

Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:

Systemy wbudowane

Sprawozdanie z ćwiczenia laboratoryjnego nr 2:
Wytwarzanie i testowanie oprogramowania.

Prowadzący:

mgr inż. Artur Miktus

Wykonał: Radosław Relidzyński

Grupa: WCY20IY4S1

Data laboratoriów: 13.05.2022 r.

Deklarowana ocena: 3, 4, 5

Spis treści

A.	Treść zadania	2
	<i>Zadanie na Lab 1 SWB</i>	Błąd! Nie zdefiniowano zakładki.
B.	Zadanie na ocenę dostateczną	5
	Opis mojego rozwiązania.....	5
	Schemat blokowy rozwiązania	6
	Listing programu.....	8
	Sprawdzenie poprawności.....	9
	Prezentacja realizacji zadania przez program	10
C.	Zadanie na ocenę dobrą	Błąd! Nie zdefiniowano zakładki.
	Opis mojego rozwiązania.....	Błąd! Nie zdefiniowano zakładki.
	Schemat blokowy rozwiązania	Błąd! Nie zdefiniowano zakładki.
	Listing programu.....	Błąd! Nie zdefiniowano zakładki.
	Sprawdzenie poprawności.....	Błąd! Nie zdefiniowano zakładki.

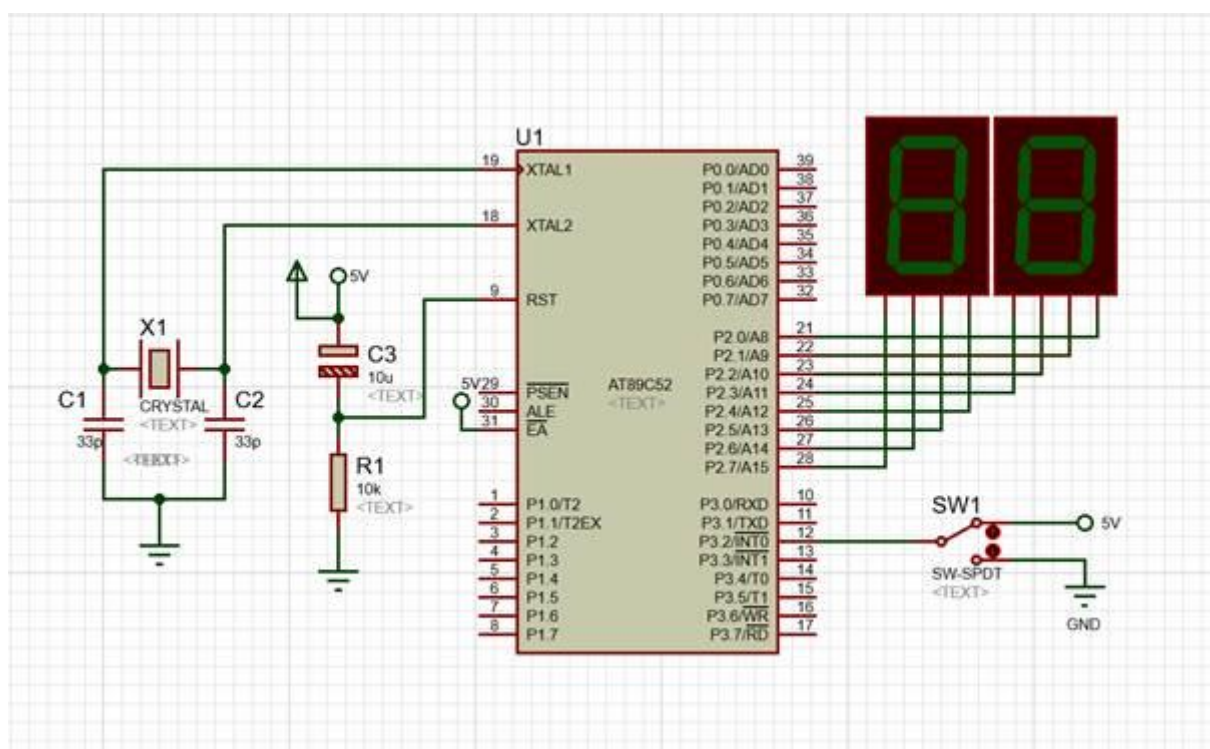
- Prezentacja realizacji zadania przez program**Błąd! Nie zdefiniowano zakładki.**
- D. Zadanie na ocenę bardzo dobrą**Błąd! Nie zdefiniowano zakładki.**
- Opis mojego rozwiązania.....**Błąd! Nie zdefiniowano zakładki.**
- Schemat blokowy rozwiązania**Błąd! Nie zdefiniowano zakładki.**
- Listing programu.....**Błąd! Nie zdefiniowano zakładki.**
- Sprawdzenie poprawności.....**Błąd! Nie zdefiniowano zakładki.**
- Prezentacja realizacji zadania przez program**Błąd! Nie zdefiniowano zakładki.**

A. Treść zadania

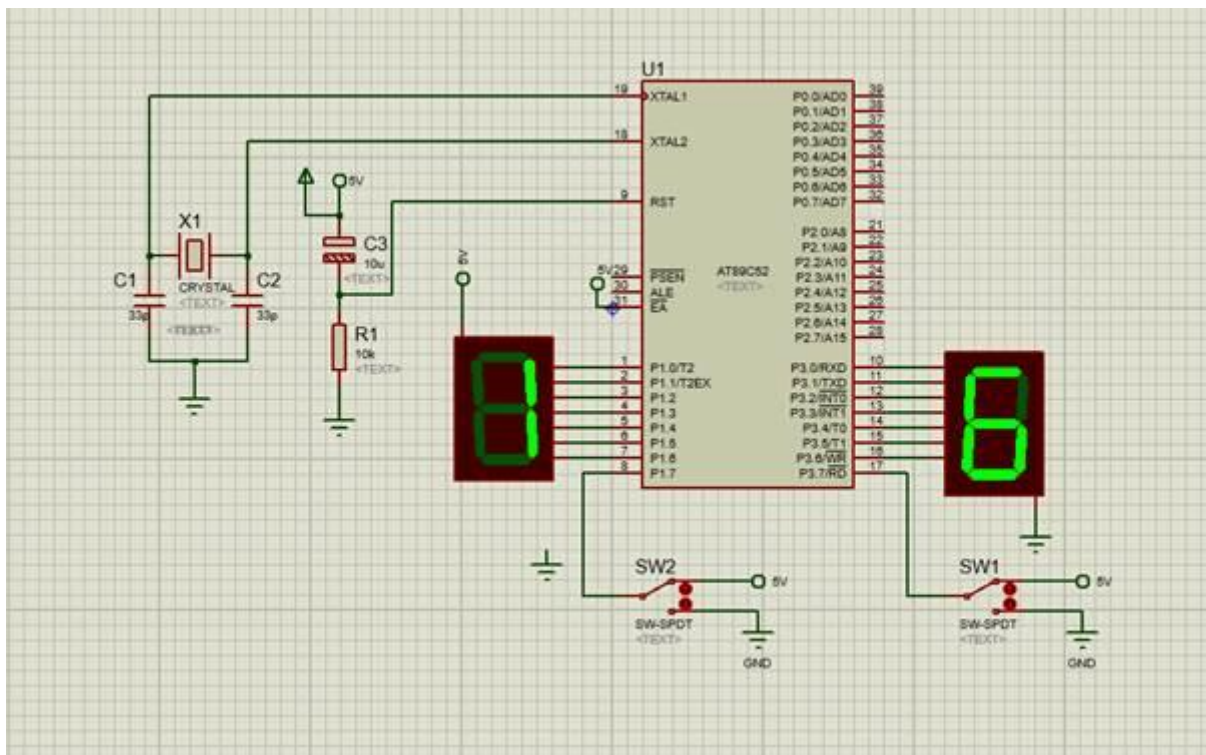
Zadanie na laboratorium nr 2.

Dane są schematy układów jak na rysunkach:

[Zad 2.pdsprj](#): gdzie oba wyświetlacze są typu 7seg-bcd-grn



[Zad 3.pdsprj](#): gdzie wyświetlacz prawy to układ ze wspólną katodą a wyświetlacz lewy to układ ze wspólną anodą.



Zadania laboratoryjne:

1. Na ocenę **dostatecznie** napisać dla schematu **Zad_2.pdsprj** program **lab2_1.c** w języku C, który w pętli wewnętrznej będzie kolejno, **startując od "FF" na wyświetlaczach**:
 - a. Dla studentów o numerze w dzienniku nieparzystym
 - i. Zwiększał stan zmiennej „**zmiennaAA**” od zera do granicy = $(5 + \text{numer w dzienniku})$ z **krokiem o 1**;
 - ii. Jeśli **zmiennaAA** będzie **nieparzysta**, jej stan wyświetli w **postaci liczby dziesiętnej na obu** wyświetlaczach 7-segmentowych „przez chwilę”; w przeciwnym razie pozostawia na wyświetlaczach poprzednią wartość (FF, 01, 03, 05, ..., FF, 01, 03, 05, ...). Po osiągnięciu wartości granicznych odpowiednich zmiennych mają one zostać wyzerowane i programy powtarzają swoje działanie. Stan SW1 nie ma wpływu na zachowanie programu.
 - b. Dla studentów o numerze w dzienniku parzystym
 - i. Zwiększał stan zmiennej „**zmiennaBB**” od zera do granicy = $(6 + \text{numer w dzienniku})$ z **krokiem o 2**;
 - ii. Jeśli **zmiennaBB** będzie **podzielna przez 4**, jej stan wyświetli **w postaci liczby dziesiętnej na obu** wyświetlaczach 7-segmentowych „przez chwilę”; w przeciwnym razie pozostawia na wyświetlaczach poprzednią wartość (FF, 04, 08, 12, ..., FF, 04, 08, 12, ...). Po osiągnięciu wartości granicznych odpowiednich zmiennych mają one zostać wyzerowane i programy powtarzają swoje działanie. Stan SW1 nie ma wpływu na zachowanie programu.

W obu wersjach określenie „przez chwilę” oznacza taki odcinek czasu, aby wygodnie obserwować poprawność działania napisanego programu.

Uwaga: zmienna zliczająca zmiennaAA (nieparzyści) i zmiennaBB (parzyści) ma zmieniać swój stan od zera do granicy, **np. dla studenta o numerze 19 ma przechodzić od zera do $19 + 5 = 24$ z krokiem o 1 a dla studenta o numerze 20 zmiennaBB ma przechodzić przez wartości od zera do $20 + 6 = 26$ z krokiem co 2**, a działanie podejmować po spełnieniu odpowiedniego warunku. Zmiany stanu programu przenoszą się czasami na wyświetlacze.

2. Na ocenę **dobrze** zrobić to, co na **dostatecznie, oraz ponadto** napisać dla schematu **Zad_2.pdsprj** program **lab2_2.c** w języku C, który **startując od "FF" na wyświetlaczach** w pętli wewnętrznej będzie kolejno **na podstawie zliczania przełączeń przełącznika SW1 ze stanu „0” na „1” (nie "automatycznie", jak to było w zadaniu na dostatecznie)**:

- a. Dla studentów o numerze w dzienniku nieparzystym
- Zwiększał stan zmiennej „**zmiennaAA**” od zera do (5+numer w dzienniku) z **krokiem o 1**;
 - Jeśli **zmiennaAA** będzie **nieparzysta**, **stan jej mniej znaczącej cyfry dziesiętnej wyświetli w postaci liczby dziesiętnej** na **prawym wyświetlaczu** 7-segmentowym, w tym samym czasie na lewym wyświetlaczu ma być wyświetlana **najmniej znacząca cyfra dziesiętna numeru w dzienniku wykonawcy zadania**; w przeciwnym razie pozostawia na wyświetlaczach poprzednią pokazywaną wartość. Po osiągnięciu (albo, jeśli to niemożliwe, po przekroczeniu) wartości granicznych odpowiednich zmiennych mają one zostać wyzerowane, **wyświetlacze pokazują "FF"** i programy powtarzają swoje działanie.
- b. Dla studentów o numerze w dzienniku parzystym
- Zwiększał stan zmiennej „**zmiennaBB**” od zera do (6+numer w dzienniku) z **krokiem o 2**;
 - Jeśli **zmiennaBB** będzie **podzielna przez 4**, **stan jej mniej znaczącej cyfry dziesiętnej wyświetli w postaci liczby dziesiętnej** na **prawym wyświetlaczu** 7-segmentowym, w tym samym czasie na lewym wyświetlaczu ma być wyświetlana **najmniej znacząca cyfra dziesiętna numeru w dzienniku wykonawcy zadania**; w przeciwnym razie pozostawia na wyświetlaczach poprzednią pokazywaną wartość. Po osiągnięciu (albo, jeśli to niemożliwe, po przekroczeniu) wartości granicznych odpowiednich zmiennych mają one zostać wyzerowane, **wyświetlacze pokazują "FF"** i programy powtarzają swoje działanie.

