```c
//###########################################################################
// Designer : keonil kang

//저자: 강건일
//220620: Audio signal generator for 1khz
// avmart 계제 와 Book
//
#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  30

float FunctionSin(float x ) ;
float A = 2048; //
const float fDesired ;  //

float phase = 0 ;
float pi = 3.141592535 ;
float phaseIncrement ;

const float c1 = 1.5E-5 ;

 Uint32 fs = 20000 ; // step 50us

```

```
 63 interrupt void cpu_timer1_isr(void);
 64
 65 void InitDacModules(void);
 66 void InitEPwmModules(void);
 67 void InitAdcModule(void);
 68 __interrupt void MainIsr(void);
 69
 70 Uint16  BackTicker;
 71 Uint16  IsrTicker;
 72
 73  float  Adc_B0_result; //
 74  float  Adc_B2_result; //
 75  float  Adc_B3_result; //
 76  float  Adc_D0_result; //
 77
 78 void main(void)
 79
 80 {
 81
 82 // Step 1. Initialize System Control:
 83
 84   //  IER = 0x0000; //210914
 85   //  IFR = 0x0000;
 86
 87    InitSysCtrl();
 88 //
 89 #ifdef _FLASH
 90    InitFlash();
 91 #endif
 92 //
 93 // Step 2. Initialize GPIO:
 94 //
 95   InitGpio(); // Skipped for this example
 96    EALLOW;
 97
 98          GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;
 99          GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
100          GpioCtrlRegs.GPBDIR.bit.GPIO38 = 1; //L2
101          GpioCtrlRegs.GPBDIR.bit.GPIO44 = 1;
102          GpioCtrlRegs.GPCDIR.bit.GPIO67 = 1; //L5,
103          GpioCtrlRegs.GPBDIR.bit.GPIO37 = 1; //L0,
104          GpioCtrlRegs.GPBDIR.bit.GPIO47 = 1; //L6,
105          GpioCtrlRegs.GPBDIR.bit.GPIO40 = 1; //L1,
106          GpioCtrlRegs.GPBDIR.bit.GPIO39 = 1; //L3,
107          GpioCtrlRegs.GPBDIR.bit.GPIO43 = 1; //L4,
108          GpioCtrlRegs.GPBDIR.bit.GPIO46 = 1; //L7,
109
110          GPIO_SetupPinOptions(31, GPIO_OUTPUT, GPIO_PUSHPULL);
111          GPIO_SetupPinMux(31, GPIO_MUX_CPU1, 0);
112          GPIO_SetupPinOptions(34, GPIO_OUTPUT, GPIO_PUSHPULL);
113          GPIO_SetupPinMux(34, GPIO_MUX_CPU1, 0);
114          GPIO_SetupPinOptions(38, GPIO_OUTPUT, GPIO_PUSHPULL);
115          GPIO_SetupPinMux(38, GPIO_MUX_CPU1, 0);
116          GPIO_SetupPinOptions(44, GPIO_OUTPUT, GPIO_PUSHPULL);
117          GPIO_SetupPinMux(44, GPIO_MUX_CPU1, 0);
118          GPIO_SetupPinOptions(37, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L0
119          GPIO_SetupPinMux(37, GPIO_MUX_CPU1, 0);
120          GPIO_SetupPinOptions(40, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L1
121          GPIO_SetupPinMux(40, GPIO_MUX_CPU1, 0);
122          GPIO_SetupPinOptions(39, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L3
123          GPIO_SetupPinMux(39, GPIO_MUX_CPU1, 0);
124          GPIO_SetupPinOptions(43, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L4
```

```
125           GPIO_SetupPinMux(43, GPIO_MUX_CPU1, 0);
126           GPIO_SetupPinOptions(67, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L5
127           GPIO_SetupPinMux(67, GPIO_MUX_CPU1, 0);
128           GPIO_SetupPinOptions(47, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L6
129           GPIO_SetupPinMux(47, GPIO_MUX_CPU1, 0);
130           GPIO_SetupPinOptions(46, GPIO_OUTPUT, GPIO_PUSHPULL); //cpu1, L7
131           GPIO_SetupPinMux(46, GPIO_MUX_CPU1, 0);
132           GPIO_SetupPinOptions(41, GPIO_INPUT, GPIO_QUAL6);  //
133           GPIO_SetupPinMux(41, GPIO_MUX_CPU1, 0);
134           GPIO_SetupPinOptions(45, GPIO_INPUT, GPIO_QUAL6);  //
135           GPIO_SetupPinMux(45, GPIO_MUX_CPU1, 0);
136           //
137           EDIS;
138
139           GpioDataRegs.GPADAT.bit.GPIO31 = 1;
140           GpioDataRegs.GPBDAT.bit.GPIO38 = 1; // L2
141           GpioDataRegs.GPBDAT.bit.GPIO44 = 1; //
142           GpioDataRegs.GPBDAT.bit.GPIO34 = 1; //
143
