HYCON 紘康科技

血糖計應用說明書

HY16F198B

Glucose Meter

Table of Contents

1. 內容簡介 4

2. 原理說明 5

2.1. 血糖原理說明 5

2.2. 試紙說明 5

2.3. 控制晶片 6

3. 系統設計 9

3.1. 硬體說明 9

3.2. 軟體說明 11

4. 操作流程 12

5. 技術規格 15

5.1. 電阻量測數據 15

6. DEMO CODE 及相關檔案 16

7. 參考文獻 27

8. 修訂記錄 28

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

本文將介紹以量測電阻值訊號，相關應用如：一氧化碳(CO)濃度計、二氧化碳(CO2)濃度計、血糖計、糖(甜)度計等電化學信號，此類應用廣泛，本文僅以血糖計作為說明。使用HY16F198B內建高精密Sigma-delta 24 Bit ADC與8 bit DAC電路與R2R OPA運算放大器來實現一個血糖儀應用電路。

2. 原理說明

2.1. 血糖原理說明

正常情況下，身體會將吃進去的澱粉類食物分解及轉變成葡萄糖，做為生命的能源。而胰島素是由胰臟製造的一種荷爾蒙，它可以幫助葡萄糖進入細胞內，提供熱能。糖尿病患因為胰臟製造的胰島素不足或功能不良，無法使葡萄糖充分進入細胞利用，而留在血中使血糖濃度升高。長期的高血糖症狀，有時會引起視網膜病變、腎臟、神經病變（如導致截肢）、心臟和血管（如中風）、高血壓、生理需求功能減弱，嚴重者會導致死亡。糖尿病的發生與遺傳體質相當有關係，其他如肥胖、情緒壓力、藥物、營養生理失調，都會促使糖尿病的發生。

糖尿病已成為全球最主要的慢性病之一，「早期診斷、適當治療」與「患者自我配合」是控制糖尿病的不二法門，除了嚴重者長期注射胰島素外，糖尿病平日的血糖監測亦非常重要，配合用藥、飲食的控制及持續監測，可延緩各種糖尿病併發症的發生以擁有健康的生命。

利用測試試紙做為葡萄糖訊號的擷取來源，以固定偏壓方式激發試紙與血液的電化學反應成為電阻訊號輸出，再經由紘康科技的 32 位元單片機 HY16F198B 量測電阻訊號、運算、數位輸出顯示，如圖 1，以最少的元件達成偏壓式電阻量測方案。

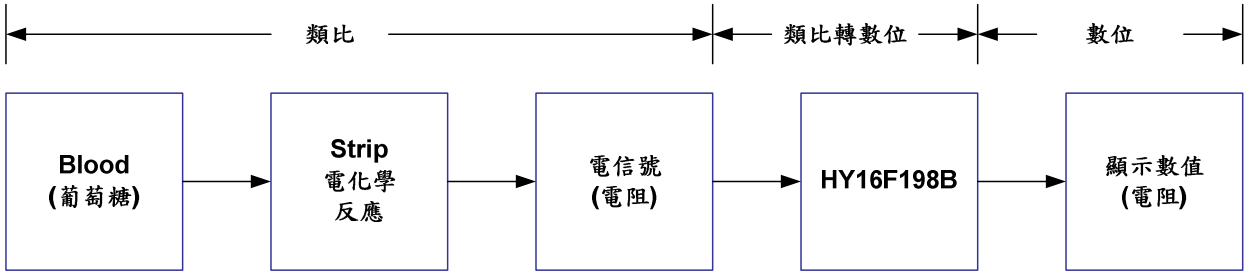


圖 1 類比與數位訊號轉換

2.2. 試紙說明

由於各家試紙組成成分不同，本文以等效電路作為探討的方向，方案開發前，必須先瞭解試紙的等效電路(如圖 2)、電化學反應時間(圖 3)、電流轉換公式…等。

試紙等效電路:

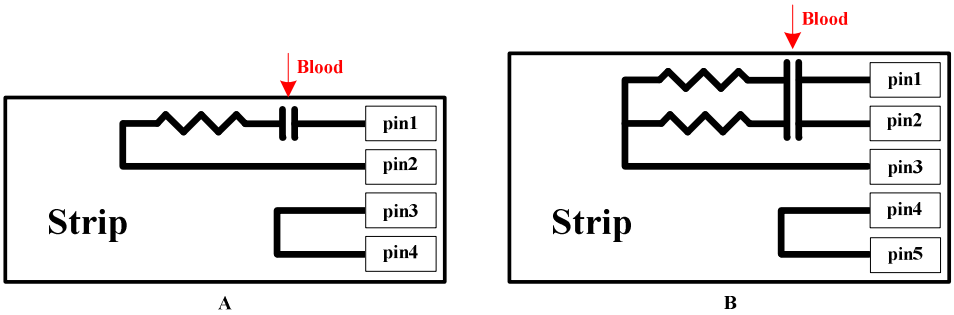


圖 2 試紙等效電路；A：單通道電阻，B：雙通道電阻

電化學反應時間:

