HY16F188 Voltage Current Meter Application Manual

Index

Note:

1、 With the advancement of our product, content in this instruction manual might be amended without prior notification. Customers are welcomed to constantly download at our company's website http://www.hycontek.com

2、 Regarding to illustrations and applied circuits in this specification manual, our company is not responsible for problems arise from industrial ownership in the third party.

3、 In situations where our product is applied individually, our company guarantees that its performance, typical application and function are consistent with conditions specified in the instruction manual. In situations where our product is applied in our client's product or equipment, our company does not guarantee the aforementioned conditions. We further recommend our clients to conduct sufficient evaluation and testing before doing so.

4、 Please pay extra note to application conditions in input voltage, output voltage, and load current, so that internal IC consumption does not surpass allowable consumption in the packaging. If our clients were to use our products under conditions where set values in the instruction manual were surpassed, even if immediately, our company is not responsible for the loss thus arise.

5、 Even though antistatic protective circuit is installed within this product, please do not exert excessive electrostatic which surpasses protective circuit performance.

6、 Without written permission, the product in the specification manual is not allowed to be applied in highly reliable electrical circuit, including instruments or devices affecting human bodies, such as medical instruments, disaster prevention instruments, vehicle instruments, loading instruments, and aircraft instruments.

7、 Our company has been dedicated to increasing quality and reliability in our products, thus possible failure rates present in all our semiconductor products. These failure rates might lead to some personal or fire accidents. Therefore, while designing products, please pay extra attention to redundancy design and adopt safe indicator, so as to prevent such accidents from happening.

8、 Without our company's permission, contents in this specification manual are prohibited to be transferred or copied for other purpose.

1. Content Introduction

Regarding to industrial application, voltage and current measurement are the most fundamental yet the most important factors. As to industrial measurement in pressure, temperature and humidity, sensor is applied to convert signals into voltage or current, which are in turn analyzed by electrical instrument before displaying on the meter. Therefore, there is a significant importance in measuring accurate voltage and current. This article mainly discusses about HYCON HY16F188 Series chip application in voltage and current measurement. HY16F188 chip is equipped with internal-integrated accurate display ΣΔ ADC. In addition, ADC output frequency can reach a maximum of 10KHZ. Since IC HY2613B is driven by external LCD, HY16F188 possesses a relatively high accuracy in measuring voltage and current.

2. Theory Explanation

Voltage Measurement:
Electrical circuit chart is demonstrated in illustration 1. This circuit is a simple voltage division circuit, with a voltage division ratio being 20:1. Due to program setting relationship, maximum voltage difference between AIO0 and AIO1 is 1.2V. Therefore, upper limit to voltage measurement is 20V. Current measurement: electrical circuit chart is presented in illustration 2. Once current source flows through 10Ω resistance, a voltage drop is generated. By measuring voltage drop, electrical current can be inferred.

Resolution is separated into external resolution and internal resolution. External resolution is the voltage ratio between maximum output voltage and minimum voltage to be identified. Minimum measurement voltage in this application is 10mV.
Generally, internal resolution defined by our visual method refers to one grid rolling in LCD display after being processed by our software. Under this moment, full-range grid indicate internal resolution, with one grid representing a signal 2-3 times higher than RMS Noise. The smaller the resolution ratio between internal and external resolution, the higher the accuracy will be in voltage current meter. However, there is still limitation to resolution ratio between internal and external resolution. For example, full-range voltage difference is 1.1V. To reach 2000 count, voltage current meter with a 1:10 resolution ratio between internal and external resolution must process a minimum signal of 1.1V/(2000×10)=55uV, if hasn't gone through signal amplification. SD24 can process a minimum signal value of 65nV. As a result, it is extremely easy to complete accurate measurement in this specification.

To test whether ADC performance can achieve the required specification, we apply RMS Noise to infer whether externality can stabilize internal resolution ratio. Regarding to development in electronic product, the bottle neck for HY16F188 chip to achieve the maximum internal resolution lies in Input RMS Noise rather than ADC resolution. After being amplified by PGA and AD multiplying regulator (PGA=32, ADGN=4), HY16F188 ADC signal is subjected to a condition where OSR=32768 outputs 10 ADC values per second, further leading to an Input RMS Noise being approximately 65nV. However, since Input Noise is primarily composed of Thermal Noise, we can further reduce Input Noise through average software processing.