144           GpioDataRegs.GPBDAT.bit.GPIO37 = 1;// L0,
145           GpioDataRegs.GPBDAT.bit.GPIO47 = 1; // L6 ,
146           GpioCtrlRegs.GPCDIR.bit.GPIO67 = 1; // L5,
147           GpioDataRegs.GPBDAT.bit.GPIO40 = 1; // L1,
148
149           GpioDataRegs.GPBDAT.bit.GPIO39 = 1; // L3
150           GpioDataRegs.GPBDAT.bit.GPIO43 = 1; // L4
151           GpioDataRegs.GPBDAT.bit.GPIO46 = 1; // L7
152
153 // Step 3. Clear all interrupts and initialize PIE vector table:
154 // Disable CPU interrupts
155 //
156     DINT;
157
158 //
159     InitPieCtrl();
160 //
161 // Disable CPU interrupts and clear all CPU interrupt flags:
162 //
163     IER = 0x0000;
164     IFR = 0x0000;
165 //
166 //
167     InitPieVectTable();
168 //
169     EINT;  // Enable Global interrupt INTM
170     ERTM;  // Enable Global realtime interrupt DBGM
171
172     EALLOW;
173     PieVectTable.ADCB1_INT = &MainIsr;
174     EDIS;
175
176     PieCtrlRegs.PIEIER1.bit.INTx2 = 1;
177     IER |= M_INT1;
178 //
179 // Step 6. IDLE loop. Just sit and loop forever (optional):
180
181         InitEPwmModules();  // Initialize EPWM Modules
182         InitDacModules();   // Initialize DAC Modules
183         InitAdcModule();    // Initialize On-Chip ADC Module
184
185   //  Step 7
186         //  7.1 Initialize S/W modules and Variables
```

```
187             BackTicker = 0;
188             IsrTicker = 0;
189
190         //  Step 8
191         //  8.1 Enable Global real-time interrupt DBGM
192         //  8.2 Enable Global Interrupt
193
194             EINT;   // Enable Global interrupt INTM
195             ERTM;   // Enable Global real-time interrupt DBGM
196
197         //  Step 9
198         //  9.1 Idle Loop
199
200             int j ;
201                     for (j = 0; j <  setup ; j++) //
202                     {
203
204                                 RED_LED_LOW ;
205                                 BLU_LED_LOW ;
206                                 GRN_LED_LOW ;
207                                 L0_LED_LOW ;
208                                 L1_LED_LOW ;
209                                 L2_LED_LOW ;
210                                 L3_LED_LOW ;
211                                 L4_LED_LOW ;
212                                 L5_LED_LOW ;
213                                 L6_LED_LOW ;
214                                 L7_LED_LOW ;
215
216                     DELAY_US(100000); //
217
218                                         RED_LED_HIGH ;
219                                         BLU_LED_HIGH ;
220                                         GRN_LED_HIGH ;
221                                         L0_LED_HIGH ;
222                                         L1_LED_HIGH ;
223                                         L2_LED_HIGH ;
224                                         L3_LED_HIGH ;
225                                         L4_LED_HIGH;
226                                         L5_LED_HIGH ;
227                                         L6_LED_HIGH ;
228                                         L7_LED_HIGH ;
229                     DELAY_US(100000);
230                                                     }
231
232                     float tmp ; //
233                     int An  ; //
234                     float xn  ;
235                     xn = 0 ; //
236                     An = 4 ; //
237     for(;;) //Endless Loop
238     {
239
240         BackTicker++ ;
241         L4_LED_LOW  ;
242
243            const float fDesired = 3 ;
244            xn = 140 ;
245            phaseIncrement = 2* pi* fDesired / fs ;
246            phase =  phase + ( phaseIncrement + (c1 * xn * 8 ) );
247                 if(phase >= 2*pi)
248                 phase -= 2*pi ;
```

```
249                 phase = (phase > 2.0f) ? (phase -4.0f) : phase ;
250         An = 4 ;
