SPRAWOZDANIE

SYSTEMY WBUDOWANE

Układy peryferyjne mikrokontrolera AVR. Porty.

IMIĘ I NAZWISKO: Jacek Wójcik

NUMER ĆWICZENIA: 3

Grupa laboratoryjna: 10

Data wykonania ćwiczenia: 27.10.2021

Spis treści

1. Zadanie 1 - kod

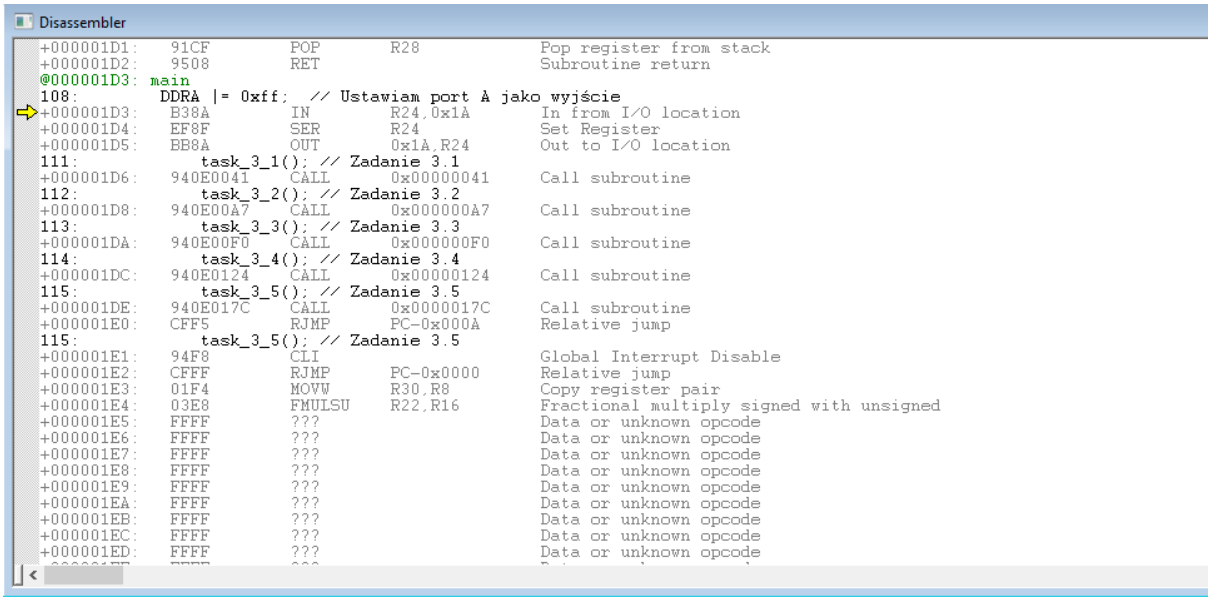
1.1 Kod w C

```c
#define F_CPU 1000000  //Deklaruję częstotliwość zgodną z zestawem w laboratorium
// Konieczne biblioteki
#include <avr/io.h> //Biblioteka do portów
#include <util/delay.h> // Biblioteka do opóźnień

const int half_second = 500, second = 1000; // Zmienne przechowujące wartości opóźnień

void task_3_1()  // Zadanie 3.1
{
    PORTA |= 0xc0;  // Ustawiam stan PORTA na początek animacji
    PORTA &= 0xc0;   // Zeruję wszystkie pozostałe bity
    for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
    {
        for(int j = 7; j > 1; j--)  // Dopóki nie dojdę do końca linijki zapalam kolejne LED-y
        {
            _delay_ms(half_second); // Czekam 0,5s
            PORTA ^= _BV(j) | _BV(j-2); // Przesuwam animację
        }
        for(int j = 0; j < 6; j++) // Dopóki nie wrócę na początek linijki zapalam kolejne LED-y
        {
            _delay_ms(half_second); // Czekam 0,5s
            PORTA ^= _BV(j) | _BV(j+2); // Przesuwam animację
        }
    }
}

void task_3_2()  // Zadanie 3.2
{
    PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
    PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
    for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
    {
        for(int j = 6; j > 3; j--)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
        {
            _delay_ms(second); // Czekam 1s
            PORTA ^= _BV(j); // Przesuwam animację
        }
        for(int j = 4; j < 7; j++) // Dopóki nie wrócę do punktu z którego zacząłem gaszę kolejne LED-y
        {
            _delay_ms(second); // Czekam 1s
            PORTA ^= _BV(j); // Przesuwam animację
        }
    }
}