If we apply 8 output software to evenly processing its Input RMS Noise, 40nV can be achieved after considering other noise factors, with 3 times RMS Noise representing approximately 1 grid rolling, or 120nV. After applying 2.4V drive voltage, 1mV/V full-range voltage difference can achieve 2.4mV. Under this situation, we can obtain a 20000 counts internal resolution.

2.1 Measurement Theory

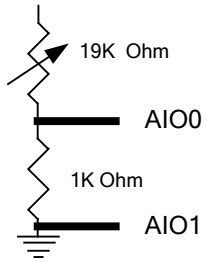


Illustration 1: Voltage Measurement Circuit Chart

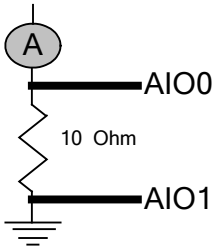
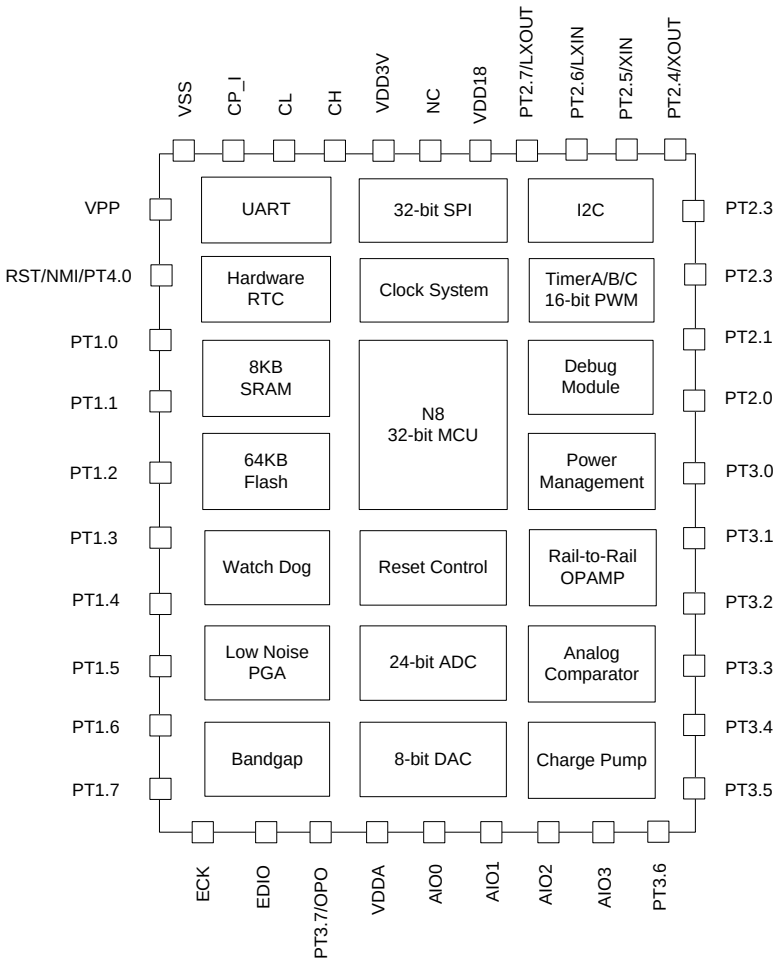


Illustration 2: Current Measurement Circuit

2.2 Control Chip

single-chip microcomputer: HY16F Series 32 Bit High Performance Flash Single-Chip Microcomputer (HY16F188)



HYCON HY16F Series 32 Bit High Performance Flash Single-Chip Microcomputer (HY16F188)

(1) Apply the most advanced Andes 32 Bit CPU Kernel N801 Processor.
(2) Voltage operation range is 2.4 – 3.6V, while working temperature range is -40 ℃~85℃.
(3) Support for external 20MHz quartz crystal resonator or internal 20MHz highly accurate RC oscillator, with various CPU working clock switching options, further enabling user to achieve the optimum power saving plan.
(3.1) Operating Mode: 350uA@2MHz/2(3.2)Idle Mode: 10uA @ 32KHz/2(3.3)Sleep Mode: 2.5uA
(4) Program Memory: 64KBytes Flash ROM
(5) Data Memory: 08KBytes SRAM
(6) Equipped with BOR and WDT, so as to prevent CPU crash.
(7) 24-bit Highly Accurate ΣΔADC Analog to Digital Converter
(7.1) Installed with PGA (Programmable Gain Amplifier), capable of achieving a maximum of 128 times amplification.
(7.2) Installed with temperature sensor.
(8) Lowest Input Noise Operational Amplifier OPAMP.
(9) 16-bit Timer A
(10) 16-bit Timer B Module with PWM Generation Function
(11) 16-Bit Timer C Module with Digital Capture/Compare Function
(12) Hardware Series Communication SPI Module
(13) Hardware Series Communication I2C Module
(14) Hardware Series Communication UART Module
(15) Hardware RTC Clock Function Module
(16) Hardware Touch KEY Function Module

3. System Design

3.1 Hardware Explanation

Regarding to voltage current measurement application, HY16F188 integral circuit includes S2 and S3 button and LCD display module, as demonstrated in illustration 3.

