Politechnika Opolska

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

PRZEDMIOT:	Technika mikroprocesorowa

KIERUNEK STUDIÓW:	***AUTOMATYKA I ROBOTYKA***	ROK STUDIÓW:	***3***
SPECJALNOŚĆ:	**-**		
SEMESTR:	**6**	ROK AKADEMICKI:	***2019/2020***

Temat ćwiczenia:

Wykorzystanie funkcji przerwań w mikrokontrolerze MSP430

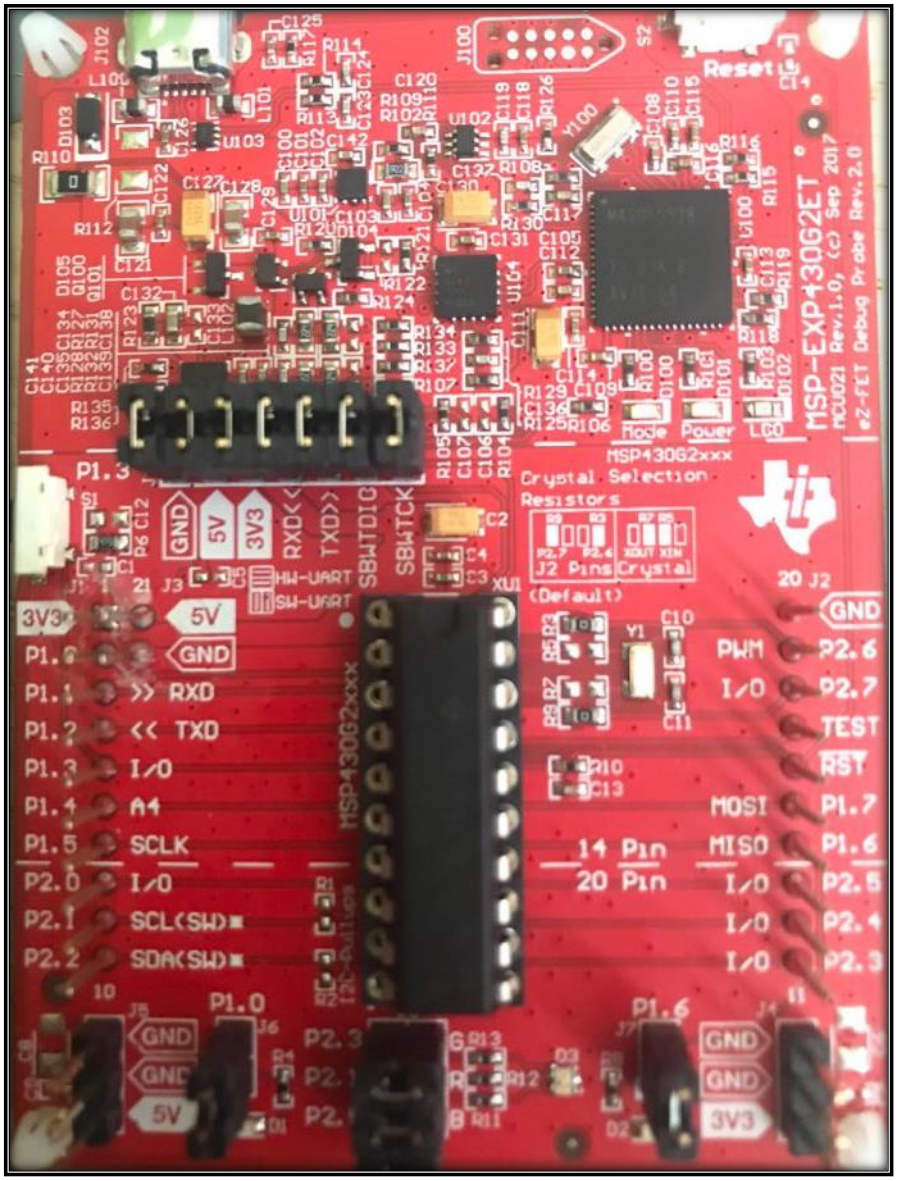
Ćwiczenie wykonali:					
Nazwisko:		*Imię:*	*Nazwisko:*		*Imię:*
1.	**Janas**	**Kamil**	**2.**	**Indyk**	**Dominik**
3.	**Dittrich**	**Sebastian**			

Uwagi:	*Data:*	*Ocena za sprawozdanie:*

Termin zajęć:
Zajęcia odbywały się w sposób zdalny

1. Cel ćwiczenia.

Celem ćwiczenia było użycie układu MSP430G2553 do zaprogramowania lub zmodyfikowanie skryptu z pierwszego ćwiczenia dodając do programu funkcję przerwania oraz zaimplementowanie go do układu. Do stworzenia tego projektu wykorzystano oprogramowanie CodeComposer. Układ jest przedstawiony na zdjęciu 1.



Zdj.1. Mikrokontroler MSP430G2553 wykorzystany w ćwiczeniu.

