HYCON TECHNOLOGY

HY16F188
Touch Precision Kitchen Scales Application Manual

Index

1. CONTENT INTRODUCTION 4
2. THEORY EXPLANATION 4
2.1 Control Chip 6
3. SYSTEM DESIGN 8
3.1 Hardware Explanation 8
3.2 Circuit Explanation 10
3.3 Software Explanation 13
4. OPERATIONAL PROCEDURE 14
4.1 Operational Method 14
4.2 Main Program Procedure 18
4.3 Subroutine Calibration 19
5. TECHNICAL SPECIFICATION 21
6. MEASUREMENT RESULT 21
7. RESULT CONCLUSION 21
8. ATTACHED DOCUMENT 22
9. REFERENTIAL LITERATURE 22
10. AMENDMENT RECORD 22

Notes:

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.

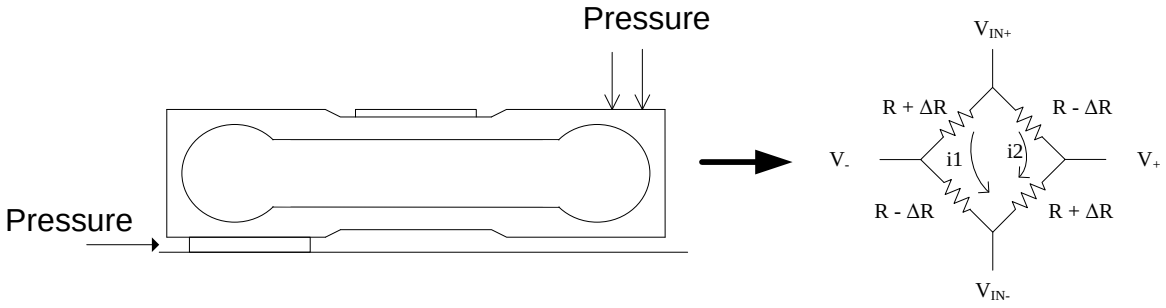
1. Content Introduction

In our daily lives, electronic scale have gradually replaced traditional spring scale and weighing scale, such as electronic price computing scale and electronic weighing scale. Major components applied to designing electronic scale products include sensor, ADC and MCU single chip. In this article, the designed electronic scale has applied pressure sensor (Load Cell) in converting physical quantity-pressure into voltage signal, which is in turn transferred into numerical display. Since voltage belongs to analog quantity, ADC must be applied to convert it into digital signal. At the same time, MCU single chip should also be applied to control signal processing and display function in electronic scale motherboard.

Highly accurate SD 24 Bit ADC, programmable amplifiable PGA, and multi-segment voltage stabilizer output are built within HYCON HY16F188 control chip, which has greatly simplified PCB peripheral circuit. SD 24 AD converter possesses high resolution and low temperature drift, which facilitate accurately analog to digital converting. Despite mediocre output speed, it is highly applicable to electronic scale, which has a rather low requirement for converting speed.

2. Theory Explanation

Basic theory in load cell is to attach a strain gauge composed of bridge resistance onto the aluminum rod, or a Wheatstone bridge, as presented in illustration 2-1. Since four resistances on the electrical bridge share common resistances value, electrical bridge is balanced once voltages are exerted onto both terminals at VIN+ and VIN-, where V+ = V-.



Voltage variation at both terminals in signal generated from ΔR variation is:

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

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

Resolution is separated into two parts, external resolution and internal resolution. External resolution equals to ratio between output voltage in full-range load cell and voltage generated from minimum weight to be identified, with minimum weight being defined as 1g, 0.5g, or 0.1g. Internal resolution is an important indicator to electronic scale level. Generally, internal resolution determined by our visual method is one grid rolling on LCD screen after software processing. Under this moment, full-range grid is equaled to internal resolution, with one grid representing signal which is 2-3times higher than RMS Noise.

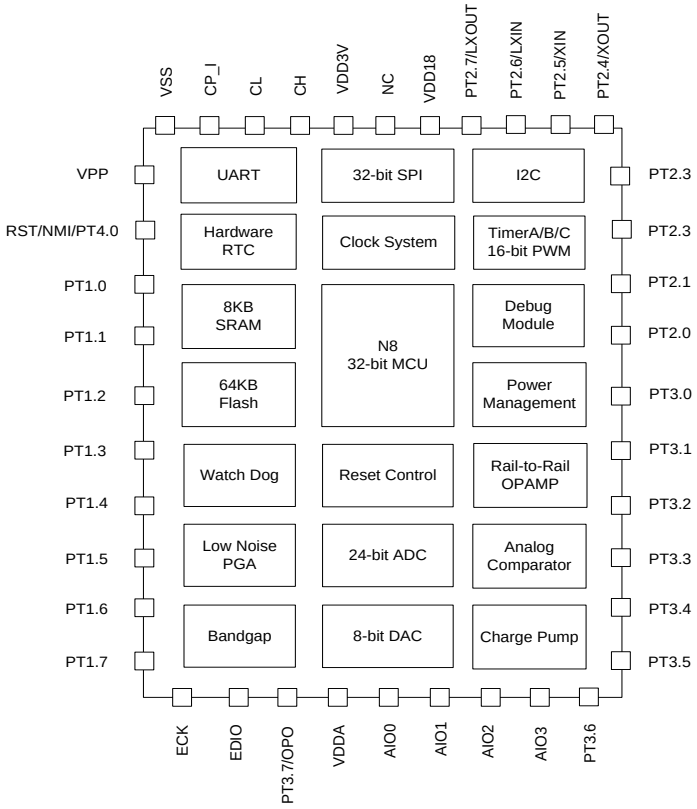
With lower resolution ratio comes higher electronic scale accuracy. However, there are some restrictions to ratio between internal and external resolution. For example, full-range voltage difference in load cell equals to 3mV. In order to reach 3000 counts, resolution ratio in the electronic scale is 1:10. Without signal amplification, minimum signal to be processed must be 3mV / (3000X10) = 0.1 μV, while minimum signal to be processed by SD24 will be approximately 65nV. Therefore, if ratio between internal and external resolution were to be further reduced, signal which cannot be identified by ADC will be generated. If OPAMP were to be applied, cost will increase. As a result, ratio between internal and external resolution must be kept at a certain range.

Whether chip ADC performance can achieve specification requirement largely depends on whether external resolution stabilize internal resolution ratio calculated through RMS Noise. Regarding to electronic scale, the bottleneck for HY16F88 chip to achieve maximum internal resolution lies in Input RMS Noise instead of ADC resolution. After being amplified by PDA and AD amplifiers (PGA=32; ADGN=4), HY16F188 ADC signal to be measured possesses an Input RMS Noise of 65nV under a condition where OSR=32768 output 10 ADC values per second. Since its Input Noise is mainly composed of Thermal Noise, we can apply average software processing to further reduce Input Noise.

If we apply average software with 8 outputs to process its Input RMS Noise, 40nV can be acquired. Three times RMS Noise represents approximately one grid rolling, which is 120nV. While applying 2.4V Load Cell drive voltage, voltage difference in full range 1mV/V Load Cell can achieve 2.4mV. Therefore, we can obtain 20000 Counts internal resolution under this condition.