Uwaga: zmienna zliczająca zmiennaAA (nieparzyści) i zmiennaBB (parzyści) ma zmieniać swój stan od zera do granicy, **np. dla studenta o numerze 19 ma przechodzić od zera do $19+5=24$ z krokiem o 1 a dla studenta o numerze 20 zmiennaBB ma przechodzić przez wartości od zera do $20+6=26$ z krokiem co 2**, a działanie podejmować po spełnieniu odpowiedniego warunku. Zmiany stanu programu przenoszą się czasami na wyświetlacze.

-
3. Na ocenę **bardzo dobrze** zrobić to, co na **dobrze, oraz ponadto** napisać dla schematu **Zad_3.pdsprj** program **lab2_3.c** w języku C, który w pętli wewnętrznej będzie kolejno:

- a. Dla studentów o numerze w dzienniku nieparzystym dla SW1 = 1 zwiększał, a dla SW1 = 0 zmniejszał stan „**zmiennaAA**” z **krokiem o 1**. Stan początkowy zmiennaAA to zero, po przełączeniach SW1 zawartość zmiennej jest przez chwilę utrzymywana, zmienia się kierunek zmian zmiennaAA (rośnie albo odpowiednio maleje). Stan maksymalny zmiennaAA, po przekroczeniu którego zmiennaAA jest zerowana to **10 plus numer w dzienniku** wykonawcy. Przy zmniejszaniu się stanu zmiennaAA po osiągnięciu zera ma nastąpić "przekręcenie" wartości do maksymalnej i dalsze zmniejszanie. Aktualny stan zmiennaAA ma być "przez chwilę" wyświetlany poprawnie na obu wyświetlaczach siedmiosegmentowych w konwencji (**cyfra mniej znacząca na prawym**) i (**cyfra bardziej znacząca na lewym**).
- b. Dla studentów o numerze w dzienniku parzystym dla SW2 = 1 zwiększał, a dla SW2 = 0 zmniejszał stan „**zmiennaBB**” z **krokiem o 1**. Stan początkowy zmiennaBB to zero, po przełączeniach SW2 zawartość zmiennej jest przez chwilę utrzymywana, zmienia się kierunek zmian zmiennaBB (rośnie albo odpowiednio maleje). Stan maksymalny zmiennaBB, po przekroczeniu którego zmiennaBB jest zerowana to **12 plus numer w dzienniku** wykonawcy. Przy zmniejszaniu się stanu zmiennaBB po osiągnięciu zera ma nastąpić "przekręcenie" wartości do maksymalnej i dalsze zmniejszanie. Aktualny stan zmiennaBB ma być „przez chwilę” wyświetlany poprawnie na obu wyświetlaczach

siedmiosegmentowych w konwencji (cyfra mniej znacząca na lewym) i (cyfra bardziej znacząca na prawym).

W obu wersjach określenie „przez chwilę” oznacza taki odcinek czasu, aby wygodnie obserwować poprawność działania napisanego programu.

Ponieważ czasem studenci mają problem z określeniem zachowania programu przy zmniejszaniu, podkreślam:

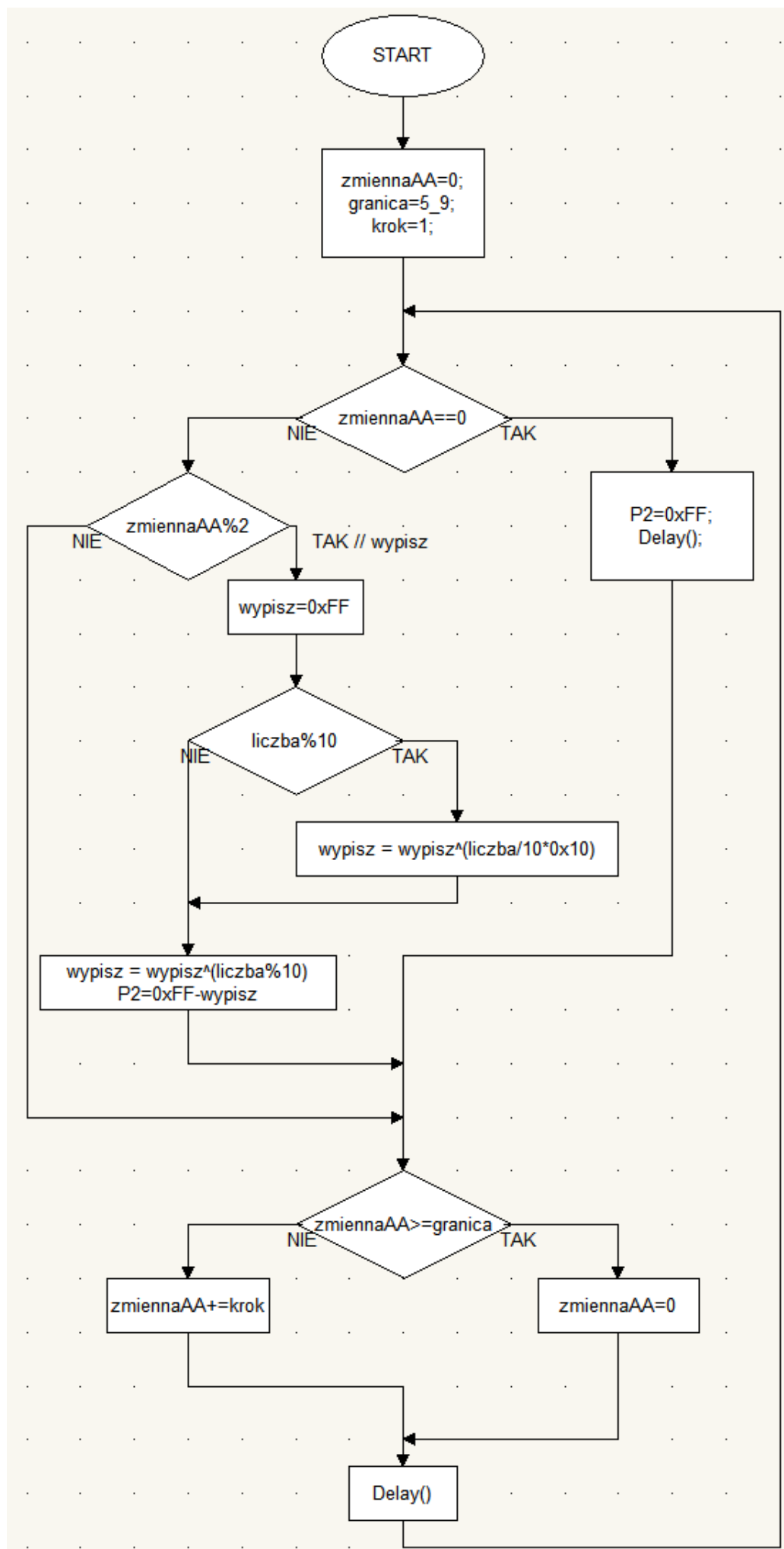
Przy kierunku malejących zmiennych po osiągnięciu wartości zero odpowiednich zmiennych **mają one zostać ustawiane na wartości odpowiednie maksymalne** i programy powtarzają swoje działanie. Uwaga: zmienna zliczająca zmiennaAA (nieparzyści) i zmiennaBB (parzyści) ma zmieniać swój stan od zera do granicy, **np. dla studenta o numerze 19 ma przechodzić od zera do $19+10=29$ z krokiem o 1 a dla studenta o numerze 20 zmiennaBB ma przechodzić przez wartości od zera do $20+12=32$ z krokiem co 1.** Zmiany stanu programu przenoszą się bezwarunkowo na wyświetlacze.

B. Zadanie na ocenę dostateczną

Opis mojego rozwiązania

W ramach tego zadania początkowo wyświetlane będzie „FF” symbolizujące brak wyświetlania liczby. Z każdą kolejną iteracją zmiennaAA będzie rosła od 0 o 1 do momentu osiągnięcia liczby maksymalnej 14, wtedy zamyka cykl zmieniając wartość na 0. Na wyświetlacz zostanie przekazana liczba dopiero w momencie, gdy zmiennaAA będzie miała wartość nieparzystą.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

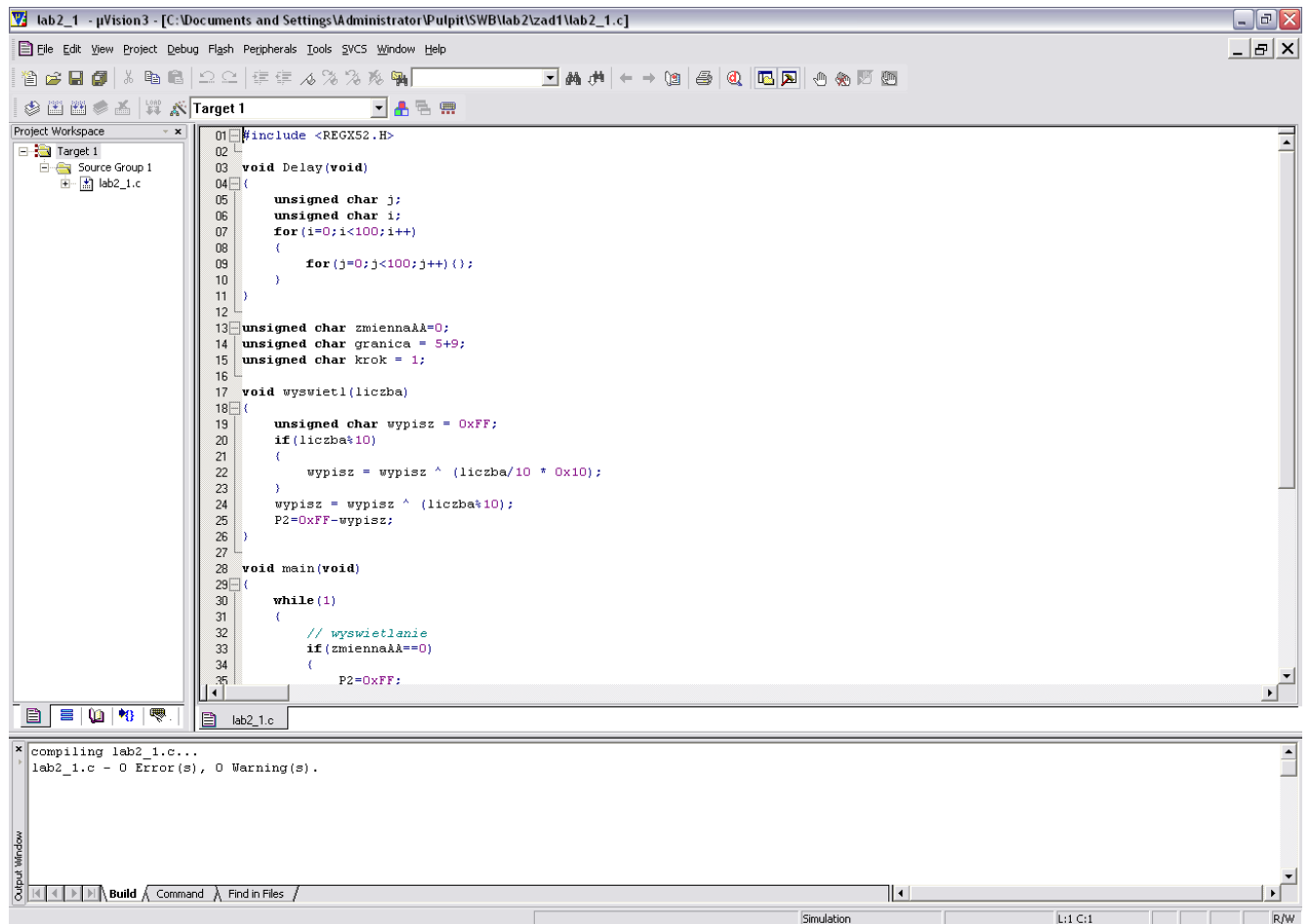
void Delay(void)
{
    unsigned char j;
    unsigned char i;
    for(i=0;i<100;i++)
    {
        for(j=0;j<100;j++){};
    }
}

unsigned char zmiennaAA=0;
unsigned char granica = 5+9;
unsigned char krok = 1;

void wyswietl(liczba)
{
    unsigned char wypisz = 0xFF;
    if(liczba%10)
    {
        wypisz = wypisz ^ (liczba/10 * 0x10);
    }
    wypisz = wypisz ^ (liczba%10);
    P2=0xFF-wypisz;
}