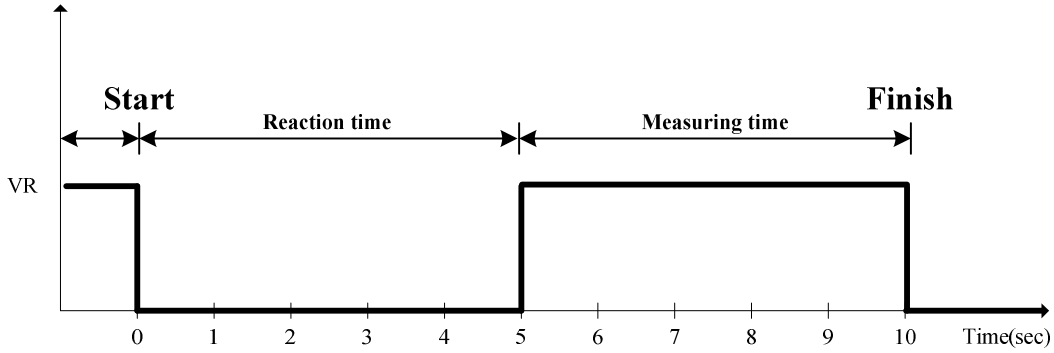


圖 3 電化學反應時間

VR：固定偏壓
Reaction time：試紙反應時間
Measuring time：試紙量測時間

電流轉換公式範例：

$$Glucose = I \times F(code) \times T(t)$$

I：電流(μA)
F(code)：不同試紙會有不同的值
T(t)：溫度係數

2.3. 控制晶片

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

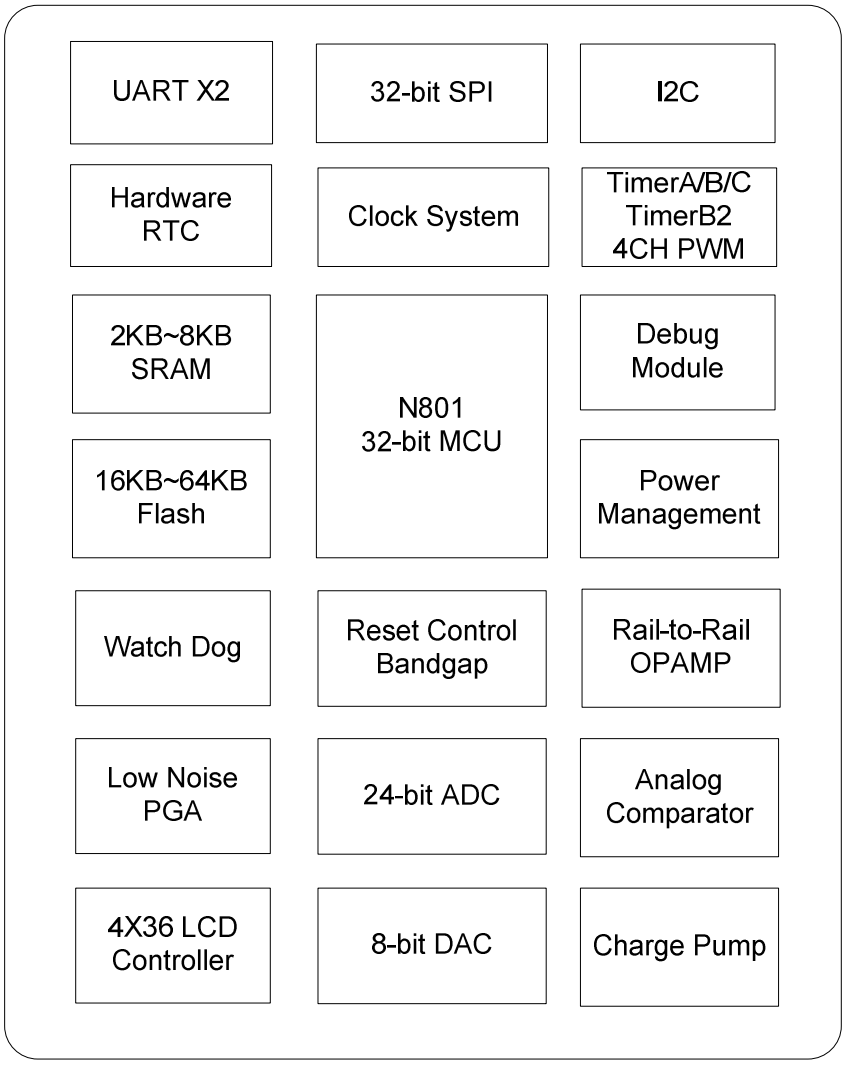


圖 3 紘康 HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)

(1) 採用最新 Andes 32 位元 CPU 核心 N801 處理器。
(2) 電壓操作範圍 2.2~3.6V，以及-40℃~85℃工作溫度範圍。
(3) 支援外部 16MHz 石英震盪器或內部 16MHz 高精度 RC 震盪器，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(Sleep Mode), 5uA(Idle Mode)
(4) 程式記憶體 64KBytes Flash ROM
(5)資料記憶體 8KBytes 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。
(8)超低輸入雜訊運算放大器。
(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 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

HY16F198B 血糖儀硬體連接電路如下:

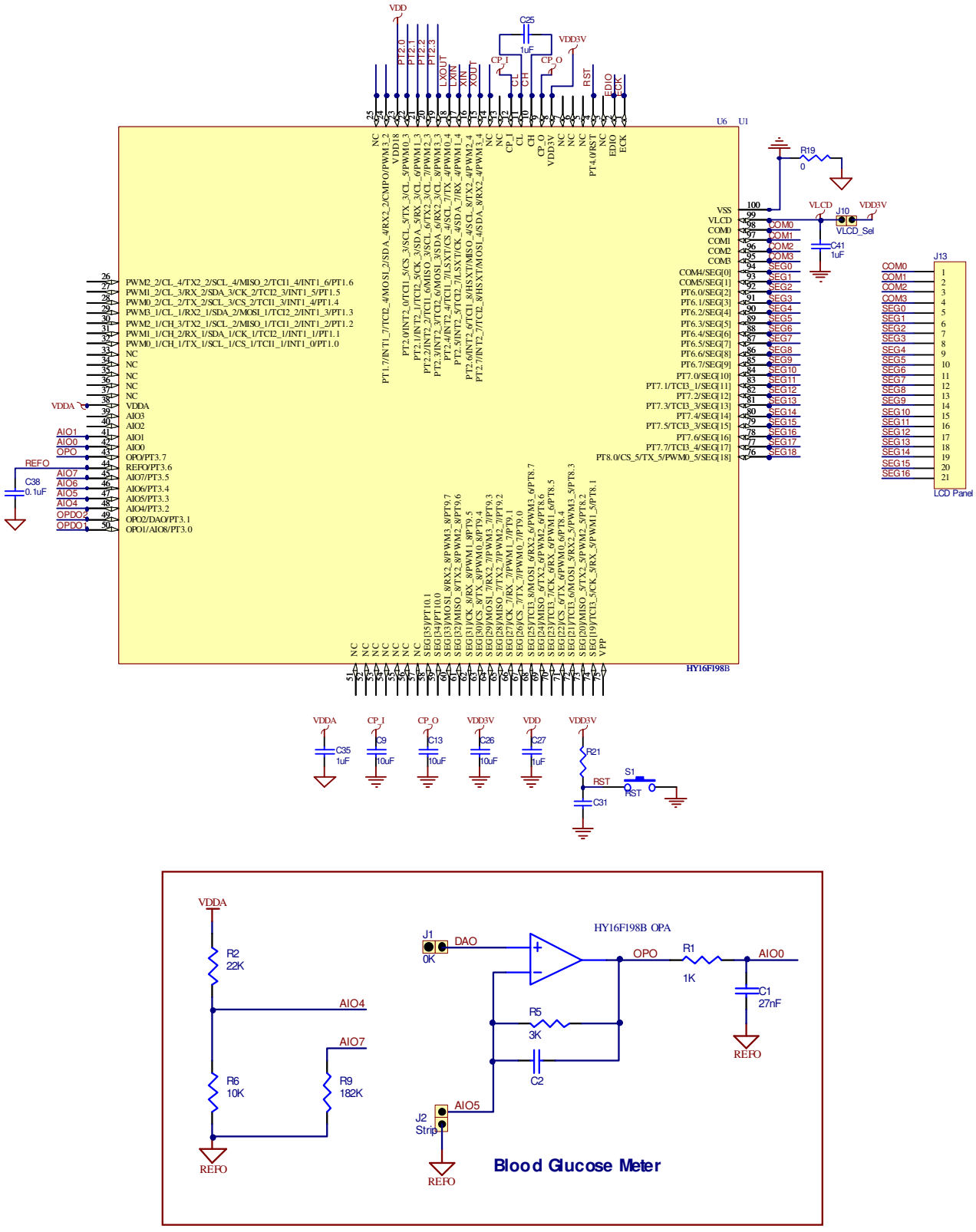


圖4 HY16F198B 血糖計硬體線路連接圖

主要元件介紹 ：

(1)MCU:HY16F198B，(Andes 32-bit MCU Core + HYCON 24-bit ΣΔADC + MCU 64K Flash + 8 bit DAC + R2R OPAMP + LCD Driver)。

(2)LCD Display：負責顯示量測出來的電阻數值或 ADC Count 數值，可由程式控制選擇。

(3)R2=22k & R6=10k 電阻 ： 控制進入 DAC 正端與 DAC 負端的輸入範圍，電阻值的設計影響每一階 DAC 輸出電壓所產生的大小。

(4)R5=3k 電阻: 回授電阻的選擇設計，回授電阻的選擇會決定 Rstrip 電阻可量測範圍。

(5)R1=1k & C1=27nF: 低通濾波電路(預留使用)。

(6)R9=182k：與 HY16F198b 內部的 8 bit DAC 內部阻抗分壓設計使用。

HY16F198B 血糖儀應用方塊圖 ：

舉例：DAC=330mV偏壓計算
理想值：VDDA=2.4V, REFO=1.2V, DAC內阻=180k
AI4= (2.4-1.2) X 10/(10+22)= (1.2X10)/32=0.375V
AI7= (0.375X182)/ (182+180) = (0.4X182)/362=
0.1885V
0.375V與0.1885V分256階=每階0.7285(mV)
DAO=330mV=(Vrefp-Vrefn)*(X/256)+Vrefn
DAO=330mV=(375mV-188.5mV)*(X/256)+188.5mV
當X=194.23 (DAC可輸出330mV偏壓)

