```
1 //*****************************************************************************
2 //*                                                                           *
3 //*                           INITIALIZE PORTS                                *
4 //*                                                                           *
5 //*****************************************************************************
6 #include <msp430.h>
7 #include "stdlib.h"
8 #include "stdio.h"
9 #include "string.h"
10 #ifndef __MSP430FR2311
11 #define __MSP430FR2311
12 #include "MSP430FR2311.h"
13 #endif
14 #ifndef i2c_h
15 #define i2c_h
16 #include "i2c.h"
17 #endif
18 #ifndef init_h
19 #define init_h
20 #include "init.h"
21 #endif
22
23 #define NO_DEBUG
24 #define MCLK_FREQ_MHZ 1
25 #define MCLK_FREQ_8MHZ 8
26
27 void int_cpu_PMM(void)
28 {
29     PMMCTL0_H=0xA5;         //enable write to PMM registers
30
31     PMMCTL0_H=0xA4;           //disable write to PMM registers
32
33 }
34
35 void init_ports(void)        // init ports
36 {
37
38        P1DIR=0x40;                              //set as input but initialize as input
39        P2DIR=00;                                      //1.7 as output
40        // config i2c
41 //       P1SEL0 |=BIT2;
42     //   P1SEL1 &=~BIT2;
43  //      P1SEL0 |=BIT3;
44    //    P1SEL1 &=~BIT3;
45        P1OUT=0x40;
46 }
47 void init_ext_32k(void)
48 {
49
50      P2SEL0 &=~BIT6;
51 //     P2SEL1 |=BIT6 | BIT7;
52      P2SEL1 |=BIT6;
53      P1SEL0 &=BIT7;
54      P2SEL1 |=BIT7;
55
56 }
57
```

```
 58
 59
 60 void Software_Trim(const int mclk_freq)
 61 {
 62     unsigned int oldDcoTap = 0xffff;
 63     unsigned int newDcoTap = 0xffff;
 64     unsigned int newDcoDelta = 0xffff;
 65     unsigned int bestDcoDelta = 0xffff;
 66     unsigned int csCtl0Copy = 0;
 67     unsigned int csCtl1Copy = 0;
 68     unsigned int csCtl0Read = 0;
 69     unsigned int csCtl1Read = 0;
 70     unsigned int dcoFreqTrim = 3;
 71     unsigned char endLoop = 0;
 72
 73     do
 74     {
 75         CSCTL0 = 0x100;                           // DCO Tap = 256
 76         do
 77         {
 78             CSCTL7 &= ~DCOFFG;                    // Clear DCO fault flag
 79         }while (CSCTL7 & DCOFFG);                 // Test DCO fault flag
 80         switch(mclk_freq)
 81          {
 82             case MCLK_FREQ_MHZ:
 83               __delay_cycles((unsigned int)3000 *MCLK_FREQ_MHZ );// Wait FLL lock status
   (FLLUNLOCK) to be stable
 84               break;
 85             case MCLK_FREQ_8MHZ:
 86                __delay_cycles((unsigned int)3000 *MCLK_FREQ_8MHZ );// Wait FLL lock status
   (FLLUNLOCK) to be stable
 87                break;
 88             case MCLK_FREQ_16MHZ:
 89               __delay_cycles((unsigned int)3000 *MCLK_FREQ_16MHZ );// Wait FLL lock status
   (FLLUNLOCK) to be stable
 90               break;
 91            default:
 92               break;
 93         }
 94
 95         while((CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)) && ((CSCTL7 & DCOFFG) == 0));
 96
 97         csCtl0Read = CSCTL0;                     // Read CSCTL0
 98         csCtl1Read = CSCTL1;                     // Read CSCTL1
 99
100         oldDcoTap = newDcoTap;                   // Record DCOTAP value of last time
101         newDcoTap = csCtl0Read & 0x01ff;         // Get DCOTAP value of this time
102         dcoFreqTrim = (csCtl1Read & 0x0070)>>4;// Get DCOFTRIM value
103
104         if(newDcoTap < 256)                      // DCOTAP < 256
