```
 1 /*****************************************************************************/
 2 /* Files to Include                                                          */
 3 /*****************************************************************************/
 4 #include "mcc_generated_files/interrupt_manager.h"
 5 #include "mcc_generated_files/mcc.h"
 6 #include    "user.h"
 7 #include    "DataTypes.h"
 8 ////////////////////////////////
 9 //      Interrupt Rountine      //
10 ////////////////////////////////
11 void __interrupt() INTERRUPT_InterruptManager (void)    // interrupt handler
12   {
13    uchar    ch;
14    if(INTCONbits.PEIE == 1)
15      {
16       if(PIE1bits.TMR2IE == 1 && PIR1bits.TMR2IF == 1)  // 1mS time-base
17         {
18          PIR1bits.TMR2IF = 0;
19          if(power_f==1)
20             pwm_out();
21          else
22             pwm_op=0;                                   // output=0
23          timebase_f=1;                                  // 設定時間旗號
24          t125ms_c--;
25          if(t125ms_c==0)
26            {
27             t125ms_c=t125ms_cc;                         // 設定125mS計時
28             t125ms_f=1;
29             flash_c++;
30             if(flash_c & 0b0010)
31                f250ms_f=1;
32             else
33                f250ms_f=0;
34             led_enable=1;                               // LED翻新
35            }
36          if(adc_start)
37            {                                          // 開始A/D取樣
38             ADCON0bits.GO_nDONE = 1;                  // A/D開始轉換
39            }
40          else if(adc_sync)
41            {
42             adc_sync=0;
43             adc_start=1;                              // 開始A/D取樣
44             adc_sample_c=adc_sample_cc;               // 設定取樣次數
45             ADCON0 = voltage_ch;                      // 選擇電壓讀取
46            }
```

```
47          }
48        else if(PIE1bits.ADIE == 1 && PIR1bits.ADIF == 1)
49          {
50           PIR1bits.ADIF=0;
51           if(adc_start)
52             {                                        // 開始A/D取樣
53              ch=adc_sample_c & 1;
54              sum[ch]+=((ADRESH << 8) + ADRESL);      // 讀取ADC值
55              adc_sample_c--;
56              if(adc_sample_c==0)
57                {
58                 average[0]=sum[0] >>average_sc;      // 除以8(移位3次)
59                 average[1]=sum[1] >>average_sc;      // 除以8
60                 sum[0]=0;                            // 清除合計暫存器
61                 sum[1]=0;
62                 adc_sample_c=adc_sample_cc;          // 重設取樣次數
63                 adc_start=0;                         // 清除A/D旗號
64                 adc_sync=0;
65                 adc_f=1;                             // A/D全部取樣完成
66                }
67              if(adc_sample_c & 1)                    // Bit0=0-CH0,1-CH1
68                 ADCON0 = current_ch;                 // 選擇電流讀取
69              else
70                 ADCON0 = voltage_ch;                 // 選擇電壓讀取
71             }
72          }
73        else  { }                                     //Unhandled Interrupt
74       }
75     else { }                                         //Unhandled Interrupt
76    }
77 ////////////////////////////
78 //      按鍵服務程式      //
79 ////////////////////////////
80 void keybroad()
81   {
82    if(key_p==0)                    // 按鍵按下?
83      {
84       if(keylock_f==0)
85         {
86          key_count++;
87          if(key_count>longpress)            // 按鍵長按?
88            {
89             if(power_f==0)
90                power_on();                   // 開啟電源
91             else
92                power_off();                  // 關閉電源
```

```
 93             keylock_f=1;                        // 鎖住按鍵
 94             key_count=0;
 95             second_f=0;
 96            }
 97         }
 98      }
 99    else
100      {
101       if(power_f==1 && keylock_f==0)
102         {
103          if(key_count>shortpress)                // 輕按按鍵?
104            {
105             if(second_f==0)                      // 第二次輕按嗎?
106               {
107                key_sc=key_scc;                   //設定第二次輕按等待時間
108                second_f=1;
109               }
110             else
111               {
112                power_select++;
113                if(power_select>2)
114                   power_select=0;
115                power_set();                      // 設定功率參數
116                store_c=store_cc;                 // 重置儲存時間
117                store_f=1;                        //設定儲存旗號
118                second_f=0;
119               }
120             poweroff_c=poweroff_cc;              // 重置關機時間
121             led_fr=0;
122             remind_f=0;
123            }
124          else
125            {
126             if(second_f==1)                      // 第二次輕按旗號=1嗎?
127               {
128                key_sc--;
129                if(key_sc==0)                     // 第二次輕按等待時間結束?