```

```
45
46  void task_3_3()  // Zadanie 3.3
47  {
48      for(int i=0; i < 3; i++) // Wykonuję 3 powtórzenia
49      {
50          PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
51          PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
52          _delay_ms(second); // Czekam 1s
53          for(int j = 6; j > -1; j--)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
54          {
55              PORTA ^= _BV(j); // Przesuwam animację
56              _delay_ms(second); // Czekam 1s
57          }
58      }
59  }
60
61  void task_3_4()  // Zadanie 3.4
62  {
63      for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
64      {
65          PORTA |= 0x08;  // Ustawiam stan PORTA na początek animacji
66          PORTA &= 0x08;   // Zeruję wszystkie pozostałe bity
67          _delay_ms(second); // Czekam 1s
68          for(int j = 2; j > -1; j--)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
69          {
70              PORTA ^= _BV(j); // Przesuwam animację
71              _delay_ms(second); // Czekam 1s
72          }
73          PORTA ^= 0x1f;  //Odwracam zapalone LED-y w celu przejścia do kolejnego kroku animacji
74
75          _delay_ms(second); // Czekam 1s
76          for(int j = 5; j < 8; j++)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
77          {
78              PORTA ^= _BV(j); // Przesuwam animację
79              _delay_ms(second); // Czekam 1s
80          }
81      }
82  }
83
```

```
83
84  void task_3_5() // Zadanie 3.5
85  {
86      for(int i=0; i < 3; i++) // Wykonuję 3 powtórzenia
87      {
88          PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
89          PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
90          _delay_ms(second); // Czekam 1s
91          for(int j = 6; j > 3; j--)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
92          {
93              PORTA ^= _BV(j); // Przesuwam animację
94              _delay_ms(second); // Czekam 1s
95          }
96          PORTA ^= 0xf1; //Odwracam zapalone LED-y w celu przejścia do kolejnego kroku animacji
97          _delay_ms(second); // Czekam 1s
98          for(int j = 1; j <4; j++)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
99          {
100             PORTA ^= _BV(j); // Przesuwam animację
101             _delay_ms(second); // Czekam 1s
102         }
103     }
104 }
105
106 int main(void)
107 {
108     DDRA |= 0xff;  // Ustawiam port A jako wyjście
109     while(1) // Nieskończona pętla
110     {
111         task_3_1(); // Zadanie 3.1
112         task_3_2(); // Zadanie 3.2
113         task_3_3(); // Zadanie 3.3
114         task_3_4(); // Zadanie 3.4
115         task_3_5(); // Zadanie 3.5
116     }
117 }
118
```

1.2 Zdeasemblowany kod funkcji main()



2. Opis zadania 1

Zadanie 1 polegało na zaprogramowaniu pięciu różnych animacji na linijce LED używając portu A mikrokontrolera. **Należało to wykonać używając jedynie instrukcji selektywnego ustawienia, zerowania i negowania.**

dołączono linijkę diodową D0…D7. Korzystając wyłącznie z operacji selektywnego: ustawienia, zerowania i negowania napisz kolejne programy w języku C dla poniższych

Wykonałem je używając podejścia opartego o wydzielenie oddzielnej funkcji dla każdej animacji a następnie wywołanie tych funkcji w nieskończonej pętli w funkcji `main(void)`. Poniżej opiszę poszczególne części mojego kodu zaczynając od instrukcji preprocesora, przez funkcję `main(void)` po opis poszczególnych animacji.

2.1 Instrukcje preprocesora

```
1   #define F_CPU 1000000  //Deklaruję częstotliwość zgodną z zestawem w laboratorium
2   // Konieczne biblioteki
3   #include <avr/io.h> //Biblioteka do portów
4   #include <util/delay.h> // Biblioteka do opóźnień
5
6   const int half_second = 500, second = 1000; // Zmienne przechowujące wartości opóźnień
7
```

W tej części programu ustalam częstotliwość zegara, która mówi funkcji “ `_delay_ms()` “ ile cykli zegara składa się na jedną sekundę, co pozwala na upewnienie się, że po wywołaniu funkcji “ `_delay_ms(500);` “ mikrokontroler naprawdę zaczeka 500ms.
Następnie dołączam dwie konieczne biblioteki:
“ `#include <avr/io.h>` ” - obsługa portów mikrokontrolera
“ `#include <util/delay.h>` “ - możliwość używania funkcji “ `_delay_ms()` “
Na samym końcu tworzę dwie zmienne, “ `half_second` ” i “ `second` ” w celu przechowywania wartości opóźnień i usprawnienia czytelności kodu.