105         {
106             newDcoDelta = 256 - newDcoTap;       // Delta value between DCPTAP and 256
107             if((oldDcoTap != 0xffff) && (oldDcoTap >= 256)) // DCOTAP cross 256
108                 endLoop = 1;                     // Stop while loop
109             else
110             {
111                 dcoFreqTrim--;
```

```
112                 CSCTL1 = (csCtl1Read & (~DCOFTRIM)) | (dcoFreqTrim<<4);
113             }
114         }
115         else                                    // DCOTAP >= 256
116         {
117             newDcoDelta = newDcoTap - 256;      // Delta value between DCPTAP and 256
118             if(oldDcoTap < 256)                 // DCOTAP cross 256
119                 endLoop = 1;                    // Stop while loop
120             else
121             {
122                 dcoFreqTrim++;
123                 CSCTL1 = (csCtl1Read & (~DCOFTRIM)) | (dcoFreqTrim<<4);
124             }
125         }
126
127         if(newDcoDelta < bestDcoDelta)          // Record DCOTAP closest to 256
128         {
129             csCtl0Copy = csCtl0Read;
130             csCtl1Copy = csCtl1Read;
131             bestDcoDelta = newDcoDelta;
132         }
133
134     }while(endLoop == 0);                       // Poll until endLoop == 1
135
136     CSCTL0 = csCtl0Copy;                        // Reload locked DCOTAP
137     CSCTL1 = csCtl1Copy;                        // Reload locked DCOFTRIM
138     while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked
139 }
140
141 void change_clock_freq( int mclk_freq)
142 {
143
144   __bis_SR_register(SCG0);                  // Disable the FLL control loop
145
146   __delay_cycles(150000);
147
148   // Loop until XT1,XT2 & DCO fault flag is cleared
149   do
150   {
151      CSCTL7 &= ~(XT1OFFG | DCOFFG);      // Clear XT1 and DCO fault flag
152      SFRIFG1 &= ~OFIFG;                      // Clear fault flags
153   }while (SFRIFG1 & OFIFG);                   // Test oscillator fault flag
154   __bis_SR_register(SCG0);                // Disable FLL
155    CSCTL3 = SELREF__XT1CLK;                // Set XT1 as FLL reference source
156
157      switch(mclk_freq)
158      {
159       case MCLK_FREQ_MHZ:
160            CSCTL1 = DCOFTRIMEN_1 | DCOFTRIM0 | DCOFTRIM1 | DCORSEL_0;// DCOFTRIM=3, DCO Range
    = 1MHz
161            CSCTL2 = FLLD_0 + 30;                    // DCODIV = 1MHz
162            break;
163
164       case MCLK_FREQ_8MHZ:
165            CSCTL1 = DCOFTRIMEN_1 | DCOFTRIM0 | DCOFTRIM1 | DCORSEL_3;// DCOFTRIM=3, DCO Range
    = 8MHz
166            CSCTL2 = FLLD_0 + 243;                   // DCODIV = 8MHz
```

```
167             break;
168         case MCLK_FREQ_16MHZ:
169             CSCTL1 &= ~(DCORSEL_7);                     // Clear DCO frequency select bits
    first
170             CSCTL1 |= DCORSEL_5;                        // Set DCO = 16MHz
171             CSCTL2 = FLLD_0 + 487;                   // DCODIV = 16MHz
172             break;
173         default:
174             break;
175       }
176     __delay_cycles(3);
177     __bic_SR_register(SCG0);                // Enable FLL
178     Software_Trim(mclk_freq);                       // Software Trim to get the best DCOFTRIM
    value
179     CSCTL4 = SELMS__DCOCLKDIV | SELA__XT1CLK;  // set ACLK = XT1CLK = 32768Hz
180                                                 // DCOCLK = MCLK and SMCLK source
181     // Now that osc is running enable fault interrupt
182     SFRIE1 |= OFIE;
183 }
184
```