```
1
2  /*****************************************************************************
3   *
4   * main.c
5   *
6   * Out of Box Demo for the MSP-EXP430FR2311 Light sensing demo with FRAM write
7   *
8   * Main loop, initialization, and interrupt service routines
9   *
10  *   Description: Sample Op-Amp output with ADC10, PWM LED using Timer1 B1
11  *   with ADC value, store current ADC value to FRAM and enter LPM3, wake up
12  *   every ~50ms to start a new ADC conversion.
13  *   NOTE: Running this example for extended periods will impact the FRAM
14  *   endurance.
15  *   ACLK = REFO= ~32768kHz, MCLK = SMCLK = ~1.0MHz
16  *
17  *   Tested On: MSP430FR2311
18  *            -----------------
19  *        /|\|                 |
20  *         | |        P1.2/OA0-|<---External Pad
21  *         --|RST     P1.3/OA0O|--->A3(Internal)
22  *           |        P1.4/OA0+|<---External Pad
23  *           |                 |
24  *           |       P2.0/TB1.1|---> LED
25  *           |                 |
26  *
27  * December 2015
28  * J. Collins
29  *
30  *****************************************************************************/
31
32 #include "driverlib.h"
33
34 //void initPWM(void);
35 void initSleepTimer(uint16_t);
36 //void initSACOA(void);
37 //void initADC10(int x);
38 void initGPIO(void);
39 //void initTRIOA(void);
40 //void initRtc(void);
41
42 #define CALADC_15V_30C *((unsigned int *)0x1A1A)
43 #define CALADC_15V_85C *((unsigned int *)0x1A1C)
44
45 volatile float temp, IntDegF, IntDegC;
46
47
48 /* FRAM Variable that stores ADC results*/
49 //#pragma PERSISTENT(ADC_Conversion_Result)
50 unsigned int ADC_Conversion_Result = 0;
51 #pragma PERSISTENT(ADJ64)
52 unsigned int ADJ64=0xffff, adflag=0, n;          // ADC conversion result
53 #pragma PERSISTENT(ADCTEMP)
54 unsigned int ADCTEMP=256, rtc_cnt=0;
55 #pragma PERSISTENT(RTC_MOD)
56 unsigned int RTC_MOD=0xffff;
57 #pragma PERSISTENT(SMK_TRC)
58 unsigned int SMK_TRC=0xffff;
59 unsigned int CurTemp, k;
60 unsigned int buffer30[32],buf30[32],j=0, i=0, predata, postdata, kdata, ScanCnt=0;
61 unsigned int InitFlag=0, TmCnt=0, ADC32=0, t30, t85, rtccount, rtcdelt, rtccnt1;
62 unsigned int ScanFlag=0, TempFlag=0, AlarmFlag=0, AlarmCnt=0, BatCnt=0, BatFlag=0;
```

```
63 unsigned int BatLowFlag=0, AlarmAct=0, BatVolt, ManuIn, ManuInFlag=0, ManuInCnt=0;
64 const unsigned int AlarmValue=118, ActCnt=3, BatChkCnt=60, BatLowValue=750; //750=2.2v
65 //
66 //
67 int main(void) {
68     //Stop WDT
69     WDT_A_hold(WDT_A_BASE);
70
71
72     initGPIO();
73
74     while(1){
75
76         TmCnt++;
77
78         initSleepTimer(65535);  //
79         __bis_SR_register(LPM3_bits + GIE);
80         PM5CTL0 &= ~LOCKLPM5;
81
82     }
83
84 }
85
86 void initSleepTimer(uint16_t period){
87     //Start timer
88     Timer_B_clearTimerInterrupt(TIMER_B0_BASE);
89
90     Timer_B_initUpModeParam param = {0};
91     param.clockSource = TIMER_B_CLOCKSOURCE_ACLK;
92     param.clockSourceDivider = TIMER_B_CLOCKSOURCE_DIVIDER_1;
93     param.timerPeriod = period;
94     param.timerInterruptEnable_TBIE = TIMER_B_TBIE_INTERRUPT_DISABLE;
