Politechnika Opolska

LABORATORIUM

Technika mikroprocesorowa

KIERUNEK STUDIÓW:	Automatyka i Robotyka		ROK STUDIÓW:	III
SEMESTR:	VI	ROK AKADEMICKI:		2019/2020

Temat ćwiczenia:
Obsługa przerwana mikrokontrolerze MSP430

Projekt wykonali:			
Nazwisko i imię:		*Nazwisko i imię:*	
1.	Ryszard Hałapacz	**2.**	
3.	Leszek Cieśla	**4.**	

Ocena:	*Data:*	*Uwagi:*

1. Wstęp teoretyczny

W trakcie działania programu mogą pojawić się różne zdarzenia, na które musi reagować mikrokontroler. Zdarzenia te zwykle są związane z działaniem układu, w którym pracuje mikrokontroler. Zadanie to da się zrealizować za pomocą pętli, która kolejno sprawdza każde z nich. Niestety, zajmuje to cenny czas pracy mikrokontrolera oraz komplikuje sam program. Dlatego wymyślono przerwania (ang. interrupts). Przerwanie jest zdarzeniem, które przerywa wykonywanie programu głównego i uruchamia specjalną funkcję obsługi przerwania. Gdy funkcja obsłuży przerwanie, następuje powrót do przerwanego programu i wznowienie jego wykonywania od miejsca, w którym został przerwany.

2. Zasada działania

Celem ćwiczenia była zapoznanie się z obsługa przerwań na mikrokontrolerze MSP430 producenta Texas industries. Do wykorzystania mieliśmy elementy wbudowane w mikrokontroler tzn dwie diody zielona, czerwona oraz przycisk.

Idea działania jest prosta. Dioda czerwona jest obsługiwana przez kod w funkcji main. Zależało nam na pokazaniu szybkości obsłużenia wątku i powrotu wykonywania się głównego wątku (zwłaszcza pętli while). W przeciwieństwie od diody czerwonej, praca diody zielonej determinowana jest w kodzie zawartym w obsłudze przerwań. Dioda reaguje na przycisk. Każde wciśnięcie zaświeca badź gasi diodę (zależnie od jej stanu poprzedniego)

3. Opis programu

Główny. Program rozpoczyna się od definiowania nazw, żeby uniknąć potrzeby odwoływania się do konkretnych bitów oraz definicją zmiennej globalnej . Funkcje main dzieli się na dwie zasadnicze części, w pierwszej jest konfiguracja mikrokontrolera , tzn porty I/O, obsługę przerwań. Drugą częścią jest pętla while, która obsługuje czerwoną diodę. W pętli na przemian dokonuję się zmiana stanu diody z tak dobranym opóźnieniem by ludzkie oko mogło dostrzec jej pulsację. Mamy dwie częstotliwości, których wybór uwarunkowany jest od zmiennej globalnej. Kod zawarty w obsłudze przerwań inkrementuje zmienną flag oraz zmienia status diody zielonej.

Reasumując obydwa kody dokonują tego samego – decydują o pracy diod. Czerwona obslugiwana jest przez program główny, zielona uwarunkowana jest obsługą przerwań,

Kod programy napisany w języku C:

```
#include <msp430.h>


#define SW      BIT3                    // Podstawowe Definiowanie
#define GREEN   BIT6
#define RED     BIT0
#define Time1   62000
#define Time2   32000

volatile int flag=0;                    // ustawianie zmiennej globalnej flag i nadanie jej wartosci 1
                                        // slowo kluczowe volatile mowi kompilatorowi o mozliwosci
                                        // zamiany wartosci zmiennej z zewnatrz
                                        // Dzieki  slowu kompilator nie dokonuje optymalizacji kodu
                                        // (w normalnym warunkach przeypisalby jej wartosc constant
                                        //   w procesie optymalizacji)
                                        // i za kazdym razem sprawdza rzeczywista wartosc zmiennej



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

    P1DIR |= GREEN + RED;               // Diody Led jako wyjscie
    P1DIR &= ~SW;                       //Switch jako wejscie
```

```
    P1REN |= SW;                          //Aktywowany jest rezystor podciągający typu pull-up
    P1OUT |= SW;

    P1IES &= ~SW;
    P1IE |= SW;                           //Włączana jest obsługa przerwań dla wejścia

    __bis_SR_register( GIE);             // odblokowanie obslugi przerwan

    while(1)                              // pętla główna programu
     {
          P1OUT ^= GREEN;
          if(flag%2) __delay_cycles(Time1);       // Warunek warunkujacy czestotliwosc
          else __delay_cycles(Time2);
     };
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{

    P1OUT ^= RED;                                                          // Zmiana stanu Green LED
    flag++;                                                                // inkremenentacja zmieonnej globalnej
    P1IFG &= ~SW;                                                          // Zerowana jest flaga przerwania:
    __bic_SR_register_on_exit( LPM4_bits );                                 //instrukcja która nakazuje zakończenie
procedury obsługi przerwania

}
```