電壓電流計_應用說明書

HY16F188

Voltage Current Meter

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

工業上的應用對於電壓及電流的量測，是最基本卻也是最要的。工業上的壓力、溫度、濕度等許多量測都是透過感測器後將物理訊號變成電壓或者電流，再透過電子儀器的解析後顯示於儀表上，因此如何量測到精準的電壓、電流是相當重要的。本文主要是介紹 HYCON HY16F188 Series 晶片在電壓電流量測的應用。由於 HY16F188 晶片內部集成高精度顯示。∑△ADC，且 ADC 輸出頻率最快可以到達 10KHZ，並透過外部 LCD 驅動 IC HY2613B 完成 HY16F188 用於電壓電流的量測時，擁有相當高的精準度。

2.原理說明

電壓量測：
電路圖如圖1所示，此電路為簡易分壓電路，分壓比為20:1，並由於程式設定關係，AIO0、AIO1兩端電壓差最大為1.2V。因此量測電壓上限為20V。電流量測：電路圖如圖2所示，分法為當電流源流過10Ω電阻時，產生壓降。透過量測壓降的方式反推流經電流大小。

解析度分為外部解析度和內部解析度，外部解析度為最大量測的輸出電壓值與需要識別的最小電壓值的電壓值之比，本應用最小量測電壓值為 10mV。
一般我們以目視法認定的內部解析度通常是指我們經軟體處理後 LCD 顯示只有 1 格滾動時，此時滿量程的格數就是內部解析度，其 1 格所代表的訊號約為 2-3 倍 RMS Noise。
內外解析度之比越小，電壓電流表精度越高，但內外解析度之比是有限制的。比如滿量程壓差為 1.1V，要做到 2000 Count，內外比為 1：10 的電壓電流表，如果不經過信號放大，那最小要處理的信號為 1.1V/(2000×10)=55uV。而 SD24 所能處理的最小信號值大約為 65nV，所以要完成此規格的量測示相當容易且精準的。

ADC 性能能否達到規格要求，通常是以 RMS Noise 來推算外部是否穩定內部解析度比值。對於開發電子產品而言，使用 HY16F188 晶片其所能達到的最大內部解析度的瓶頸在於 Input RMS Noise 而不在於 ADC 的解析度。HY16F188 的 ADC 待測信號在由 PGA、AD 倍率調整器的放大後(PGA=32，ADGN=4)，經 OSR=32768 每秒輸出 10 筆 ADC 值的條件下，其 Input RMS Noise 約為 65nV，但由於其 Input Noise 主要由 Thermal Noise 組成，所以如果我們透過平均的軟體處理是可以再將 Input Noise 進一步降低。
如果我們使用 8 筆的軟體平均處理其 Input RMS Noise 考慮其他雜訊因素後，可達約為 40nV，3 倍 RMS Niose 代表約 1 格的滾動，即為 120nV。在使用 2.4V 驅動電壓，1mV/V 的滿量程時壓差可達 2.4mV，所以在此情形下我們可以得到 20000 Counts 的內部解析度。

2.1 量測原理

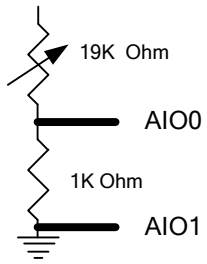


圖1.電壓量測電路圖

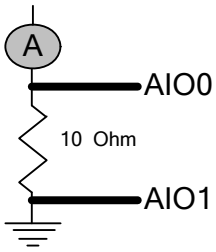
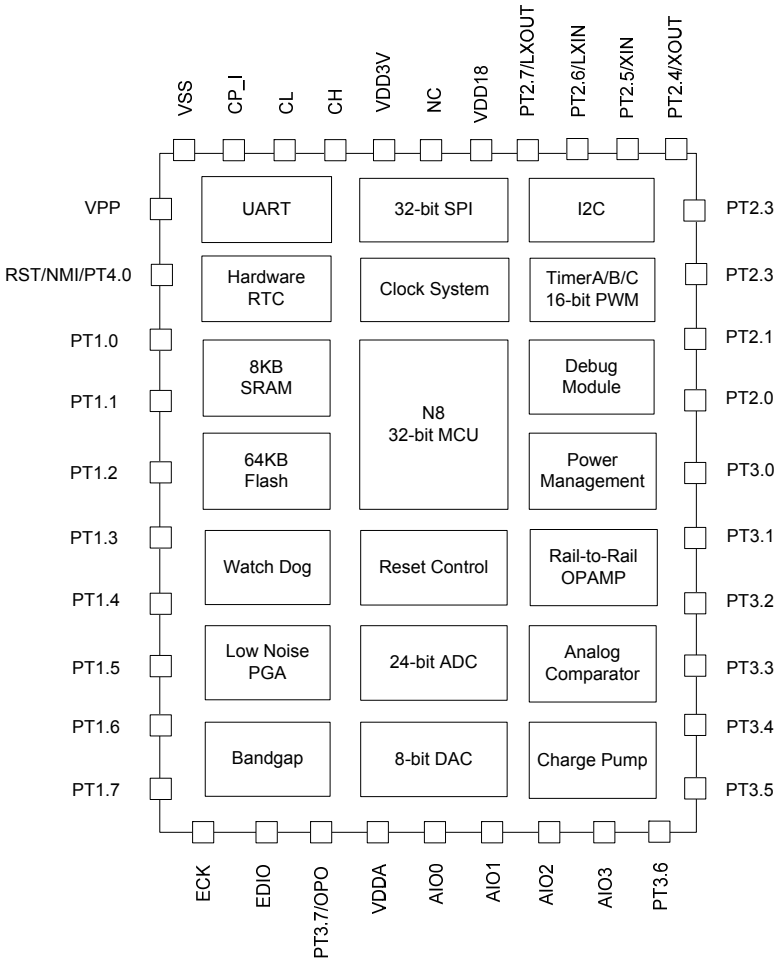