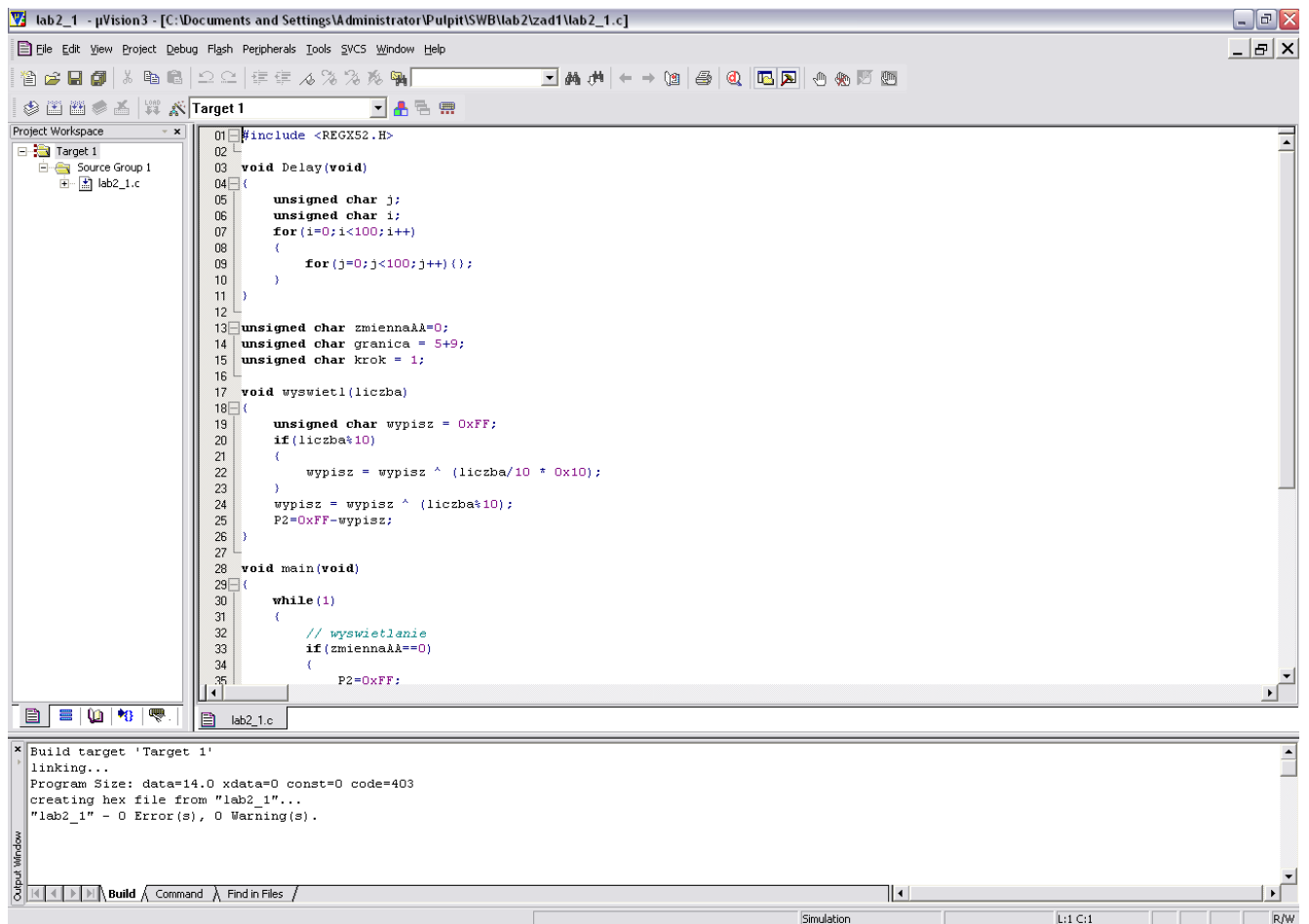
void main(void)
{
    while(1)
    {
        // wyswietlanie
        if(zmiennaAA==0)
        {
            P2=0xFF;
            Delay();
        }
        else if(zmiennaAA%2)
        {
            wyswietl(zmiennaAA);
        }
        //iteracja
        if(zmiennaAA>=granica)
        {
            zmiennaAA=0;
        }
        else
        {
            zmiennaAA+=krok;
        }
        Delay();
    }
}
```


Sprawdzenie poprawności Kompilowanie



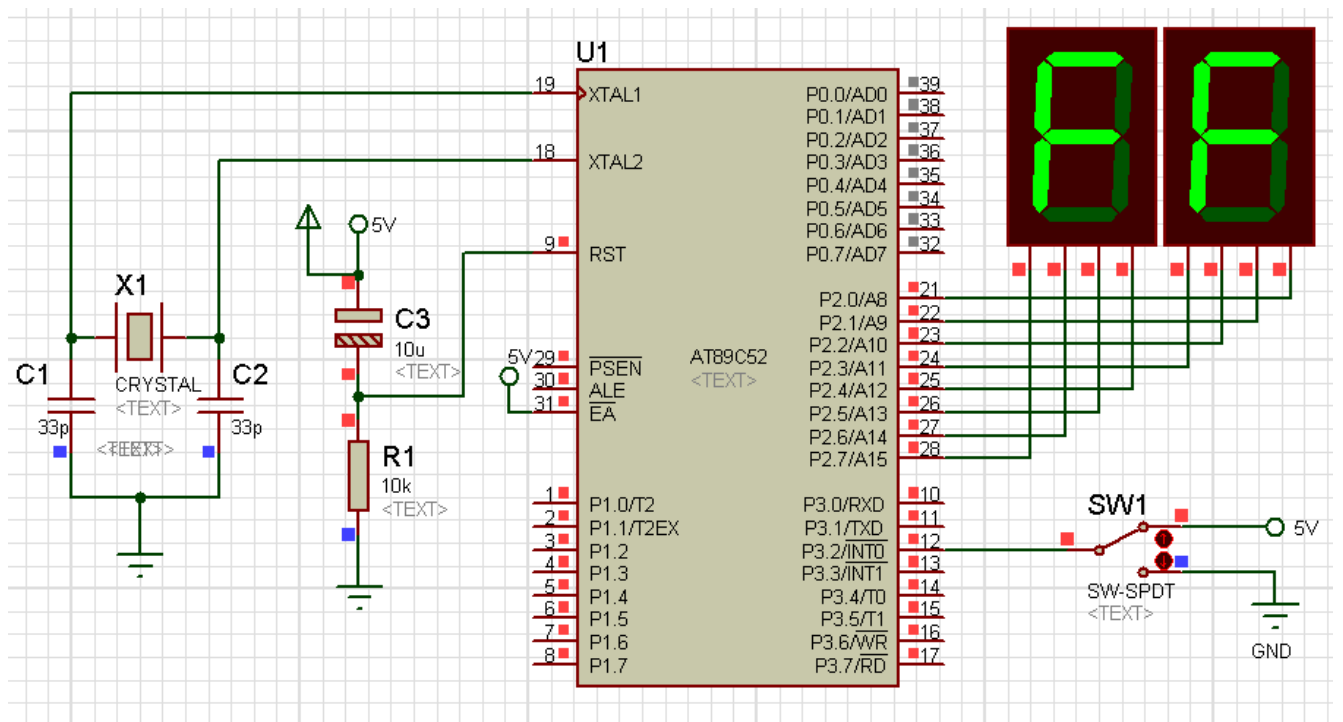
```
lab2_1 - pVision3 - [C:\Documents and Settings\Administrator\Pulpit\SWB\lab2\zad1\lab2_1.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
lab2_1.c
01 #include <REGX52.H>
02
03 void Delay(void)
04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0; i<100; i++)
08     {
09         for(j=0; j<100; j++) {}
10     }
11 }
12
13 unsigned char zmiennaAA=0;
14 unsigned char granica = 5+9;
15 unsigned char krok = 1;
16
17 void wyswietl(liczba)
18 {
19     unsigned char wypisz = 0xFF;
