Instytut Informatyki Politechniki Śląskiej
Zespół Mikroinformatyki i Teorii Automatów Cyfrowych

Laboratorium SMiW

Rok akademicki	Rodzaj studiów*: SSI/NSI/NSM	Numer ćwiczenia:	Grupa	Sekcja
2019/2020	SSI	9	2	1

Data i godzina planowana ćwiczenia: dd/mm/rrrr - gg:mm	13/12/2019 10:30	**Prowadzący**: OA/JP/KT/GD/BSz/GB	JP
Data i godzina wykonania ćwiczenia: dd/mm/rrrr - gg:mm	27/01/2020 10:30		

Sprawozdanie

Temat ćwiczenia:

Mikrokontrolery serii AVR - Przerwania

Skład sekcji:	1. Krzysztof Chodkiewicz 2. Łukasz Pojda 3. Damian Tabaka

Treść zadania:

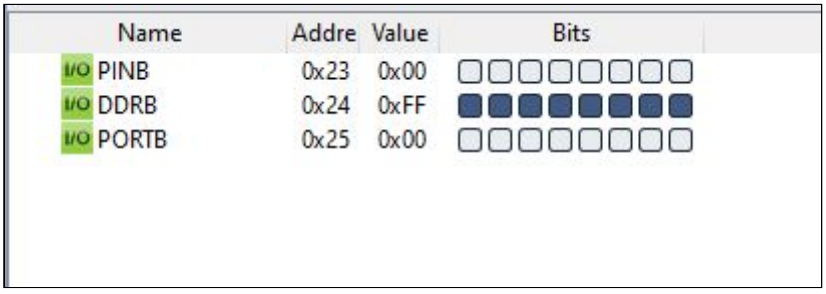
Napisać program który kopiuje bez końca zawartość jednej tablicy do drugiej w sytuacji gdy wciśnięty jest przynajmniej jeden przycisk. Po przekopiowaniu jednego bajtu program powinien zaczekać 0.2 sekundy, natomiast po przekopiowaniu całej tablicy powinna zapalić się dioda na 1 sekundę.

Rozwiązanie (kod programu):

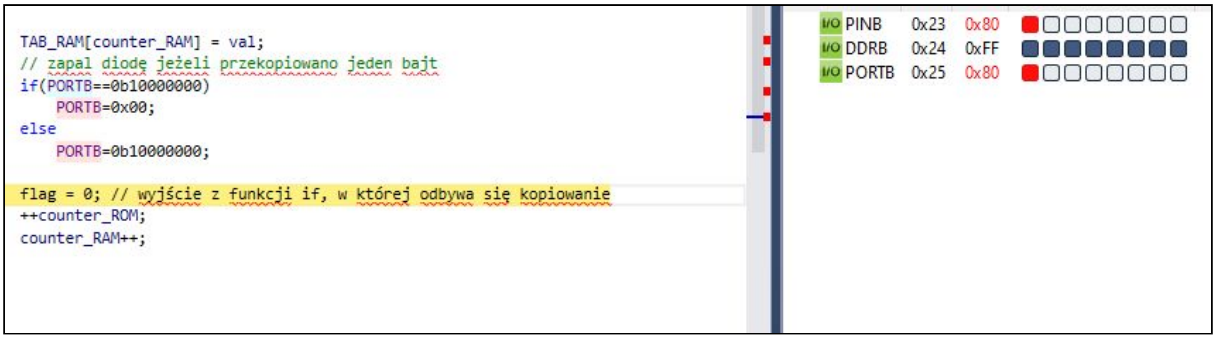
```
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#define F_CPU 16000000L
#define TCNT1_Start_Val (65535 - 10)
#define nLength 50
uint8_t TAB_RAM[nLength];
// skrócona tablica w celach testowych
uint8_t const TAB_ROM[] PROGMEM = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                                   0x08, 0x09, 0x0A, 0xFF, 0xFF, 0x08, 0x00, 0x00};

volatile uint8_t flag = 0; // flaga informująca o przekopiowaniu jednego bajtu
volatile uint32_t counter_byte = 0; // licznik spowalniający kopiowanie

ISR(TIMER1_OVF_vect)
{
    counter_byte++;
    TCNT1 = TCNT1_Start_Val;
    TIFR1 &= ~(1 << TOV1);
};

void setTimer() // inicjalizacja przerwania
{
    TCCR1B = (1 << CS10);  //ustawienie prescalera na 1024
    TCNT1 = TCNT1_Start_Val;
    TIMSK1 |= (1 << TOIE1);
}

int main (void)
{
    // wygaszenie wszystkich diod oraz ustawienie portu A jako wejście
    DDRA = 0x00;
    PORTA = 0xFF;
    DDRB = 0xFF;
    PORTB = 0x00;
    DDRC = 0xFF;
    PORTC = 0x00;
    DDRL = 0xFF;
    PORTL = 0x00;

    uint8_t counter_ROM = 0; // zmienna do poruszania się po tablicy TAB_ROM
```

```
    uint8_t counter_RAM = 0; // zmienna do poruszania się po tablicy TAB_RAM
    uint8_t counter_EOC = 0; // licznik pozwalający odliczyć sekundę
    uint8_t end_of_cycle = 0; // flaga końca jednego cyklu (end_of_cycle = 1 - koniec
cyklu)

    setTimer();
    sei();

    while(1) {
        while (PINA != 0xFF) // wykonuj jezeli jakikolwiek przycisk na porcie A jest
wciśnięty
        {
                if (counter_byte >= 44000) // jeżeli counter_byte doliczy do 44000
(200ms) to pozwól na dalsze kopiowanie
                {
                        flag = 1; // kontynuuj kopiowanie
                        counter_byte = 0;
                }
                if (flag == 1)
                {
                        if (end_of_cycle == 1) // jeżeli koniec cyklu
                        {
                                ++counter_EOC;
                                PORTB=0b0100000; // zapal diodę jezeli koniec cyklu
                                flag=0;
                                if (counter_EOC == 5) // 1 sek. (5*200ms)
                                {
                                  PORTB=0x00; // wyłącz diodę po upływie 1 sekundy
                                  end_of_cycle = 0;
                                  counter_EOC = 0;
                                }
                        }
                        else if (end_of_cycle == 0) // jeżeli cykl wciąż trwa
                        {
                          // kopiowanie danych
                          uint32_t* addr = pgm_get_far_address(TAB_ROM)+counter_ROM;
                          uint32_t bity = *addr;
                          bity = bity>>16;
                          RAMPZ=bity;
                          uint8_t val =
pgm_read_byte_far(pgm_get_far_address(TAB_ROM)+counter_ROM);
                                uint8_t val2 =
pgm_read_byte_far(pgm_get_far_address(TAB_ROM)+counter_ROM+1);

if (counter_RAM == nLength) // wykrycie końca tablicy TAB_RAM
{
   counter_RAM = 0;
}

if(val == 0x00 && val2 == 0x00) // wykrycie końca tablicy TAB_ROM
```