圖2.電流量測電路

2.2 控制晶片

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



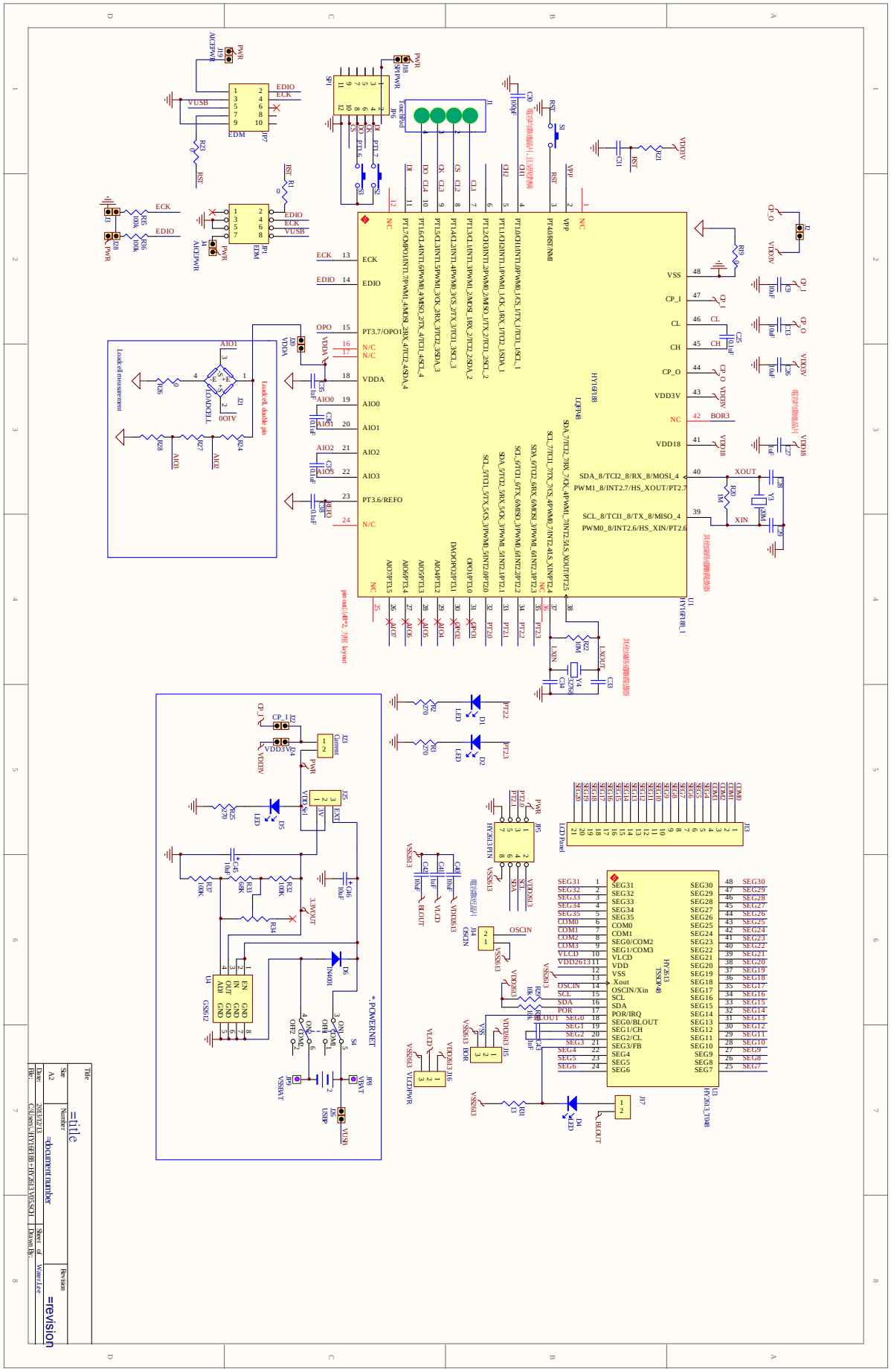
紘康 HY16F 系列 32 位元高性能 Flash 單片機(HY16F188)

