Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:

Systemy wbudowane

Sprawozdanie z ćwiczenia laboratoryjnego nr 4:

Obsługa systemu wejścia - wyjścia

Prowadzący:
mgr inż. Artur Miktus

Wykonał: Radosław Relidzyński
Nr albumu: 76836
Grupa: WCY20IY4S1
Data laboratoriów: 26.05.2022 r.
Deklarowana ocena: 3, 4, 5

Spis treści

A. Treść zadania

Zadanie na Laboratorium nr 4

Zadanie na dostatecznie.

Na podstawie przykładowego programu ze strony

http://www.ita.wat.edu.pl/~a.miktus/SWB/Timer/Timer.html

napisać program **zad1.c** w języku C na układ z projektu

http://www.ita.wat.edu.pl/~a.miktus/SWB/Timer/Przerwania1.pdsprj

tak, aby przy wykorzystaniu procedury przerwania od Timera 0, pracującego w trybie 1 uzyskać powtarzające się **świecenie zielonej diody LED** (uwaga - nie cały okres tylko **stan niski na P2.0**) przez **:**

- 15 ms * numer w dzienniku na studentów o numerze nieparzystym;
- 10 ms * numer w dzienniku na studentów o numerze parzystym.

W sprawozdaniu poza kompletem plików (konieczny projekt w Keil uVision! i projekt w Proteus) przedstawić:

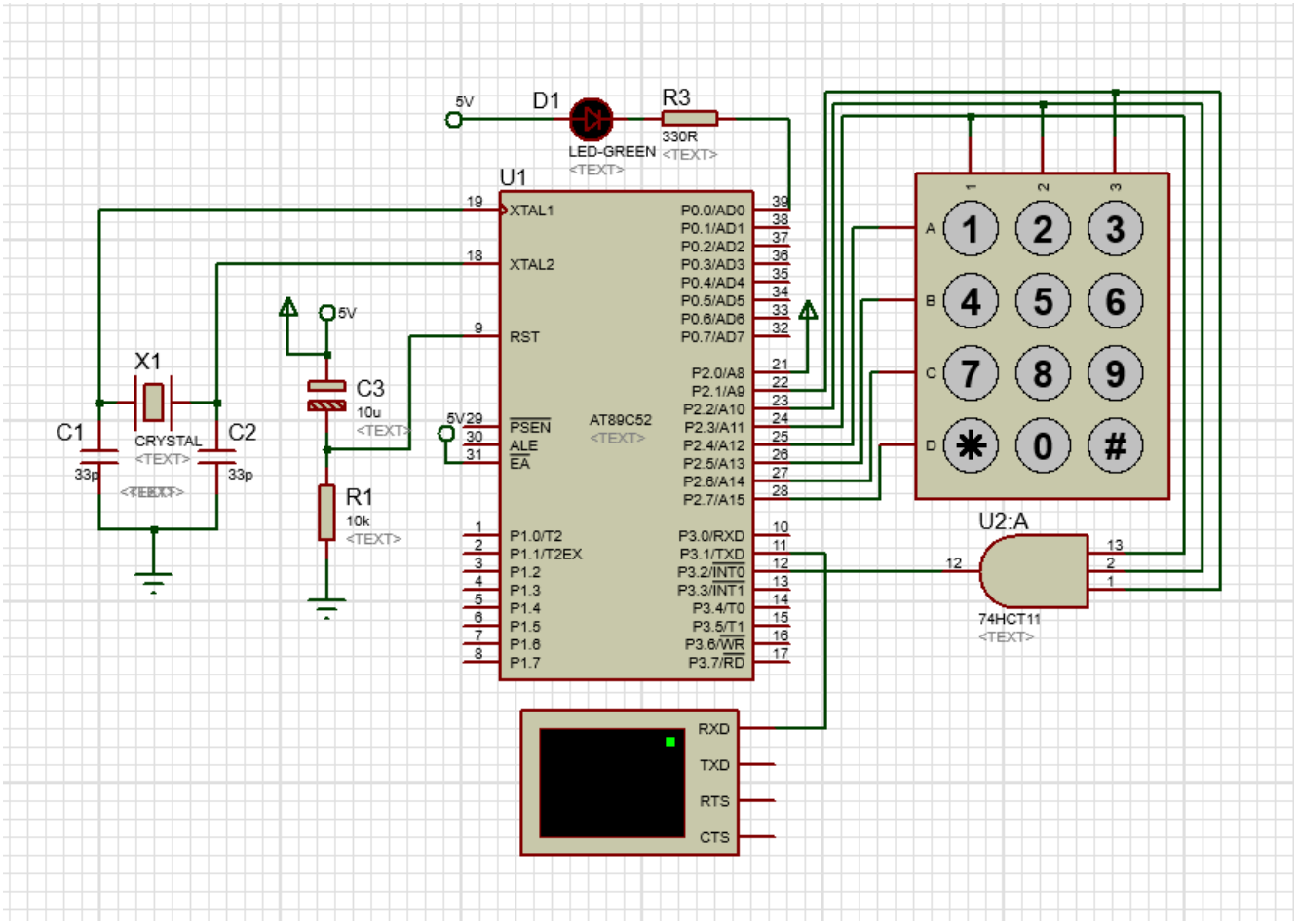
a) **wszystkie konieczne obliczenia**, prowadzące do wartości w odpowiednich rejestrach - **bez obliczeń i uzasadnienia ocena za zadanie to niedostatecznie**;
b) zrzuty ekranu, prezentujące potwierdzenie poprawnego działania programu (pomiary czasu świecenia diody) przy wykorzystaniu **obu narzędzi**

