# Politechnika Opolska

# L A B O R A T O R I U M

| PRZEDMIOT: | **Technika Mikroprocesorowa** |
|---|---|

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

| *Nr ćwiczenia:* | **2** | |
|---|---|---|

| *Temat ćwiczenia:* |
|---|
| ***Obsługa przerwań*** |

| *Ćwiczenie wykonali:* | | | | | |
|---|---|---|---|---|---|
| *Nazwisko:* | | *Imię:* | *Nazwisko:* | | *Imię:* |
| **1.** | **Halek** | **Krzysztof** | **2.** | **Heisig** | **Henryk** |
| **3.** | **Pryszcz** | **Krzysztof** | **4.** | | |

| *Uwagi:* | *Data:* | *Ocena za sprawozdanie:* |
|---|---|---|
| | | |

| Termin zajęć: | | | | | |
|---|---|---|---|---|---|
| Data: | 18.04.2020 | Dzień tygodnia: | Sobota | Godzina: | 15:40 |

## 1. Cel ćwiczenia

Celem ćwiczenia było napisanie programu, w którym zostanie wykorzystane przerwanie. Po wykryciu przerwania program zrealizuje dowolny kod. Elementem wykonawczym jest mikrokontroler MSP430G2553.

## 2. Zakres ćwiczenia

Naszym zadaniem było napisanie programu, który ma polegać na obsłudze przerwania programu w układzie MSP430G2553. Program jest wykonywany w momencie wykrycia zbocza rosnącego na linii wejściowej P1.3, do której podłączony jest przycisk S1. Sprzętowo ten przycisk ustawia stan niski po przyciśnięciu i dopiero po zwolnieniu następuje zbocze narastające.

Fragment programu odpowiadający za skonfigurowanie przerwania na wejściu P1.3
i reagowanie na zbocze narastające oraz za uruchomienie trybu oszczędzania energii:

```
P1IES &= ~SW;
P1IE |= SW;

__bis_SR_register(LPM4_bits + GIE);
```

Fragment kodu odpowiadający za wyzwolenie po puszczeniu przycisku:

```
__interrupt void Port_1(void){

a=a+1;
wykonaj();
P1IFG &= ~SW;
}
```

Pozostała część kodu pozostała z poprzedniego ćwiczenia:
Polega on na zmianie kombinacji zapalonych diod LED po wciśnięciu przycisku. Każdorazowe wciśnięcie przycisku powoduje zwiększenie wartości zmiennej „a” o jeden. Jeżeli zmienna ta ma zdefiniowaną jakąś wartość na płytce zostają zaświecone diody
o konfiguracji odpowiadającej wartości tej zmiennej. Liczba możliwych kombinacji wynosi 7. W celu poprawienia działania programu zastosowano instrukcje „for” aby nadać opóźnienie, które powoduje inkrementacje zmiennej „a”, w taki sposób aby przy jednym naciśnięciu przycisku warunek był wykonywany tylko raz. Jeżeli wartość zmiennej jest większa lub równa 8 następuje zmiana jej wartości na 1.

