觸控溫度計_應用說明書

HY16F188

TPS Touch Thermometer

目　錄

1.內容簡介 4
2.原理說明 4
2.1 量測原理 4
2.4 控制晶片 5
3.系統設計 6
3.1 硬體說明 6
3.2 溫度設定 8
3.3 觸控設定 9
3.4 顯示設定 10
4.操作流程 11
4.1 程式流程(01) 12
4.2 程式流程(02) 13
4.3 程式流程(03) 14
5.技術規格 15
6.結果總結 15
7.附加檔案 15
8.參考文獻 15
9.修訂紀錄 15
附件:範例應用程式 16

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。

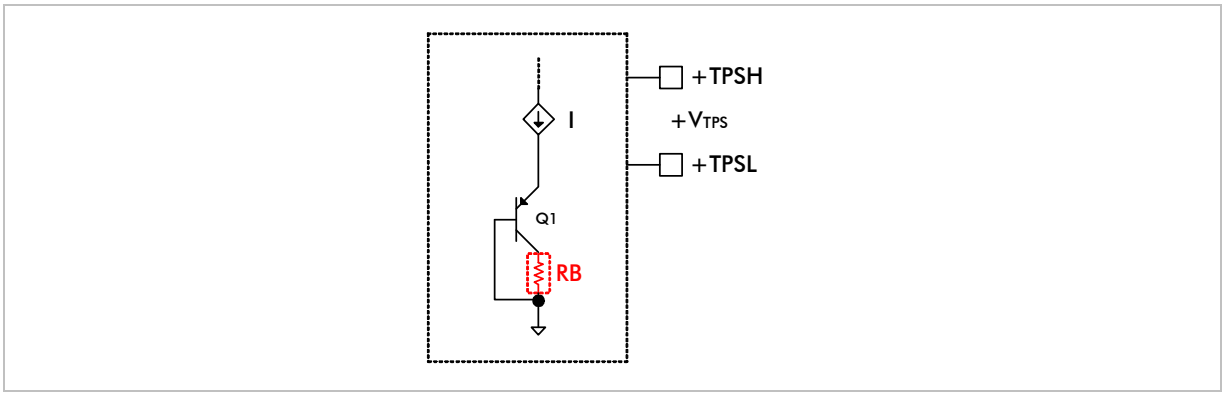
1.內容簡介

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 HY16F188 Series 晶片在溫度量測上的應用，並透過 Touch Key 的介面進行操作。由於 HY16F188 晶片內部集成高精度∑△ADC，且 ADC 輸出頻率最快可以到達 10KHZ，藉由外部 LCD 驅動 IC HY2613B 完成顯示。HY16F188 用於溫度上的量測，不需外接的感測元件即完成，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。



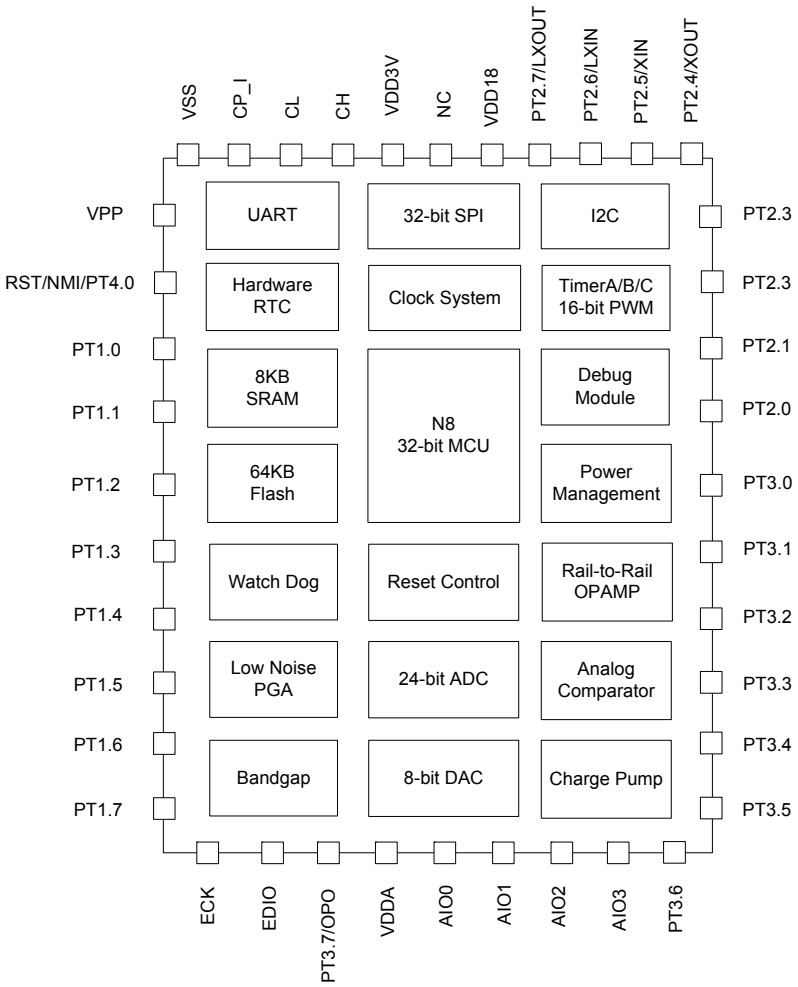
HY16F 啟用時，TPS 的功能隨即被自動啟用。
在同一溫度 TA(℃)下，量測到 VTPS0 與 VTPS1 的數值後，將兩數相加並取平均值即可求得在溫度 TA 下測得 TPS 相對應的電壓值 VTPS@TA。
TPS 的輸出電壓 VTPS 對溫度變化為一線性曲線，故可推倒得出其增益值 GTPS(或稱斜率)