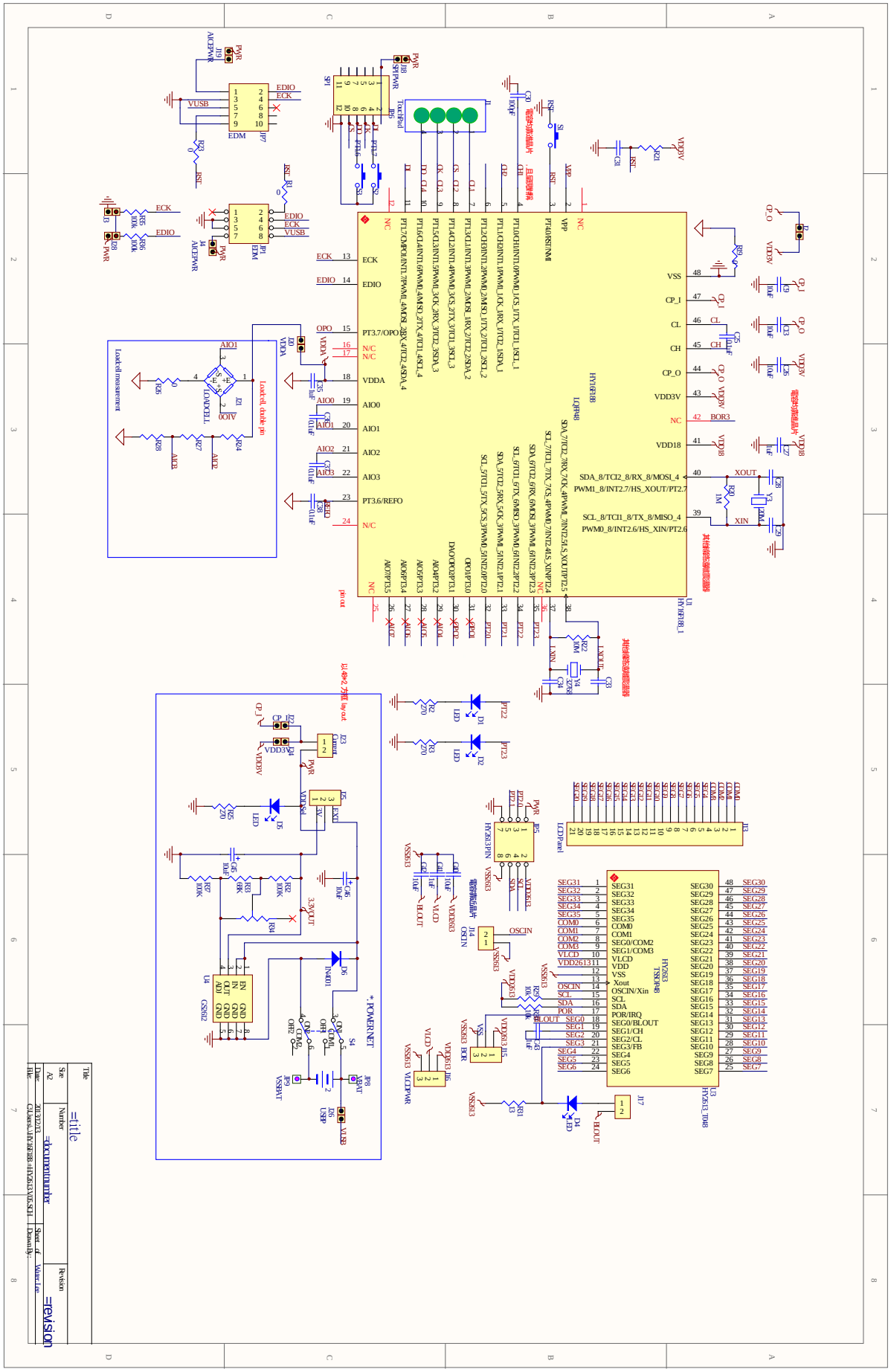
(A) Central Processing Unit (CPU):
HY16F188 (Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + UMC 64K Flash)

