```c
#include <c6x.h>
#include <stdio.h>
#include <string.h>
#include <ti/csl/csl_tmr.h>
#include <ti/csl/csl_tmrAux.h>
#include <ti/csl/soc/k2h/src/cslr_device.h>
#include <ti/csl/src/intc/csl_intc.h>
#include <ti/csl/src/intc/csl_intcAux.h>
#include <ti/csl/hw_types.h>
#include <ti/csl/src/intc/cslr_intc.h>

/* INTC Objects */
CSL_IntcObj                     tmrIntcObj;
CSL_IntcContext                 context;
CSL_IntcEventHandlerRecord      EventHandler[30];
CSL_IntcGlobalEnableState        state;
 CSL_IntcParam                  vectId;
   CSL_IntcObj                  intcObj69,intcObj68;
   CSL_IntcHandle               hIntc69,hIntc68;
   CSL_Status                   intStat,intStat1,intStat2;
  CSL_IntcEventHandlerRecord  EventRecord;

 int hightimer_flag=0 ;
 int timer_flag;
 int t1,t2,t3,t1_un;
  Uint32 timer_availcount = 0 ;
  Uint32 timer_unavailcount = 0;
  Uint32 timer0_count = 0;
  Uint32 timer0_count1 =0;
void isr_Timer0();
void initialize_interrupt (void);


int i;
/* defining timer0*/


//#define Timer0_CNTLO0  ((volatile unsigned int *)0x02200010)
//#define Timer0_CNTHI0  ((volatile unsigned int *)0x02200014)
//#define Timer0_PRDLO0  ((volatile unsigned int *)0x02200018)
//#define Timer0_PRDHI0  ((volatile unsigned int *)0x0220001C)
//#define Timer0_TCR     ((volatile unsigned int *)0x02200020)
//#define Timer0_TGCR    ((volatile unsigned int *)0x02200024)
//
///* defining timer1*/
//#define Timer1_CNTLO1  ((volatile unsigned int *)0x02210010)
//#define Timer1_CNTHI1  ((volatile unsigned int *)0x02210014)
//#define Timer1_PRDLO1  ((volatile unsigned int *)0x02210018)
//#define Timer1_PRDHI1  ((volatile unsigned int *)0x0221001C)
//#define Timer1_TCR     ((volatile unsigned int *)0x02210020)
//#define Timer1_TGCR    ((volatile unsigned int *)0x02210024)

/* defining timer8*/
#define Timer8_CNTLO8  ((volatile unsigned int *)0x02280010)
#define Timer8_CNTHI8  ((volatile unsigned int *)0x02280014)
#define Timer8_PRDLO8  ((volatile unsigned int *)0x02280018)
#define Timer8_PRDHI8  ((volatile unsigned int *)0x0228001C)
#define Timer8_TCR     ((volatile unsigned int *)0x02280020)
#define Timer8_TGCR    ((volatile unsigned int *)0x02280024)


```

```
63 #define    CSL_INTC_EVENTID_TINTLO0         (66)
64
65 void  timer0_initilisation(Uint32 countlow,Uint32 counthigh)
66 {
67     *Timer0_CNTLO0 = 0x00000000;
68     *Timer0_CNTHI0 = 0x00000000;
69     *Timer0_PRDLO0 = countlow;
70     *Timer0_PRDHI0 = counthigh;
71     *Timer0_TGCR   = 0x00000003;
72     *Timer0_TCR    = 0x00000088;
73     /**Timer0_PRDLO0=count;
74     *Timer0_PRDHI0=0x00000000;
75     *Timer0_TCR=0x00000088;
76     *Timer0_TGCR=0x00000003;*/
77
78
79 }
80
81
82 void  timer1_initilisation(Uint32 countlow,Uint32 counthigh)
83 {
84     *Timer1_CNTLO1 = 0x00000000;
85     *Timer1_CNTHI1 = 0x00000000;
86     *Timer1_PRDLO1 = countlow;
87     *Timer1_PRDHI1 = counthigh;
88     *Timer1_TGCR   = 0x00000003;
89     *Timer1_TCR    = 0x00000088;
90     /**Timer0_PRDLO0=count;
91     *Timer0_PRDHI0=0x00000000;
92     *Timer0_TCR=0x00000088;
93     *Timer0_TGCR=0x00000003;*/
94 }
95
96
97 void  timer8_initilisation(Uint32 countlow,Uint32 counthigh)
98 {
99     *Timer8_CNTLO8 = 0x00000000;
100    *Timer8_CNTHI8 = 0x00000000;
101    *Timer8_PRDLO8 = countlow;
102    *Timer8_PRDHI8 = counthigh;
103    *Timer8_TGCR   = 0x00000003;
104    *Timer8_TCR    = 0x00000088;
105    /**Timer0_PRDLO0=count;
106    *Timer0_PRDHI0=0x00000000;
107    *Timer0_TCR=0x00000088;
108    *Timer0_TGCR=0x00000003;*/
109
110
111 }
112
113 //void  timer8_disable(Uint32 countlow,Uint32 counthigh)
114 //{
115 //    *Timer8_CNTLO8 = 0x00000000;