2.1 Control Chip

Single-Chip Microcomputer Introduction: 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 latest Andes 32 Bit CPU Kernel N801 Processor.
(2) Voltage operational range is between 2.4V and 3.6V, while working temperature range is between -40℃ and 85℃.
(3) Support external 20MHz quartz crystal resonator or internal 20MHz highly accurate RC oscillator, with various CPU working clock converting 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 Digital Converter
(7.1)Build-in PGA(Programmable Gain Amplifier)can achive amplification maximumly up to 128 times.(7.2)Build-in Temperature sensor TPS
(8) ultralow Input Noise Operational Amplifier OPAMP
(9)16-bit Timer A
(10) 16-bit Timer B Module possesses PWM wave shape generation function
(11) 16-Bit Timer C Module possesses Digital Capture/Compare Function

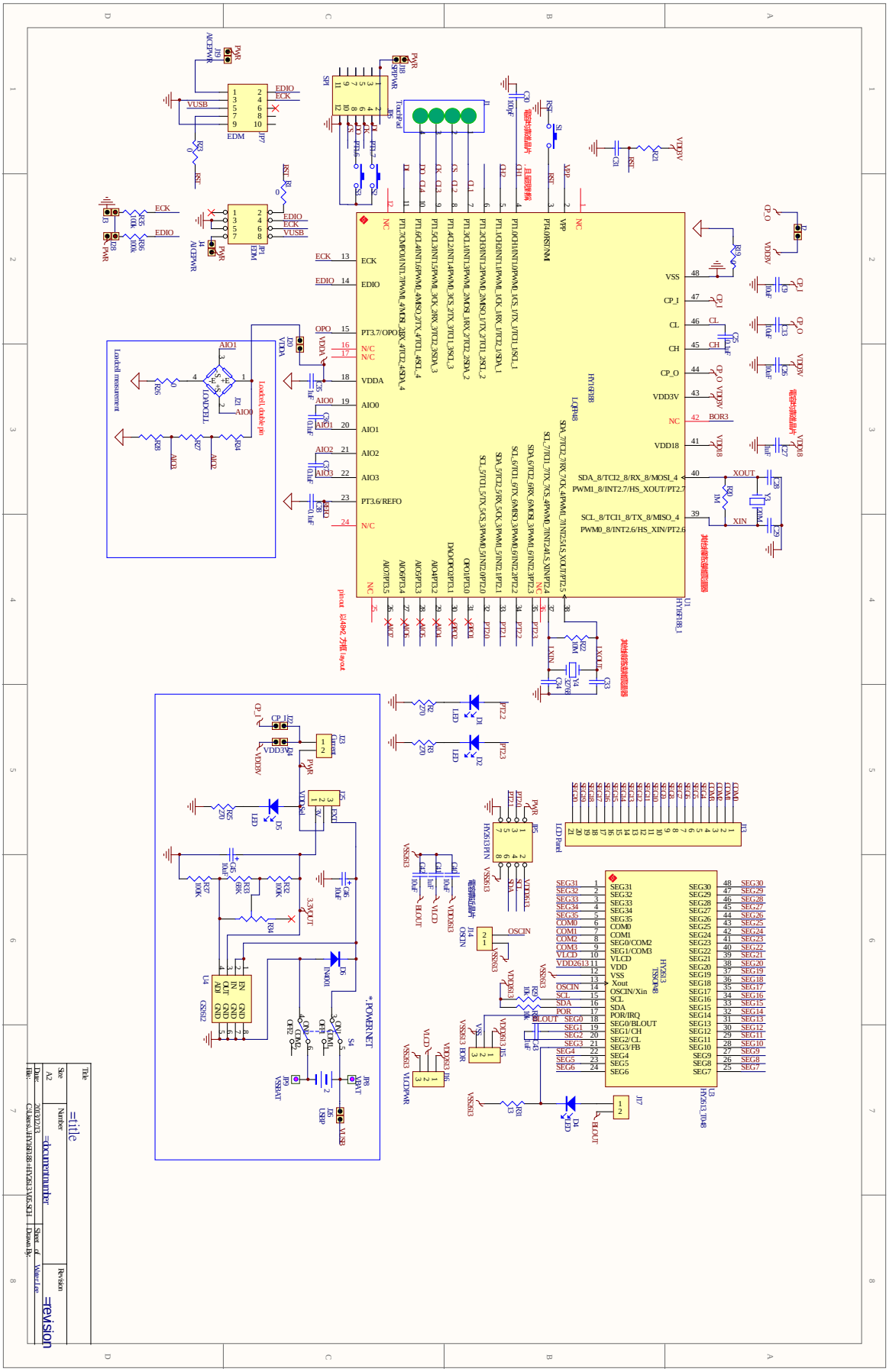
(12) Hardware Series Communication SPI Module
(13) Hardware Series Communication I2C Module
(14) Hardware Series Communication UART Module
(15) Hardware RTC Clock Functional Module
(16) Hardware Touch KEY Functional Module

3. System Design

3.1 Hardware Explanation

Regarding to HY16F188 application in precision kitchen scale, integral circuit includes four touch keys and LCD display module.

(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 to 3.3V Power Supply System
(D) Analog Sensor Module: Internal ADC
(E) Through EDM connection, online burning and ICD connecting circuit can support online burning simulation. It is equipped with a strong C platform IDE, HYCON analog analytical instrument and GUI support.

