WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI
INSTYTUT AUTOMATYKI
KIERUNEK AUTOMATYKA I ROBOTYKA
STUDIA NIESTACJONARNE I STOPNIA

LABORATORIUM - GRUPA L2

Technika mikroprocesorowa

ĆWICZENIE 3.

Temat: Wykorzystanie timerów i liczników w mikrokontrolerze MSP430.

Wykonali:

97938- Kamil Moczygęba
96249- Konrad Riabowski

Prowadzący:

MGR INŻ. ANDREAS KOWOL

1. Opis ćwiczenia.

Głównym celem ćwiczenia było poznanie działania timerów i liczników w płytce MSP430 oraz stworzenie programu opartego na ich działaniu. Kod został napisany w programie CodeComposer. Na początku zdefiniowano zmienne w oparciu o porty, bity oraz zmienne całkowite odpowiedzialne za licznik. W mainie zostały zadeklarowane bity diod, przycisk oraz ustawienia obu timerów. Następnie ustawiono przerwanie timera 0 oraz wyzerowanie licznika. W kolejnej części kodu zostało określone przerwanie po odliczeniu czasu 3s oraz wariant przełączania się diód zliczanych w liczniku z wykorzystaniem instrukcji warunkowej *if*. W ostatniej części kodu określono wykonanie przerwania na timera 1 w którym zawarto wykonanie 2 mignięć diody czerwonej w opóźnieniu 1 sekundy po czym następuje zakończenie programu oraz zatrzymanie timera 1. Pozwala to przy ponownym przyciśnieciu przycisku na kolejne wykonywanie programu.

2. Kod programu.

```
#include "msp430g2553.h"
// DEFINICJA DIOD I PRZELACZNIKA
#define ZIELONA BIT0                    // ZIELONA DIODA P1.0
#define CZERWONA BIT6                   // CZERWONA DIODA P1.6
#define PRZELACZNIK BIT3                // PRZELACZNIK P1.3
// DEFINICJA ZMIENNYCH CALKOWITYCH DO REALIZACJA PROGRAMU
int counter;                            // ZMIENNA CALKOWITA
int x10=0;                              // ZMIENNA CALKOWITA
void main(void){
    WDTCTL = WDTPW + WDTHOLD;           // STOP WATCHDOG TIMER
    // DIODA ZIELONA
    P1DIR |= ZIELONA;                   // USTAWIENIE NA WYJSCIU BITU DIODY
    P1OUT &= ~ZIELONA;                  // WYLACZENIE DIODY ZIELONEJ
    // DIODA CZERWONA
    P1DIR |= CZERWONA;                  // USTAWIENIE NA WYJSCIU BITU DIODY
    P1OUT &= ~CZERWONA;                 // WYLACZENIE DIODY CZERWONEJ
    // PRZELACZNIK
    P1DIR &= ~PRZELACZNIK;              // USTAWIENIE PRZELACZNIKA
    P1REN |= PRZELACZNIK;               // URUCHOMIENIE REZYSTORA P1.3
    P1OUT |= PRZELACZNIK;               // USTAWIENIE PULL-UP NA REZYSTORZE
    // TIMER
    TA0CCTL0 |= CCIE;                   // USTAWIENIE PRZERWANIA PO TIMERZE0
    TA0CTL |= TASSEL_2 + TACLR + ID_3;  // WYKONANIE ZLICZANIA NA TIMERZE SMCLK
    TA1CCTL0 |= CCIE;                   // USTAWIENIE PRZERWANIA PO TIMERZE1
    TA1CTL |= TASSEL_2 + TACLR + ID_3;  // WYKONANIE ZLICZANIA NA TIMERZE SMCLK
    TA0CCR0 = 37000;                    // TIMER0 - 3s
    TA1CCR0 = 12500;                    // TIMER1 - 1s
    // PRZERWANIE
    P1IES &= ~PRZELACZNIK;                      // USTAWIENIE ZBOCZA NARASTAJACEGO PRZY PUSZCZANIU PRZYCISKU
    P1IE |= PRZELACZNIK;                        // PRZERWANIE NA PRZYCISKU

    __bis_SR_register(LPM0_bits + GIE);
}
```

```
// USTAWIENIE PRZERWANIA NA WCISNIECIE PRZYCISKU
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){
    counter=0;                          // WARTOSC ZMIENNEJ USTAWIONEJ NA 0
    TACTL |= MC_1;                      // PRZERWANIE TIMERA0
    P1IFG &= ~PRZELACZNIK;              // NASTEPUJE CZYSZCZENIE FLAGI PRZYCISKU
}
// NASTEPUJE PRZERWANIE PO ODLICZENIU TIMERA0
#pragma vector = TIMER0_A0_VECTOR
__interrupt void CCR0_ISR(void){

    x10++;                              // INKREMENTACJA ZMIENNEJ X10
    if (x10==10){                       // JESLI ZMIENNA X10 BEDZIE WYNOSILA 10
        if (counter==0){                // ORAZ ZMIENNA COUNTER 0 TO
            P1OUT |= ZIELONA;           // NASTEPUJE ZAPALENIE DIODY ZIELONEJ
        }
        else if (counter<4){            // GDY ZMIENNA COUNTER MNIEJSZA OD 4
            P1OUT ^= ZIELONA;           // NASTEPUJE ZMIANA STANU NA DIODZIE ZIELONEJ
            P1OUT ^= CZERWONA;          // ORAZ ZMIANA STANU NA DIODZIE CZERWONEJ
        }
        else{                           // W POZOSTALYCH PRZYPADKACH
            counter=0;                  // ZMIENNA COUNTER WYNOSI 1
            TA0CTL &= ~MC_3;            // ZATRZYMA SIE TIMER0 ORAZ
            TA1CTL |= MC_1;             // URUCHOMI SIE TIMER1
            P1OUT &= ~ZIELONA;          // WYLACZENIE DIODY ZIELONEJ ORAZ
            P1OUT &= ~CZERWONA;         // WYLACZENIE DIODY CZERWONEJ
            x10=0;                      // WYZEROWANIE ZMIENNEJ X10
        }
        x10=0;                          // WYZEROWANIE ZMIENNEJ X10
        counter++;                      // ZMIENNA COUNTER JEST INKREMENTOWANA
    }
}

// WYKONUJE SIE PRZERWANIE ODLICZANIE TIMERA1
#pragma vector = TIMER1_A0_VECTOR
__interrupt void CCR1_ISR(void){
    while(counter<=4){                          // WYKONANIE 2 MIGNIEC DIODY CZERWONEJ
        P1OUT ^= CZERWONA;                      // ZMIANA STANU WYJSCIA DIODY CZERWONEJ
        __delay_cycles(1000000);                // OPOZNIENIE WYKONANIA PETLI - 1S
        counter++;                              //ZMIENNA COUNTER JEST INKREMENTOWANA
    }

    TA1CTL &= ~MC_3;                            // ZATRZYMANIE TIMERA1

}
```

3. Podsumowanie.

Powyższy kod udowadnia, że za pomocą algorytmów wykorzystujących liczniki oraz timery możemy bez problemu sterować świeceniem diod, bez konieczności używania przycisków. Z powodu braku płytki, której używaliśmy na laboratorium, nie mamy możliwości przetestowania programu i sprawdzenia jego poprawności. Jednak uważamy, że program powinien działać prawidłowo.