```
/*************************************************************
 Target MCU & clock speed: ATmega328P @ 1Mhz internal
 Name    : intrpt.c
 Author  : Insoo Kim (insoo@hotmail.com)
 Created : May 15, 2015
 Updated : May 17, 2015

 Description: Get system compile time & date and display on LCD 2*16
    Button toggling to turn on or off the backlight of LCD

 HEX size[Byte]:

 Ref:
    Donald Weiman    (weimandn@alfredstate.edu)
    http://web.alfredstate.edu/weimandn/programming/lcd/ATmega328/
      LCD_code_gcc_4d.html
 *************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <avr/sleep.h>
#include "externs.h"
#include "defines.h"
#include <util/delay.h>

/*
//Function prototypes of outside this code module
extern uint8_t hour, min, sec;
extern uint8_t year, month, date;
extern uint8_t monthEndDate, day;

extern void lcd_dispRealClock();
extern void lcd_dispAccumulatedTime();
extern void lcd_dispProgInfo();
*/
//Function prototypes of this code module
//void proceedClock();

ISR(PCINT0_vect)
{
    //if (PINB & _BV(PB6))
    // read switch status
    if (tactile_Switch_port & _BV(tactile_Switch_bit))
    {
        //Disable PC(Pin Change) interrupt
        //  to use the button for user menu selection
        //For PCINT7-0, DS: Ch 12.2.8
        //PCMSK0 &= ~_BV(PCINT6);
        PCMSK0 &= ~_BV(tactile_Switch_bit);

        countButton();
```

```
        //Enable PC(Pin Change) interrupt
        //For PCINT7-0, DS: Ch 12.2.8
        //PCMSK0 |= _BV(PCINT6);
        PCMSK0 |= _BV(tactile_Switch_bit);
    }
    _delay_ms(200); // for debounce
}//ISR(PCINT0_vect)

//-----------------------------------
ISR(WDT_vect)
{
    //PORTB |= _BV(PB4);
    proceedClock();
    lcd_dispWords(wd%MAXWORDCNT);
    wd++;
    //lcd_dispRealClock();
    //PORTB &= ~_BV(PB4);
}//ISR(WDT_vect)

//-----------------------------------
void WDT_Init(void)
{
    //disable interrupts
    cli();
    //MCUSR = 0;

    wdt_disable();
    //set up WDT interrupt
    //WDTCSR = _BV(WDCE)|_BV(WDE);
    //Start watchdog timer with 4s prescaller
    //WDTCSR |= _BV(WDIE)|_BV(WDE)|_BV(WDP3);
    //WDTCSR |= _BV(WDIE)| _BV(WDP3);
    //WDTCSR |= _BV(WDIE)|_BV(WDE)|_BV(WDP2) | _BV(WDP1); // 1s

    //reset watchdog
    wdt_reset();
    wdt_enable(WDTO_500MS);
    //Enable global interrupts
    sei();
}//WDT_Init

void initINT()
{
    cli();
    //**** PC(Pin Change) interrupt setting
    //ref) https://gist.github.com/Wollw/2598827
    // enable PC INT
    //GIMSK |= _BV(PCIE);  //Enable PC interrupt
    // Enable pin change interrupt for PB3
    //PCMSK |= _BV(PCINT3);
    //PCMSK |= _BV(startPin);
    /**
     * Pin Change Interrupt enable on PCINT6 (PB6)
     */
    //For PCINT7-0, DS: Ch 12.2.4
    PCICR |= _BV(PCIE0);
```

```
    //For PCINT7-0, DS: Ch 12.2.8
    //PCMSK0 |= _BV(PCINT4);
    PCMSK0 |= _BV(tactile_Switch_bit);



    //**** WDT interrupt setting
    check_wdt();
    setup_wdt();
    //set prescale timer
    //DS: ch10.9.2 table10.2
    //set up WDT interrupt
    //WDTCSR |= _BV(WDCE)| _BV(WDE);
    //WDTCSR = _BV(WDIE) | _BV(WDP3) | _BV(WDP0); // 8s
    //WDTCSR = _BV(WDIE) | _BV(WDP3); // 4s
    //WDTCSR = _BV(WDIE) | _BV(WDP2) | _BV(WDP0); // 0.5s


    //wdt_reset();
    //enalbe global interrupt
    sei();


}//initINT

/*
Utmost(!) help to get the WDT of ATmega328p work
http://elegantcircuits.com/2014/10/14/introduction-to-the-avr-watchdog-timer/
*/
void check_wdt(void)
{
    // If a reset was caused by the Watchdog Timer...
    if(MCUSR & _BV(WDRF))
    {
        // Clear the WDT reset flag
        MCUSR &= ~_BV(WDRF);
        // Enable the WD Change Bit
        WDTCSR |= (_BV(WDCE) | _BV(WDE));
        // Disable the WDT
        WDTCSR = 0x00;
    }
}//check_wdt

void setup_wdt(void){
// Set up Watch Dog Timer for Inactivity
    // Enable the WD Change Bit
    // Enable WDT interrupt
    WDTCSR |= _BV(WDCE) | _BV(WDE);

    // Set Timeout to ~8 seconds
    WDTCSR = _BV(WDIE) | _BV(WDP3) | _BV(WDP0); // 8s
    // Set Timeout to ~1 seconds
    //WDTCSR =   _BV(WDIE) |  _BV(WDP2) | _BV(WDP1);
    // Set Timeout to ~500 ms
    //WDTCSR =   _BV(WDIE) | _BV(WDP2);
}//setup_wdt
```

```
void init_devices(void){
    //stop errant interrupts until set up
    cli(); //disable all interrupts

    //timer0_init();

    MCUCR = 0x00;
    EICRA = 0x00; //extended ext ints
    EIMSK = 0x00;

    TIMSK0 = 0x02; //timer 0 interrupt sources

    PRR = 0x00; //power controller
    sei(); //re-enable interrupts
    //all peripherals are now initialized
}

void proceedClock()
{
    //WDT interrupt occurs every 8 seconds
    sec += 8;
    accumulatedSec += 8;

    //real-time fetched from compiled time constant __TIME__
    //calculate minutes
    if (sec >= 60)
    {
        sec%=60;
        min++;
        sec += 2;
    }
    //system run-time acculated time
    //calculate minutes
    if (accumulatedSec >= 60)
    {
        accumulatedSec%=60;
        accumulatedMin++;
        accumulatedSec += 2;
    }

    //real-time fetched from compiled time constant __TIME__
    //calculate hours
    if (min >= 60)
    {
        min%=60;
        hour++;
    }
    //system run-time acculated time
    //calculate hours
    //if ((accumulatedMin == 60) && (hourlyAdjusted == 0))
    if (accumulatedMin == 60)
    {
        // adjust time by experiments
        //hourlyAdjusted = 1;
        sec -= 20;
```

```
        accumulatedMin = 0;
        accumulatedHour++;
    }
    /*
    if (accumulatedMin == 59)
        hourlyAdjusted = 0;
    */
    if (hour >= 24)
    {
        hour=0;
        date++;
        day++;
        if (day >= 7)
            day %= 7;
    }
    switch (month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            monthEndDate = 31;
            break;
        case 2:
            monthEndDate = 28;
            break;
        default:
            monthEndDate = 30;
    }//switch (month)

    if (date > monthEndDate)
    {
        date=1;
        month++;
    }

    if (month > 12)
    {
        month=1;
        year++;
    }
}//proceedClock
```