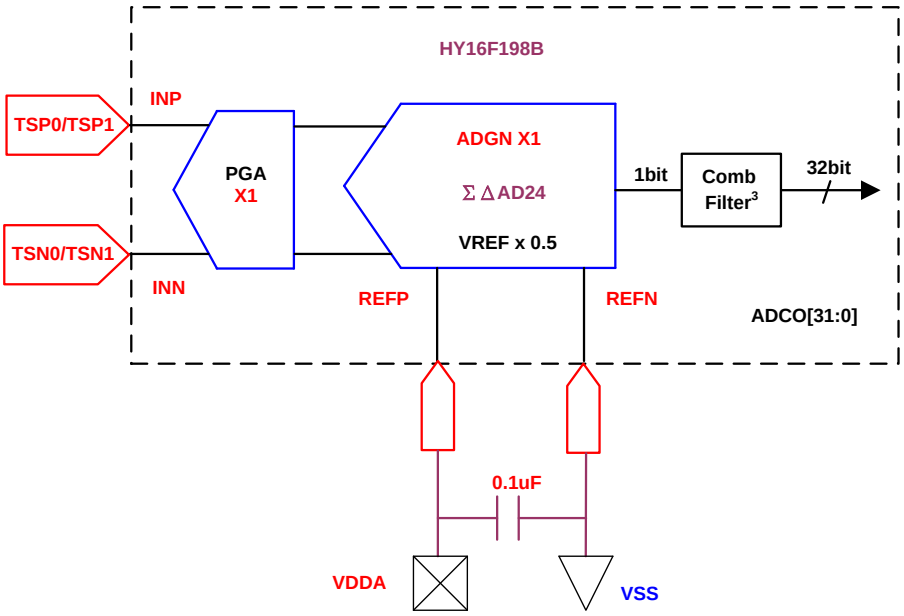
(A) MCU：HY16F198B
(B) 顯示方式： HY16F198B 內部硬體驅動 4x36 LCD (LCD Driver Segment 4X36)
(C) 電源電路：5.0V 轉 3.3V 電源系統
(D) 類比感測模組：內部 ADC
(E) 線上燒錄與 ICE 連結電路，透過 EDM 的連接，可支援線上燒錄模擬.
並擁有強大的 C 平台 IDE 以及 HYCON 類比軟體分析工具與 GUI 等支援.


Power Management
EEPROM
HY16F198B
LOADCELL connector
Writer Programming
EDM for HY-Protool Debug
HY16F198B Target Board Circuit
V05

3.2. 功能說明

3.2.1. 溫度設定

TPS 量測圖: ADC 內部的 PGA 放大 1 倍，ADGN 放大 1 倍，參考電壓由 VDDA –VSS 供給，則 ΔVR_I=1.2V



- TPS 初始化設置與計算方式如下操作：

啓用 ADC 則 TPS 的功能隨即被自動啓用。
本範例程式只使用 VTSP1 / VTSN1 進行 TPS 量測.
測量 TSP0 / TSN0 時，ADINP[3:0]設置<0110>且 ADINN[3:0]設置<0111>
測量 TSP1 / TSN1 時，ADINP[3:0]設置<0111>且 ADINN[3:0]設置<0110>

- 精準的溫度校正(搭配 Copper 方式)進行量測步驟:

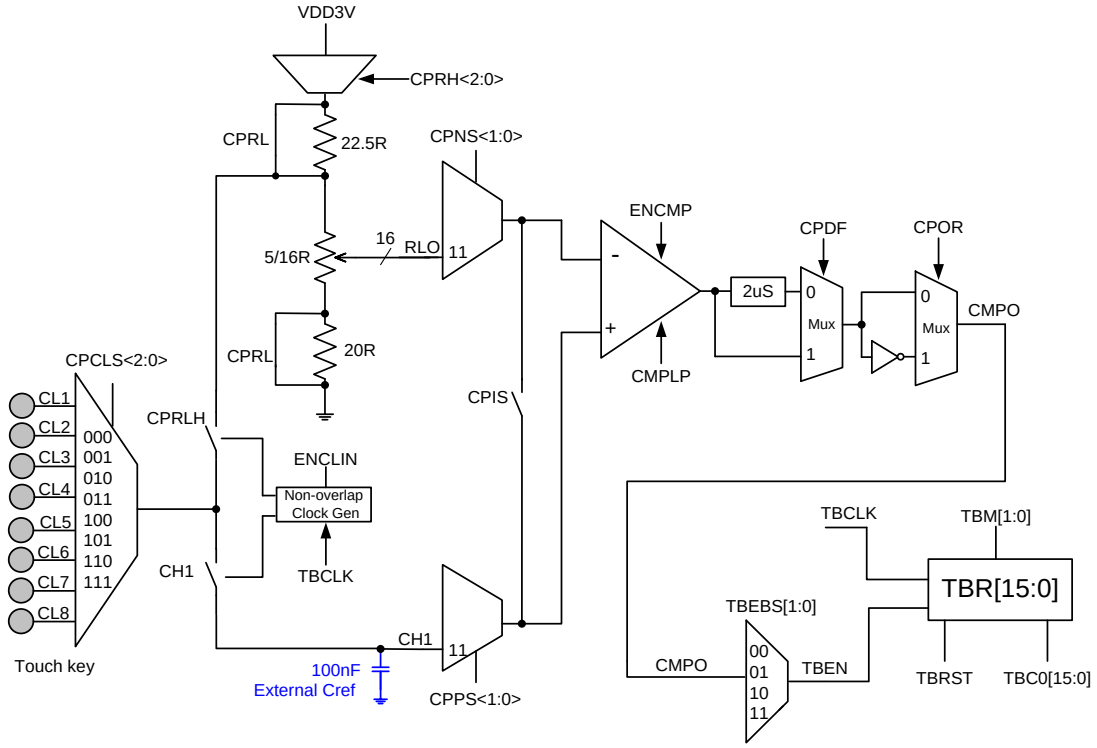
STEP1:
測量得VTSP0 / VTSN0 與VTSP1 / VTSN1 的數值後，在同一溫度TA(℃)下，量測到 V_{TPS0} 與V_{TPS1}的ADC數值.
STEP2:
將兩數相加並取平均值即可求得在溫度TA(℃)下測得TPS相對應的ADC數值($V_{TPS@TA}$)
STEP3:
再帶回TPS增益公式,計算G_{TPS}.

$$G_{TPS} = \left(\frac{V_{TPS@TA}}{273 + T_{Offset} + T_A} \right) = \left(\frac{V_{TPS@TA}}{289 + T_A} \right)$$

STEP4:
最後將ADC平均值(VTS@Ta)除以G_{TPS},求得實際溫度(代入公式)
實際溫度 = (VTS@Ta / G_{TPS}) – (273.15+T_{offset})

3.2.2. 觸控設定

內建硬體觸控模組(使用類比比較器方塊)

