Code used for Temperature calibration from ADS1248.

Blue line is definition.

Red line is Configuration

Green line is Function’s on which Page number they are used and called.

main()

{

sar\_InitializeHardware(); /// function used in Page-10

while(1)

{

Temp\_All\_Ch\_Data\_Process(); /// function used in Page-2

MB1\_process(); // Communication Purpose

}

}

void Temp\_All\_Ch\_Data\_Process(void) /// function called in Page-2

{

float temp = 0.0;

// IC1

ENABLE\_ADS1248\_IC1 // Enable IC1 & Disable all IC

//#define CONNFIG\_ADS1248\_IC1 GPIOC->CRL = (GPIOC->CRL & 0xFFFF0000)|0x00003338; // IC1

//#define ENABLE\_ADS1248\_IC1 GPIOC->ODR = 0xBBBD;

/DRDY = 1;

/CS\_PC1 = 0;

START = 1;

/RESET = 1;

Other IC’s

/DRDY = 1;

/CS\_PC1 = 1;

START = 0;

/RESET = 1;

MB1\_process(); // Communication Purpose

Delay\_Flag\_16ms = 0;

while(Delay\_Flag\_16ms == 0); // 16 ms Delay (every 16ms this flag is made high)

MB1\_process(); // Communication Purpose

/\*\*\*\*\*\* Configuration for MUX0 Register \*\*\*\*\*\*/

IC1\_MUX\_SP\_AIN0 // Positive i/p channel Selected

//#define IC1\_MUX\_SP\_AIN0 IC1\_MUX0\_Data&=0xC7; IC1\_MUX0\_Data|=0x00;

//char IC1\_MUX0\_Data = 0x01;

IC1\_MUX\_SN\_AIN1 // Negative i/p channel Selected

//#define IC1\_MUX\_SN\_AIN1 IC1\_MUX0\_Data&=0xF8; IC1\_MUX0\_Data|=0x01;

//char IC1\_MUX0\_Data = 0x01;

sar\_Write\_2\_Register(IC1\_MUX0\_Address,IC1\_MUX0\_Data); /// function used in Page-4

//#define IC1\_MUX0\_Address 0x00 //MUX0\_Register

//char IC1\_MUX0\_Data = 0x01;

/\*\*\*\*\*\* Configuration for IDAC1 Register \*\*\*\*\*\*/

IC1\_I1DIR\_AIN2 // 1st current source

//#define IC1\_I1DIR\_AIN2 IC1\_IDAC1\_Data&=0x0F; IC1\_IDAC1\_Data|=0x20;

//char IC1\_IDAC1\_Data= 0xFF;

IC1\_I2DIR\_AIN3 // 2nd current source // AIN2 is selected as excitation path (First Current Excitation Path selection)

//#define IC1\_I2DIR\_AIN3 IC1\_IDAC1\_Data&=0xF0; IC1\_IDAC1\_Data|=0x03;

//char IC1\_IDAC1\_Data= 0xFF;

sar\_Write\_2\_Register(IC1\_IDAC1\_Address,IC1\_IDAC1\_Data); /// function used in Page-4

//#define IC1\_IDAC1\_Address 0x0B //IDAC1\_Register

//char IC1\_IDAC1\_Data= 0xFF;

MB1\_process(); // Communication Purpose

Delay\_Flag\_16ms = 0;

while(Delay\_Flag\_16ms == 0); // 16 ms Delay

temp = Read\_IC1\_REF0(); // READ REF0 with Onboard REF.

