```c

/**
 * 2022.05.10
 * 작성자 : 강건일
 * main.c
 */

#include "F28x_Project.h"

#define SW2_STATE   GpioDataRegs.GPBDAT.bit.GPIO41
#define SW3_STATE   GpioDataRegs.GPBDAT.bit.GPIO45

#define BLU_LED_HIGH        GpioDataRegs.GPASET.bit.GPIO31=1    //
#define BLU_LED_LOW         GpioDataRegs.GPACLEAR.bit.GPIO31=1  //

#define RED_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO34=1    //
#define RED_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO34=1  //

#define GRN_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO44=1    //
#define GRN_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO44=1  //

#define L2_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO38=1
#define L2_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO38=1  //

#define L0_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO37=1    //
#define L0_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO37=1  //

#define L1_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO40=1    //
#define L1_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO40=1  //

#define L3_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO39=1    //
#define L3_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO39=1  //

#define L4_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO43=1    //
#define L4_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO43=1  //

#define L5_LED_HIGH        GpioDataRegs.GPCSET.bit.GPIO67 = 1 //
#define L5_LED_LOW         GpioDataRegs.GPCCLEAR.bit.GPIO67 = 1

#define L6_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO47=1    //
#define L6_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO47=1  //

#define L7_LED_HIGH        GpioDataRegs.GPBSET.bit.GPIO46=1    //
#define L7_LED_LOW         GpioDataRegs.GPBCLEAR.bit.GPIO46=1  //

#define REFERENCE_VDAC  0
#define REFERENCE_VREF  1

#define setup  100

interrupt void cpu_timer1_isr(void);
void InitDacModules(void);
void InitEPwmModules(void);
void InitAdcModule(void);
__interrupt void MainIsr(void);

Uint16  BackTicker;
Uint16  IsrTicker;

 int main(void)
 {
```

```
63    // Step 1. Initialize System Control:
64    // PLL, WatchDog, enable Peripheral Clocks
65    // This example function is found in the F2837xD_SysCtrl.c file.
66
67       InitSysCtrl();
68
69    #ifdef _FLASH
70       InitFlash();
71    #endif
72
73    // Step 2. Initialize GPIO:
74    // This example function is found in the F2837xD_Gpio.c file and
75    // illustrates how to set the GPIO to it's default state.
76
77        InitGpio(); // Skipped for this example
78        EALLOW;
79
80        GpioCtrlRegs.GPADIR.bit.GPIO31 = 1; //Blu
81        GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; //Red
82        GpioCtrlRegs.GPBDIR.bit.GPIO38 = 1; //L2
83        GpioCtrlRegs.GPBDIR.bit.GPIO44 = 1; //Grn
84        GpioCtrlRegs.GPCDIR.bit.GPIO67 = 1; //L5,
85        GpioCtrlRegs.GPBDIR.bit.GPIO37 = 1; //L0,
86        GpioCtrlRegs.GPBDIR.bit.GPIO47 = 1; //L6,
87        GpioCtrlRegs.GPBDIR.bit.GPIO40 = 1; //L1,
88        GpioCtrlRegs.GPBDIR.bit.GPIO39 = 1; //L3,
89        GpioCtrlRegs.GPBDIR.bit.GPIO43 = 1; //L4,
90        GpioCtrlRegs.GPBDIR.bit.GPIO46 = 1; //L7,
91
92        GPIO_SetupPinOptions(31, GPIO_OUTPUT, GPIO_PUSHPULL);
93        GPIO_SetupPinMux(31, GPIO_MUX_CPU1, 0);
94        GPIO_SetupPinOptions(34, GPIO_OUTPUT, GPIO_PUSHPULL);
95        GPIO_SetupPinMux(34, GPIO_MUX_CPU1, 0);
96        GPIO_SetupPinOptions(38, GPIO_OUTPUT, GPIO_PUSHPULL);
97        GPIO_SetupPinMux(38, GPIO_MUX_CPU1, 0);
98        GPIO_SetupPinOptions(44, GPIO_OUTPUT, GPIO_PUSHPULL);
99        GPIO_SetupPinMux(44, GPIO_MUX_CPU1, 0);
100       GPIO_SetupPinOptions(37, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L0
101       GPIO_SetupPinMux(37, GPIO_MUX_CPU1, 0);
102       GPIO_SetupPinOptions(40, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L1
103       GPIO_SetupPinMux(40, GPIO_MUX_CPU1, 0);
104       GPIO_SetupPinOptions(39, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L3
105       GPIO_SetupPinMux(39, GPIO_MUX_CPU1, 0);
106       GPIO_SetupPinOptions(43, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L4
107       GPIO_SetupPinMux(43, GPIO_MUX_CPU1, 0);
