```
#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;


// Configure GPIO

  P6DIR |= BIT1;                                  // Set P1.0/LED to output direction

  P6OUT &= ~BIT1;                                   // P1.0 LED off


       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;
        IntDegC1 = (temp1-50)*(25-50)/(5000-2000)+50;


  if (IntDegC1 < 35)
    P6OUT |= BIT1;                          // Set P1.0 LED on
  else
    P6OUT &= ~BIT1; // Clear P1.0 LED off
  __delay_cycles(5000);


            }

}

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_0 | 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
}
```