```c
#include <msp430.h>
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "stdint.h"
//#include "msp430f6736a.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#ifndef __MSP430FR2311
#define __MSP430FR2311
#include "MSP430FR2311.h"
#endif
#ifndef init_h
#define init_h
#include "init.h"
#endif
#ifndef tempsensor_h
#define tempsensor_h
#include "tempsensor.h"
#endif
#ifndef i2c_h
#define i2c_h
#include "i2c.h"
#endif
#ifndef i2c1_h
#define i2c1_h
#include "i2c1.h"
#endif
unsigned char TXData[]=
{0xA1,0xB1,0xC1,0xD1},TXmit[20],RCv[20],IC2_TXmit[20],IC2_RCv[20],*PRxData,
*PTxData,RX_DONE,error_flag,testi2c,IC2_on,
*PRx_I2CData ,*PTx_I2CData,test;        // Pointer to TX data
unsigned char SlaveAddress[]= {0x0A,0x0B,0x0C,0x0D};
unsigned int TXByteCtr,RXByteCtr,TX_I2CByteCtr, RX_I2CByteCtr;
unsigned char SlaveFlag = 0;

/*
 * main.c
 */
int main(void) {

    // __bic_SR_register( GIE);
    WDTCTL = WDTPW + WDTHOLD;                               // Stop WDT   // Enter LPM0,
interrupts enabled
    init_ports();
    init_ext_32k();
    change_clock_freq(MCLK_FREQ_MHZ);
    IC2_TXmit[0]=0x01;
    IC2_TXmit[1]=0x81;     //sec
    IC2_TXmit[2]=0x04;
    IC2_TXmit[3]=0x77;
    IC2_TXmit[4]=0x88;
    IC2_TXmit[5]=0x99;
    IC2_TXmit[6]=0xAA;
    IC2_TXmit[7]=0xBB;
    IC2_TXmit[8]=0x33;
```

```
      IC2_TXmit[9]=0x33;
      IC2_TXmit[10]=0x33;

   IC2_RCv[0]=00;
   IC2_RCv[1]=00;
   IC2_RCv[2]=00;
   IC2_RCv[3]=00;
   IC2_RCv[4]=00;
   IC2_RCv[5]=00;
   IC2_RCv[6]=00;
   IC2_RCv[7]=00;
   IC2_RCv[8]=00;
   IC2_RCv[9]=00;
   IC2_RCv[10]=00;


 __bis_SR_register( GIE);           // Enter LPM3 w/ interrupts
     __no_operation();

     init_i2c();
    i2c_xmit( &IC2_TXmit[0],0x68,7);
    __no_operation();
   P1OUT =0x40;
    __bis_SR_register( GIE);
    __no_operation();

    __bis_SR_register( GIE);
while(1)
{
   i2c_xmit( &IC2_TXmit[0],0x68,1);
   IC2_on=0;
   test=0;
       i2c_rcv( &IC2_RCv[0],0x48,2);
   i2c_xmit( &IC2_TXmit[0],0x68,2);
      i2c_xmit( &IC2_TXmit[0],0x68,3);
    __no_operation();

  __no_operation();


    IC2_RCv[0]=00;
       IC2_RCv[1]=00;
       IC2_RCv[2]=00;
       IC2_RCv[3]=00;
       IC2_RCv[4]=00;
       IC2_RCv[5]=00;
       IC2_RCv[6]=00;
       IC2_RCv[7]=00;
       IC2_RCv[8]=00;
       IC2_RCv[9]=00;
       IC2_RCv[10]=00;
}
    P1SEL0 &= ~BIT2 + ~BIT3;
    P1DIR |= BIT2 | BIT3;
    P1OUT |= BIT2 | BIT3;

    __bis_SR_register( GIE);
```

```
     xmit_i2c_add(0x68,0x00);
      xmit_data( &IC2_TXmit[0],10);
      stop();
     while(1)
     {
         xmit_i2c_add(0x68,0x00);
         xmit_data( &IC2_TXmit[0],10);
         stop();
     xmit_i2c_add(0x68,0x01);
    rcv_data(&IC2_RCv[0],10);

  //        xmit_i2c_add(0x68,0x00);
  //        xmit_data( &IC2_TXmit[0],1);

      stop();
     }
     while(1);
 //     stop();
      xmit_i2c_add(0x68,0x01);
      rcv_data(&IC2_RCv[4],3);
      stop();
       __no_operation();

/*
while(1)
{
    __no_operation();
     IC2_TXmit[0]=00;
     PRx_I2CData= &IC2_RCv[0];


     UCB0CTLW0 &= ~UCSWRST;
      UCB0IE |= UCTXIE0;                     // transmit and NACK interrupt enable
         UCB0I2CSA = 0x68;
         TX_I2CByteCtr=1;
         PTx_I2CData= &IC2_TXmit[0];
        __no_operation();
        while (UCB0CTLW0 & UCTXSTP);                      // Ensure stop condition got sent
        UCB0CTLW0 |= UCTR | UCTXSTT;                      // I2C TX, start condition
        __no_operation();
        __bis_SR_register(GIE);               // Enter LPM0 w/ interrupts
     while(TX_I2CByteCtr !=0);
     while (UCB0CTL1 & UCTXSTP);              // Ensure stop condition got sent
      while (UCB0STAT & UCBBUSY);            // make sure buss in not busy
       __no_operation();
      P1OUT =0x40;
     __bis_SR_register( GIE);

     __no_operation();
    i2c_rcv( &IC2_RCv[0],0x68,7);
}
     while(1)
      {
        P1OUT |=BIT7;
          __delay_cycles(15000);
         P1OUT &=~BIT7;
         __delay_cycles(15000);
      }

     while(1)
     {
        __no_operation();
//       __bic_SR_register( GIE);
//       put_temp_sensor_reg(&IC2_TXmit[0],0x01);
//       get_temp_sensor_data(&IC2_TXmit[0],&IC2_RCv[0]);

     }
                 // default DCODIV as MCLK and SMCLK source
*/
                                       // to activate previously configured port settings