TPS 增益公式

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

2.4 控制晶片

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



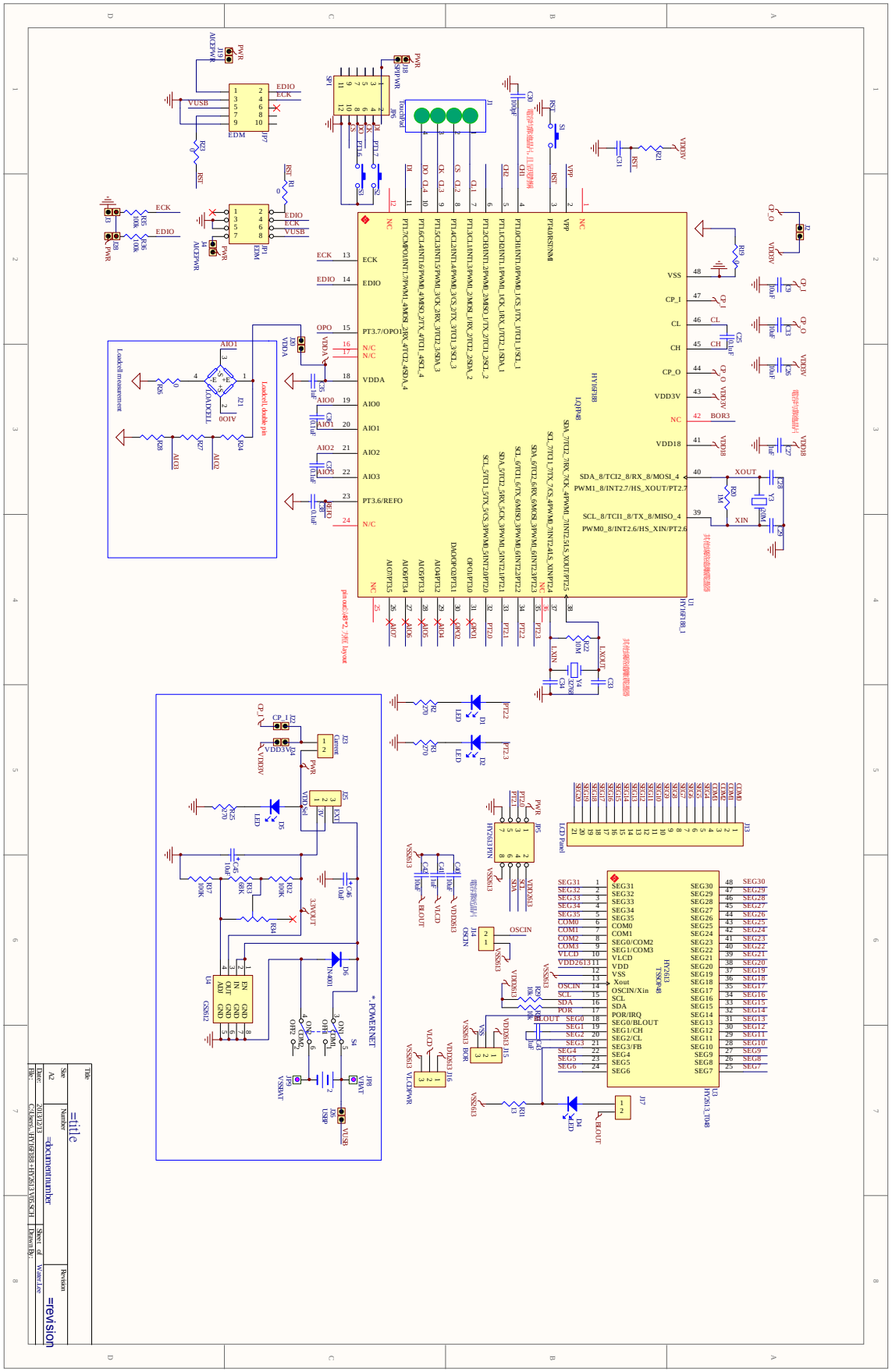
紘康 HY16F 系列 32 位元高性能 Flash 單片機(HY16F188)