/// Read\_IC1\_REF0() function used in Page-5

if(((temp\*4) > 1.0)&&((temp\*4) < 2.0)) //break point to check reference voltage (after 2 cycle of code run)

{

/\*\*\*\*\*\* Configuration for MUX1 Register \*\*\*\*\*\*/

IC1\_VREFCON\_Int\_ON // Internal Reference is always ON

//#define IC1\_VREFCON\_Int\_ON IC1\_MUX1\_Data&=0x9F; IC1\_MUX1\_Data|=0x20;// Internal\_Ref\_ON

//char IC1\_MUX1\_Data = 0x00;

IC1\_REFSELT\_REF0

//#define IC1\_REFSELT\_REF0 IC1\_MUX1\_Data&=0xE7; IC1\_MUX1\_Data|=0x00; // REF0\_i/p\_pair\_select

//char IC1\_MUX1\_Data = 0x00;

IC1\_MUXCAL\_Normal // Normal operation is selected as system monitor

//#define IC1\_MUXCAL\_Normal IC1\_MUX1\_Data&=0xF8; IC1\_MUX1\_Data|=0x00; //Select\_Normal\_Operation

//char IC1\_MUX1\_Data = 0x00;

sar\_Write\_2\_Register(IC1\_MUX1\_Address,IC1\_MUX1\_Data); /// function used in Page-4

//#define IC1\_MUX1\_Address 0x02 //MUX1\_Register

//char IC1\_MUX1\_Data = 0x00;

MB1\_process(); // Communication Purpose

Delay\_Flag\_16ms = 0;

while(Delay\_Flag\_16ms == 0); // 16 ms Delay

sar\_IC1\_CH1\_process\_function(); /// function used in Page-6

}

else

parameter1 = 1000.0\*10;

------------------------------------------------------------------------------------------------------------------------------------------

void sar\_Write\_2\_Register(int StartAddress,int Data) /// function called in Page-2, Page-5, Page-10

{

int tx\_flag=0, fl\_i=0;

SPI1\_Tx\_buffer[0] = 0x40 + StartAddress;

SPI1\_Tx\_buffer[1] = 0x00; //NumRegs

SPI1\_Tx\_buffer[2] = Data;

for(fl\_i=0;fl\_i<=2;fl\_i++)

{

do

{

if(SPI2\_TX\_BUF\_EMPTY\_FLAG) // Checks for Transmit buffer empty

//#define SPI2\_TX\_BUF\_EMPTY\_FLAG (SPI2->SR & 0x02)

tx\_flag=1;

}

while(!tx\_flag) ;

SPI2->DR = SPI1\_Tx\_buffer[fl\_i];

SPI2\_TX\_BUF\_EMPTY\_INT\_DIS

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

tx\_flag = 0;

}

}

------------------------------------------------------------------------------------------------------------------------------------------

float Read\_IC1\_REF0(void) /// function called in Page-3

{

volatile float tot\_adc\_val = 0.0;

volatile unsigned short int fl\_k = 0;

volatile unsigned short int fl\_index = 0;

volatile unsigned char SPI1\_Rx\_buffer[3]={0};

volatile unsigned int ADC\_Converted\_Data = 0;

volatile unsigned short int fl\_index1 = 0;

/\*\*\*\*\*\* Configuration for MUX1 Register \*\*\*\*\*\*/

IC1\_VREFCON\_Int\_ON // Internal Reference is always ON

//#define IC1\_VREFCON\_Int\_ON IC1\_MUX1\_Data&=0x9F; IC1\_MUX1\_Data|=0x20;

// Internal\_Ref\_ON

char IC1\_MUX1\_Data = 0x00;

IC1\_REFSELT\_Onboard\_Ref // Onboard Reference selected

//#define IC1\_REFSELT\_Onboard\_Ref IC1\_MUX1\_Data&=0xE7; IC1\_MUX1\_Data|=0x10;

// Onboard\_Ref\_select

char IC1\_MUX1\_Data = 0x00;

IC1\_MUXCAL\_Ext\_Ref0 // External REF0 is selected as system monitor

//#define IC1\_MUXCAL\_Ext\_Ref0 IC1\_MUX1\_Data&=0xF8; IC1\_MUX1\_Data|=0x05;