2.2 Funkcja main()

```
105
106  int main(void)
107  {
108      DDRA |= 0xff;  // Ustawiam port A jako wyjście
109      while(1) // Nieskończona pętla
110      {
111          task_3_1(); // Zadanie 3.1
112          task_3_2(); // Zadanie 3.2
113          task_3_3(); // Zadanie 3.3
114          task_3_4(); // Zadanie 3.4
115          task_3_5(); // Zadanie 3.5
116      }
117  }
118
```

W funkcji main() najpierw ustalam port A jako wyjście za pomocą instrukcji “ `DDRA = 0xff;` ”. Dzięki temu mogę zmieniać stan portów za pomocą zmiennej “ `PORTA` “.
Drugą częścią funkcji main() jest pętla “ `while(1)` ”, która ma za zadanie wyświetlać wszystkie pięć animacji. Realizuję ją poprzez wywołania funkcji “ `task_3_X()` ”, gdzie X to numer animacji (np. “ `task_3_1()` ” to animacja nr. 1).

2.3 Animacja 1

```
7
8   void task_3_1()  // Zadanie 3.1
9   {
10      PORTA |= 0xc0;  // Ustawiam stan PORTA na początek animacji
11      PORTA &= 0xc0;   // Zeruję wszystkie pozostałe bity
12      for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
13      {
14          for(int j = 7; j > 1; j--)  // Dopóki nie dojdę do końca linijki zapalam kolejne LED-y
15          {
16              _delay_ms(half_second); // Czekam 0,5s
17              PORTA ^= _BV(j) | _BV(j-2); // Przesuwam animację
18          }
19          for(int j = 0; j < 6; j++) // Dopóki nie wrócę na początek linijki zapalam kolejne LED-y
20          {
21              _delay_ms(half_second); // Czekam 0,5s
22              PORTA ^= _BV(j) | _BV(j+2); // Przesuwam animację
23          }
24      }
25  }
26
```

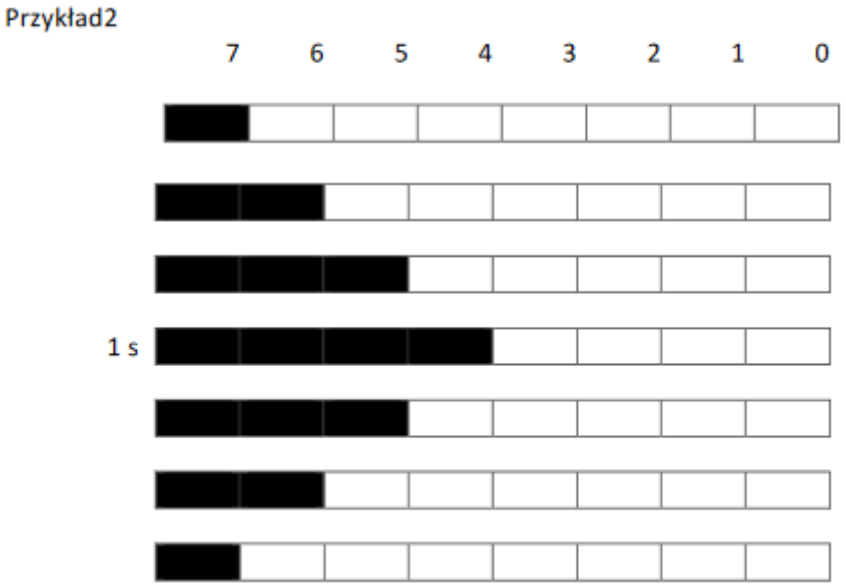
Ta animacja polega na zapaleniu dwóch ostatnich LED-ów a następnie “przesuwanie ich” w stronę dwóch pierwszych LED-ów a następnie powrót do dwóch ostatnich LED-ów.

