```
 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)
12   {
13 //===   timer0 200uS   ===
14    if(INTCONbits.TMR0IE == 1 && INTCONbits.TMR0IF == 1)
15      {
16       TMR0 = timer0_tc;                          // reload 200uS
17       INTCONbits.TMR0IF=0;                       // clean timer0 interrupt flag
18       ADCON0bits.GO_nDONE = 1;                   // ADC Start the conversion
19      }
20 //===   port wakeup   ===
21    else if(INTCONbits.IOCIE == 1 && INTCONbits.IOCIF == 1)
22      {
23       INTCONbits.IOCIF=0;
24       if(IOCAFbits.IOCAF4 == 1)             // 物件觸發？
25         {
26          object_f=1;
27          if(Trigger_in)
28             trigin_f=1;                 // 觸發輸入旗號=0
29          else
30             trigin_f=0;                 // 觸發輸入旗號=1
31          object_f=1;
32          goto wakeup;
33         }
34       else if(IOCAFbits.IOCAF5 == 1)    // 板機觸發？
35         {
36          trigin_f=0;                    // 觸發輸入旗號=0
37 wakeup:  ObjectLo_tc=ObjectLo_tcc;      // 設定物件濾波計數器
38          ObjectHi_tc=ObjectHi_tcc;
39          TrigLo_tc=TrigLo_tcc;          // 設定板機濾波計數器
40          TrigHi_tc=TrigHi_tcc;
41          trig_f=0;                      // 清除觸發旗號
42          origin_f=0;                    // 清除原點旗號
43          IOCAFbits.IOCAF4 = 0;
44          IOCAFbits.IOCAF5 = 0;
45         }
46       HallHi_tc=HallHi_tcc;                 // 設定霍爾High濾波計數器
```

```
47        HallLo_tc=HallLo_tcc;                    // 設定霍爾Low濾波計數器
48        hall_f=0;
49        origin_f=0;
50        INTCONbits.IOCIE=0;                      // 喚醒後關閉中斷
51       }
52     else if(INTCONbits.PEIE == 1)
53       {
54        if(PIE1bits.ADIE == 1 && PIR1bits.ADIF == 1)
55          {
56           PIR1bits.ADIF = 0;
57           adc_v=((ADRESH << 8) + ADRESL);                    // 讀取ADC值
58           if(channel_select)
59             {
60              if(motor_out==0)                                //
61                {
62                 volt_sum+=adc_v;
63                 sample_c--;
64                 if(sample_c==0)
65                   {
66                    adc_v=volt_sum/sample_vcc;
67                    if(adc_v>battHi)                         // 電池檢查
68                       battLow_f=0;
69                    else if(adc_v<battLo)
70                       battLow_f=1;                          //  電池電量不足
71                    volt_sum=0;
72                    sample_c=sample_vcc;                     // 電壓取樣次數
73                   }
74                }
75 //             ADCON0 = current_ch;                       //  下個選擇電流讀取
76 //             channel_select=0;
77             }
78           else
79             {
80              if(adc_v>overCurrent)
81                {                                          // 馬達超載
82                 motor_off();                              // 馬達關閉
83                 overload_f=1;                             // 設定超載旗號
84 //              motor_out=0;                              // 馬達關閉
85 //              ADCON0=voltage_ch;                        // 電壓偵測模式
86 //              channel_select=1;                         // 電壓模式
87 //              TMR0 = timer0_tcl;                        // reload 2mS(開始轉換ADC)
88                }
89 //           ADCON0 = voltage_ch;                        // 選擇電壓讀取
90 //           channel_select=1;
91             }
92          }
```

```
 93 //===   Timer2 interrupt 1mS timebase   ===
 94       else if(PIE1bits.TMR2IE == 1 && PIR1bits.TMR2IF == 1)
 95         {
 96          PIR1bits.TMR2IF = 0;
 97          timebase_f=1;
 98         }
 99       else
100         { }
101      }
102    else
103      { }
104 }
105 void adc_initize()
106   {
107    ADCON1 = 0xD3;                        // ADFM right; ADPREF FVR; ADCS FOSC/16;
108    ADCON0 = voltage_ch;                  // 選擇電壓讀取
109    channel_select=1;                     // 電壓偵測Channel
110    ANSELC = 0x06;                        // ANSELx registers
111    ANSELA = 0x00;
112 // ADRESL = 0x00;
113 // ADRESH = 0x00;
