

patch_main.c

```
1#include <msp430.h>
2#include "stdio.h"
3#include "stdlib.h"
4#include "string.h"
5#include "stdint.h"
6//#include "msp430f6736a.h"
7#include "stdio.h"
8#include "stdlib.h"
9#include "string.h"
10#ifndef __MSP430FR2311
11#define __MSP430FR2311
12#include "MSP430FR2311.h"
13#endif
14#ifndef init_h
15#define init_h
16#include "init.h"
17#endif
18#ifndef tempsensor_h
19#define tempsensor_h
20#include "tempsensor.h"
21#endif
22#ifndef i2c_h
23#define i2c_h
24#include "i2c.h"
25#endif
26#ifndef i2c1_h
27#define i2c1_h
28#include "i2c1.h"
29#endif
30unsigned 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,
31*PRx_I2CData ,*PTx_I2CData,test; // Pointer to TX data
32unsigned char SlaveAddress[]= {0x0A,0x0B,0x0C,0x0D};
33unsigned int TXByteCtr,RXByteCtr,TX_I2CByteCtr, RX_I2CByteCtr;
34unsigned char SlaveFlag = 0;
35
36/*
37 * main.c
38 */
39int main(void) {
40
41    // __bic_SR_register( GIE);
42    WDTCTL = WDTPW + WDTCTL; // Stop WDT // Enter LPM0,
    interrupts enabled
43    init_ports();
44    init_ext_32k();
45    change_clock_freq(MCLK_FREQ_MHZ);
46    IC2_TXmit[0]=0x01;
47    IC2_TXmit[1]=0x81; //sec
48    IC2_TXmit[2]=0x04;
49    IC2_TXmit[3]=0x77;
50    IC2_TXmit[4]=0x88;
51    IC2_TXmit[5]=0x99;
52    IC2_TXmit[6]=0xAA;
53    IC2_TXmit[7]=0xBB;
54    IC2_TXmit[8]=0x33;
```

patch_main.c

```
55     IC2_TXmit[9]=0x33;
56     IC2_TXmit[10]=0x33;
57
58     IC2_RCv[0]=00;
59     IC2_RCv[1]=00;
60     IC2_RCv[2]=00;
61     IC2_RCv[3]=00;
62     IC2_RCv[4]=00;
63     IC2_RCv[5]=00;
64     IC2_RCv[6]=00;
65     IC2_RCv[7]=00;
66     IC2_RCv[8]=00;
67     IC2_RCv[9]=00;
68     IC2_RCv[10]=00;
69
70
71     __bis_SR_register( GIE);           // Enter LPM3 w/ interrupts
72     __no_operation();
73
74     init_i2c();
75     i2c_xmit( &IC2_TXmit[0],0x68,7);
76     __no_operation();
77     P1OUT =0x40;
78     __bis_SR_register( GIE);
79     __no_operation();
80
81     __bis_SR_register( GIE);
82 while(1)
83 {
84     i2c_xmit( &IC2_TXmit[0],0x68,1);
85     IC2_on=0;
86     test=0;
87     i2c_rcv( &IC2_RCv[0],0x48,2);
88     i2c_xmit( &IC2_TXmit[0],0x68,2);
89     i2c_xmit( &IC2_TXmit[0],0x68,3);
90     __no_operation();
91
92     __no_operation();
93
94
95     IC2_RCv[0]=00;
96     IC2_RCv[1]=00;
97     IC2_RCv[2]=00;
98     IC2_RCv[3]=00;
99     IC2_RCv[4]=00;
100    IC2_RCv[5]=00;
101    IC2_RCv[6]=00;
102    IC2_RCv[7]=00;
103    IC2_RCv[8]=00;
104    IC2_RCv[9]=00;
105    IC2_RCv[10]=00;
106 }
107     P1SEL0 &= ~BIT2 + ~BIT3;
108     P1DIR |= BIT2 | BIT3;
109     P1OUT |= BIT2 | BIT3;
110
111     __bis_SR_register( GIE);
```

patch_main.c

```

112 xmit_i2c_add(0x68,0x00);
113 xmit_data( &IC2_TXmit[0],10);
114 stop();
115 while(1)
116 {
117     xmit_i2c_add(0x68,0x00);
118     xmit_data( &IC2_TXmit[0],10);
119     stop();
120 xmit_i2c_add(0x68,0x01);
121 rcv_data(&IC2_RCv[0],10);
122
123 //     xmit_i2c_add(0x68,0x00);
124 //     xmit_data( &IC2_TXmit[0],1);
125
126     stop();
127 }
128 while(1);
129 // stop();
130 xmit_i2c_add(0x68,0x01);
131 rcv_data(&IC2_RCv[4],3);
132 stop();
133 __no_operation();
134
135 /*
136 while(1)
137 {
138     __no_operation();
139     IC2_TXmit[0]=00;
140     PRx_I2CData= &IC2_RCv[0];
141
142
143     UCB0CTLW0 &= ~UCSWRST;
144     UCB0IE |= UCTXIE0; // transmit and NACK interrupt enable
145     UCB0I2CSA = 0x68;
146     TX_I2CByteCtr=1;
147     PTx_I2CData= &IC2_TXmit[0];
148     __no_operation();
149     while (UCB0CTLW0 & UCTXSTP); // Ensure stop condition got sent
150     UCB0CTLW0 |= UCTR | UCTXSTT; // I2C TX, start condition
151     __no_operation();
152     __bis_SR_register(GIE); // Enter LPM0 w/ interrupts
153     while(TX_I2CByteCtr !=0);
154     while (UCB0CTL1 & UCTXSTP); // Ensure stop condition got sent
155     while (UCB0STAT & UCBBUSY); // make sure buss in not busy
156     __no_operation();
157     P1OUT =0x40;
158     __bis_SR_register( GIE);
159
160     __no_operation();
161     i2c_rcv( &IC2_RCv[0],0x68,7);
162 }
163 while(1)
164 {
165     P1OUT |=BIT7;
166     __delay_cycles(15000);
167     P1OUT &=~BIT7;
168     __delay_cycles(15000);