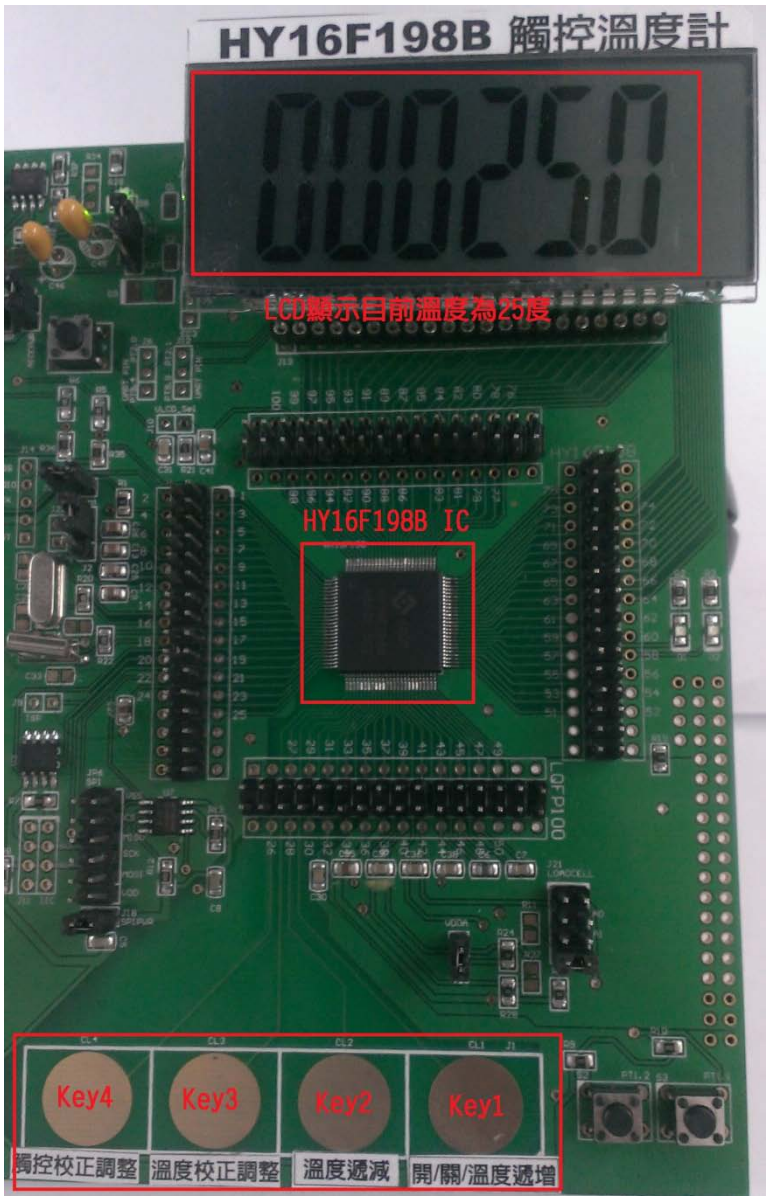


- 觸控原理說明(程式控制方式)：
  STEP1: 將 CMP 的 CPIS(短路)，讓 CH1 上的 Cref 通過 RLO 接到 VSS 進行放電.
  STEP2: 啓動 CMP,預設 CMPO=High,並且讓 Timer B 開始計數.
  STEP3: 啓動 Non-over lap,使 VDD 對 Touch Key(CL1)充電,sharing to CH1,使 CH1 電位慢慢提升,當提升到比較點 RLO 電位時，比較器會轉態(CMPO=Low).
  STEP4: 關閉 Timer B 計數功能(透過 CMPO Flag 判斷關閉 Timer B).
  STEP5: 紀錄 Timer B count,並判斷是否大於門檻值(Yes 表示有按鍵).
  STEP6: 重複放電到充電的循環,依序掃描各 Touch Key(CL1~CL4)

Note:

Cref = 100nF
Charge sharing power= 3V
Non-overlap clock = TBCLK=HAO/4=500KHz
Timer B Enable Flag is CMPO.
RLO=4/16*VDD=0.75V
CPUCLK=HAO;
Comparator: low power

4. 操作流程



4.1. 操作方法

- 上電後，顯示溫度
- Touch Key 功能說明:

  Touch Key1: 開/關/溫度遞增

  Touch Key2: 溫度遞減

  Touch Key3:(進入或離開)溫度校正調整

  Touch Key4: (進入或離開)觸控校正調整

- 溫度校正調整

  STEP1:在顯示溫度情況下，透過 Touch Key3 可進入溫度校正調整畫面.

溫度校正調整畫面

STEP2:在溫度校正調整模式下

- Touch Key1 可遞增溫度.
- Touch Key2 可遞減溫度.
- Touch Key3 長壓(1 秒以上)可離開溫度校正調整模式.

- 觸控校正調整

  在顯示溫度情況下，透過 Touch Key4 可進入觸控校正調整.

  在觸控校正調整模式 LCD 會依序出現如下:

  888888 -> 777777 -> 666666 -> 555555 ->

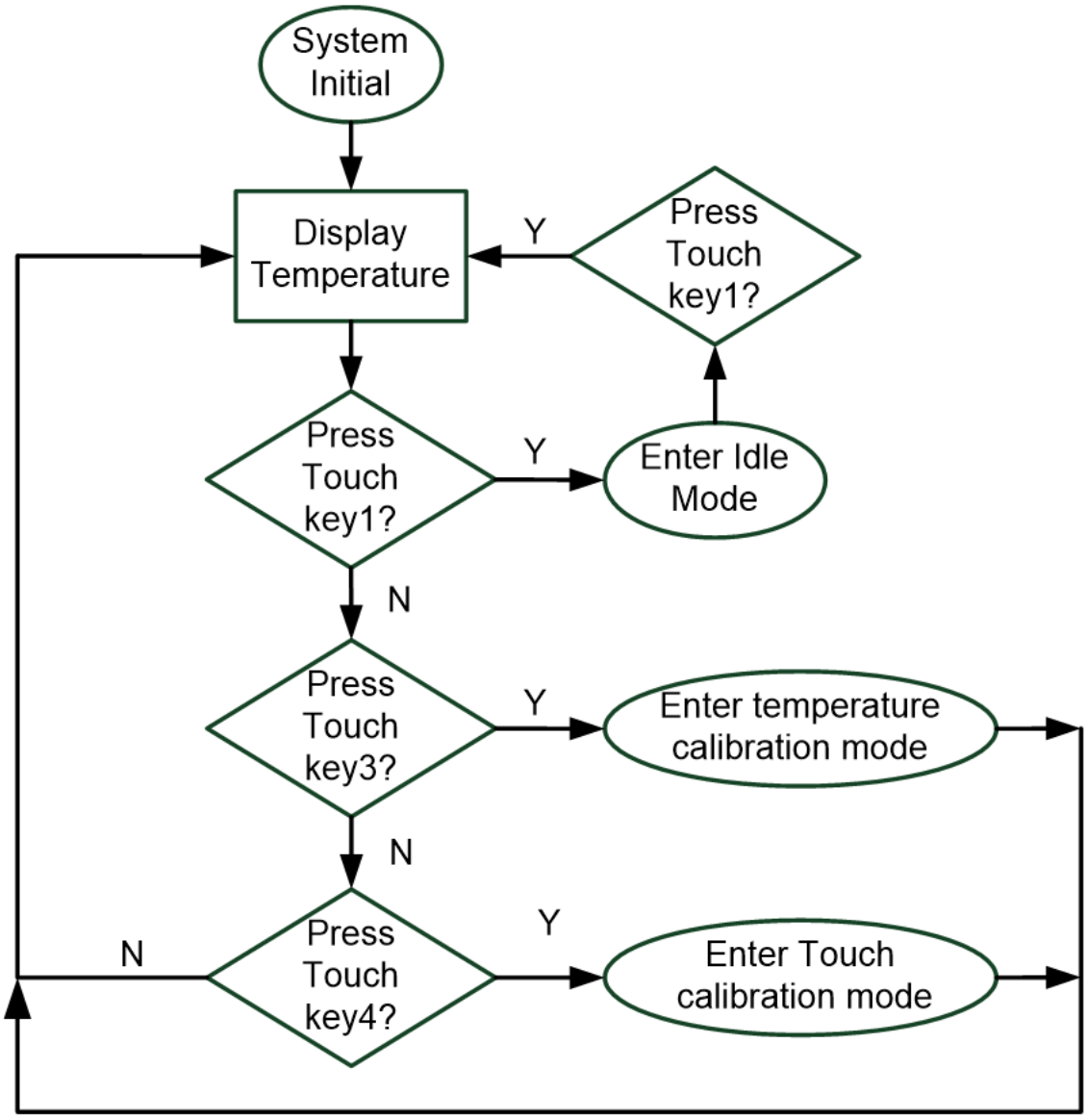
  994444->(請按下 Touch Key1)-> 993333->(請按下 Touch Key2)->