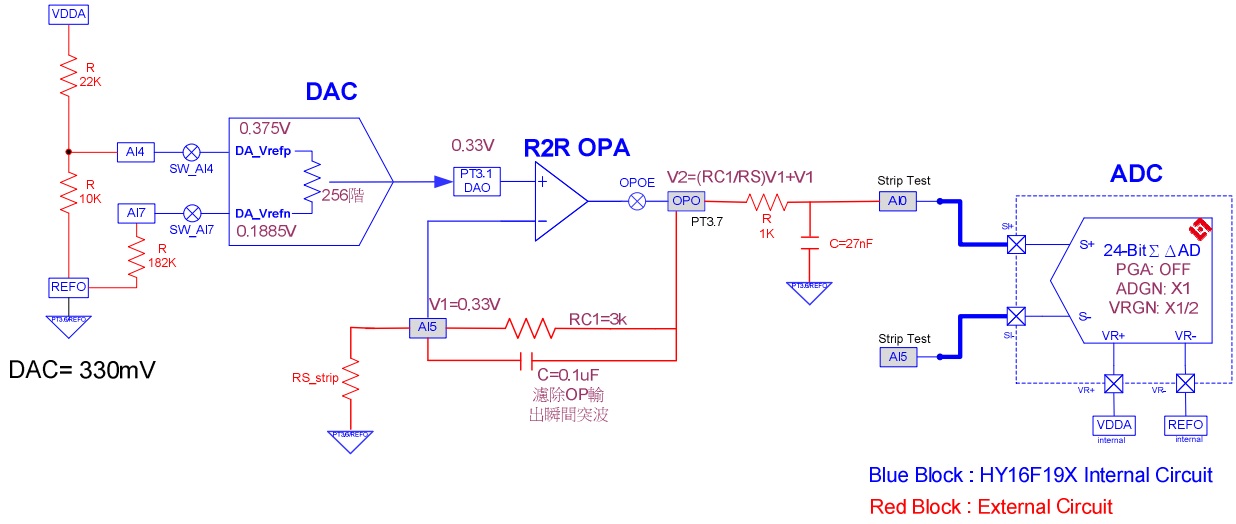
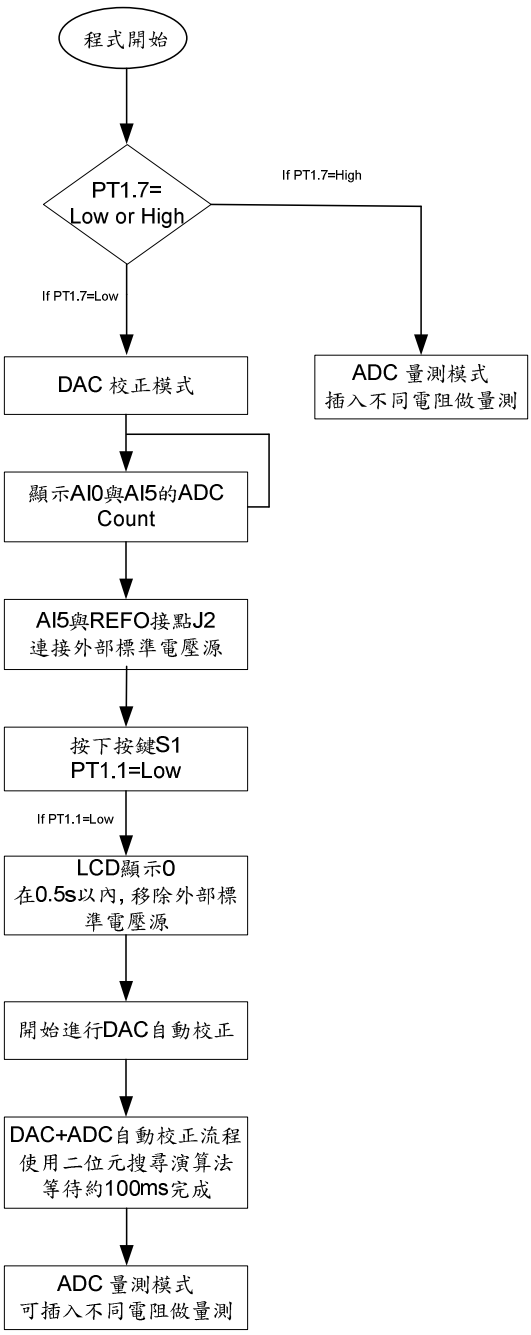


圖5 HY16F198B 血糖儀應用方塊圖

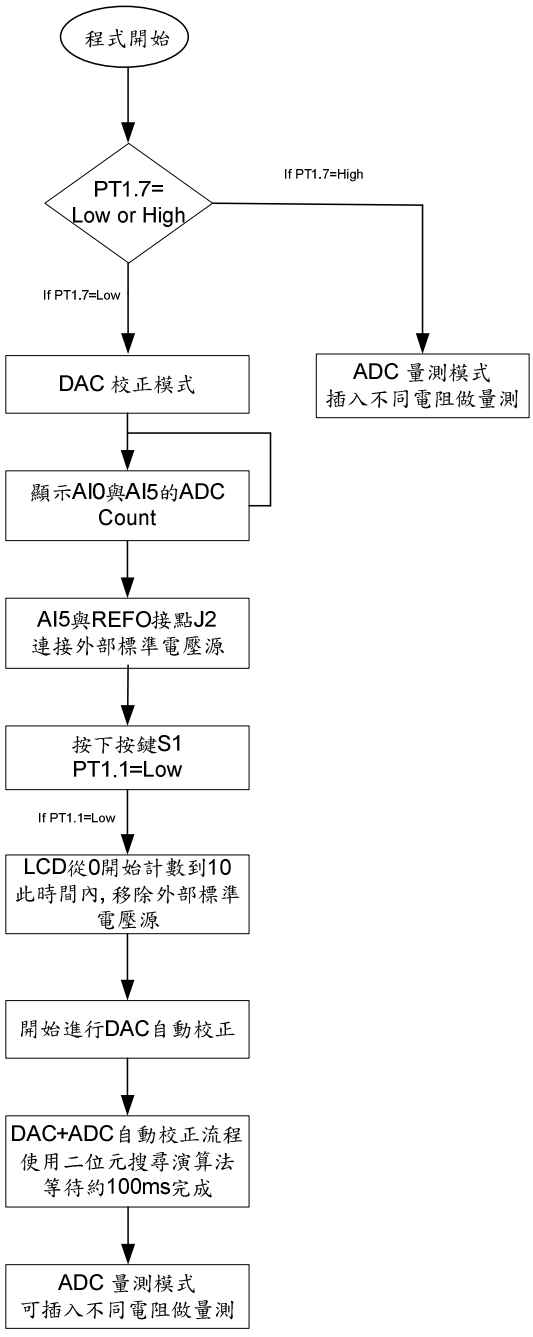
3.2. 軟體說明

程式流程圖

#define DAC_Calibration_fast_mode;



#define DAC_Calibration_teaching_mode;



4. 操作流程

1. 需先準備好血糖儀硬體板子。本文使用 HY16F198B 開發板(A14011 V04)搭配血糖儀上板(A14022 V01)做為開發測試使用，可先準備幾個不同阻值的電阻，在量測模式時候，可把電阻插在血糖儀上板(A14022 V01)的 J2 Jump 上面。血糖儀硬體板子如下圖 ：