20     if(liczba%10)
21     {
22         wypisz = wypisz ^ (liczba/10 * 0x10);
23     }
24     wypisz = wypisz ^ (liczba%10);
25     P2=0xFF-wypisz;
26 }
27
28 void main(void)
29 {
30     while(1)
31     {
32         // wyswietlanie
33         if(zmiennaAA==0)
34         {
35             P2=0xFF;
36         }
37     }
38 }
lab2_1.c
Output Window
x
compiling lab2_1.c...
lab2_1.c - 0 Error(s), 0 Warning(s).
Build Command Find in Files
Simulation L:1 C:1 R/W
```

Linkowanie

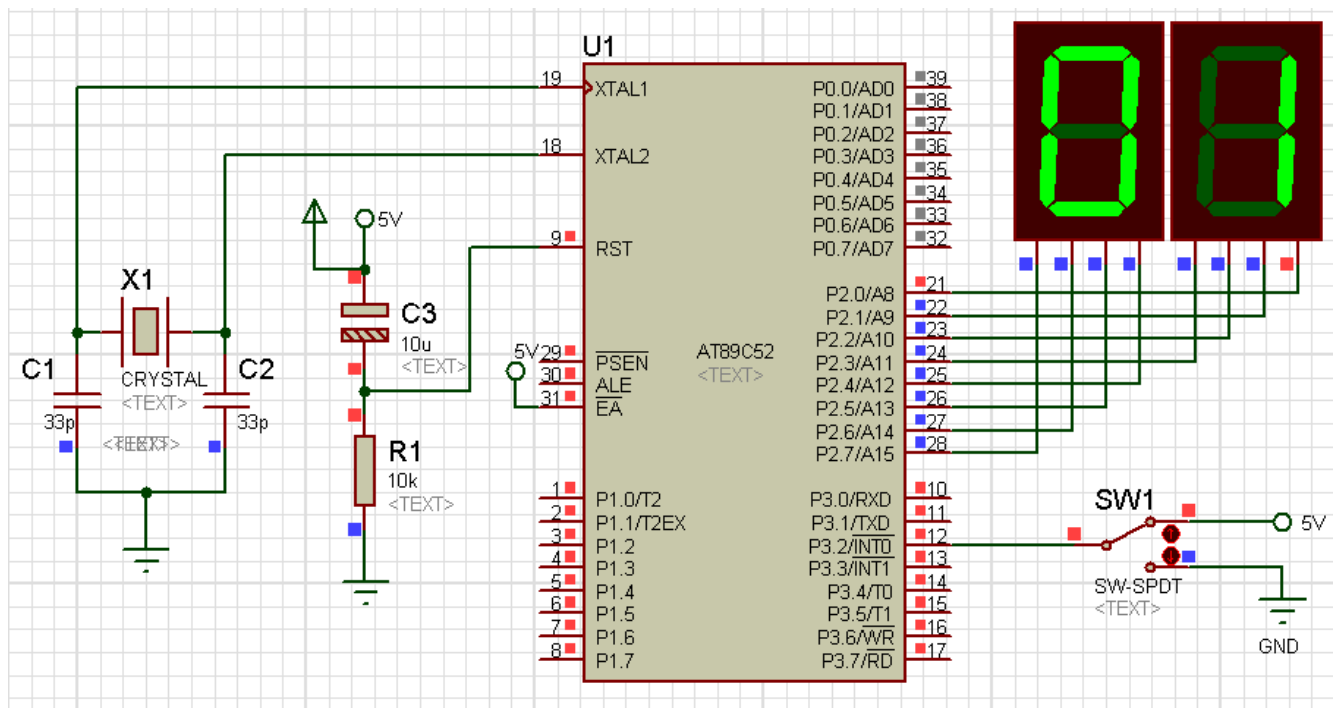


Prezentacja realizacji zadania przez program

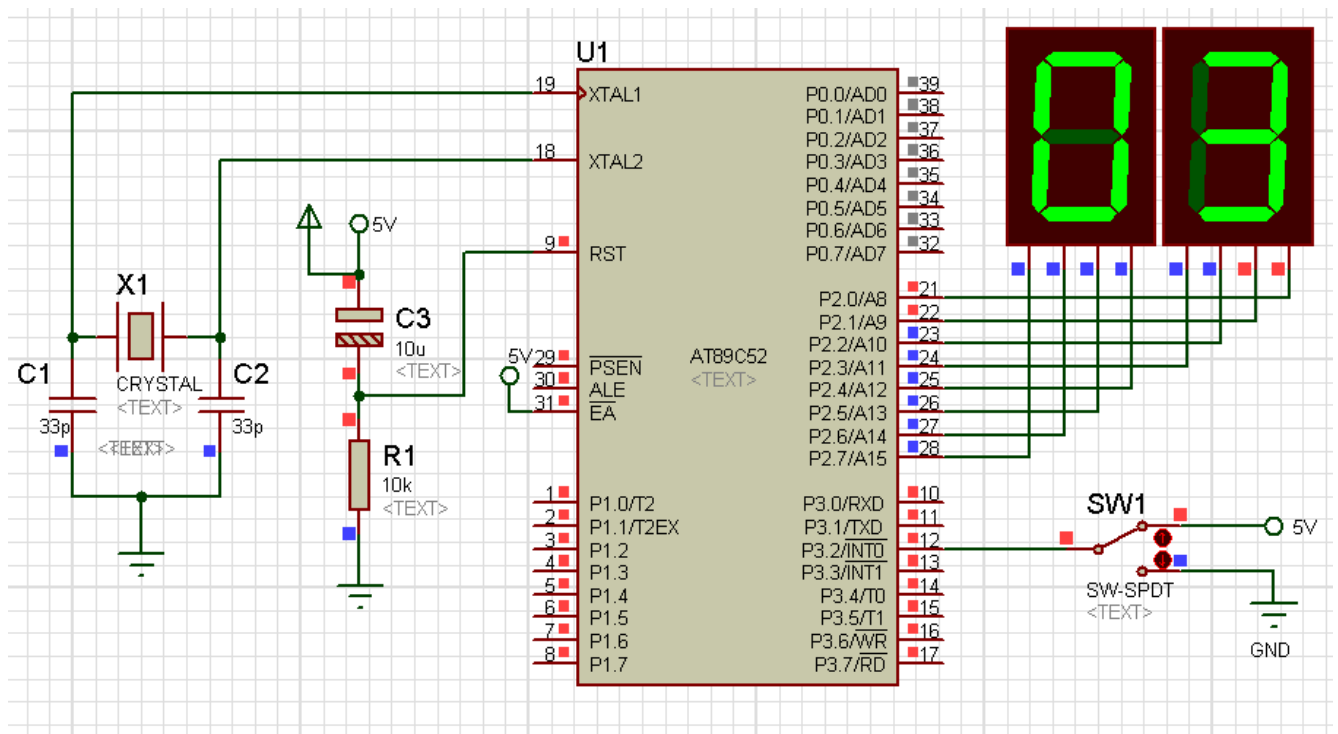
Stan początkowy



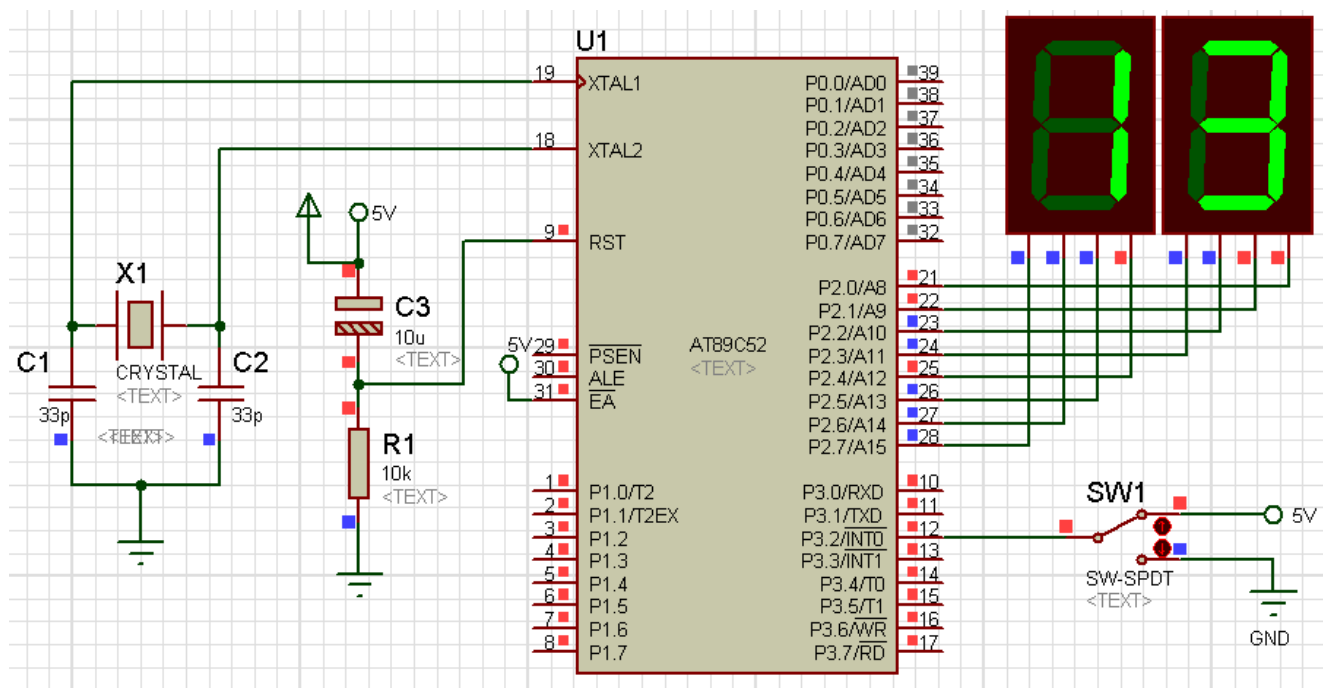
Pierwsza zmiana na wyświetlaczach



Druaga zmiana na wyświetlaczu



Ostatnia zmiana na wyświetlaczu przez wyświetleniem FF

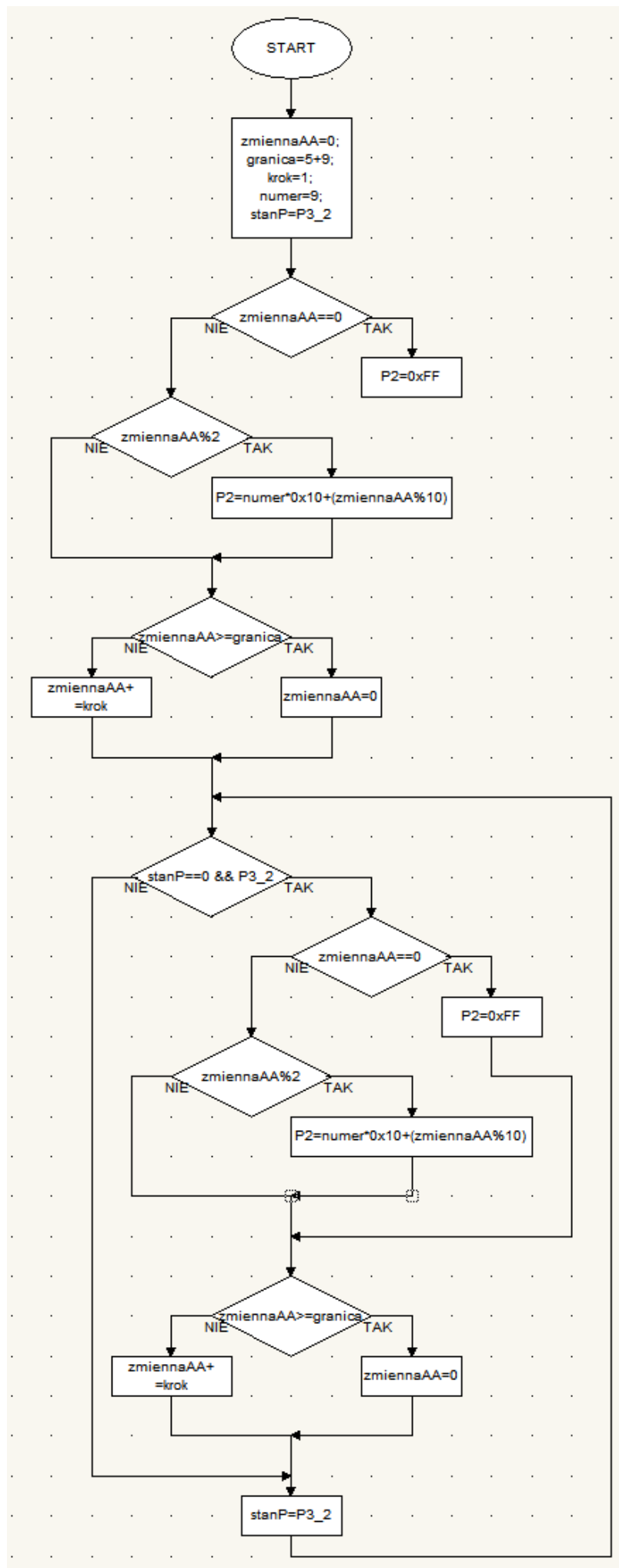


A. Zadanie na ocenę dobrą

Opis mojego rozwiązania

W ramach bieżącego problemu stworzyłem warunek sprawdzający, czy nastąpiło przełączenie SW1 ze stanu 0 na stan 1. Przy spełnieniu warunku następuje cykliczna