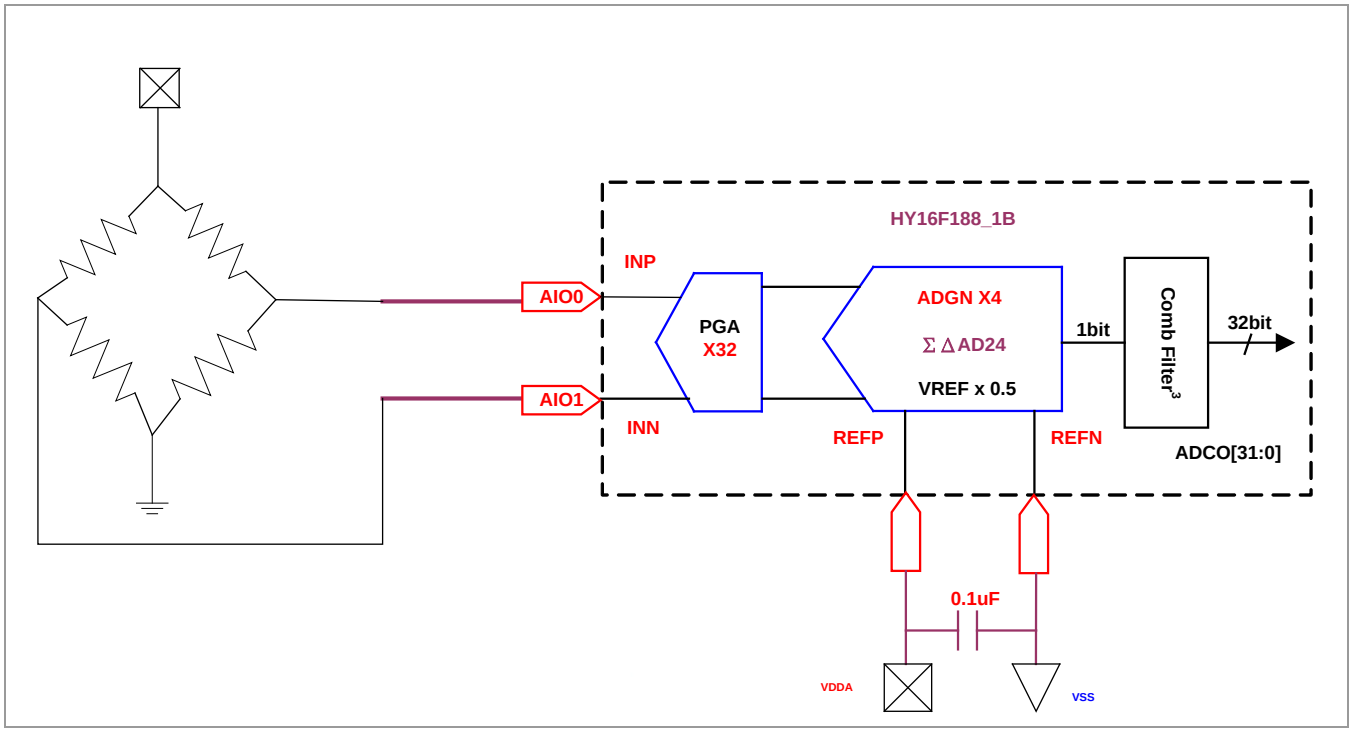


3.2 Circuit Explanation

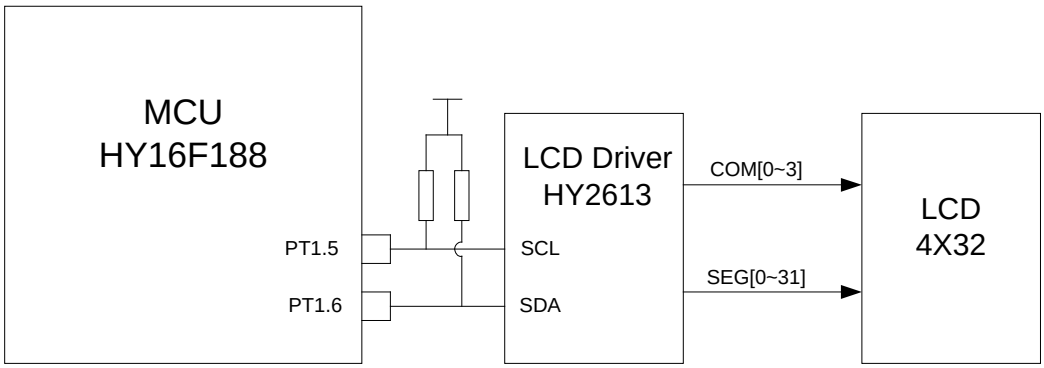
ADC Measurement Circuit

ADC internal PGA is amplified for 32 times, while ADGN is amplified for 4 times.

Referential voltage is provided by VDDA-VSS, ΔVR_I=1.2V.

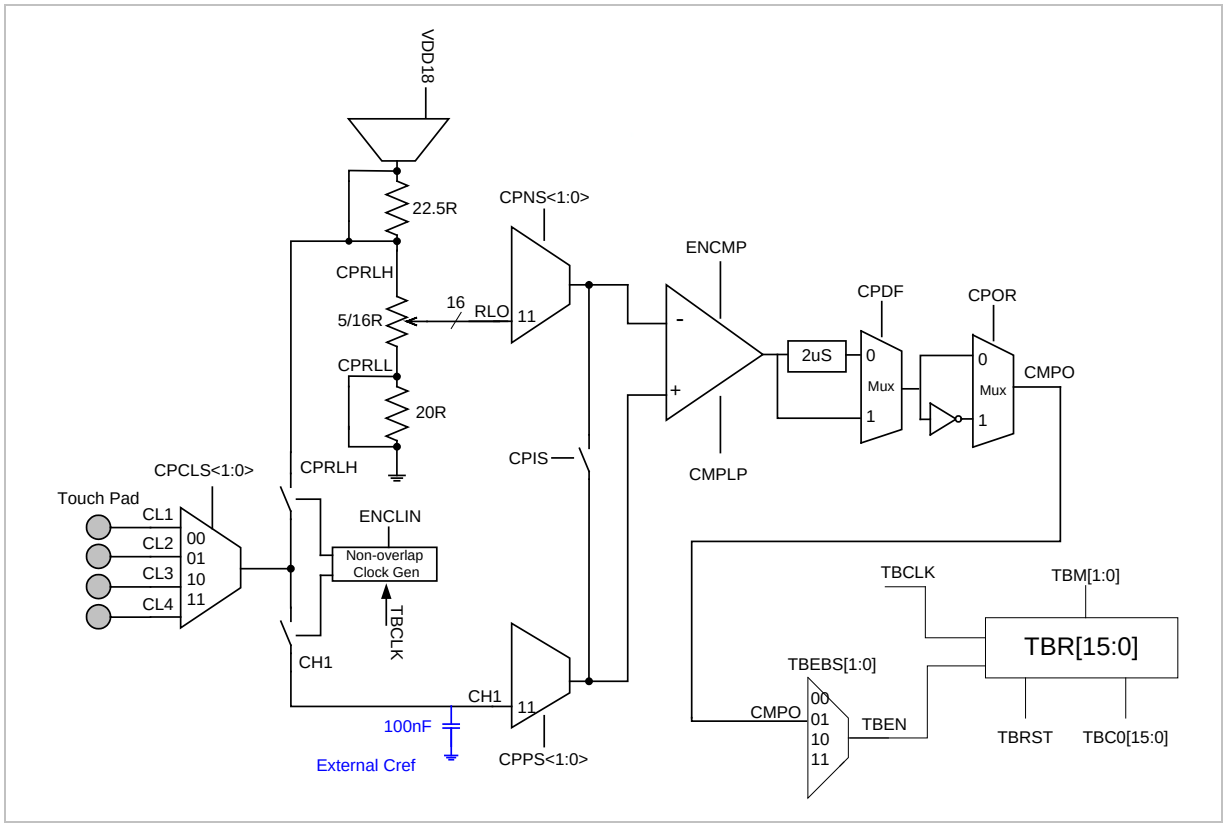


LCD Driver Route



Through communication between IIC and LCD driver, MCU circuit becomes extremely easy, and convenient for operation. Once data is delivered to LCD driver HY2613, MCU can deal with other things, with information update becoming easy.

Inbuilt Hardware Touch Module (with analog comparator block being applied)



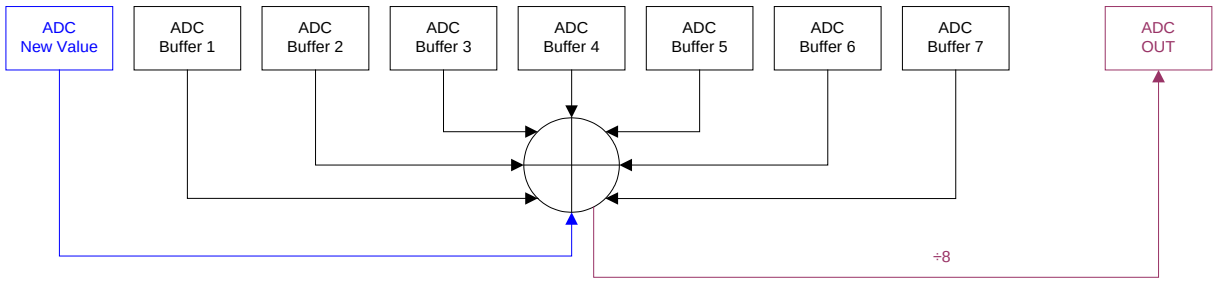
As shown in the illustration above, peripheral circuit connection regarding to touch key is rather simple. Insert a referential capacity Cref=10nf into CMP positive input terminal CH1; CMP positive input terminal arrangement is CH1, which is connected to terminal CH1 at the touch key pad; negative input terminal arrangement is RLC, which is connected to output terminal RLO at NON-OVERLAP; NON-OVERLAP voltage option is VDD18=1.8V, while CRLS=1 short circuits resistances 22.5R and 20R. NON-OVERLAP voltage division output is 1/16 R; Start up TMB with counting source being CMPO. Through CPIS=1 setting, short circuit CMP input terminal. Allow electrical amount on CH1 Cref capacity to flow through RLO before connecting to VSS, so as to conduct full electrical discharge. Startup comparator and TMB to initiate the counting. Startup NON-OVERLAP so as to facilitate VDD charging to the touch pad. Due to NON-OVERLAP switching function, touch PAD charges CH1 Cref, further facilitating voltage at CH1 terminal to increase gradually. Once voltage at CH1 terminal achieves RLO electrical level, comparator output is transformed into CMPO=0, further generating CMP interruption label bit. Stop TMB counting and record TMBR counting value. Compared to counting critical value of the touch key, if data is below critical value, touch pad is touched. By contrast, if the data is above critical value, touch pad is untouched. Scan different touch pads respectively.