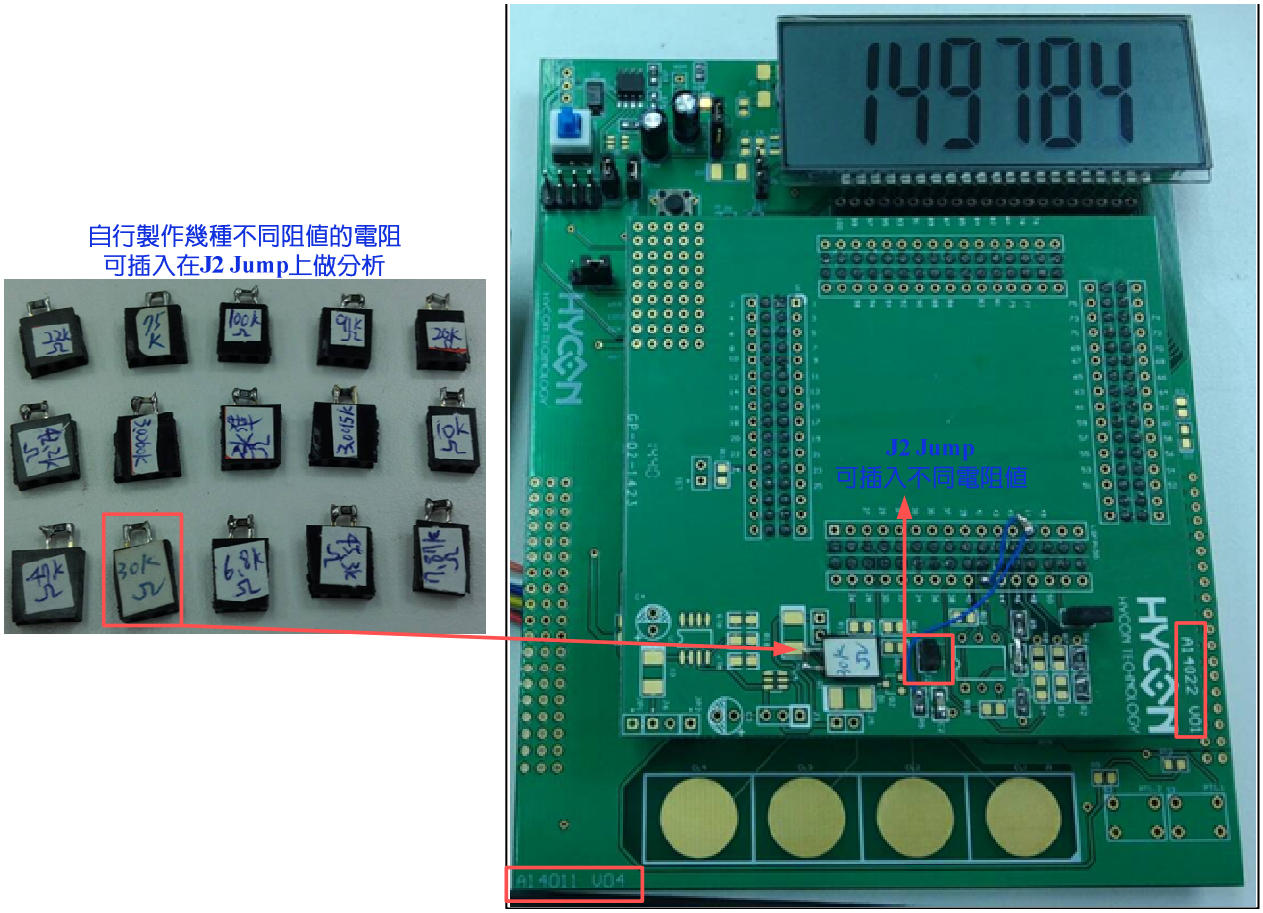


圖6 HY16F198B 血糖儀硬體開發板

2. DAC 校正模式。在上電之前就直接把 PT1.7 拉 Low，上電之後進入 DAC 校正模式，LCD 會顯示 AI5-REFO 的 ADC Count。程式在 DAC 校正模式時候，預設是 DAC 輸出=0/255，輸出電壓約 0.173V 左右，此時可以先嘗試在 J2 Jump 接點外灌目標校正電壓值，LCD 會顯示相對應的 ADC Count 數值。當按下 S1 按鍵的時候，可開始進行 DAC+ADC 自動校正，執行約 100ms 時間以內可完成。

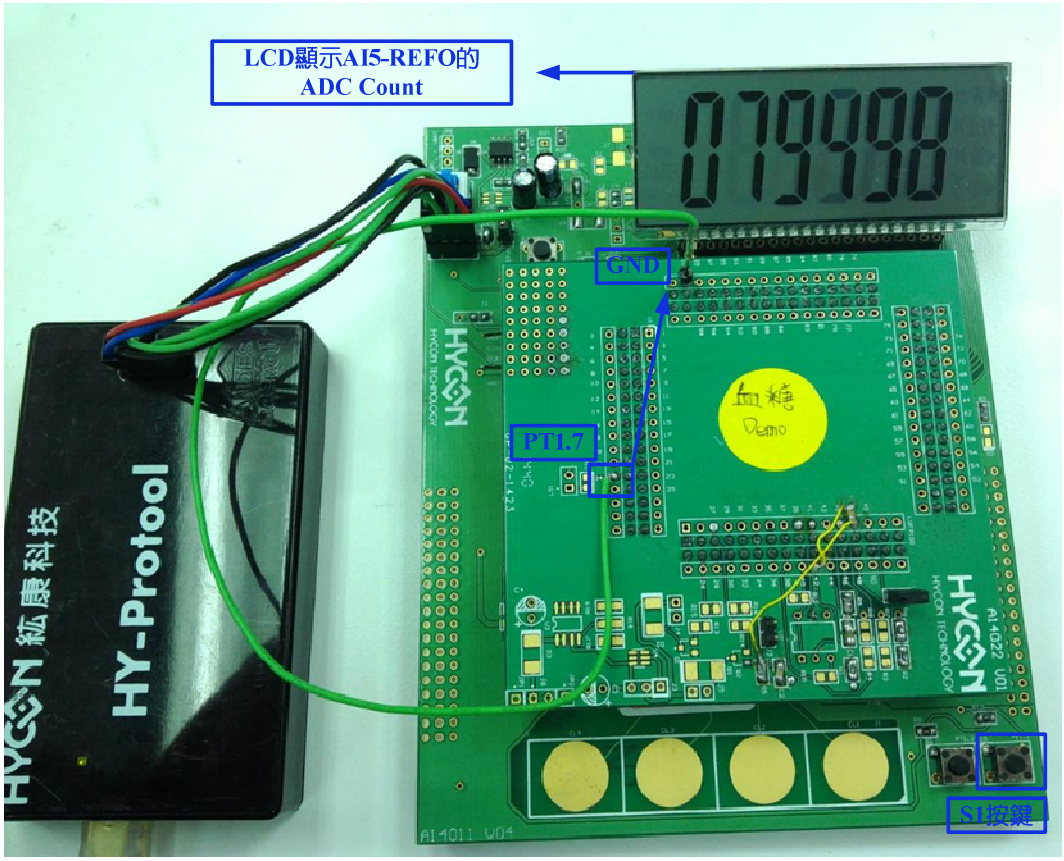


圖 7 HY16F198B 血糖儀硬體開發板(DAC 校正模式)

3. ADC 量測模式。如果上電之前 PT1.7 不拉 Low，可以直接進入電阻量測模式。程式在電阻量測模式時候，預設是 DAC 輸出=228/255，輸出電壓約 0.330V 左右，此時可以直接在 J2 Jump 接點插入不同的電阻值，LCD 會顯示相對應的電阻數值，下圖為插入 3k 電阻時候的 LCD 顯示。

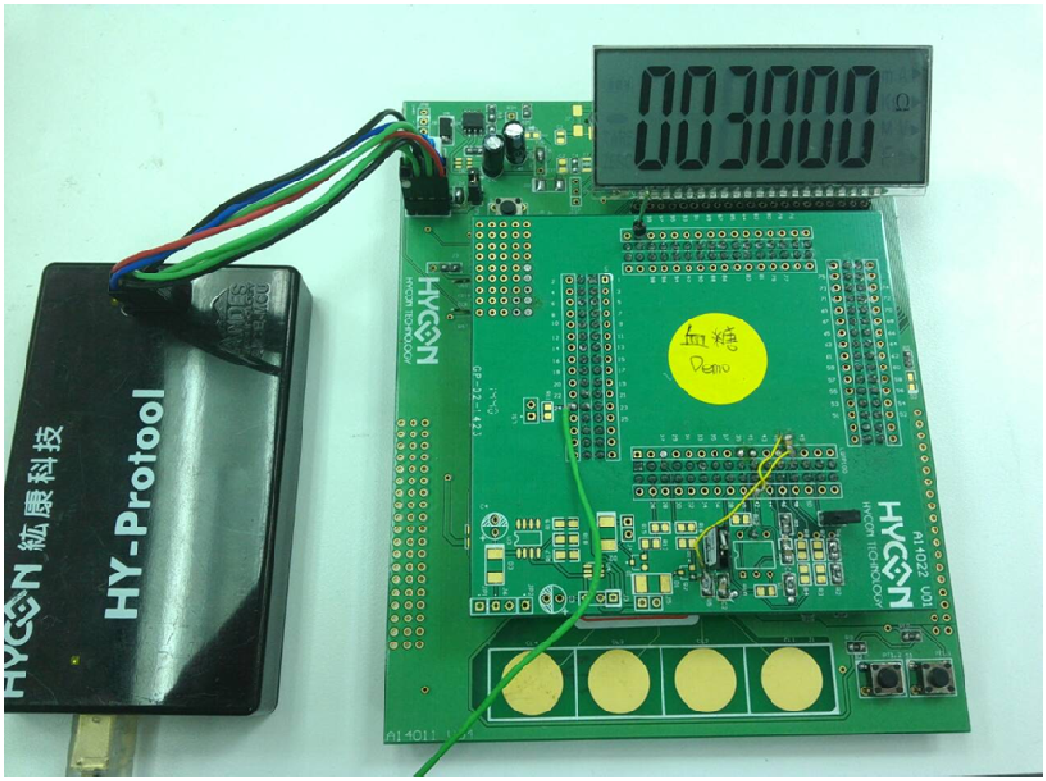


圖 7 HY16F198B 血糖儀硬體開發板(電阻量測模式)