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

3.系統設計

3.1 硬體說明

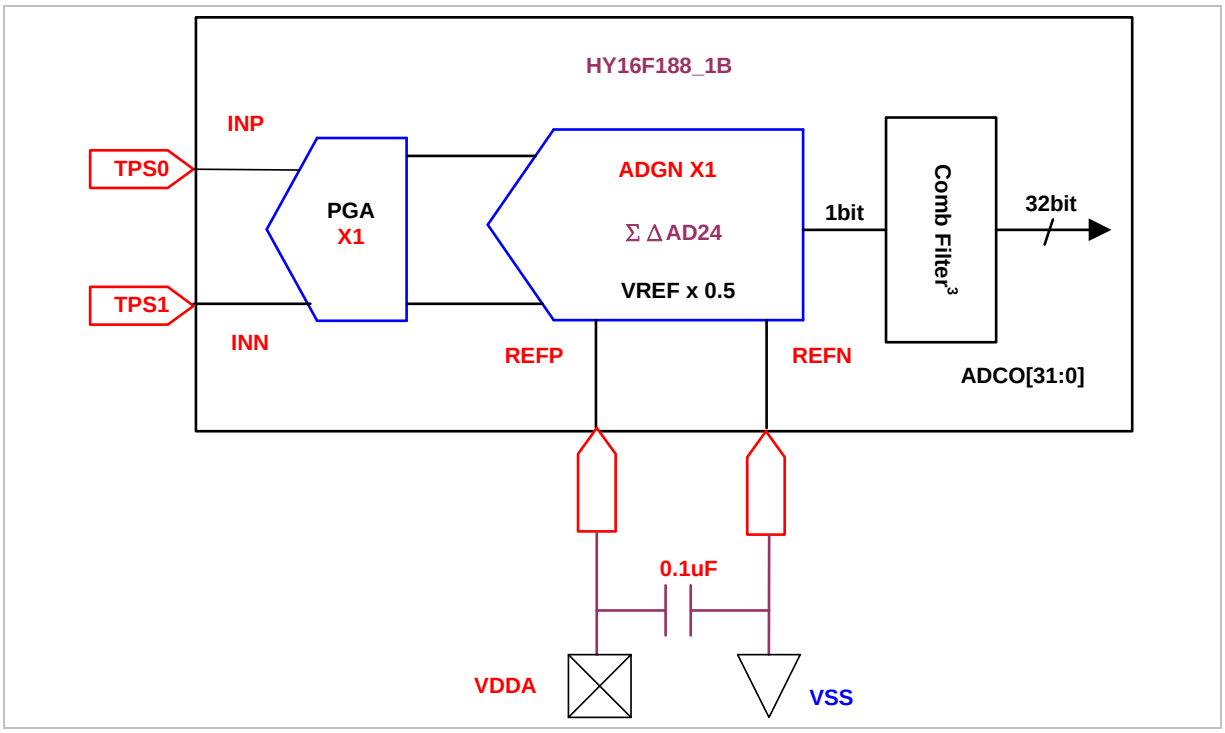
HY16F188對於觸控溫度計的應用，
整體電路就只需HY16F開發板上之Touch Key及LCD顯示晶片及LCD。