- Debuggera środowiska Keil uVision
- Oscyloskopu w środowisku Proteus (przycisk Cursors).

Zadanie na dobrze.

To co na dostatecznie i ponadto:

Dla układu z projektu przedstawionego poniżej:



w oparciu o szablon programu, prezentowanego na wykładach (**bez konieczności wykorzystania wszystkich zmiennych lub procedur**):

napisać program **zad2.c** w języku C, realizujący obsługę **za pomocą przerwań** zamieszczonej w układzie klawiatury tak, aby po naciśnięciu dowolnego przycisku mikrokontroler przez obsługę przerwania INT0 z pinu P3.2 wywołał odpowiadającą mu sekwencję błysków diody LED tak, że

a) dla studentów o numerach nieparzystych będzie to liczba błysków równa numerowi wiersza (liczonemu od góry 1,2,3,4), przerwa o czasie równym trzykrotności czasu świecenia LED przy błysku, jeden błysk.

b) dla studentów o numerach parzystych będzie to liczba błysków równa numerowi kolumny (liczonemu od lewej 1,2,3), przerwa o czasie równym dwukrotności czasu świecenia LED przy błysku, dwa błyski.

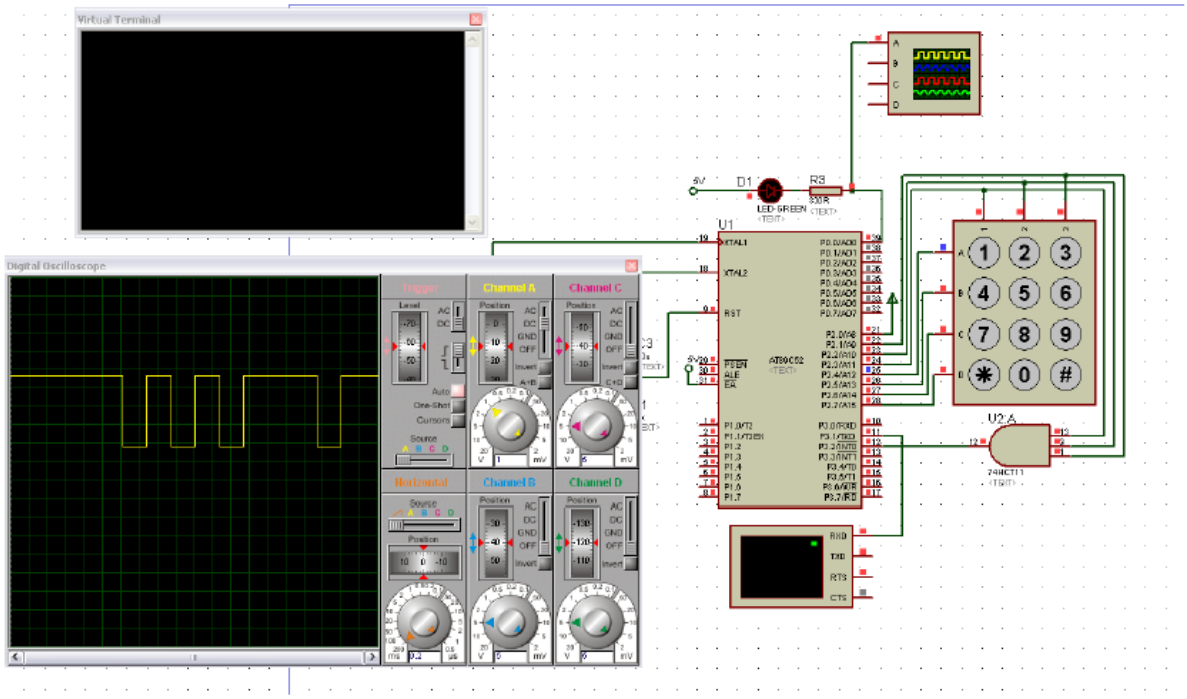
Czas trwania błysku LED to 500 ms dla numerów w dzienniku nieparzystych i 750 ms dla studentów o numerach w dziennku parzystych.

