Blink_using_WDT.c

```
/*****************************************************************************
 *  MSP430 Blink the LED Demo -  Toggle P1.0
 *
 *  This version uses the Watchdog Timer ("WDT") in interval mode.
 *  The default system clock is about 1.1MHz.  The WDT is configured to be driven by this
 *  clock and produce an interrupt every 8K ticks ==> interrupt interval = 8K/1.1Mhz ~ 7.5ms
 *  When the WDT interrupt occurs, the WDT interrupt handler is called.
 *  This handler decrements a global counter variable (blink_counter).  If the counter
 *  reaches 0, then the handler toggles the LED and reinitializes blink_counter to the
 *  value of another global variable (blink_interval).
 *  The result is that the LED toggles at intervals of ~7.5ms * blink_interval.
 *  (the program starts with blink_interval=67, but this value can be changed in the debugger
 *
 *  NOTE: Between interrupts the CPU is OFF!
 *****************************************************************************************/

#include <msp430g2553.h>

volatile unsigned int blink_interval;  // number of WDT interrupts per blink of LED
volatile unsigned int blink_counter;   // down counter for interrupt handler
```

```
int main(void) {
       // setup the watchdog timer as an interval timer
       WDTCTL =(WDTPW + // (bits 15-8) password
                        // bit 7=0 => watchdog timer on
                        // bit 6=0 => NMI on rising edge (not used here)
                        // bit 5=0 => RST/NMI pin does a reset (not used here)
                WDTTMSEL + // (bit 4) select interval timer mode
                WDTCNTCL +  // (bit 3) clear watchdog timer counter
                                0 // bit 2=0 => SMCLK is the source
                                +1 // bits 1-0 = 01 => source/8K
                         );
       IE1 |= WDTIE;          // enable the WDT interrupt (in the system interrupt register IE1)

       P1DIR |= 0x01;                                  // Set P1.0 to output direction

       // initialize the state variables
       blink_interval=67;                // the number of WDT interrupts per toggle of P1.0
       blink_counter=blink_interval;     // initialize the counter

       _bis_SR_register(GIE+LPM0_bits);  // enable interrupts and also turn the CPU off!
}
```

```
// ===== Watchdog Timer Interrupt Handler =====
// This event handler is called to handle the watchdog timer interrupt,
//    which is occurring regularly at intervals of about 8K/1.1MHz ~= 7.4ms.
```

```
interrupt void WDT_interval_handler(){
  if (--blink_counter==0){          // decrement the counter and act only if it has reached 0
       P1OUT ^= 1;                   // toggle LED on P1.0
       blink_counter=blink_interval; // reset the down counter
  }
}
```

```
// DECLARE function WDT_interval_handler as handler for interrupt 10
// using a macro defined in the msp430g2553.h include file
ISR_VECTOR(WDT_interval_handler, ".int10")
```

DoubleBlink.c

```
/*****************************************************************************
 *  MSP430 Blink both LEDs Demo -  Toggle P1.0 and P1.6 at different rates
 *
 *  This version uses the Watchdog Timer ("WDT") in interval mode.
 *  The default system clock is about 1.1MHz.  The WDT is configured to be driven by this
 *  clock and produce an interrupt every 8K ticks ==> interrupt interval = 8K/1.1Mhz ~ 7.5ms
 *  When the WDT interrupt occurs, the WDT interrupt handler is called.
 *
 *  The red (P1.0) and green (P1.6) LED's are blinked by the WDT handler which keeps track of
 *  down counters for each LED.  The action of the WDT handler is to decrement each counter
 *  and when it reaches 0, to toggle the corresponding LED.
 *
 *  NOTE: Between interrupts the CPU is OFF!
 *****************************************************************************************/

#include <msp430g2553.h>

// define constants corresponding to the bits for the two LED's
#define RED 0x01
#define GREEN 0x40

// Global variables (arrays with one element for each LED)
unsigned char mask[]={RED,GREEN};     //bit masks for the two LED pins P1.0 and P.6

volatile unsigned int blink_interval[2]={100,30};// number of WDT interrupts per blink of the
LED's
volatile unsigned int blink_counter[2];   // down counter for interrupt handler

int main(void) {
       int led; // loop variable for the two LED's: led=0 is Red,led=1 is green
       // setup the watchdog timer as an interval timer
       WDTCTL =(WDTPW + // (bits 15-8) password
                        // bit 7=0 => watchdog timer on
                        // bit 6=0 => NMI on rising edge (not used here)
                        // bit 5=0 => RST/NMI pin does a reset (not used here)
                WDTTMSEL + // (bit 4) select interval timer mode
                WDTCNTCL +  // (bit 3) clear watchdog timer counter
                                0 // bit 2=0 => SMCLK is the source
                                +1 // bits 1-0 = 01 => source/8K
                         );
       IE1 |= WDTIE;          // enable the WDT interrupt (in the system interrupt register IE1)

       // initialize the output direction and initial state for eacj of the led's
       for(led=0;led<2;++led){
         P1DIR |= mask[led];   // set the current LED P1 pin to output direction
         P1OUT &= ~ mask[led]; // turn the current LED off (by clearing corresponding P1OUT bit)
         blink_counter[led]=blink_interval[led]; // initialize the LED counter
       }
       _bis_SR_register(GIE+LPM0_bits);  // enable interrupts and also turn the CPU off!
}

// ===== Watchdog Timer Interrupt Handler =====
// This event handler is called to handle the watchdog timer interrupt,
//    which is occurring regularly at intervals of about 8K/1.1MHz ~= 7.4ms.

interrupt void WDT_interval_handler(){
     int led;
     for (led=0;led<2;++led){
       if (--blink_counter[led]==0){ // decrement the counter and act only if it has reached 0
                P1OUT ^= mask[led];     // toggle current LED
                blink_counter[led]=blink_interval[led]; // reset the counter
          }
     }
}
// DECLARE function WDT_interval_handler as handler for interrupt 10
// using a macro defined in the msp430g2553.h include file
ISR_VECTOR(WDT_interval_handler, ".int10")
```