#include <msp430.h>  
#define CALADC\_15V\_30C  \*((unsigned int \*)0x1A1A)                 // Temperature Sensor Calibration-30 C //6682                                                                 // See device datasheet for TLV table memory mapping //6684  
#define CALADC\_15V\_85C  \*((unsigned int \*)0x1A1C)                 // Temperature Sensor Calibration-High Temperature (85 for Industrial, 105 for Extended)  
  
  
  
volatile long temp1;  
volatile float IntDegC1;  
  
void ConfigClocks(void);  
void port\_init();  
void ConfigureAdc\_temp1();  
void initialize\_Adc();  
void Software\_Trim();  
  
void main(void)  
{  
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer  
    PM5CTL0 &= ~LOCKLPM5;  
  
  
        ConfigClocks();  
        port\_init();  
  
  
        \_delay\_cycles(5);                // Wait for ADC Ref to settle  
  
        while(1){  
  
          initialize\_Adc();  
          PMMCTL0\_H = PMMPW\_H;                                          // Unlock the PMM registers read 2.2.8 & 2.2.9 form the manual  
          PMMCTL2 |= INTREFEN | TSENSOREN | REFVSEL\_0;                  // Enable internal 1.5V reference and temperature sensor  
  
          ConfigureAdc\_temp1();  
          ADCCTL0 |= ADCENC + ADCSC +ADCMSC;        // Converter Enable, Sampling/conversion start  
          while((ADCCTL0 & ADCIFG) == 0);    // check the Flag, while its low just wait  
          \_delay\_cycles(2000);  
          temp1 = ADCMEM0;                    // read the converted data into a variable  
          ADCCTL0 &= ~ADCIFG;  
          IntDegC1 = (temp1-CALADC\_15V\_30C)\*(85-30)/(CALADC\_15V\_85C-CALADC\_15V\_30C)+30;  
                    }  
  
}  
  
void ConfigClocks(void)  
{  
    CSCTL3 = SELREF\_\_REFOCLK;               // Set REFO as FLL reference source  
    CSCTL1 = DCOFTRIMEN\_1 | DCOFTRIM0 | DCOFTRIM1 | DCORSEL\_0;// DCOFTRIM=3, DCO Range = 1MHz  
    CSCTL2 = FLLD\_0 + 30;                   // DCODIV = 1MHz  
    \_\_delay\_cycles(3);  
    \_\_bic\_SR\_register(SCG0);                // Enable FLL  
    Software\_Trim();                        // Software Trim to get the best DCOFTRIM value  
    CSCTL4 = SELMS\_\_DCOCLKDIV | SELA\_\_REFOCLK; // set default REFO(~32768Hz) as ACLK source, ACLK = 32768Hz                                             // default DCODIV as MCLK and SMCLK source  
}  
  
  
void port\_init(){  
    P1SEL0 |= BIT3;// | BIT7;  
    P1SEL1 |= BIT3;// | BIT7;  
}  
  
  
// Configure ADC Temperature  
void ConfigureAdc\_temp1(){  
     ADCCTL0 |= ADCSHT\_8 | ADCON;                                  // ADC ON,temperature sample period>30us  
     ADCCTL1 |= ADCSHP;                                            // s/w trig, single ch/conv, MODOSC  
     ADCCTL2 &= ~ADCRES;                                           // clear ADCRES in ADCCTL  
     ADCCTL2 |= ADCRES\_2;                                          // 12-bit conversion results  
     ADCMCTL0 |= ADCSREF\_1 | ADCINCH\_12;                           // ADC input ch A12 => temp sense  
     //ADCMCTL0 |= ADCSREF\_1 | ADCINCH\_3;                           // ADC input ch A12 => temp sense  
     ADCIE |=ADCIE0;  
}  
  
  
void initialize\_Adc(){  
     ADCCTL0 &= ~ADCIFG;//CLEAR FLAG  
     ADCMEM0=0x00000000;  
     //ADCAE0=0x00;  
     ADCCTL0=0x0000;  
     ADCCTL1=0x0000;  
}  
  
void Software\_Trim()  
{  
    unsigned int oldDcoTap = 0xffff;  
    unsigned int newDcoTap = 0xffff;  
    unsigned int newDcoDelta = 0xffff;  
    unsigned int bestDcoDelta = 0xffff;  
    unsigned int csCtl0Copy = 0;  
    unsigned int csCtl1Copy = 0;  
    unsigned int csCtl0Read = 0;  
    unsigned int csCtl1Read = 0;  
    unsigned int dcoFreqTrim = 3;  
    unsigned char endLoop = 0;  
  
    do  
    {  
        CSCTL0 = 0x100;                         // DCO Tap = 256  
        do  
        {  
            CSCTL7 &= ~DCOFFG;                  // Clear DCO fault flag  
        }while (CSCTL7 & DCOFFG);               // Test DCO fault flag  
  
        //\_\_delay\_cycles((unsigned int)3000 \* MCLK\_FREQ\_MHZ);// Wait FLL lock status (FLLUNLOCK) to be stable  
                                                           // Suggest to wait 24 cycles of divided FLL reference clock  
        while((CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)) && ((CSCTL7 & DCOFFG) == 0));  
  
        csCtl0Read = CSCTL0;                   // Read CSCTL0  
        csCtl1Read = CSCTL1;                   // Read CSCTL1  
  
        oldDcoTap = newDcoTap;                 // Record DCOTAP value of last time  
        newDcoTap = csCtl0Read & 0x01ff;       // Get DCOTAP value of this time  
        dcoFreqTrim = (csCtl1Read & 0x0070)>>4;// Get DCOFTRIM value  
  
        if(newDcoTap < 256)                    // DCOTAP < 256  
        {  
            newDcoDelta = 256 - newDcoTap;     // Delta value between DCPTAP and 256  
            if((oldDcoTap != 0xffff) && (oldDcoTap >= 256)) // DCOTAP cross 256  
                endLoop = 1;                   // Stop while loop  
            else  
            {  
                dcoFreqTrim--;  
                CSCTL1 = (csCtl1Read & (~DCOFTRIM)) | (dcoFreqTrim<<4);  
            }  
        }  
        else                                   // DCOTAP >= 256  
        {  
            newDcoDelta = newDcoTap - 256;     // Delta value between DCPTAP and 256  
            if(oldDcoTap < 256)                // DCOTAP cross 256  
                endLoop = 1;                   // Stop while loop  
            else  
            {  
                dcoFreqTrim++;  
                CSCTL1 = (csCtl1Read & (~DCOFTRIM)) | (dcoFreqTrim<<4);  
            }  
        }  
  
        if(newDcoDelta < bestDcoDelta)         // Record DCOTAP closest to 256  
        {  
            csCtl0Copy = csCtl0Read;  
            csCtl1Copy = csCtl1Read;  
            bestDcoDelta = newDcoDelta;  
        }  
  
    }while(endLoop == 0);                      // Poll until endLoop == 1  
  
    CSCTL0 = csCtl0Copy;                       // Reload locked DCOTAP  
    CSCTL1 = csCtl1Copy;                       // Reload locked DCOFTRIM  
    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked  
}