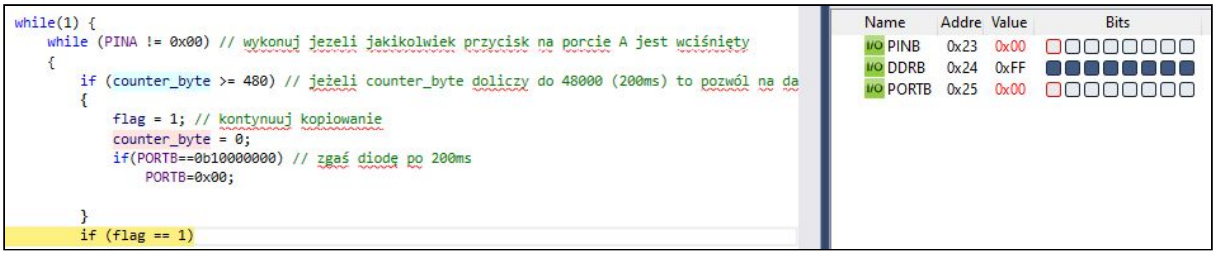
```
{
   counter_ROM = 0;
   end_of_cycle = 1;
}

TAB_RAM[counter_RAM] = val;
// zapal diodę jeżeli przekopiowano jeden bajt
if(PORTB==0b10000000)
   PORTB=0x00;
else
   PORTB=0b10000000;

flag = 0; // wyjście z funkcji if, w której odbywa się kopiowanie
++counter_ROM;
counter_RAM++;
    }
   }
  }
 }
}
```