  992222->(請按下 Touch Key3)-> 991111->(請按下 Touch Key4)->

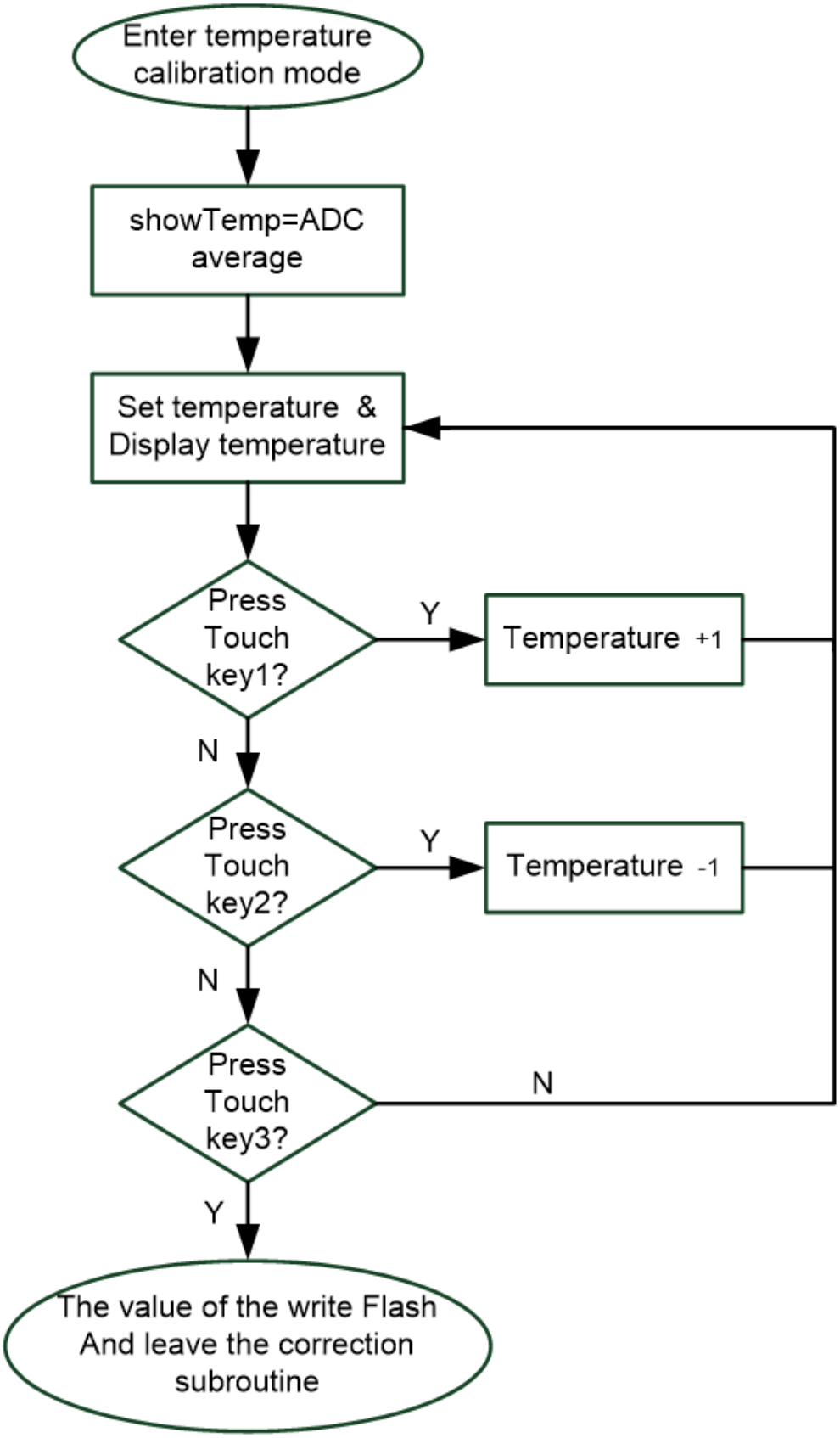
  回到顯示溫度模式.

