# Politechnika Opolska

**LABORATORIUM**

**Technika mikroprocesorowa**

| | | | |
|---|---|---|---|
| KIERUNEK STUDIÓW: | Automatyka i Robotyka | ROK STUDIÓW: | III |
| SEMESTR: | VI | ROK AKADEMICKI: | 2019/2020 |

| *Temat ćwiczenia:* |
|---|
| Obsługa timerów w mikrokontrolerze MSP430 |

| *Projekt wykonali:* | | | |
|---|---|---|---|
| *Nazwisko i imię:* | | *Nazwisko i imię:* | |
| **1.** | Leszek Cieśla | **2.** | Ryszard Hałapacz |
| **3.** | ---------- | **4.** | ---------- |

| *Ocena:* | *Data:* | *Uwagi:* |
|---|---|---|
| | | |

## 1. Wstęp teoretyczny

Kazdy mikrokontroler posiada timer. Timer to jest układ liczący o rozdzielczości 16 bitów. W modelu msp430 znajdujemy dwa timery, które użytko do zademonstrowania możliwości programu.

## 2. Zasada działania programu.

Poprzez wzgląd na wymagania prowadzącego zastosowano implementację pokazaną w kolejnym punkcie. Program można podzielić na cztery zasadnicze części. Pierwsza to funkcja main, gidzie następuje konfiguracją peryferii a także timerów. Program reaguje na przerwanie w postaci wciśnięcia przycisku. Następuje w nim aktywacja Timera0 i na tym ten wątej się kończy. Timer jeden obsługuję dwie diody wymuszając ich naprzemienne świecenie, następnie ję wyłącza i aktywuje timer2. W tej ostatniej części zostaję uruchomiony kod obsługujący migotanie diody czerwonej.

## 3. Skrypt napisany w języku C na mikrokontrolerze msp430

```
#include "msp430g2553.h"

#define RED BIT6                        // Red LED -> P1.6
#define GRN BIT0                        // Green LED -> P1.0
#define SW BIT3                         // Switch -> P1.3
int number_of_timer=0;
int impulse=0;                            // deklaracja zmiennej pomocniczych

void main(void){
    WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer

    TA0CCTL0 |= CCIE;                   //Ustawienie wywołania przerwania timera0 po odliczeniu ustawionego czasu - timer 0
    TA1CCTL0 |= CCIE;                   //Ustawienie wywołania przerwania timera1 po odliczeniu ustawionego czasu - timer 1

    TA0CTL |= TASSEL_2  + TACLR + ID_3; // Ustawienie liczenia - "w górę", Zegar -> ACLK, Zerowanie stanu czasu - timer 0
    TA1CTL |= TASSEL_2  + TACLR + ID_3; // Ustawienie liczenia - "w górę", Zegar -> ACLK, Zerowanie stanu czasu - timer 1


    P1DIR |= RED;                       // konfiguracja obslugi diod
    P1OUT &= ~RED;
    P1DIR |= GRN;
    P1OUT &= ~GRN;
    P1DIR &= ~SW;
    P1REN |= SW;
    P1OUT |= SW;



    TA0CCR0 = 37500;                    // konfiguracja Timera 0 na 3 sekundy
    TA1CCR0 = 12500;                    // konfiguracja Timera 1 na 1 sekundę

    P1IES &= ~SW;                       // konfiguracja przycisku
    P1IE |= SW;

    __bis_SR_register(LPM0_bits + GIE); // aktywacja  przerwania CPU
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){          //Przerwanie wywolane przyciskiem
    number_of_timer=0;                          // ustawienie wartości zmiennej number_of_timer na "0"
    TACTL |= MC_1;                      // Uruchomienie  timera 0
    P1IFG &= ~SW;                       // Czyszczenie flagi przerwania
}
```

```
#pragma vector = TIMER1_A0_VECTOR       // Przerwanie po odliczeniu czasu przez timer 1
__interrupt void CCR1_ISR(void){
    impulse++;                            // inkrementacja zmiennej
    if (impulse==10)
    {
        for( number_of_timer;number_of_timer<=8;number_of_timer++)
        {                   // wykonanie 8 zmian stanu diody czerownej

            P1OUT ^= RED;
            __delay_cycles(200000);
        }

        TA1CTL &= ~MC_3;                    // Zatrzymanie wykonania timera 1
    }
}

#pragma vector = TIMER0_A0_VECTOR       // Przerwanie po odliczeniu czasu przez timera 0
__interrupt void CCR0_ISR(void){

    impulse++;                            // inkrementacja zmiennej

    if(impulse==5)
    {                       // wykonanie kodu po zliczeniu 5 przerwań
        switch (number_of_timer)
        {
        case 0:
            {
                P1OUT |= GRN;                   //wlaczanie diody zielonej
                break;
            }
        case 1:
        case 2:
        case 3:
            {
                P1OUT ^= GRN;                   // przełączenie stanu diody zielonej
                P1OUT ^= RED;                   // przełączenie stanu diody czerwonej
                break;
            }
        default:
            {
                number_of_timer=0;              // ustawienie wartości zmiennej number_of_timer na "0"
                TA0CTL &= ~MC_3;                // Zatrzymanie timera 0
                TA1CTL |= MC_1;                 // Uruchomienie timera1
                P1OUT &= ~GRN;                  // Wyłączenie led zielonej
                P1OUT &= ~RED;                  // wyłączenie led czerwonej
                impulse=0;
            }
        }

        number_of_timer++;                          // inkrementacja zartości zmiennej number_of_timer
        impulse=0;
    }
}
```

## 4. Wnioski.

Obsluga Timerów przypomina prace programów wielowątkowych. Jeżeli kod zawarty w poszczególnych przerwaniach obsługiwanych przez Timery dotyczy tych samych peryferii albo rejestrów pamięci istnieje ryzyko wyścigu. Stwarza to ryzyko nie przewidzianego zachowania. Taka sytuacja jest bardzo trudna do wykrycia i może skutkować błędem w zupełnie innym fragmencie kodu.