

```

1  /*****
2  *****/
3  Target MCU: ATTiny13, 9.6Mhz
4  Internal Clcok
5  Name      :
6  CLKinternal_MenuAlarm_NoMillis_Set
7  tableAlarm.ino
8  Author   : Insoo Kim
9  Date     : March 20, 2015
10 Notes    : Set alarm by pressing
11 button counts
12 Power on Default operation:
13     Press the button
14         once for 3 min alarm,
15         twice for 5 min,
16         3 times for 15 min
17     To set temporary alarm
18 period and change the "once"
19 button temporarily,
20     press the button 4
21 times, like seleting menu
22     after blinking LED 4
23 times to confirm your menu
24 selection
25     press your temporary
26 alarm period which will be
27 assinged to the
28     "once" button.
29     Your temporary setting
30 is maintained only before power
31 recycling
32 *****/
33 *****/
34
35 boolean alarmEnable = false;
36 boolean start = false;
37 boolean BLINK_NOTICED = false;
38
39 #define BUTTON_MENU 0
40 #define BUTTON_TEMP_ALARM_NUM 1
41 //Duration between numbers
42 // #define DURATION 2900 // for
43 170 pin bread b'd
44 // #define buzzPin 4 // for 170
45 pin bread b'd
46 // #define startPin 3 // for 170
47 pin bread b'd
48
49 #define DURATION 370 // for 2*8
50 perf b'd
51 // #define menuSelCompleteINTERVAL
52 (DURATION*2)
53 #define menuSelCompleteINTERVAL 8
54 #define shortBuzz 3 // buzzing 3
55 times
56 #define longBuzz 10 // buzzing 10
57 times
58
59 #define buzzPin 3 // for 2*8
60 perf b'd
61 #define startPin 4 // for 2*8
62 perf b'd
63 #define ledPin 0 // for 2*8 perf
64 b'd
65
66 byte clockCnt;
67 byte secCnt;
68 byte minCnt;
69 byte alarm[3] = {3, 5, 15};
70
71 byte menuCnt=0, tempAlarmCnt=0;
72 byte prevLoop=0, curLoop=0,
73 lapse=0;
74 byte loopCnt=0;
75
76 //----- FUNCTION PROTOTYPES
77 // Arduino Sketch C doesn't need
78 to declare function prototypes
79 // But to conform with ANSI C,
80 here i follow the standard C
81 rules.
82 void startClock(byte );
83 void countButton(byte);
84 void blinkLED(byte );
85 void buzz(byte);
86 void chkAlarm(byte );
87
88 //----- SETUP
89 void setup() {
90     pinMode(startPin, INPUT);
91     pinMode(buzzPin, OUTPUT);
92     pinMode(ledPin, OUTPUT);
93 } //setup
94
95 //----- LOOP
96 void loop()
97 {
98     loopCnt++;
99     if (menuCnt <= 3)
100         countButton(BUTTON_MENU);
101     else if (menuCnt == 4)
102         countButton(BUTTON_TEMP_ALARM_NUM);
103 ;
104
105     curLoop = loopCnt;
106     lapse = curLoop - prevLoop;
107
108     if (lapse >
109 menuSelCompleteINTERVAL)
110     {
111         if (menuCnt != 0)
112         {
113             loopCnt = 0;

```

```

1      if (!BLINK_NOTICED)
2      {
3          blinkLED(menuCnt);
4      } //if (!BLINK_NOTICED)
5
6      switch (menuCnt)
7      {
8          case 1:
9              alarmEnable = true;
10             startClock(alarm[0]);
11             break;
12          case 2:
13              alarmEnable = true;
14              startClock(alarm[1]);
15              break;
16          case 3:
17              alarmEnable = true;
18              startClock(alarm[2]);
19              break;
20      } //switch (menuCnt)
21      } //if (menuCnt != 0)
22
23      //when menuCnt == 4,
24      buttonCount function counts
25      "tempAlarmCnt"
26      if (tempAlarmCnt != 0)
27      {
28          loopCnt = 0;
29          if (!BLINK_NOTICED)
30          {
31              blinkLED(menuCnt);
32          }
33          if (clockCnt % 2 == 0)
34          {
35              secCnt++;
36
37              //check minute
38              if (secCnt == 60)
39              {
40                  minCnt++;
41                  clockCnt = 0;
42                  blinkLED(menuCnt);
43                  //blinkLED(menuCnt) routine
44                  consumes around 1 sec, so we need
45                  to complement the loss
