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 - MSP430.

Ćwiczenie wykonali:

Nazwisko:	*Imię:*	*Nazwisko:*	*Imię:*
1. Dziembowski	**Mateusz**	**2. --**	**--**
3. --	**--**	**4.**	

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

1. Założenia

Wykorzystując zestaw MSP-EXP430G2 produkcji Texas Instruments, napisać program realizujący sygnalizator optyczny reagujący na kolejne naciśnięcia przycisku -z **wykorzystaniem przerwania sprzętowego.** Stan programu (numer kolejnego naciśnięcia przycisku S2) odzwierciedlany będzie zachowaniem pokładowych diod świetlnych (LED1, LED2) w konfiguracji podanej w tabeli ***1.1*** poniżej ***:***

Tabela 1-1 *Tabela stanów sygnalizatora optycznego*

stan programu	LED1 czerwona	LED2 zielona
uruchomienie	OFF	OFF
pierwsze naciśnięcie przycisku S2	OFF	ON
drugie naciśnięcie przycisku S2	ON	ON
trzecie naciśnięcie przycisku S2	OFF	OFF
czwarte naciśnięcie przycisku S2	OFF	PULS
piąte naciśnięcie przycisku S2	PULS	OFF
szóste naciśnięcie przycisku S2	PULS	PULS
	synchronicznie	
siódme naciśnięcie przycisku S2	PULS	PULS
	naprzemiennie	
ósme naciśnięcie przycisku S2	OFF	OFF
	kasacja zliczania	

Opracowanie jest kontynuacją pracy „*Obsługa portów -MSP430*” autorstwa własnego. Stany sygnalizatora będą identyczne – zmieni się jedynie sposób detekcji naciśnięcia przycisku i inkrementacji zmiennej. Wykorzystane zostanie przerwanie wykonywania programu pochodzące od naciśnięcia przycisku.

2. Opis kodu programu

W linii nr 19 programu głównego, włączany jest ***Stop Watchdog Timer.*** Następnie w liniach 20 i 21 następuje ustawienie portu nr 1 Bit0 oraz Bit6 jako wyjście (P1.0 – LED1 , P1.6 – LED2) oraz w liniach 22 i 23 ustawienie wyjść na off. Linie 27-27 dotyczą przycisku S2 podłączonego do P1.3 , zadeklarowania wejścia oraz włączenia rezystora podciągającego. Nowym zagadnieniem jest obsługa przerwania na wejściu P.1.3 – linie 28-31. Wybrano zbocze sygnału binarnego które to przerwanie wywoła oraz włączono funkcję przerwania na wejściu. Opisywany fragment znajduje się poniżej :

```
17 int main(void)
18 {
19     WDTCTL = WDTPW + WDTHOLD;
20     P1DIR |= BIT0;                  // ustawienie  P1.0 (czerwona-LED) jako wyjście
21     P1DIR |= BIT6;                  // ustawienie  P1.6 (zielona-LED) jako wyjście
22     P1OUT &= ~BIT6;                 // ustawienie  P1.6 na off
23     P1OUT &= ~BIT0;                 // ustawienie  P1.0 na off
24     P1DIR &= ~BIT3;                 // ustawienie  P1.3 (przycisk S2) jako wejście
25     P1REN |=  BIT3;                 // wlaczenie rezystora pull-up na P1.3 (przycisk S2)
26
27     P1OUT |= BIT3;                  // gdy przycisk jest w górze
28     P1IES &= BIT3;                  // wybrane zbocze narastające do wyzwolenia przerwania
29     P1IE |= BIT3;                   // włączenie przerwania
30
31    _bis_SR_register(GIE);
32
```

Program wykonywany jest w pętli ***while*** zadeklarowanej w linii 33. Warunki spełnienia i wywołane akcje zostały wcześniej opisane w pracy [1] ; fragment jest identyczny z poprzednią pracą a jego składnia pokazana jest poniżej:

```
33 while(1)
34     {      /*-------warunki i akcje ----------------------------------------*/
35
36           if (b==1)                    // pierwsze nacisniecie - dioda 2 on
37           { P1OUT |=BIT6; }
38           if (b==2)                    // drugie nacisniecie - dioda 1 on
39           { P1OUT |=BIT0; }
40           if (b==3)                   // trzecie nacisniecie - obie diody off i przypisanie b=0;
41           { P1OUT &= ~BIT6;P1OUT &= ~BIT0; }
42  i=0; j=0;
43           if (b==4)                    // czwarte nacisniecie - pulsowanie diody 2 (zielona)
44           {
45               for(i;i<t_on;i++)        // czas wlaczenia
46               {P1OUT |=BIT6; }          // czas wylaczenia
47               for(j;j<t_off;j++)
48               {P1OUT &= ~BIT6; }
49
50           }
51           if (b==5)                    // piate nacisniecie - pulsowanie diody 1 (czerwona)
52           {
53               for(i;i<t_on;i++)        // czas wlaczenia
54               {P1OUT |=BIT0;}
55               for(j;j<t_off;j++)       // czas wylaczenia
56               {P1OUT &= ~BIT0;}
57
58           }
59           if (b==6)                    // szóste nacisniecie - pulsowanie obu diód synchronicznie
60           {
61               for(i;i<t_on;i++)        // czas wlaczenia
62               {P1OUT |=BIT6;P1OUT |=BIT0;}
63               for(j;j<t_off;j++)       // czas wylaczenia
64               {P1OUT &= ~BIT6;P1OUT &= ~BIT0;}
65
66           }
67           if (b==7)                    // siódme nacisniecie - pulsowanie obu diód naprzemiennie
68           {
69               for(i;i<t_on;i++)        // czas wlaczenia
70               {P1OUT |=BIT6;P1OUT &= ~BIT0;}
71               for(j;j<t_off;j++)       // czas wylaczenia
72               {P1OUT &= ~BIT6;P1OUT |=BIT0;}
73
74           }
```