iteracja zmiennej zmiennaAA od 0 do 14 co 1 (po wartości 14 zmiennaAA wynosi 0). Program wypisuje na pierwszym wyświetlaczu numer (9), a na drugim wypisuje wartość jedności liczby, jeśli jest ona nieparzysta, w przeciwnym wypadku zostawia znak poprzedni znak. Jeśli cykl się zamknie, na wyświetlaczu widnieje „FF”.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

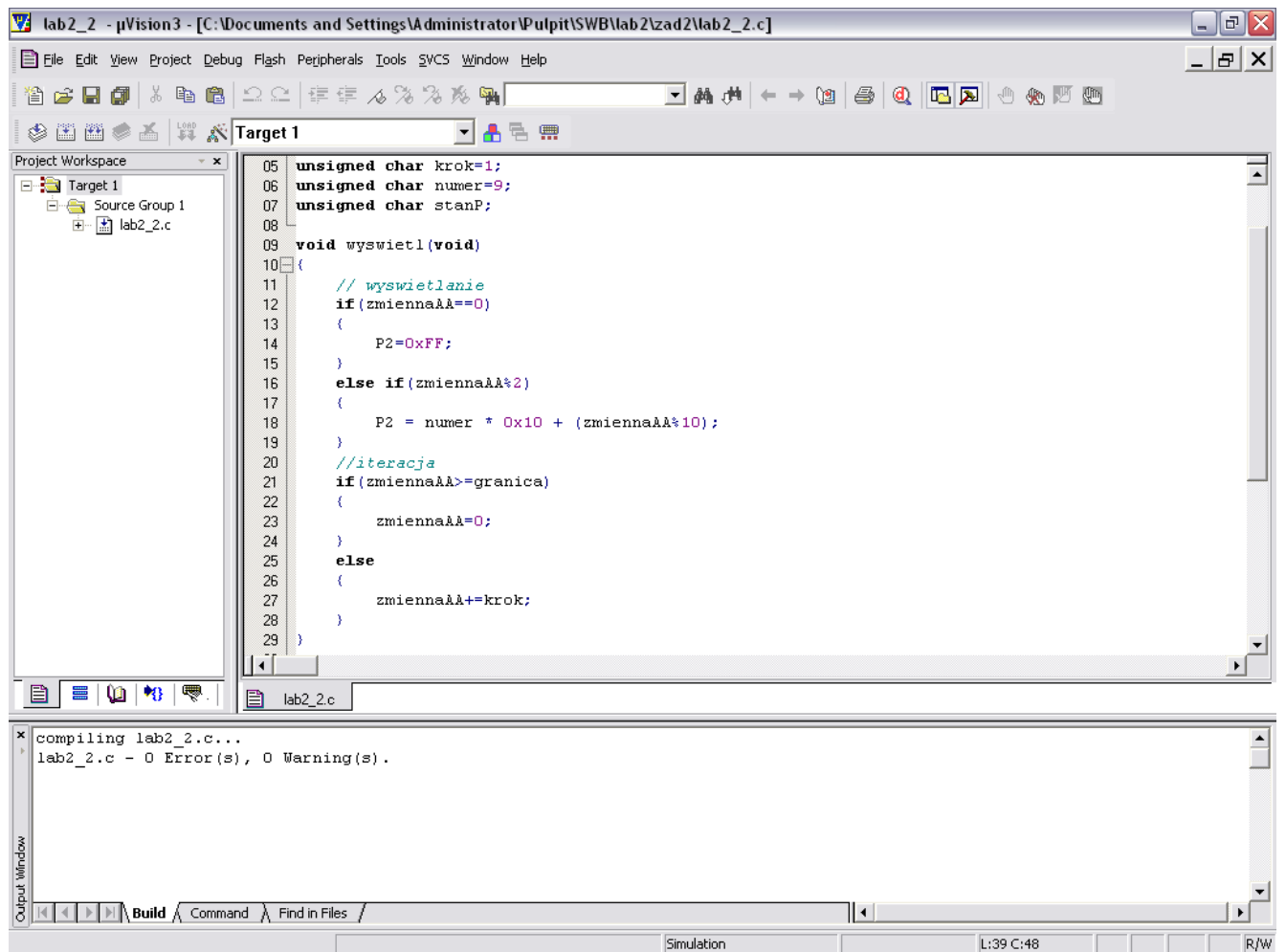
unsigned char zmiennaAA=0;
unsigned char granica=5+9;
unsigned char krok=1;
unsigned char numer=9;
unsigned char stanP;

void wyswietl(void)
{
    // wyswietlanie
    if(zmiennaAA==0)
    {
        P2=0xFF;
    }
    else if(zmiennaAA%2)
    {
        P2 = numer * 0x10 + (zmiennaAA%10);
    }
    //iteracja
    if(zmiennaAA>=granica)
    {
        zmiennaAA=0;
    }
    else
    {
        zmiennaAA+=krok;
    }
}

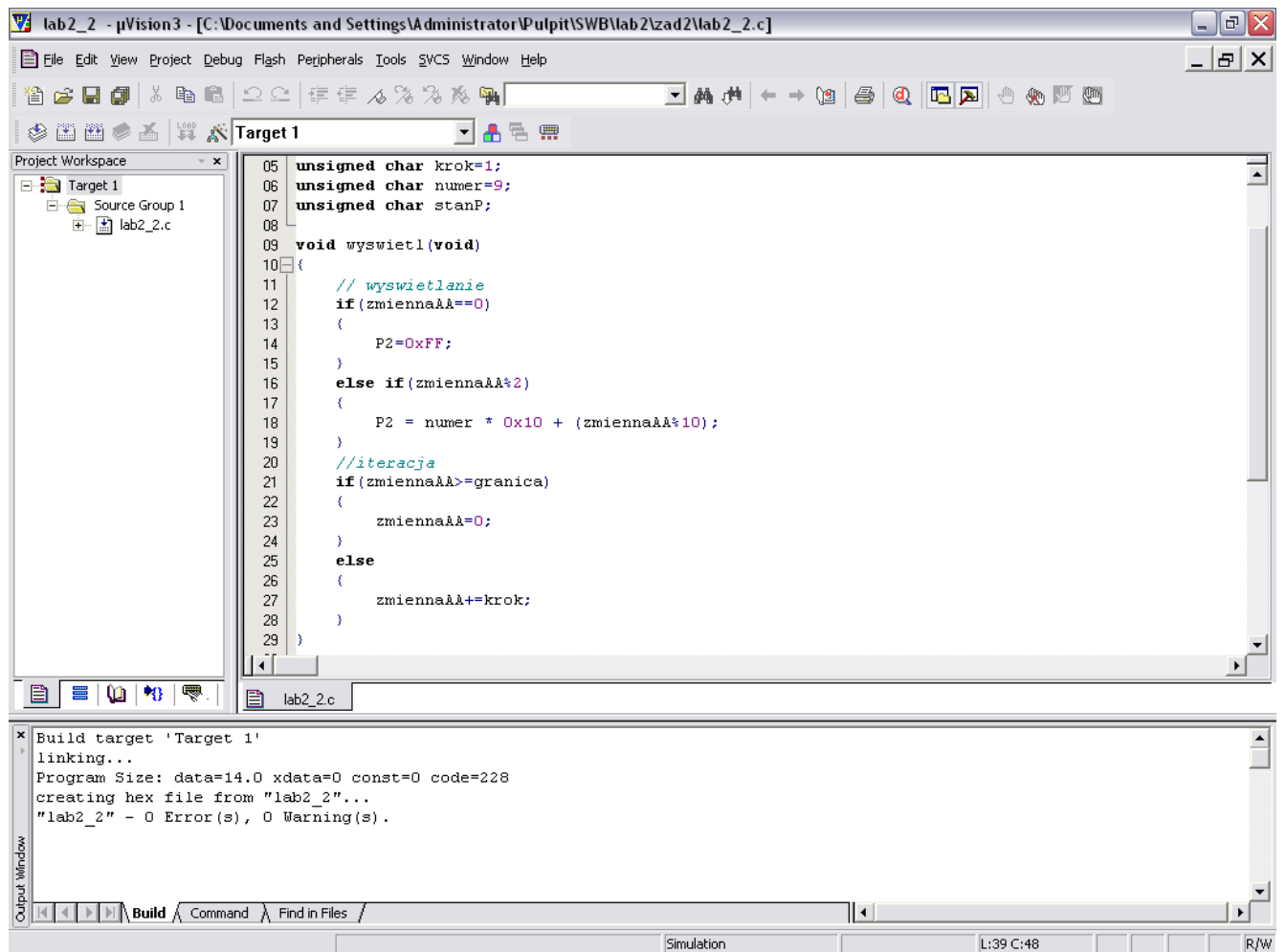
void main (void)
{
    stanP=P3_2;
    wyswietl();
    while(1)
    {
        if(stanP==0 && P3_2)
        {
            wyswietl();
        }
        stanP=P3_2;
    }
}
```