4. 補充 ： 校正電阻表格的建立方式。 在此範例應用程式中，因為是考慮量測電阻時候，DAC 偏壓輸出為 330mV，所以需要先能夠打出目標 330mV DAC 偏壓。當打出 DAC 偏壓為 330mV 之後，可以調整程式設定 ADC 通道為 AI0-AI5。這時候 J2 Jump 插入一個非常準確的 3k 電阻，觀察並記錄下來 AI0-AI5 的 ADC Count 為多少，以此方法建立電阻換算表格。

5. 技術規格

(1) Operation voltage : 2.4~3.6V
(2) Operation current : 1.335mA@HSRC=2MHz(3.0V)
(3) Idle mode current : 5uA(3.0V)
(4) DAC 解析度 : 0.7mV/每階 (DAC 輸出範圍 : Min: 0.1730V, Max :0.3510V)
(5) DAC+ADC 自動量測校正時間 :100ms (條件:OSR=5, ADC Output Rate=326Hz)
(6) Resolution(電阻/誤差值) : 3kΩ/1Ω, 6.8kΩ/2Ω, 30kΩ/15Ω
(7) 電阻量測誤差量% : 100kΩ/0.225%, 30kΩ/0.046% , 3kΩ/0%
(8) 測試範圍 : 3KΩ~100KΩ
Note (6)~(8)測試條件為： 電阻校正點在 3kΩ & ADC Output Rate=20Hz & DAC=330mV.

5.1. 電阻量測數據

SMD 電阻	A	B	(A-B)/A
Rstrip 電標籤值 (理論 KΩ)	接 DMM 電錶量測 (實際 KΩ) 電表 BRYMEN : BM859CF	HY16F 顯示(實際 Ω) 校正點 3k Ω	A 和 B 誤差%
3	3.000	3000	0.000
3	3.001	3001	0.000
3	3.006	3006	0.000
6.8	6.805	6803	0.029
7.87	7.861	7861	0.000
10	10.005	10006	-0.010
20	19.981	19987	-0.030
22	21.998	21990	0.036
30	30.285	30299	-0.046
42.2	42.315	42360	-0.106
45.3	45.485	45523	-0.084
47	47.106	47144	-0.081
75	75.16	75280	-0.160
91	90.82	90946	-0.139
100	100.23	100456	-0.225

6. Demo Code 及相關檔案

```
/*****************************************************************************
 * HY16F198B
 * ---------------------------------------------------------------------------
 * HYCON Technology
 * http://www.hycontek.com/
 * Author : FAE/Robert Wang
 * C Library release V1.0
 * ClucoseMeter Demo Code V0.1
 * Maintenance:2016/06/21
 *****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "DrvADC.h"
#include "DrvClock.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvPMU.h"
#include "ModuleID.h"
#include "Sysinfra.h"
#include "System.h"
#include "Display.h"
#include "my define.h"
#include "DrvREG32.h"
#include "DrvOP.h"
#include "DrvDAC.h"
#include "DrvFlash.h"
/*---------------------------------------------------------------------------*/
/* STRUCTURES                                                                */
/*---------------------------------------------------------------------------*/
volatile typedef union _MCUSTATUS
{
  char  _byte;
  struct
  {
    unsigned b_ADCdone:1;
    unsigned b_TMAdone:1;
    unsigned b_TMBdone:1;
    unsigned b_TMC0done:1;
    unsigned b_TMC1done:1;
    unsigned b_RTCdone:1;
    unsigned b_UART_TxDone:1;
    unsigned b_UART_RxDone:1;
  };
} MCUSTATUS;

typedef union _PTINTSTATUS
{
  char  _byte;
  struct
  {
    unsigned b_PTINT0done:1;
    unsigned b_PTINT1done:1;
    unsigned b_PTINT2done:1;
    unsigned b_PTINT3done:1;
    unsigned b_PTINT4done:1;
    unsigned b_PTINT5done:1;
    unsigned b_PTINT6Done:1;
    unsigned b_PTINT7Done:1;
  };
} PTINTSTATUS;

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
//#define ADC_Count_show_19bit; //to show ADC_Count
#define ADC_ohm_show_19bit;  //to show measure ohm
#define DAC_Calibration_teaching_mode;
//#define DAC_Calibration_fast_mode;

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

```
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
MCUSTATUS  MCUSTATUSbits;
PTINTSTATUS  PT1INTSTATUSbits;

unsigned char i;
unsigned char PT1_7_high;
unsigned char PT1_7_low;
unsigned char DAC_Output;
unsigned char ADfinish;
unsigned char DAC_Calibration_Flag;

int ADC_Array[2];
int ADCData_Chopper;
int DAC_ADCData;
int DAC_Calibration_ADCData;
int ADCData;
int K_ohm_ADCount[4];

float K_ohm[4];
float ADCData_temp;
float ADCData_ohm_show;
float ADC_over_19bit;


/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void InitalADC(void);  //ADC initial for Measure RS_Strip
void InitalADC_DAC_Calibration(void);  //ADC initial for auto DAC Calibration
void Delay(unsigned int num);

/*---------------------------------------------------------------------------*/
/* Note Something                                                            */
/*---------------------------------------------------------------------------*/
//此Code是在RStrip=3k效正, 3k ohm 電組應該是要非常的準
//因為目前此線路分壓設計, DAC最大輸入電壓是0.38V, 當DAC輸出255, ADC Count最大量測是160500左右


/*---------------------------------------------------------------------------*/
/* MAIN  function                                                            */
/*---------------------------------------------------------------------------*/
int main(void)
{

  //ADC chopper temp data
  for(i=0;i<=1;i++)
  {
      ADC_Array[i]=0;
      ADC_Array[i]=0;
  }

  //S1=PT1.1 setting. If press S1 that means to save the DAC calibration voltage.
  DrvGPIO_ClearIntFlag(E_PT1,0x02);       //clear PT1.1 interrupt flag
  DrvGPIO_Open(E_PT1,0x02,E_IO_INPUT);    //set PT1.1 INPUT
  DrvGPIO_Open(E_PT1,0x02,E_IO_PullHigh); //enable PT1.1 pull high
  DrvGPIO_Open(E_PT1,0x02,E_IO_IntEnable);  //PT1.1 interrupt enable
  DrvGPIO_ClkGenerator(E_HS_CK,1);        //Set IO sampling clock input source is HS_CK
  DrvGPIO_IntTrigger(E_PT1,0x02,E_N_Edge);  //PT1.1 interrupt trigger method is negative edge

  //PT1.7 setting. If PT1.7=Low, To do DAC Auto-Calibration
  DrvGPIO_Open(E_PT1,0x80,E_IO_INPUT);    //set PT1.7 INPUT
  DrvGPIO_Open(E_PT1,0x80,E_IO_PullHigh); //enable PT1.7 pull high
  PT1_7_high=0x82; //0x82 : PT1.7=High, PT1.1=High
  PT1_7_low=0x02; //0x02 : PT1.7=Low, PT1.1=High

  //check DAC calibration time
  //DrvGPIO_Open(E_PT2,0x08,E_IO_OUTPUT);  //PT2.3 Output

//if PT1.7=Low
if(pio1_3 == PT1_7_low) //if PT1.7=Low. To to auto DAC Calibration by ADC measure
{
  //Default DAC Output Setting
  DAC_Output=0; //default DAC = 0/255
  DAC_ADCData=0;
  DAC_Calibration_ADCData=0;
  DAC_Calibration_Flag=0;
```

```
  DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,DAC_Output);    //DAC_Vrefp= VDDA, DAC_Vrefn= VSSA, IF DAO=0
  DrvDAC_SetoutputIO(ENABLE);       //DAC output with PT3.1
  DrvDAC_EnableOutput();           //DAC output enable
  DrvDAC_Enable();                //DAC IP enable

  // R2ROPA Setting
  DrvOP_PInput(E_OPP_DAO);
  DrvOP_NInput(E_OPN_AIO5);
  DrvOP_Open();
  DrvOP_OPOoutEnable(); //OPA OPEO=1

  InitalADC_DAC_Calibration();
  DisplayInit();
  ClearLCDframe();