46                  secCnt = 1;
47              }
48
49              //===== check Alarm
50              enable status
51              if (alarmEnable == true)
52              {
53                  //digitalWrite(ledPin,
54                  HIGH);
55                  chkAlarm(alarmMin);
56              }
57              else
58              {
59                  //digitalWrite(ledPin, LOW);
60
61                  } //if (!BLINK_NOTICED)
62                  //DONE incUnit = true;
63                  blinkLED(tempAlarmCnt);
64                  alarm[0] = tempAlarmCnt;
65                  tempAlarmCnt = 0;
66                  menuCnt = 1;
67                  } //if (incUnitCnt != 0)
68                  } //if (lapse >
69                  menuSelCompleteINTERVAL)
70                  if (!start)
71                  {
72                      //delay should be short enough
73                      to catch button press by user
74                      delay(DURATION/4);
75                  }
76                  } //loop
77                  //-----
78                  ---
79                  void startClock(byte alarmMin)
80                  {
81                      start = true;
82
83                      clockCnt=0;
84                      secCnt=0;
85                      minCnt=0;
86
87                      while (start)
88                      {
89                          clockCnt++;
90
91                          start = false;
92                      }
93
94                      delay(DURATION); //
95                      delay in between reads for
96                      stability
97                      //DelayNoBlock(DURATION);
98                      } //while (start)
99                      } //startClock
100                     //-----
101                     ---
102                     void countButton(byte cate)
103                     {
104                         //if pressed, LOW
105                         if (digitalRead(startPin) ==
106                         LOW)
107                         {
108                             delay(200); // for debounce
109                             switch (cate)
110                             {
111                                 case BUTTON_MENU:
112                                     menuCnt++;
113                                     break;
114                                 case BUTTON_TEMP_ALARM_NUM:

```

```
114         tempAlarmCnt++;
115         break;
116     } //switch (cate)
117
118     prevLoop = loopCnt;
119     } //if (digitalRead(startPin) ==
120 LOW)
121 } //countButton
122
123 //-----
124 ---
125 void blinkLED(byte num)
126 {
127     byte i;
128     for (i=0; i<num; i++)
129     {
130         digitalWrite(ledPin, HIGH);
131         delay(DURATION/3);
132         digitalWrite(ledPin, LOW);
133         delay(DURATION/3);
134     }
135     BLINK_NOTICED = true;
136 } //blinkLED
137
138 //-----
139 ---
140 void buzz(byte times)
141 {
142     const byte buzzInterval = 200;
143     byte i;
144     for (i=0; i<times; i++)
145     {
146         digitalWrite(buzzPin, HIGH);
147         delay(buzzInterval);
148         digitalWrite(buzzPin, LOW);
149         delay(buzzInterval);
150     }
151 } //buzz
152
153 //-----
154 ---
155 void chkAlarm(byte num)
156 {
157     //if the current minute has
158 reached to alarm set
159     if(num == minCnt)
160     {
161         //buzzing
162         buzz(shortBuzz);
163         //disable alarm setting
164         alarmEnable = false;
165         //turn off the set alarm LED
166         digitalWrite(ledPin, LOW);
167         //reset menu selection count
168         menuCnt=0;
169         //prevMS = millis();
170         prevLoop = loopCnt;
171         start = false;
172         BLINK_NOTICED = false;
173     }
174 } //chkAlarm
```