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

3.系統設計

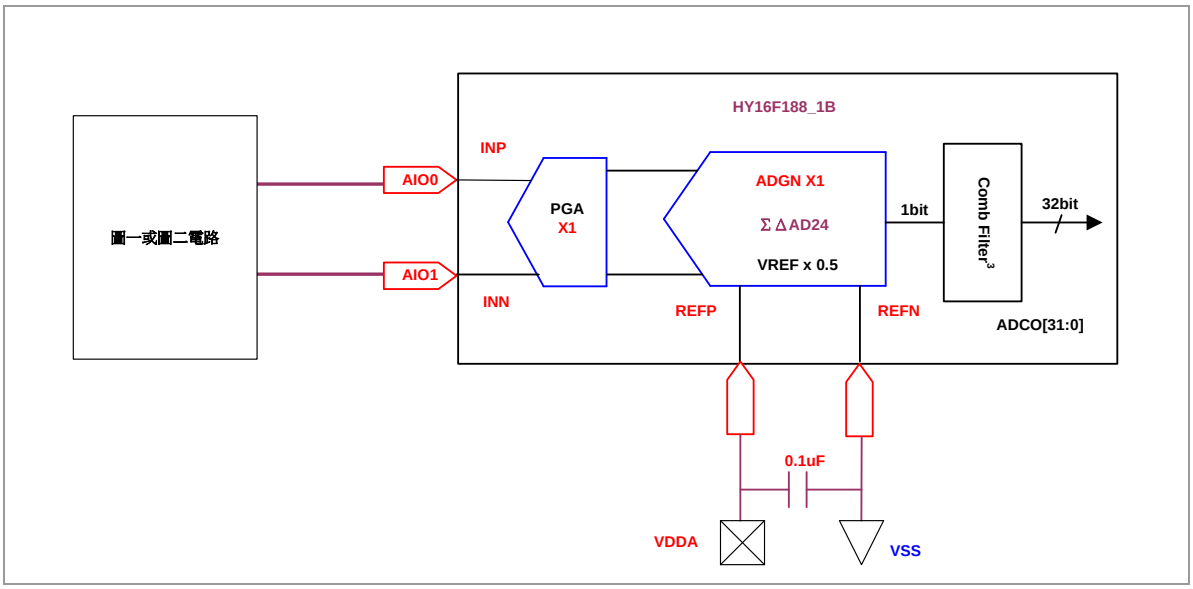
3.1 硬體說明

HY16F188 對於電壓電流量測應用，
整體電路包含 S2、S3 按鈕部分及 LCD 顯示模組如圖 3 所示。
(A)中央處理器：
HY16F188 (Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + UMC 64K Flash)
(B)顯示晶片：HY2613 (HYCON LCD Driver LCD Segment 4X36)
(C)電源電路：5.0V 轉 3.3V 電源系統
(D)類比感測模組：內部 ADC
(E)線上燒錄與 ICE 連結電路，透過 EDM 的連接，可支援線上燒錄模擬。
並擁有強大的 C 平台 IDE 以及 HYCON 類比軟體分析工具與 GUI 等支援。



3.2 功能說明

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。



3.2.1 電壓量測

電壓量測模式下，量測範圍為±20V，量測時須配合圖 1 電路圖。顯示至 1mV，精準度至 10mV

3.2.2 電流量測

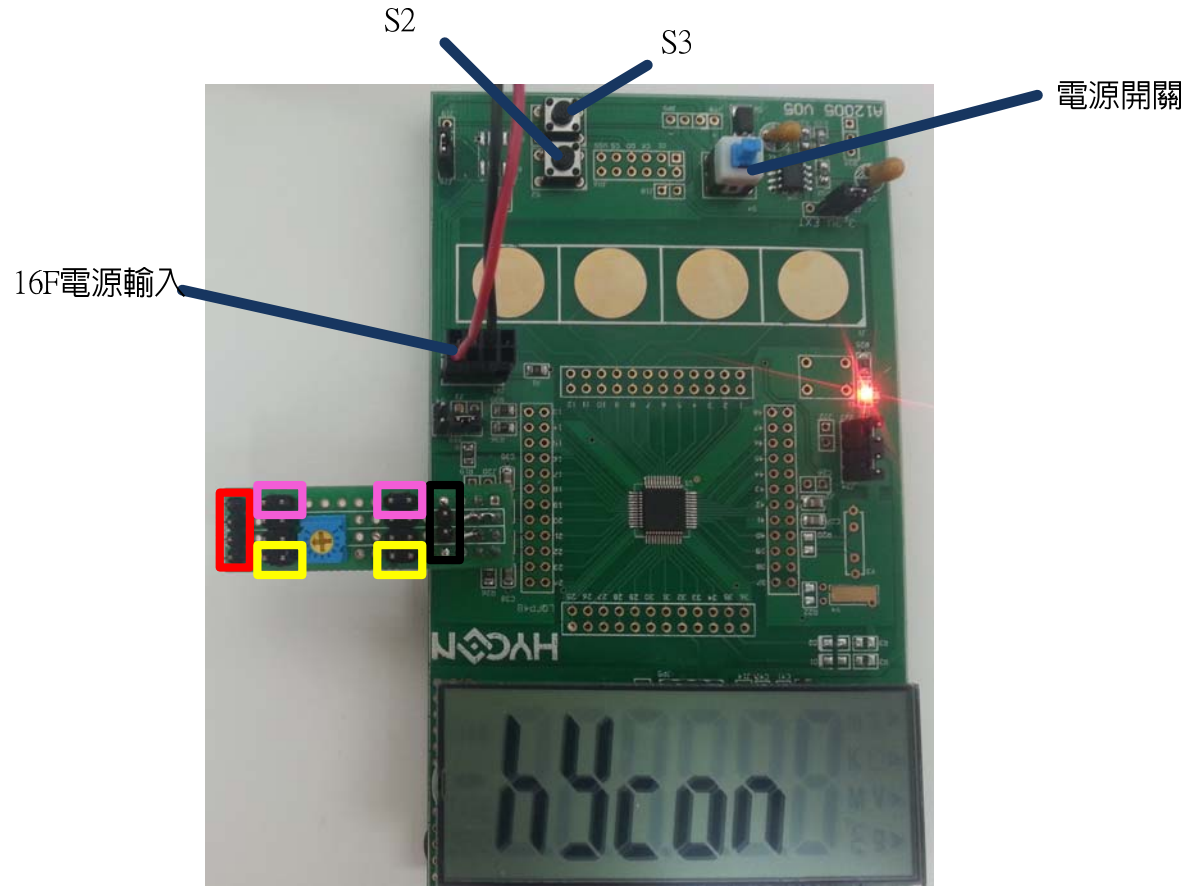
電流量測主要範圍為±10A，量測時須配合圖 2 電路。顯示及量測精準度皆為 0.1mA