}
//if PT1.7=high
else  //if PT1.7!=Low  //ADC Measure RS_Strip
{
  ADC_over_19bit=262143; //The max ADC Count +262143
  ADfinish=0;
  DAC_Output=228;  //default DAC = 228/255
  DAC_Calibration_Flag=0;

  //DAC Output Setting
  DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,DAC_Output);    //DAC_Vrefp= VDDA, DAC_Vrefn= VSSA, IF DAO=0
  DrvDAC_SetoutputIO(ENABLE);       //DAC output with PT3.1
  DrvDAC_EnableOutput();           //DAC output enable
  DrvDAC_Enable();                //DAC IP enable

  // R2ROPA Setting
  DrvOP_PInput(E_OPP_DAO);
  DrvOP_NInput(E_OPN_AIO5);
  DrvOP_Open();
  DrvOP_OPOoutEnable(); //OPA OPEO=1
}

  SYS_EnableGIE(7,0x1FF);  //Enable GIE(Global Interrupt)
  MCUSTATUSbits._byte = 0;
  PT1INTSTATUSbits._byte = 0;

  while((pio1_3<=PT1_7_low)&&(DAC_Calibration_Flag==0)) //0x02, if PT1.7=Low, PT1.1=High
  {

      if(MCUSTATUSbits.b_ADCdone)
      {
           while(!ADfinish);
           DAC_ADCData=ADCData;
         LCD_DATA_DISPLAY(DAC_ADCData);     //To Do DAC Calibration
           ADfinish=0;
           MCUSTATUSbits.b_ADCdone=0;
      }

       if(PT1INTSTATUSbits.b_PTINT0done)    //0x00, if PT1.7=low, PT1.1=low
       {

#if defined(DAC_Calibration_teaching_mode)
           PT1INTSTATUSbits.b_PTINT0done=0;
           DAC_Calibration_ADCData=DAC_ADCData; //if goal is 330mV, DAC_ADCData=0x24fdf
         LCD_DATA_DISPLAY(0);
         Delay(100000);
         LCD_DATA_DISPLAY(1);
         Delay(100000);
         LCD_DATA_DISPLAY(2);
         Delay(100000);
         LCD_DATA_DISPLAY(3);
         Delay(100000);
         LCD_DATA_DISPLAY(4);
           Delay(100000);
         LCD_DATA_DISPLAY(5);
           Delay(100000);
         LCD_DATA_DISPLAY(6);
           Delay(100000);
           LCD_DATA_DISPLAY(7);
         Delay(100000);
           LCD_DATA_DISPLAY(8);
```

```
            Delay(100000);
            LCD_DATA_DISPLAY(9);
            Delay(100000);
            LCD_DATA_DISPLAY(10);
            Delay(100000);

          int Vdac_OutputMin=0x00;
            int Vdac_OutputMax=0xff;
            int Vdac_Initial=0;
            int Vdac_Buffer=0;
            int Vdac_Output=0;

            Vdac_Initial=(Vdac_OutputMax+Vdac_OutputMin)/2;  //Vdac_Initial=127
            DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,Vdac_Initial);    //DAC_Vrefp= VDDA, DAC_Vrefn= VSSA
            Vdac_Buffer=dac_04+1; //Vdac_Buffer=127+1=128

            ADfinish=0;
            for(i=1;i<=8;i++)
            {
                  while(!ADfinish);  //if ADfinish=0, stop at here

                 if(i<=7 && DAC_Calibration_ADCData<= ADCData) //if Vin<Vdac,  Vdac Output code - Vdac_Buffer
                 {
                DrvADC_Disable();
                DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                    Vdac_Buffer=Vdac_Buffer/2;
                      Vdac_Output=dac_04;
                      Vdac_Output=Vdac_Output-Vdac_Buffer;
                      dac_04=0xff00+Vdac_Output; //To do Vdac output voltage
                      DrvADC_Enable();
                      DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                      DrvADC_ClearIntFlag();
                      DrvADC_EnableInt();
                      while(!ADfinish);
                 }

                  else if(i<=7 && DAC_Calibration_ADCData>= ADCData)  //if Vin>Vdac, Vdac Output code + Vdac_Buffer
                 {
                      DrvADC_Disable();
                      DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                      Vdac_Buffer=Vdac_Buffer/2;
                      Vdac_Output=dac_04;
                      Vdac_Output=Vdac_Output+Vdac_Buffer;
                      dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
                      DrvADC_Enable();
                      DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                      DrvADC_ClearIntFlag();
                      DrvADC_EnableInt();
                      while(!ADfinish);
                 }

                  if(i==8 && DAC_Calibration_ADCData<= ADCData) //if Vin<Vdac, Vdac Output code + 0
                 {
                      DrvADC_Disable();
                      DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                      Vdac_Buffer=0;
                      Vdac_Output=dac_04;
                      Vdac_Output=Vdac_Output+Vdac_Buffer;
                      dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
                      DrvFlash_Burn_Word(0xf000,2000,Vdac_Output);  //DAC calibration Burn_Word at 0x9f000
                      DAC_Calibration_Flag=1; //If DAC_Calibration_Flag=1, leave
while((pio1_3<=PT1_7_low)&&(DAC_Calibration_Flag==0))
                      DrvADC_Enable();
                      DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                      DrvADC_ClearIntFlag();
                      DrvADC_EnableInt();
                      while(!ADfinish);
                 }

                  if(i==8 && DAC_Calibration_ADCData>= ADCData)  //if Vin>Vdac, Vdac Output code + 1
                 {
                      DrvADC_Disable();
                      DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                      Vdac_Buffer=1;
                      Vdac_Output=dac_04;
                      Vdac_Output=Vdac_Output+Vdac_Buffer;
                      dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
```

```
                DrvFlash_Burn_Word(0xf000,2000,Vdac_Output);  //DAC calibration Burn_Word at 0x9f000
                DAC_Calibration_Flag=1; //If DAC_Calibration_Flag=1, leave
while((pio1_3<=PT1_7_low)&&(DAC_Calibration_Flag==0))
                DrvADC_Enable();
                DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                DrvADC_ClearIntFlag();
                DrvADC_EnableInt();
                while(!ADfinish);
            }

            LCD_DATA_DISPLAY(ADCData);    // Doing DAC Calibration, and show ADCData
              ADfinish=0;
              MCUSTATUSbits.b_ADCdone=0;
        }
#endif

#if defined(DAC_Calibration_fast_mode)
           PT1INTSTATUSbits.b_PTINT0done=0;
           DAC_Calibration_ADCData=DAC_ADCData; //if goal is 330mV, Reference DAC_ADCData=0x24fdf
       // check DAC calibration time
       //DrvGPIO_SetPortBits(E_PT2,0x08);  //PT2.3=High
       //Delay(1000); //wait ? second
       //DrvGPIO_ClrPortBits(E_PT2,0x08);  //PT2.3=Low
           LCD_DATA_DISPLAY(0);
         Delay(100000); //Delay(100000)=wait 500ms
       // check DAC calibration time
       //DrvGPIO_SetPortBits(E_PT2,0x08);  //PT2.3=High
       //Delay(1000); //wait ? second
       //DrvGPIO_ClrPortBits(E_PT2,0x08);  //PT2.3=Low
         int Vdac_OutputMin=0x00;
           int Vdac_OutputMax=0xff;
           int Vdac_Initial=0;
           int Vdac_Buffer=0;
           int Vdac_Output=0;

           Vdac_Initial=(Vdac_OutputMax+Vdac_OutputMin)/2;  //Vdac_Initial=127
           DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,Vdac_Initial);    //DAC_Vrefp= VDDA, DAC_Vrefn= VSSA
           Vdac_Buffer=dac_04+1; //Vdac_Buffer=127+1=128