108       GPIO_SetupPinOptions(67, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L5
109       GPIO_SetupPinMux(67, GPIO_MUX_CPU1, 0);
110       GPIO_SetupPinOptions(47, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L6
111       GPIO_SetupPinMux(47, GPIO_MUX_CPU1, 0);
112       GPIO_SetupPinOptions(46, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L7
113       GPIO_SetupPinMux(46, GPIO_MUX_CPU1, 0);
114       GPIO_SetupPinOptions(41, GPIO_INPUT, GPIO_QUAL6);  //cpu1 sw2 input
115       GPIO_SetupPinMux(41, GPIO_MUX_CPU1, 0);
116       GPIO_SetupPinOptions(45, GPIO_INPUT, GPIO_QUAL6);  //cpu1 sw3 input
117       GPIO_SetupPinMux(45, GPIO_MUX_CPU1, 0);
118       //
119       EDIS;
120
121       GpioDataRegs.GPADAT.bit.GPIO31 = 1;
122       GpioDataRegs.GPBDAT.bit.GPIO38 = 1; //L2
123       GpioDataRegs.GPBDAT.bit.GPIO44 = 1; // Grn
124       GpioDataRegs.GPBDAT.bit.GPIO34 = 1; // Red
```

```
125
126         GpioDataRegs.GPBDAT.bit.GPIO37 = 1;//L0,
127         GpioDataRegs.GPBDAT.bit.GPIO47 = 1; // L6 ,
128         GpioCtrlRegs.GPCDIR.bit.GPIO67 = 1; // L5,
129         GpioDataRegs.GPBDAT.bit.GPIO40 = 1; //cpu1 L1,
130
131         GpioDataRegs.GPBDAT.bit.GPIO39 = 1; //L3
132         GpioDataRegs.GPBDAT.bit.GPIO43 = 1; //L4
133         GpioDataRegs.GPBDAT.bit.GPIO46 = 1; //L7
134     //
135     // Step 3. Clear all interrupts and initialize PIE vector table:
136     // Disable CPU interrupts
137     //
138         DINT;
139
140     // Initialize the PIE control registers to their default state.
141     // The default state is all PIE interrupts disabled and flags
142     // are cleared.
143     // This function is found in the F2837xD_PieCtrl.c file.
144     //
145         InitPieCtrl();
146     //
147     // Disable CPU interrupts and clear all CPU interrupt flags:
148     //
149         IER = 0x0000;
150         IFR = 0x0000;
151     //
152     // Initialize the PIE vector table with pointers to the shell Interrupt
153     // Service Routines (ISR).
154     // This will populate the entire table, even if the interrupt
155     // is not used in this example.  This is useful for debug purposes.
156     // The shell ISR routines are found in F2837xD_DefaultIsr.c.
157     // This function is found in F2837xD_PieVect.c.
158     //
159         InitPieVectTable();
160
161     // Enable global Interrupts and higher priority real-time debug events:
162     //
163         EINT;  // Enable Global interrupt INTM
164         ERTM;  // Enable Global realtime interrupt DBGM
165
166             EALLOW;
167
168             PieVectTable.ADCB1_INT = &MainIsr;  // Interrupt Service Routine Re-mapping
169
170             EDIS;
171
172             PieCtrlRegs.PIEIER1.bit.INTx2 = 1;  // Enable PIE group 1 interrupt 1 for
    ADCB1_INT
173             IER |= M_INT1;
174
175     // Step 6. IDLE loop. Just sit and loop forever (optional):
176
177             InitEPwmModules();  // Initialize EPWM Modules //210914
178             InitDacModules();   // Initialize DAC Modules
179             InitAdcModule();    // Initialize On-Chip ADC Module
180
181      //  Step 7
182             //  7.1 Initialize S/W modules and Variables //210914
183                 BackTicker = 0;
184                 IsrTicker = 0;
185
```

```
186             //  Step 8
187             //  8.1 Enable Global real-time interrupt DBGM
188             //  8.2 Enable Global Interrupt
189
190                 EINT;   // Enable Global interrupt INTM
191                 ERTM;   // Enable Global real-time interrupt DBGM
192
193             //  Step 9
194             //  9.1 Idle Loop
195
196                 int j ;
197                         for (j = 0; j <  setup ; j++) // 실측정:150ms on/off
198                         {
199
200                                     RED_LED_LOW ;  // LOW 일때 켜짐
201                                     BLU_LED_LOW ;  // LOW 일때 켜짐
202                                     GRN_LED_LOW ;
203                                     L0_LED_LOW ;
204                                     L1_LED_LOW ;
205                                     L2_LED_LOW ;
206                                     L3_LED_LOW ;
207                                     L4_LED_LOW ;
208                                     L5_LED_LOW ;