4.2. 程式流程

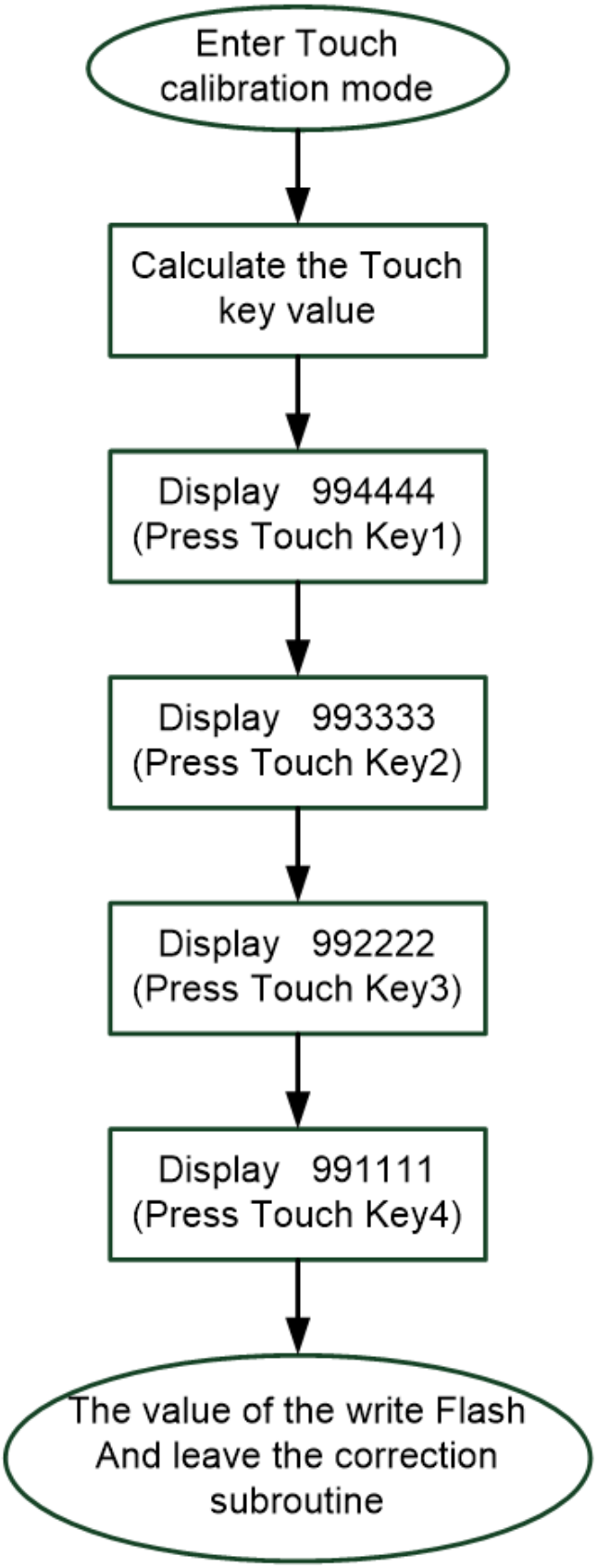
4.2.1. Main Loop 流程圖:



4.2.2. 溫度校正模式流程圖:



4.2.3. 觸控校正模式流程圖:



5. 技術規格

(1) VDD=3.3V
(2) 功耗:工作模式約 2.24mA(HAO=4MHz,ADC Enable)
(3) 適用範圍：各種環境溫度量測
(4) 工作溫度:-40℃~ +85℃
(5) 存貯溫度：-55℃ ～ +125℃；
(6) 相對濕度：＜95%（20±5℃條件）

6. 結果總結

以 HY16F198B 高精准度ΣΔADC,搭配內建的 TPS 溫度感測模式,可大幅減少外部元件,達到環境溫度監控的功能.並使用內建硬體觸控模組(使用類比比較器方塊),完成觸控按鍵功能.

7. 附件

7.1. 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.
 *
 * ----------------------------------------------------------------------------
 * File Name       : main.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 : 2016/05/28
 * Created by     :
 *
 *******************************************************************************/
/*-----------------------------------------------------------------------------*/
/* 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"

#include "DrvCMP.h"
#include "DrvPMU.h"
/*-----------------------------------------------------------------------------*/
/* STRUCTURES                                                                  */
/*-----------------------------------------------------------------------------*/
#if (1)
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;
#endif
/*-----------------------------------------------------------------------------*/
/* Global CONSTANTS                                                            */
/*-----------------------------------------------------------------------------*/
MCUSTATUS   MCUSTATUSbits;

extern unsigned char seg[16];
```

```
/*---------------------------------------------------------------------------*/
/* Constant value definition                                                 */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Function prototype declaration                                            */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* MAIN    function                                                          */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/


/*---------------------------------------------------------------------------*/
/* MAIN    function                                                          */
/*---------------------------------------------------------------------------*/
int main(void)
{
//        unsigned char scan_idx = 0;
      User_Sample_Init();
         //User task initial going ?
      if ( UserFlag0.b_InitGo_Flag == TRUE )
      {
                PowerOn_Init_Go();
        }
//------Enter Main program loop-----------------
      while(1)
      {
                User_Sample_Process();
                Key_In_Process();
      }
      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.        */
/* Created Date : 2016/05/28                                                 */
/* Created by     :                                                          */
/*****************************************************************************/
#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW0_ISR()                                                  */
/* Description    : I2C/UART/SPI interrupt Service Routine (HW0).            */
/* Arguments       : None.                                                   */
/* Return Value : None.                                                      */
/* Remark           :                                                        */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{

}
#endif

#if (1)
/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description    : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).  */
/* Arguments       : None.                                                   */
/* Return Value : None.                                                      */
/* Remark           :                                                        */
```

```
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{

//------TIMER A Int ?----------------------------
      DrvTIMER_ClearIntFlag(E_TMA);              //Clear TMA interrupt flag
      UserFlag0.b_TB8mS_Flag = TRUE;
      t=1;
}
#endif



#if (1)
/*---------------------------------------------------------------------------*/
/* Function Name: HW2_ISR()                                                  */
/* Description   : ADC interrupt Service Routine (HW2).                      */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{
        if(DrvADC_ReadIntFlag()==1) //讀取 ADC 插斷要求標誌位元
        {
          DrvADC_ClearIntFlag();
       ADCData=DrvADC_GetConversionData()>>14;    //32bits>>14, only get 18bit ADC data
       //ADCData=DrvADC_GetConversionData()>>12;    //32bits>>12, only get 20bit ADC data
       //ADCData=DrvADC_GetConversionData();
       ADfinish=1;
        }

}
#endif

#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW3_ISR()                                                  */
/* Description   : CMP & OPA interrupt Service Routine (HW3).                */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{
      DrvCMP_ClearIntFlag();                     //Clear CMP interrupt flag
}
#endif

#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW4_ISR()                                                  */
/* Description   : PT1 interrupt Service Routine (HW4).                      */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{
}
#endif

#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW5_ISR()                                                  */
/* Description   : PT2 interrupt Service Routine (HW5).                      */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW5_ISR(void)
{

}
#endif
```

```
#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW7_ISR()                                                  */
/* Description   : UART2 interrupt Service Routine (HW7).                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW7_ISR(void)
{

}
#endif

#if (0)
/*---------------------------------------------------------------------------*/
/* Function Name: HW8_ISR()                                                  */
/* Description   : Timer B2 interrupt Service Routine (HW8).                 */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void HW8_ISR(void)
{
    if( DrvTIMER_GetIntFlag (E_TMB2) )
    {
        DrvTIMER_ClearIntFlag(E_TMB2);          // Clear TMB interrupt flag
    }
}
#endif

/*---------------------------------------------------------------------------*/
/* 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"

#include "DrvCMP.h"
#include "DrvPMU.h"
/*---------------------------------------------------------------------------*/
/* Structure definition                                                      */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Global variable definition                                                */
/*---------------------------------------------------------------------------*/

unsigned int    Sleep_Time_Cnt;
unsigned char DisplayBuffer[18];
unsigned char ADfinish;

