****

**HACETTEPE UNIVERSITY**

**ELECTRICAL AND ELECTRONICS ENGINEERING**

**ELE417 – EMBEDDED SYSTEM DESIGN**

**EXPERIMENT III – INTERRUPT USAGE ON MSP430**

**PRELIMINARY WORK III**

**2021-2022 SPRING**

**Student**

**Name: Egemen Can**

**Surname: Ayduğan**

**ID: 21728036**

**Date of Submission: 29.04.2022**

**Q1.)**

Timer A has 4 different modes which can be selected using MCx bit in the TACTL (Timer A control register): Stop, Up, Continuous and Up/Down.

MCx is 2 bits so it can have 4 different values, one for each mode.

* **Continuous Mode** is used when a constant interval is asked.
* **Up Mode** is used for the intervals changing. It can be used to have delays made of different intervals.
* **Up/Down Mode** is used while it is needed to have 2 flags. Which can be used to drive control/motor circuits.
* **Stop Mode** is used to stop all clock oscillations.

**Q2.)**

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process.

* **ISR** (also called an interrupt handler) is a software process invoked by an interrupt request from a hardware device. It handles the request and sends it to the CPU, interrupting the active process. When the ISR is complete, the process is resumed.
* Pragma Macro **TIMERA0 VECTOR** or **PORT1 VECTOR**

Pragma communicates to the C compiler that the following code is to be treated as the interrupt vector for the MSP430.

#pragma vector= command

This specific pragma tells the compiler that the next function following the pragma is an ISR and needs an entry in the interrupt vector table.

**Q3.)**

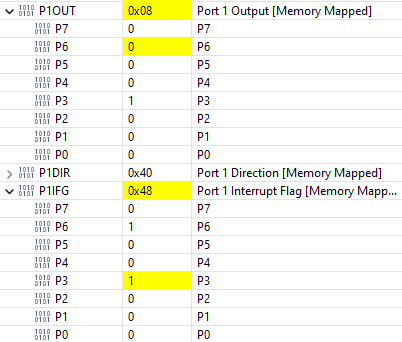
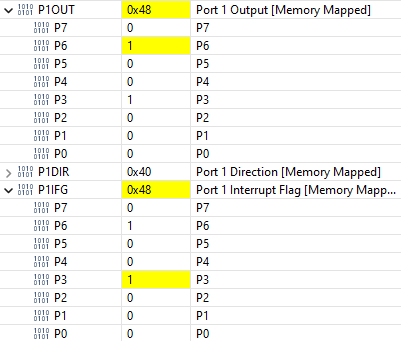
By using interrupt edge select register, we can choose if an interrupt should happen when a GPIO goes from low-to-high, or high-to-low.

**The Port x Interrupt Edge Select register (PxIES)** controls which edge an interrupt happens on. We use the term edge to mean the transition from when a signal changes from low-to-high or high-to-low.

* When a signal goes from low to high we call that a rising edge, since the signal value “rises up” to a higher level.
* When a signal goes from high to low we call that a falling edge, since the signal value “falls down” to a lower level.

**Q4.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** Switch BIT3 // Switch -> P1.3  **#define** RedLed BIT6 // RED LED -> P1.6  **void** **main**(**void**){  WDTCTL = WDTPW | WDTHOLD; // Stop Watchdog Timer  P1DIR |= RedLed; // Set LED pin -> Output  P1DIR &= ~Switch; // Set Switch pin -> Input  P1REN |= Switch; // Enable Resistor for Switch pin  P1OUT |= Switch; // Select Pull Up for Switch pin  P1IES &= ~Switch; // Select Interrupt on Rising Edge  P1IE |= Switch; // Enable Interrupt on Switch pin  **\_\_bis\_SR\_register**(LPM4\_bits + GIE); // Enter LPM4 and Enable CPU Interrupt  }  **#pragma** vector = PORT1\_VECTOR  \_\_interrupt **void** **Port\_1**(**void**){  P1OUT ^= RedLed; // Toggle Green LED  P1IFG &= ~Switch; // Clear SW interrupt flag  } |

*When the button is pressed and the led turns OFF.* *When the button is pressed and the led turns ON*.

Port1 Interrupt Flag (P1IFG) is connected to the button. When the button is pressed, button connected pin of P1IFG is 1 and the LED's status changes. In the Interrupt, I set the button connected pin of P1IFG to 0. If the button is pressed again, it enters the interrupt and changes the state of the led.

**Q5.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** RedLed BIT6 // Red LED -> P1.6  **volatile** **unsigned** **int** Counter = 0;  **volatile** **unsigned** **int** TempTACCR0 = 0;  **void** **main**(**void**){  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer  P1DIR |= RedLed; // Set LED pin -> Output  P1OUT &=~ RedLed; // Turn OFF LED  TACCR0= 2000; // Set Timer Timeout Value  TACCTL0 |= CCIE; // Enable Overflow Interrupt  TACTL |= MC\_1 + TASSEL\_1 + TACLR ; // Set Mode -> Up Count, Clock -> ACLK, Clear Timer  **\_\_bis\_SR\_register**(LPM3\_bits + GIE); // Go to LPM3 (Only ACLK active), Enable CPU Interrupt  **while**(1);  }  **#pragma** vector = TIMER0\_A0\_VECTOR // CCR0 Interrupt Vector  \_\_interrupt **void** **CCR0\_ISR**(**void**){  P1OUT ^= RedLed; // Toggle LED  Counter++;  **if**(Counter == 2){  TempTACCR0 = TACCR0;  TACCR0+= TACCR0;  **if**(TempTACCR0 > TACCR0){  TACCR0= 2000;  }  Counter = 0;  }  } |



0x07D0 = 2000



0x0FA0 = 4000



0x1F40 = 8000



0x3E80 = 16000



0x7D00 = 32000



0xFA00 = 64000



0x07D0 = 2000

**Q6.)**

|  |
| --- |
| **#include** <msp430.h>  **#define** RedLed BIT6 // Red LED -> P1.6  **volatile** **int** Counter = 0;  **void** **main**(**void**){  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer  P1DIR |= RedLed; // Set LED pin -> Output  P1OUT &=~ RedLed; // Turn OFF LED  TACCR0 = 10000; // Set Timer Timeout Value  TACCTL0 |= CCIE; // Enable Overflow Interrupt  TACTL |= MC\_1 + TASSEL\_1 + TACLR ; // Set Mode -> Up Count, Clock -> ACLK, Clear Timer  **\_\_bis\_SR\_register**(LPM3\_bits + GIE); // Go to LPM3 (Only ACLK active), Enable CPU Interrupt  **while**(1);  }  **#pragma** vector = TIMER0\_A0\_VECTOR // CCR0 Interrupt Vector  \_\_interrupt **void** **CCR0\_ISR**(**void**){  P1OUT ^= RedLed; // Toggle LED  **if**(Counter >= 0 && Counter < 2){  TACCR0 = 10000; // Short Delay  Counter++;  }  **else** **if**(Counter >= 2){  TACCR0 = 40000; // Long Delay  Counter++;  **if**(Counter == 4){  Counter = 0;  }  }  } |

Short Delay



0x2710 = 10000

Long Delay



0x9C40 = 40000