Realizuję to zaczynając od ustawienia portu A na stan początkowy animacji - selektywnie włączam dwa ostatnie ledy instrukcją “ `PORTA |= 0xc0;` “ a następnie gaszę wszystkie inne, zerując je instrukcją “ `PORTA &= 0xc0;` ”. Następnie znajduje się deklaracja pętli for, która ma za zadanie wykonać trzy powtórzenia animacji. Jest to instrukcja “ `for(int i=0; i < 3; i++)` ”. W tej pętli znajdują się dwie podobne pętle for. Pierwsza z nich ma postać “ `for(int j = 7; j > 1; j--)` ” i zawiera instrukcję opóźniającą “ `_delay_ms(half_second);` “, w celu umożliwienia zauważenia stanu animacji, a następnie znajduje się instrukcja “ `PORTA ^= _BV(j) | _BV(j-2);` “, która ma za zadanie przesunąć animację poprzez zanegowanie ostatniego włączonego LED-a i pierwszego wyłączonego LED-a w animacji (czyli dla pierwszej iteracji jest to operacja 0b11000000 ^ 0b10100000 = 0b01100000). Druga pętla for działa analogicznie do pierwszej, przesuwając w identyczny sposób animację w lewo.

2.4 Animacja 2

```
26
27  void task_3_2()  // Zadanie 3.2
28  {
29      PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
30      PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
31      for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
32      {
33          for(int j = 6; j > 3; j--)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
34          {
35              _delay_ms(second); // Czekam 1s
36              PORTA ^= _BV(j); // Przesuwam animację
37          }
38          for(int j = 4; j < 7; j++) // Dopóki nie wrócę do punktu z którego zacząłem gaszę kolejne LED-y
39          {
40              _delay_ms(second); // Czekam 1s
41              PORTA ^= _BV(j); // Przesuwam animację
42          }
43      }
44  }
45
```

Ta animacja polega na utworzeniu efektu equalizera - “dojście” LED-ami z końca do środka linijki i powrót do końca.

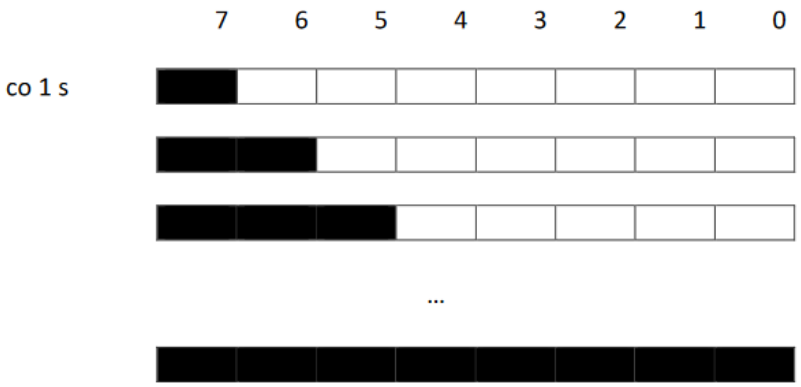


Realizuję to zaczynając od ustawienia portu A na stan początkowy animacji - selektywnie włączam ostatniego LED-a instrukcją “ `PORTA |= 0x80;` “ i gaszę wszystkie pozostałe, zerując je instrukcją “ `PORTA &= 0x80;` ”. Następnie znajduje się deklaracja pętli for, która ma za zadanie wykonać trzy powtórzenia animacji. Jest to instrukcja “ `for(int i=0; i < 3; i++)` ”. W tej pętli znajdują się dwie podobne pętle for. Pierwsza z nich ma postać “ `for(int j = 6; j > 3; j--)` “ i służy do zapalania kolejnych LED-ów aż dojdę do środka linijki. Realizuję to instrukcją “ `PORTA ^= _BV(j);` “, która neguje następny bit, a jako że wszystkie LED-y poza 7 były wyłączone i zaczynamy od LED-a 6 to w ten sposób włączamy te LED-y. Następnie opóźniam pętlę instrukcją “ `_delay_ms(half_second);` “. Druga pętla ma analogiczną budowę do pierwszej, jednak zaczynamy od LED-a 4 i idziemy do LED-a 6 negując ich stan, co powoduje ich wyłączenie.

2.5 Animacja 3

```
45
46  void task_3_3()  // Zadanie 3.3
47  {
48      for(int i=0; i < 3; i++) // Wykonuję 3 powtórzenia
49      {
50          PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
51          PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
52          _delay_ms(second); // Czekam 1s
53          for(int j = 6; j > -1; j--)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
54          {
55              PORTA ^= _BV(j); // Przesuwam animację
56              _delay_ms(second); // Czekam 1s
57          }
58      }
59  }
60
```

Ta animacja polegała na zapaleniu wszystkich LED-ów od ostatniego do pierwszego z krokiem 1.

