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:	**3**	

Temat ćwiczenia:
Zastosowanie liczników oraz timerów

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

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

Termin zajęć:					
Data:	16.05.2020	Dzień tygodnia:	Sobota	Godzina:	15:40

1. Cel ćwiczenia

Celem ćwiczenia było napisanie programu, w którym zostaną użyte zarówno liczniki jak i timery. Zapoznanie się z działaniem tych elementów oraz odpowiednie zaprogramowanie według podanego algorytmu.

2. Zakres działanie programu

Naszym zadaniem było napisanie programu, który będzie startował timer T_0 (3 sekundy) po wciśnięciu przycisku i po odliczeniu ustawionego czasu zapala się dioda zielona oraz inkrementuje się wartość licznika. Po upływie kolejnych 3 sekund dioda zielona gaśnie, a czerwona się zaświeca i wartość licznika wzrasta, aż do chwili gdy licznik doliczy do 5. Po zliczeniu do 5 uruchamia się timer T_1, który odlicza 1 sekundę. Po sekundzie program wraca na początek i czeka na wciśnięcie przycisku. W chwili liczenia T_1, dioda mruga 2 razy na czerwono informując o zakończeniu programu.

- Fragment programu odpowiadający za skonfigurowanie timerów:

```
      //Konfiguracja Timer 0
    TACCTL0 |= CCIE;                    // Ustawienie wywołania przerwania timera0 po odliczeniu ustawionego czasu
    TACTL |= TASSEL_2  + TACLR + ID_3;  // Zegar SMCLK, Zerowanie stanu czasu
    TA0CCR0 = 37500;                    // Ustawienie czas na 0,3 sekundy

    //Konfiguracja Timer 1
    TA1CCTL0 |= CCIE;                   // Ustawienie wywołania przerwania timera1 po odliczeniu ustawionego czasu
    TA1CTL |= TASSEL_2  + TACLR + ID_3; // Zegar SMCLK, Zerowanie stanu czasu
    TA1CCR0 = 12500;                    // Ustawienie czasu na 0,1 sekundy
```

- Fragment kodu odpowiadający za przerywanie wywołane wciśnięciem przycisku:

```
//konfiguracja przerwania z przycisku
    P1IES &= ~SW;                       // Wybranie zbocza narastającego na wyzwalanie przerwania (puszczenie przycisku)
    P1IE |= SW;                         // Włączenie przerwania na przycisku
```

- Uruchomienie timera T_0:

```
#pragma vector = TIMER0_A0_VECTOR       // Przerwanie po odliczeniu czasu timera 0
__interrupt void CCR0_ISR(void){

    zerojeden++;                        // inkrementacja zmiennej zliczającej wyzwalanie przerwania

    if(zerojeden==10){                  // kod wykonany po 10 przerwaniach ( 10*0,3s=3s )
        if (licznik==0){
            P1OUT |= GRN;               // Zapalenie diody zielonej po 3 sekundach wykonania kodu
        }
        else if (licznik<4){
            P1OUT ^= GRN;               // zmiana stanu na przeciwny diody zielonej
            P1OUT ^= RED;               // zmiana stanu na przeciwny diody czerwonej
        }
        else{
            licznik=0;                  // zerowanie zmiennej "licznik"
            TACTL &= ~MC_3;             // Zatrzymanie timera 0
            TA1CTL |= MC_1;             // Uruchomienie liczenia timera 1 "w górę"
            P1OUT &= ~GRN;              // Wyłączenie led zielonej
            P1OUT &= ~RED;              // wyłączenie led czerwonej
            zerojeden=0;                // zerowanie zmiennej "zerojeden"
        }
```

- Uruchomienie timera T_1:

```
   //kod po wywołaniu przerwaania uruchomiony przez timer 1
#pragma vector = TIMER1_A0_VECTOR
__interrupt void CCR1_ISR(void){
    zerojeden++;                        // inkrementacja zmiennej zliczającej wyzwalanie przerwania
    if (zerojeden==10){                 // kod wykonany po 10 przerwaniach ( 10*0,1s=1s )
        while(licznik<=4){              // wykonanie 4 zmian stanu diody czerwonej
            P1OUT ^= RED;               // zmiana stanu diody czerwonej
            __delay_cycles(150000);     // opóźnienie wykonania pętli while
            licznik++;                  // inkrementacja zmiennej licznik
        }
        TA1CTL &= ~MC_3;                // Zatrzymanie wykonania timera 1
    }
}
```

3. 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 licznik,zerojeden=0;                // deklaracja zmiennej licznik jako integer