4.操作流程

4.1 操作方法

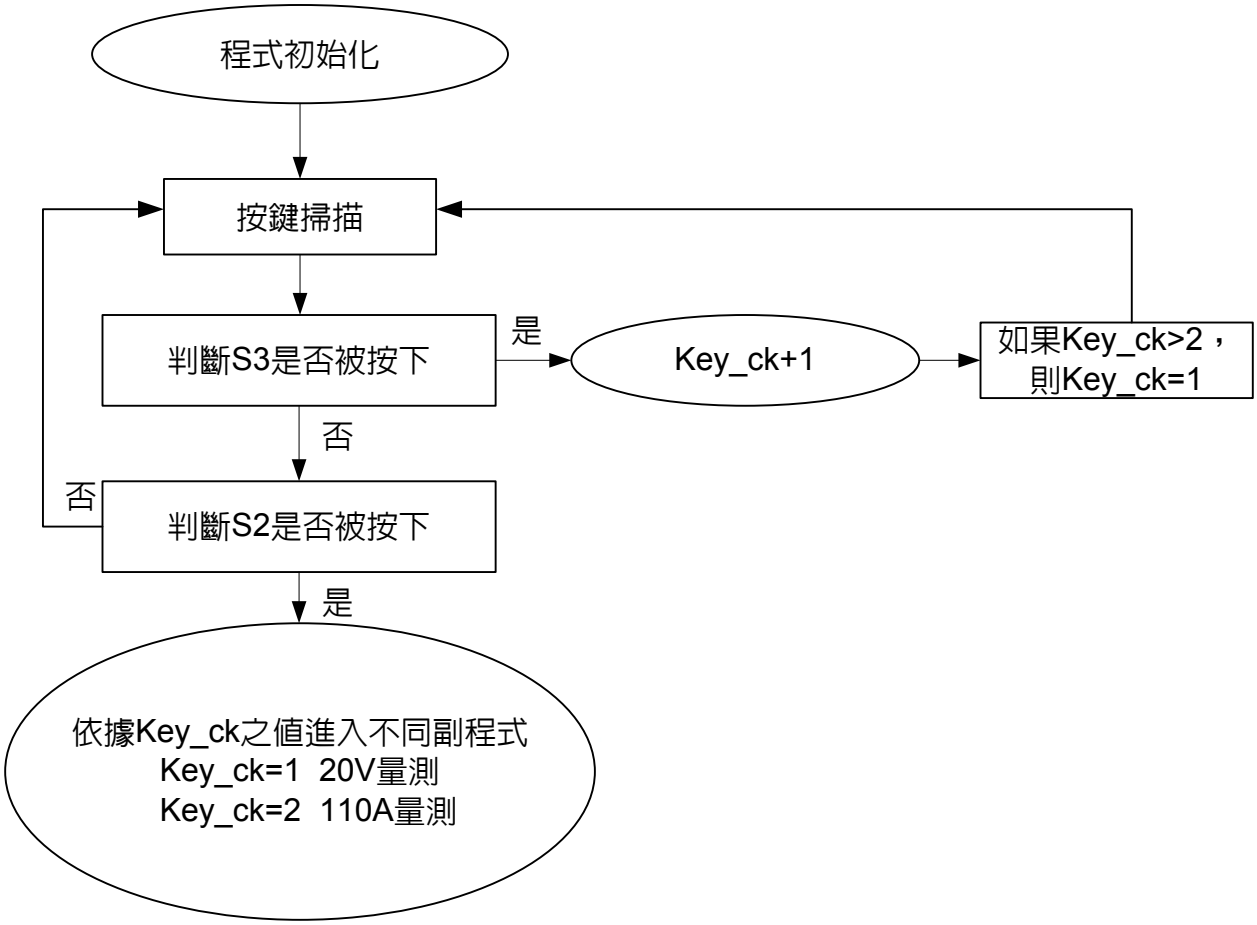
啓動後，首先會進行初始畫及 HYCON 字樣顯示。

透過 S3 按鈕進行量測選擇的切換；S2 按鈕代表確定。
選單顯示：
20V 即代表±20V 量測(須配合外部進行適當的短路)
10A 即代表±10A 量測(須配合外部進行適當的短路)
在量測模式下，在次按下 S2 離開回到初始狀態