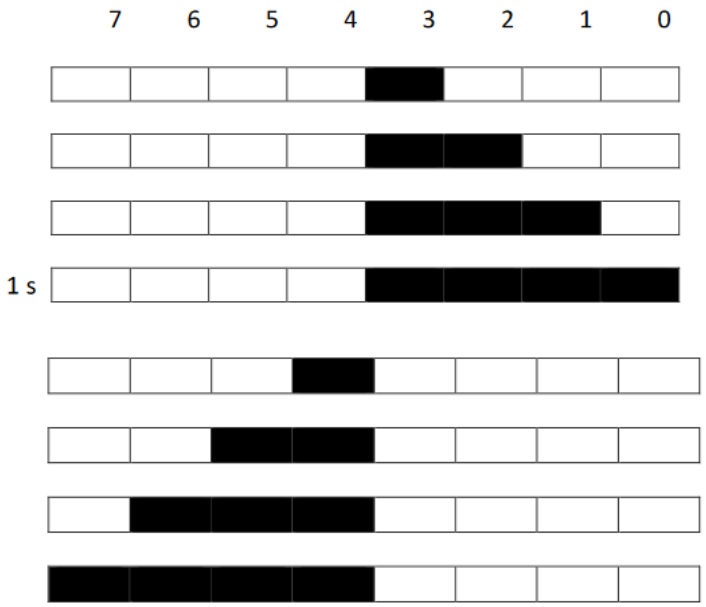


Zrealizowałem ją jako zagnieżdżoną pętlę for. Zewnętrzna pętla for ma postać “ `for(int i=0; i < 3; i++)` “, dzięki czemu cała animacja powtarza się trzy razy. W środku pętli najpierw selektywnie włączam ostatniego LED-a instrukcją “ `PORTA |= 0x80;` ” a następnie gaszę wszystkie pozostałe instrukcją “ `PORTA &= 0x80;` “, która powoduje wyzerowanie wszystkich bitów poza ostatnim. Po inicjalizacji animacji czekam sekundę żeby można było zauważyć stan LED-ów. W następnej części pętli znajduje się zagnieżdżona pętla for “ `for(int j = 6; j > -1; j--)` “, która w każdej iteracji neguje stan pojedynczego LED-a za pomocą instrukcji “ `PORTA ^= _BV(j);` “. Jako że wszystkie LED-y poza ostatnim są wyłączone i zaczynam fora od przedostatniego LED-a, ta instrukcja włącza danego LED-a. Po zmianie stanu linijki LED czekam sekundę w celu umożliwienia zauważenia zmiany.

2.6 Animacja 4

```
60
61  void task_3_4()  // Zadanie 3.4
62  {
63      for(int i=0; i < 3; i++)  // Wykonuję 3 powtórzenia
64      {
65          PORTA |= 0x08;  // Ustawiam stan PORTA na początek animacji
66          PORTA &= 0x08;   // Zeruję wszystkie pozostałe bity
67          _delay_ms(second); // Czekam 1s
68          for(int j = 2; j > -1; j--)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
69          {
70              PORTA ^= _BV(j); // Przesuwam animację
71              _delay_ms(second); // Czekam 1s
72          }
73          PORTA ^= 0x1f;  //Odwracam zapalone LED-y w celu przejścia do kolejnego kroku animacji
74
75          _delay_ms(second); // Czekam 1s
76          for(int j = 5; j < 8; j++)  // Dopóki nie dojdę do jednego z końców linijki zapalam kolejne LED-y
77          {
78              PORTA ^= _BV(j); // Przesuwam animację
79              _delay_ms(second); // Czekam 1s
80          }
81      }
82  }
83
```

Ta animacja polega na stopniowym zapaleniu połowy linijki zaczynając najpierw od 3 LED-a i idąc do LED-a 0, a następnie wygaszeniu linijki i ponownym zapaleniu połowy, tym razem zaczynając od LED-a 4 i idąc do LED-a 7.

