```
/********************************************************************
Copyright 2010-2017 K.C. Wang, <kwang@eecs.wsu.edu>
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
********************************************************************/

// timer.c file

#define CTL_ENABLE          ( 0x00000080 )
#define CTL_MODE            ( 0x00000040 )
#define CTL_INTR            ( 0x00000020 )
#define CTL_PRESCALE_1      ( 0x00000008 )
#define CTL_PRESCALE_2      ( 0x00000004 )
#define CTL_CTRLEN          ( 0x00000002 )
#define CTL_ONESHOT         ( 0x00000001 )

typedef volatile struct timer{
  u32 LOAD;     // Load Register, TimerXLoad                             0x00
  u32 VALUE;    // Current Value Register, TimerXValue, read only        0x04
  u32 CONTROL;  // Control Register, TimerXControl                       0x08
  u32 INTCLR;   // Interrupt Clear Register, TimerXIntClr, write only    0x0C
  u32 RIS;      // Raw Interrupt Status Register, TimerXRIS, read only   0x10
  u32 MIS;      // Masked Interrupt Status Register,TimerXMIS, read only 0x14
  u32 BGLOAD;   // Background Load Register, TimerXBGLoad                0x18
  u32 *base;
}TIMER;

volatile TIMER *tp[4];  // 4 timers; 2 timers per unit; at 0x00 and 0x20

// timer0 base=0x101E2000; timer1 base=0x101E2020
// timer3 base=0x101E3000; timer1 base=0x101E3020
int kprintf(char *fmt, ...);
//extern int strcpy(char *, char *);
extern int row, col;
int kpchar(char, int, int);
int unkpchar(char, int, int);
int srow, scol;
char clock[16];
char *blanks = "  :  :  ";
int hh, mm, ss;
u32 tick=0;
int oldcolor;


void add_timer(int event)
{
    TQE *newT;

    newT->pid = running->pid;

}

void timer0_handler() {
    int ris,mis, value, load, bload, i;
    ris = tp[0]->RIS;
    mis = tp[0]->MIS;
    value = tp[0]->VALUE;
    load  = tp[0]->LOAD;
    bload=tp[0]->BGLOAD;

    tick++; ss = tick;
    ss %= 60;
    if ((ss % 60)==0){
      mm++;
      if ((mm % 60)==0){
        mm = 0;
        hh++;
      }
    }
    oldcolor = color;
    color = GREEN;
    for (i=0; i<8; i++){
      kpchar(clock[i], 0, 70+i);
    }

    clock[7]='0'+(ss%10); clock[6]='0'+(ss/10);
    clock[4]='0'+(mm%10); clock[3]='0'+(mm/10);
    clock[1]='0'+(hh%10); clock[0]='0'+(hh/10);

    for (i=0; i<8; i++){
      kpchar(clock[i], 0, 70+i);
    }

  timer_clearInterrupt(0);
  color = oldcolor;
  return;
}

void enque_timer(TQE **queue, TQE *time_p)
{
    TQE *t = *queue;
    if(t == 0)
    {
        *queue = time_p;
        time_p->next = t;
        return;
    }
        while(t->next)
            t= t->next;

    time_p->next = t->next;
    t->next = time_p;
}

void deque_timer(TQE **queue)
{
    TQE *t = *queue;
  if (t)
    *queue = t->next;
  return t;
}

void timer_init()
{
  int i;
  kprintf("timer_init() ");

  // set timer base address
  tp[0] = (TIMER *)(0x101E2000);
  tp[1] = (TIMER *)(0x101E2020);
  tp[2] = (TIMER *)(0x101E3000);
  tp[3] = (TIMER *)(0x101E3020);

 // set control counter regs to defaults
  for (i=0; i<4; i++){
    tp[i]->LOAD = 0x0;   // reset
    tp[i]->VALUE= 0xFFFFFFFF;
    tp[i]->RIS  = 0x0;
    tp[i]->MIS  = 0x0;
    tp[i]->LOAD    = 0x100;
    tp[i]->CONTROL = 0x62;  // 011- 0000=|NOTEn|Pe|IntE|-|scal=00|32-bit|0=wrap|
    tp[i]->BGLOAD  = 0xF0000;
  }
  kstrcpy(clock, "00:00:00");
  hh = mm = ss = 0;
}

void timer_start(int n) // timer_start(0), 1, etc.
{
  TIMER *tpr;
  kprintf("timer_start: ");
  tpr = tp[n];
  tpr->CONTROL |= 0x80;  // set enable bit 7
}
int timer_clearInterrupt(int n) // timer_start(0), 1, etc.
{

  TIMER *tpr = tp[n];
  tpr->INTCLR = 0xFFFFFFFF;
}

void timer_stop(int n) // timer_start(0), 1, etc.
{
  TIMER *tptr = tp[n];
  tptr->CONTROL &= 0x7F;  // clear enable bit 7
}
```