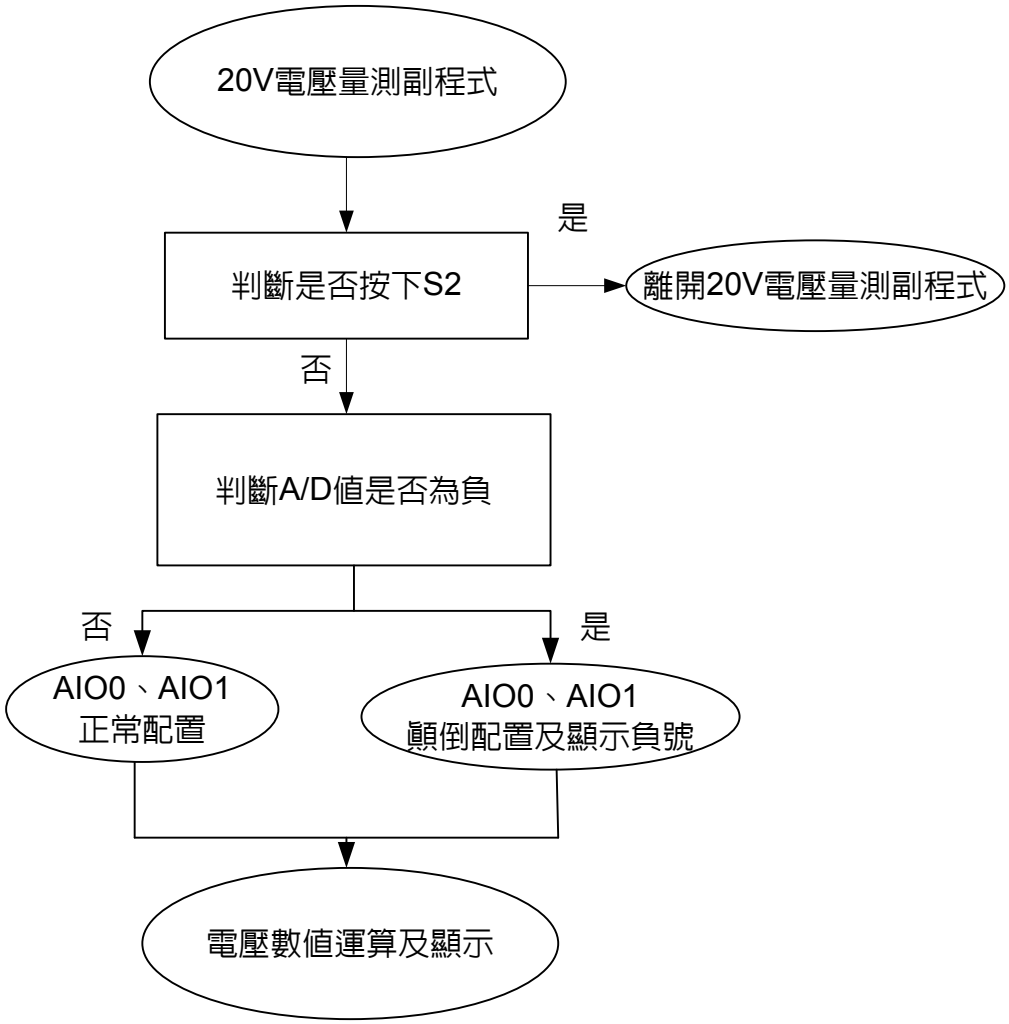


紅色方框部分為 量測+
黑色方框部分為 量測-
選擇量測電流時，須將兩個粉紅色方框處短路
選擇量測電壓時，須將兩個黃色方框處短路

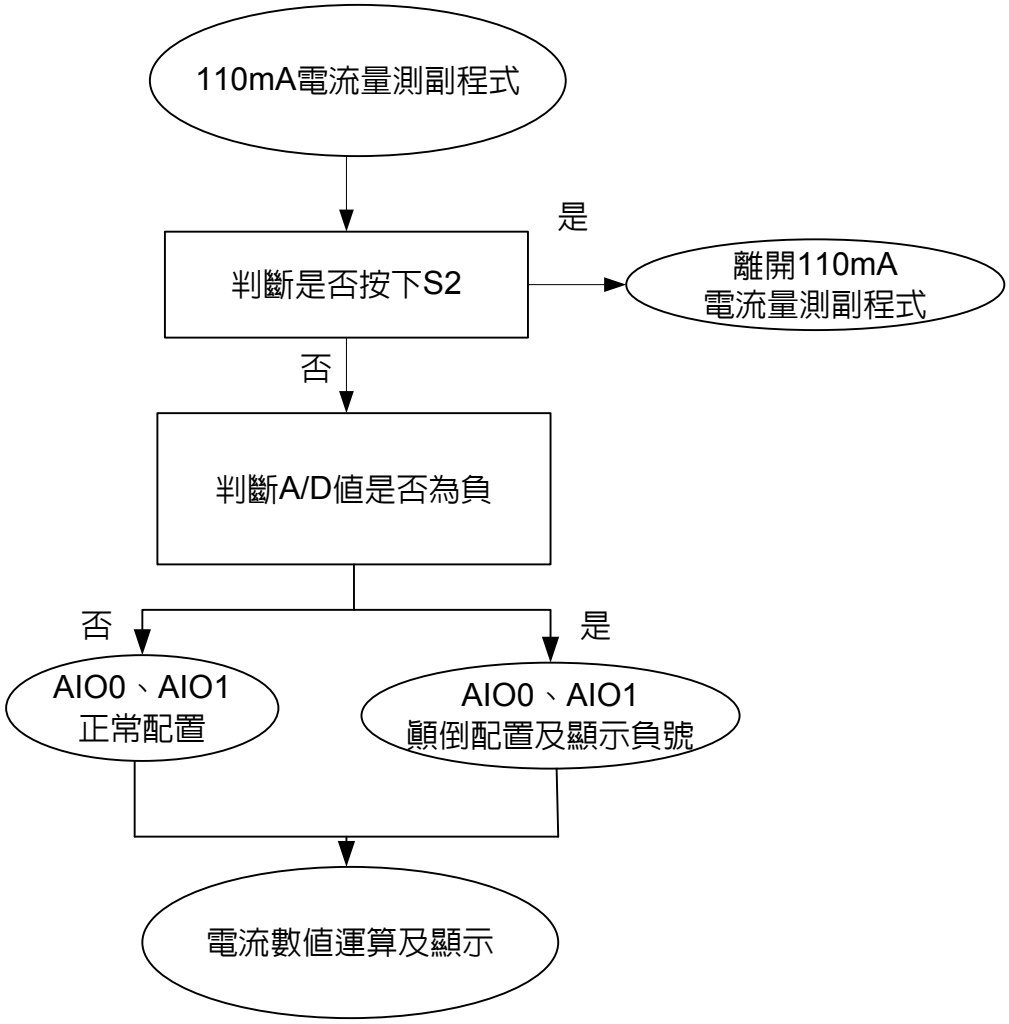
4.2 程式流程


程式初始化
按鍵掃描
判斷S3是否被按下
是
Key_ck+1
如果Key_ck>2，
則Key_ck=1
否
否
判斷S2是否被按下
是
依據Key_ck之值進入不同副程式
Key_ck=1 20V量測
Key_ck=2 110A量測

