```
#include "OC.h"

//---------------------OC_Init0--------------------
// arm output compare 0 for 1 Hz periodic interrupt
// also enables timer to 16 us period
// Input: none
// Output: none
signed short volatile hours;
signed short volatile minutes;
signed short volatile seconds;
void OC_Init0(void){
  seconds = 0;    // debugging monitor
  DDRP |= 0x80;   // debugging monitor
  TIOS |= 0x01;   // activate TC0 as output compare
  TIE  |= 0x01;   // arm OC0
  TSCR1 = 0x80;   // Enable TCNT, 24MHz boot mode, 8MHz in run mode
  TSCR2 = 0x07;   // divide by 128 TCNT prescale, TOI disarm, sets period to 16us
  PACTL = 0;      // timer prescale used for TCNT
  TC0   = TCNT+50;// first interrupt right away
}

//---------------------OC_Init1--------------------
// arm output compare 0 for 800 Hz periodic interrupt
// Input: none
// Output: none
int volatile alarmOn; // whether the alarm is sounding or not
void OC_Init1(void) {
  alarmOn = 0;
  DDRP |= 0x01;   // sets output for speaker
  DDRT |= 0xFC;   // sets output for alarm LED pins
  PTT  |= 0xFC;   // turns off LEDs (negative logic)
  TIOS |= 0x02;   // activate TC1 as output compare
  TIE  |= 0x02;   // arms OC1
  TC1   = TCNT+50;// first interrupt right away
}

interrupt 8 void TOC0handler(void){ // executes at 1 Hz
  TFLG1 = 0x01;         // acknowledge OC0
  seconds++;            // increments seconds
  minutes += seconds/60;// incremements minutes if seconds goes to 60
  seconds %= 60;        // subtracts 60 from seconds if needed
  hours += minutes/60;  // increments hours if minutes goes to 60
  minutes %= 60;        // subtracts 60 from minutes if needed
  hours %= 24;          // subtracts 24 from hours if needed
  TC0 = TC0 + 62500;    // interrupts again after 1 second
  PTP ^= 0x80;          // debugging monitor
}

interrupt 9 void TOC1handler(void) {  // executes at 800 Hz
  static short cycles = 0;  // Number of cycles for LED flashing
  TFLG1 = 0x02;             // acknowledge OC1
  if(alarmOn) {             // if alarm is going off
    cycles++;
    PTP ^= 0x01;            // create square wave to speaker
    if(cycles >= 400) {     // flash LEDs at 2 Hz
      PTT ^= 0xFC;
      cycles = 0;           // resets cycle count
    }
  }
  else {          // when alarm is not sounding
    PTP &= ~0x01; // makes sure output to speaker is low
    PTT |= 0xFC;  // makes sure LEDs are off
  }
```

```
  TC1 = TCNT + 39;
}
```