3.3 Software Explanation

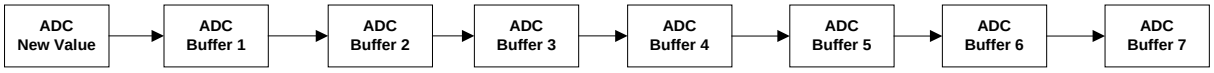
ADC Data Processing

ADC setting amplifies input signal ΔSI for 128 times. With data output frequency being ADC-CK/32768, it can output 10 data per second and finally adopt effective 18 Bit (users are free to make adjustment). After intercepting original data 18 Bit, average slide filtering can be processed. Average value is conducted every eight data, with the acquired average value serving as an ADC final converting value. Average slide filter is demonstrated in the illustration.

Since small signal is amplified 128 times, ADC output Bit can only achieve 18 Bit. If software average approach were to be applied, value can be greatly stabilized by increasing ADC resolution by 1~2Bit. Please add new ADC value with seven ADC Buffer values and divide the sum by 8 before outputting to ADC OUT as demonstrated in the illustration. By averagely outputting these 8 ADC data, Noise can averagely elevate signal output Bit number.



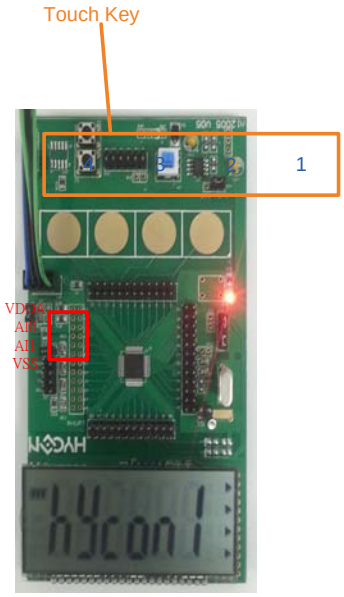
After ADC average output, ADC new value is transmitted to Buffer 1, from buffer 1 to buffer 2,…, and from buffer 6 to buffer 7, as demonstrated in the illustration below.



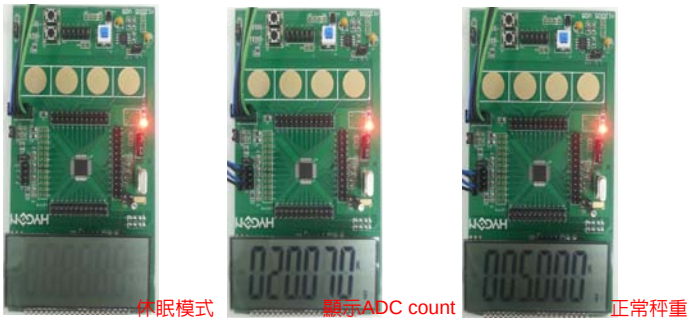
4. Operational Procedure

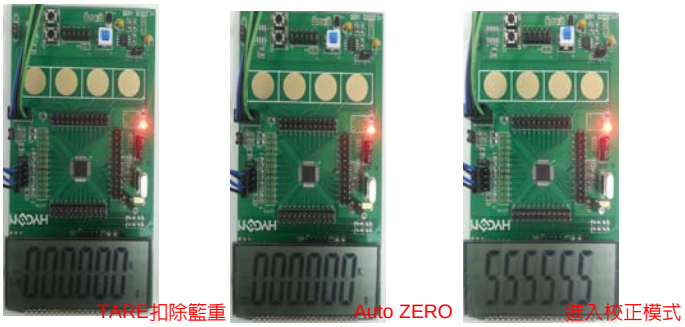
4.1 Operational Method

Upon starting up, HYCON will be displayed if internal Flash ROM has been calibrated. After catching FlashROM calibration value and determining whether zero point needed to be updated, measurement mode will be automatically accessed.



Under measurement mode, pressing Touch Key 1 will turn off LCD display and access sleep mode, pressing Touch key 2 converts LCD display from g to AD count, pressing 3 eliminates basket weight, while pressing Touch key 4 accesses calibration mode.





TARE will be displayed under basket weight elimination, while ZERO will be displayed under Auto zero. A backward counting from 5 to 1 will manifest before accessing calibration mode.

Calibration mode:
Under calibration mode, no unit will be displayed.
Procedure 1: Set bit number (24-1) adopted by ADC. Preinstall 18bit,display AD count of now at ordinary time.Bit count can be changed ang displayed through Touch Key1.2.Upon setting completion, click touch key 4 to access the next procedure.

Procedure 2: Set the maximum weight, default as 8000.
Procedure 3: Set the calibrated weight, default as 5000.

Procedure 2 and 3 relies on touch key 1 and 2 to conduct numerical adjustment. Through black small triangle on the right, we can be informed thet current digit is one, with the first black triangle representing thousand and the second triangle representing hundred, and so on. Touch Key 3 changes the position in black triangle, while Touch Key 4 accesses the next procedure.

Procedure 4: Zero Calibration. Under this moment, MCU will automatically catch ADC value to determine its stability. After stabilization, the next step will be accessed. If stabilization fails to show up, user can gain access to the next procedure by clicking Touch Key 4 to force value catching.

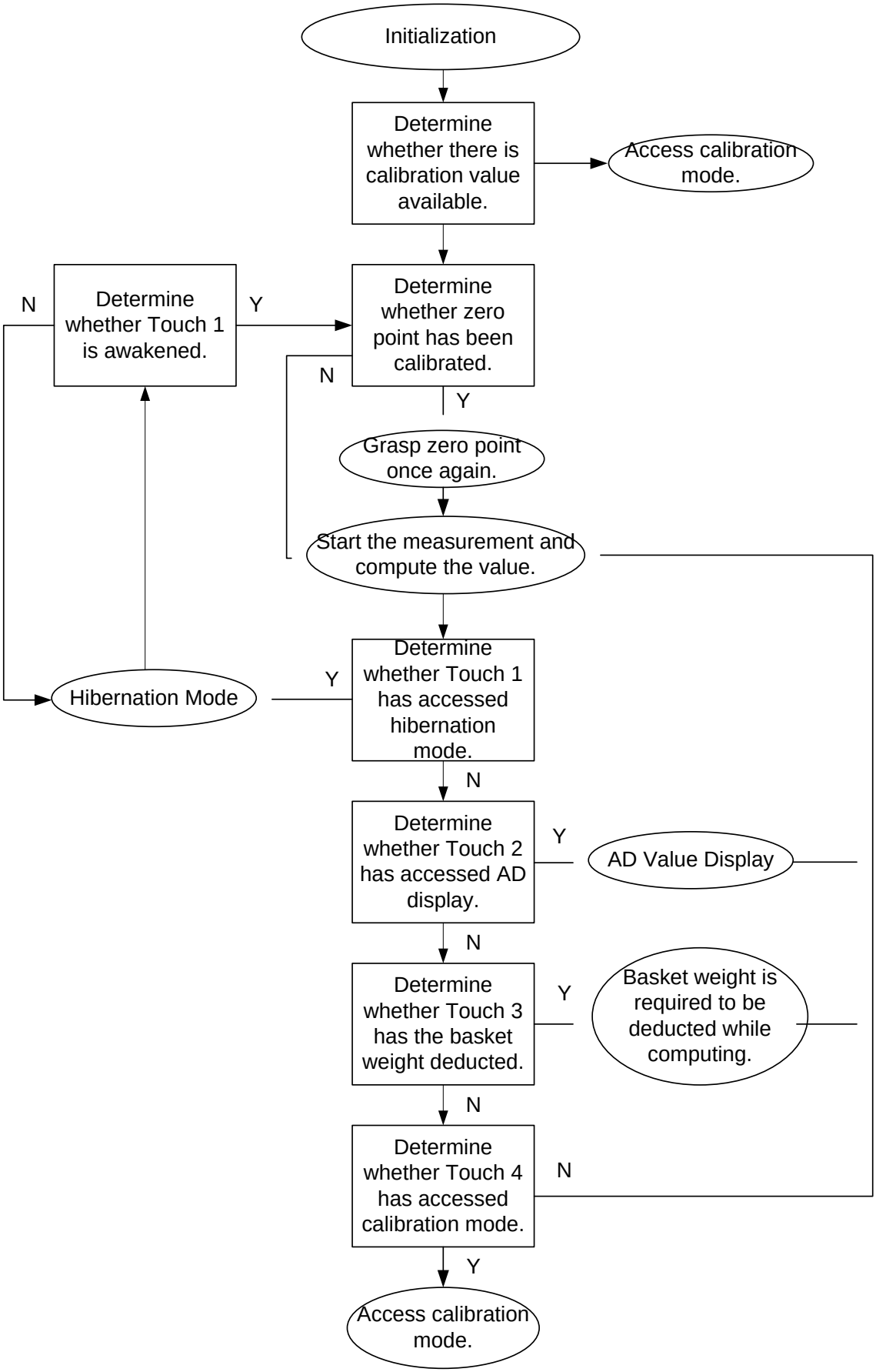
Procedure 5: Correction Matter Calibration. Put a correction matter heavier than 50g onto the electronic scale to facilitate ADC display. Determined to be stable by MCU, the next procedure will be automatically accessed. If stabilization fails to show up, user can gain access to the next procedure by clicking Touch Key 4 to force value catching.