觸控溫度量測應用內部電路圖

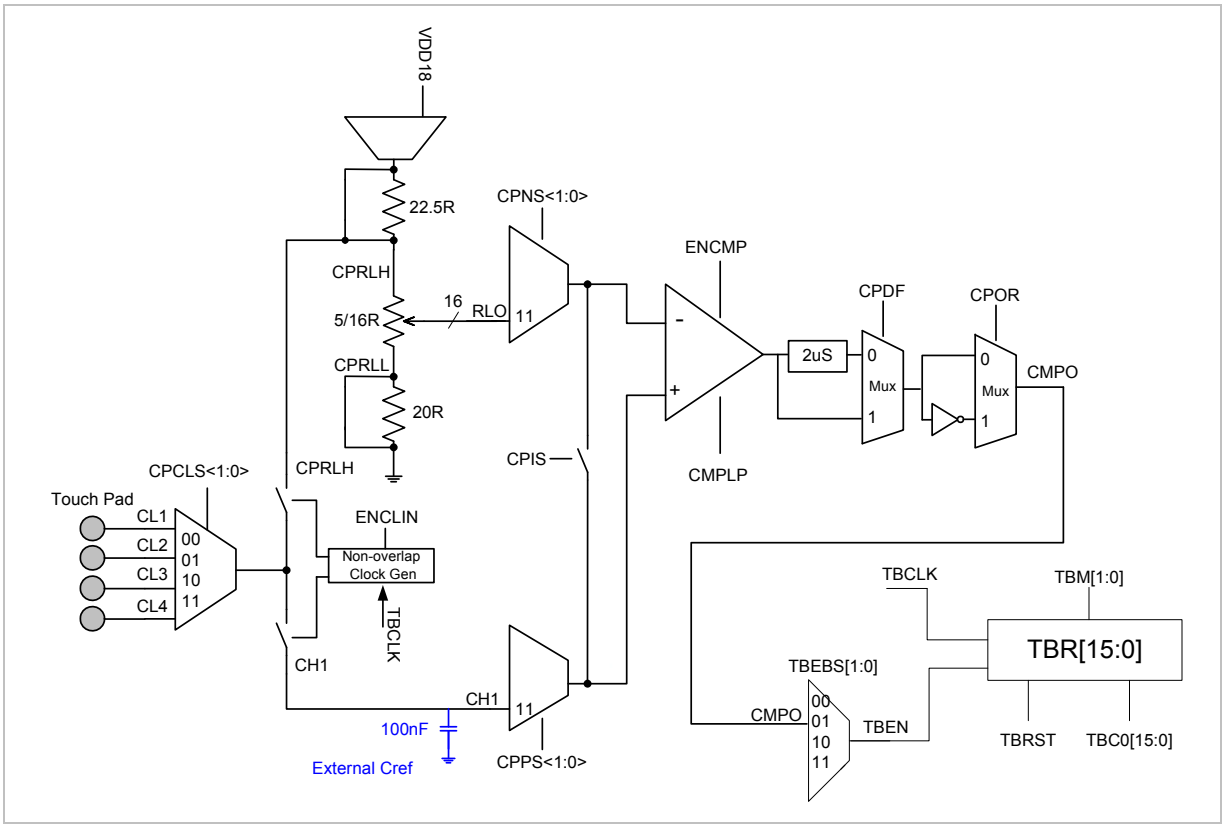
3.2 溫度設定

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



3.3 觸控設定

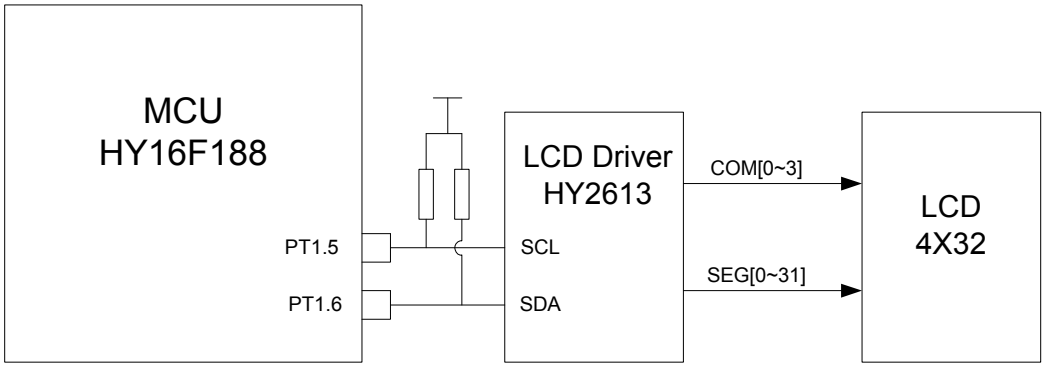
內建硬體觸控模組(使用類比比較器方塊)



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

3.4 顯示設定

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



4.操作流程

一開機後，隨即會顯是當下溫度，觸摸 Touch Key1 會使程式進入 Idle Mode，程式進入 Idle Mode 後，開啓 TimeA 開始計數，每 0.3S 喚醒一次掃 Touch Key1 判斷 Touch Key1 是否被觸碰，如有則離開 Idle Mode。
在顯示溫度情況下如按下 Touch Key3，則會進入溫度校正模式。
在顯示溫度情況下如按下 Touch Key4，則會進入觸控校正模式。

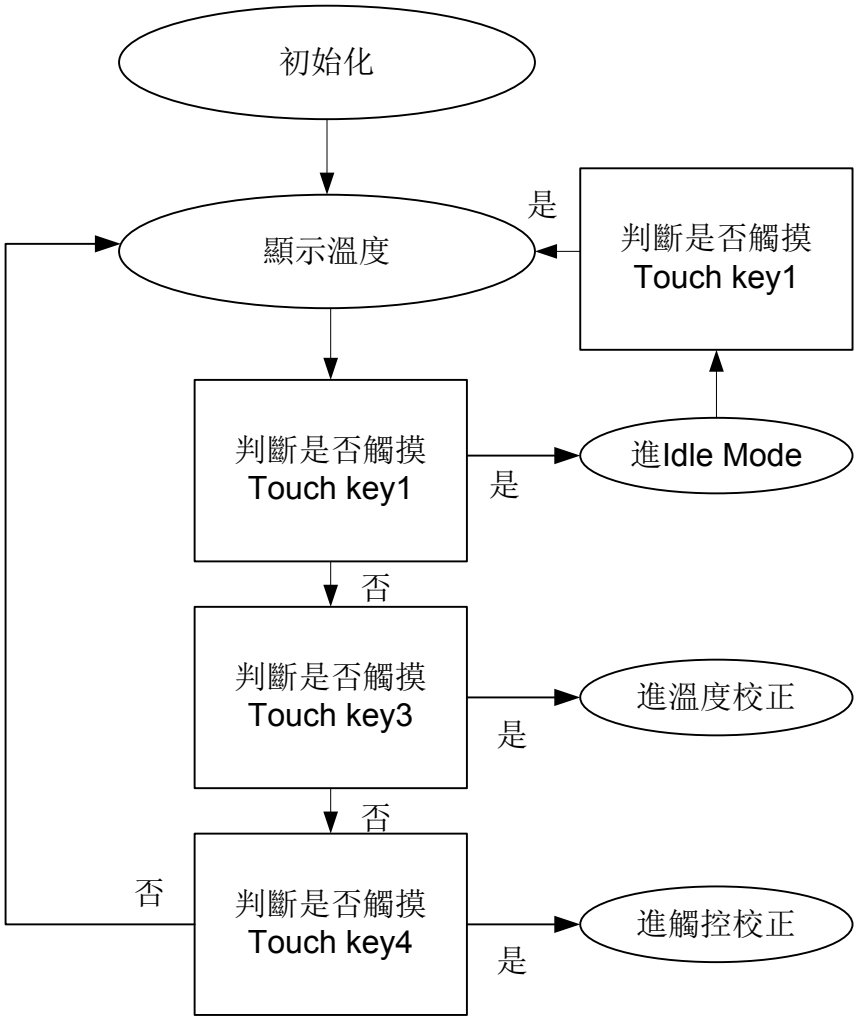
在溫度校正模式下，一開始 16F 會自動抓取現在 AD value 並記錄下來，完成後使用者需透過 Touch Key1、2 設定現在溫度，設定完成後再次觸摸 Touch Key3 及設定完成，離開溫度校正副程式。

