****

**HACETTEPE UNIVERSITY**

**ELECTRICAL AND ELECTRONICS ENGINEERING**

**ELE417 – EMBEDDED SYSTEM DESIGN**

**EXPERIMENT II – TIMER USAGE ON MSP430**

**PRELIMINARY WORK II**

**2021-2022 SPRING**

**Student**

**Name: Egemen Can**

**Surname: Ayduğan**

**ID: 21728036**

**Date of Submission: 22.04.2022**

**Q1.)**

A timer in the embedded system concept is a special type of clock used to measure time intervals. A timer that counts up from zero to measure elapsed time is often called a stopwatch. A device that counts down from a specified time interval and is used to create a time delay, for example an hourglass is a timer.

Usages of timers:

1. Measuring the interval of time between two-time instants.
2. Frequency of a specific signal
3. Implementing a PWM output
4. Controlling the sampling rate of the ADCs

**Q2.)**

Using any loop is not a reliable way to measure or pass the required time. The Loops are not an efficient way as processing time is difficult to detect, can vary by hardware, and uses unnecessary processing units.

On the other hand, timers provide precise measurement times and intervals. Because only hardware is used and it works asynchronously with the code. Thus, timers are more accurate and efficient than idle loops, leaving the CPU free for other processes.

**Q3.)**

The main function of the watchdog timer is to protect the system against malfunctions. The Watchdog timer can be used instead of interval timers. Also, if we need to soft reset the chip, we can use the Watchdog timer.

The Watchdog timer does not have a compare mode whereas Timer A has the TAxCCR and TAxCCTLy register to compare the value of the timer with a predefined value.

Timer A is a special type of clock used to measure specific time intervals. On the other hand, a monitoring timer essentially ensures that the embedded system is not constantly in a state of software or hardware failure.

Timer B can drive outputs such as pulse width modulation (PWM), while the Watchdog timer cannot.

**Q4.)**

MSP-EX430G2ET LaunchPad kits are required by practitioners at Hacettepe University. MSP430G2553 microcontroller is used in LaunchPad kit.

|  |  |  |
| --- | --- | --- |
|  | **Register Name** | **Definition** |
| **Watchdog Timer** | **WDTCTL** | Watchdog timer control register, 16-bit |
| **Timer A** | **TAxCCR0/1/2** | Capture/compare registers. Three for each timer |
| **TAxCCTL0/1/2** | Capture/compare control registers. Three for each timer |
| **TAxR** | Timer A counter register |
| **TAxCTL** | Timer A control register, 16-bit |
| **TAxIV** | Timer A interrupt vector |

**Q5.)**

A soft reset clears any information in the random access memory of the device related to the applications. A soft reset can help in fixing malfunctioning applications, solve problems of slowness in the device, fix incorrect settings or help in solving software related problems. It can often help

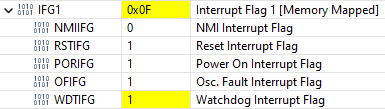
in cases where the device appears frozen or is running inefficiently.

There are several ways to reset an MSP430 in software. Some of them:

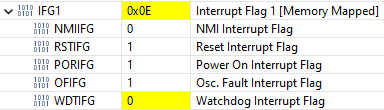
1. Enable the Watchdog and make an empty loop. This will cause a watchdog reset.
2. Writing to the WDT register without usign the proper password. This will cause a password violation reset.
3. Writing to flash while flash write flags are disabled generates a fault that cause a NMI,very similar to a reset.

**Q6.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** GreenLed BIT0  **#define** RedLed BIT6  **void** **main**(**void**){  WDTCTL = WDT\_ADLY\_1000; // WDT is set for 1 second intervals  P1DIR |= (GreenLed|RedLed); // P1.0 and P1.6 are configured as outputs  P1OUT &= ~(GreenLed|RedLed); // All Leds OFF.  **while**(1){  **while**(!(IFG1 & WDTIFG)); // Wait for WDT to reach the limit  // Reaches the limit.  IFG1 &= ~WDTIFG; // Clear WDT interrupt flag when counter  P1OUT ^= (GreenLed|RedLed); // Toggle the LEDs.  }  } |



*Figure 1. When the Timer Ends*

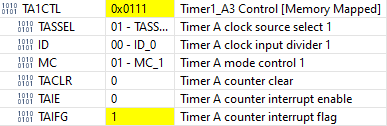


*Figure 2. Waiting on the Timer*

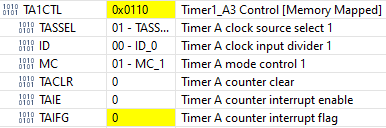
WDTIFG is 0 while waiting on the timer. It becomes 1 when it exits the timer and is set to 0 to enter the timer again.

**Q7.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** GreenLed BIT0  **#define** RedLed BIT6  **void** **main**(**void**){  WDTCTL = WDTPW | WDTHOLD; // Stop Watchdog Timer  // Timer A is configured to work in up mode and generate 1 second intervals  // TASSEL\_1 selects ACLK (32 KHz), MC\_1 selects up mode  // TACLR is Timer A counter clear and ID\_0 is Timer A input divider with 1  TA1CTL = TASSEL\_1 | MC\_1 | TACLR | ID\_0;  TA1CCR0 = 31999; // Count to 31999  P1DIR = (GreenLed|RedLed); //LEDs are configured as outputs  P1OUT &= ~(GreenLed|RedLed);  **while**(1){  **while**((TA1CTL & TAIFG) != TAIFG);  TA1CTL &= ~TAIFG; // Clear Timer A Interrupt Flag  P1OUT ^= (GreenLed|RedLed); // Toggle the LEDs  }  } |



*Figure 3. When the Timer Ends*



*Figure 4. Waiting on the Timer*

**Q8.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** GreenLed BIT0  **Void TimerA\_Delay**(**int** DelayFlag){ // Function to change Timer A delay  **if**(DelayFlag){ // Short Delay  TA1CTL = TA1CTL = TASSEL\_1 | MC\_1 | TACLR | ID\_0;  TA1CCR0 = 31999;  }  **else**{ // Long Delay  TA1CTL = TA1CTL = TASSEL\_1 | MC\_1 | TACLR | ID\_1;  TA1CCR0 = 63999;  }  **while**(!(TA1CTL & TAIFG));  }  **void** **main**(**void**){  WDTCTL = WDTPW | WDTHOLD; // Stop Watchdog Timer  P1DIR = GreenLed; // P1.0 (Green LED) configured as output  P1OUT &= ~GreenLed;  **volatile** **int** DelayFlag = 1; // flag controls when to change delay  **while**(1){  **if**(DelayFlag){  P1OUT ^= GreenLed;  TimerA\_Delay(DelayFlag);  P1OUT ^= GreenLed;  TimerA\_Delay(DelayFlag);  DelayFlag = 0;  }  **else**{  P1OUT ^= GreenLed;  TimerA\_Delay(DelayFlag);  P1OUT ^= GreenLed;  TimerA\_Delay(DelayFlag);  DelayFlag = 1;  }  }  } |

I am specifying LongDelay and ShortDelay using TimerA\_Delay function. If I send "1" to the function, Short Delay is applied, if I send "0", LongDelay is applied. Waiting is also done inside the function.