Procedure 6: Four numbers will be displayed, with the first number representing precision, the second number representing decimal point display digit, and the third and fourth number representing bit numbers adopted by ADC under procedure one. Operation method for one and two digit is the same as maximum weight setting. Touch Key 1 and Key 2 facilitates number adjustment, Touch Key 3 allows position transmission in black triangle, while Touch Key 4 enables accession to the next calibration procedure.

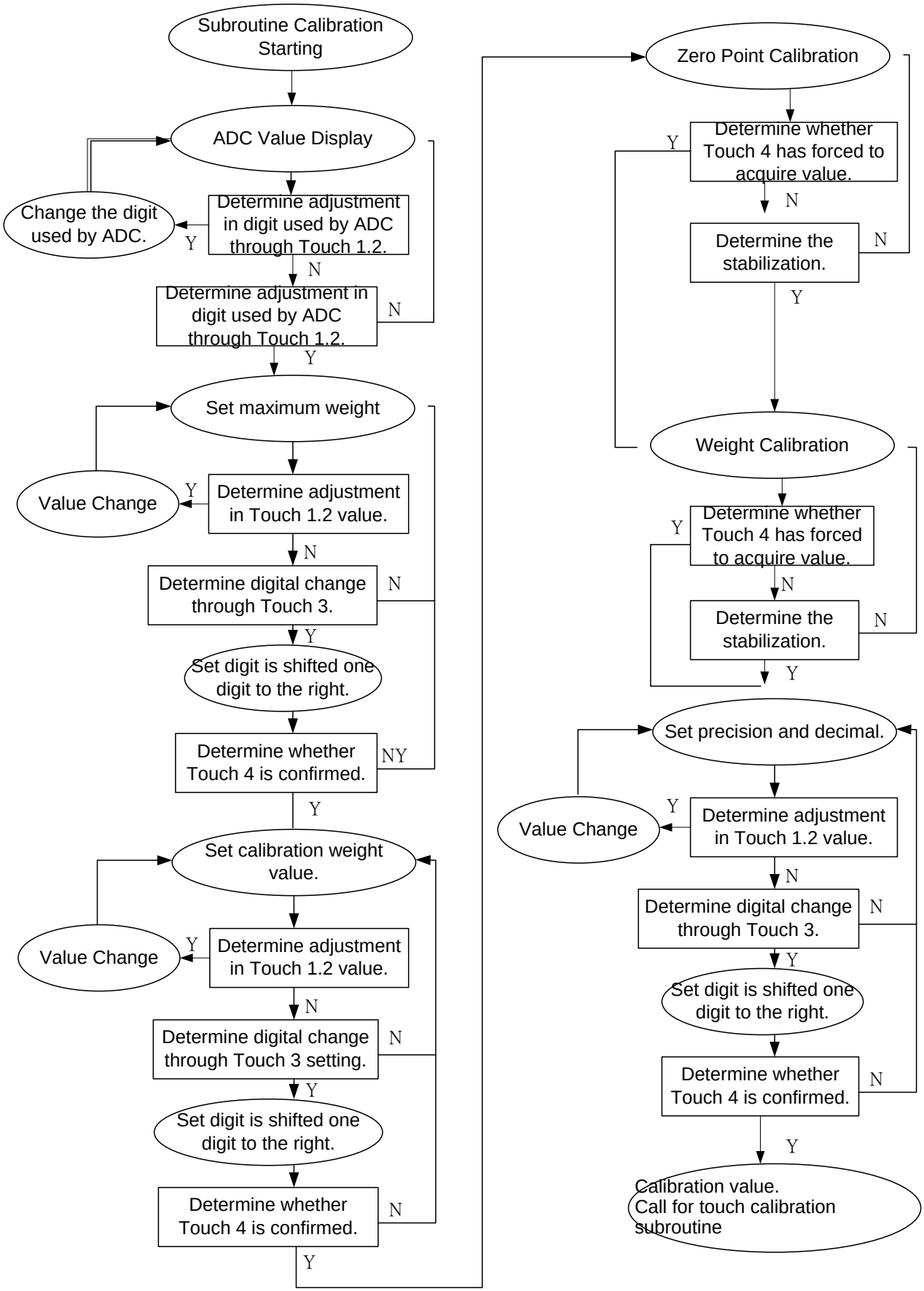
Procedure 7: Initial calibration in procedure 7 is touch key calibration. Untouched value will be caught at first, with LCD display altering between 888888 and 000000. Please don't click touch key under this moment. The next procedure will be automatically accessed upon completion.

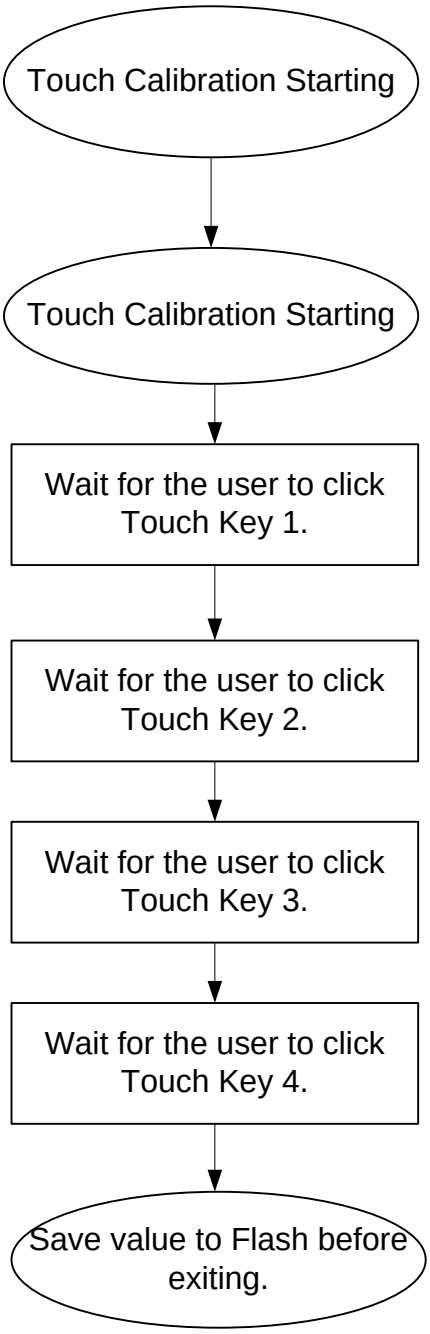
Procedure 8: Touch key threshold catching. 99XXXX will be displayed on the LCD screen. If XXXX takes form in numbers, please click the corresponding touch keys. After the four touch keys has been touched, meter will automatically return to measurement mode from calibration mode.