在觸控校正模式下，一開始 16F 會自動抓取 untouch value，此時 LCD 會自動倒數。在倒數時切勿碰觸 Touch Key。當自動抓取完成後，LCD 會依序出現 994444、99333、99222、991111，使用者需一出現數字觸摸對應 Touch Key。對應表如下表 1

994444	Touch Key1
993333	Touch Key2
992222	Touch Key3
991111	Touch Key4

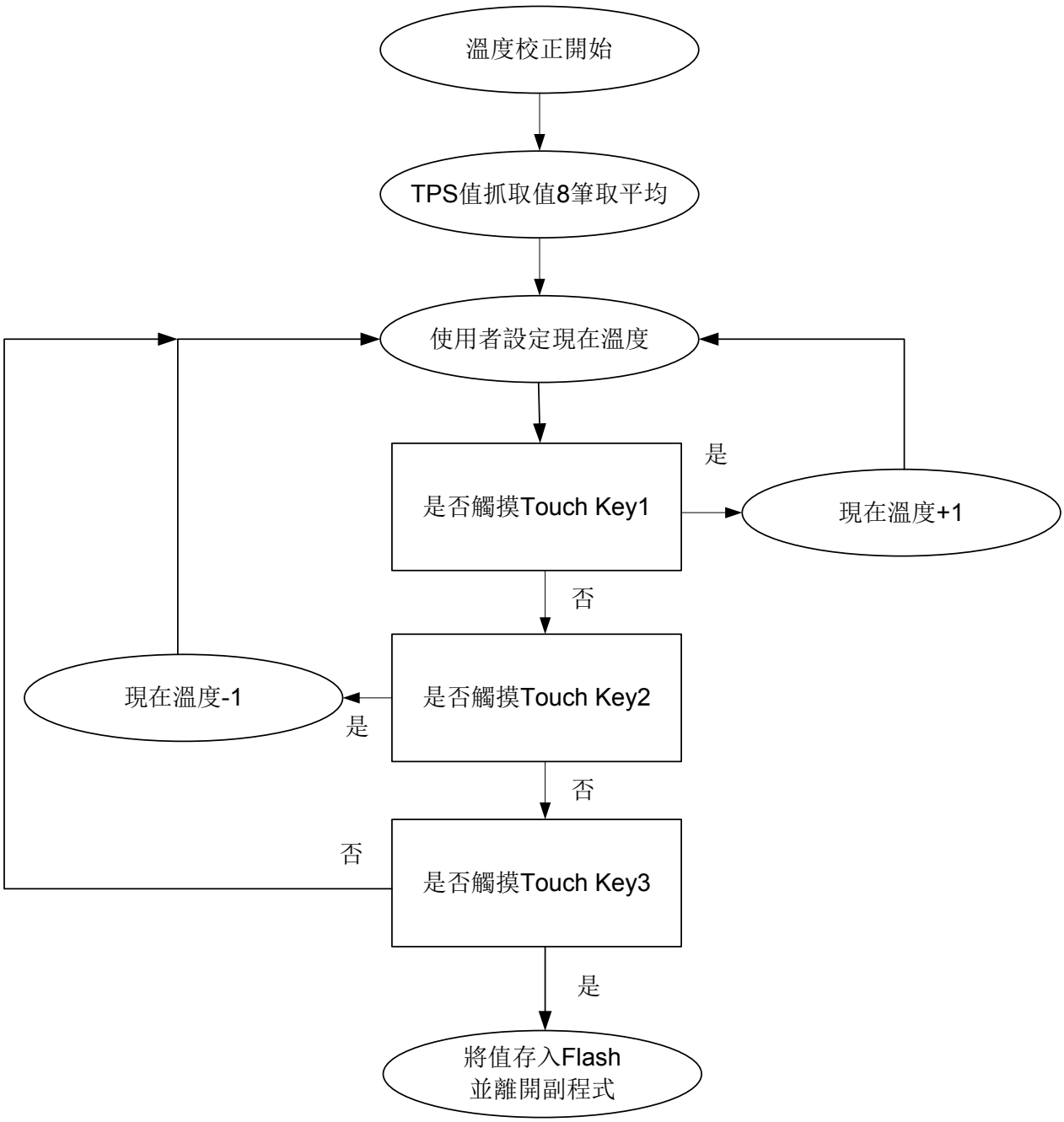
4.1 程式流程(01)