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

    //Konfiguracja I/O
    P1DIR |= RED;                       // Set LED pin -> Output
    P1OUT &= ~RED;                      // Wyłączenie diody zielonej
    P1DIR |= GRN;                       // Set LED pin -> Output
    P1OUT &= ~GRN;                      // wyłączenie diody czerwonej
    P1DIR &= ~SW;                       // Ustawienie wejścia na pinie P1.3 (SW2)
    P1REN |= SW;                        // włączenie rezystora podciągającego
    P1OUT |= SW;                        // ustawienie rezystora podciągającego jako pull-up

    //Konfiguracja Timer 0
    TACCTL0 |= CCIE;                    // Ustawienie wywołania przerwania timera0 po odliczeniu ustawionego czasu
    TACTL |= TASSEL_2  + TACLR + ID_3;  // Zegar SMCLK, Zerowanie stanu czasu
    TA0CCR0 = 37500;                    // Ustawienie czas na 0,3 sekundy

    //Konfiguracja Timer 1
    TA1CCTL0 |= CCIE;                   // Ustawienie wywołania przerwania timera1 po odliczeniu ustawionego czasu
    TA1CTL |= TASSEL_2  + TACLR + ID_3; // Zegar SMCLK, Zerowanie stanu czasu
    TA1CCR0 = 12500;                    // Ustawienie czasu na 0,1 sekundy

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

    //wejście w tryb oszczędzania energii
    __bis_SR_register(LPM1_bits + GIE); // Wejście w tryb LPM1 (Low-power Mode) oraz włączenie przerwania CPU
}
```

```
//kod po wywołaniu przerwania uruchomiony przez timer 0
#pragma vector = TIMER0_A0_VECTOR       // Przerwanie po odliczeniu czasu timera 0
__interrupt void CCR0_ISR(void){

    zerojeden++;                        // inkrementacja zmiennej zliczającej wyzwalanie przerwania

    if(zerojeden==10){                  // kod wykonany po 10 przerwaniach ( 10*0,3s=3s )
        if (licznik==0){
            P1OUT |= GRN;               // Zapalenie diody zielonej po 3 sekundach wykonania kodu
        }
        else if (licznik<4){
            P1OUT ^= GRN;               // zmiana stanu na przeciwny diody zielonej
            P1OUT ^= RED;               // zmiana stanu na przeciwny diody czerwonej
        }
        else{
            licznik=0;                  // zerowanie zmiennej "licznik"
            TACTL &= ~MC_3;             // Zatrzymanie timera 0
            TA1CTL |= MC_1;             // Uruchomienie liczenia timera 1 "w górę"
            P1OUT &= ~GRN;              // Wyłączenie led zielonej
            P1OUT &= ~RED;              // wyłączenie led czerwonej
            zerojeden=0;                // zerowanie zmiennej "zerojeden"
        }

        licznik++;                      // inkrementacja zartości zmiennej licznik
        zerojeden=0;                    // zerowanie zmiennej "zerojeden"
    }


//kod po wywołaniu przerwaania uruchomiony przez timer 1
#pragma vector = TIMER1_A0_VECTOR
__interrupt void CCR1_ISR(void){
    zerojeden++;                        // inkrementacja zmiennej zliczającej wyzwalanie przerwania
    if (zerojeden==10){                 // kod wykonany po 10 przerwaniach ( 10*0,1s=1s )
        while(licznik<=4){              // wykonanie 4 zmian stanu diody czerwonej
            P1OUT ^= RED;               // zmiana stanu diody czerwonej
            __delay_cycles(150000);     // opóźnienie wykonania pętli while
            licznik++;                  // inkrementacja zmiennej licznik
        }
        TA1CTL &= ~MC_3;                // Zatrzymanie wykonania timera 1
    }
```

4. Wnioski

Układ działa według przedstawionych założeń. W kodzie zastosowane zostały przerwania wywoływane za pomocą timerów oraz liczniki inkrementowane w każdym cyklu. Z racji że użyliśmy zegara SMCLK który ustawiony został na 1 MHz musieliśmy zamienię tę wartość na czas w którym zostanie odliczona 1 sekunda. Do tego celu został włączony dzielnik zegara o wartości 8 oraz ustawienie ilości impulsów licznika na 37500 (0,3s) i 12500 (0,1s). Dodatkowo w każdym przerwaniu został użyty licznik inkrementowany co każde wywołanie przerwania. Dopiero po 10 wywołaniach został wykonany docelowy kod. Kod ten ma na celu ustawianie stanów wysokich na wyjściach w odpowiednich ramach czasowych. Po wykonaniu pełnego cyklu program oczekuje ponownego impulsu do rozpoczęcia pracy (naciśnięcie przycisku). Podsumowując student pracujący z mikrokontrolerem rozpoczyna cykl pracy, mikrokontroler prowadzi cykl do końca. Nie ma potrzeby potwierdzania zakończenia cyklu pracy.