Sprawdzenie poprawności

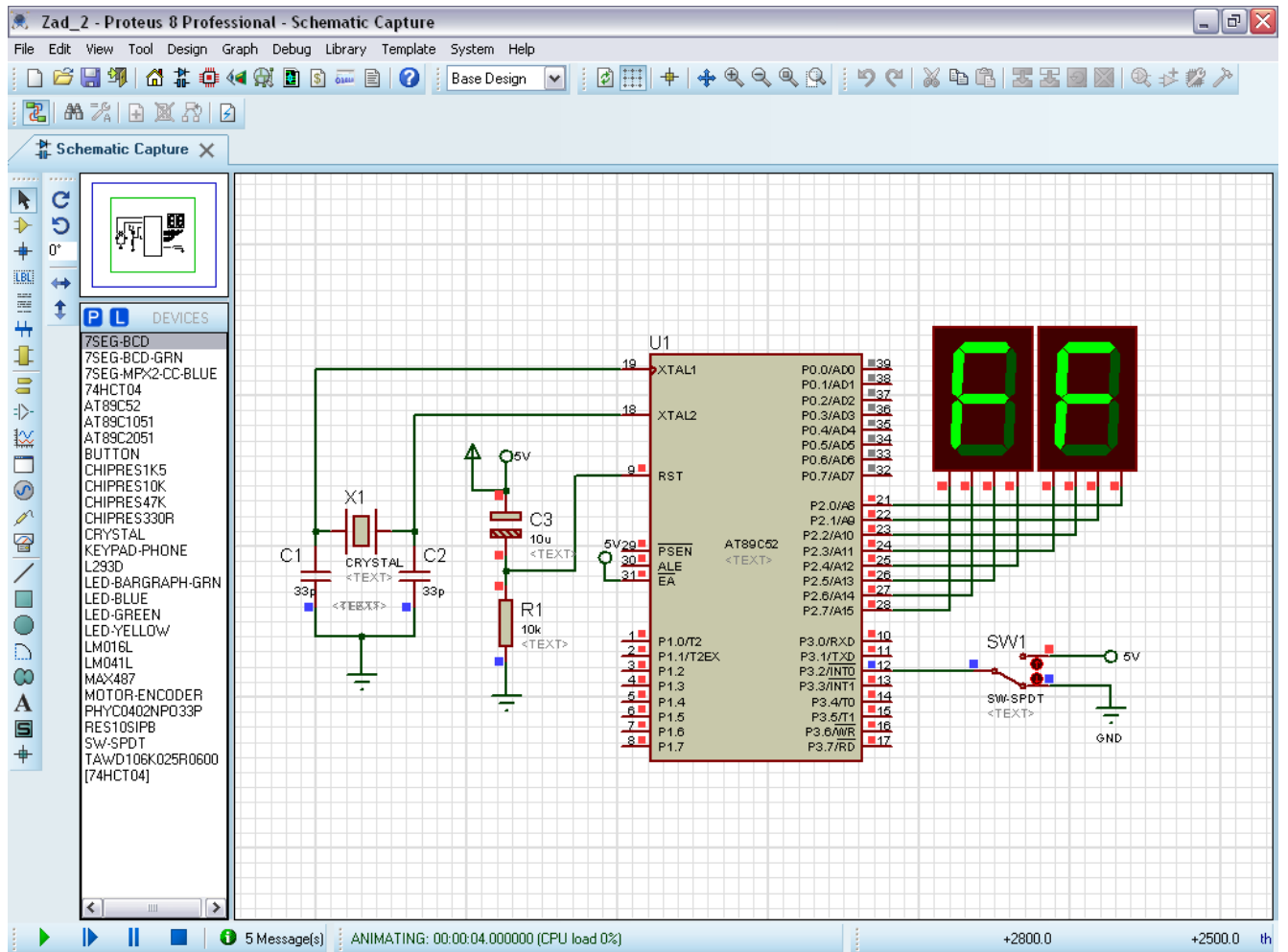
Kompilowanie



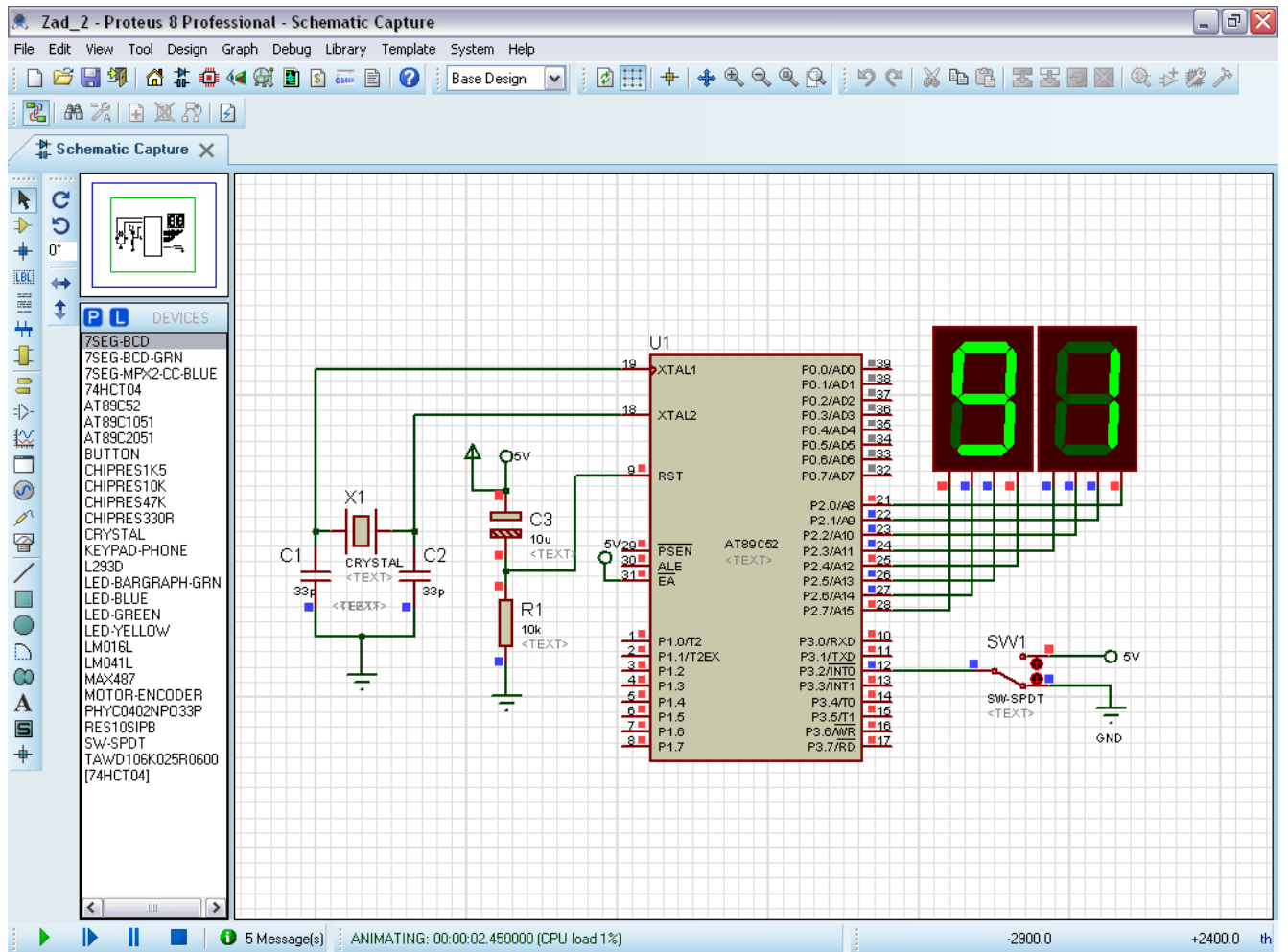
Linkowanie



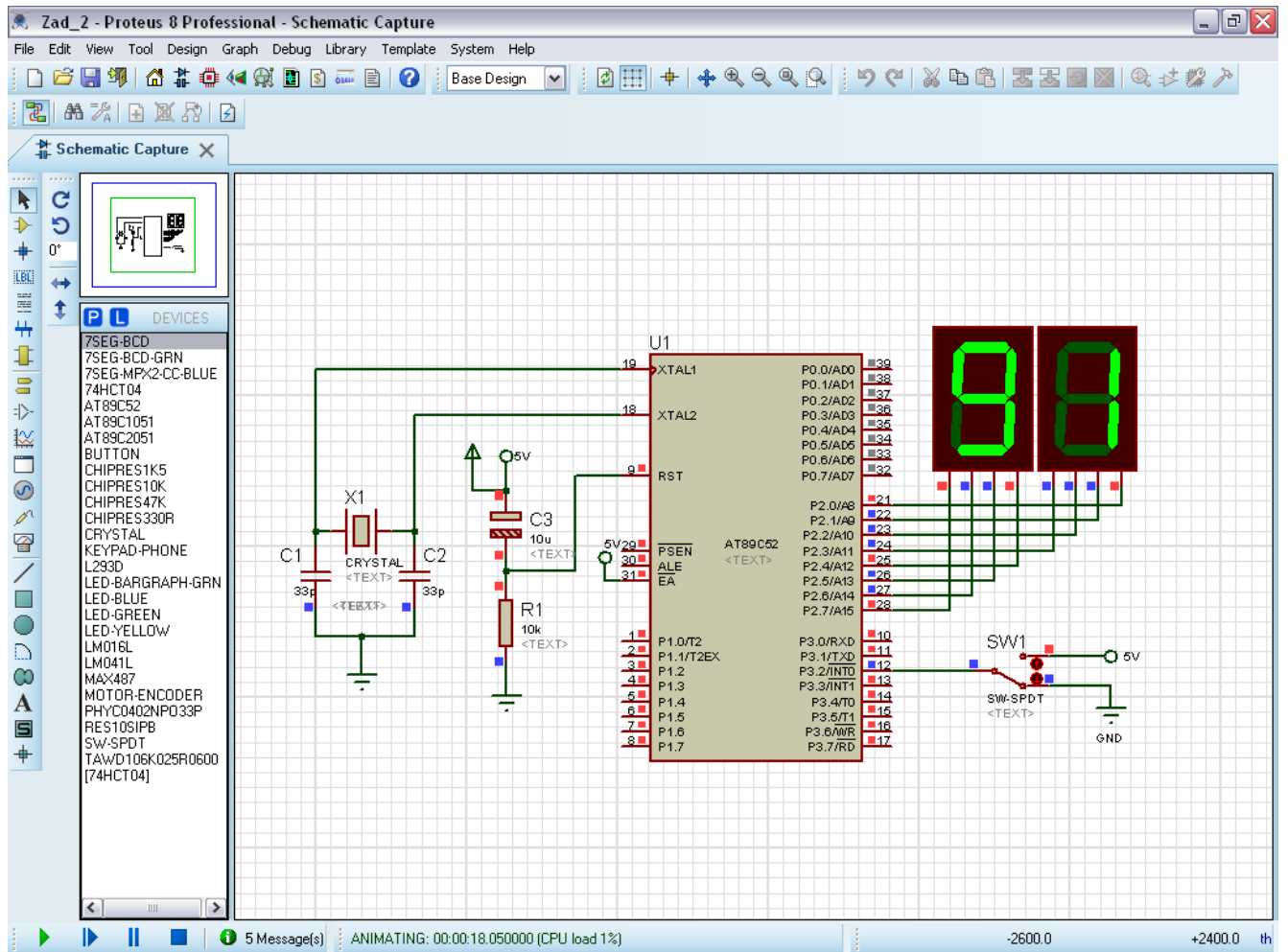
Prezentacja realizacji zadania przez program
Stan początkowy



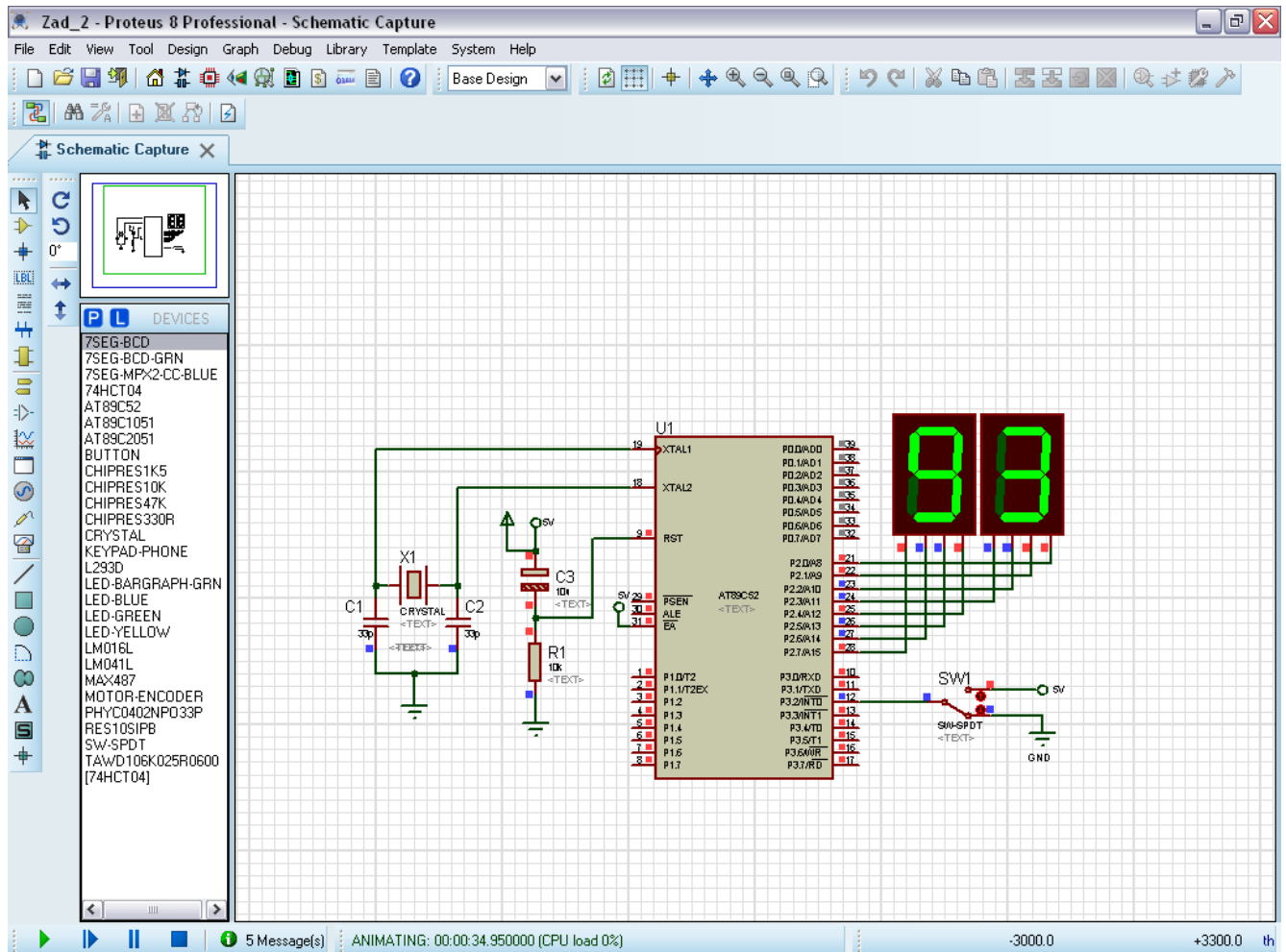
Pierwsze przejście 0->1 (zmiana wartości)



Drugie przejście 0->1 (brak zmiany wartości)



Trzecie przejście 0->1 (zmiana wartości)