251         tmp =  (A* FunctionSin(phase)) / An ;
252         if (!SW2_STATE)
253         {
254             RED_LED_LOW ;
255             BLU_LED_LOW ;
256             GRN_LED_LOW ;
257         DacaRegs.DACVALS.bit.DACVALS =  tmp + 0x800 ;
258         DacbRegs.DACVALS.bit.DACVALS =  xn ;
259         }
260         else
261         {
262             RED_LED_HIGH ;
263             BLU_LED_HIGH ;
264             GRN_LED_HIGH ;
265         }
266
267           L4_LED_HIGH ;
268                               }  //end of for
269                                 } //end of main
270//  Step 10
271//  10.1 Local Interrupt Service Routines & Functions
272
273   __interrupt void MainIsr(void)
274   {
275
276    IsrTicker++;
277    L5_LED_LOW ;
278
279    if (  DacaRegs.DACVALS.bit.DACVALS > 0x800 ) //
280                {
281                    L3_LED_LOW ;
282                }
283
284                else if (  DacaRegs.DACVALS.bit.DACVALS > 0x7B0 )
285                {
286                    L2_LED_LOW ;
287                }
288
289                else if (  DacaRegs.DACVALS.bit.DACVALS > 0x760 )
290                         {
291                             L1_LED_LOW ;
292                         }
293
294                else if (  DacaRegs.DACVALS.bit.DACVALS > 0x710 )
295                         {
296                             L0_LED_LOW ;
297                         }
298                else if (  DacaRegs.DACVALS.bit.DACVALS > 0x6C0 )
299                                         {
300                                             L7_LED_LOW ;
301                                         }
302                else if (  DacaRegs.DACVALS.bit.DACVALS > 0x670 )
303                                         {
304                                             L6_LED_LOW ;
305                                         }
306                else
307                {
308                   L6_LED_HIGH ;
309                   L7_LED_HIGH ;
310                   L0_LED_HIGH ;
```

```
311                 L1_LED_HIGH ;
312                 L2_LED_HIGH ;
313                 L3_LED_HIGH ;
314             }
315
316
317    AdcbRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;  // Clear INT1 flag
318    PieCtrlRegs.PIEACK.bit.ACK1 = 1;   // Acknowledge this interrupt to receive more
   interrupts from group 1
319
320    L5_LED_HIGH ;
321    DELAY_US(10);
322             }  //Main Isr 끝
323
324  // Sin function using 5th order polynomial
325  float FunctionSin(float x)
326  {
327      const float a1 = 1.57032002 ;
328      const float a3 = - 0.64211317 ;
329      const float a5 = 0.07186085 ;
330      float sinx,x2 ;
331
332      x = ( x > 1.0f) ? (2.0f - x) : x ;
333      x = ( x < -1.0f) ? (-2.0f - x) : x ;
334
335      x2 = x * x ;
336      sinx = ((a5*x2 + a3)*x2 + a1) * x ;
337
338      return (sinx) ;
339           }
340
341 void InitDacModules(void)
342  {
343    EALLOW;
344    DacaRegs.DACCTL.bit.DACREFSEL  = REFERENCE_VREF ;
345    // DAC-A Reference select: ADC VREFHI/VREFLO are the reference voltages
346    DacbRegs.DACCTL.bit.DACREFSEL  = REFERENCE_VREF;
347    // DAC-A Reference select: ADC VREFHI/VREFLO are the reference voltages
348    DaccRegs.DACCTL.bit.DACREFSEL  = REFERENCE_VREF;
349
350    DacaRegs.DACOUTEN.bit.DACOUTEN = 1;     // Enable DAC-A output
351    DacbRegs.DACOUTEN.bit.DACOUTEN = 1;     // Enable DAC-B output
352    DaccRegs.DACOUTEN.bit.DACOUTEN = 1;
353    EDIS;
354
355    DacaRegs.DACVALS.bit.DACVALS = 0;       // Clear DAC-A value register
356    DacbRegs.DACVALS.bit.DACVALS = 0;      // Clear DAC-B value register
357    DaccRegs.DACVALS.bit.DACVALS = 0;
358  }
359
360 void InitEPwmModules(void)
361  {
362
363    EALLOW;
364    CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;
365    EDIS;
366
367    EALLOW;