(B) Display Chip:HY2613 (HYCON LCD Driver LCD Segment 4X36)

(C) Power Supply Circuit: 5.0V convert to 3.3V Power Supply System

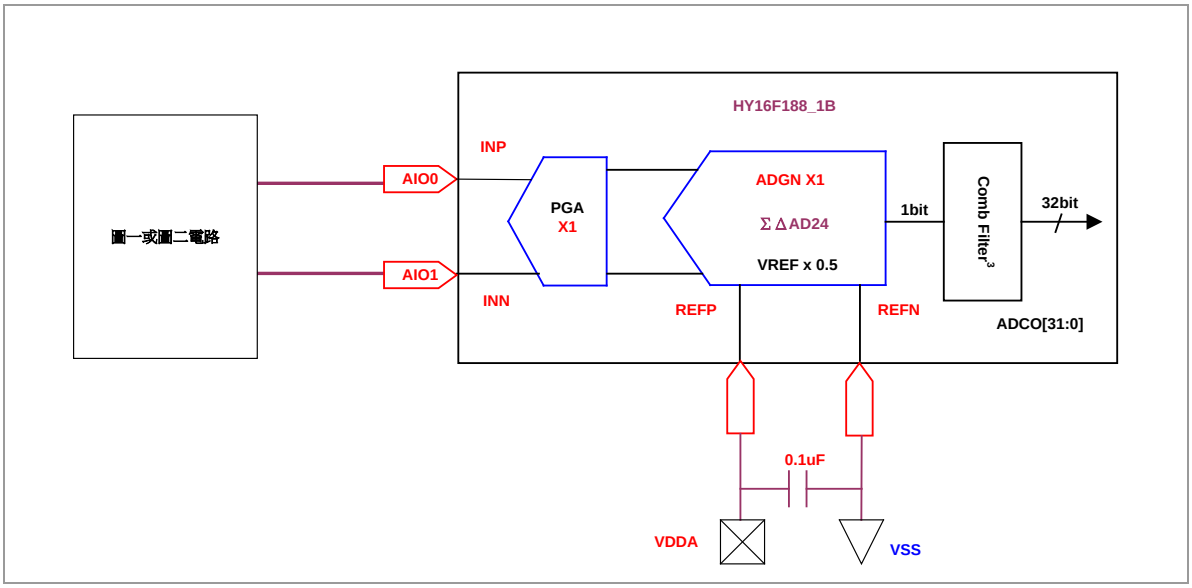
(D) Analog Sensory Module: Internal ADC

(E) Through EDM connection, online burning and ICE connection circuit can support for online burning simulation. It is equipped with a strong C Platform IDE, HYCON analog software analysis instrument and GUI support.



3.2 Function Explanation

ADC internal PGA amplifies for 1 time, while ADGN amplifies for 1 time as well. Referential voltage is provided through VDDA-VSS, resulting in ΔVR_I=1.2V.



3.2.1 Voltage Measurement

Under voltage measurement mode, measurement range is between ±20V. While conducting measurement, please comply with circuit chart demonstrated in illustration 1. Display can reach 1mV, while accuracy can reach 10mV.

3.2.2 Current Measurement

Current measurement range is between ±10A. While conducting measurement, please comply with circuit chart demonstrated in illustration 2. Display and measurement accuracy are both 0.1mA.

4. Operation Procedure

4.1 Operation Method

After starting up, initialization and HYCON display will be conducted in the first place.