           ADfinish=0;
           for(i=1;i<=8;i++)
           {
                while(!ADfinish);  //if ADfinish=0, stop at here

              if(i<=7 && DAC_Calibration_ADCData<= ADCData) //if Vin<Vdac,  Vdac Output code - Vdac_Buffer
              {
            DrvADC_Disable();
            DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                Vdac_Buffer=Vdac_Buffer/2;
                  Vdac_Output=dac_04;
                  Vdac_Output=Vdac_Output-Vdac_Buffer;
                  dac_04=0xff00+Vdac_Output; //To do Vdac output voltage
                  DrvADC_Enable();
                  DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                  DrvADC_ClearIntFlag();
                  DrvADC_EnableInt();
                  while(!ADfinish);
              }

                else if(i<=7 && DAC_Calibration_ADCData>= ADCData)  //if Vin>Vdac, Vdac Output code + Vdac_Buffer
              {
                  DrvADC_Disable();
                  DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                  Vdac_Buffer=Vdac_Buffer/2;
                  Vdac_Output=dac_04;
                  Vdac_Output=Vdac_Output+Vdac_Buffer;
                  dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
                  DrvADC_Enable();
                  DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                  DrvADC_ClearIntFlag();
                  DrvADC_EnableInt();
                  while(!ADfinish);
              }

                if(i==8 && DAC_Calibration_ADCData<= ADCData) //if Vin<Vdac, Vdac Output code + 0
              {
                  DrvADC_Disable();
```

```
                DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                Vdac_Buffer=0;
                Vdac_Output=dac_04;
                Vdac_Output=Vdac_Output+Vdac_Buffer;
                dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
                DrvFlash_Burn_Word(0xf000,2000,Vdac_Output);  //DAC calibration Burn_Word at 0x9f000
                DAC_Calibration_Flag=1; //If DAC_Calibration_Flag=1, leave
while((pio1_3<=PT1_7_low)&&(DAC_Calibration_Flag==0))
            // check DAC calibration time
            //DrvGPIO_SetPortBits(E_PT2,0x08);  //PT2.3=High
            //Delay(1000); //wait ? second
            //DrvGPIO_ClrPortBits(E_PT2,0x08);  //PT2.3=Low
                DrvADC_Enable();
                DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                DrvADC_ClearIntFlag();
                DrvADC_EnableInt();
                while(!ADfinish);

            }

              if(i==8 && DAC_Calibration_ADCData>= ADCData)  //if Vin>Vdac, Vdac Output code + 1
            {
                DrvADC_Disable();
                DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                Vdac_Buffer=1;
                Vdac_Output=dac_04;
                Vdac_Output=Vdac_Output+Vdac_Buffer;
                dac_04=0xff00+Vdac_Output;  //To do Vdac output voltage
                DrvFlash_Burn_Word(0xf000,2000,Vdac_Output);  //DAC calibration Burn_Word at 0x9f000
                DAC_Calibration_Flag=1; //If DAC_Calibration_Flag=1, leave
while((pio1_3<=PT1_7_low)&&(DAC_Calibration_Flag==0))
            // check DAC calibration time
            //DrvGPIO_SetPortBits(E_PT2,0x08);  //PT2.3=High
            //Delay(1000); //wait ? second
            //DrvGPIO_ClrPortBits(E_PT2,0x08);  //PT2.3=Low
                DrvADC_Enable();
                DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                DrvADC_ClearIntFlag();
                DrvADC_EnableInt();
                while(!ADfinish);

            }

              ADfinish=0;
              MCUSTATUSbits.b_ADCdone=0;

          }
#endif

       }

  }


#if defined(ADC_ohm_show_19bit)
//Ohm table established in  DAC=330mV
//ADC setting : DrvADC_FullRefRange(1), OSR(0)
  K_ohm[0]=3.00;  //3k ohm
  //K_ohm[1]=30.00;  //30k ohm
  //K_ohm[2]=75.00;  //75k ohm
  //K_ohm[3]=300.00;  //300k ohm
  K_ohm_ADCount[0]=151790;  //3k ohm, ADC Count
  //K_ohm_ADCount[1]=15xxx;  //30k
  //K_ohm_ADCount[2]=60xx;  //75k
  //K_ohm_ADCount[3]=1xx;  //300k
#endif

  //Start to do ADC Measure RS_Strip

  DrvADC_Disable();
  DrvADC_CombFilter(DISABLE);  //Disable CombFilter

  //DAC Output Setting
  if(DAC_Calibration_Flag==0)  //Use default DAC output value
  {
    DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,DAC_Output);
    DrvDAC_SetoutputIO(ENABLE);       //DAC output with PT3.1
```

```
    DrvDAC_EnableOutput();           //DAC output enable
    DrvDAC_Enable();                 //DAC IP enable
  }

  //DAC Output Setting
  if(DAC_Calibration_Flag==1)  //Use Calibrated DAC output value
  {
      DAC_Output=ReadWord(0xf000);  //Read out Flash address 0x9f000
    DrvDAC_Open(E_DAC_PAIO4,E_DAC_NAIO7,DAC_Output);
    DrvDAC_SetoutputIO(ENABLE);       //DAC output with PT3.1
    DrvDAC_EnableOutput();           //DAC output enable
    DrvDAC_Enable();                 //DAC IP enable
  }

  // R2ROPA Setting
  DrvOP_PInput(E_OPP_DAO);
  DrvOP_NInput(E_OPN_AIO5);
  DrvOP_Open();
  DrvOP_OPOoutEnable(); //OPA OPEO=1


  InitalADC();  //initial ADC : P=AIO0, N=AIO5
  DisplayInit();
  ClearLCDframe();

  SYS_EnableGIE(7,0x1FF);  //Enable GIE(Global Interrupt)
  MCUSTATUSbits._byte = 0;
  PT1INTSTATUSbits._byte = 0;


  while(1)
  {

#if defined(ADC_Count_show_19bit)
      //No ADC chopper
      /*
      if(MCUSTATUSbits.b_ADCdone)
      {
         while(!ADfinish);
         LCD_DATA_DISPLAY(ADCData);   //to show ADC_Count_show_19bit
         ADfinish=0;
         MCUSTATUSbits.b_ADCdone=0;
      }
      */

       //ADC chopper
       for(i=0;i<=1;i++)
       {
            switch (i)
            {
                 case 0:  //已經抓到第一筆
                 while(!ADfinish);  //if ADfinish=0, stop at here, until enter ADC INT
                 ADfinish=0;
                 ADC_Array[0]=ADCData;
                 DrvADC_Disable();
                 DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                 DrvADC_SetADCInputChannel(ADC_Input_AIO5,ADC_Input_AIO0); //Set the ADC positive/negative input
voltage source.
                 DrvADC_Enable();
                 DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                 DrvADC_ClearIntFlag();
                 DrvADC_EnableInt();
                 while(!ADfinish);  //if ADfinish=0, stop at here
                 break;

                 case 1:  //已經抓到第二筆
                 ADfinish=0;
                 ADC_Array[1]=ADCData;
                 DrvADC_Disable();
                 DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                 ADCData_Chopper=(ADC_Array[0]-ADC_Array[1])>>1;  //divider by 2
                 LCD_DATA_DISPLAY(ADCData_Chopper);
                 DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO5); //Set the ADC positive/negative input
voltage source.
                 DrvADC_Enable();
                 DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                 DrvADC_ClearIntFlag();
```

```
                DrvADC_EnableInt();
                while(!ADfinish);      //if ADfinish=0, stop at here
                break;
             }
         }

#endif


#if defined(ADC_ohm_show_19bit)

      //No ADC chopper mode
      /*
    if(MCUSTATUSbits.b_ADCdone)
    {
         while(!ADfinish);
         ADCData_temp=ADCData;
         //Case1 : Abnormal case, show 0 ohm
         if((ADCData_temp==ADC_over_19bit)) //that means ADC Count over
         {
           ADCData_ohm_show=0;
           LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
         }
         //Case2 : Abnormal case, show 0 ohm
         if(ADCData_temp<=0)  //it is abnormal case
         {
           ADCData_temp=0;
           ADCData_ohm_show=ADCData_temp;
           LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
         }
         //Case3 : Normal case.
         if( (ADCData_temp>0) && (ADCData_temp!= ADC_over_19bit)) //It is normal case
         {
           ADCData_ohm_show=(K_ohm_ADCount[0]/ADCData_temp)*K_ohm[0];
           ADCData_ohm_show=ADCData_ohm_show*1000;

           if(ADCData_ohm_show>=500000)
           {
             ADCData_ohm_show=0;
             LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000