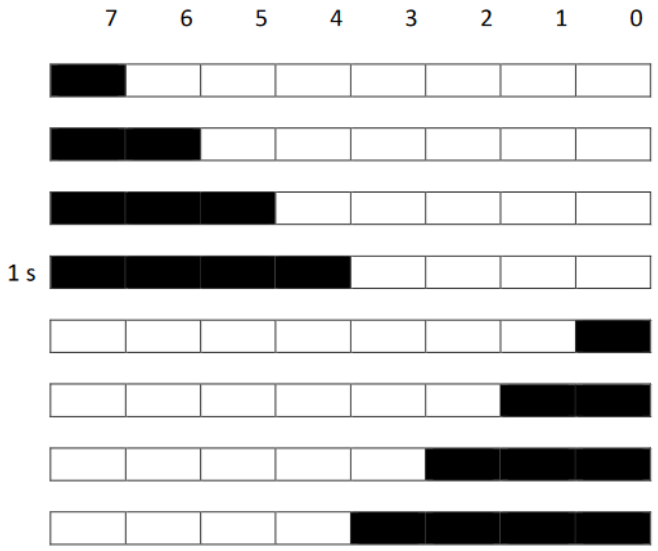


Zrealizowałem je identycznie jak wcześniejsze trzy: najpierw jest pętla for, która powoduje trzy powtórzenia animacji, następnie ustawiam stan PORTA na początek animacji włączając odpowiedniego LED-a i negując pozostałe. Po tym jest pierwsza zagnieżdżona pętla for o postaci “ `for(int j = 2; j > -1; j--)` ”, która zapala LEDy od 2 do 0 za pomocą wybiórczej negacji (tak jak w animacji 3, instrukcją “ `PORTA ^= _BV(j);` ”) i czeka sekundę po każdej zmianie. Po pierwszej pętli wybiórczo neguję LED-y od 0 do 4 za pomocą instrukcji “ `PORTA ^= 0x1f;` ” w celu przejścia do kolejnej fazy animacji. Kolejna zagnieżdżona pętla for o postaci “ `for(int j = 5; j < 8; j++)` ” zapala LED-y od 4 do 7 w identyczny sposób jak pierwsza, za pomocą instrukcji “ `PORTA ^= _BV(j);` ”.

2.7 Animacja 5

```
83
84  void task_3_5() // Zadanie 3.5
85  {
86      for(int i=0; i < 3; i++) // Wykonuję 3 powtórzenia
87      {
88          PORTA |= 0x80;  // Ustawiam stan PORTA na początek animacji
89          PORTA &= 0x80;   // Zeruję wszystkie pozostałe bity
90          _delay_ms(second); // Czekam 1s
91          for(int j = 6; j > 3; j--)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
92          {
93              PORTA ^= _BV(j); // Przesuwam animację
94              _delay_ms(second); // Czekam 1s
95          }
96          PORTA ^= 0xf1; //Odwracam zapalone LED-y w celu przejścia do kolejnego kroku animacji
97          _delay_ms(second); // Czekam 1s
98          for(int j = 1; j <4; j++)  // Dopóki nie dojdę do środka linijki zapalam kolejne LED-y
99          {
100             PORTA ^= _BV(j); // Przesuwam animację
101             _delay_ms(second); // Czekam 1s
102         }
103     }
104 }
105
```

Ta animacja polega na zapaleniu połowy linijki LED najpierw zaczynając od LED 7 i idąc do LED 4 a następnie zgaszeniu wszystkich zapalonych LED-ów i zapaleniu połowy linijki LED zaczynając od LED 0 i idąc do LED 3.



Zrealizowałem tę animację podobnie jak animację 4, zaczynając od pętli for realizującej 3 powtórzenia animacji, następnie ustawiając stan PORTA na włączonego LED-a 7 za pomocą instrukcji selektywnego ustawiania “ `PORTA |= 0x80;` ” i wyłączenia pozostałych ledów instrukcją “ `PORTA &= 0x80;` ”, następnie jest pierwsza pętla for “ `for(int j = 6; j > 3; j--)` ”, działająca podobnie do tej z animacji 4, włączając LED-y od 6 do 4 (LED 7 jest włączany podczas inicjalizacji animacji) i czekając sekundę po każdym zmianie stanu. Następnie wybiórczą negacją “ `PORTA ^= 0xf1;` “ zmieniam fazę animacji gasząc wszystkie zapalone LED-y i włączając LED 0. W kolejnej pętli for “ `for(int j = 1; j <4; j++)` “ zapalam LED-y od 1 do 3 czekając sekundę po każdej zmianie stanu linijki LED.

3. Wnioski

Podczas tych laboratoriów nauczyłem się korzystać z portów mikrokontrolera za pomocą instrukcji wybiórczej negacji (“ ^= “), selektywnego ustawiania (“ |= “) oraz zerowania maską(“ &= “). Przećwiczyłem również używanie pętli for oraz funkcji “ _BV(i) “. Za pomocą deasemblera zobaczyłem jak mój kod jest tłumaczony na język avr assmembly.