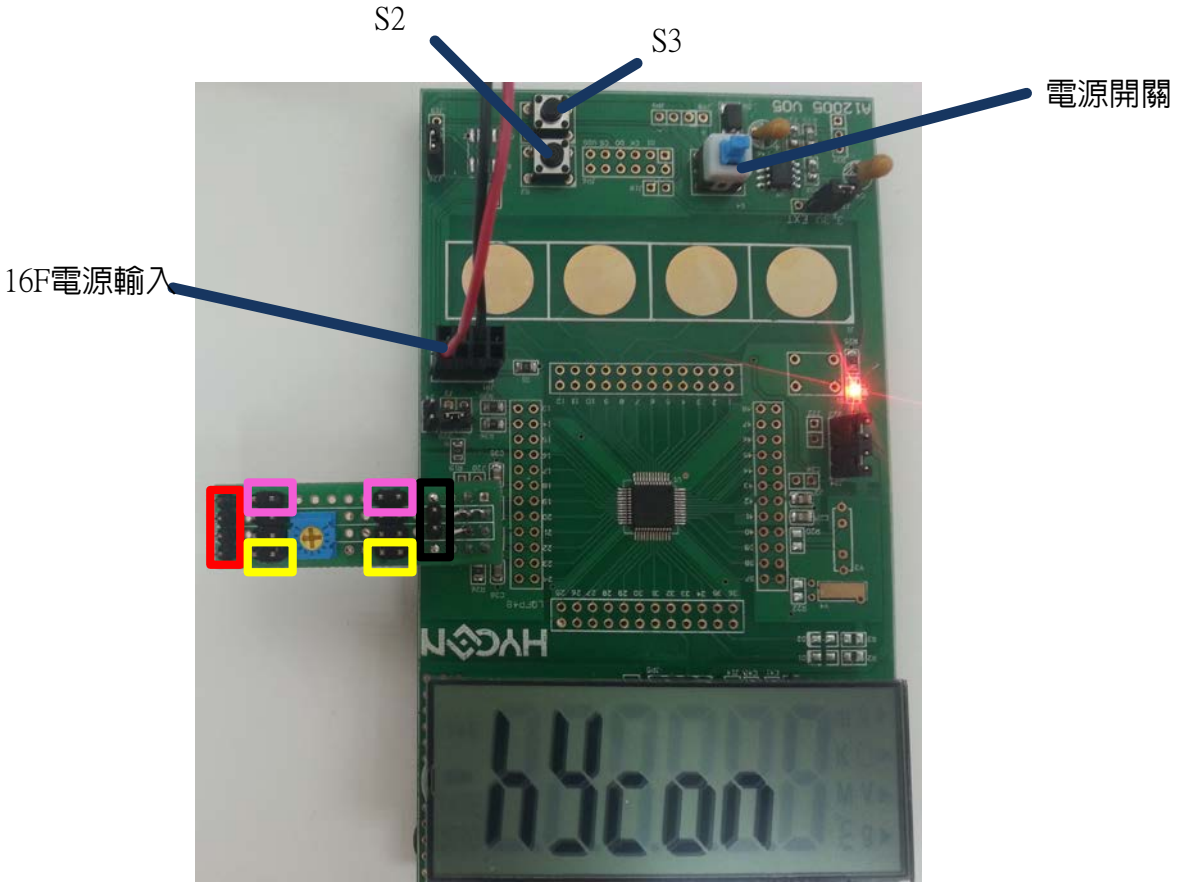
Through S3 button, measurement optional converting can be conducted, with S2 representing confirmation.

Selection Display:

20V stands for ±20V measurement, with a need to comply with the externality so as to implement appropriate short circuit.

10A stands for ±10A measurement, with a need to comply with the externality so as to implement appropriate short circuit.

Under measurement mode, please click S2 to leave for initial status.