130                   second_f=0;                    // 清除第二次輕按旗號
131               }
132             else
133               {
134                key_sc=key_scc;                   //設定第二次輕按等待時間
135               }
136            }
137         }
138       keylock_f=0;
```

```
139       key_count=0;
140 //      pwm_op=0;                                   // output=0
141       }
142   }
143 ////////////////////////
144 //   Clear A/D register
145 ////////////////////////
146 void adc_clear()
147   {
148    sum[0]=0;                                  // 清除合計暫存器
149    sum[1]=0;
150    adc_sample_c=adc_sample_cc;                // 重設取樣次數
151    adc_start=0;                               // 清除A/D旗號
152    adc_sync=0;
153   }
154 ////////////////////////
155 //      類比讀取      //
156 ////////////////////////
157 void adc_sub()
158   {
159    uchar     ch;
160    if(adc_start)
161      {                                            // 開始A/D取樣
162       ADCON0bits.GO_nDONE = 1;                    // A/D開始轉換
163       __delay_us(50);
164       while (ADCON0bits.GO_nDONE)   { }           // 等待轉換完成
165       ch=adc_sample_c & 1;
166       sum[ch]+=((ADRESH << 8) + ADRESL);          // 讀取ADC值
167       adc_sample_c--;
168       if(adc_sample_c==0)
169         {
170          average[0]=sum[0] >>average_sc;          // 除以8(移位3次)
171          average[1]=sum[1] >>average_sc;          // 除以8
172          sum[0]=0;                                // 清除合計暫存器
173          sum[1]=0;
174          adc_sample_c=adc_sample_cc;              // 重設取樣次數
175          adc_start=0;                             // 清除A/D旗號
176          adc_sync=0;
177          adc_f=1;                                 // A/D全部取樣完成
178         }
179       if(adc_sample_c & 1)                        // Bit0=0-CH0,1-CH1
180          ADCON0 = current_ch;                     // 選擇電流讀取
181       else
182          ADCON0 = voltage_ch;                     // 選擇電壓讀取
183      }
184    else if(adc_sync)
```

```
185       {
186        adc_sync=0;
187        adc_start=1;                              // 開始A/D取樣
188        adc_sample_c=adc_sample_cc;               // 設定取樣次數
189        ADCON0 = voltage_ch;                      // 選擇電壓讀取
190       }
191    }
192 ////////////////////////////////
193 //      記憶體，I/O初始值設定      //
194 ////////////////////////////////
195 void InitApp()                                   // i/o port initize
196    {
197     t125ms_c=t125ms_cc;                         // 設定128mS計時
198     led_r=0;                                    // LED熄滅
199     led_fr=0;
200     led_enable=1;                               //
201     pwm_r=0;                                    // PWM輸出OFF
202     pwm_c=0;
203     pwm_op=0;
204     LATA = 0x00;
205     standby_c=standby_pcc;                      // 待機狀態計數器，LED閃爍用
206     adc_clear();                                // 清除A/D站暫存器
207     store_f=0;
208     pwm_cpc=pwm_cpcc;
209     pwm_h=255;
210     pwm_l=250;
211     keylock_f=0;
212     key_count=0;
213     second_f=0;
214     key_sc=key_scc;
215     CLRWDT();                                       // clear watch-dog timer
216     __delay_ms(150);                                // 延遲150mS
217     CLRWDT();                                       // clear watch-dog timer
218     eeprom_read_nbyte(pcode_adr,&pcode,2);          // 檢查power code
219     if(pcode!=pcode_c)
220       {
221        pcode=pcode_c;
222        power_select=0;
223        eeprom_write_nbyte(pcode_adr,&pcode,2);      //寫入power code
224        DATAEE_WriteByte(sp_adr,power_select);       // 寫入設定資料
225       }
226     power_select=DATAEE_ReadByte(sp_adr);           // 讀取設定資料
227     if(power_select>2)                              // select = 0-2
228        power_select=0;
229     history=power_select;
230     power_f=0;
```

```
231     t50ms_c=t50ms_cc;                                   // 設定50mS counter
232     CLRWDT();                                           // clear watch-dog timer
233    }
234 ////////////////////////////////
235 //      LED燈點亮/熄滅及閃爍      //
236 //      led_r bit0: LED綠燈      //
237 //      led_r bit1: LED紅燈      //
238 ////////////////////////////////
239 void led()
240    {
241     uchar    n;
242     if(led_enable)
243       {
244        n=~led_fr;                          // 閃爍暫存器反向()
245        if(f250ms_f)
246           n |= 0b0011;
247        n &= led_r;
248        if(n & 1)
249           GLED_p=1;                        // 綠燈點亮
250        else
251           GLED_p=0;                        // 綠燈熄滅