#define    Disable 0
#define    Enable    1
int threh0,threh1,threh2,threh3;                                    //各touch鍵 閥值
int CL,t,a,b,TCH,cmpresult,tmrbcnt;
signed int sum,ADtemp10,LED,ADCData,c,average;//ADC用
float gain,temp,tempp,temploop,showTemp;              //溫度計算用
unsigned int unth0,unth1,unth2,unth3,th0,th1,th2,th3;
int value_buf[16];
int ADC_Array[2];
```

```
int i,ADCData_Chopper;
/*---------------------------------------------------------------------------*/
/* Constant value definition                                                 */
/*---------------------------------------------------------------------------*/
unsigned char seg[]=
{
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f, // char "0"
    seg_b+seg_c,                         // char "1"
    seg_a+seg_b+seg_d+seg_e+seg_g,       // char "2"
    seg_a+seg_b+seg_c+seg_d+seg_g,       // char "3"
    seg_b+seg_c+seg_f+seg_g,             // char "4"
    seg_a+seg_c+seg_d+seg_f+seg_g,       // char "5"
    seg_a+seg_c+seg_d+seg_e+seg_f+seg_g, // char "6"
    seg_a+seg_b+seg_c,                   // char "7"
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f+seg_g,   // char "8"
    seg_a+seg_b+seg_c+seg_d+seg_f+seg_g, // char "9"
    seg_a+seg_b+seg_c+seg_e+seg_f+seg_g, // char "A"
    seg_c+seg_d+seg_e+seg_f+seg_g,       // char "b"
    seg_a+seg_d+seg_e+seg_f,             // char "C"
    seg_b+seg_c+seg_d+seg_e+seg_g,       // char "d"
    seg_a+seg_d+seg_e+seg_f+seg_g,       // char "E"
    seg_a+seg_e+seg_f+seg_g              // char "F"
};

/*---------------------------------------------------------------------------*/
/* Function prototype declaration                                            */
/*---------------------------------------------------------------------------*/

//------MISC Function declaration---------------
void User_Sample_Init(void);
void Delay(unsigned int num);
void PowerOn_Init_Go(void);
void Enter_SLEEP_Mode(void);
void On_Mode_Display(void);
void Key_In_Process(void);
//------ADC Function declaration----------------
void ADC_Init(void);

//------GPIO Function declaration---------------

//------TIMER Function declaration--------------

//------LCD Function declaration----------------
void LCD_Init(void);
void LCD_DATA_DISPLAY(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer);
void RAM2LCD(unsigned char *Buffer_Adr );
void ClearLCDframe(void);
void DisplayHYcon(void);

void Ini_Touch(void);
void Initial_ALL(void);
void touch_ct(void);
void ScanKey4(unsigned int TCH);
void User_Sample_Process(void);
void AD(void);       //TPS0 buffer -
void temp_Process(void);
void temp_ct(void);

/*---------------------------------------------------------------------------*/
/* Function Name: User_Sample_Init                                           */
/* Description   : Sample code Initialization Subroutines                    */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void User_Sample_Init(void)
{
    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--------------------
    LCD_Init();
```

```
    UserFlag0._byte = 0;
//------Power on initial going-----------------
//Beep tone on & LCD display HYcon SCALE about 1Sec.

    DisplayHYcon();
    UserFlag0.b_InitGo_Flag = TRUE;
}


void User_Sample_Process(void)
{
    if(t==1)                        //(t=1代表TMA已進入中斷)
    {
        t=0;                        //喚醒CPU
        pmu_00=0x03031b1a;          //開啟比較器功能(開啟touch功能)
        cmp_00=0xff0f;
        cmp_04=0xff01ff03;
        clk_00=0xff01;              //開啟高速
        clk_08=0xffcc;
        TCH=0;
    }
    t=0;
}


void Key_In_Process(void)
{
        ScanKey4(TCH);              //掃描touch1是否被按下，touch1被按下LED+1
        if(LED==1)
        {
                DrvLCD_DisplayMode( E_LCD_NORMAL );
            DrvLCD_VLCDTrim( 3 );

            while(1)
            {

                if(LED==15)                             //tocuh1長時間備按住當LED加至15時，離開溫度顯示，再次進入省電狀態
                {
                    LED=0;
                    ClearLCDframe();
                    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.
                            Enter_SLEEP_Mode();
                            break;
                }

                TCH=3;
                ScanKey4(TCH);                  //溫度顯示下，如果touch4按下，進入touch校正模式(touch4按下 LED=10000)
                if(c==-15)
                {
                    touch_ct();             //呼叫touch校正負程式
                }
                TCH=2;
                ScanKey4(TCH);                  //當touch2按下時，LED會減1，當LED減至-15時，進入溫度校正
                if(c==15)
                {
                    temp_ct();              //呼叫溫度校正副程式
                }
                //Delay(0x1000);
                TCH=0;
                ScanKey4(TCH);
                        temp_Process();
            }
        }
}

void temp_Process(void)
{
        DrvADC_Enable();                //開啟ADC功能
    AD();                               //ADC副程式
       gain=(273.15+tempp)/temp;        //溫度換算
       temploop=(gain*showTemp)-273.15;
       temploop*=10;
       LCD_DATA_DISPLAY1(temploop);     //顯示現在溫度
```

```
}

void temp_ct(void)                                          //溫度校正副程式
{
        AD();
        LCD_DATA_DISPLAY(showTemp);
        Delay(0xFFF0);
        Delay(0xFFF0);
      c=0;
        temp=showTemp;                                      //環境ADC值
      LED=tempp;                                            //設定現在溫度
      while(1)
      {
          LCD_DATA_DISPLAY(LED);
          Delay(0xA000);
          TCH=0;
          ScanKey4(TCH);                                //touch1=溫度上升一度
          TCH=1;
          ScanKey4(TCH);                                //touch2=溫度下降一度
          TCH=2;
          ScanKey4(TCH);                                //touch3確定鍵
          if(c==3)
          {
                ClearLCDframe();                            //清除LCD
                c=0;
                tempp=LED;                                      //將剛剛設定值放入溫度運算暫存器
                DrvADC_DisableInt();                        //關閉ADC中斷
                DrvTIMER_DisableInt(E_TMA);                 //關閉TMA中斷
                DrvFlash_Burn_Word(0xA014,0x8000,0x1110);       //將判斷是否校正過用數值燒入Flash
                DrvFlash_Burn_Word(0xA018,0x8000,tempp);        //將校正值燒入Flash
                DrvFlash_Burn_Word(0xA01C,0x8000,temp);         //環境ADC值
                Delay(0xff00);
                DrvTIMER_EnableInt(E_TMA);                  //開啟TMA中斷
                DrvADC_EnableInt();                             //開起ADC中斷
                LED=0;
                break;
          }
      }
}

void AD(void)                                                       //TPS0副程式
{

        while(ADfinish)                                                     //ADC中斷後OK=1
        {
                ADfinish=0;
                ADtemp10=ADCData;
                value_buf[11]=value_buf[10];
                value_buf[10]=value_buf[9];
                value_buf[9]=value_buf[8];
                value_buf[8]=value_buf[7];                  //平均滑動濾波處理
                value_buf[7]=value_buf[6];
                value_buf[6]=value_buf[5];
                value_buf[5]=value_buf[4];
                value_buf[4]=value_buf[3];
                value_buf[3]=value_buf[2];
                value_buf[2]=value_buf[1];
                value_buf[1]=ADtemp10;
                sum=0;
                for(t=9; t>5; t--)
                {
                        sum+=value_buf[t];
                }
                average=((sum)>>2);
        }
        if(average<0)
        {
                average*=-1;
        }
        showTemp=average;
}
```