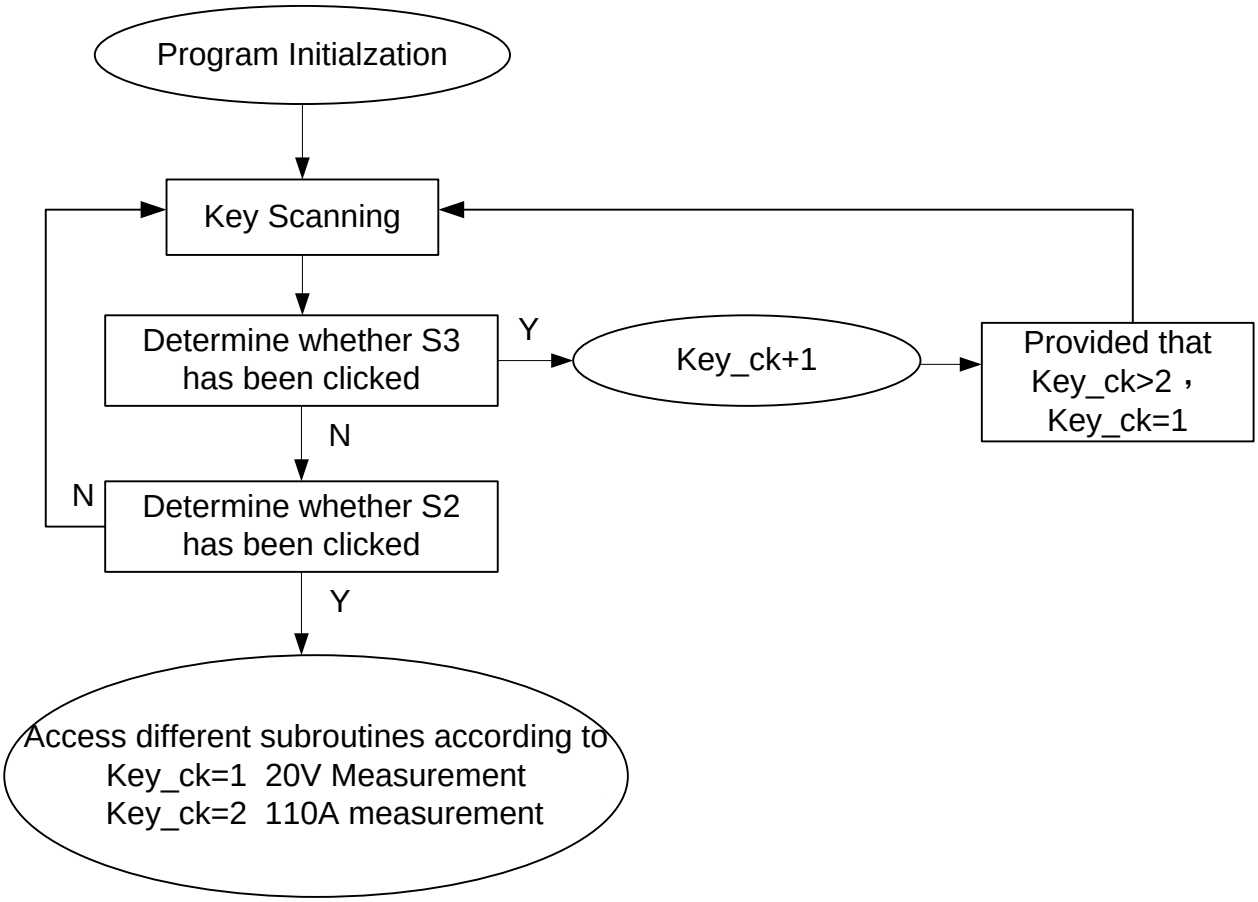
Portion within the red frame stands for measurement +.

Portion within the black frame stands for measurement -.