252        if(n & 0b10)
253           RLED_p=1;                        // 紅燈點亮
254        else
255           RLED_p=0;                        // 紅燈熄滅
256        led_enable=0;                       // 翻新關閉
257       }
258    }
259 ////////////////////////////
260 //      計算選擇的PWM值      //
261 ////////////////////////////
262 uchar pwm_set()
263    {
264     float n;
265     uslong p;
266     uint ps;
267     CLRWDT();                                  // clear watch-dog timer
268     p = voltage;
269     p *=current;
270     n=factor/p;
271     ps=n;
272     if(ps<10)
273        ps=30;
274     else if(ps>200)
275        ps=200;
276     CLRWDT();                                  // clear watch-dog timer
```

```
277     return(ps);
278   }
279 ////////////////////////////
280 //       PWM 輸出程式        //
281 //       PWM=0-249          //
282 ////////////////////////////
283 #define     pwm_max          199
284 void pwm_out()
285   {
286    pwm_c++;
287    if(pwm_c>pwm_max)
288      {
289       if(pwm_r==0)                  // pwm=0
290          pwm_op=0;                  // output=0
291       else
292          pwm_op=1;                  // output=1
293       pwm_c=0;
294       pwm_f=1;                      // pwm旗號(pwm開始)
295      }
296    else
297      {
298       if(pwm_c < pwm_r)
299         {
300          pwm_op=1;                  // output=1
301          if(pwm_c<10 && pwm_c>5)    // 5mS後
302            {
303             if(adc_start==0)
304               {
305                adc_sync=1;
306                adc_sample_c=adc_sample_cc;
307               }
308            }
309         }
310       else
311          pwm_op=0;                  // output=0
312      }
313   }
314 ////////////////////////
315 //       電源開啟       //
316 ////////////////////////
317 void power_on()
318   {
319    power_f=1;
320    power_set();                     // 設定功率參數
321    second_f=0;
322    t50ms_c=t50ms_cc;                // 設定50mS counter
```

```
323   }
324 ////////////////////////
325 //      電源關閉      //
326 ////////////////////////
327 void power_off()
328   {
329    power_f=0;
330    led_r=0;                          // 關閉LED
331    led_fr=0;
332    pwm_r=0;                          // pwm output=0
333    pwm_op=0;                        // output=0
334    GLED_p=0;                        // LED OFF
335    RLED_p=0;
336    standby_c=standby_cc;
337   }
338 ////////////////////////////
339 //      儲存設定資料      //
340 //      設定於1mS位置     //
341 ////////////////////////////
342 void store()
343   {
344    if(store_f)                       // 儲存嗎?
345      {
346       store_c--;
347       if(store_c==0)
348         {
349          if(power_select!=history)
350            {
351             if(power_select>2)
352                power_select=0;
353             DATAEE_WriteByte(sp_adr,power_select);      // 寫入資料
354             history  = power_select;                    // 儲存到歷史資料
355            }
356          store_f=0;
357         }
358      }
359   }
360 ////////////////////////
361 //      電力設定      //
362 ////////////////////////
363 void power_set()
364   {
365    factor=power_table[power_select];        // 載入係數
366    led_r=led_table[power_select];           // 設定LED
367    led_fr=0;                                // 關閉閃爍
368    poweroff_c=poweroff_cc;                  // 重置關機時間
```

```
369     t125ms_c=t125ms_cc;                     // 設定125mS計時
370     remind_f=0;                             //
371    }
372 ////////////////////////////////
373 //   Write N-Byte to EEPROM   //
374 ////////////////////////////////
375 void eeprom_write_nbyte(uchar adr,uchar *p,uchar lc)
376    {
377     uint addr;
378     addr=adr;
379     addr |=0xf000;
380     while(lc>0)
381       {
382        DATAEE_WriteByte(addr, *p);        // write data to eeprom
383        CLRWDT();
384        addr++;                 // to next address
385        p++;                   // to next data
386        lc--;                  // count-1
387       }
388    }
389 ////////////////////////////////////
390 //   read eeprom n-byte to memory   //
391 ////////////////////////////////////
392 void eeprom_read_nbyte(uchar adr,uchar *p,uchar lc)
393    {
394     uint addr;
395     addr=adr;
396     addr |=0xf000;
397     while(lc>0)
398       {
399        *p=DATAEE_ReadByte(addr);  // Read EEPROM
400        addr++;                   // to next address
401        p++;                     // to next data
402        lc--;                    // count-1
403       }
404    }
```