```
/*---------------------------------------------------------------------------*/
/* Function Name: PowerOn_Init_Go                                            */
/* Description   : Initial all user tasks (likes LCD, ADC for scale etc),    */
/* Arguments     : None.                                                     */
/* Return Value : None.                                                      */
/* Remark        :                                                           */
/*---------------------------------------------------------------------------*/
void PowerOn_Init_Go(void)
{
    Ini_Touch();
      ADC_Init();
    Initial_ALL();
    a=ReadWord(0xA014);                         //判斷溫度是否被校正過，如果被校正過抓Flash內值
    if(a==0x1110)
    {
        temp=ReadWord(0xA01C);
        tempp=ReadWord(0xA018);
    }
    else                                        //如果判斷為沒校正過，抓原先內部設定值
    {
        temp=6047;
        tempp=25;
    }

    a=ReadWord(0xA000);                         //判斷touch是否被校正過，如果被校正過抓Flash內值
    if(a==0x5566)
    {
        threh0=ReadWord(0xA004);
        threh1=ReadWord(0xA008);
        threh2=ReadWord(0xA00c);
        threh3=ReadWord(0xA010);
    }
    else                                        //如果判斷為沒校正過，抓原先內部設定值
    {
        touch_ct();

    }

      DrvADC_Enable();                  //開啟ADC功能

      for(i=0;i<=1;i++)
            {
                switch (i)
                {
                 case 0:   //已經抓到第一筆
                 while(!ADfinish);   //if ADfinish=0, stop at here, until enter ADC INT
                 ADfinish=0;
                 ADC_Array[0]=ADCData;
                 DrvADC_Disable();
                 DrvADC_CombFilter(Disable);   //Disable CombFilter
                 //DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);
                    DrvADC_SetADCInputChannel(TPS0,TPS1);
                 DrvADC_Enable();
                 DrvADC_CombFilter(Enable);   //Disable CombFilter
                 DrvADC_ClearIntFlag();
                    DrvADC_EnableInt();
                 while(!ADfinish);   //if ADfinish=0, stop at here
                 break;

                 case 1:   //已經抓到第二筆
                 ADfinish=0;
                 ADC_Array[1]=ADCData;//>>16;
                 DrvADC_Disable();
                 DrvADC_CombFilter(Disable);   //Disable CombFilter
                 ADCData_Chopper=(ADC_Array[0]-ADC_Array[1])>>1;         //divider by 2
                    DrvADC_SetADCInputChannel(TPS1,TPS0);
                 DrvADC_Enable();
                 DrvADC_CombFilter(Enable);   //Disable CombFilter
                 DrvADC_ClearIntFlag();
                 DrvADC_EnableInt();
                 while(!ADfinish);        //if ADfinish=0, stop at here
                 break;
                }
            }

      DrvADC_Disable();
```

```
        for(i=0; i<16; i++)
        {
                value_buf[i] = ADCData_Chopper;
        }

        for(t=4; t>0; t--)
                {
                        sum+=value_buf[t];
                }
        average=((sum)>>2);
        if(average<0)
        {
                average*=-1;
        }
        showTemp=average;


        gain=(273.15+tempp)/temp;               //溫度換算
        temploop=(gain*showTemp)-273.15;
        temploop*=10;

    UserFlag1._byte = 0;
    UserFlag0.b_InitGo_Flag = FALSE;
}

/*---------------------------------------------------------------------------*/
/* Function Name: Enter_SLEEP_Mode                                           */
/* Description   : Prepare to enter SLEEP mode.                              */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark          :                                                         */
/*---------------------------------------------------------------------------*/
void Enter_SLEEP_Mode(void)
{
    pmu_00=0xff00ff00;                  //將比較器功能關閉(省電)
    cmp_00=0xff00;
    cmp_04=0xff00ff00;
    asm("NOP");
    ClearLCDframe();
    DrvLCD_DisplayMode( E_LCD_PIXELOFF );       //All LCD off.
    DrvLCD_VLCDMode( E_VLCD_DISABLE );          //close VLCD.
    while( (inw(0x41B00) & (1<<IDF) ) == 0);
    clk_08=0xff00ff0d;                                  //關閉高速震盪器，使用低速(省電)
    Delay(0x100);
    clk_00=0xff00ff00;

    //sys_04=0xff10;
    asm("syscall 11"); // idle = 11
    //asm("standby 0");
    asm("NOP");
    //DrvADC_Disable();                         //關閉ADC功能(省電)
}


/*---------------------------------------------------------------------------*/
/* Function Name: Delay()                                                    */
/* Description   : Software delay subroutines.                               */
/* Arguments      : unsigned int num.                                        */
/* Return Value : None.                                                      */
/* Remark          : num= 10,     delay time= 18u S @4MH HAO, 3V.            */
/*                   : num= 100,    delay time= 40u S @4MH HAO, 3V.          */
/*                   : num= 1000, delay time= 260uS @4MH HAO, 3V.            */
/*                                                                           */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for( ; num >0; num-- )
        asm( "NOP" );
}

/*****************************************************************************/
/* File Name      : user_GPIO.c                                              */
/* Description   : This file implements the customer's GPIO function.        */
/* Created Date : 2016/05/28                                                 */
/* Created by     :                                                          */
```

```
/*****************************************************************************/


/*****************************************************************************/
/* File Name       : user_ADC.c                                              */
/* Description    : This file implements the customer's ADC function.        */
/* Created Date : 2016/05/28                                                 */
/* Created by     :                                                          */
/*****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Function Name: ADC_Init                                                   */
/* Description    : ADC Initialization Subroutines                           */
/* Arguments       : None.                                                   */
/* Return Value : None.                                                      */
/* Remark           :                                                        */
/*---------------------------------------------------------------------------*/
void ADC_Init(void)
{
        unsigned int i;
#if (1)
      //set ADC input pin
         DrvADC_PInputChannel(7);                    //set ADC positive input pin is TSP1.
      DrvADC_NInputChannel(6);                   //set ADC negative input pin is TSN1.

      DrvADC_InputSwitch( OPEN );                  //set ADC input short-switch is open
      DrvADC_RefInputShort( OPEN );                //set VREF input short-switch is open
      DrvADC_Gain( ADC_PGA_Disable, ADC_ADGN_1 ); //set ADC    GAIN is PGA * ADGN = 1x1 = 1.