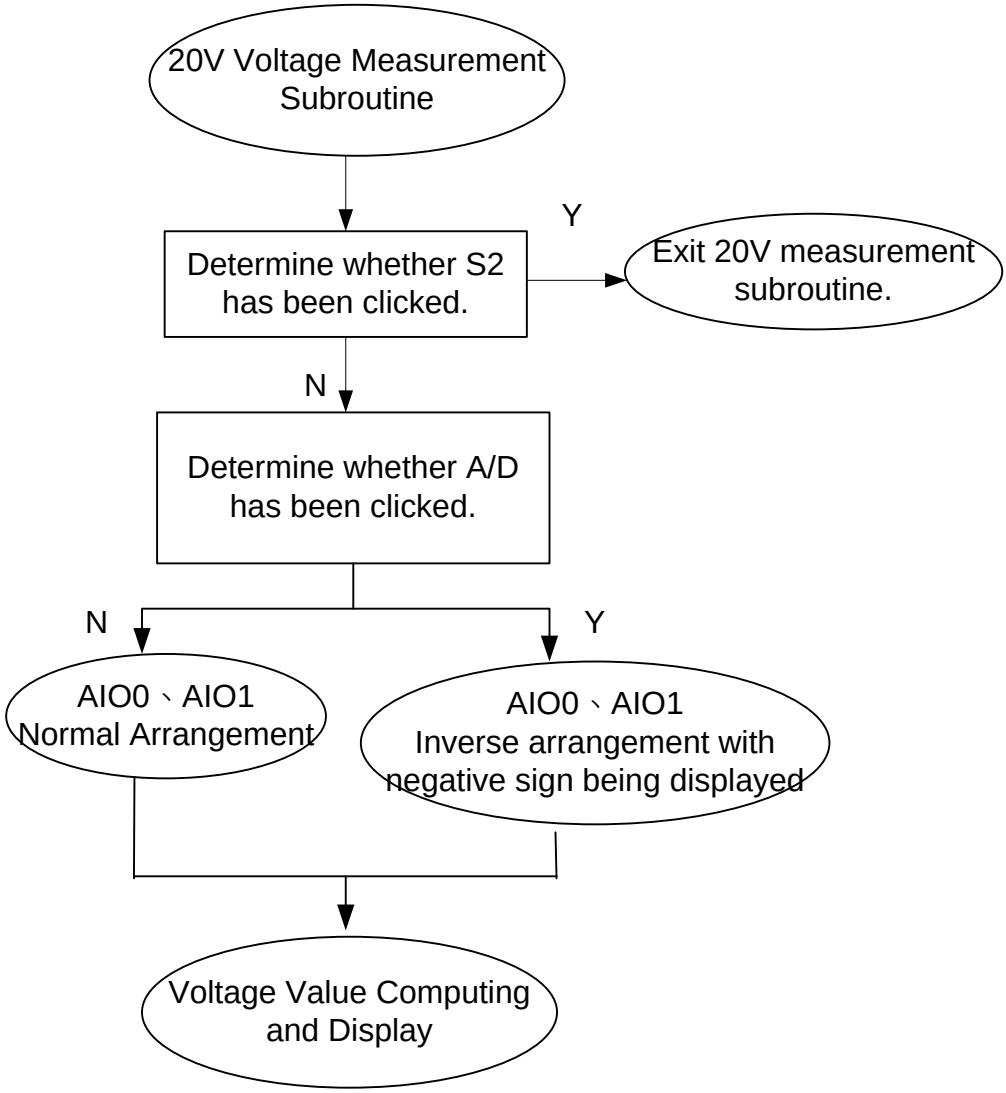
While selecting current measurement, make sure that portions within the two pink frames are short circuited.