Zasadnicza różnica leży w sposobie inkrementacji zmiennej pomocniczej **b** . Tym razem wykonywane jest przerwanie w wykonywaniu programu głównego i przejście do linii 79 skryptu gdzie zmienna b zostanie zwiększona o 1 – ewentualnie zostanie wyzerowana po osiągnięciu wartości **b=8.** Następnie zostanie wyczyszczona flaga przerwania (linia 85) i po zwłoce z linii 86 program wróci do linii w której był przed wywołaniem przerwania przyciskiem S2. Składnia przerwania poniżej:

```
77 #pragma vector=PORT1_VECTOR
78
79 __interrupt void Port_1(void)
80 {
81
82    b=b+1;                          // inkrementacja zmiennej "b"
83    if (b==8)                       // wylaczenie
84    {   P1OUT &= ~BIT6; P1OUT &= ~BIT0; b=0;}
85    P1IFG &= ~BIT3;                   // czyszczenie pamięci przerwania
86    __delay_cycles(30000);
87 }
```

Należy mieć na uwadze , że jeżeli program znajdował się w którejś z pętli **for** użytych do odmierzania czasu i wówczas nastąpiło przerwanie – to jego powrót nastąpi do danej pętli **for** i zostanie ona dokończona. Jeżeli oczekiwałoby się wyjścia z pętli i jej przerwania – należałoby dostawić instrukcje **break** .

3. Pełen kod programu

```
/* PROGRAM SYGNALIZATORA OPTYCZNEGO 06062020*/
/* wykorzystanie przerwania w MSP430*/
/* AiR_ns 2019/2020  */
/* Mateusz Dziembowski */

#include <msp430.h>
#include <intrinsics.h>
#include <stdio.h>

unsigned int b=0;             // zmienna inkrementowana w przerwaniu
unsigned int i;                       // zmienna pomocnicza
unsigned int j;                       // zmienna pomocnicza
unsigned int p=0;
unsigned int t_on=30000;              // czas stanu wysokiego w pulsach
unsigned int t_off=30000;             // czas stanu niskiego w pulsach

int main(void)
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR |= BIT0;                  // ustawienie  P1.0 (czerwona-LED) jako wyjście
    P1DIR |= BIT6;                  // ustawienie  P1.6 (zielona-LED) jako wyjście
    P1OUT &= ~BIT6;                 // ustawienie  P1.6 na off
    P1OUT &= ~BIT0;                 // ustawienie  P1.0 na off
    P1DIR &= ~BIT3;                 // ustawienie  P1.3 (przycisk S2) jako wejście
    P1REN |=  BIT3;                 // wlaczenie rezystora pull-up na P1.3 (przycisk S2)

    P1OUT |= BIT3;                  // gdy przycisk jest w górze
    P1IES &= BIT3;                  // wybrane zbocze narastajace do wyzwolenia przerwania
    P1IE |= BIT3;                   // włączenie przerwania

   _bis_SR_register(GIE);

while(1)
    {      /*-------warunki i akcje ---------------------------------------*/

          if (b==1)                   // pierwsze nacisniecie - dioda 2 on
          { P1OUT |=BIT6; }
          if (b==2)                   // drugie nacisniecie - dioda 1 on
          { P1OUT |=BIT0; }
          if (b==3)                  // trzecie nacisniecie - obie diody off i przypisanie b=0;
          { P1OUT &= ~BIT6;P1OUT &= ~BIT0; }
 i=0; j=0;
          if (b==4)                   // czwarte nacisniecie - pulsowanie diody 2 (zielona)
          {
              for(i;i<t_on;i++)       // czas wlaczenia
              {P1OUT |=BIT6; }          // czas wylaczenia
              for(j;j<t_off;j++)
              {P1OUT &= ~BIT6; }

          }
          if (b==5)                    // piate nacisniecie - pulsowanie diody 1 (czerwona)
          {
              for(i;i<t_on;i++)       // czas wlaczenia
              {P1OUT |=BIT0;}
              for(j;j<t_off;j++)      // czas wylaczenia
              {P1OUT &= ~BIT0;}

          }
          if (b==6)                    // szóste nacisniecie - pulsowanie obu diód synchronicznie
          {
              for(i;i<t_on;i++)       // czas wlaczenia
              {P1OUT |=BIT6;P1OUT |=BIT0;}
              for(j;j<t_off;j++)      // czas wylaczenia
              {P1OUT &= ~BIT6;P1OUT &= ~BIT0;}

          }
          if (b==7)                    // siódme nacisniecie - pulsowanie obu diód naprzemiennie
          {
              for(i;i<t_on;i++)       // czas wlaczenia
              {P1OUT |=BIT6;P1OUT &= ~BIT0;}
              for(j;j<t_off;j++)      // czas wylaczenia
              {P1OUT &= ~BIT6;P1OUT |=BIT0;}

          }
      }
}
#pragma vector=PORT1_VECTOR

__interrupt void Port_1(void)
{

    b=b+1;                          // inkrementacja zmiennej "b"
    if (b==8)                       // wylaczenie
    {   P1OUT &= ~BIT6; P1OUT &= ~BIT0; b=0;}
    P1IFG &= ~BIT3;                 // czyszczenie pamięci przerwania
     __delay_cycles(30000);
}}
```