主程式流程



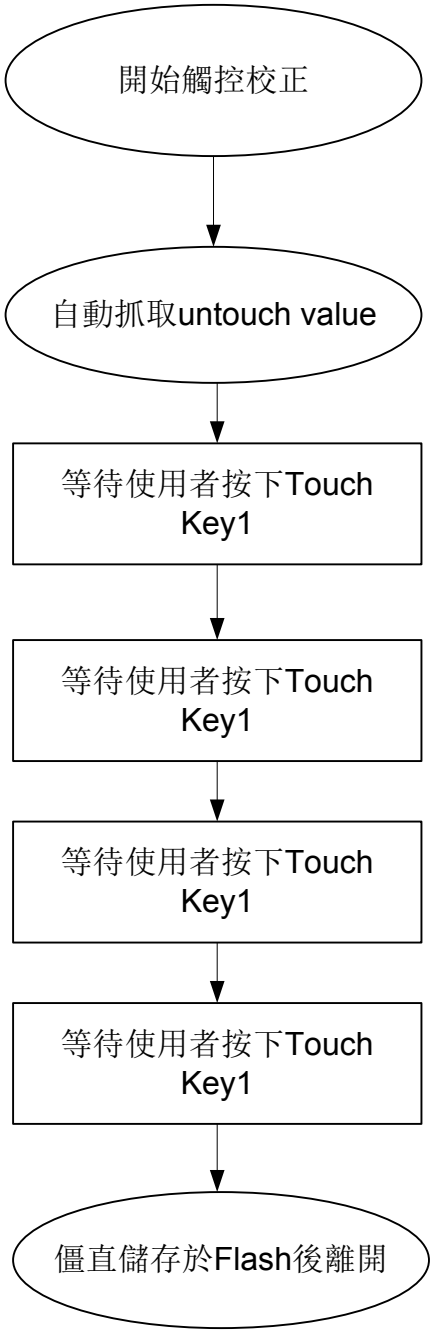
4.2 程式流程(02)

溫度校正程式



4.3 程式流程(03)

觸控校正程式



5.技術規格

(1)電源接點：3.6V
(2)功耗：
功耗：工作模式1mA
待機模式 20uA
(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 HY16F188 為主控結合內部 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。

7.附加檔案


HY16F010V03.zip

8.參考文獻

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

9.修訂紀錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

版本	頁次	變更摘要	日期
V1.0	ALL	初版發行	2013/10/30
V2.0	ALL	更改腳位名稱 VDD->VDD18	2013/12/13
V3.0	ALL	更新程序	2014/12/18

附件:範例應用程式

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

#include "DrvREG32.h"
#include "DrvPMU.h"
#include "DrvCMP.h"
#include "DrvCLOCK.h"
```

```
#include "DrvTimer.h"
#include "DrvADC.h"
#include "DrvFlash.h"

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

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

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS  MCUSTATUSbits;
extern unsigned char seg[16];
#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;                    //ADC 用
float gain,temp,tempp,temploop;        //溫度計算用
```

```
unsigned int unth0,unth1,unth2,unth3,th0,th1,th2,th3; //touch 校正用
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void ClearLCDframe(void);
void LCD_DATA_DISPLAY(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY1();
void Delay(unsigned int num);
void Ini_Touch(void);
void Initial_Array(void);
void Initial_ALL(void);
void InitalADC(void);
void  ScanKey4(unsigned int TCH);
void AD(void);
void touch_ct(void);
void temp_ct(void);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
  //unsigned int m;

  DrvCLOCK_EnableHighOSC(E_INTERNAL,50);        // Select HAO 4MHz
  InitalI2C();
  SYS_EnableGIE(7,0x3F);  // Enable GIE(Global Interrupt)

  Ini_Display();
  ClearLCDframe();
  Delay(10000);
  //DisplayHYcon();
  LCD_DATA_DISPLAY(123456);

  Delay(50000);
  MCUSTATUSbits._byte = 0;
```

```
Ini_Touch();
InitalADC();
Initial_ALL();

a=ReadWord(0x8000);          //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(0x8004);
        tempp=ReadWord(0x8008);
}
else          //如果判斷為沒校正過，抓原先內部設定值
{
        temp=6047;
        tempp=25;
}

a=ReadWord(0x800c);       //判斷 touch 是否被校正過，如果被校正過抓 Flash 內值
if(b==0x5566)
{

        threh0=ReadWord(0x8010);
        threh1=ReadWord(0x8014);
        threh2=ReadWord(0x8018);
        threh3=ReadWord(0x801c);
}
else                  //如果判斷為沒校正過，抓原先內部設定值
{
        touch_ct();
        //threh0=300;//CL0
        //threh1=220;//CL1
        //threh2=155;//CL2
        //threh3=150;//CL3
}