209                                     L6_LED_LOW ;
210                                     L7_LED_LOW ;
211
212                         DELAY_US(100000); //
213
214                                         RED_LED_HIGH ;
215                                         BLU_LED_HIGH ;
216                                         GRN_LED_HIGH ;
217                                         L0_LED_HIGH  ;
218                                         L1_LED_HIGH  ;
219                                         L2_LED_HIGH  ;
220                                         L3_LED_HIGH  ;
221                                         L4_LED_HIGH  ;
222                                         L5_LED_HIGH  ;
223                                         L6_LED_HIGH  ;
224                                         L7_LED_HIGH  ;
225                         DELAY_US(100000);
226                                                 }
227
228         for(;;)
229         {
230             L4_LED_LOW  ;
231             DELAY_US(10);
232
233            BackTicker++;    //
234
235               if (SW2_STATE) // switch 2 off (HIGH)
236                       {
237
238                        if(SW3_STATE)  // 스위치 모두 OFF,
239                             {
240                                     GRN_LED_LOW ; // GRN LED ON
241                                     RED_LED_HIGH ;
242                                     BLU_LED_HIGH ;
243
244                                     DELAY_US(10);
245
246                                  }  //switch 모두 off, sw3 동작 끝
247
```

```
248                             else  // switch 3  만 on (LOW),BLUE on
249                             {
250                                 GRN_LED_HIGH ; //GRN OFF
251                                 RED_LED_HIGH ; //
252                                 BLU_LED_LOW ; // BLUE ON low 일때 켜짐,
253
254                                 DELAY_US(10);
255                                                 } //blue 의 if 문 끝
256                                                  }  //if sw2 끝
257
258                             else if (SW3_STATE) // switch2 on, switch 3 off (HIGH), RED LED
    켜짐 상태
259
260                               {
261                                             GRN_LED_HIGH ;
262                                             RED_LED_LOW ; // ON low 일때 켜짐, RED led 켜짐
263                                             BLU_LED_HIGH ;
264
265                                             DELAY_US(10);
266
267                                         }  //sw2 Red else if 의끝  END
268
269                         else  // 30spk 실험220325: 스위치 모두 ON,  LED 모두 켜짐
270                             {
271                               GRN_LED_LOW ;
272                               RED_LED_LOW ;
273                               BLU_LED_LOW ;
274
275                               DELAY_US(10);
276
277                                   } //처음 else switch2
278                 L4_LED_HIGH ;
279                 DELAY_US(10);
280
281                           }  //end of for
282                              } // end of main
283 //  Step 10
284 //  10.1 Local Interrupt Service Routines & Functions
285
286    __interrupt void MainIsr(void)
287    {
288
289     IsrTicker++;
290
291     L5_LED_LOW ;  //L5 켜짐, 주기확인용
292
293     //  Clear INT flag and Acknowledge to receive more interrupts
294     AdcbRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;  // Clear INT1 flag
295     PieCtrlRegs.PIEACK.bit.ACK1 = 1;        // Acknowledge this interrupt to receive more
    interrupts from group 1
296
297     DELAY_US(10);
298     L5_LED_HIGH ; //L5 꺼짐,
299     DELAY_US(10);
300
301               }  //Main Isr 끝
302
303  void InitDacModules(void)
304   {
305     EALLOW;
306     DacaRegs.DACCTL.bit.DACREFSEL   = REFERENCE_VREF ;
307     DacbRegs.DACCTL.bit.DACREFSEL   = REFERENCE_VREF;
```

```
308     DaccRegs.DACCTL.bit.DACREFSEL  = REFERENCE_VREF;
309
310     DacaRegs.DACOUTEN.bit.DACOUTEN = 1;     // Enable DAC-A output
311     DacbRegs.DACOUTEN.bit.DACOUTEN = 1;     // Enable DAC-B output
312     DaccRegs.DACOUTEN.bit.DACOUTEN = 1;
313     EDIS;
314
315     DacaRegs.DACVALS.bit.DACVALS = 0; // Clear DAC-A value register