Uwaga: kod w zadaniu na db **NIE MA** wysyłać informacji na Terminal Wirtualny, jak w zadaniu na bdb - elementem tego zadania jest umiejętność usunięcia (zakomentowania) tych fragmentów przykładowego kodu, które odpowiadają za realizację tych działań.

Dynamikę "migania" diodą LED np. przy obsłudze klawiatury można oddać na oscyloskopie, dodanym samodzielnie do projektu. Na poniższym rysunku widać ile razy i w jakich odstępach czasowych na pinie P0.0 wystąpił stan niski, proszę wobec tego nie pisać, "że się nie dało udowodnić działania na zrzucie ekranu z Proteusa":

Ocena db

Do schematu dołączony został w celu prezentacji oscyloskop („Zad_7_osc.pdsprj"). Po wciśnięciu klawisza 7:



Zadanie na bardzo dobrze.

To co na dobrze i ponadto: napisać program **zad3.c**, który ma być programem na dobrze, rozbudowanym o **obsługę za pomocą przerwań transmisji przez port szeregowy na Terminal Wirtualny** kodów ASCII znaków naciskanych pojedynczo na klawiaturze (cyfry 0…9, *,#) . Terminal ma wyświetlać cyfry i symbole (np. po naciśnięciu 1*9 na terminalu ma się pojawić 1*9, diody LED też oczywiście sygnalizują według zadania na dobrze), Terminal nie ma wyświetlać wartości dziesiętnych ani szesnastkowych kodów ASCII. Czasy trwania błysków takie, jak w zadaniu na dobrze.

Dodatkowo, **tylko w zadaniu na bdb naciśnięcie każdego przycisku ma być potwierdzone krótkim (250ms) pojedynczym błyskiem LED - przed serią błysków rozpoznających wiersz albo kolumnę i przed wysłaniem informacji na Wirtualny Terminal .**

Sekwencja działań przy naciskaniu klawisza:
a) błysk potwierdzający 250ms;

b) błyski rozpoznające wiersz albo kolumnę, przerwa, jeden(nieparzyści) albo dwa (parzyści) błyski (jak w zadaniu na dobrze);
c) wyświetlenie wprowadzonego znaku na Wirtualnym Terminalu.

Uwaga - kody ASCII można znaleźć np. na stronie http://www.asciitable.com/).

B. Zadanie na ocenę dostateczną

Opis mojego rozwiązania

W ramach zadania zaprogramowałem mikrokontroler tak, aby stale błyskał on zieloną diodą (błyśnięcie trwa 135ms) oraz żeby zaświecał diodę żółtą wraz z wciśnięciem przycisku. Wykorzystam zmienną SW, która posłuży jako wejście, pobierać będzie stan przycisku, oraz zmienną TMOD = 0x01 określającą tryb timera (tutaj – 16-bitowy). Dodatkowo flagi ET0 oraz są ustawione na 1, aby umożliwić zaistnienie przerwań oraz ich obsługę w układzie. Dodatkowo, flaga TR0=1, włączająca timer0.