2. Założenia ćwiczenia.

Program w założeniu po wciśnięciu przycisku P1.3 będzie zmieniał kolory świecenia diód LED D1, D2, D3 w danej sekwencji określonej programowo. Do stworzenia programu wykorzystano program stworzony w pierwszym ćwiczeniu oraz został on zmodyfikowany o funkcję przerwań. Jak poprzednim razem wykonano 5 wariantów świecenia diód. Program jest opisany kodem w języku C przedstawiony poniżej:

```
#include <msp430.h>

#define SW BIT3          // Przycisk P1.3
#define LEDG BIT0         // Zielona dioda P1.0
#define LEDR BIT6        // Czerwona dioda P1.6

int x=0;

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

    P1DIR |= LEDR+LEDG;
    P2DIR |= BIT5+BIT3+BIT1;
    P1DIR &= ~SW;
    P1REN |= SW;
    P1OUT |= SW;
    P1IES &= ~SW;                        // Ustawienie zbocza, które powoduje
                                         // wygenerowanie flagi przerwania
    P1IE |= SW;                          // Odblokowuje możliwość modyfikacji bitu
                                         // rejestru flagi

    __bis_SR_register(LPM4_bits + GIE);  // Glowny wlacznik przerwan maskowanych
}

#pragma vector=PORT1_VECTOR

    __interrupt void Port_1(void){
    x = x+1;0;                           // Zmien wartosc

    switch(x){
        case 1:                          // Swieci sie dioda P1.6 i P2.1

            P1OUT &= ~(LEDG+LEDR);       // Negacja diody P1.0 i P1.6
            P2OUT &= ~(BIT3+BIT5);       // Negacja diody P2.3 I P2.5
            P2OUT |= BIT1;               // Ustawienie logicznej 1 na P2.1 (dioda
                                         // czerwona)
            P1OUT |= LEDR;               // Ustawienie logicznej 1 na P1.6 (dioda
                                         // czerwona)
            break;
        case 2:                          // Swieci sie dioda P1.0 i P2.3

            P1OUT &= ~LEDR;              // Negacja diody P1.6
            P2OUT &= ~BIT1;              // Negacja diody P2.1
            P1OUT |= LEDG;               // Ustawienie logicznej 1 na P1.0 (dioda
                                         // zielona)
            P2OUT |= BIT3;               // Ustawienie logicznej 1 na P2.3 (dioda
                                         // zielona)
            break;
```

```
        case 3:                         // Swieci sie dioda P2.5 (niebieska), P2.1
                                        // (czerwona), P1.0 (zielony) i P1.6
                                        // (czerwony)
            P2OUT &= ~BIT3;             // Negacja diody P2.3
            P1OUT |=LEDR+LEDG;          // Ustawienie logicznej 1 na P1.0 i P1.6
            P2OUT |= BIT5+BIT1;         // Ustawienie logicznej 1 na P2.5 (dioda
                                        // niebieska), P.2.1 (dioda czerwona)
            break;
        case 4:                         // Swieci dioda P2.5 (niebieska)
            P1OUT &= ~(LEDR+LEDG);      // Negacja diody P1.6 (czerwonej), P1.0
                                        // (zielonej)
            P2OUT &= ~BIT1;             // Negacja diody P2.1 (czerwonej)
            break;
        case 5:
             P2OUT &= ~BIT5;            // Negacja diody P2.5 (niebieskiej)
            x=0;                        // Reset instrukcji Switch case
            break;
        default:                        // Wylaczenie wszystykich diód
            P1OUT &= ~LEDG+LEDR;        // Negacja P1.0 i P1.6
            P2OUT &= ~BIT1+BIT3+BIT5;   // Negacja P2.1; P2.3; P2.5
    }
            P1IFG &= ~SW;               // Czyszczenie flagi przerwania na
                                        // przycisku
                                        // Bit 1 : Flaga ustawiana z pojawieniem
                                        // sie zbocza Hi-Lo
}
```

3. Działanie programu.

Na początku skryptu zdefiniowano przycisk, diody zieloną P1.0 oraz czerwoną P1.6, a także zaimplementowano zmienną x wykorzystywaną później w instrukcji warunkowej switch. Przed użyciem instrukcji warunkowej dodano P1IES oraz P1IE na przycisku, który służy ustawieniu zbocza, a następne powoduje odblokowanie możliwości modyfikacji bitu w rejestrze flagi. Następna linia kodu „`__bis_SR_register(LPM4_bits + GIE)`” to główny włącznik przerwań maskowanych powoduje również załączenie bitu GIE. Wewnątrz `__interrupt void Port_1(void)` umieszczono instrukcję warunkową działającą w następującej sekwencji:

1. Świecą diody czerwone P1.6 oraz P2.1.
2. Świecą diody zielone P1.0 oraz P2.3.
3. Dioda D3 wyzwala kolor niebieski P2.5 oraz czerwona P2.1 co w efekcie daje kolor fioletowy oraz czerwonej na P1.6 i zielonej na P1.0.
4. Zanegowanie poprzednich diód oraz zostawienie P2.5 co oznacza świecenie diody D3 na niebiesko.
5. Wyłączenie wszystkich diód oraz reset programu. Następne wciśnięcie spowoduje rozpoczęcie sekwencji od nowa.

Dodatkowo na płycie znajduje się przycisk reset, który powoduje po jego wciśnięciu wyświetlenie aktualnego wariantu, lecz wciśnięcie przycisku P1.3 spowoduje, że program zacznie się wykonywać od początku niezależnie na którym wariancie był, więc ostatni case 5 nie potrzebuje zwrotu zerującego zmienną x, która służy do odliczania w instrukcji warunkowej.

4. Wnioski podsumowujące.

Powyżej wykorzystano funkcję przerwań w języku C oraz jego działanie na mikrokontrolerze MSP430G2553. Zamieniono miganie diód wykonywanego w poprzednim skrypcie na połączenie kolorów świecenia na diodzie D3, ponieważ funkcja przerwań w instrukcji warunkowej pozwala jedynie na zaświecenie diody oraz jej zgaszenie, nie dokonano pomimo wielu prób możliwości migania, a to dlatego, że wykonywane w pętli for zmiany stanu na diodzie było przerywane funkcją co potwierdza jej działanie. Zmiana wariantu poprzez wciśnięcie przycisku P1.3 powodowała często niepożądane świecenie diody migającej. Stosując pętlę while, można jedynie określić po jakiej ilości mignieć ma się pętla zakończyć. W czasie działania pętli nie można było wykonać przejścia do następnego wariantu świecenia przy pomocy przycisku P1.3.