95     param.captureCompareInterruptEnable_CCR0_CCIE =
96         TIMER_B_CAPTURECOMPARE_INTERRUPT_ENABLE;
97     param.timerClear = TIMER_B_DO_CLEAR;
98     param.startTimer = true;
99     Timer_B_initUpMode(TIMER_B0_BASE, &param);
100 }
101
102
103 void initGPIO(void){
104
105     P1DIR = 0xFC;   //p1 IO-In
106
107     P1OUT = 0xFF;  //p1 pull-up
108
109     P1REN = 0x1D;  //P1.0 pull high
110     P1SEL0 = 0xe2;
111     P1SEL1 = 0xe2;
112         P2DIR = 0x7f;
113         P2OUT = 0xe0;  //P2.5~6  low active
114         P2REN = 0x80;
115 //        TRI_disable(TRI0_BASE);
116     PM5CTL0 &= ~LOCKLPM5;
117
118 }
119
120 // ADC interrupt service routine
121 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
122 #pragma vector=ADC_VECTOR
123 __interrupt void ADC_ISR(void)
124 #elif defined(__GNUC__)
```

```
125 void __attribute__ ((interrupt(ADC_VECTOR))) ADC_ISR (void)
126 #else
127 #error Compiler not supported!
128 #endif
129 {
130     //Get previous write protection setting
131     uint8_t state = HWREG8(SYS_BASE + OFS_SYSCFG0_L);
132 #ifdef DFWP
133     uint8_t wp = DFWP | PFWP;
134 #else
135     uint8_t wp = PFWP;
136 #endif
137
138 #ifdef FRWPPW
139     HWREG16(SYS_BASE + OFS_SYSCFG0) = FWPW | (state & ~wp);
140 #else
141     HWREG8(SYS_BASE + OFS_SYSCFG0_L) &= ~wp;
142 #endif
143
144     //ADC_Conversion_Result = (ADCMEM0 >> 1);       // Get conversion result and scale for
    lower light level situations (divide by 2)
145     ADC_Conversion_Result = ADCMEM0;
146     //Restore previous write protection setting
147 #ifdef FRWPPW
148     HWREG16(SYS_BASE + OFS_SYSCFG0) = FWPW | state;
149 #else
150     HWREG8(SYS_BASE + OFS_SYSCFG0_L) = state;
151 #endif
152      adflag=0;
153  // __bic_SR_register_on_exit(LPM3_bits);              // Sleep Timer Exits LPM3
154 }
155
156 //******************************************************************************
157 //
158 //This is the Timer B0 interrupt vector service routine.
159 //
160 //******************************************************************************
161 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
162 #pragma vector=TIMER0_B0_VECTOR
163 __interrupt
164 #elif defined(__GNUC__)
165 __attribute__((interrupt(TIMER0_B0_VECTOR)))
166 #endif
167 void TIMER0_B0_ISR(void)
168 {
169     __bic_SR_register_on_exit(LPM3_bits);              // Sleep Timer Exits LPM3
170 }
171 //RTC_VECTOR
172 //******************************************************************************
173 //
174 //This is the RTC interrupt vector service routine.
175 //
176 //******************************************************************************
177
178 #if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
179 #pragma vector=RTC_VECTOR
180 __interrupt
181 #elif defined(__GNUC__)
182 __attribute__((interrupt(RTC_VECTOR)))
183 #endif
184 void RTC_ISR(void)
185 {
```

```
186     RTC_clearInterrupt(RTC_BASE,
187                        RTCIV__RTCIFG);
188     rtc_cnt++;
189     if(rtc_cnt >0)
190     {
191         rtc_cnt = 0;
192         __bic_SR_register_on_exit(LPM3_bits); // Sleep Timer Exits LPM3
193
194     }
195 }
196
197
198
199
```