114    PIE1bits.ADIE = 1;                    // Enabling ADC interrupt.
115    volt_sum=0;
116    sample_c=sample_vcc;
117   }
118 ////////////////////////////////
119 //     記憶體，I/O初始值設定     //
120 ////////////////////////////////
121 void InitApp()                           // i/o port initize
122   {
123    clean_input_state();
124    Led_p=1;                              //
125    Hall_pout=1;                         // 霍爾電力開啟
126    wakeup_f=1;                          // 喚醒
127    power_tc=power_u25s;                 // 電力開啟計時25秒
128    battLow_f=0;                          // 電池電力正常
129    overload_f=0;                         // 清除過載旗號
130    motor_to_f=0;                         // 清除馬達超時旗號
131    adc_initize();
132    motor_off();
133   }
134 //===   清除輸入狀態   ===
135 void clean_input_state()
136   {
137    ObjectLo_tc=ObjectLo_tcc;            // 設定物件濾波計數器
138    ObjectHi_tc=ObjectHi_tcc;
```

```
139     object_f=0;
140     unlock_f=0;                         // 關閉解鎖
141     TrigLo_tc=TrigLo_tcc;               // 設定板機濾波計數器
142     TrigHi_tc=TrigHi_tcc;
143     trigin_f=0;                         // 觸發狀態=Low
144     trig_f=0;                           // 清除觸發旗號
145     HallHi_tc=HallHi_tcc;               // 設定霍爾High濾波計數器
146     HallLo_tc=HallLo_tcc;               // 設定霍爾Low濾波計數器
147     hall_f=0;
148     origin_f=0;                         // 清除原點
149    }
150 ////////////////////
151 //   Power-Down   //
152 ////////////////////
153 void power_down()
154    {
155 //===  sleep  ===
156     FVRCON = 0;                        // FVR All OFF
157     ADCON0 = 0;
158     LATC=0;                            // Motor,LED,Power OFF
159     IOCAN=0b110000;                    // RA4,5 High -> low interrupt
160     IOCAP=0b000000;
161     IOCAF=0b000000;                    // clear RA change interrupt
162     PIR1=0;                            // clear interrupt flag
163     PIR2=0;
164     PIE1=0b00000000;                   // all interrupt disable
165     PIE2=0b00000000;                   //
166     INTCON=0b10001000;                 // interrupt enable(IOC)
167     CLRWDT();                          // clear watch-dog timer
168     battLow_f=0;                       // 電池電力正常
169     overload_f=0;                      // 清除過載旗號
170     SWDTEN=0;                          // watchdog disable
171 //  __debug_break();
172     SLEEP();                           // Enter Sleep mode
173 //===  wakeup   ===
174     INTCONbits.IOCIE = 0;              // IOC interrupt disable
175     FVRCON = 0b10000101;               // FVREN=1,ADC VREF=1.024V
176     hall_f=0;                          // 清除霍爾旗號
177     origin_f=0;                        // 清除原點旗號
178     power_tc=power_u100ms;             // 喚醒100mS等待
179     wakeup_f=0;                        // 等待喚醒
180     SWDTEN=1;                          // watchdog enable
181     adc_initize();                     // adc initize
182     TMR0 = timer0_tc;                  // TIMER0 200uS
183     T2CON = 0b00000110;                // TIMER2 1mS interrupt
184     PR2 = 0xF9;
```

```
185    TMR2 = 0x00;
186    INTCON=0b11100000;              // GIE=1,PEIE=1,TIMER0IE=1
187    PIE1=0b01000010;                // ADIE=1,TMR2IE=1
188    PIR1=0b00000000;
189    PIE2=0b00000000;
190    PIR2=0b00000000;
191    motor_off();                    // 馬達關閉
192    CLRWDT();                       // clear watch-dog timer
193    motor_to_f=0;                   // 清除馬達超時旗號
194   }
195 //===   輸入取樣與過濾   ===
196 void input_sample()
197   {
198    if(Hall_in)                             // 原點霍爾感測
199      {
200       if(HallHi_tc>0)
201          HallHi_tc--;
202       else
203          hall_f=1;
204       HallLo_tc=HallLo_tcc;                // 設定霍爾Low濾波計數器
205      }
206    else
207      {
208       if(HallLo_tc>0)
209          HallLo_tc--;
210       else
211         {
212          if(hall_f==1)                    // hall輸出 hi--> lo
213            {
214             hall_f=0;
215             origin_f=1;                   // 原點旗號
216            }
217         }
218       HallHi_tc=HallHi_tcc;                // 設定霍爾Low濾波計數器
219      }
220 //===   板機取樣   ===
221    if(Trigger_in)
222      {
223       if(TrigHi_tc>0)
224          TrigHi_tc--;
225       else
226          trigin_f=1;                          // 板機釋放
227       TrigLo_tc=TrigLo_tcc;                   // 重設觸發Low取樣次數
228      }
229    else
230      {
```

```
231       if(TrigLo_tc>0)                          // 觸發Low取樣次數結束嗎?
