Politechnika Opolska

LABORATORIUM

PRZEDMIOT:	Technika mikroprocesorowa

KIERUNEK STUDIÓW:	AiR Ns	ROK STUDIÓW:	III
SEMESTR:	VI	ROK AKADEMICKI:	2019/2020

Temat ćwiczenia:
Połączone Liczniki z timerami

Projekt wykonali:			
Nazwisko i imię:		*Nazwisko i imię:*	
1.	Marek Kaczmarczyk	**2.**	Dariusz Woźnica
3.	Michał Wacławczyk	**4.**	

Ocena:	*Data:*	*Uwagi:*

Opis zadania.
Program startuje timer T1 po w ciśnięciu przycisku, a w tym samym momencie uruchamia się zielona dioda. Po 3 sek dioda zielona gaśnie a czerwona się zaświeca i tak na zmianę póki licznik nie doliczy do 5, licznik inkrementuje się zawsze kiedy T1 skończy liczyć 3s. Kiedy licznik doliczy do 5 uruchamia się timer T2, który odlicza 1 sek. Po sekundzie program wraca na początek i czeka na wciśnięcie przycisku. W trakcie liczenia T2 dioda czerwona zamruga 2x na czerwono, która informuje o zakończeniu programu.

Skrypt programu

```
#include <msp430.h>



/**
 * program z użyciem timerów
 *
 */

#define GREEN_LED BIT0
#define RED_LED BIT6                    // Green + Red LED
#define SW BIT3                         // Switch -> P1.3
int toggle=0;
int loop_count = 0;

void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;          // Stop watchdog timer
   //inicjalizacja wejść/wyjść
   P1DIR |= (GREEN_LED+RED_LED);      // Set LED pin -> Outputs
   P1DIR &= ~SW;                      // Set SW pin -> Input
   P1REN |= SW;                       // Enable Resistor for SW pin
   P1OUT |= SW;                       // Select Pull Up for SW pin
   P1OUT &=~ GREEN_LED;               //Green LED -> OFF
   P1OUT &=~ RED_LED;                 //Red LED -> OFF
```

```
   P1IES &= ~SW;                  // Select Interrupt on Rising Edge
   P1IE |= SW;                    // Enable Interrupt on SW pin
//inicjalizacja timerów
 //timer 0
   TACCTL0 |= CCIE;               //Włączenie przerwania po przepełnieniu
   TA0CTL |=MC0 + TASSEL_2  + TACLR + ID_3 ;  // tryb -> stop, Zegar -> SMCLK, zeruje timer, podzielenie wejscia przez 8
 //timer 1
   TACCTL1 |= CCIE;
   TA1CTL |= MC0 + TASSEL_2  + TACLR + ID_3 ;  // Set Mode -> Up Count, Clock -> ACLK, Clear Timer

   TACCR0 = 37500; //limit czasu t0 - 3sek
   TACCR1 = 12500;  //limit  czasu t1 - 1sek

    __bis_SR_register(LPM0_bits + GIE);

   while(1)//program pusty, do optymalizacji
    {

        }
    }


#pragma vector = TIMER0_A0_VECTOR       //  Przerwanie po odliczeniu czasu timera 0- 3sek
__interrupt void CCR0_ISR(void)         //przerwanie dla t1
{
   volatile unsigned long i, ix;//zmienne pomocnicze

   if(loop_count < 5){
   P1OUT ^= GREEN_LED;
    P1OUT ^= RED_LED;                       // przelaczenie stanu diod
    TA0CTL |= TACLR ;                   //czyszczenie zegara
```

```
   loop_count += loop_count;
   }else if(loop_count == 5)
   {
      TA0CTL |= MC_0;  //stop timer 0
      P1OUT &=~ GREEN_LED;                  //Green LED -> OFF
        P1OUT &=~ RED_LED;                  //Red LED -> OFF
        TA0CTL |= TACLR + MC_0 ;    //timer 0 - stop i zerowanie
        TA1CTL |= MC_1 ;
        for(ix=0;i=2;i++){ //dwukrotne mruganie diodą
        P1OUT |= BIT0;            //Red LED -> ON
        for(i = 0; i<20000; i++);   //delay
        P1OUT &= ~BIT0;           //Red LED -> OFF
        for(i = 0; i<20000; i++);   //delay
        }
   }
}


#pragma vector = TIMER1_A0_VECTOR       // CCR0 Interrupt Vector
__interrupt void CCR1_ISR(void)        //przerwanie dla t1 - 1sek
{
  toggle = 0; //pozwolenie na ponowne uruchomienie programu - przyciskiem
  loop_count = 0;                           // zerowanie pętli
  TA1CTL |=  TACLR + MC_0; //timer 1 - stop i zerowanie

}


#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)               //przerwanie dla switcha
```

```
{
   if(toggle == 0){
   __delay_cycles(20000);              // Wait 20ms to debounce
   while(!(P1IN & SW));               // Wait till SW Released
   toggle =1;                      //bit nie pozwalający na ponowne uruchomienie programu
   P1OUT |= GREEN_LED ;               //green led on
   TA0CTL |= MC_1;                  // Set timer interrupt flag
   P1IFG &= ~SW;                   // Clear SW interrupt flag
   }
}
```