```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3  #include <util/delay.h>
4  #include <avr/sleep.h>
5
6  uint8_t alarmEnable = 0;
7  uint8_t start = 0;
8  uint8_t DONE = 0;
9  uint8_t BLINK_NOTICED = 0;
10
11 #define BUTTON_MENU 0
12 #define BUTTON_TEMP_ALARM_NUM 1
13
14 #define DURATION 60 // for 2*8 perf b'd
15 #define menuSelCompleteINTERVAL  4
16 #define shortBuzz 3  // buzzing 3 times
17 #define longBuzz 10  // buzzing 10 times
18
19 #define startPin 4  // for 2*8 perf b'd
20 #define buzzPin 3   // for 2*8 perf b'd
21 #define ledPin 0  // for 2*8 perf b'd
22
23 uint8_t clockCnt;
24 uint8_t secCnt;
25 uint8_t minCnt;
26 uint8_t alarm[3] = {3, 5, 15};
27
28 uint8_t menuCnt=0, tempAlarmCnt=0;
29 uint8_t prevLoop=0, curLoop=0, lapse=0;
30 uint8_t loopCnt=0;
31
32 //-------- FUNCTION PROTOTYPES
33 // Arduino Sketch C doesn't need to declare function prototypes
34 // But to conform with ANSI C, here i follow the standard C rules.
35 void initIO();
36 void startClock(uint8_t );
37 void countButton(uint8_t);
38 void blinkLED(uint8_t );
39 void buzz(uint8_t);
40 void chkAlarm(uint8_t );
41 void chkInterval();
42 void pollingButton();
43
44 volatile unsigned char timer_overflow_count = 0;
45 uint8_t secInt=0, minInt=0;
46 uint8_t intDrivenAlmEn=0, intDrivenAlmPeriod=3;
47
48 //-----------------------------------
49 ISR(TIM0_OVF_vect)
50 {
51     if (++timer_overflow_count > 70)
52     {   // with 1024/256/64 prescaler, a timer overflow occurs 4.6/18/73 times ➲
          per second accordingly.
53         // Toggle Port B pin 4 output state
54         PORTB ^= 1<<ledPin;
55         timer_overflow_count = 0;
```

```
56          secInt++;
57          if (secInt == 60)
58          {
59              minInt++;
60              secInt=0;
61          }
62          if (intDrivenAlmEn == 1)
63              if(minInt == intDrivenAlmPeriod)
64              {
65                  buzz(shortBuzz);
66                  intDrivenAlmEn=0;
67              }
68      }
69  }//ISR(TIM0_OVF_vect)
70
71  //-----------------------------------
72  ISR(PCINT0_vect)
73  {
74      // Toggle PBn output state
75      //PORTB ^= 1<<PB0;
76      //buzz(3);
77      /*
78      DONE = 0;
79      pollingButton();
80      */
81      timer_overflow_count = 0;
82      secInt = 0;
83      minInt = 0;
84      intDrivenAlmEn=1;
85
86  }//ISR(PCINT0_vect)
87
88  //-----------------------------------
89  void initIO()
90  {
91    DDRB &= ~_BV(startPin); //input
92    DDRB |= _BV(buzzPin); //output
93    DDRB |= _BV(ledPin); //output
94  }//initIO
95
96  //-----------------------------------
97  int main()
98  {
99      uint8_t debugMode = 0;
100     initIO();
101
102     //PC interrupt setting
103     // enable PC(Pin Change) interrupt
104     GIMSK |= _BV(PCIE);  //Enable PC interrupt
105     // Enable pin change interrupt for PB3
106     //PCMSK |= _BV(PCINT3);
107     PCMSK |= _BV(startPin);
108
109     //TMR0 interrupt setting
110     // prescale timer to 1/64th the clock rate
111     TCCR0B |= (1<<CS01) | (1<<CS00);
```

```
112     // enable timer overflow interrupt
113     TIMSK0 |=1<<TOIE0;
114
115     sei();
116
117     // Use the Power Down sleep mode
118     //set_sleep_mode(SLEEP_MODE_PWR_DOWN);
119     set_sleep_mode(SLEEP_MODE_IDLE);
120     while(1)
121     {
122       if (debugMode)
123       {
124         blinkLED(3);
125         //chkInterval();
126       }
127       else
128       {
129           // go to sleep and wait for interrupt...