Obsługa zielonej diody będzie działała tak, że w nieskończonej pętli w układzie będą zachodzić przerwania, w ramach których będzie następować zmiana stanu diody (czas między zmianami jest zależny od wartości TH0 oraz TL0). W związku z tym, początkowo ładujemy wartości TH0 oraz TL0, jednakże za każdym razem, kiedy nastąpi przerwanie, będziemy musieli je od nowa załadować.

Wartości TH0 i TL0 obliczam w następujący sposób:

1. Wchodzę na stronę http://www.csgnetwork.com/timer8051calc.html i korzystam z dostępnego tam kalkulatora.
2. W polu „Microcontroller Clock Frequency” ustawiam wartość „1.3824” [Mhz], a w polu „Desired Timer Run-Time (DRT)” wpisuję mój czas „135” [ms].
3. Po wciśnięciu przycisku „Calculate” wyświetlają się poniższe wartości:

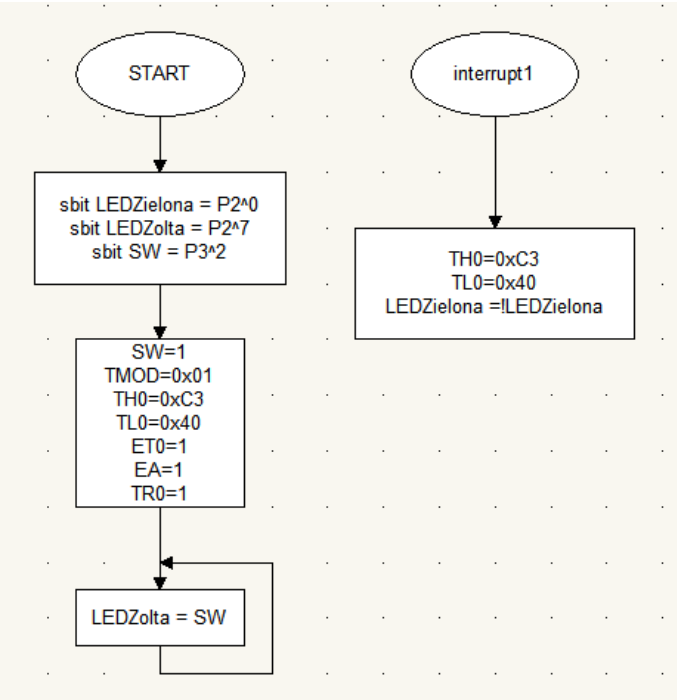
Designation	Value
Required Data Entry	
Microcontroller Clock Frequency	1.3824 Megahertz (mhz)
Desired Timer Run-Time (DRT)	135 Milliseconds (mSec)
Calculate Clear Values	
Calculated Results	

Timing Task	Standard 12-Clock Parts	Enhanced 6-Clock Parts	Time In
Single Timer TIC Duration	8,680555555555555 1 / (mhz/12)	4,340277777777778 1 / (mhz/6)	uSec
8-bit Timer Counter Maximum Run-Time	2,2222222222222223 TIC12 * 256 / 1000	1,1111111111111112 TIC6 * 256 / 1000	mSec
16-bit Timer Counter Maximum Run-Time	568,8888888888889 TIC12 * 65536 / 1000	284,44444444444446 TIC6 * 65536 / 1000	mSec
8-Bit DRT Reload Value	Beyond Values 256 - (DRT / TIC12 * 1000)	Beyond Values 256 - (DRT / TIC6 * 1000)	mSec
16-Bit DRT Reload Value	49984 65536 - (DRT / TIC12 * 1000)	34432 65536 - (DRT / TIC6 * 1000)	mSec

4. Odczytuję wartość „Standard 12-Clock Parts” dla zadania „16-Bit DRT Reload Value”. Jest to 49984
5. Konwertuję liczbę na postać hexaddecymalną: 49984 = 0xC340.
6. 2 najbardziej znaczące cyfry to wartość TH0, a 2 pozostałe to TL0