4.2 Main Program Procedure



4.3 Subroutine Calibration




Touch Calibration Starting
Touch Calibration Starting
Wait for the user to click Touch Key 1.
Wait for the user to click Touch Key 2.
Wait for the user to click Touch Key 3.
Wait for the user to click Touch Key 4.
Save value to Flash before exiting.

5. Technical Specification

(1) Power Supply Connection Point: 3.0V
(2) Power Consumption: Working mode consumes a total current of 7.51mA.
Chip consumes a total current of 1.76mA.
Load Cel consumes current 5.75mA (with internal resistance being 400Ω).
(3) Applicable Range: Respective Scales
(4) Internal and external weight ratio: 8:1
(5) Resolution: 1/8g
(6) Accuracy: 1g
(7) Reaction Time: 0.1s
(8) Working Temperature: -40℃ ~ +85℃
(9) Storage Temperature: -55℃ ~ +125℃
(10) Relative Humidity: <95%(Under 20±5℃ Condition）

6. Measurement Result

Weight	measured weight
1	1
2	2
3	3
4	4
5	5
10	10
100	100
1000	1000
2000	2000
4000	4000
6000	6000
8000	8000

Unit: g

7. Result Conclusion

Measurement based on HY16F188,combined with internal high precision, multi-channel input, and speedy ADC.In addition, HY16F188 is suitable for developing and applying in both kitchen scale and price computing scales.

8. Attached Document


HY16F011V03.zip

9. Referential Literature

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

10. 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	2013/11/26
V2.0	ALL	Amend into Touch Key	2013/12/12
V3.0	ALL	Procedure Amendment	2014/12/18

