CPE 325: Embedded Systems Laboratory

Lab07

MSP430 Timers, Watchdog Timer, Timers A and B

Submitted by: Esther Shore

Date of Experiment: June 27, 2023

Report Deadline: July 5, 2023

Demonstration Deadline: July 10, 2023

**Introduction**

In this lab, the concepts of timers in MSP430 are introduced, including the watchdog timer and use of pulse width modulation through timer manipulation. The Timer B counts are used to control the duty cycle of the PWM signal. The use of the buzzer on the experimenter board is also examined, including changing its frequency. Low power mode is entered when no actions are required and interrupts are enabled.

**Theory Topics**

1. Watchdog timer

The watchdog timer is a type of timer peripheral device supported by the MSP430 that is able to generate a system reset after a software problem occurs or generate an interrupt after a selected time interval for the watchdog timer module. The watchdog interval timer mode can be used to provide periodic interrupts that could be used to enable a constant blinking frequency for an LED. The watchdog timer counter is configured through the watchdog timer control register, specifying the clock source and time interval.

1. Timers

TimerA and TimerB are two more timer peripheral devices supported by the MSP430. They are able to raise interrupt requests at regular intervals that can be specified by the capture and compare blocks as well as the corresponding control register. Each timer consts of a time block that supports different counting modes and source clocks. The MSP430 is able to have multiple independent timers. Timers also allow precise timestamping of events.

**Program 1 (PWM of LEDs)**

***Program Description:***

This C program, pulse width modulation (PWM) is used to control the duty cycle of power supplied and thus the brightness of LED1. Timer B is set as the source of the PWM signal. The program allows 5 levels of brightness, where SW1 is set to increase the level of brightness and SW2 is set to decrease the level of brightness. This is achieved by increasing or reducing the difference between TB0CCR1 and TB0CCR0. When TB0CCR0 is set to a constant value, increasing TB0CCR1 reduces the duty cycle and thus the brightness. Debouncing is used within the interrupt for Port 1 to verify switch presses.

**Program 2 (Buzzer)**

***Program Description:***

This C program, LED1 and the buzzer toggle every second, resulting in a frequency of 0.5Hz. The buzzer sounds at 1kHz using Timer B and is synchronized with the LED blinking. The delay is generated through the use of the ISR call induced by the watchdog timer. The microcontroller is set to sleep mode when no actions are required.

Calculations:

In this C program, the default watchdog interval timer in combination with the watchdog timer interrupt service routine to generate the approximately 1s delay between toggles. The default interval is 32ms and then within the watchdog interrupt, 32 iterations of engaging the interrupt are counted until the toggle is triggered and then the count is reset. (i.e. 31.25\*32 = 1000)

Timer\_B is configured for up mode using ACLK as the clock source. In this mode, the timer TB counts from 0 up to the value stored in TB0CCR0. So, the counter period is CCR0\*1us. The TB0 output signal is configured to toggle every time the counter reaches the value in TB0CCR0. The output frequency of the buzzer will be f = ACLK/(2\*CCR0) = 1kHz and solve for CCR0 where ACLK is 32.768kHz. This results in a TB0CCR0 value of about 16 for a buzzer frequency of 1kHz.

**Conclusion**

At first, it was a bit difficult to understand how the timers worked and setting the proper registers to get the desired effect, but it just took a lot of reading the slides and documentation to start to get the hang of it. Other than that, the buzzer was straightforward to set up and using interrupts was familiar. This lab taught a lot about using TimerB as well as the watchdog timer. It was also useful to learn the different modes and clock configurations to control the timers.

***Appendix:***

**Table 1:** Program 1 Source Code

| /\*----------------------------------------------------------------------  \* File: Lab7\_P1.c  \* Function: LED1 variable brightness using S1 and S2 interrupts to alter PWM duty cycle (MPS430FG4618)  \* Input: None  \* Output: LED1 at different brightnesses  \* Author: Esther Shore  \*---------------------------------------------------------------------\*/  #include <msp430fg4618.h>  #include <stdio.h>  #define SW1 P1IN&BIT0  #define SW2 P1IN&BIT1  int main(void) {  WDTCTL = WDTPW + WDTHOLD; // Stop WDT  P2DIR |= BIT2; // Set P2.2 (LED1) as output  P2OUT |= BIT2;  P2SEL |= BIT2; // Select Timer B0 output on P2.2  P1IE |= BIT0 | BIT1; // P1IE.BIT0 and BIT1 interrupt enabled  P1IES |= BIT0 | BIT1; // P1IES.BIT0 and BIT1 hi/low edge  P1IFG &= ~(BIT0 | BIT1); // P1IFG.BIT0 and BIT1 are cleared  TBCTL = TBSSEL\_2 + MC\_1; // SMCLK, Up mode  TBCCTL1 = OUTMOD\_2; // toggle/reset mode  TBCCR0 = 40; // Set max count value  TBCCR1 = 0; // Set count value  \_BIS\_SR(LPM0\_bits + GIE); // low power mode w/ interrupts  }  // Port 1 interrupt service routine  #pragma vector = PORT1\_VECTOR  \_\_interrupt void Port1\_ISR (void) {  if (SW1 && SW2) {  \_\_delay\_cycles(20000); // 20ms debounce  if (SW1 && SW2) {  printf("both pressed\n");  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  }  } else if (!SW1 && SW2) { // SW2 pressed  \_\_delay\_cycles(20000);  if (!SW1 && SW2) {  if (TBCCR1 > 0) {  TBCCR1 += 10; // decreases brightness  }  printf("sw2 pressed %u\n", TBCCR1);  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  }  } else if (SW1 && !SW2) { // SW1 pressed  \_\_delay\_cycles(20000);  if (SW1 && !SW2) {  if (TBCCR1 < TBCCR0) {  TBCCR1 -= 10; // increases brightness  }  printf("sw1 pressed %u\n", TBCCR1);  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  }  } else {  P1IFG &= ~BIT0; // P1IFG.BIT0 is cleared  P1IFG &= ~BIT1; // P1IFG.BIT1 is cleared  }  } |
| --- |

**Table 2:** Program 2 Source Code

| /\*----------------------------------------------------------------------  \* File: Lab7\_P2.c  \* Function: LED1 and buzzer toggle every second (MPS430FG4618)  \* Input: None  \* Output: Buzzer at 1kHz and LED1 toggle every second  \* Author: Aleksandar Milenkovic, milenkovic@computer.org  \*---------------------------------------------------------------------\*/  #include <msp430xG46x.h>  unsigned int buzz = 0;  void main(void) {  WDTCTL = WDT\_MDLY\_32; // 32ms 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 buzzer  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; // static int to track num of iterations  i++;  if (i == 32) { // ~32 iterations of 32ms = 1s  P2OUT ^= BIT2; // Toggle P2.2 using exclusive-OR  if (buzz == 0) {  buzz = 1; // toggle buzz  TB0CCR0 = 16; // 1kHz frequency (ACLK 32.768 kHz)  } else {  buzz = 0;  TB0CCR0 = 0;  }  i = 0; // reset counter  }  } |
| --- |