```c
/***************************************************************
 Target MCU & clock speed: ATtiny13A @ 1.2Mhz internal
 Name    : main.c
 C modules of this project, ISR:
main.c
Custom Headers:
    Nothing
 Author  : Insoo Kim (insoo@hotmail.com)
 Created : May 15, 2015
 Updated : Aug 20, 2018 (On Atmel Studio 7)

 Description:
    ATtiny13A controls power up or down to ESP-01 module by 2n2222 NPN transistor.
    ATtiny13A sleeps in most of operation time and wake up periodically to measure temperature and humidity by DHT22 attached to ESP-01.

 HEX size[Byte]: 348 out of 1024

How to upload to the target MCU
<For Windows Atmel Studio>
Slect Tool ? USBtiny (USBtiny memu should be configured in the external tool memu)

<For CMD window or DOS prompt>
cd " C:₩Users₩insoo₩Documents₩GitHub₩ATmelStudio₩ATtiny13A₩ClockGen₩ISR₩Debug "
avrdude -c usbtiny -P usb -p attiny13 -U flash:w:ISR.hex:i

 Ref:
*****************************************************************/

#include <avr/interrupt.h>
#include <avr/sleep.h>

//#include <util/delay.h>
#include <avr/eeprom.h>

//#define UNIT_DELAY_WDT 4 //<=== TEST VALUE in development;
#define UNIT_DELAY_WDT 8 //<=== SELECTED VALUE in production; WDT period in seconds

// # of UNIT_DELAY_WDT, Max 253
//#define SET_DELAY_UNIT 4  // <=== TEST VALUE in development
//#define SET_DELAY_UNIT 15  // 2 min when UNIT_DELAY_WDT is 8
//#define SET_DELAY_UNIT 150 // 20 min when UNIT_DELAY_WDT is 8
#define SET_DELAY_UNIT 225 //<=== SELECTED VALUE in production; 30 min when UNIT_DELAY_WDT is 8
//#define SET_DELAY_UNIT 253 // (34 min - 8 sec) when UNIT_DELAY_WDT is 8

#define WDTTICK_CTR_ADDR    0
#define WDTTICK_30MIN_ADDR  1

#define WAKEUP_PERIOD   2 // 2:one hour, 4:two hours,
                          // when SET_DELAY_UNIT is 225 of ATtiny13a at 1.2Mhz

```

```c
#define NPN_TR_PORT PB4

uint8_t WDTtick = 0;
uint8_t WDTtick30min = 0;

ISR(WDT_vect)
{
    // --------- HOW MANY WDT HAS OCCURED ? ---------------------------------
    // On every watch dog timer interrupt,
    //  get the WDTtick counter value every UNIT_DELAY_WDT sec
    //  from the designated EEPROM address
    WDTtick30min = eeprom_read_byte((uint8_t*)WDTTICK_30MIN_ADDR);
    WDTtick = eeprom_read_byte((uint8_t*)WDTTICK_CTR_ADDR);

    // On every one hour
    if (WDTtick30min >= WAKEUP_PERIOD)
    {
        // --------- DO TO PROPER ACTION TO WDT TICK COUNT ----------------------
        // When the accumulated WDT reaches every SET_DELAY_UNIT, turn on ESP-01
        if (WDTtick == 0)
        {
            // Give logic HIGH to port 4 to turn ON NPN transistor(2n2222),
            //  so let the GND of ESP-01 module CONNECT to system GND.
            // This will power ON ESP-01 and measure temperature & humidity via DHT22
            PORTB = 1<<NPN_TR_PORT; //turn on GND of ESP-01
        }//if (WDTtick == 0)
        else if (WDTtick == 1)
        {
            // Give logic LOW to port 4 to turn OFF NPN transistor(2n2222),
            //  so let the GND of ESP-01 module DISCONNECT to system GND.
            // This will power OFF ESP-01 and don't measure temperature & humidity via DHT22
            PORTB = (0<<NPN_TR_PORT); //turn off GND of ESP-01
        }//else if (WDTtick == 1)
        else if (WDTtick >= SET_DELAY_UNIT)
        {
            // Reset WDT counter value of the designated address in the EEPROM of ATtiny13A
            eeprom_update_byte((uint8_t*)WDTTICK_CTR_ADDR, 0);

            //Reset WDT Half-hour counter value of the designated address in the EEPROM of ATtiny13A
            eeprom_update_byte((uint8_t*)WDTTICK_30MIN_ADDR, 0);
        }//else if (WDTtick >= SET_DELAY_UNIT)
    }

    // On every half-hour except last half-hour
    if ((WDTtick >= SET_DELAY_UNIT) && (WDTtick30min < WAKEUP_PERIOD))
    {
        // Reset WDT counter value of the designated address in the EEPROM of ATtiny13A
        eeprom_update_byte((uint8_t*)WDTTICK_CTR_ADDR, 0);

        //Increase WDT Half-hour counter value of the designated address in the EEPROM of ATtiny13A
        eeprom_update_byte((uint8_t*)WDTTICK_30MIN_ADDR, ++WDTtick30min);
    }//else if (WDTtick >= SET_DELAY_UNIT)

    // --------- INCREASE WDT TICK COUNT ----------------------
    // increase WDTtick every UNIT_DELAY_WDT sec
    //  and update it at the designated EEPROM address
    if (WDTtick < SET_DELAY_UNIT)
        eeprom_update_byte((uint8_t*)WDTTICK_CTR_ADDR, ++WDTtick);

}//ISR(WDT_vect)

int main(void) {
    // Set up NPN_TR_PORT & PNP_TR_PORT mode to output
    //DDRB = 1<<DDB4;
    DDRB = (1<<NPN_TR_PORT);

    //DDRB = (1<<NPN_TR_PORT) | (1<<PNP_TR_PORT);

    //PORTB = 0<<NPN_TR_PORT;
    // temporarily prescale timer to UNIT_DELAY_WDT seconds so we can measure current
    switch (UNIT_DELAY_WDT)
    {
        case 4:
            WDTCR |= (1<<WDP3); // 4s
            break;
        case 8:
            WDTCR |= (1<<WDP3) | (1<<WDP0); // 8s
            break;
        default:
            WDTCR |= (1<<WDP3) | (1<<WDP0); // 8s
    }
    // (1<<WDP2) | (1<<WDP0);

    // Enable watchdog timer interrupts
    WDTCR |= (1<<WDTIE);

    sei(); // Enable global interrupts

    // Reset the WDTtick at the designated EEPROM address
    //eeprom_update_byte((uint8_t*)WDTTICK_CTR_ADDR, 0);

    // Use the Power Down sleep mode
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);

    for (;;) {
        sleep_mode();   // go to sleep and wait for interrupt...
    }
}//main
```