Appendix: Example Program

```
/* 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 "DrvTimer.h"
#include "DrvADC.h"
#include "DrvPMU.h"
#include "DrvCMP.h"
#include "HY2613.h"
#include "my define.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];
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES
*/
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Ini_Touch(void);
void Initial_ALL(void);
void InitalADC(void);
void scalse_cn(void);
```

```
void AD01(void);
int ScanKey4(unsigned int TCH);
void touch_ct(void);
void g1(void);

void LCD_DATA_DISPLAY(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY1(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY2(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY3(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY4(unsigned int LcdBuffer);

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
int main(void)
{
        Ini_Touch();
        InitalADC();
        Initial_ALL();
        InitalI2C();
        Ini_Display();
        ClearLCDframe();

        MCUSTATUSbits._byte = 0;
        pass=ReadWord(0x801c);                  // Determining whether the
correction value contained within Flash
        if(pass==0x11100)
        {
                decimal=ReadWord(0x8004);    //Yes，read all the correction
                ADnum=ReadWord(0x8008);
                resolution=ReadWord(0x800c);
                ADzero=ReadWord(0x8010);
                ADcn=ReadWord(0x8014);
                cn=ReadWord(0x8018);
                gain=ReadWord(0x8030);
                threh0=ReadWord(0x802c);
```

```
                threh1=ReadWord(0x8020);
                threh2=ReadWord(0x8024);
                threh3=ReadWord(0x8028);
        }
        else
        {
                touch_ct();
                //threh0=238;
                //threh1=153;
                //threh2=143;
                //threh3=111;
                scalse_cn();
        }

  DisplayHYcon();
  while(1)
    {

        pmu_00=0xff00ff00;                  // The comparator is off (save power)
        cmp_00=0xff00;
        cmp_04=0xff00ff00;
        asm("NOP");
        clk_08=0xff00ff0d;                  // Switch off High-speed oscillator，use
low speed (save power)
        Delay(0x100);
        clk_00=0xff00ff00;
        sys_04=0xff10;
        asm("standby 0");
        asm("NOP");
        DrvADC_Disable();                           // Switch off ADC function
(save power)

        if(t==1)                            //(t=1Representation TMA has entered the
broken)
        {
                t=0;                            //Awakening CPU
                pmu_00=0x03031b1b;  // Open comparator function (Open touch
function)
                cmp_00=0x13c3c;
```

```
                cmp_04=0x101d3d3;

                clk_00=0xff01;          // Open high-speed
                clk_08=0xff4c;
                TCH=0;ScanKey4(TCH);          //scan touch1 Is pressed，touch1 Is
pressed LED
                if(LED>=1)
                {
                        DrvADC_Enable();      //Open ADC function
                        DisplayHYcon();;      //wait ADC Stable display HYCON
                        //Delay(0xf0000);
                        LED=0;      // every time to start，It will be detection of a zero

                        AD01();
                        g=average-ADzero;
                        if(g<70)      //ADD value and with last time zero difference
within 70, will be automatic update to zero
                        {
                                c=10;
                                while(c)
                                {
                                        AD01();
                                        ADzero=average;
                                        Delay(0xf000);
                                        AD01();
                                        g=ADzero-average;
                                        if(g==0)
                                        {
                                                next++;
                                        }
                                        else {next=0;}
                                        if(next>3) c=0;
                                }
                                ADzero=average;
                        }

                        ClearLCDframe();
                        while(1)
                        {
```

```
                    g1();   //call is lightning, Calculation subroutine
                    if(LED>=10)
        //tocuh1 (hold press for long-time when LED all to 15，Leaving temperature display，Entering the power-save status again
                    {
                        LED=0;
                        ClearLCDframe();    //RemoveLCD
                        Delay(0xFF00);
                        //LCD_DATA_DISPLAY(ADzero);
                        //Delay(0xf1000);
                        goto bb;        //Switch OFF LCD entering to power save mode
                    }
                    TCH=1;ScanKey4(TCH);          //scan touch key2
                    if(LED<=-10)
                    {
                        LED=0;
                        if(bb==0) bb=1;     //Display AD value Flag=1
                        else bb=0;      // Press again to cancel
                    }
                    g1();                   // call is lightning, Calculation subroutine
                    TCH=3;ScanKey4(TCH);//scan touch key4
                    if(c<=-10)
                    {
                        scalse_cn();    // call correction subroutine
                    }
                    g1();                   // call is lightning, Calculation subroutine
                    TCH=2;ScanKey4(TCH);                    // scan touch key3
                    if(enter>=10)
                    {
                        enter=0;
                        if(tare==0) tare=1;
                        else tare=0;
                        tareg=g;
                    }
                    Delay(0x2000);
```

```
                                        TCH=0;ScanKey4(TCH);// scan touch key1
                                }
                        }
                }
                t=0;
   bb:      ClearLCDframe();                    //Remove LCD display
     }



   while(1);
   return 0;
}
void g1(void)
{
        unsigned long long buu;
//Calculate register
        buu=0;
        buu=buu-1;
          AD01();                                   //call ADC Subroutine

          if(bb==1) LCD_DATA_DISPLAY(average);      // AD value display

          else
          {
                  switch(decimal)       // Analyzing decimal place and give the
corresponding multiple values             {
                          case 4:{dou=100;}break;
                          case 3:{dou=10;}break;
                          case 2:{dou=1;}break;
                          case 5:{dou=1000;}break;
                          case 6:{dou=10000;}break;
                          default: break;
                  }
                  if(average<ADzero)                // Weight is negative
                  {
                          gg=ADzero-average;        //Autozero determine
                          if(gg<=2)
```

```
                {
                        autozero=1;
                        g=0;
                }
                else
                {
                        autozero=0;
                        nn=1;                           //Symbol as Flag 1
                        buu=10*cn/resolution;
                        buu=buu*(ADzero-average);    // Weight calculation
                        buu=buu/dou*100;
                        buu=buu/gain;
                        gg=buu%10;
                        if(gg>5) buu+=10;
                        g=buu/10;
                        g=g*resolution;
                }
        }
        else
        {
                gg=average-ADzero;          // Weight is positive
                if(gg<=2)                     //Autozero determine
                {
                        autozero=1;
                        g=0;
                }
                else
                {
                        autozero=0;
                        nn=0;                   // Symbol as Flag 0
                        buu=10*cn/resolution;
                        buu=buu*(average-ADzero);   // Weight calculation
                        buu=buu/dou*100;
                        buu=buu/gain;
                        gg=buu%10;
                        if(gg>5) buu+=10;
                        g=buu/10;
                        g=g*resolution;
                }
```

```
                    }
                    if(tare==1)
                    {
                            g=g-tareg;
                            if(g<0)
                            {
                                    g=g*-1;
                                    nn=1;
                            }
                            else nn=0;
                    }
                    if(g==0) nn=0;
                    LCD_DATA_DISPLAY(g);

            }
}
void scalse_cn(void)                    // Temperature correction subroutine
{
        DrvADC_Enable();                //Open ADC
        nn=0;
        autozero=0;
        tare=0;
        intest=1;
        ClearLCDframe();
        LCD_DATA_DISPLAY(555555);Delay(0x10000);
        LCD_DATA_DISPLAY(444444);Delay(0x10000);
        LCD_DATA_DISPLAY(333333);Delay(0x10000);
        LCD_DATA_DISPLAY(222222);Delay(0x10000);
        LCD_DATA_DISPLAY(111111);Delay(0x10000);
        ADnum=6;
        c=2;
        LED=ADnum;
        int o;
        while(c)
        {
                AD01();
                LCD_DATA_DISPLAY(average); //Display current ADC value
                TCH=3;ScanKey4(TCH);        //scan touch key1.2.4
                TCH=0;ScanKey4(TCH);
```

```
                TCH=1;ScanKey4(TCH);
                o=ADnum-LED;
                if(o==0) asm("NOP");
                else
                {
                        ADnum=LED;
                        if(LED>16) LED=0;
                        ADnumber=24-ADnum;  // The digits to the right side to use bit
ADC number
                        LCD_DATA_DISPLAY(ADnumber);          //display bit number
                        Delay(0x1f700);
                }
                Delay(0x2000);
        }
        ADnumber=24-ADnum;
        ClearLCDframe();
        Delay(0x1ffff);
        unsigned char lcd1,lcd2,lcd3,lcd4;
        nu=1;
        lcd1=8;
        lcd2=lcd3=lcd4=0;
        c=2;
        first=1;
        while(c)
        {
                 for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}   //scan all touch key
                 LCD_DATA_DISPLAY1(lcd1);
                 LCD_DATA_DISPLAY2(lcd2);
                 LCD_DATA_DISPLAY3(lcd3);
                 LCD_DATA_DISPLAY4(lcd4);
                 switch(nu)
                 {
                          case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;// change the value +1
                          case
2:{if(first==1){first=0;LED=lcd2;}if(LED>10)LED=0;lcd2=LED;}break;// change the value -1
                          case
3:{if(first==1){first=0;LED=lcd3;}if(LED>10)LED=0;lcd3=LED;}break;// change the value
```

```
                    case
4:{if(first==1){first=0;LED=lcd4;}if(LED>10)LED=0;lcd4=LED;}break;  //confirm
                }
                if(enter>=1)
                {
                    enter=0;
                    nu++;                    // change the value
                    first=1;
                    if(nu>=5) nu=1;
                }
                Delay(0x7f00);
        }
        max=(lcd1*1000)+(lcd2*100)+(lcd3*10)+lcd4;    // total conversion，Store in
maximum weighing register        ClearLCDframe();
        Delay(0x1f000);
        lcd1=5;
        lcd2=lcd3=lcd4=0;
        c=2;