116 //    *Timer8_CNTHI8 = 0x00000000;
117 //    *Timer8_PRDLO8 = countlow;
118 //    *Timer8_PRDHI8 = counthigh;
119 //    *Timer8_TGCR   = 0x00000003;
120 //    *Timer8_TCR    = 0x00000048;
121 //    /**Timer0_PRDLO0=count;
122 //    *Timer0_PRDHI0=0x00000000;
123 //    *Timer0_TCR=0x00000088;
124 //    *Timer0_TGCR=0x00000003;*/
```

```
125 //
126 //
127 //}
128 void main(void)
129 {
130     int count=0;
131
132     initialize_interrupt();
133     timer8_initilisation(0x0003000,0x00);
134 //
135     while(1)
136     {
137     timer8_initilisation(0x0003000,0x00);
138 //       initialize_interrupt();
139
140        while(1)
141        {
142           if(hightimer_flag==1)
143               {
144               printf("\n timer complete");
145
146               hightimer_flag=0;
147              // timer8_disable(0x00,0x00);
148               break;
149               }
150
151
152          else
153                 {
154                    printf("\n timer not complete");
155                  }
156        }
157 //       timer0_initilisation(0x0000300,0x00);
158 //       for(count=0;count<1000;count++);
159 //
160 ////          // timer0_initilisation(0x0002710,0x00);
161 ////       while(1)
162 ////
163 ////     {
164 ////         timer0_initilisation(0x0300,0x00);
165 ////         for(count=0;count<1000;count++);
166 ////
167 ////        }
168 //    // printf("timer");
169    }
170 }
171
172
173
174   void initialize_interrupt(void)
175 {
176     CSL_IntcGlobalEnableState   state;
177
178     /* INTC module initialization */
179     context.eventhandlerRecord = EventHandler;
180     context.numEvtEntries      = 10;
181     if (CSL_intcInit(&context) != CSL_SOK)
182         return -1;
183
184     /* Enable NMIs */
185     if (CSL_intcGlobalNmiEnable() != CSL_SOK)
186         return -1;
```

```
187
188     /* Enable global interrupts */
189     if (CSL_intcGlobalEnable(&state) != CSL_SOK)
190         return -1;
191
192     /* INTC has been initialized successfully. */
193
194     vectId = CSL_INTC_VECTID_6;//TIMER0VectId
195
196  /************ handler of timer0**********************/
197
198                 hIntc68 = CSL_intcOpen (&intcObj68,  CSL_INTC_EVENTID_TINTLO0, &vectId ,
    NULL);
199
200
201                 //3. hook isr to event //
202                 intStat2= CSL_intcHookIsr(CSL_INTC_VECTID_6,&isr_Timer0);
203                 if (intStat2 != CSL_SOK)
204                 {
205                     printf("INTR: error while hooking isr for  interrupt .\n");
206                     return;
207                 }
208 //                /* Bind ISR to Interrupt */
209 //                  EventRecord.handler = &isr_Timer0;
210 //                  EventRecord.arg     = (hIntc68);
211 //                  CSL_intcPlugEventHandler(hIntc68, &EventRecord);
212                 // 4. hw control event //
213                 intStat2=CSL_intcHwControl(hIntc68,CSL_INTC_CMD_EVTENABLE,NULL);
214                 if (intStat2 != CSL_SOK)
215                 {
216                     printf("INTR: error while invoking hw control for  interrupt .\n");
217                     return;
218                 }
219             //     CSL_Status  CSL_intcClose ( hIntc68 );
220  //return 0;
221 }
222 /***************** TIMER0 INTERRUPT***********/
223  void isr_Timer0()
224             {
225               printf("In ISR");
226                timer0_count++;
227                hightimer_flag = 1;
228               *(Uint32*)0x00810000 = 4*timer0_count;
229             //  *(Uint32*)0x01800048 = 0x00000004;        // clearing the event 66
230
231             }
232
233
```