//Select\_Ext\_Ref0\_Measurement

char IC1\_MUX1\_Data = 0x00;

sar\_Write\_2\_Register(IC1\_MUX1\_Address,IC1\_MUX1\_Data); /// function used in Page-4

#define IC1\_MUX1\_Address 0x02 //MUX1\_Register

char IC1\_MUX1\_Data = 0x00;

/\*\*\*\*\*\* Delay \*\*\*\*\*\*/

Delay\_Flag\_16ms = 0;

while(Delay\_Flag\_16ms == 0); // 16 ms Delay

MB1\_process(); // Communication Purpose

/\*\*\*\*\*\* Read Data Once \*\*\*\*\*\*/

while((GPIOC->IDR & 0x0001)==0);

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0x12; // Read Once Cmd is 12 or 13.

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_dummy\_buffer = SPI2->DR; // receive 1st byte which is a dummy data

while((GPIOC->IDR & 0x0001)==0);

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0x12; // Read Once Cmd is 12 or 13.

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_dummy\_buffer = SPI2->DR; // receive 1st byte which is a dummy data

for(fl\_k=0;fl\_k<3;fl\_k++)

{

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0xff; // NOP Cmd

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_buffer[fl\_index++]=SPI2->DR;

}

/\*\*\*\*\* ADC COUNT CONVERTION \*\*\*\*\*/

if(fl\_index >= 3)

{

ADC\_Converted\_Data = SPI1\_Rx\_buffer[0] << 16; // MSB

ADC\_Converted\_Data |= SPI1\_Rx\_buffer[1] << 8; // Mid-Byte

ADC\_Converted\_Data |= SPI1\_Rx\_buffer[2]; // LSB

tot\_adc\_val = ADC\_BIT\_VALUE \* ADC\_Converted\_Data;

//#define ADC\_BIT\_VALUE 2.4414065410383392618106915725102e-7 // (2.048/2^23)

}

return(tot\_adc\_val);

}

--------------------------------------------------------------------------------------------------------------------------------------

void sar\_IC1\_CH1\_process\_function(void) /// function called in Page-3

{

volatile float Temp\_calibration = 0.0;

float calib\_degree=0.0;

Temp\_calibration = sar\_IC1\_CH1\_Read\_Data\_Once(); /// function used in Page-7

calib\_degree = sar\_IC1\_Temperature\_calculation(Temp\_calibration);

/// sar\_IC1\_Temperature\_calculation function used in Page-11

parameter1 = calib\_degree\*10;

}

-------------------------------------------------------------------------------------------------------------------------------------

float sar\_IC1\_CH1\_Read\_Data\_Once(void) /// function called in Page-9

{

volatile float fl\_Avg\_Data = 0.0;

volatile float tot\_adc\_val = 0.0;

volatile unsigned short int fl\_k = 0;

volatile unsigned short int fl\_index = 0;

volatile unsigned char SPI1\_Rx\_buffer[3]={0};

volatile unsigned int ADC\_Converted\_Data = 0;

while((GPIOC->IDR & 0x0001)==0);

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0x12; // Read Once Cmd is 12 or 13.

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_dummy\_buffer = SPI2->DR; // receive 1st byte which is a dummy data

while((GPIOC->IDR & 0x0001)==0);

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0x12; // Read Once Cmd is 12 or 13.

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_dummy\_buffer = SPI2->DR; // receive 1st byte which is a dummy data

for(fl\_k=0;fl\_k<3;fl\_k++)

{

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI2->DR =0xff; // NOP Cmd

while(SPI2\_BUSY\_FLAG); // Wait for Busy flag reset to send next data byte

SPI1\_Rx\_buffer[fl\_index++]=SPI2->DR;

}

/\*\*\*\*\* ADC COUNT CONVERTION \*\*\*\*\*/

if(fl\_index >= 3)