4.3 電壓量測副程式



4.4 電流量測副程式


110mA電流量測副程式
判斷是否按下S2
是
離開110mA
電流量測副程式
否
判斷A/D值是否為負
否
是
AIO0、AIO1
正常配置
AIO0、AIO1
顛倒配置及顯示負號
電流數值運算及顯示

5.技術規格

(1)電源接點：3.0V
(2)功耗：工作模式 1mA
(3)量測精準度：電壓:10(mV)　以及　電流 0.1(mA)
(4)適用範圍：各種電壓電流量測
(5)工作溫度：-40℃ ～ +85℃；
(6)存貯溫度：-55℃ ～ +125℃；
(7)相對濕度：＜95%（20±5℃條件）

5.1 測試數據

20V 電壓測試

Fluke輸出電壓	16F188測得電壓		誤差%	
mV	+	-	+	-
10	10	10	0.0000	0.0000
20	20	20	0.0000	0.0000
30	30	30	0.0000	0.0000
40	40	40	0.0000	0.0000
50	50	50	0.0000	0.0000
60	60	60	0.0000	0.0000
70	70	70	0.0000	0.0000
80	80	80	0.0000	0.0000
90	90	90	0.0000	0.0000
100	100	100	0.0000	0.0000
200	200	200	0.0000	0.0000
300	300	300	0.0000	0.0000
400	400	400	0.0000	0.0000
500	500	500	0.0000	0.0000
600	600	600	0.0000	0.0000
700	700	700	0.0000	0.0000
800	800	800	0.0000	0.0000
900	900	900	0.0000	0.0000
1000	1000	1000	0.0000	0.0000
1500	1500	1500	0.0000	0.0000
1600	1600	1600	0.0000	0.0000
1700	1700	1700	0.0000	0.0000

1800	1800	1800	0.0000	0.0000
1900	1900	1900	0.0000	0.0000
2000	2000	2000	0.0000	0.0000
2500	2500	2500	0.0000	0.0000
3000	3000	3000	0.0000	0.0000
3500	3500	3500	0.0000	0.0000
4000	4000	4000	0.0000	0.0000
4500	4500	4500	0.0000	0.0000
5000	5000	5000	0.0000	0.0000
6000	6000	6000	0.0000	0.0000
7000	7000	7000	0.0000	0.0000
8000	8000	8000	0.0000	0.0000
9000	9000	9000	0.0000	0.0000
10000	10000	10000	0.0000	0.0000
11000	11000	11000	0.0000	0.0000
12000	12000	12000	0.0000	0.0000
13000	13000	13000	0.0000	0.0000
14000	14000	14000	0.0000	0.0000
15000	15000	15000	0.0000	0.0000
16000	16000	16000	0.0000	0.0000
17000	17000	17000	0.0000	0.0000
18000	18000	18000	0.0000	0.0000
19000	19000	19000	0.0000	0.0000
20000	20000	20000	0.0000	0.0000

110mA 電流測試

Fluke輸出電流	16F188測得電流		誤差%	
mA	+	-	+	-
0	0	0	0	0
1	1	1	0	0
2	2	2	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
6	6	6	0	0
7	7	7	0	0
8	8	8	0	0
9	9	9	0	0
10	10	10	0	0
11	11	11	0	0
12	12	12	0	0
13	13	13	0	0
14	14	14	0	0
15	15	15	0	0
16	16	16	0	0
17	17	17	0	0
18	18	18	0	0
19	19	19	0	0
20	20	20	0	0
25	25	25	0	0
30	30	30	0	0
40	40	40	0	0
50	50	50	0	0
60	60	60	0	0
70	70	70	0	0
80	80	80	0	0
90	90	90.1	0	0.111
100	100	100.1	0	0.1
110	110.1	110.1	0.09	0.09

6.結果總結

以 HY16F188 為主控結合內部高精度、多通道輸入、快速 ADC 的量測。不論電壓或者電流的量測，相較於市售電表，不僅僅耗電量低於一般市售電表，在精準度上也有不輸市售電表的表現。HY16F188 內部 ADC 不僅可用來量測電壓電流，也可以結合外部感測器進行其他量測，依然有相當不錯的表現。

