set blinking LED1

if ((S1) && level < 5)

level++

if (S2) && level > 0)

level–

/\*------------------------------------------------------------------------------

\* File: Lab7\_D6.c (CPE 325 Lab7 Demo code)

\* Function: Blinking LED1 & LED2 using Timer\_B with interrupts (MPS430FG4618)

\* Description: In this C program, Timer\_B is configured for up/down mode with

\* ACLK source and interrupts for channel 0 and channel 1 are

\* enabled. In up/down mode timer TB counts the value from 0 up to

\* value stored in TB0CCR0 and then counts back to 0. The interrupt

\* for TB0 is generated when the counter reaches value in TB0CCR0.

\* The interrupt TB1 is generated whenever the counter reaches value

\* in TB0CCR1. Thus, TB1 gets two interrupts while counting upwards

\* and counting downwards. This simulates a PWM control - adjusting

\* the TB1 and TB0 CCR register values adjusts the duty cycle of the

\* PWM signal.

\* Clocks: ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = DCO = default (~1MHz)

\* An external watch crystal between XIN & XOUT is required for ACLK

\*

\* MSP430xG461x

\* -----------------

\* /|\| XIN|-

\* | | | 32kHz

\* --|RST XOUT|-

\* | |

\* | P2.1|--> LED2(YELLOW)

\* | P2.2|--> LED1(GREEN)

\* Input: None

\* Output: LED1 blinks at 1.64Hz with 20-80% duty cycle and LED2 blinks at

\* 0.82Hz with 50-50% duty cycle.

\* Author: Aleksandar Milenkovic, milenkovic@computer.org

\*------------------------------------------------------------------------------\*/

#include <msp430fg4618.h>

#define LED1\_ON P2OUT |= BIT2

#define LED1\_OFF P2OUT &= ~BIT2

#define LED2\_ON P2OUT |= BIT1

#define LED2\_OFF P2OUT &= ~BIT1

volatile unsigned int brightness1 = 100; // Brightness level for LED1

volatile unsigned int brightness2 = 10000; // Brightness level for LED2

int main(void) {

WDTCTL = WDTPW + WDTHOLD; // Stop WDT

P2DIR |= BIT1 + BIT2; // Set P2.1 (LED1) and P2.2 (LED2) as outputs

P2SEL |= BIT1 + BIT2; // Select Timer B0 output on P2.1 and P2.2

TBCCR0 = brightness1; // Set initial duty cycle for LED1

TBCCR1 = brightness2; // Set initial duty cycle for LED2

TBCTL = TBSSEL\_2 + MC\_1; // SMCLK, Up mode

TBCTL |= TBCLR; // Clear TBR

TBCCTL0 = OUTMOD\_7; // CCR1 reset/set mode

TBCCTL1 = OUTMOD\_7; // CCR2 reset/set mode

while (1) {

LED1\_ON; // Turn on LED1

LED2\_ON; // Turn on LED2

}

}

#include <msp430xG46x.h>

void main(void)

{

WDTCTL = WDT\_ADLY\_1000; // 1 s interval timer

P2DIR |= BIT2; // Set P2.2 to output direction

P2SEL |= BIT2; // Set P2.2 as peripheral function (Timer B output)

P3DIR |= BIT5; // Set P3.5 to output direction

P3SEL |= BIT5; // Set P3.5 as peripheral function (Timer B output)

TB0CTL = TBSSEL\_1 + MC\_1; // ACLK is clock source, Up mode

TB0CCTL1 = OUTMOD\_4; // Toggle mode for TB0CCR1 (buzzer)

TB0CCR0 = 32; // Set TB0CCR0 for 1 kHz frequency (ACLK is assumed to be 32.768 kHz)

TB0CCR1 = 16; // Set TB0CCR1 for 50% duty cycle

TB0CCTL4 = CCIE; // Enable capture/compare interrupt for TB0CCR4 (LED toggle)

IE1 |= WDTIE; // Enable WDT interrupt

\_BIS\_SR(LPM0\_bits + GIE); // Enter LPM0 with interrupt

}

// Watchdog Timer interrupt service routine

#pragma vector=WDT\_VECTOR

\_\_interrupt void watchdog\_timer(void)

{

P2OUT ^= BIT2; // Toggle P2.2 using exclusive-OR

}

// Timer B0 interrupt service routine

#pragma vector=TIMER0\_B1\_VECTOR

\_\_interrupt void Timer\_B0(void)

{

if (TB0IV == TB0IV\_TBCCR4) // Check if interrupt was triggered by TB0CCR4

{

P2OUT ^= BIT2; // Toggle P2.2 using exclusive-OR

TB0CCR4 += TB0CCR0; // Update TB0CCR4 for next toggle (1 kHz frequency)

}

}

#include <msp430xG46x.h>

unsigned int buzz = 0;

void main(void)

{

WDTCTL = WDT\_MDLY\_32; // 1 s interval timer

P2DIR |= BIT2; // Set P2.2 to output direction

P3DIR |= BIT5; // Set P3.5 to output direction

P3SEL |= BIT5; // Set P3.5 as peripheral function (Timer B output)

P3OUT &= ~BIT5;

TB0CTL = TBSSEL\_1 + MC\_1; // ACLK is clock source, Up mode

TB0CCTL4 = OUTMOD\_4; // Toggle mode for TB0CCR1 (buzzer)

// TB0CCR1 = 16; // Set TB0CCR1 for 50% duty cycle

IE1 |= WDTIE; // Enable WDT interrupt

\_BIS\_SR(LPM0\_bits + GIE); // Enter LPM0 with interrupt

}

// Watchdog Timer interrupt service routine

#pragma vector=WDT\_VECTOR

\_\_interrupt void watchdog\_timer(void) {

static int i = 0;

i++;

if (i == 32) {

P2OUT ^= BIT2; // Toggle P2.2 using exclusive-OR

if (buzz == 0) {

buzz = 1;

TB0CCR0 = 16; // Set TB0CCR0 for 1 kHz frequency (ACLK is assumed to be 32.768 kHz)

} else {

buzz = 0;

TB0CCR0 = 0;

}

i = 0;

}

}