{

ADC\_Converted\_Data = SPI1\_Rx\_buffer[0] << 16; // MSB

ADC\_Converted\_Data |= SPI1\_Rx\_buffer[1] << 8; // Mid-Byte

ADC\_Converted\_Data |= SPI1\_Rx\_buffer[2]; // LSB

tot\_adc\_val = ADC\_BIT\_VALUE\_REF0\_IC1 \* ADC\_Converted\_Data;

//#define ADC\_BIT\_VALUE\_REF0\_IC1 0.0000001965045928955078125

// Read Ref0 = 1.6484 ADC\_BIT\_VALUE\_REF0\_IC1 = (1.6484 / 2^23)

if((tot\_adc\_val < 0.8098)&&(tot\_adc\_val > 0.03068))

// 0.809868 volt = 900 degree & 0.0306878 volt = -210 Degree (If I = 1.0mA)

{

IC1\_Ch1\_buffer[IC1\_Ch1\_Data\_index++] = tot\_adc\_val ;

if(IC1\_Ch1\_Data\_index >= c\_AvgLength)

//#define c\_AvgLength 2

IC1\_Ch1\_Data\_index = 0;

fl\_Avg\_Data =0.0 ;

for(fl\_k=0;fl\_k < c\_AvgLength;fl\_k++)

{

fl\_Avg\_Data += IC1\_Ch1\_buffer[fl\_k] ;

}

fl\_Avg\_Data /= c\_AvgLength;

}

else if(tot\_adc\_val > 0.8098)

fl\_Avg\_Data = 0.03; // 2 volt -210 degree

else if(tot\_adc\_val <= 0.03068)

fl\_Avg\_Data = 0.03; // 0.045 volt

}

return(fl\_Avg\_Data);

}

----------------------------------------------------------------------------------------------------------------------------

void sar\_InitializeHardware(void) /// function called from Page-2

{

CONNFIG\_ADS1248\_IC1

ADS1248\_Wakeup\_Function(); /// function used in Page-9

ENABLE\_ADS1248\_IC1 // Enable IC1 & Disable IC2 & IC3. NOT Reseting Any IC's

Delay\_Flag\_16ms = 0;

while(Delay\_Flag\_16ms == 0); // 16 ms Delay

sar\_IC1\_Config\_function(); // Configure Vref, Amount of Current Generation & Gain

/// sar\_IC1\_Config\_function(); function used in Page-10

}

void ADS1248\_Wakeup\_Function(void) /// function called from Page-9

{

volatile u16 fl\_m = 0,fl\_n = 0; // unsigned short = u16

/\*\*\*\*\*\*\*\*\*\*\*\*\* WAKE UP CMD\*\*\*\*\*\*\*\*\*\*/

while(SPI\_GetFlagStatus(SPI2\_TX\_BUF\_EMPTY\_FLAG)==0);

SPI2->DR = 0X00; // Wakeup Cmd 0x00h or 0x01h (Exit Sleep Mode)

while(SPI\_GetFlagStatus(SPI2\_BUSY\_FLAG)==1);

/\*\*\*\*\*\*\*\*\*\*\*\*\* RESET CMD\*\*\*\*\*\*\*\*\*\*/

for(fl\_m=0;fl\_m<=5;fl\_m++) // 16ms delay

{

for(fl\_n=0;fl\_n<57600;fl\_n++);

}

while(SPI\_GetFlagStatus(SPI2\_TX\_BUF\_EMPTY\_FLAG)==RESET);

SPI2->DR = 0X06; // Reset Cmd 0x06h or 0x07h (Exit Sleep Mode)

while(SPI\_GetFlagStatus(SPI2\_BUSY\_FLAG)==SET);

}

----------------------------------------------------------------------------------------------------------------------------------------