```

patch_main.c

```

169     }
170
171     while(1)
172     {
173         __no_operation();
174 //         __bic_SR_register( GIE);
175 //         put_temp_sensor_reg(&IC2_TXmit[0],0x01);
176 //         get_temp_sensor_data(&IC2_TXmit[0],&IC2_RCv[0]);
177
178     }
179         // default DCODIV as MCLK and SMCLK source
180 */
181         // to activate previously configured port settings
182 // DCOCLK = MCLK and SMCLK source
183
184
185
186
187         // to activate previously configured port settings
188     while(1)
189     {
190         P1OUT |=BIT7;
191         __delay_cycles(15000);
192         P1OUT &=~BIT7;
193         __delay_cycles(15000);
194     }
195     return 0;
196 }
197
198 #pragma vector = USCI_B0_VECTOR
199 __interrupt void USCIB0_ISR(void)
200 {
201     switch(__even_in_range(UCB0IV,USCI_I2C_UCBIT9IFG))
202     {
203         case USCI_NONE: break; // Vector 0: No interrupts break;
204         case USCI_I2C_UCALIFG: break;
205         case USCI_I2C_UCNACKIFG:
206             __no_operation();
207             UCB0CTL1 |= UCTXSTT; //resend start if NACK
208             break; // Vector 4: NACKIFG break;
209         case USCI_I2C_UCSTTIFG:
210             __no_operation();
211             break; // Vector 6: STTIFG break;
212         case USCI_I2C_UCSTPIFG:
213             __no_operation();
214             break; // Vector 8: STPIFG break;
215         case USCI_I2C_UCRXIFG3:
216             __no_operation();
217             break; // Vector 10: RXIFG3 break;
218         case USCI_I2C_UCTXIFG3:
219             __no_operation();
220             break; // Vector 14: TXIFG3 break;
221         case USCI_I2C_UCRXIFG2:
222             __no_operation();
223             break; // Vector 16: RXIFG2 break;
224         case USCI_I2C_UCTXIFG2:

```

patch_main.c

```

225     __no_operation();
226     break; // Vector 18: TXIFG2 break;
227 case USCI_I2C_UCRXIFG1:
228     __no_operation();
229     break; // Vector 20: RXIFG1 break;
230 case USCI_I2C_UCTXIFG1:
231     __no_operation();
232     break; // Vector 22: TXIFG1 break;
233 case USCI_I2C_UCRXIFG0:
234     RX_I2CByteCtr--; // Decrement RX byte counter
235     IC2_on++;
236     if (RX_I2CByteCtr) {
237         *PRx_I2CData++ = UCB0RXBUF; // Move RX data to address
PRxData
238         if (RX_I2CByteCtr == 1) // Only one byte left?
239         {
240             test++;
241             UCB0CTL1 |= UCTXSTP; // Generate I2C stop condition
242             __no_operation();
243         }
244         // UCB0IE &= ~UCRXIE;
//turn off rcv interrupt
245     } else {
246         *PRx_I2CData = UCB0RXBUF; // Move final RX data to
PRxData
247         // __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
248         // *PRx_I2CData = 0x77;
249         testi2c = 0;
250         // UCB0IFG &= ~UCRXIFG;
251     }
252     __no_operation();
253     break; // Vector 24: RXIFG0 break;
254 case USCI_I2C_UCTXIFG0:
255     __no_operation();
256     if (TX_I2CByteCtr) // Check TX byte counter
257     {
258         // UCB0TXBUF = 0x01 ; // Load TX buffer
259         __no_operation();
260         UCB0TXBUF = *PTx_I2CData++; // Load TX buffer
261         TX_I2CByteCtr--; // Decrement TX byte counter
262     }
263 }
264 else
265 {
266     __no_operation();
267     UCB0CTLW0 |= UCTXSTP; // I2C stop condition
268     UCB0IFG &= ~UCTXIFG; // Clear USCI_B0 TX int flag
269 // __bic_SR_register_on_exit(LPM0_bits); // Exit LPM0
270 }
271 break; // Vector 26: TXIFG0 break;
272 case USCI_I2C_UCBCNTIFG:
273     __no_operation();
274     break; // Vector 28: BCNTIFG
275 case USCI_I2C_UCCLTOIFG:
276     __no_operation();
277     break; // Vector 30: clock low timeout
278 case USCI_I2C_UCBIT9IFG:

```

patch_main.c

```
279         __no_operation();
280         break; // Vector 32: 9th bit
281     default: break;
282 }
283 }
284 #pragma vector = TIMER0_B0_VECTOR
285 __interrupt void Timer_B (void)
286
287 {
288     P1OUT ^= BIT6;
289     TB0CCR0 += 50;
290 }
291
292
293
294
295
```