POLITECHNIKA OPOLSKA	**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 licznika oraz zegara na płytce programowalnej MSP430 - Program 3

Ćwiczenie wykonali:					
Nazwisko:		*Imię:*	*Nazwisko:*		*Imię:*
1.	**Idzi**	**Dawid**	**2.**	**Pawlak**	**Kamil**

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

Zajęcia zdalne.

1. Cel ćwiczenia

Celem ćwiczenia było wykorzystanie zdobytej wiedzy na temat liczników oraz zegara (zwanego dalej „timer”, bądź skrótowo T1/T2) płytki MSP 430.

2. Zakres ćwiczenia

Działanie programu powinno odbywać się tak aby startował timer T1 (5 sekund) po wciśnięciu przycisku, a w tym samym momencie niech uruchomi się dioda zielona. Po 5 sekundach dioda zielona gaśnie, a czerwona się zaświeca i tak na zamianę póki licznik nie doliczy do 5, licznik inkrementuje się zawsze kiedy T1 skończy liczyć 5s. Kiedy licznik doliczy do 5 niech uruchomi się timer T2, który odlicza 1 sekundę. Po sekundzie program wraca na początek i czeka na wciśnięcie przycisku. W trakcie liczenia T2 niech dioda czerwona zamruga 2x na czerwono informując o zakończeniu programu.

Co możemy formalnie zapisać za pomocą takiego prostego algorytmu:

a) [Naciśnięcie przycisku]
b) Start timera T1 (odliczenie do 5s), włączenie diody zielonej
c) [T1=5s], reset timera T1, wyłączenie diody zielonej, włączenie diody czerwonej, inkrementacja licznika „i” o 1.
d) Powtórz krok poprzedni póki licznik „i” nie będzie równy 5.
e) [licznik „i”= 5], reset timera T1, start timera T2(odliczanie do 1s), mrugnięcie dwukrotne czerwonej diody
f) [T2=1s], reset timera T2
g) Oczekiwanie na naciśnięcie przycisku

3. Kod programu

```
#include "msp430g2553.h"

#define GRN BIT6                        // Green LED -> P1.6
#define RED BIT0                        // Red LED -> P1.0
#define SW BIT3                         // Switch -> P1.3
int i;                                   // zmienna pomocnicza - licznik i

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

    P1DIR |= RED;                       // Set LED pin -> Output
    P1OUT &= ~RED;                      // Dioda zielona OFF
    P1DIR |= GRN;                       // Set LED pin -> Output
    P1OUT &= ~GRN;                      // Dioda czerwona OFF
    P1DIR &= ~SW;                       // Ustawienie przycisku jako wyjście
    P1REN |= SW;
    P1OUT |= SW;

    TA0CCTL0 |= CCIE;                   // Timer T1 - aktywacja po skończeniu
                                          odliczenia
    TA0CTL |= TASSEL_1  + TACLR ;       // Zliczanie w górę Timera T1, reset T1
    TA1CCTL0 |= CCIE;                   // Timer T2 - aktywacja po skończeniu
                                          odliczenia
    TA1CTL |= TASSEL_1  + TACLR ;       // Zliczanie w górę Timera T1, reset T1

    TA0CCR0 = 25000;                    // Timer T1=5 sekund
    TA1CCR0 = 5000;                     // Timer T2=1 sekunda

    P1IES &= ~SW;                       // Wybranie zbocza narastajacego na
```

```
                                        wyzwalanie przerwania (puszczenie
                                        przycisku)
    P1IE |= SW;                       // Włączenie przerwania na przycisku

    __bis_SR_register(LPM0_bits + GIE); // Aktywacja trybu low power oraz
                                        aktywacja przerwań
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){          // Przerwanie wykonane przez
                                        naciśnięcie przycisku
    i=0;                              // ustawienie wartości zmiennej pomocniczej
                                        licznika na "0"
    TACTL |= MC_1;                      // Uruchomienie przerwania timera T1
    P1IFG &= ~SW;                       // Czyszczenie flagi przerwania na
                                        Przycisku
}

#pragma vector = TIMER0_A0_VECTOR       //Aktywacja przerwania kiedy T1 = 5s
__interrupt void CCR0_ISR(void){

    if (i==0){
        P1OUT |= GRN;                   // Włączenie diody zielonej
    }
    else if (i<4){
        P1OUT ^= GRN;                   // Zmiana stanu zielonej diody (na
                                        przeciwny)
        P1OUT ^= RED;                   // Zmiana stanu czerwonej diody (na
                                        przeciwny)
    }
    else{
        i=0;                          // licznik restart
        TA0CTL &= ~MC_3;                    // Timer 1 STOP
        TA1CTL |= MC_1;                     // Timer 2 START
        P1OUT &= ~GRN;                      // DIODA ZIELONA OFF
        P1OUT &= ~RED;                      // DIODA CZERWONA OFF
    }

    i++;                              // inkrementacja wartości zmiennej
                                        pomocniczej licznika
}

#pragma vector = TIMER1_A0_VECTOR       // Aktywacja przerwania kiedy T2 = 1s
__interrupt void CCR1_ISR(void){
    while(i<=4){                      // mrugnięcie czerwonej diody (instrukcja w
                                        pętli)
        P1OUT ^= RED;
        __delay_cycles(200000);

           i++;                         //  inkrementacja wartości zmiennej
                                        pomocniczej licznika
    }

    TA1CTL &= ~MC_3;                    // Timer T2 stop
}
```

4. Działanie programu na obiekcie rzeczywistym

Program wykonuje prawidłowo swoją funkcję. Należy pamiętać, że nie zawiera on algorytmów, które go potrafią przerwać, oraz w dalszych etapach rozwoju można by go było rozbudować, o funkcje zabezpieczające.

5. Wnioski

Ćwiczenie przebiegło pomyślnie. Zaprogramowana płytka, robi dokładnie to co w algorytmie przedstawionym w punkcie 2. Zarówno licznik jak i zegar to bardzo przydatne narzędzia przy programowaniu profesjonalnych płyt programowalnych. Dzięki tym funkcjonalnościom jesteśmy wstanie lepiej zawładnąć nad krokami wykonywania programów. Możemy dowolnie odliczać czas i w tym odpowiednim (czasie) wykonać odpowiedni krok. A licznik będzie czuwał nad ilością kroków. Jest to istotne, gdy chcemy zaprogramować ważne urządzenia, w których liczy się precyzja kroków oraz odpowiednie wyczucie czasu.