      //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 pin(REFP) is VDDA, negative input pin(REFN) is
VSSA.
      DrvADC_FullRefRange( 1 );                   //set VREF gain = 1/2 VREF= 2.4V/2= 1.2V
//    DrvADC_RefVoltage ( REF_BUFFER_OUT, VSSA ); //set VREF positive input pin(REFP) is REFO_I, negative input pin(REFN) is
VSSA.
//    DrvADC_FullRefRange( 0 );                   //set VREF gain = 1 VREF= 1.2V

      //set ADC comb filter
      DrvADC_OSR( 0 );                            //set OSR= ADCK/32768 = (4MHz/12)/32768 = 10 sps.
      //DrvADC_OSR( 1 );                          //set OSR= ADCK/16384 = (4MHz/12)/16284 = 20 sps.
      DrvADC_CombFilter( 1 );                     //enable Comb filter.

      //set ADC clock
      DrvADC_ClkEnable( 1,1 );                    //set ADC CLOCK ADCK = HS_CK/12 = 2MHz/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_BandgapEnable();                  //band gap voltage
         DrvPMU_REFO_Enable();                    //enable REFO 1.2V
      DrvPMU_AnalogGround( 1 );                   //set analog enable buffer and use internal source
      DrvPMU_LDO_LowPower(0);                                  // bin    VDD LDO with low power control.
         Delay( 0x1000 );

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

         DrvADC_CombFilter( 0 );                  //reset !
         DrvADC_CombFilter( 1 );                  //enable !
         for(i=0; i<16; i++)
         {
                 value_buf[i] = 0;
         }

         ADfinish=0;
         ADCData_Chopper=0;
         for(i=0;i<=1;i++)
         {
                 ADC_Array[i]=0;
```

```
        }

#endif
}



/*****************************************************************************/
/* File Name       : user_TIMER.c                                            */
/* Description    : This file implements the customer's TIMER unction.       */
/* Created Date : 2016/05/28                                                 */
/* Created by     :                                                          */
/*****************************************************************************/


/*****************************************************************************/
/* File Name       : user_LCD.c                                              */
/* Description    : This file implements the customer's LCD function.        */
/* Created Date : 2016/05/28                                                 */
/* Created by     :                                                          */
/*****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_Init                                                   */
/* Description    : LCD Initialization Subroutines                           */
/* Arguments       : None.                                                   */
/* Return Value : None.                                                      */
/* Remark           : 1. LCD working@ 4COM x 36SEG, VLCD= from charge pump 2.93V. */
/*                    : 2. Charge pump's clock from HS_CK/16= 4M/16= 250KHz).  */
/*                    : 3. LCD CLK(LCK) from (HS_CK/64)/9/16= 4M/9216= 434Hz.  */
/*                    :       so LCD frame Freq. = 434Hz/(4x2)= 54Hz.         */
/*---------------------------------------------------------------------------*/
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, 0x07 );                     //Set PT7(SEG10~SEG12) as LCD mode.
      DrvLCD_LCDBuffer( ENABLE );                    //enable LCD working.
      DrvLCD_DisplayMode( E_LCD_NORMAL );            //LCD at normal mode.
      ClearLCDframe();
}

/*---------------------------------------------------------------------------*/
/* Clear LCD RAM Data                                                        */
/*---------------------------------------------------------------------------*/
void ClearLCDframe(void)
{
        int i=0;
        for(i=0; i<18; i++)
        {
                DisplayBuffer[i]=0x00;
        }
        RAM2LCD( DisplayBuffer );
}

/*---------------------------------------------------------------------------*/
/* Display HYcon Char                                                        */
/*---------------------------------------------------------------------------*/
void DisplayHYcon(void)
{
        DisplayBuffer[0]=0x00;
        DisplayBuffer[1]=Char_H;
        DisplayBuffer[2]=Char_Y;
        DisplayBuffer[3]=Char_c;
        DisplayBuffer[4]=Char_o;
        DisplayBuffer[5]=Char_n;
        DisplayBuffer[6]=0x00;
        DisplayBuffer[7]=0x00;
        DisplayBuffer[8]=0x00;
        RAM2LCD( DisplayBuffer );
```

```
}


/*---------------------------------------------------------------------------*/
/* Display RAM Data                                                          */
/*---------------------------------------------------------------------------*/
void LCD_DATA_DISPLAY(unsigned int LcdBuffer)
{
        DisplayBuffer[0]=0x00;
        DisplayBuffer[1]=0x00;
        DisplayBuffer[2]=0x00;
        DisplayBuffer[3]=0x00;
        DisplayBuffer[4]=0x00;
        DisplayBuffer[5]=0x00;
        int i;
        for(i=5; i>=0; i--)
        {
                DisplayBuffer[i]=seg[LcdBuffer%10];
                LcdBuffer=LcdBuffer/10;
        }
        RAM2LCD( DisplayBuffer );
}

/*---------------------------------------------------------------------------*/
/* Display RAM Data                                                          */
/*---------------------------------------------------------------------------*/
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer)
{
        DisplayBuffer[0]=0x00;
        DisplayBuffer[1]=0x00;
        DisplayBuffer[2]=0x00;
        DisplayBuffer[3]=0x00;
        DisplayBuffer[4]=0x00;
        DisplayBuffer[5]=0x00;
        int i;
        for(i=5; i>=0; i--)
        {
                DisplayBuffer[i]=seg[LcdBuffer%10];
                LcdBuffer=LcdBuffer/10;
        }
        DisplayBuffer[4]=DisplayBuffer[4]|seg_h;          //小數點
        RAM2LCD( DisplayBuffer );
}

/*---------------------------------------------------------------------------*/
/* Function Name: RAM2LCD                                                    */
/* Description    : RAM buffer data transfer to LCD RAM                      */
/* Arguments      : unsigned char *Buffer_Adr => buffer pointer address.     */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
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++;
        }
}


void Initial_ALL(void)
{
      pio1_1=0xff00ff00;              //PT1  0x40800      //確保沒用的腳位關閉
      pio1_2=0xff00ff00;              //              0x40804
      pio1_3=0x00;                    //              0x40808
      pio2_1=0xff00ff03;              //PT2  0x40810
      pio2_2=0xff00ff03;              //              0x40814