While selecting voltage measurement, make sure that portions within the two yellow frames are short circuited.

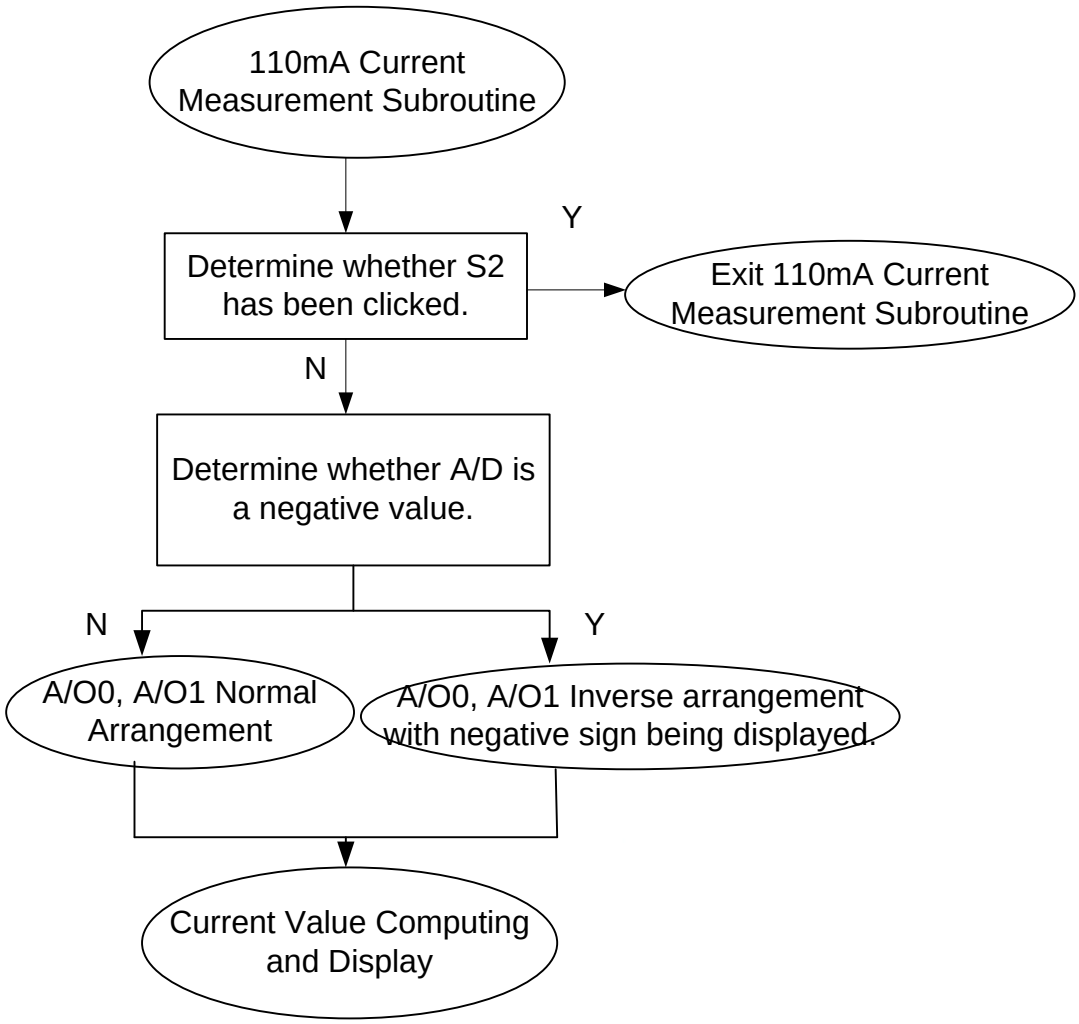
4.2 Program Procedure



4.3 Voltage Measurement Subroutine



4.4 Current Measurement Subroutine



5. Technical Specification

(1) Power Supply Connecting Point: 3.0V
(2) Power Consumption: Working Mode 1mA
(3) Measurement Precision: Voltage: 10 (mV) and Current 0.1 (mA)
(4) Applicable Range: Various Current Voltage Measurement
(5) Working Temperature: -40℃ ～ +85℃
(6) Storage Temperature: -55℃ ～ +125℃
(7) Relative Humidity: ＜95% (Under 20±5℃ Condition)

5.1 Measurement Data

20V Voltage Measurement

Fluke Output Voltage	16F188 Measured Voltage		Error %	
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 Current Measurement

Fluke Output Current	16F188 Measured Current		Error %	
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. Result Conclusion

HY16F188 combines internal highly accurate, multi-channel input, and speedy ADC measurement. Compared to other meters sold in the market, our current and voltage measurement meter consumes lower power and exhibit outstanding measurement accuracy in voltage or current measurement. HY16F188 internal ADC is not only applied in voltage and current measurement, but can also be connected to external sensors, so as to conduct other measurement. All measurements conducted by our meter exhibit satisfactory performance.

7. Attached Document


HY16F008V03.zip

8. Referential Literature

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

9. Amendment Record

Greater differences in the document are presented below, with variation in punctuation and font excluded.

Version	Page Number	Amendment Summary	Date
V1.0	ALL	Initial Version Publication	2013/10/30
V2.0	ALL	Procedure Amendment	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;                          //button
signed int ADtemp10,ADCData,sum,ADtemp101,ADtemp100,average;//AD use
signed int ADtemp20n,ADtemp20p,ADtemp200,ADtemp110n,ADtemp110p,ADtemp1100;
//use by calculate & display
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 sub-program
{
          unsigned char t;
        while(ok)                                       //ADC after Interrupt OK=1
          {
                    ok=0;
                    ADtemp10=(ADCData>>17);
```

```
                value_buf[8]=value_buf[7];              // sliding filtering average
                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_AIO1,ADC_Input_AIO0);//AIO0、AIO1
Normal configuration
                        goto by;
                }
                if(ch==1)
                {
                        nu=1;
                        ch=1;

    DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);//AIO0、
AIO1Reversed
                        goto by;
                }
        }
        if(sum<0)
        {
                if(ch==0)
                {
                        nu=1;
                        ch=1;

    DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);//AIO0、
AIO1Reversed configuration
                        goto by;
                }
```

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

    DrvADC_SetADCInputChannel(ADC_Input_AIO1,ADC_Input_AIO0);//AIO0、AIO1
Normal configuration
                            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;                          //close all the IO
        pio_04=0xff00ff00;
        pio_08=0x00;
        pio_10=0xff00ff03;
        pio_14=0xff00ff03;
        DrvGPIO_Open(E_PT1,0xc0,E_IO_INPUT);          //PT_6-7 is 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()                                     // Sub-program button
{
        if(c==0)
        {
                pb7=1;
                c=1;
                while(1)            //If continue put the button, may not sign out
                {
                        c=DrvGPIO_GetBit(E_PT1,7);
                        if(c==1)break;
                }

        }
}
/*-------------------------------------------------------------------------*/
void Pb_6()
{
        if(b==0)
        {
                pb6=1;
                b=1;
                while(1) // If continue put the button, may not sign out
                {
                        b=DrvGPIO_GetBit(E_PT1,6);
                        if(b==1)break;
```

```
                    }

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

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