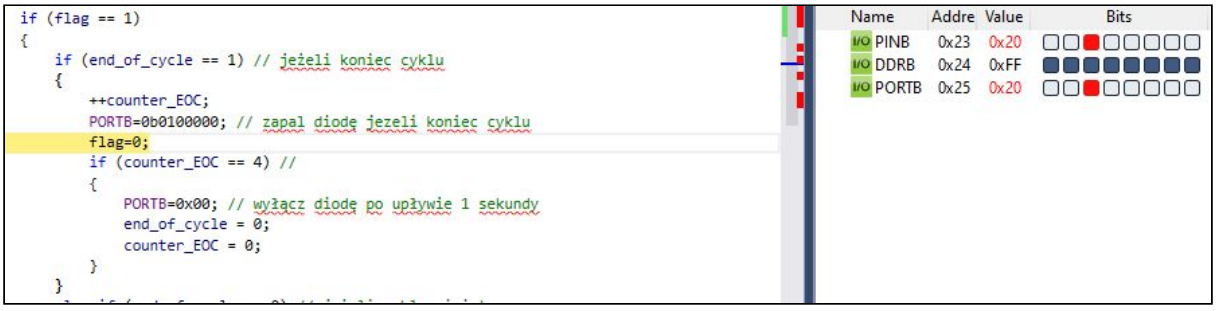
Zrzut 1. Początkowy stan na porcie B



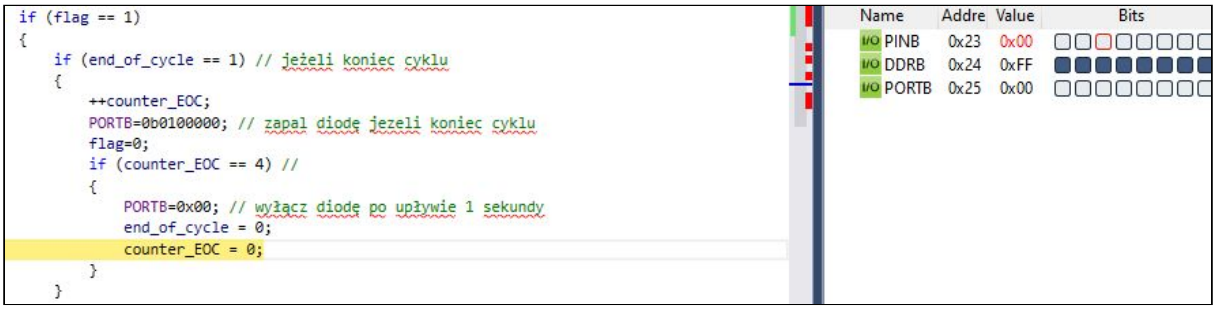
Zrzut 2. Stan portu B po przekopiowaniu 1B



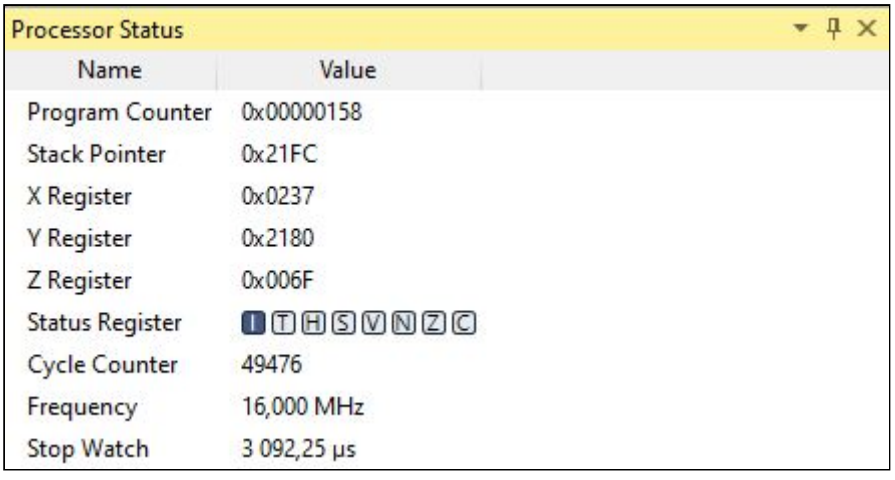
Zrzut 3. Stan portu B po 200ms od przekopiowania bajtu



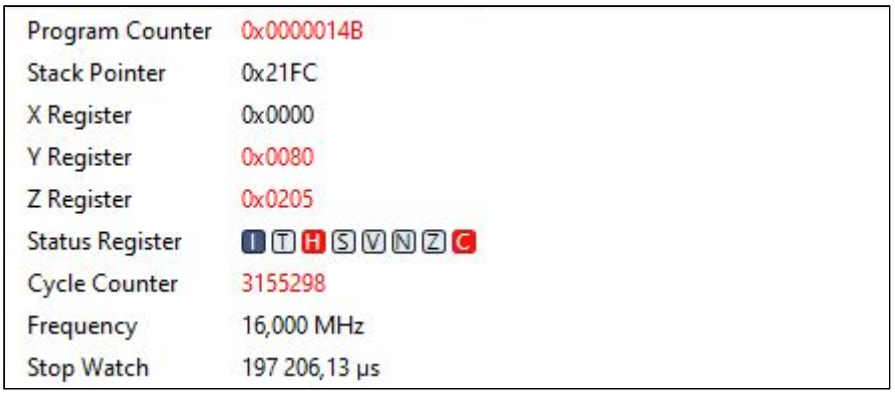
Zrzut 4. Stan portu B po zakończeniu cyklu kopiowania



Zrzut 5. Stan portu B po 1s od zakończenia cyklu



Zrzut 6. Stan procesora w chwili uruchomienia programu



Zrzut 7. Stan procesora po przekopiowaniu 1B

Wnioski i uwagi końcowe:

Ćwiczenie pomogło nam bliżej zapoznać się z mikrokontrolerami z rodziny AVR. Dzięki laboratorium mogliśmy poćwiczyć operacje na przerwaniach sprzętowych dzięki, którym kod stał się bardziej przejrzysty a sam program pracuje bardziej ekonomicznie. Największym problemem było, odpowiednie ustawienie zegara dla naszego przerwania by cały program działał zgodnie z założeniami.