SPRAWOZDANIE

SYSTEMY WBUDOWANE

Program wbudowany dla mikrokontrolera AVR.

Praca w środowisku IDE.

IMIĘ I NAZWISKO: Jacek Wójcik

NUMER ĆWICZENIA: 2

Grupa laboratoryjna: 10

Data wykonania ćwiczenia: 20.10.2021

Spis treści

Spis treści **2**

1. Zadanie 1 - kod **3**

2. Opis zadania 1 **4**

3. Zadanie 2 **6**

4. Opis zadania 2 **7**

5. Wnioski **8**

1. Zadanie 1 - kod

```c
#include <avr/io.h> // Konieczne biblioteki

unsigned long pczekaj=1500; // zmienna zawierająca wartość opóźnienia

void czekaj(unsigned long pt)  //procedura  opóźnienia  o  zadany  czas pt
{
    for(;pt>0;pt--) //główna pętla opóźnienia
    {
        for(unsigned char tp1=255;tp1!=0;tp1--) // druga pętla opóźnienia
            asm volatile("nop");  // Instrukcja konieczna do działania programu na fizycznej atmedze a nie tylko w HAPSIM
    }
}

int main(void) //program główny
{
    unsigned char ledy,i,licznik; // deklaracja zmiennych
    DDRB=0xff;  //konfiguracja wszystkich wyprowadzeń portu B jako wyjścia

    while(1)    //nieskończona pętla główna programu
    {
        //efekt węża
        for(licznik=0;licznik<10;licznik++) //pętla długości trwania efektu (liczba cykli danego efektu)
        {
            PORTB=0;  //wygaś LED-y
            czekaj(pczekaj); //opóźnij o zadany czas
            for(i=0;i<8;i++) //pętla zmieniająca fazę efektu
            {
                PORTB|=_BV(i); //wysteruj (zapal) pojedynczego LED-a
                czekaj(pczekaj); //opóźnij o zadany czas
            }
            for(i=0;i<8;i++)   //pętla zmieniająca fazę efektu
            {
                PORTB&=~_BV(i); //wysteruj (zgaś) pojedynczego LED-a
                czekaj(pczekaj);  //opóźnij o zadany czas
            }
        }

        //efekt biegnącego punktu
        for(licznik=0;licznik<10;licznik++)  //pętla długości trwania efektu (liczba cykli danego efektu)
        {
            for(ledy=0xfe;ledy!=0xff;ledy=(ledy<<=1)+1) //pętla zmieniająca fazę efektu
            {
                PORTB=ledy; //wysterowanie LED-ów zgodne z wartością zmiennej ledy
                czekaj(pczekaj);  //opóźnij o zadany czas
            }
        }
    }
}

```

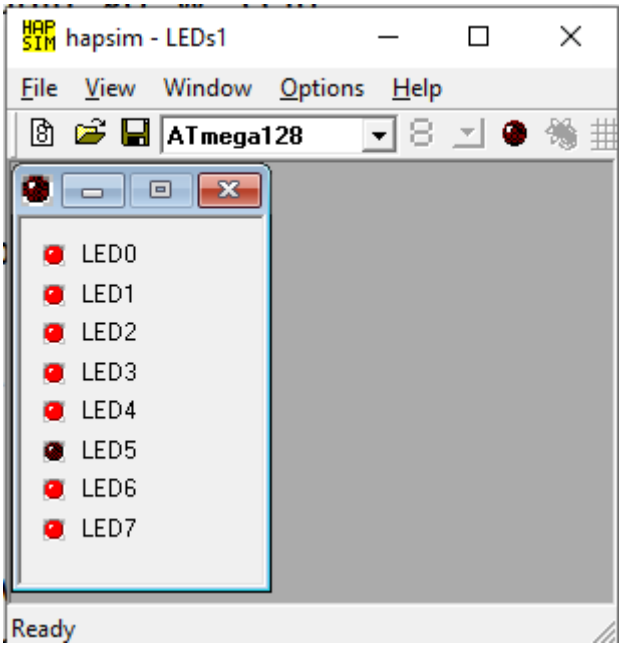
2. Opis zadania 1

Zadanie 1 polegało na przeklejeniu kodu z instrukcji a następnie zmodyfikowaniu go w celu umożliwienia działania tego programu na zestawie w laboratorium.

Zrealizowałem je poprzez:

1. Usunięcie zbędnej linijki " `#include <string.h>` "
2. Dopisanie instrukcji w linijce 10 " `asm volatile("nop");` ", która sprawiła, że pętla opóźniająca opóźniała działanie programu o zadaną liczbę milisekund.
3. Zmianę instrukcji w linijce 24 z postaci " `PORTB=0xff;` " na postać " `PORTB=0;` ", dzięki czemu listwa LED została naprawdę wygaszona (komentarz do tej linijki w instrukcji brzmiał "wygaś LED-y".
4. Usunąłem instrukcję " `PORTB=0xff;` " pod linijką 38, ponieważ to ustawienie portu zostaje nadpisane w pętli for z linijki 44 bez żadnego opóźnienia, przez co jest niedostrzegalne i zbędne.

Podczas wykonywania tego zadania napotkałem na problem ze strony zestawu obecnego w laboratorium. Pomimo zastosowania powyższych kroków, program zawieszał się na wykonywaniu pętli z linijki 7. Po wielokrotnych próbach znalezienia problemu poprzez zapalanie i gaszenie różnych LED-ów w różnych miejscach funkcji "czekaj", program nagle zaczął działać. Po przeklejeniu kodu z instrukcji i wprowadzeniu zmian z powyższych punktów program dalej działał pomimo tego, że ten sam kod wcześniej nie działał. Uważam, że to mogła być wina sprzętu, ponieważ na symulatorze program działał od razu i za każdym razem tak samo.