368    ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV = 0;
369    EDIS;
370
371    // Setup Counter Mode and Clock
```

```
372     EPwm1Regs.TBCTL.bit.CTRMODE = 2; // Count Up and Down (Symmetric)
373     EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;
374     // TBCLK = (SYSCLKOUT / EPWMCLKDIV) / (HSPCLKDIV * CLKDIV) = 200MHz
375     EPwm1Regs.TBCTL.bit.CLKDIV = 0;
376
377     // Setup Period (Carrier Frequency)
378
379     EPwm1Regs.TBPRD = 5000;     // Set Timer Period, (200MHz/20KHz)/2 = 5,000,
380
381     EPwm1Regs.TBCTR = 0;                    // Clear Counter
382
383     // Set up Event Trigger(SOC) with CNT_zero enable for Time-base of EPWM1
384     EPwm1Regs.ETSEL.bit.SOCAEN = 1;         // Enable SOCA
385     EPwm1Regs.ETSEL.bit.SOCASEL = 1;        // Enable CNT_zero event for SOCA
386     EPwm1Regs.ETPS.bit.SOCAPRD = 1;         // Generate SOCA on the 1st event
387     EPwm1Regs.ETCLR.bit.SOCA = 1;           // Clear SOCA flag
388
389     EALLOW;
390     CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
391     EDIS;
392
393    }
394
395 void InitAdcModule(void)
396  {
397
398     EALLOW;
399     AdcbRegs.ADCCTL2.bit.PRESCALE = 14;     // Set ADCCLK divider to /8 (25MHz @ 200MHz
    SYSCLKOUT)
400     AdcSetMode(ADC_ADCB, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
401     AdcbRegs.ADCCTL1.bit.INTPULSEPOS = 1;   // Set pulse positions to late
402     AdcbRegs.ADCCTL1.bit.ADCPWDNZ = 1;      // Power up the ADC
403     DELAY_US(1000);                         // Delay for 1ms to allow ADC time to power up.
404     EDIS;
405
406
407         EALLOW;
408         AdcdRegs.ADCCTL2.bit.PRESCALE = 14;     // Set ADCCLK divider to /8 (25MHz @ 200MHz
    SYSCLKOUT)
409         AdcSetMode(ADC_ADCD, ADC_RESOLUTION_12BIT, ADC_SIGNALMODE_SINGLE);
410         AdcdRegs.ADCCTL1.bit.INTPULSEPOS = 1;   // Set pulse positions to late
411         AdcdRegs.ADCCTL1.bit.ADCPWDNZ = 1;      // Power up the ADC
412         DELAY_US(1000);                         // Delay for 1ms to allow ADC time to power
    up.
413         EDIS;
414
415     // Configuration ADC module
416     EALLOW;
417     AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;
418     // B0,VDAC,ADCRESULT0, SOC0 : ADCINB0, Variable Voltage Input (Controlled by
    Potentiometer)
419     AdcbRegs.ADCSOC0CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
    75nsec
420     AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
421
422     AdcbRegs.ADCSOC1CTL.bit.CHSEL = 2;
423     // B2, ADCRESULT1, ADC-B SOC1 : ADCINB2, Variable Voltage Input (Controlled by
    Potentiometer)
424     AdcbRegs.ADCSOC1CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
    75nsec
425     AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA //170203
    addB2
```

```
426
427     AdcbRegs.ADCSOC2CTL.bit.CHSEL = 3;
428     // B3, ADCRESULT2, ADC-B SOC2 : ADCINB3, Variable Voltage Input (Controlled by
    Potentiometer)
429     AdcbRegs.ADCSOC2CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
    75nsec
430     AdcbRegs.ADCSOC2CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
431
432
433    AdcdRegs.ADCSOC3CTL.bit.CHSEL = 0;
434    // D0, ADCRESULT3, ADC-B SOC3 : ADCIND0, Variable Voltage Input (Controlled by
    Potentiometer)
435    AdcdRegs.ADCSOC3CTL.bit.ACQPS = 14;     // Sample and hold time : (14 + 1) SYSCLK =
    75nsec
436    AdcdRegs.ADCSOC3CTL.bit.TRIGSEL = 5;    // Trigger source : EPWM1, ADCSOCA
437
438
439     AdcbRegs.ADCINTSEL1N2.bit.INT1SEL = 0;  // End of SOC0 will set INT1 flag
440     AdcbRegs.ADCINTSEL1N2.bit.INT1E = 1;    // Enable INT1 flag
441     AdcbRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;  // Make sure INT1 flag is cleared
442     EDIS;
443   }
444                   // END
445 //
446 // End of file
447 //
448
```