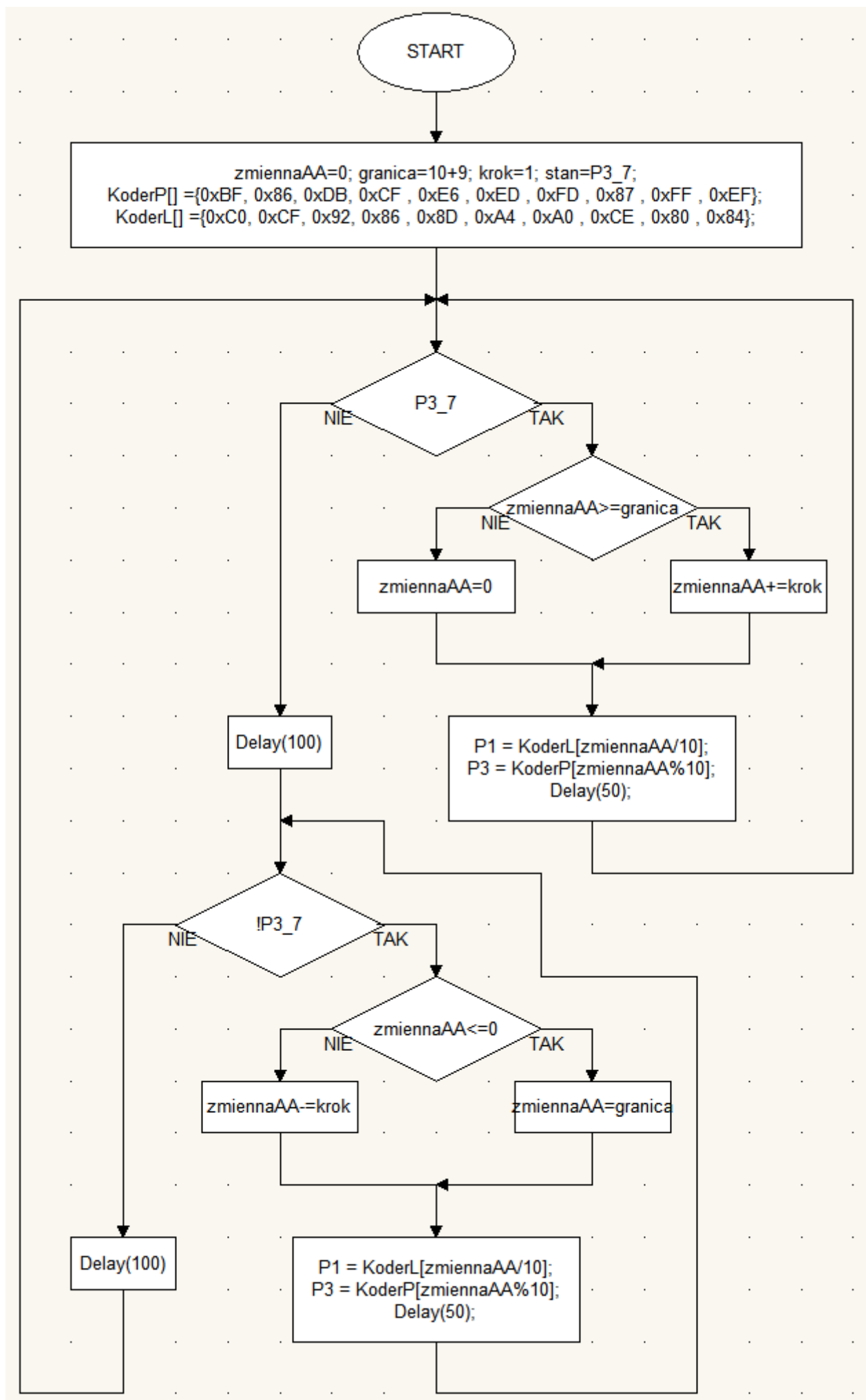


A. Zadanie na ocenę bardzo dobrą

Opis mojego rozwiązania

W bieżącym problemie stworzyłem 2 tablice: KoderP i KorerL, które przechodzą obliczone wcześniej wartości do wyświetlenia odpowiedniej cyfry kolejno na wyświetlaczu prawym i lewym. W ramach programu wykonuję pętlę, która składa się z 4 elementów: inkrementacji wartości, dekrementacji wartości i opóźnień między nimi dla lepszego zobrazowania zmian między nimi. Zarówno inkrementacja, jak i dekrementacja zmienia wartość zmiennej zmiennaAA cyklicznie, jak wcześniej, w przedziale od 0 do 19 co 1. Cykl zamyka się przy zmianie 0 na 19 lub 19 na 0. Program wypisuje liczbę dziesiątek na lewym wyświetlaczu, a liczbę jedności na prawym wyświetlaczu.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

void Delay(unsigned char x)
{
    unsigned char j;
    unsigned char i;
    for(i=0;i<100;i++)
    {
        for(j=0;j<x;j++){};
    }
}

unsigned char zmiennaAA=0;
unsigned char granica=10+9;
unsigned char krok=1;
unsigned char stan;

unsigned char code KoderP[] =
    {0xBF, 0x86, 0xDB, 0xCF , 0xE6 , 0xED , 0xFD , 0x87 , 0xFF , 0xEF};
unsigned char code KoderL[] =
    {0xC0, 0xCF, 0x92, 0x86 , 0x8D , 0xA4 , 0xA0 , 0xCE , 0x80 , 0x84};

void operacja(unsigned char q)
{
    if(q)
    {
        if(zmiennaAA>=granica)
        {
            zmiennaAA=0;
        }
        else
        {
            zmiennaAA+=krok;
        }
    }
    else
    {
        if(zmiennaAA<=0)
        {
            zmiennaAA=granica;
        }
        else
        {
            zmiennaAA-=krok;
        }
    }
    P1 = KoderL[zmiennaAA/10];
    P3 = KoderP[zmiennaAA%10];
    Delay(50);
}

void main(void)
```



```

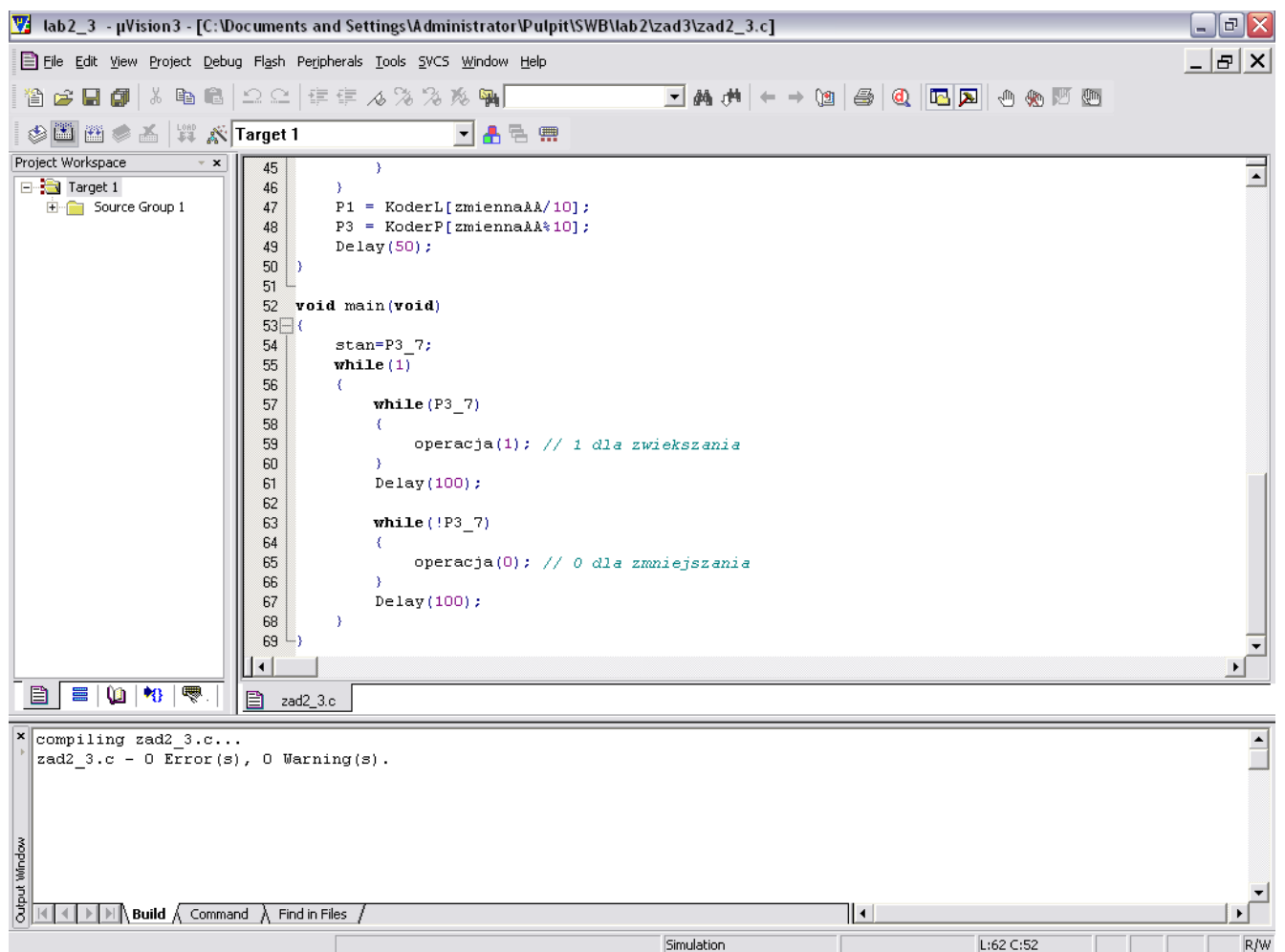
{
    stan=P3_7;
    while(1)
    {
        while(P3_7)
        {
            operacja(1); // 1 dla zwiększania
        }
        Delay(100);

        while(!P3_7)
        {
            operacja(0); // 0 dla zmniejszania
        }
        Delay(100);
    }
}