/\* IC1 Configuration \*/

void sar\_IC1\_Config\_function(void) /// function called from Page-9

{

/\*\*\*\*\*\* Configuration for MUX1 Register \*\*\*\*\*\*/

IC1\_VREFCON\_Int\_ON // Internal Reference is always ON

//#define IC1\_VREFCON\_Int\_OFF IC1\_MUX1\_Data&=0x9F; IC1\_MUX1\_Data|=0x00;

// Internal\_Ref\_OFF

//char IC1\_MUX1\_Data = 0x00;

IC1\_REFSELT\_REF0

//#define IC1\_REFSELT\_REF0 IC1\_MUX1\_Data&=0xE7; IC1\_MUX1\_Data|=0x00;

// REF0\_i/p\_pair\_select

//char IC1\_MUX1\_Data = 0x00;

IC1\_MUXCAL\_Normal // Normal operation is selected as system monitor

//#define IC1\_MUXCAL\_Normal IC1\_MUX1\_Data&=0xF8; IC1\_MUX1\_Data|=0x00; //Select\_Normal\_Operation

//char IC1\_MUX1\_Data = 0x00;

sar\_Write\_2\_Register(IC1\_MUX1\_Address,IC1\_MUX1\_Data); /// function used in Page-4

//#define IC1\_MUX1\_Address 0x02 //MUX1\_Register

//char IC1\_MUX1\_Data = 0x00;

/\*\*\*\*\*\* Configuration for SYS0 Register \*\*\*\*\*\*/

IC1\_PGA\_2 // Gain = 4 is selected

//#define IC1\_PGA\_2 IC1\_SYS0\_Data&=0x0F; IC1\_SYS0\_Data|=0x10;

//char IC1\_SYS0\_Data = 0x00;

IC1\_DOR\_80\_SPS // Output data rate is selected as 2000 samples

//#define IC1\_DOR\_80\_SPS IC1\_SYS0\_Data&=0x70; IC1\_SYS0\_Data|=0x04;

//char IC1\_SYS0\_Data = 0x00;

sar\_Write\_2\_Register(IC1\_SYS0\_Address,IC1\_SYS0\_Data); /// function used in Page-4

//#define IC1\_SYS0\_Address 0x03 //SYS0\_Register

//char IC1\_SYS0\_Data = 0x00;

/\*\*\*\*\*\* Configuration for IDAC0 Register \*\*\*\*\*\*/

IC1\_DRDYMode\_DOUT // Functions as Data Out

//#define IC1\_DRDYMode\_DOUT IC1\_IDAC0\_Data&=0x07; IC1\_IDAC0\_Data|=0x00;

//only Dout default

//char IC1\_IDAC0\_Data= 0x00;

IC1\_IMAG\_1000uA // 1mA

//#define IC1\_IMAG\_1000uA IC1\_IDAC0\_Data&=0x07; IC1\_IDAC0\_Data|=0x06; // 1 mA

//char IC1\_IDAC0\_Data= 0x00;

sar\_Write\_2\_Register(IC1\_IDAC0\_Address,IC1\_IDAC0\_Data); /// function used in Page-4

//#define IC1\_IDAC0\_Address 0x0A //IDAC0\_Register

//char IC1\_IDAC0\_Data= 0x00;

}

float sar\_IC1\_Temperature\_calculation(float Temp\_calibration) /// function called in Page-6

{

static float resistance=0.0;

static float temperature=0.0;

static float Pow\_1=0.0,Pow\_2=0.0;

float calib\_degree=0.0;

resistance = Temp\_calibration/2.0;

resistance = resistance/0.001;

if(resistance > 96.675) // 96.675ohm = -8.5 Degree Celsius

{

temperature=sqrt(0.15274-(0.000231 \* (resistance-100)));

degree = (0.39083 - temperature)/0.0001155;

}

else

{

Pow\_1 = resistance;

Pow\_2 = Pow\_1\*resistance;

temperature = -242.97+(2.2838\*Pow\_1)+(1.4727e-3\*Pow\_2);

degree = temperature;

}

return(degree);

}