7. TH0 = 0xC3, TL0=0x40
8. Obliczone wartości wpisuję do programu.

Schemat blokowy rozwiązania



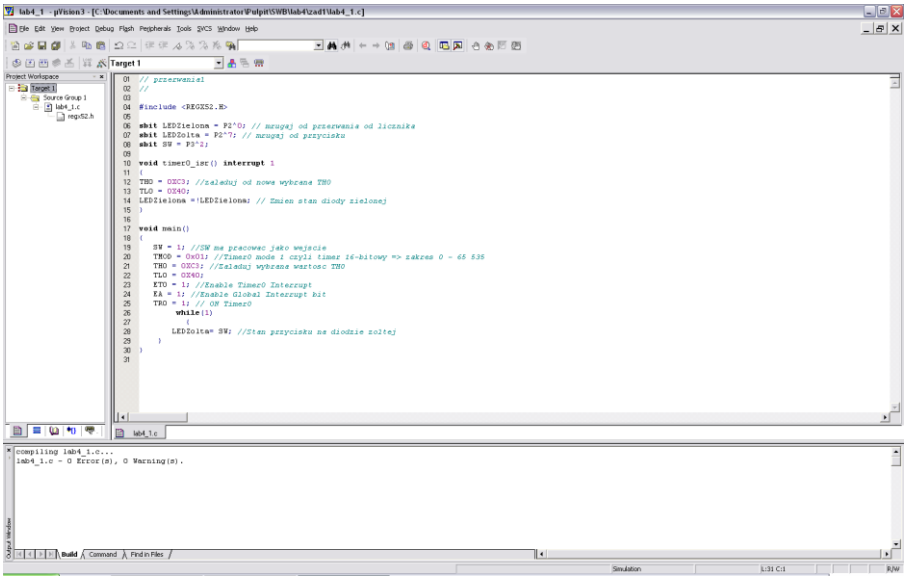
Listing programu

```
// przerwania1
//

#include <REGX52.H>

sbit LEDZielona = P2^0; // mrugaj od przerwania od licznika
sbit LEDZolta = P2^7; // mrugaj od przycisku
sbit SW = P3^2;

void timer0_isr() interrupt 1
{
TH0 = 0XC3; //zaladuj od nowa wybrana TH0
TL0 = 0X40;
LEDZielona =!LEDZielona; // Zmien stan diody zielonej
}

void main()
{
   SW = 1; //SW ma pracowac jako wejscie
   TMOD = 0x01; //Timer0 mode 1 czyli timer 16-bitowy => zakres 0 - 65 535
   TH0 = 0XC3; //Zaladuj wybrana wartosc TH0
   TL0 = 0X40;
   ET0 = 1; //Enable Timer0 Interrupt
   EA = 1; //Enable Global Interrupt bit
   TR0 = 1; // ON Timer0
        while(1)
          {
       LEDZolta= SW; //Stan przycisku na diodzie zoltej
    }
```