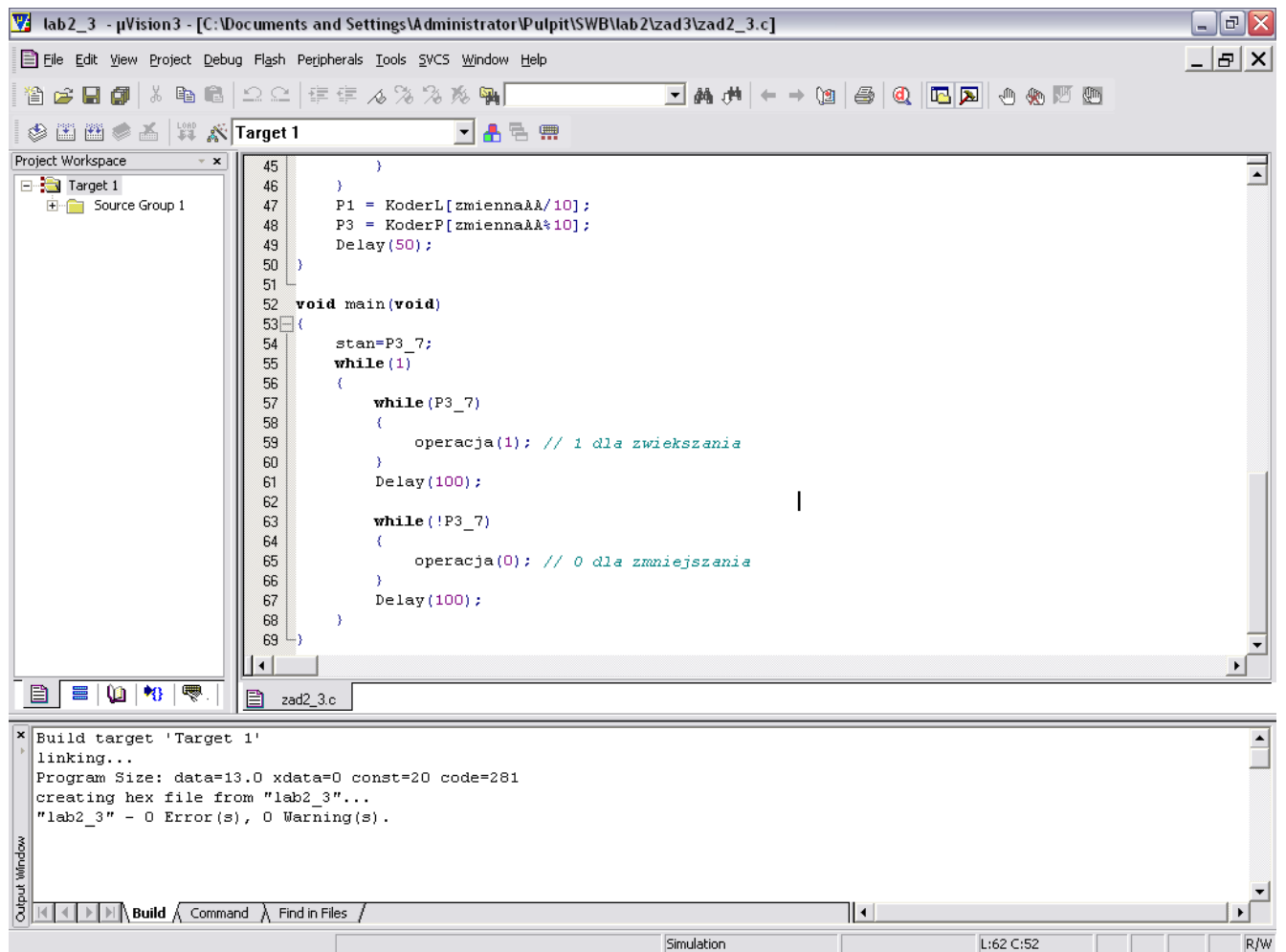
```

Sprawdzenie poprawności

Kompilowanie

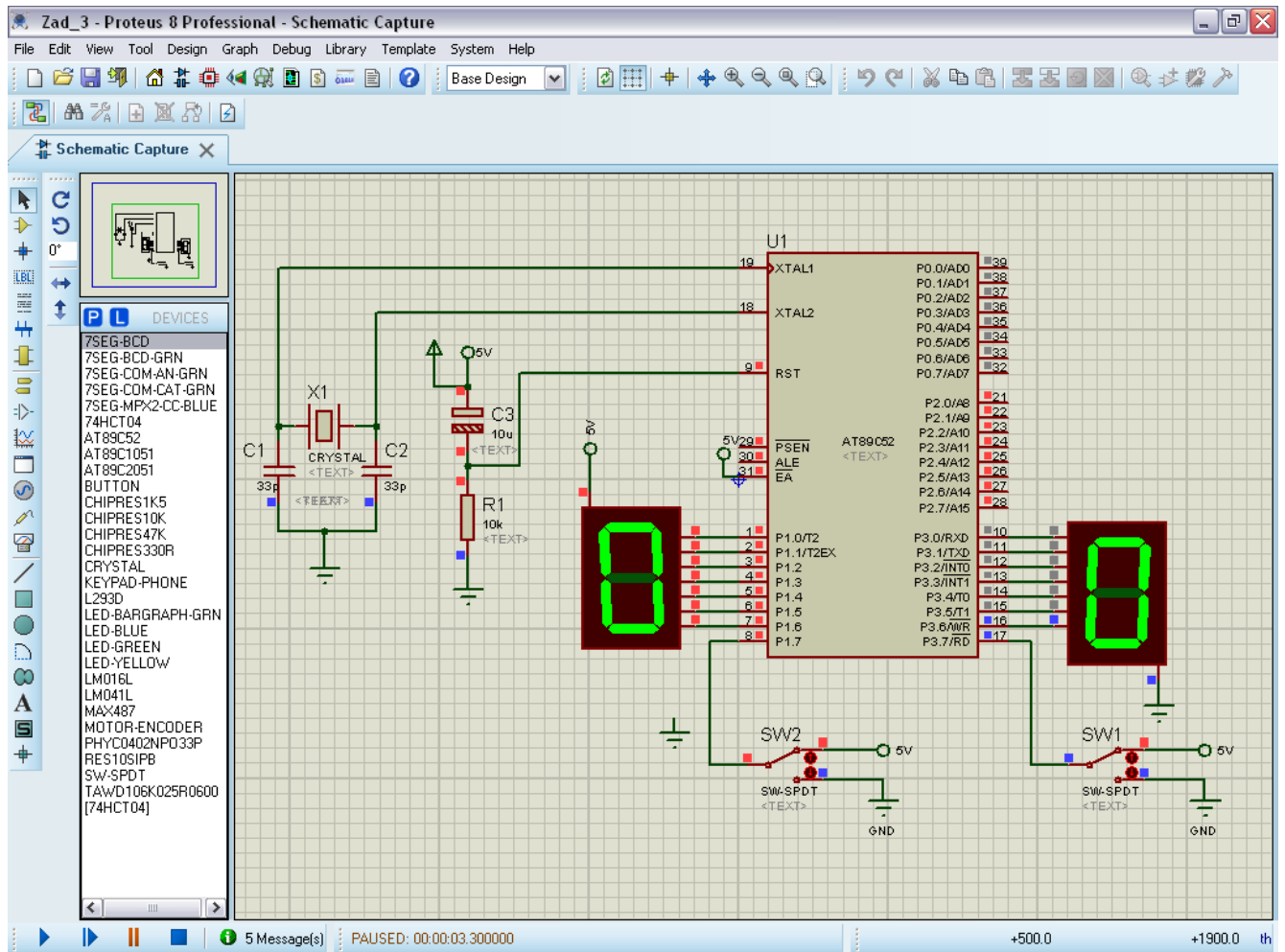


Linkowanie

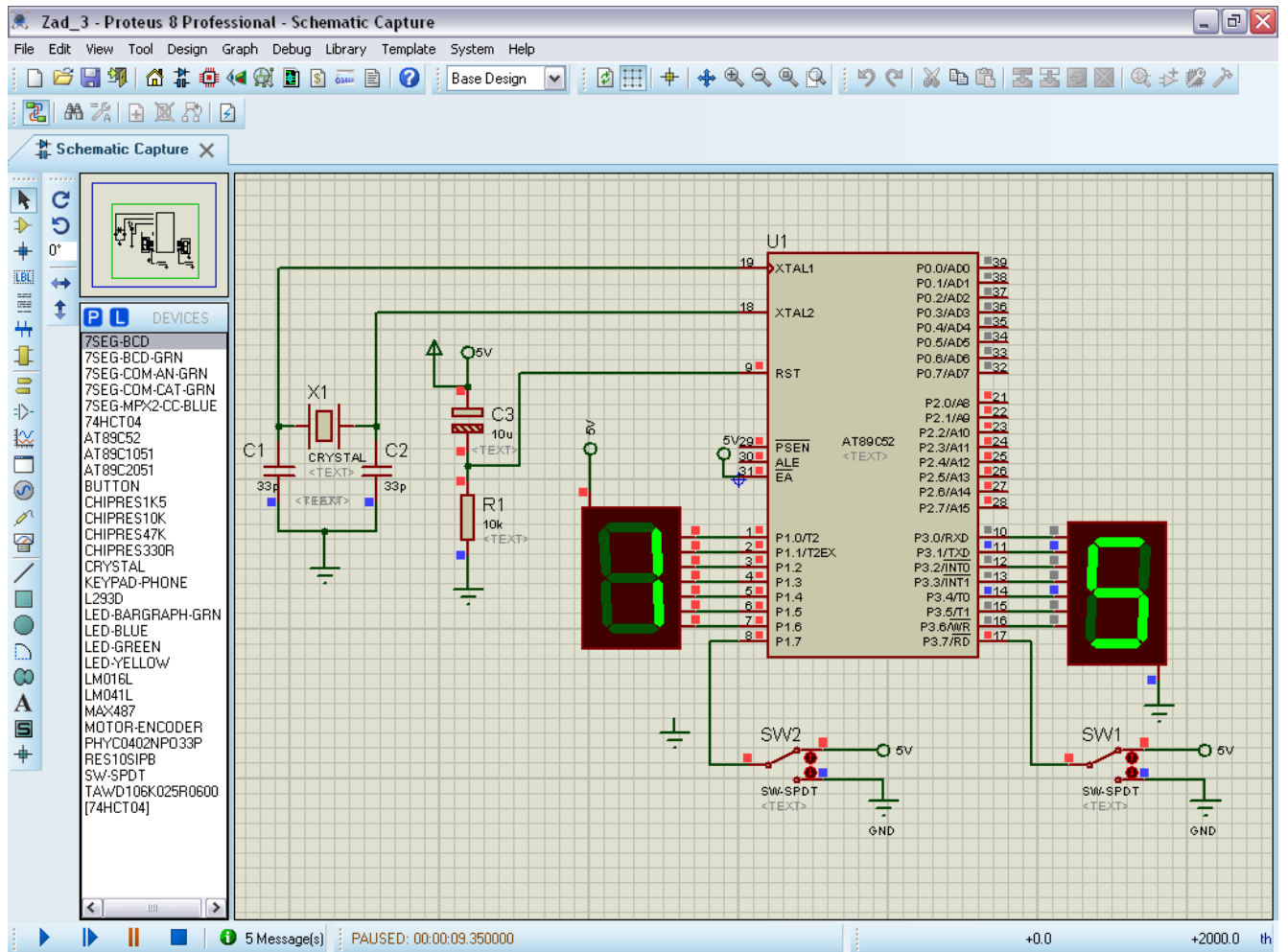


Prezentacja realizacji zadania przez program

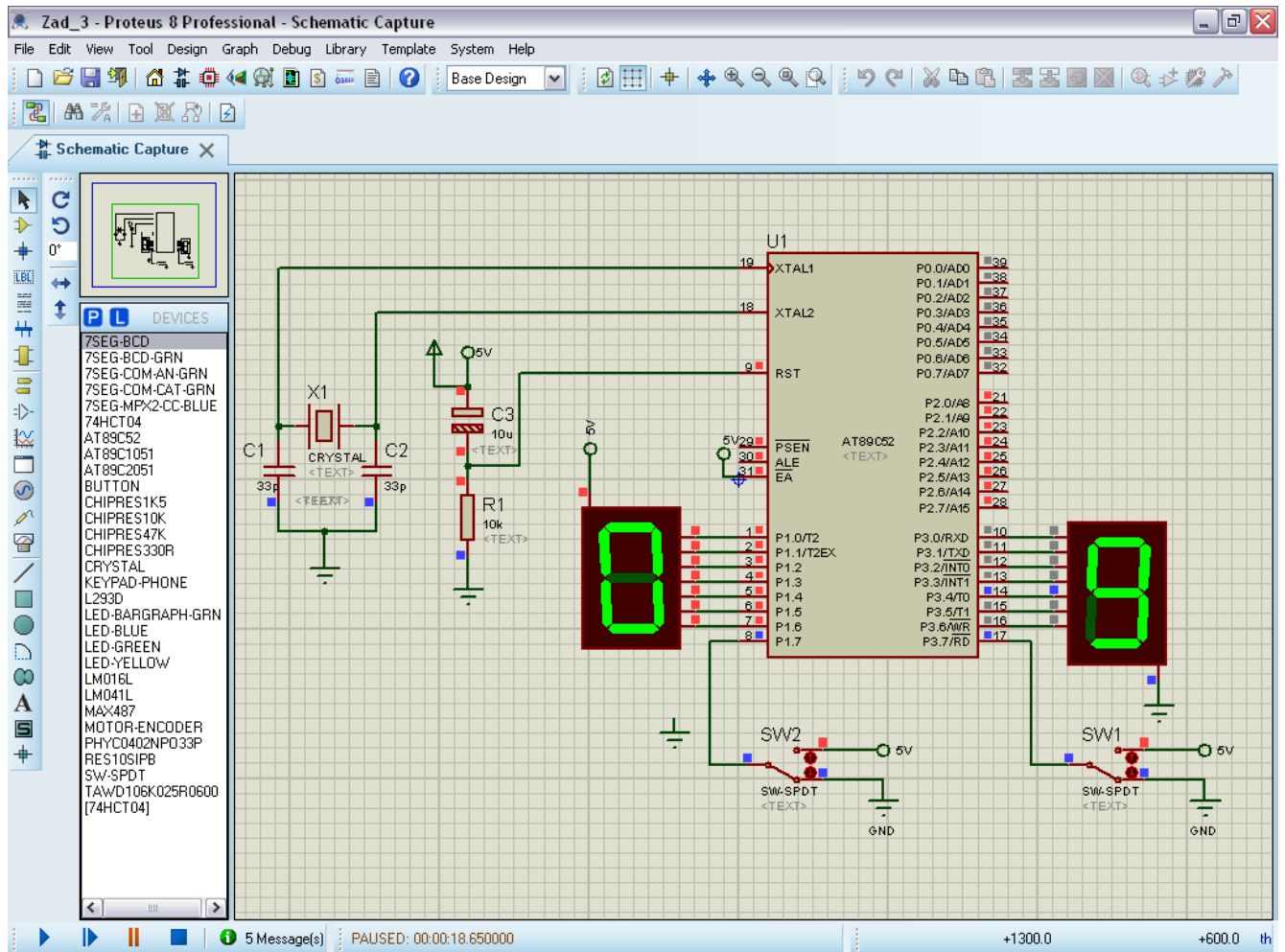
Inicjalizacja – wartości równe 0



Chwila oczekania – wzrost do wartości 15



Zmiana stanu – wartości mającą, tutaj – na 9



Bez przełączania – wartość maleje, aż zmienia wartość z 0 na 19

