**Hacettepe University Department of Electrical and Electronics Engineering**

**ELE 417-EMBEDDED SYSTEM DESIGN**

**Experiment 2 Report – Timer Usage on MSP430**

**Student Name**: Baki ALMACI **Due Date**: 12.16.2020

**Student ID**: 21627983

## Questions and Answers

1. **What is a timer in the concept of embedded system design? Give at least three examples of usages for timers.**

   Timers are very versatile, being able to measure time periods, to determine pulse width, to measure speed, to measure frequency, or to provide output signals.

   i. Measuring the RPM of a car's engine.
   ii. Timing an exact period of time.(We'll use timers instead of delay or loop later)
   iii. Generating tones with PWM.

2. **What are the differences between using a timer and using an empty for loop to measure or pass the required amount of time?**

   Empty loops require the require the CPU constantly working by increasing counter and comparing it. That CPU time is spent doing nothing.
   Also, empty(dummy) loops depend on processor speed. Timers are depending on clock pulses. Clocks pulses do not change.

3. **Is there any difference between Watchdog timer and Timer A(or B)? If so, what are the differences?**

   The purpose of watchdog timer is to reset the microcontroller when there is an un-solicited status in the program unless it is disabled. It counts to only a couple of certain time intervals. But a timer is a specialized type of clock used for measuring specific time intervals. Timers can be categorized into two main types. A timer which counts upwards from zero for measuring elapsed time is often called a stopwatch, while a device which counts down from a specified time interval is more usually called a timer.

4. **What are the registers used to program timers in your choice of MSP430 kit you obtained for your project?(Emphasize the model of your kit.)**

   I am using MSP430FR4133 development board, which is supports TAxCTL, TAxCCTLn, TAxR, TAxCCRn, TAxEX0, TAxIV.

5. **How can you reset an MSP430 in software? What are the usages of software reset?(Hint: Check the course slides.)**
   i. Enabling the watchdog and make a while(1); This will cause a watchdog reset.
   ii. Doing a FLASH read (from RAM) while the flash controller is busy writing (this will cause an access violation reset).
   iii. Password Violation Reset. Write to the Watchdog timer control register without the valid password. i.e. WDTCTL = 0; Also write to flash without the password or read/jump to an invalid address. Easiest with the WDTCTL. This is a Power-Up Clear (PUC), not a full reset.
   iv. Full reset by pulling the RST pin low. We must use a spare GPIO pin and set it to output low. Produces a full Power-On Reset (POR).

6. **Led blinking by WDT.**

Watchdog Timer has been adjusted to 1000 milliseconds. WDT has been tracked by SFRIFG1 interrupt flag.

```
#include <msp430.h>

void main(void)
{
    WDTCTL = WDT_ADLY_1000; //WDT enabled to 1sec.
    PM5CTL0 &= ~LOCKLPM5;   //previous port settings are activated.
    __bis_SR_register(GIE);  //general interrupts enabled.

    P1DIR |= BIT0; // P1.0 is configured as output
    P1OUT = ~BIT0; // Led on P1.0 is turned off

    while (1)
    {
        if (SFRIFG1 & BIT0 == WDTIFG) // check interrupt
        {
            P1OUT ^= BIT0; // toggle led pin.
            SFRIFG1 &= ~WDTIFG; // Clear interrupt flag to next time.
        }
    }
}
```

7. **Write a code in C with following flow of operation:**
   i. Configure Timer A to work in up mode.
   ii. Configure Timer A interval to generate 1 second intervals.
   iii. Configure LEDs as outputs on necessary ports.
   iv. Turn on and off the LEDs with 1 second intervals by checking the Timer A interrupt flag. Usage of any kind of delay function/loop is FORBIDDEN.
   v. Run this code on your MSP430 launchpad and show Watchdog timer interrupt flag register value.(You only need to read the interrupt flag value manually; you don't need to do any interrupt handling.).

```
#include <msp430.h>

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watch dog timer
    PM5CTL0 &= ~LOCKLPM5;   //previous port settings are activated.

    P1DIR |= BIT0; // P1.0 is configured as output
    P1OUT = ~BIT0; // Led on P1.0 is turned off

    TA1CCR0 = 32797;
    TA1CTL = TASSEL_1 | MC_1 | TACLR;

    while (1)
    {
        if (TA1CTL & TAIFG == TAIFG)
        {
            P1OUT ^= BIT0;
            TA1CTL &= ~TAIFG;
        }
    }

}
```

We easily see the changes of "WDT interrupt flag" when the code directly jumped to inside of if statement.

| Name | Value | Description |
|---|---|---|
| SFRIE1 | 0x0000 | Interrupt Enable 1 [Memory Mapped] |
| SFRIFG1 | 0x00C3 | Interrupt Flag 1 [Memory Mapped] |
| JMBOUTIFG | 1 | JTAG Mail Box output Interrupt Flag |
| JMBINIFG | 1 | JTAG Mail Box input Interrupt Flag |
| NMIIFG | 0 | NMI Interrupt Flag |
| VMAIFG | 0 | Vacant Memory Interrupt Flag |
| OFIFG | 1 | Osc Fault Flag |
| WDTIFG | 1 | WDT Interrupt Flag |
| SFRRPCR | 0x001C | RESET Pin Control Register [Memory Mapped] |

8. **Write a code in C that blinks one of the LEDs on your launchpad. Blinking intervals should use Timer A operation like the previous question**.
**Usage of any kind of delay function/loop is FORBIDDEN. Blinking pattern should alternate between a short blink and long blink with each cycle,** i.e. LED is on → Short delay → LED is off → Short delay → LED is on → Long delay → LED is off → Long delay → LED is on → Short delay. . .

```
#include <msp430.h>
#include <stdbool.h>

#define LONG_DELAY 40000
#define SHORT_DELAY LONG_DELAY/4
#define SHORT 0
#define LONG 1

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // stop watch dog timer
    PM5CTL0 &= ~LOCKLPM5;   //previous port settings are activated.

    P1DIR |= BIT0; // P1.0 is configured as output
    P1OUT = ~BIT0; // Led on P1.0 is turned off

    TA1CCR0 = LONG_DELAY;
    TA1CTL = TASSEL_1 | MC_1 | TACLR;

    volatile bool delay_mode = LONG;
    volatile unsigned int counter = 0;

    while (1)
    {
        if (TA1CTL & TAIFG == TAIFG) // toggle when interrupt flag is set
        {
            P1OUT ^= BIT0;
            TA1CTL &= ~TAIFG;
            counter++;
        }

        if (counter == 2)
        {
            counter = 0;
            delay_mode ^= true;
        }

        TA1CCR0 = delay_mode ? LONG_DELAY : SHORT_DELAY;
    }

}
```