Zrzut ekranu z działającego programu w symulatorze.

3. Zadanie 2

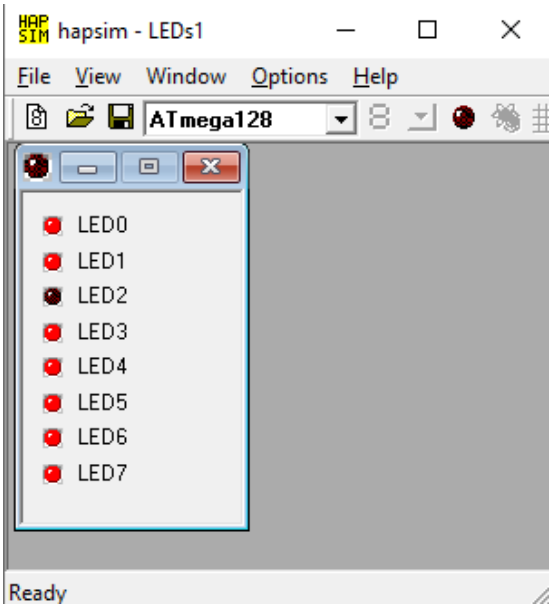
```c
#define F_CPU 1000000 // Ustalenie częstotliwości zegara na zgodny z zestawem w laboratorium
// Konieczne biblioteki
#include <avr/io.h> // Obsługa portów
#include <util/delay.h> // Obsługa opóźnień

unsigned long pczekaj=1500; // zmienna zawierająca wartość opóźnienia

int main(void) //program główny
{
    DDRB=0xff;  //konfiguracja wszystkich wyprowadzeń portu B jako wyjścia

    while(1)    //nieskończona pętla główna programu
    {
        //efekt węża
        for(unsigned char licznik=0;licznik<10;licznik++) //pętla długości trwania efektu (liczba cykli danego efektu)
        {
            PORTB=0;  //wygaś LED-y
            _delay_ms(pczekaj); //opóźnij o zadany czas
            for(int i=7;i>=0;i--) //pętla zmieniająca fazę efektu
            {
                PORTB|=_BV(i); //wysteruj (zapal) pojedynczego LED-a
                _delay_ms(pczekaj); //opóźnij o zadany czas
            }
            for(int i=7;i>=0;i--)   //pętla zmieniająca fazę efektu
            {
                PORTB&=~_BV(i); //wysteruj (zgaś) pojedynczego LED-a
                _delay_ms(pczekaj); //opóźnij o zadany czas
            }
        }

        //efekt biegnącego punktu
        for(unsigned char licznik=0;licznik<10;licznik++)  //pętla długości trwania efektu (liczba cykli danego efektu)
        {
            for(unsigned char ledy=0x7f;ledy!=0xff;ledy=(ledy>>1) + 128) //pętla zmieniająca fazę efektu
            {
                PORTB=ledy; //wysterowanie LED-ów zgodne z wartością zmiennej ledy
                _delay_ms(pczekaj); //opóźnij o zadany czas
            }
        }
    }
}
```

4. Opis zadania 2

Zadanie 2 polegało na modyfikacji kodu z instrukcji w celu zamienienia pętli opóźniającej na wywołania metody “ `_delay_ms(pczekaj);` ”, oraz na odwróceniu kierunku działania animacji.

Zrealizowałem je poprzez:

1. Dodanie linijki " `#include <util/delay.h>` " na początku programu w celu umożliwienia używania metody “ `_delay_ms(pczekaj);` “.
2. Usunięcie metody “ `void czekaj(unsigned long pt)` ”.
3. Zamienienie wszystkich wywołań metody “ `void czekaj(unsigned long pt)` ” na wywołania funkcji “ `_delay_ms(pczekaj);` “.
4. Zamienienie deklaracji pętli w linijkach 19 i 24 z wyrażenia “ `for(i=0;i<8;i++)` “ na “ `for(int i=7;i>=0;i--)` ”, co powoduje przesuwanie się “efektu węża” w przeciwną stronę.
5. Zamienienie deklaracji w pętli w linijce 34 z wyrażenia
   “ `for(ledy=0xfe;ledy!=0xff;ledy=(ledy<<=1)+1)` “
   na
   “ `for(unsigned char ledy=0x7f;ledy!=0xff;ledy=(ledy>>1) + 128)` “.
   Ta pętla działa w następujący sposób:
   Zaczynam z wartością zmiennej ledy = 0b0111111.
   Następnie tę zmienną przesuwam bitowo w lewo i otrzymuję ledy = 0b0011111.
   Po dodaniu liczby 128 (czyli bitowo 0b10000000) otrzymuję ledy = 0b1011111.
   Znowu przesuwam w lewo i mam ledy = 0b0101111.
   Po ponownym dodaniu 128 otrzymuję ledy = 0b1101111.
   W ten sposób realizuję “efekt biegnącego punktu”.



Zrzut ekranu z działającego programu w symulatorze.

5. Wnioski

Podczas wykonywania tego ćwiczenia dowiedziałem się, że problemy z działaniem programu nie muszą oznaczać problemów z kodem, a problemy np. z programatorem lub samym układem. Nauczyłem się sterować stanami portu B poprzez wyświetlanie za jego pomocą różnych wzorów na linijce LED-ów. Poznałem zastosowanie biblioteki “<util/delay.h>” oraz funkcji “_delay_ms()”.