      LED=0;
      t=0;
      TCH=0;
      c=0;
```

```
    DrvTMA_Open(9,1);                                   //TimerA Overflow
                                                                //9:taclk/1024/32;TMRDV=÷32
                                                                //00:LS_CK
    DrvTIMER_ClearIntFlag(E_TMA);                       //Clear Timer A interrupt flag
    DrvTIMER_EnableInt(E_TMA);                                  //Timer A interrupt enable
    SYS_EnableGIE(4,0x06 );                                     //Enable GIE
}


void Ini_Touch(void)
{
    DrvCMP_Enable();                                    // CMP enable
    DrvCMP_Open(1,1,1);                                 // CMP open
    DrvCMP_PInput(0);                                   // Comparator positive input    CH1
    DrvCMP_NInput(3);                                   // Comparator negative input    RLO
    DrvCMP_RLO_refV(2,1);                                       // RLO Ref = VDD3V , CPRL=1
    DrvCMP_DisableNonOverlap();                                 // disable Nonoverlap
    DrvCMP_RLO_Ctrl(0,1);                                       // 0 , uCPDM=1 (enable)
    DrvCMP_InputSwitch(1);                                      // CPIS short
    DrvCMP_InputSwitch(0);                                      // CPIS open
    DrvCMP_RLO_Ctrl(1,0);                                       // 1/16 (CPRLH – CPRLL) uCPDM=0 (disable)
    DrvTMB_ClearTMB();                                                  // clear TimerB
    DrvCLOCK_SelectIHOSC(0x00);                                 // HAO =2MHz
    DrvCLOCK_SelectMCUClock(0,0);                       // select MCU clock as HS_CK,div1
    DrvTMBC_Clk_Source(1,0);                            // TimerB Clock enable pre_scale 1 (LS_ck)
    DrvTMB_Open(E_TMB_MODE0,E_TMB_CMP_HIGH,0xffff); // TimerB overflow 0xffff-> CMP High Level trigger
    DrvTMB_ClearTMB();                                                  // clear TimerB
    DrvPMU_VDDA_Voltage(E_VDDA2_4);                             // VDDA=2.4V
    DrvPMU_VDDA_LDO_Ctrl(E_LDO);                                // VDDA LDO，VDDA可調輸出，由VDAS決定
    DrvPMU_BandgapEnable();                                     // ebable bandgap
    DrvPMU_REFO_Enable();                                       // REFO enable
    DrvPMU_AnalogGround(Enable);                        //ADC analog ground source selection.
                                                                //1 : Enable buffer and use internal source(need
to work with ADC)
    DrvPMU_LDO_LowPower(Enable);                        //VDD LDO with low power control.//0 : Normal;1 : Low power
    Delay(0x1000);
}

void touch_ct(void)                                             //touch key校正副程式
{
    int z,y;

    LCD_DATA_DISPLAY(888888);
    unth0=0;                                                    //抓取touch1 untouch值(8筆取平均)
    for(z=0; z<8; z++)
    {
        TCH=0;
        ScanKey4(TCH);
        unth0+=tmrbcnt;
        Delay(0xFFF);
    }
    unth0=unth0>>3;
    LCD_DATA_DISPLAY(777777);
    unth1=0;                                                    //抓取touch2 untouch值(8筆取平均)
    for(z=0; z<8; z++)
    {
        TCH=1;
        ScanKey4(TCH);
        unth1+=tmrbcnt;
        Delay(0xFFF);
    }
    unth1=unth1>>3;
    LCD_DATA_DISPLAY(666666);
    unth2=0;                                                    //抓取touch3 untouch值(8筆取平均)
    for(z=0; z<8; z++)
    {
        TCH=2;
        ScanKey4(TCH);
        unth2+=tmrbcnt;
        Delay(0xFFF);
    }
    unth2=unth2>>3;
    LCD_DATA_DISPLAY(555555);
    unth3=0;                                                    //抓取touch4 untouch值(8筆取平均)
    for(z=0; z<8; z++)
    {
```

```
        TCH=3;
        ScanKey4(TCH);
        unth3+=tmrbcnt;
        Delay(0xfff);
    }
    unth3=unth3>>3;

    LCD_DATA_DISPLAY(994444);
    th0=0;                                          //抓取touch1 touch值(8筆取平均)
    y=8;
    while(y)
    {
        TCH=0;
        ScanKey4(TCH);
        if(tmrbcnt<435)
        {
            y--;
            th0+=tmrbcnt;
        }
    }
    th0=th0>>3;
    LCD_DATA_DISPLAY(993333);
    th1=0;                                          //抓取touch2 touch值(8筆取平均)
    y=8;
    while(y)
    {
        TCH=1;
        ScanKey4(TCH);
        if(tmrbcnt<220)
        {
            y--;
            th1+=tmrbcnt;
        }
    }
    th1=th1>>3;
    LCD_DATA_DISPLAY(992222);
    th2=0;                                          //抓取touch3 touch值(8筆取平均)
    y=8;
    while(y)
    {
        TCH=2;
        ScanKey4(TCH);
        if(tmrbcnt<180)
        {
            y--;
            th2+=tmrbcnt;
        }
    }
    th2=th2>>3;
    LCD_DATA_DISPLAY(991111);
    th3=0;                                          //抓取touch4 touch值(8筆取平均)
    y=8;
    while(y)
    {
        TCH=3;
        ScanKey4(TCH);
        if(tmrbcnt<180)
        {
            y--;
            th3+=tmrbcnt;
        }
    }
    th3=th3>>3;
    threh0=unth0-(((unth0-th0)*3)/4);               //計算校正值
    threh1=unth1-(((unth1-th1)*3)/4);
    threh2=unth2-(((unth2-th2)*3)/4);
    threh3=unth3-(((unth3-th3)*3)/4);
    LED=0;
    c=0;
    DrvADC_DisableInt();                            //關閉ADC中斷
    DrvTIMER_DisableInt(E_TMA);                     //關閉TMA中斷
    DrvFlash_Burn_Word(0xA000,0x8000,0x5566);       //將判斷是否校正過用值，存入Flach
    DrvFlash_Burn_Word(0xA004,0x8000,threh0);       //將校正值存入Flash
    DrvFlash_Burn_Word(0xA008,0x8000,threh1);
    DrvFlash_Burn_Word(0xA00c,0x8000,threh2);
```

```
    DrvFlash_Burn_Word(0xA010,0x8000,threh3);
    Delay(0x8000);
    DrvTIMER_EnableInt(E_TMA);                          //開啟TMA中斷
    DrvADC_EnableInt();                                 //開啟ADC中斷
}

void ScanKey4(unsigned int TCH)                         //touch key副程式
{
    CL=TCH;
    pio1_1=0x0100;
    DrvTMB_ClearTMB();
    DrvCMP_EnableNonOverlap(CL);
    cmpresult=DrvCMP_ReadData();
    while(cmpresult==1)
    {
        cmpresult=DrvCMP_ReadData();
    }
    tmrbcnt=DrvTMB_CounterRead();                       //判斷充電時間是否低於閥值
    switch(CL)
    {
    case 0 :
        if(tmrbcnt<=threh0)
        {
            LED++;
        }
        break;
    case 1 :
        if(tmrbcnt<=threh1)
        {
            LED--;
        }
        break;
    case 2 :
        if(tmrbcnt<=threh2)
        {
            c++;
        }
        break;
    case 3 :
        if(tmrbcnt<=threh3)
        {
            c--;
        }
        break;
    default:
        LED=0;
    }
    pio1_1=0x0101;
    pio1_2=0x0100;

}

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

7.2. 附加檔案

APD-HY16F024_DemoCode_V01.zip

8. 參考文獻

[1] http://www.hycontek.com/attachments/MSP/APD-HY16F010_TC.pdf
紘康科技HY16F188 觸控溫度計
[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.

9. 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
2016/08/05	V1.0	ALL	初版發行