130           sleep_mode();
131       }//else (debugMode == 0)
132     }//while(1)
133
134     return 0;
135 }//main
136
137 //-----------------------------------
138 void startClock(uint8_t alarmMin)
139 {
140   start = 1;
141
142   clockCnt=0;
143   secCnt=0;
144   minCnt=0;
145
146   while (start)
147   {
148     clockCnt++;
149     if (clockCnt % 2 == 0)
150       secCnt++;
151
152     //check minute
153     if (secCnt == 60)
154     {
155       minCnt++;
156       clockCnt = 0;
157       blinkLED(menuCnt);
158       //blinkLED(menuCnt) routine consumes around 1 sec, so we need to ↵
            complement the loss
159       secCnt = 1;
160     }//if (secCnt == 60)
161
162     //========== check Alarm enable status
163     if (alarmEnable == 1)
164     {
165       //digitalWrite(ledPin, HIGH);
166       chkAlarm(alarmMin);
```

```
167       }
168       else
169       {
170         //digitalWrite(ledPin, 0);
171         start = 0;
172       }
173
174       _delay_ms(DURATION);        // _delay_ms in between reads for stability
175     }//while (start)
176   }//startClock
177
178   //-----------------------------------
179   void countButton(uint8_t cate)
180   {
181     uint8_t val;
182
183     //check if startPin pressed to 0
184     val = PINB & _BV(startPin);
185     if (val == 0)
186     {
187       _delay_ms(DURATION); // for debounce
188       switch (cate)
189       {
190         case BUTTON_MENU:
191           menuCnt++;
192           break;
193         case BUTTON_TEMP_ALARM_NUM:
194           tempAlarmCnt++;
195           break;
196       } //switch (cate)
197
198       prevLoop = loopCnt;
199     }//if(val == 0)
200   }//countButton
201
202   //-----------------------------------
203   void blinkLED(uint8_t num)
204   {
205     uint8_t i;
206     for (i=0; i<(2*num); i++)
207     {
208       //digitalWrite(ledPin, HIGH);
209       PORTB ^= _BV(ledPin);
210       _delay_ms(DURATION/2);
211     }
212     BLINK_NOTICED = 1;
213     PORTB &= ~_BV(ledPin);
214     _delay_ms(DURATION*4);
215   }//blinkLED
216
217   //-----------------------------------
218   void buzz(uint8_t times)
219   {
220     const uint8_t buzzInterval = DURATION/2;
221     uint8_t i;
222     for (i=0; i<times; i++)
```

```
223     {
224       //digitalWrite(buzzPin, HIGH);
225       PORTB ^= _BV(buzzPin);
226       _delay_ms(buzzInterval);
227     }
228     PORTB &= ~_BV(buzzPin);
229   }//buzz
230
231   //-------------------------------------
232   void chkAlarm(uint8_t num)
233   {
234     //if the current minute has reached to alarm set
235     if(num == minCnt)
236     {
237       //buzzing
238       buzz(shortBuzz);
239       //disable alarm setting
240       alarmEnable = 0;
241       //turn off the set alarm LED
242       //digitalWrite(ledPin, 0);
243       PORTB &= ~_BV(ledPin);
244       //reset menu selection count
245       menuCnt=0;
246       //prevMS = millis();
247       prevLoop = loopCnt;
248       start = 0;
249       BLINK_NOTICED = 0;
250       DONE = 1;
251     }//if(num == minCnt)
252   }//chkAlarm
253
254   //-------------------------------------
255   void chkInterval()
256   {
257       PORTB ^= _BV(ledPin);
258       _delay_ms(DURATION);
259   }//chkInterval
260
261   //-------------------------------------
262   void pollingButton()
263   {
264       while(!DONE)
265       {
266           loopCnt++;
267           if (menuCnt <= 3)
268           countButton(BUTTON_MENU);
269           else if (menuCnt == 4)
270           countButton(BUTTON_TEMP_ALARM_NUM);
271
272           curLoop = loopCnt;
273           lapse = curLoop - prevLoop;
274
275           if (lapse > menuSelCompleteINTERVAL)
276           {
277           if (menuCnt != 0)
278           {
```

```
279             loopCnt = 0;
280             if (!BLINK_NOTICED)
281             {
282               blinkLED(menuCnt);
283             }//if (!BLINK_NOTICED)
284 
285             switch (menuCnt)
286             {
287               case 1:
288                 alarmEnable = 1;
289                 startClock(alarm[0]);
290                 break;
291               case 2:
292                 alarmEnable = 1;
293                 startClock(alarm[1]);
294                 break;
295               case 3:
296                 alarmEnable = 1;
297                 startClock(alarm[2]);
298                 break;
299             }//switch (menuCnt)
300           }//if (menuCnt != 0)
301 
302           //when menuCnt == 4, buttonCount function counts "tempAlarmCnt"
303           if (tempAlarmCnt != 0)
304           {
305             loopCnt = 0;
306             if (!BLINK_NOTICED)
307             {
308               blinkLED(menuCnt);
309             }//if (!BLINK_NOTICED)
310             //DONE_incUnit = 1;
311             blinkLED(tempAlarmCnt);
312             alarm[0] = tempAlarmCnt;
313             tempAlarmCnt = 0;
314             menuCnt = 1;
315           }//if (tempAlarmCnt != 0)
316           }//if (lapse > menuSelCompleteINTERVAL)
317 
318           if (!start)
319           {
320           //_delay_ms should be short enough
321           //to catch button press by user
322           _delay_ms(DURATION/4);
323           }
324       }//while(!DONE)
325   }//pollingButton
326 
```