while(1)
{

pmu_00=0xff00ff00;                    //將比較器功能關閉(省電)
```

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



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

        clk_00=0xff01;                //開啟高速
        clk_08=0xff4c;
        TCH=0;ScanKey4(TCH); //掃描 touch1 是否被按下,touch1 被按下 LED+1
        if(LED==1)
        {
            while(1)
            {
                DrvADC_Enable();          //開啟 ADC 功能
                AD();                     //ADC 副程式
                gain=(296+tempp)/temp;            //溫度換算
                temploop=gain*sum-296;
                temploop*=10;
                LCD_DATA_DISPLAY(temploop);  //顯示現在溫度

                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();
                                        Delay(0xFF00);
                                        goto bb;
                                }
                                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(0x2000);
                                TCH=0;ScanKey4(TCH);
                        }
                }
        }
        t=0;
bb:     ClearLCDframe();                    //清除 LCD 顯示
  }


  return 0;
}

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

  if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT)  // Get I2C Interrupt Flag
```

```
{
  I2C_Status=DrvI2C_GetStatusFlag();  // Get I2C Status Flag
  switch(I2C_Status)
  {
    case 0x90: //MACTFlag+RWFlag
        { /* START has been transmitted */
          DrvI2C_WriteData(I2C_TARGET);  //Send Slave Address & R/W Bit
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          break;
        };
    case 0x84: //MACTFlag+ACKFlag
        { /* Slave A + W has been transmitted. ACK has been received. */
          DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          break;
        };
    case 0x80: //MACTFlag
        { /* Slave A + W has been transmitted. ACK has been received. */
          DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          break;
        };
    case 0x30:
        {
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          EndFlag=1;
          break;
        };
    case 0x8C: // MACTFlag+DFFlag+ACKFlag
        { /* DATA has been transmitted and ACK has been received */
          if(DataTxIndex<DataTxLen)
          {
            DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          }
          else
          {
            if(I2C_RW == WRITE)
```

```
            {
              DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
              EndFlag=1;
            }
            else if(I2C_RW == READ)
              DrvI2C_Ctrl(1,0,0,0);  // I2C as master sends START signal
            DataTxIndex=0;
          }
          break;
        };
    case 0x88: //MACTFlag+DFFlag
        { /* DATA has been transmitted and NACK has been received */
          DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
          DataTxIndex=0;
          EndFlag=1;
          break;
        };
    case 0xB0:
        { /* A repeated START has been transmitted. */
          DrvI2C_WriteData(I2C_TARGET | READ);  //Send Slave Address & R/W Bit
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          break;
        }
    case 0x94: //MACTFlag+RWFlag
        { /* Slave A + R has been transmitted. ACK has been received. */
          DrvI2C_Ctrl(0,0,0,1);  // Set ACK bit
          break;
        };
    case 0x9C: //MACTFlag+RWFlag+DFFlag+ACKFlag
        { /* Data byte has been received. ACK has been transmitted. */
          if(DataRxLen>DataRxIndex)
          {
            Recbuf[DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          }
          else
          {
            Recbuf[DataRxIndex++]=DrvI2C_ReadData();
```

```
            DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
            EndFlag=1;
          }
          break;
        };
    case 0x98: //MACTFlag+RWFlag+DFFlag
        { /* Data byte has been received. NACK has been transmitted. */
          DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
          EndFlag=1;
          break;
        };
    default:
        {
          DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
          EndFlag=1;
          break;
        };
  }
  DrvI2C_ClearEIRQ();
  DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CIF)
  SYS_EnableGIE(7,0x3F);       // Enable GIE(Global Interrupt)
 }
 if(DrvI2C_ReadIntFlag()==E_DRVI2C_ERROR_INT)  //Get I2C Error Interrupt Flag
 {
  EndFlag=1;
  DrvI2C_ClearIRQ();
  DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CEIF)
  DrvI2C_Ctrl(0,1,0,0);   // I2C as master sends STOP signal
  SYS_EnableGIE(7,0x3F);       // Enable GIE(Global Interrupt)
 }
}
/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
 for(;num>0;num--)
  asm("NOP");
```

```
}

/*---------------------------------------------------------------------------*/
void ScanKey4(unsigned int TCH)
        //touch key 副程式
{
  CL=TCH;pio_00=0x0100;
  DrvTMB_ClearTMB();
  DrvCMP_EnableNonOverlap(CL);
  cmpresult=DrvCMP_ReadData();
  while(cmpresult==1){cmpresult=DrvCMP_ReadData();}
  tmrbcnt=DrvTMB_CounterRead();//TMB        //判斷充電時間是否低於閥值
  switch(CL)
  {
    case 0 : if(tmrbcnt<=threh0){LED++;}break;
    case 1 : if(tmrbcnt<=threh1){LED--;}break;
    case 2 : if(tmrbcnt<=threh2){c++;}break;
    case 3 : if(tmrbcnt<=threh3){c--;}break;
    default: LED=0;
  }
  pio_00=0x0101;
  pio_04=0x0100;

}

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

/*---------------------------------------------------------------------------*/