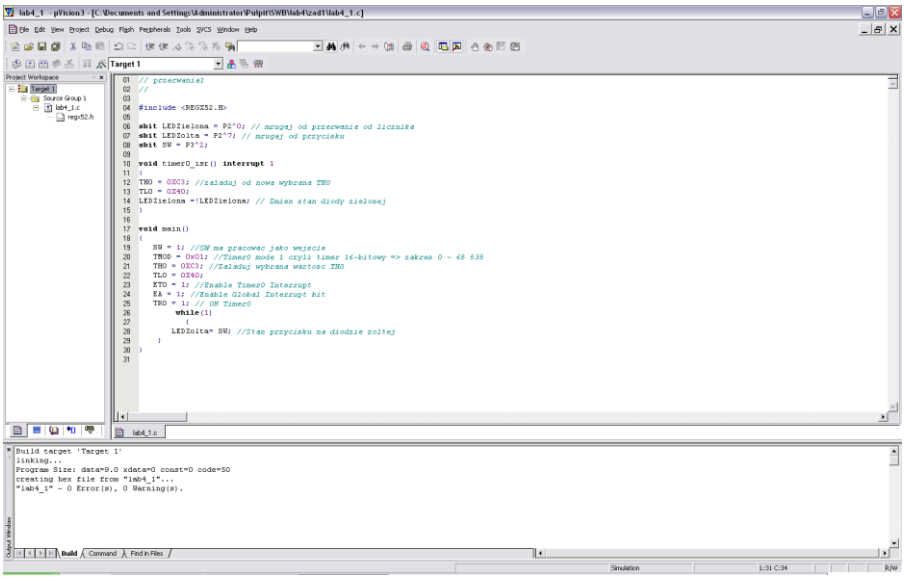
```
}
```

Sprawdzenie poprawności

Kompilowanie

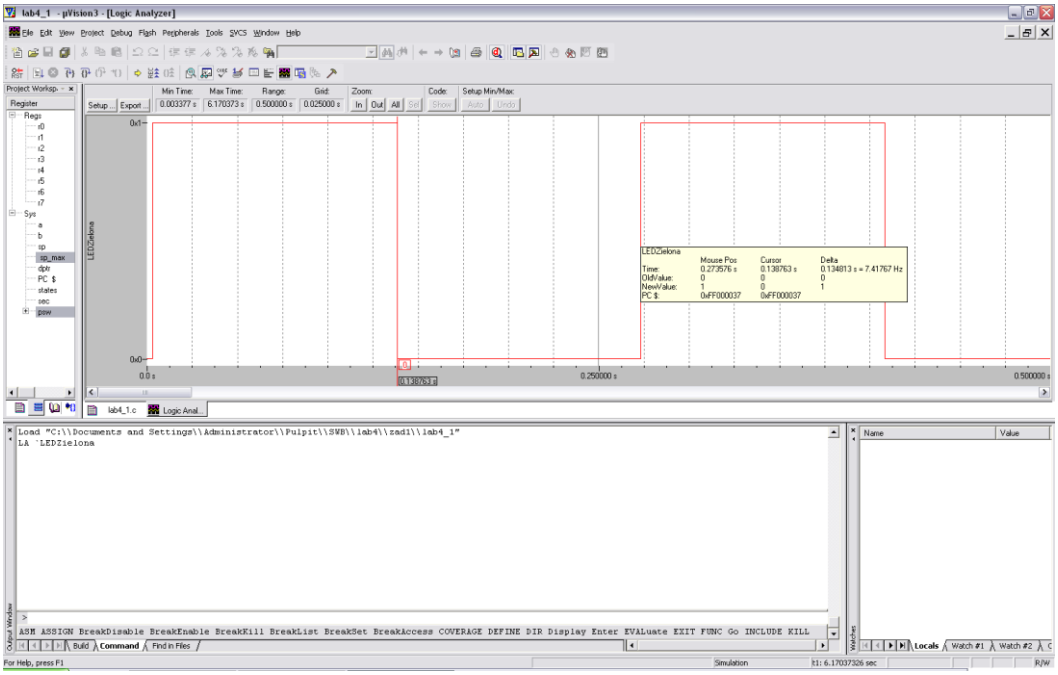


Linkowanie

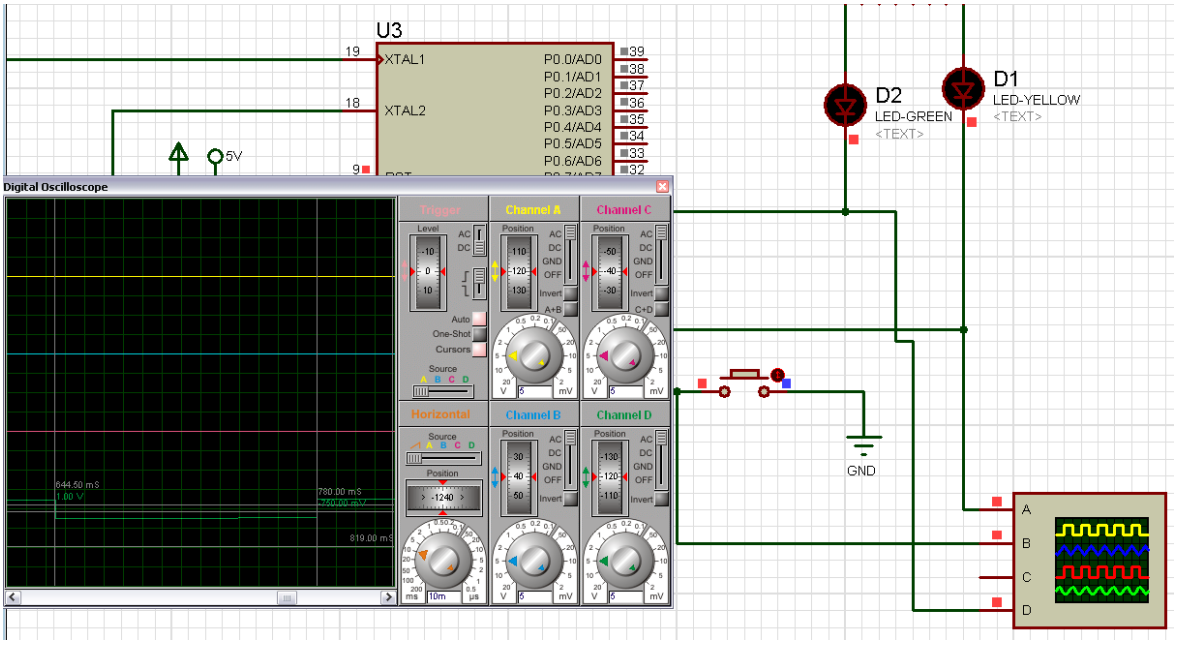


