WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI
INSTYTUT AUTOMATYKI
KIERUNEK AUTOMATYKA I ROBOTYKA
STUDIA NIESTACJONARNE I STOPNIA

LABORATORIUM - GRUPA L1

Technika Mikroprocesorowa

ĆWICZENIE 3.

Program realizujący sekwencje przerwania sygnału wyjść mikrokontrolera

Wykonali:

Adam Bunzel
Olaf Karch

Prowadzący:

MGR INŻ. Andreas Kowol

TERMIN ODDANIA: : 06.06.2020

1. Opis działania programu

Program przy pomocy dwóch timerów kontroluje miganie diód. Po wciśnięciu przycisku rozpoczyna liczenie casu. Timer 0 odpowiada za obydwie diody powodując ich naprzemienne załączanie, po określonym czasie wyłącza je i załącza timer 1 sterujący mryganiem diody czerwonej.

2. Kod programu.

```
#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){


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


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

```
        TA0CCR0 = 37500;                // konfiguracja Timera 0 na 3 sekund

        TA1CCR0 = 12500;                // konfiguracja Timera 1 na 1 sekundy

        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

   __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);
         //number_of_timer++;                  // inkrementacja zmiennej number_of_timer
      }

      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;                // wyłączeenie led czerwonej
        impulse=0;
      }
```