232          TrigLo_tc--;
233       else
234         {
235          if(wakeup_f==0)
236            {
237             Led_p=1;                            // 電源指示燈點亮
238             Hall_pout=1;                        // 霍爾電力開啟
239             wakeup_f=1;                         // 喚醒
240            }
241          power_tc=power_u25s;                   // 電力開啟計時25秒
242          if(trigin_f==1)                        // 板機high --> low
243            {
244             trigin_f=0;
245             overload_f=0;                       // 清除超載旗號
246             if(unlock_f && battLow_f==0)        // 物件接觸且電池電力充足？
247               {
248                trig_f=1;                        // 板機旗號啟動
249                motor_enable=motor_code;         // 馬達啟動
250                motor_out=1;                     // 馬達關閉
251                motor_timeout=motor_tocc;        // 設定馬達運轉超時
252                origin_f=0;
253                ADCON0=current_ch;               // 電流偵測模式
254                channel_select=0;                // 電流模式
255                TMR0 = timer0_tc;                // reload 200uS
256                ADRESL=0;
257                ADRESH=0;
258               }
259            }
260         }
261       TrigHi_tc=TrigHi_tcc;                    // 重設觸發High取樣次數
262      }
263 //===   物件偵測取樣   ===
264    if(Object_in)
265      {                                          // 物件離開
266       if(ObjectHi_tc>0)
267          ObjectHi_tc--;
268       else
269         {
270          unlock_f=0;                           // unlock失效
271          object_f=1;                           // 物件旗號=1,(無物件接觸)
272         }
273       ObjectLo_tc=ObjectLo_tcc;                // 重設物件Low取樣次數
274      }
275    else
276      {                                          // 物件接觸
```

```
277       if(ObjectLo_tc>0)
278          ObjectLo_tc--;
279       else
280         {
281          if(wakeup_f==0)
282            {
283             Led_p=1;                              // 電源指示燈點亮
284             Hall_pout=1;                          // 霍爾電力開啟
285             wakeup_f=1;                           // 喚醒
286            }
287          power_tc=power_u25s;                     // 電力開啟計時25秒
288          if(object_f)
289            {                                      // 物件信號 1-->0
290             object_f=0;
291             unlock_f=1;                           // unlock開啟
292             overload_f=0;                         // 清除超載旗號
293            }
294         }
295       ObjectHi_tc=ObjectHi_tcc;                  // 重設物件High取樣次數
296      }
297   }
298 //===   250mS
299 void led_indicate(u8 p)
300   {
301    led_tc--;
302    if(led_tc==0)
303      {
304       led_tc=led_tcc;
305       led_shift_r>>=1;
306       if(led_shift_r==0)
307          led_shift_r=p;
308       if(CARRY)
309          Led_p=1;                    // LED點亮
310       else
311          Led_p=0;                    // LED熄滅
312      }
313   }
314 //===   電力下降計時   ===
315 void PowerDown_time()
316   {
317    power_tc--;
318    if(power_tc==0)                          // 電力計時器時間器到達嗎？
319       power_down();                         // 電力下降
320    else
321      {
322       if(wakeup_f==0)
```

```
323         {
324          Led_p=0;                          // LED OFF(喚醒等待)
325          led_tc=1;
326          led_shift_r=0b0001;
327         }
328       else if(battLow_f)
329          led_indicate(0b11110000);          // 電力不足LED慢閃 1sec(on/off)
330       else if(overload_f)
331          led_indicate(0b10);                // 過載LED快閃 0.25sec (on/off)
332       else if(motor_to_f)
333          led_indicate(0b10000);             // 馬達超時短閃 0.25sec on,1sec off
334       else
335         {
336          Led_p=1;                          // LED全亮(系統運作正常)
337          led_tc=1;
338          led_shift_r=0b0001;
339         }
340      }
341   }
342 //===   馬達控制輸出   ===
343 void motor()
344   {
345    if(motor_enable==motor_code)
346      {
347       if(trig_f)                           // 馬達觸發嗎？
348         {
349          if(motor_timeout>0)               // 馬達運轉超時嗎？
350            {
351             if(origin_f)                   // 原點嗎？
352               {
353                motor_to_f=0;               // 清除馬達超時旗號
354                goto motor_dis;
355               }
356             motor_timeout--;
357             if(motor_timeout==0)
358               {
359                motor_to_f=1;               // 設定馬達超時旗號
360                goto motor_dis;
361               }
362             if(unlock_f==0)
363               {
364 motor_dis:     motor_off();                // 馬達關閉
365               }
366            }
367         }
368       }
```

```
369   }
370 //===   馬達關閉   ===
371 void motor_off()
372   {
373    motor_out=0;                          // 馬達關閉
374    motor_enable=0;                       // motor disable
375    trig_f=0;                             // clean trigger flag
376    origin_f=0;                           // 清除原點旗號
377    unlock_f=0;                           // clean unlock flag
378    ADCON0=voltage_ch;                    // 電壓偵測模式
379    channel_select=1;                     // 電壓模式
380    TMR0 = timer0_tc1;                    // reload 2mS(開始轉換ADC)
381    volt_sum=0;
382    sample_c=sample_vcc;
383   }
```