7.附加檔案


HY16F008V03.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	更新程序	2014/12/17

```
/*---------------------------------------------------------------------------*/
/*****************************************************************************
 *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 "DrvADC.h"
#include "DrvPMU.h"
#include "HY2613.h"
#include "my define.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 key_ck,b,c,pb6,pb7;                          //按鍵用
signed int ADtemp10,ADCData,sum,ADtemp101,ADtemp100,average;//AD 用
signed int ADtemp20n,ADtemp20p,ADtemp200,ADtemp110n,ADtemp110p,ADtemp1100; //計算
及顯示用
int nu,tn,ch,ok;
int value_buf[9];
/*-------------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                           */
/*-------------------------------------------------------------------------------*/
void Delay(unsigned int num);
void InitalADC(void);
```

```
void Initial_ALL(void);
void Pb_6();
void Pb_7();
void AD3(void);
void Test1(void);
void Test2(void);
void AD01(void);
/*-------------------------------------------------------------------------------*/
/* Main Function                                                                 */
/*-------------------------------------------------------------------------------*/
int main(void)
{
    InitalI2C();
    SYS_EnableGIE(7,0x3F);  // Enable GIE(Global Interrupt)

    Ini_Display();
    ClearLCDframe();

        InitalADC();
        Initial_ALL();

    MCUSTATUSbits._byte = 0;
    while(1)
    {
          pb7=0;
          LCD_DATA_DISPLAY2(key_ck);
          if(b==0) Pb_6();
          if(pb6==1)
          {
                  pb6=0;
                  key_ck++;
                  Delay(0x2000);
                  if(key_ck>=3) key_ck=1;
          }
          if(c==0) Pb_7();
          if(pb7==1)
          {
                  pb7=0;
```

```
                switch(key_ck)
                {
                        case 0x01:Test1();break;//*20V
                        case 0x02:Test2();break;//*110mA
                }
          }
    }


  return 0;
}
void Test1(void)
{
        while(DrvGPIO_GetBit(E_PT1,7))
         {
                AD3();
                key_ck=0;
                if(nu==0)
                {
                        AD01();
                        if(sum<7010)
                        {sum=sum-1;
                        ADtemp101=100*(sum-ADtemp200)/(ADtemp20n-ADtemp200);
                        ADtemp100=ADtemp101%100;
                        if(ADtemp100>70) ADtemp101+=100;
                        ADtemp101=ADtemp101/100;
                        ADtemp101=ADtemp101*10;
                        LCD_DATA_DISPLAY(ADtemp101);
                        }
                        else
                        {
                                if(sum<10507)
                                {
                                        sum-=1;

ADtemp101=(sum-ADtemp200)/(ADtemp20n-ADtemp200);
                                        ADtemp101*=10;
                                }
```

```
                                        else
                                        {
                                                sum-=3;

ADtemp101=(sum-ADtemp200)/(ADtemp20n-ADtemp200);
                                                ADtemp101*=10;
                                        }
                                        LCD_DATA_DISPLAY(ADtemp101);
                                }
                        }
                        if(nu==1)
                        {
                                AD01();
                                if(sum<7010)
                                {sum=sum-1;
                                ADtemp101=100*(sum+ADtemp200)/(ADtemp20n+ADtemp200);
                                ADtemp100=ADtemp101%100;
                                if(ADtemp100>70) ADtemp101+=100;
                                ADtemp101=ADtemp101/100;
                                ADtemp101=ADtemp101*10;
                                LCD_DATA_DISPLAY(ADtemp101);
                                }
                                else
                                {
                                        if(sum<10507)
                                        {
                                                sum-=1;

ADtemp101=(sum+ADtemp200)/(ADtemp20n+ADtemp200);
                                                ADtemp101*=10;
                                        }
                                        else
                                        {
                                                sum-=3;

ADtemp101=(sum+ADtemp200)/(ADtemp20n+ADtemp200);
                                                ADtemp101*=10;
                                        }
```

```
                                LCD_DATA_DISPLAY(ADtemp101);
                        }
                }
        }
        Delay(0x2000);
}
/*-------------------------------------------------------------------------*/
void Test2(void)
{

        while(DrvGPIO_GetBit(E_PT1,7))
        {
                AD3();
                key_ck=0;
                if(nu==0)
                {
                        AD01();
                        ADtemp101=100*(sum+1)/(ADtemp110n-ADtemp1100);
                        LCD_DATA_DISPLAY1(ADtemp101);
                }
                if(nu==1)
                {
                        AD01();
                        ADtemp101=100*(sum+1)/(ADtemp110n-ADtemp1100);
                        LCD_DATA_DISPLAY1(ADtemp101);
                }
        }
        Delay(0x2000);
}
/*-------------------------------------------------------------------------*/

void AD01(void)
        //ADC 副程式
{
        unsigned char t;
      while(ok)                                         //ADC 中斷後 OK=1
        {
                ok=0;
```

```
                ADtemp10=(ADCData>>17);

                value_buf[8]=value_buf[7];          //平均滑動濾波處理
                value_buf[7]=value_buf[6];
                value_buf[6]=value_buf[5];
                value_buf[5]=value_buf[4];
                value_buf[4]=value_buf[3];
                value_buf[3]=value_buf[2];
                value_buf[2]=value_buf[1];
                value_buf[1]=ADtemp10;
                sum=0;
                for(t=4;t>0;t--)
                {
                        sum+=value_buf[t];
                }
                average=sum>>2;
        }
        if(average<0) average*=-1;
        sum=average;
}

/*
void AD01(void)
{
        int t;
        sum=0;
        for(t=0;t<8;t++)
        {
                ADtemp10=(ADCData>>17);
                sum=sum+ADtemp10;
        }
        sum=sum>>3;
        if(sum<0)
        {
                sum*=-1;
        }
}
*/
```

```
/*-------------------------------------------------------------------------------*/
void AD3(void)
{
        int t;
        sum=0;
        for(t=0;t<8;t++)
        {
                ADtemp10=(ADCData>>18);
                sum+=ADtemp10;
        }
        if(sum>=-30)
        {
                if(ch==0)
                {
                        nu=0;
                        ch=0;

    DrvADC_SetADCInputChannel(ADC_Input_AI01,ADC_Input_AI00);//AI00、AI01 正常配置
                        goto by;
                }
                if(ch==1)
                {
                        nu=1;
                        ch=1;

    DrvADC_SetADCInputChannel(ADC_Input_AI00,ADC_Input_AI01);//AI00、AI01 顛倒費至
                        goto by;
                }
        }
        if(sum<0)
        {
                if(ch==0)
                {
                        nu=1;
                        ch=1;

    DrvADC_SetADCInputChannel(ADC_Input_AI00,ADC_Input_AI01);//AI00、AI01 顛倒配置
                        goto by;
```

```
                }
                if(ch==1)
                {
                        nu=0;
                        ch=0;

    DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);//AIO0、AIO1 正常配置
                        goto by;
                }
        }
        by:asm("NOP");
        Delay(0x8000);
}
/*---------------------------------------------------------------------------*/
/* Hardware Communication Interrupt                                          */
/* UART/SPI/I2C Interrupt Service Routines                                   */
/*---------------------------------------------------------------------------*/
void HW0_ISR(void)
{
  unsigned char I2C_Status;

  if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT)  // Get I2C Interrupt Flag
  {
    I2C_Status=DrvI2C_GetStatusFlag();  // Get I2C Status Flag
    switch(I2C_Status)
    {
      case 0x90: //MACTFlag+RWFlag
          { /* START has been transmitted */
            DrvI2C_WriteData(I2C_TARGET);  //Send Slave Address & R/W Bit
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
            break;
          };
      case 0x84: //MACTFlag+ACKFlag
          { /* Slave A + W has been transmitted. ACK has been received. */
            DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
            break;
          };
```

```
case 0x80: //MACTFlag
    { /* Slave A + W has been transmitted. ACK has been received. */
      DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      break;
    };
case 0x30:
    {
      DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      EndFlag=1;
      break;
    };
case 0x8C: // MACTFlag+DFFlag+ACKFlag
    { /* DATA has been transmitted and ACK has been received */
      if(DataTxIndex<DataTxLen)
      {
        DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  //Send Data to Slave
        DrvI2C_Ctrl(0,0,0,0);  // Clear all I2C flag
      }
      else
      {
        if(I2C_RW == WRITE)
        {
          DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
          EndFlag=1;
        }
        else if(I2C_RW == READ)
          DrvI2C_Ctrl(1,0,0,0);  // I2C as master sends START signal
        DataTxIndex=0;
      }
      break;
    };
case 0x88: //MACTFlag+DFFlag
    { /* DATA has been transmitted and NACK has been received */
      DrvI2C_Ctrl(0,1,0,0);  // I2C as master sends STOP signal
      DataTxIndex=0;
      EndFlag=1;
      break;
```