316     DacbRegs.DACVALS.bit.DACVALS = 0; // Clear DAC-B value register
317     DaccRegs.DACVALS.bit.DACVALS = 0;
318     }
319
320    void InitEPwmModules(void)
321     {
322
323     EALLOW;
324     CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;
325     EDIS;
326
327     EALLOW;
328     ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV = 0;
329     EDIS;
330
331     // Setup Counter Mode and Clock
332     EPwm1Regs.TBCTL.bit.CTRMODE = 2;        // Count Up and Down (Symmetric)
333     EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;      // TBCLK = (SYSCLKOUT / EPWMCLKDIV) /
    (HSPCLKDIV * CLKDIV) = 200MHz
334     EPwm1Regs.TBCTL.bit.CLKDIV = 0;
335
336     // Setup Period (Carrier Frequency)
337
338     EPwm1Regs.TBPRD = 5000;     // Set Timer Period, (200MHz/20KHz)/2 = 5,000, sampling
    20khz
339
340     EPwm1Regs.TBCTR = 0;                    // Clear Counter
341
342     // Set up Event Trigger(SOC) with CNT_zero enable for Time-base of EPWM1
343     EPwm1Regs.ETSEL.bit.SOCAEN = 1;         // Enable SOCA
344     EPwm1Regs.ETSEL.bit.SOCASEL = 1;        // Enable CNT_zero event for SOCA
345     EPwm1Regs.ETPS.bit.SOCAPRD = 1;         // Generate SOCA on the 1st event
346     EPwm1Regs.ETCLR.bit.SOCA = 1;           // Clear SOCA flag
347
348     EALLOW;
349     CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
350     EDIS;
351    }
352
353 void InitAdcModule(void)
354  {
355     // Initialize ADC-B module
356     EALLOW;
357     AdcbRegs.ADCCTL2.bit.PRESCALE = 14;     // Set ADCCLK divider to /8 (25MHz @ 200MHz
    SYSCLKOUT)
358     AdcSetMode(ADC_ADCB, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
359     AdcbRegs.ADCCTL1.bit.INTPULSEPOS = 1;   // Set pulse positions to late
360     AdcbRegs.ADCCTL1.bit.ADCPWDNZ = 1;      // Power up the ADC
361     DELAY_US(1000);                         // Delay for 1ms to allow ADC time to power up.
362     EDIS;
363
364     // Initialize ADC-D module,210726 add
365         EALLOW;
366         AdcdRegs.ADCCTL2.bit.PRESCALE = 14;     // Set ADCCLK divider to /8 (25MHz @ 200MHz
```

```
   SYSCLKOUT)
367         AdcSetMode(ADC_ADCD, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
368         AdcdRegs.ADCCTL1.bit.INTPULSEPOS = 1;    // Set pulse positions to late
369         AdcdRegs.ADCCTL1.bit.ADCPWDNZ = 1;       // Power up the ADC
370         DELAY_US(1000);                          // Delay for 1ms to allow ADC time to power
   up. 170727 2ms 개선 안됨
371         EDIS;
372
373     // Configuration ADC module
374     EALLOW;
375     AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;      // B0,VDAC,  ADC-B SOC0 : ADCINB0, Variable
   Voltage Input (Controlled by Potentiometer)
376     AdcbRegs.ADCSOC0CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
   75nsec
377     AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
378
379     AdcbRegs.ADCSOC1CTL.bit.CHSEL = 2;      // B2, Sensor2, ADC-B SOC1 : ADCINB2, Variable
   Voltage Input (Controlled by Potentiometer) //170203 addB2
380     AdcbRegs.ADCSOC1CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
   75nsec //170203 addB2
381     AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA //170203
   addB2
382
383     AdcbRegs.ADCSOC2CTL.bit.CHSEL = 3;      // B3, sensor1, ADC-B SOC2 : ADCINB3, Variable
   Voltage Input (Controlled by Potentiometer) //191206 add B3
384     AdcbRegs.ADCSOC2CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
   75nsec
385     AdcbRegs.ADCSOC2CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
386
387     //210726 add for D_Channel
388     AdcdRegs.ADCSOC3CTL.bit.CHSEL = 0;   // D0, FIR IN, ADC-B SOC3 : ADCIND0, Variable
   Voltage Input (Controlled by Potentiometer) //191206 add B3
389     AdcdRegs.ADCSOC3CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
   75nsec
390     AdcdRegs.ADCSOC3CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
391
392     AdcbRegs.ADCINTSEL1N2.bit.INT1SEL = 0;  // End of SOC0 will set INT1 flag
393     AdcbRegs.ADCINTSEL1N2.bit.INT1E = 1;    // Enable INT1 flag
394     AdcbRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;  // Make sure INT1 flag is cleared
395     EDIS;
396       }
397         // END
398 //
399 // End of file
400 //
401
402
```