## 3. Kod programu

```
#include <msp430.h>
#define SW  BIT3                    // definiujemy zmienną SW która będzie odpowiadała
                                       za Bit3. BIT3 odpowaida za fizyczny pin P1.3 na
                                       płytce prototypowej
#define RED BIT1                    // Zdefiniujemy bit3 jako kolor czerwony diody RGB
#define GRN BIT3                    // Zdefiniujemy bit1 jako kolor zielony diody RGB
#define BLU BIT5                    // Zdefiniujemy bit5 jako kolor niebieski diody
RGB
unsigned int i,a=1;                 // rezerwujemy zmienną "i" oraz "a" jako unsigned
integer
```

```
void wykonaj(){                         // Funkcja odpowiedzialna na ustawienie wyjść aby
                                           zapalić odpowiedni kolor diody RGB

    if(a==1){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 1
        P2OUT |= GRN;
        P2OUT |= RED;                   // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT &= ~BLU;                  // w tej konfiguracji zaświecony jest kolor
                                           czerwony i zielony
             }

    if(a==2){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 2
        P2OUT &= ~GRN;
        P2OUT |= RED;                   // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT |= BLU;                   // w tej konfiguracji zaświecony jest kolor
                                           niebieski i zielony
              }

    if(a==3){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 3
        P2OUT |= GRN;
        P2OUT |= RED;                   // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT |= BLU;                   // w tej konfiguracji zaświecony jest kolor
                                           czerwony, zielony i niebieski
              }

    if(a==4){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 4
        P2OUT |= GRN;
        P2OUT &= ~RED;                  // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT |= BLU;                   // w tej konfiguracji zaświecony jest kolor
                                           zielony i niebieski
             }

    if(a==5){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 5
        P2OUT |= GRN;
        P2OUT &= ~RED;                  // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT &= ~BLU;                  // w tej konfiguracji zaświecony jest kolor
                                           zielony
              }

    if(a==6){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 6
        P2OUT &= ~GRN;
        P2OUT |= RED;                   // ustawienie wyjść dla odpowiednich pinów do
                                           których jest podłączona dioda RGB
        P2OUT &= ~BLU;                  // w tej konfiguracji zaświecony jest kolor
                                           czerwony
             }

    if(a==7){                           // warunek, który jest spełniony jeśli zmienna "a"
                                           jest równa 7
```

```
        P2OUT &= ~GRN;
        P2OUT &= ~RED;              // ustawienie wyjść dla odpowiednich pinów do
                                       których jest podłączona dioda RGB
        P2OUT |= BLU;               // w tej konfiguracji zaświecony jest kolor
                                       niebieski
               }

    if(a>=8){                       // Jeśli zmienna "a" jest większa lub równa 8
        a = 1;                      // następuje ustawienie wartości zmiennej "a" na 1
        wykonaj();                  // ponowne wykonanie funkcji aby zrealizować
                                       kombinację dla "a = 1"
             }
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;       // Zatrzymanie zegara watchdog'a

    P2DIR |= (RED+GRN+BLU);         // Ustawienie wyjść na pinach P2.3 (Czerwona LED),
                                       P2.1 (Zielona LED), P2.5(Niebieska LED)
    P1DIR &= ~SW;                   // Ustawienie wejścia na pinie P1.3 (SW2)
    P1REN |= SW;                    // włączenie rezoystora podciągającego
    P1OUT |= SW;                    // ustawienie rezystora podciągającego jako pull-
                                       up

    P1IES &= ~SW;                   // Wybranie zbocza narastajacego na wyzwalanie
                                       przerwania (puszczenie przycisku)
    P1IE |= SW;                     // Włączenie przerwania na przycisku

    __bis_SR_register(LPM4_bits + GIE); // Wejście w tryb LPM4 (Low-power Mode) oraz
                                       włączenie przerwania CPU
             }

#pragma vector=PORT1_VECTOR

__interrupt void Port_1(void){

    a=a+1;                          // inkrementacja zmiennej "a"
    wykonaj();                      // Wykonanie funkcji w której będą ustawione
                                       odpowiednie stany na wyjściach w zależności od
                                       zmiennej "a"
    P1IFG &= ~SW;                   // Czyszczenie flagi przerwania na Przycisku
                                }
```

## 4. Wnioski

Napisany program w środowisku Code Composer Studio został zapisany w pamięci mikrokontrolera MSP430G2553. Po uruchomieniu zapala się dioda RGB, a procesor przechodzi w tryb Low-power mode 4, który znacznie ogranicza pobór prądu przez mikrokontroler. W tym trybie uśpione są niemal wszystkie jego funkcje i pobiera on wg noty katalogowej 0.5 µA. W porównaniu do trybu pracy, gdzie pobór prądu wynosi „aż” 230 µA. Wciśnięcie przycisku wywołuje wybudzenie mikrokontrolera i wykonanie programu, aż do momentu zresetowania flagi przerwania.