        nu=1;
        first=1;
        while(c)
        {
            for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}
            LCD_DATA_DISPLAY1(lcd1);
            LCD_DATA_DISPLAY2(lcd2);
            LCD_DATA_DISPLAY3(lcd3);
            LCD_DATA_DISPLAY4(lcd4);
            switch(nu)
            {
                    case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;// change the value +1
                    case
2:{if(first==1){first=0;LED=lcd2;}if(LED>10)LED=0;lcd2=LED;}break;// change the value -1
                    case
3:{if(first==1){first=0;LED=lcd3;}if(LED>10)LED=0;lcd3=LED;}break;// change the value
                    case
4:{if(first==1){first=0;LED=lcd4;}if(LED>10)LED=0;lcd4=LED;}break;  // confirm
                }
                if(enter>=1)
```

```
                {
                        first=1;
                        enter=0;
                        nu++;                   // change the value
                        if(nu>=5) nu=1;
                }
                Delay(0xff00);
        }
        cn=(lcd1*1000)+(lcd2*100)+(lcd3*10)+lcd4;       // Plus total operation，Store in
correction      ClearLCDframe();
        Delay(0x1f000);
        next=0;
        c=2;
        int z;
        while(c)
        {
                TCH=3;ScanKey4(TCH);            // touch key4
                AD01();
                ADzero=average;
                LCD_DATA_DISPLAY(ADzero);
                Delay(0xf000);
                AD01();
                z=ADzero-average;
                if(z==0)
                {
                        next++;
                }
                else {next=0;}
                LCD_DATA_DISPLAY(ADzero);
                if(next>3) c=0;
        }
        next=0;
        c=2;
        ClearLCDframe();
        Delay(0x1f000);
        while(c)
        {
                AD01();
                if(average>(ADzero*115/100))
```

```
                {
                        TCH=3;ScanKey4(TCH);
                        AD01();
                        ADcn=average;
                        LCD_DATA_DISPLAY(ADcn);
                        Delay(0xf000);
                        AD01();
                        z=ADcn-average;
                        if(z==0)
                        {
                                next++;
                        }
                        else {next=0;}
                        LCD_DATA_DISPLAY(ADcn);
                        if(next>3) c=0;
                }
        }
        ClearLCDframe();
        Delay(0x1f000);
        lcd1=1;
        lcd2=3;
        lcd3=ADnumber/10;
        lcd4=ADnumber%10;
        nu=1;
        c=2;
        first=1;
        while(c)
        {
                for(TCH=0;TCH<4;TCH++){ScanKey4(TCH);}
                LCD_DATA_DISPLAY1(lcd1);
                LCD_DATA_DISPLAY2(lcd2);
                LCD_DATA_DISPLAY3(lcd3);
                LCD_DATA_DISPLAY4(lcd4);
                switch(nu)
                {
                        case
1:{if(first==1){first=0;LED=lcd1;}if(LED>10)LED=0;lcd1=LED;}break;
                        case
2:{if(first==1){first=0;LED=lcd2;}if(LED>=6)LED=0;lcd2=LED;}break;
```

```
                }
                if(enter>=1)
                {
                        enter=0;
                        nu++;
                        first=1;
                        if(nu>=3) nu=1;
                }
                Delay(0xff00);
        }
        resolution=lcd1;
        decimal=7-lcd2;
        gain=ADcn-ADzero;
        DrvADC_DisableInt();                                    //switch Off ADC
        DrvFlash_Burn_Word(0x8004,0x600,decimal);       //The correction burn in Flash
        DrvFlash_Burn_Word(0x8008,0x600,ADnum);
        DrvFlash_Burn_Word(0x800c,0x600,resolution);
        DrvFlash_Burn_Word(0x8010,0x600,ADzero);
        DrvFlash_Burn_Word(0x8014,0x600,ADcn);
        DrvFlash_Burn_Word(0x8018,0x600,cn);
        DrvFlash_Burn_Word(0x8030,0x600,gain);
        DrvADC_EnableInt();                     //Open ADC
        ClearLCDframe();
        Delay(0xffff);
        touch_ct();

}
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* 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)
{
            DrvTIMER_ClearIntFlag(E_TMA);           //Clear TMA interrupt flag
            t=1;
            asm volatile("sethi $r0, 0xc0000");         // Prevent interruption, Unexpected
shutdown
            asm volatile("ori    $r0, $r0, 0x003f");
```

```
        asm volatile("mtsr $r0, $INT_MASK");
        asm volatile("movi $r0, 0x70009");
        asm volatile("mtsr $r0, $PSW");
}

/*---------------------------------------------------------------------------*/
/* HW2 ADC Interrupt Subroutines                                             */
/*---------------------------------------------------------------------------*/
void HW2_ISR(void)
{
        DrvADC_ClearIntFlag();                  //Remove AD, Interrupt flag
        ADCData=adc_08;
        ADCData=ADCData>>8;
        ok=1;
        asm volatile("sethi $r0, 0xc0000");     // Prevent interruption, Unexpected
shutdown
        asm volatile("ori    $r0, $r0, 0x003f");
        asm volatile("mtsr $r0, $INT_MASK");
        asm volatile("movi $r0, 0x70009");
        asm volatile("mtsr $r0, $PSW");
}

/*---------------------------------------------------------------------------*/
/* CMP/OPA Interrupt Service Routines                                        */
/*---------------------------------------------------------------------------*/
void HW3_ISR(void)
{

}

/*---------------------------------------------------------------------------*/
/* PT1 Interrupt Service Routines                                            */
/*---------------------------------------------------------------------------*/
void HW4_ISR(void)
{

}

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

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

}

/*-------------------------------------------------------------------------*/
/* Software Delay Subroutines                                              */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
   for(;num>0;num--)
      asm("NOP");
}
void AD01(void)
          //ADCSubroutine
{
          unsigned char t;
        while(ok)                          //ADC after interruption OK=1
          {
                    ok=0;
                    ADtemp10=(ADCData>>ADnum);

                    value_buf[8]=value_buf[7];          // Average sliding filtering
                    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;
          }
```

```
}
int ScanKey4(unsigned int TCH)                    //touch keySubroutine
{
   CL=TCH;pio_00=0x0100;                          // Capacitor discharge
   DrvTMB_ClearTMB();
         //TMB 計時歸零
   DrvCMP_EnableNonOverlap(CL);                          // Select charging channel
   cmpresult=DrvCMP_ReadData();                          // Read the output comparator
   while(cmpresult==1){cmpresult=DrvCMP_ReadData();}     //Confirm output Reverse
   tmrbcnt=DrvTMB_CounterRead();//TMB              // Determine charging time is less than
the threshold
   switch(CL)
   {
      case 0 : if(tmrbcnt<=threh0){LED++;}break;
      case 1 : if(tmrbcnt<=threh1){LED--;}break;
      case 2 : if(tmrbcnt<=threh2){enter++;}break;
      case 3 : if(tmrbcnt<=threh3){c--;}break;
      default: break;
   }
   pio_00=0x0101;                          // Capacitor charging
   pio_04=0x0100;
   return 0;
}
void touch_ct(void)
         //touch key Correction subroutine
{
         int z,y;

                  LCD_DATA_DISPLAY(888888);
                  Delay(0x8000);
                  unth0=0;                  //Catch touch1 untouch value (8 averaging)
                  for(z=0;z<8;z++)
                  {
                           TCH=0;ScanKey4(TCH);
                           unth0+=tmrbcnt;
                           Delay(0xFFF);
                  }
                  unth0=unth0>>3;
                  LCD_DATA_DISPLAY(000000);
```

```
unth1=0;                        // Catch touch2 untouch value (8 averaging)
for(z=0;z<8;z++)
{
        TCH=1;ScanKey4(TCH);
        unth1+=tmrbcnt;
        Delay(0xFFF);
}
unth1=unth1>>3;
LCD_DATA_DISPLAY(888888);
unth2=0;                        // Catch touch3 untouch value (8 averaging)
for(z=0;z<8;z++)
{
        TCH=2;ScanKey4(TCH);
        unth2+=tmrbcnt;
        Delay(0xFFF);
}
unth2=unth2>>3;
LCD_DATA_DISPLAY(000000);
unth3=0;                        // Catch touch4 untouch value (8 averaging)
for(z=0;z<8;z++)
{
        TCH=3;ScanKey4(TCH);
        unth3+=tmrbcnt;
        Delay(0xfff);
}
unth3=unth3>>3;
LCD_DATA_DISPLAY(991111);
th0=0;                          // Catch touch1 touch value (8 averaging)
y=8;
while(y)
{
        TCH=0;ScanKey4(TCH);
        if(tmrbcnt<435)
       {
                y--;
                th0+=tmrbcnt;
        }
}
th0=th0>>3;
```

```
LCD_DATA_DISPLAY(992222);
th1=0;                          // Catch touch2 touch value (8 averaging)
y=8;
while(y)
{
        TCH=1;ScanKey4(TCH);
        if(tmrbcnt<220)
        {
                y--;
                th1+=tmrbcnt;
        }
}
th1=th1>>3;
LCD_DATA_DISPLAY(993333);
th2=0;          // Catch touch3 touch value (8 averaging)
y=8;
while(y)
{
        TCH=2;ScanKey4(TCH);
        if(tmrbcnt<180)
        {
                y--;
                th2+=tmrbcnt;
        }
}
th2=th2>>3;
LCD_DATA_DISPLAY(994444);
th3=0;          // Catch touch4 touch value (8 averaging)
y=8;
while(y)
{
        TCH=3;ScanKey4(TCH);
        if(tmrbcnt<180)
        {
                y--;
                th3+=tmrbcnt;
        }
}
th3=th3>>3;
```

```
                threh0=unth0-(((unth0-th0)*3)/4);          // Calculate correction value
                threh1=unth1-(((unth1-th1)*3)/4);
                threh2=unth2-(((unth2-th2)*3)/4);
                threh3=unth3-(((unth3-th3)*3)/4);
                LED=0;
                enter=0;
                DrvADC_DisableInt();                    //Switch OFF ADC
                DrvTIMER_DisableInt(E_TMA);  // Switch OFF TMA
        DrvFlash_Burn_Word(0x801c,0x600,0x11100；//Determine whether or not the
corrected numerical burn in Flash
                DrvFlash_Burn_Word(0x802c,0x600,threh0);  //The correction value is
store in Flash
                DrvFlash_Burn_Word(0x8020,0x600,threh1);
                DrvFlash_Burn_Word(0x8024,0x600,threh2);
                DrvFlash_Burn_Word(0x8028,0x600,threh3);
                Delay(0x8000);
                DrvTIMER_EnableInt(E_TMA);  //Open TMA
                DrvADC_EnableInt();     // Open ADC
                intest=0;
}
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(1,3);////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_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 InitalADC(void)
{
  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);//ADC Input
  DrvADC_InputSwitch(OPEN);
  DrvADC_RefInputShort(OPEN);
  DrvADC_Gain(7,3);              //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_Enable();
}
void Initial_ALL(void)
{
            pio_00=0xff00ff00;                                  //confirm do not use Pin
close
            pio_04=0xff00ff00;
            pio_08=0x00;
            pio_10=0xff00ff03;
            pio_14=0xff00ff03;

            autozero=0;                                         // Flag setting
            LED=0;
            t=0;
            TCH=0;
            c=0;
            enter=0;
            intest=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

            DrvCLOCK_SelectIHOSC(0);
            DrvCLOCK_EnableHighOSC(E_INTERNAL,50);          // Select HAO 4MHz
            SYS_EnableGIE(7,0x3F);    // Enable GIE(Global Interrupt)
}
/*---------------------------------------------------------------------------*/
/* End Of File*/
/*---------------------------------------------------------------------------*/
```