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

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

/*-------------------------------------------------------------------------*/
/* WDT & RTC & Timer A/B/C Interrupt Service Routines
*/
/*-------------------------------------------------------------------------*/
void HW1_ISR(void)
{

}

/*-------------------------------------------------------------------------*/
/* HW2 ADC Interrupt Subroutines                                           */
/*-------------------------------------------------------------------------*/
void HW2_ISR(void)
{
        DrvADC_ClearIntFlag();
        ADCData=adc_08;
        ok=1;
}

/*-------------------------------------------------------------------------*/
/* CMP/OPA Interrupt Service Routines
*/
```

```
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* PT1 Interrupt Service Routines                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{
        DrvGPIO_ClearIntFlag(E_PT1,0xc0);          //Clear PT1 interrupt flag
          b=DrvGPIO_GetBit(E_PT1,6);
          c=DrvGPIO_GetBit(E_PT1,7);

          asm volatile("sethi $r0, 0xc0000");
          asm volatile("ori  $r0, $r0, 0x003f");
          asm volatile("mtsr $r0, $INT_MASK");
          asm volatile("movi $r0, 0x70009");
          asm volatile("mtsr $r0, $PSW");
}

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

}

/*---------------------------------------------------------------------------*/
/* Software Delay Subroutines                                                */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
  for(;num>0;num--)
    asm("NOP");
}
```

```
void InitalADC(void)
{
  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);//ADC Input
  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
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_BandgapEnable();
  DrvPMU_REFO_Enable();
  DrvPMU_AnalogGround(Enable);//ADC analog ground source selection.
                              //1:Enable buffer and use internal source
  DrvPMU_LDO_LowPower(0);     //VDD LDO with low power control.
                              //0 : Normal;1 : Low power
  Delay(0x1000);
  //Set ADC interrupt
  DrvADC_EnableInt();
  DrvADC_ClearIntFlag();
  DrvADC_Enable();
}
void Initial_ALL(void)
{
        pio_00=0xff00ff00;                  //關閉所有 IO
        pio_04=0xff00ff00;
        pio_08=0x00;
        pio_10=0xff00ff03;
        pio_14=0xff00ff03;
        DrvGPIO_Open(E_PT1,0xc0,E_IO_INPUT);   //PT_6-7 為 input
        DrvGPIO_Open(E_PT1,0xc0,E_IO_PullHigh); //enable PT1_6-7 pull high R
```

```
        SYS_EnableGIE(7,0x3f);          //Enable GIE (Global Interrupt Enable)
        DrvGPIO_ClearIntFlag(E_PT1,0xc0);         //clear PT1.6~7 interrupt flag

        DrvGPIO_ClkGenerator(E_HS_CK,1);          //GPIO CLK enable
        DrvGPIO_Open(E_PT1,0xc0,E_IO_IntEnable);  //PT1.6~7 interrupt enable
       //PT1.6~7 interrupt  trigger method is negative edge
        DrvGPIO_IntTrigger(E_PT1,0xc0,E_N_Edge);
        DrvCLOCK_EnableHighOSC(E_INTERNAL,50);       // Select HAO 4MHz
        ADtemp200=0;
        ADtemp1100=0;
        ADtemp110n=1405;
        ADtemp110p=1405;
        ADtemp20n=7;
        ADtemp20p=7;
        nu=0;
        key_ck=0;
        tn=2;
        ch=0;

}
void Pb_7()                             //按鍵副程式
{
        if(c==0)
        {
                pb7=1;
                c=1;
                while(1)                //如按鍵持續按下無放開，則無法離開
                {
                        c=DrvGPIO_GetBit(E_PT1,7);
                        if(c==1)break;
                }

        }
}
/*-------------------------------------------------------------------------*/
void Pb_6()
{
```

```
        if(b==0)
        {
                pb6=1;
                b=1;
                while(1)        //如按鍵持續按下無放開，則無法離開
                {
                        b=DrvGPIO_GetBit(E_PT1,6);
                        if(b==1)break;
                }

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

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