void HW1_ISR(void)                          //TMA 中斷副程式
{
  DrvTIMER_ClearIntFlag(E_TMA);              //Clear TMA interrupt flag
  t=1;
  asm volatile("sethi $r0, 0xc0000");       //預防中斷不可預期的關閉
  asm volatile("ori  $r0, $r0, 0x003f");
  asm volatile("mtsr $r0, $INT_MASK");
  asm volatile("movi $r0, 0x70009");
```

```
  asm volatile("mtsr $r0, $PSW");
}
/*-------------------------------------------------------------------------------*/
void HW2_ISR(void)                          //AD 中斷副程式
{
  DrvADC_ClearIntFlag();                    //清除 AD 中斷旗標
  ADCData=adc_08;
  asm volatile("sethi $r0, 0xc0000");         //預防中斷不可預期的關閉
  asm volatile("ori  $r0, $r0, 0x003f");
  asm volatile("mtsr $r0, $INT_MASK");
  asm volatile("movi $r0, 0x70009");
  asm volatile("mtsr $r0, $PSW");
}
/*-------------------------------------------------------------------------------*/
void temp_ct(void)                            //溫度校正副程式
{
        int a,d;
        d=0;
        c=0;
        for(a=0;a<8;a++)                      //將 TPS 所測得數值取 8 筆後平均
        {
                AD();
                d+=sum;
                LCD_DATA_DISPLAY(sum);
                Delay(0xff00);
        }
        d=d>>3;
        temp=d;                                 //將 AD 值放入 溫度計算用暫存器
        LED=tempp;                                    //設定現在溫度
        while(1)
        {
                LCD_DATA_DISPLAY(LED);
                Delay(0xf000);
                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(0x8000,0x600,0x1110);//將判斷是否校正過用數值燒入 Flash
                  DrvFlash_Burn_Word(0x8008,0x600,tempp);//將校正值燒入 Flash
                    DrvFlash_Burn_Word(0x8004,0x600,temp);
                    Delay(0xff00);
                    DrvTIMER_EnableInt(E_TMA);          //開啟 TMA 中斷
                    DrvADC_EnableInt();          //開起 ADC 中斷
                    LED=0;
                    break;
                }
        }
}
/*-------------------------------------------------------------------------------*/
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
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(0x800c,0x600,0x5566);//將判斷是否校正過用值，存入 Flach
                DrvFlash_Burn_Word(0x8010,0x600,threh0);//將校正值存入 Flash
                DrvFlash_Burn_Word(0x8014,0x600,threh1);
                DrvFlash_Burn_Word(0x8018,0x600,threh2);
                DrvFlash_Burn_Word(0x801c,0x600,threh3);
                Delay(0x8000);
                DrvTIMER_EnableInt(E_TMA);   //開啟 TMA 中斷
                DrvADC_EnableInt();          //開啟 ADC 中斷
}
/*-------------------------------------------------------------------------------*/
void AD(void)
        //ADC 副程式
{
        int h;
        sum=0;
        for(h=0;h<8;h++)                    //8 筆取平均
        {
                ADtemp10=(ADCData>>14);
                sum=sum+ADtemp10;
        }
        sum=sum>>3;
}


void InitalADC(void)
{
  //Set ADC input pin
  //DrvADC_SetADCInputChannel(ADC_Input_AIO3,ADC_Input_AIO2);//ADC Input
        //DrvADC_SetADCInputChannel(ADC_Input_TPSP0);//ADC Input
  //DrvADC_PInputChannel(TPS1);
  //DrvADC_NInputChannel(TPS0);
  adc_04=0x7676;
  DrvADC_InputSwitch(OPEN);
  DrvADC_RefInputShort(OPEN);
```

```
  DrvADC_Gain(0,0);           //Set the ADC Gain
  DrvADC_DCoffset(0);         //DC offset input voltage selection
  DrvADC_RefVoltage(VDDA,0); //Set the ADC reference voltage.
  DrvADC_FullRefRange(1);     //Set the ADC reference range select.
                              //0:Full reference range input
                              //1:1/2 reference range input
  DrvADC_OSR(0);              //0:OSR=32768
  DrvADC_CombFilter(Enable);
  DrvADC_ClkEnable(0,1);      //Setting ADC CLOCK ADCK=HS_CK/6
  //Set VDDA voltage
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_BandgapEnable();
  DrvPMU_REFO_Enable();
  DrvPMU_AnalogGround(Enable);//ADC analog ground source selection.
                              //1:Enable buffer and use internal source
  DrvPMU_LDO_LowPower(0);     //VDD LDO with low power control.
                              //0 : Normal;1 : Low power
  Delay(0x1000);
  //Set ADC interrupt
  DrvADC_EnableInt();
  DrvADC_ClearIntFlag();
  DrvADC_Disable();
}
void Initial_ALL(void)
{
          pio_00=0xff00ff00;                    //確保沒用的腳位關閉
          pio_04=0xff00ff00;
          pio_08=0x00;
          pio_10=0xff00ff03;
          pio_14=0xff00ff03;

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

          DrvTMA_Open(9,1);                     //TimerA Overflow
```

```
                                                //9:taclk/1024/32;TMRDV=÷32
                                                //00:HS_CK
          DrvTIMER_ClearIntFlag(E_TMA);         //Clear Timer A interrupt flag
          DrvTIMER_EnableInt(E_TMA);            //Timer A interrupt enable

          SYS_EnableGIE(7,0X3F);                 //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);
  DrvCMP_DisableNonOverlap();
  DrvCMP_RLO_Ctrl(0,1);
  DrvCMP_InputSwitch(1);
  DrvCMP_InputSwitch(0);
  DrvCMP_RLO_Ctrl(1,0);
  DrvTMB_ClearTMB();
  DrvCLOCK_SelectIHOSC(0x00);
  DrvCLOCK_SelectMCUClock(0,0);          //select MCU clock as LS_CK,div1
  DrvTMBC_Clk_Source(1,0);//(1,0)             //TimerB Clock enable pre_scale 1
//TimerB overflow 0xffff->PWM period 0xffff
  DrvTMB_Open(E_TMB_MODE0,E_TMB_CMP_HIGH,0xffff);
  DrvTMB_ClearTMB();
  //Set VDDA voltage
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_BandgapEnable();
  DrvPMU_REF0_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);
}
/*---------------------------------------------------------------------------*/
```