           }
             LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000

         }
         ADfinish=0;
         MCUSTATUSbits.b_ADCdone=0;
     }
     */

       //ADC chopper mode
       for(i=0;i<=1;i++)
       {
            switch (i)
            {
                 case 0:  //已經抓到第一筆
                 while(!ADfinish);  //if ADfinish=0, stop at here, until enter ADC INT
                 ADfinish=0;
                 ADC_Array[0]=ADCData;
                 DrvADC_Disable();
                 DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                 DrvADC_SetADCInputChannel(ADC_Input_AIO5,ADC_Input_AIO0); //Set the ADC positive/negative input
voltage source.
                 DrvADC_Enable();
                 DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                 DrvADC_ClearIntFlag();
                 DrvADC_EnableInt();
                 while(!ADfinish);  //if ADfinish=0, stop at here
                 break;

                 case 1:  //已經抓到第二筆
                 ADfinish=0;
                 ADC_Array[1]=ADCData;
                 DrvADC_Disable();
                 DrvADC_CombFilter(DISABLE);  //Disable CombFilter
                 ADCData_Chopper=(ADC_Array[0]-ADC_Array[1])>>1;  //divider by 2
                 ADCData_temp=ADCData_Chopper;
```

```
                //Case1 : Abnormal case, show 0 ohm
                if((ADCData_temp==ADC_over_19bit)) //that means ADC Count over
                {
                  ADCData_ohm_show=0;
                  LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
                }
                //Case2 : Abnormal case, show 0 ohm
                if(ADCData_temp<=0)  //it is abnormal case
                {
                  ADCData_temp=0;
                  ADCData_ohm_show=ADCData_temp;
                  LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
                 }
                 //Case3 : Normal case.
                 if( (ADCData_temp>0) && (ADCData_temp!= ADC_over_19bit)) //It is normal case
                 {
                   ADCData_ohm_show=(K_ohm_ADCount[0]/ADCData_temp)*K_ohm[0];
                   ADCData_ohm_show=ADCData_ohm_show*1000;
                   if(ADCData_ohm_show>=500000)
                   {
                      ADCData_ohm_show=0;
                      LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
                   }
                      LCD_Ohm_DISPLAY(ADCData_ohm_show); // for example : ADCData_ohm_show(3k)=3000
                 }
              DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO5); //Set the ADC positive/negative input
voltage source.
                DrvADC_Enable();
                DrvADC_CombFilter(ENABLE);  //Disable CombFilter
                DrvADC_ClearIntFlag();
                DrvADC_EnableInt();
                while(!ADfinish);      //if ADfinish=0, stop at here
                break;
            }
        }

#endif

  }
  return 0;
}

/*---------------------------------------------------------------------*/
/* ADC Interrupt Subroutines                                           */
/*---------------------------------------------------------------------*/
void HW2_ISR(void)
{
  if(DrvADC_ReadIntFlag()==1)
  {
    DrvADC_ClearIntFlag();
    ADCData=DrvADC_GetConversionData();
    ADCData=ADCData>>13; //To get high 19 bits ADC output code only
    MCUSTATUSbits.b_ADCdone=1;
    ADfinish=1;
  }

}

/*-------------------------------------------------------------------------*/
/* PT1 Interrupt Service Routines                                          */
/*-------------------------------------------------------------------------*/
void HW4_ISR(void)
{
  uint32_t PORT_IntFlag;
  PORT_IntFlag=DrvGPIO_GetIntFlag(E_PT1);
  if((PORT_IntFlag&0x02)==0x02)  // Enter PT1 INT to check PT1.1 INT Flag
  {
      PT1INTSTATUSbits.b_PTINT0done=1;
  }
    DrvGPIO_ClearIntFlag(E_PT1,0x02);  //clear PT1.1 interrupt flag
}
/*---------------------------------------------------------------------*/
/* ADC Initialization Subroutines                                      */
/*---------------------------------------------------------------------*/
void InitalADC(void)
{
```

```
  DrvADC_ClkEnable(0,1);       //Setting ADC CLOCK ADCK=HS_CK/6 & Rising edge is high
  //Set VDDA voltage
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_BandgapEnable();
  //Enable REFO
  DrvPMU_REFO_Enable();

  Delay(0x1000);
  DrvPMU_AnalogGround(ENABLE);      //ADC analog ground source selection.
                                   //1 : Enable buffer and use internal source(need to work with ADC)

  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO5); //Set the ADC positive/negative input voltage source.
  DrvADC_InputSwitch(OPEN);              //ADC signal input (positive and negative) short(VISHR) control.
  DrvADC_RefInputShort(OPEN);            //Set the ADC reference input (positive and negative) short(VRSHR)
control.

  DrvADC_Gain(ADC_PGA_Disable,ADC_PGA_Disable); //Input signal gain for modulator.
  DrvADC_DCoffset(0);                    //DC offset input voltage selection (VREF=REFP-REFN)
  DrvADC_RefVoltage(0,3);                     //Set the ADC reference voltage. VDDA-REFO
  DrvADC_FullRefRange(1);                //Set the ADC full reference range select.
                                        //0: Full reference range input
                                        //1: 1/2 reference range input
  //DrvADC_OSR(0);                      //0 : OSR=32768, output rate=10Hz
  DrvADC_OSR(1);                           //1 : OSR=16384, output rate=20Hz

  //Set ADC interrupt
  DrvADC_ClearIntFlag();
  DrvADC_EnableInt();
  DrvADC_Enable();

  DrvADC_CombFilter(ENABLE);   //Enable comb filter
  DrvADC_CombFilter(DISABLE);   //DISABLE comb filter
  DrvADC_CombFilter(ENABLE);   //Enable comb filter
}
/*---------------------------------------------------------------------*/
/* ADC Initialization Subroutines                                      */
/*---------------------------------------------------------------------*/
void InitalADC_DAC_Calibration(void)
{
  DrvADC_ClkEnable(0,1);       //Setting ADC CLOCK ADCK=HS_CK/6 & Rising edge is high
  //Set VDDA voltage
  DrvPMU_VDDA_LDO_Ctrl(E_LDO);
  DrvPMU_VDDA_Voltage(E_VDDA2_4);
  DrvPMU_BandgapEnable();
  //Enable REFO
  DrvPMU_REFO_Enable();

  Delay(0x1000);
  DrvPMU_AnalogGround(ENABLE);      //ADC analog ground source selection.
                                   //1 : Enable buffer and use internal source(need to work with ADC)

  //Set ADC input pin
  DrvADC_SetADCInputChannel(ADC_Input_AIO5,REFO_I); //Set the ADC positive/negative input voltage source.

  DrvADC_InputSwitch(OPEN);              //ADC signal input (positive and negative) short(VISHR) control.
  DrvADC_RefInputShort(OPEN);            //Set the ADC reference input (positive and negative) short(VRSHR)
control.

  DrvADC_Gain(ADC_PGA_Disable,ADC_PGA_Disable); //Input signal gain for modulator.
  DrvADC_DCoffset(0);                    //DC offset input voltage selection (VREF=REFP-REFN)
  DrvADC_RefVoltage(0,3);                     //Set the ADC reference voltage. VDDA-REFO
  DrvADC_FullRefRange(1);                //Set the ADC full reference range select.
                                        //0: Full reference range input
                                        //1: 1/2 reference range input
  //DrvADC_OSR(0);                            //0 : OSR=32768, output rate=10Hz
  //DrvADC_OSR(1);                      //1 : OSR=16384, output rate=20Hz
  DrvADC_OSR(5);                           //5 : OSR=1024, output rate=326Hz

  //Set ADC interrupt
  DrvADC_ClearIntFlag();
  DrvADC_EnableInt();
  DrvADC_Enable();

  DrvADC_CombFilter(ENABLE);   //Enable comb filter
  DrvADC_CombFilter(DISABLE);   //DISABLE comb filter
```

```
  DrvADC_CombFilter(ENABLE);   //Enable comb filter
}

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

/*---------------------------------------------------------------------------*/
/* Exception Service Routines                                                */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
  asm("nop"); //procedure define by customer.
  asm("nop");
}

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

7. 參考文獻

[1] http://www.hycontek.com/attachments/MSP/APD-HY16F002_TC.pdf, 紘康科技 HY16F188 血糖計(高精度校正)
[2] http://www.hycontek.com/attachments/MSP/DS-HY16F198_TC.pdf, 紘康科技 HY16F198 Datasheet.
[3] http://www.hycontek.com/attachments/MSP/UG-HY16F198_TC.pdf, 紘康科技 HY16F198 User Guide.

8. 修訂記錄

以下描述本文件差異較大的地方，而標點符號與字形的改變不在此描述範圍。

日期	文件版次	頁次	摘要
2016/06/22	V01	All	1.初版發行