// DCOCLK = MCLK and SMCLK source





                                             // to activate previously configured port settings
  while(1)
  {
    P1OUT |=BIT7;
      __delay_cycles(15000);
     P1OUT &=~BIT7;
     __delay_cycles(15000);
  }
    return 0;
}

#pragma vector = USCI_B0_VECTOR
__interrupt void USCIB0_ISR(void)
{
  switch(__even_in_range(UCB0IV,USCI_I2C_UCBIT9IFG))
  {
        case USCI_NONE: break;                         // Vector 0: No interrupts break;
        case USCI_I2C_UCALIFG: break;
        case USCI_I2C_UCNACKIFG:
            __no_operation();
            UCB0CTL1 |= UCTXSTT;                       //resend start if NACK
          break;                                       // Vector 4: NACKIFG break;
        case USCI_I2C_UCSTTIFG:
            __no_operation();
            break;                    // Vector 6: STTIFG break;
        case USCI_I2C_UCSTPIFG:
            __no_operation();
            break;                    // Vector 8: STPIFG break;
        case USCI_I2C_UCRXIFG3:
            __no_operation();
            break;                    // Vector 10: RXIFG3 break;
        case USCI_I2C_UCTXIFG3:
            __no_operation();
            break;                    // Vector 14: TXIFG3 break;
        case USCI_I2C_UCRXIFG2:
            __no_operation();
            break;                    // Vector 16: RXIFG2 break;
        case USCI_I2C_UCTXIFG2:
            __no_operation();
            break;                     // Vector 18: TXIFG2 break;
        case USCI_I2C_UCRXIFG1:
            __no_operation();
            break;                     // Vector 20: RXIFG1 break;
        case USCI_I2C_UCTXIFG1:
            __no_operation();
            break;                     // Vector 22: TXIFG1 break;
        case USCI_I2C_UCRXIFG0:
            RX_I2CByteCtr--;                               // Decrement RX byte counter
                    IC2_on++;
            if (RX_I2CByteCtr) {
                        *PRx_I2CData++ = UCB0RXBUF;           // Move RX data to address PRxData
                        if (RX_I2CByteCtr == 1)                   // Only one byte left?
                        {
                             test++;
                            UCB0CTL1 |= UCTXSTP;               // Generate I2C stop condition
                          __no_operation();
                        }
                        //                      UCB0IE &= ~UCRXIE;          //turn off rcv interrupt
                    } else {
                        *PRx_I2CData = UCB0RXBUF;                // Move final RX data to PRxData
                        //      __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
                //      *PRx_I2CData = 0x77;
                        testi2c = 0;
                      // UCB0IFG &= ~UCRXIFG;
                    }
            __no_operation();
            break;                     // Vector 24: RXIFG0 break;
        case USCI_I2C_UCTXIFG0:
            __no_operation();
        if (TX_I2CByteCtr)                                 // Check TX byte counter
           {
   //          UCB0TXBUF = 0x01 ;             // Load TX buffer
         __no_operation();
            UCB0TXBUF = *PTx_I2CData++;             // Load TX buffer
            TX_I2CByteCtr--;                               // Decrement TX byte counter

           }
        else
           {
        __no_operation();
            UCB0CTLW0 |= UCTXSTP;                        // I2C stop condition
            UCB0IFG &= ~UCTXIFG;                         // Clear USCI_B0 TX int flag
   //          __bic_SR_register_on_exit(LPM0_bits);      // Exit LPM0
           }
          break;                                         // Vector 26: TXIFG0 break;
        case USCI_I2C_UCBCNTIFG:
            __no_operation();
            break;                    // Vector 28: BCNTIFG
        case USCI_I2C_UCCLTOIFG:
            __no_operation();
            break;                    // Vector 30: clock low timeout
        case USCI_I2C_UCBIT9IFG:
            __no_operation();
            break;                  // Vector 32: 9th bit
        default: break;
  }
}
#pragma vector = TIMER0_B0_VECTOR
__interrupt void Timer_B (void)

{
    P1OUT ^= BIT6;
    TB0CCR0 += 50;
}




```