Prezentacja realizacji zadania przez program

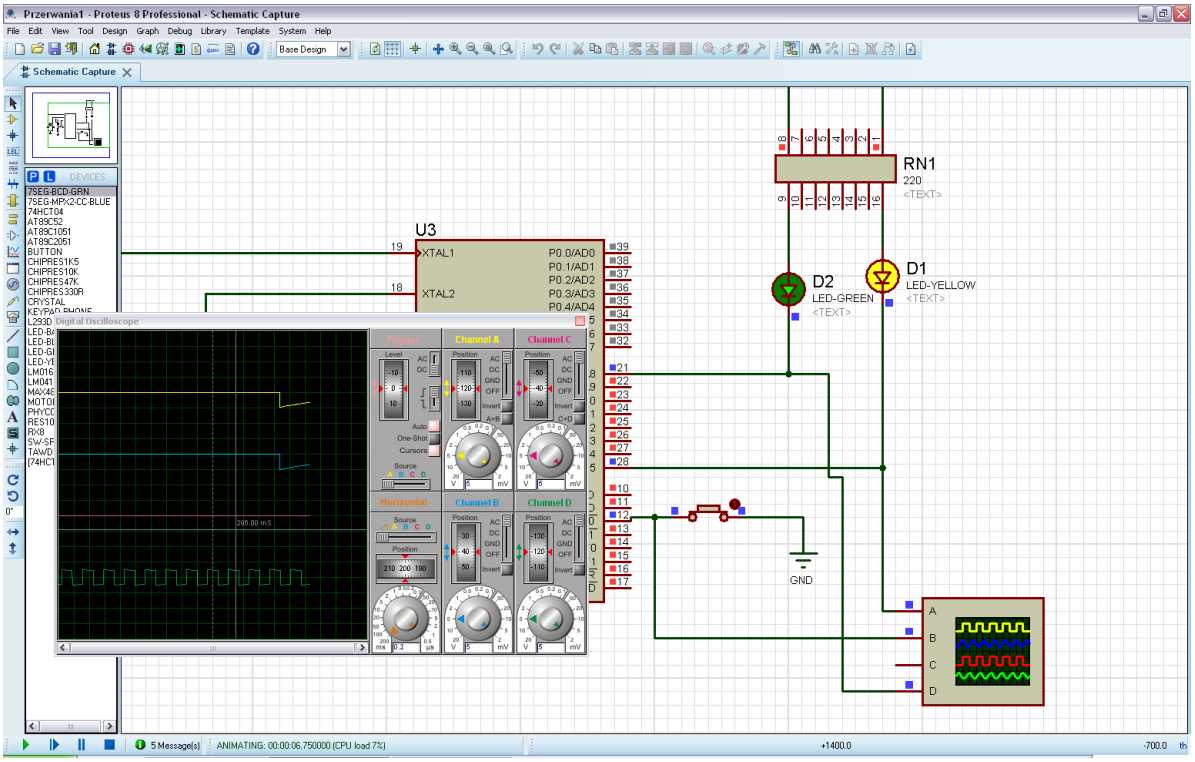
Pomiar czasu w Keilu między zmianami stanu diody (zmiennej „LEDZielona”). Na zrzucie widoczna delta równa w przybliżeniu 0.135s.



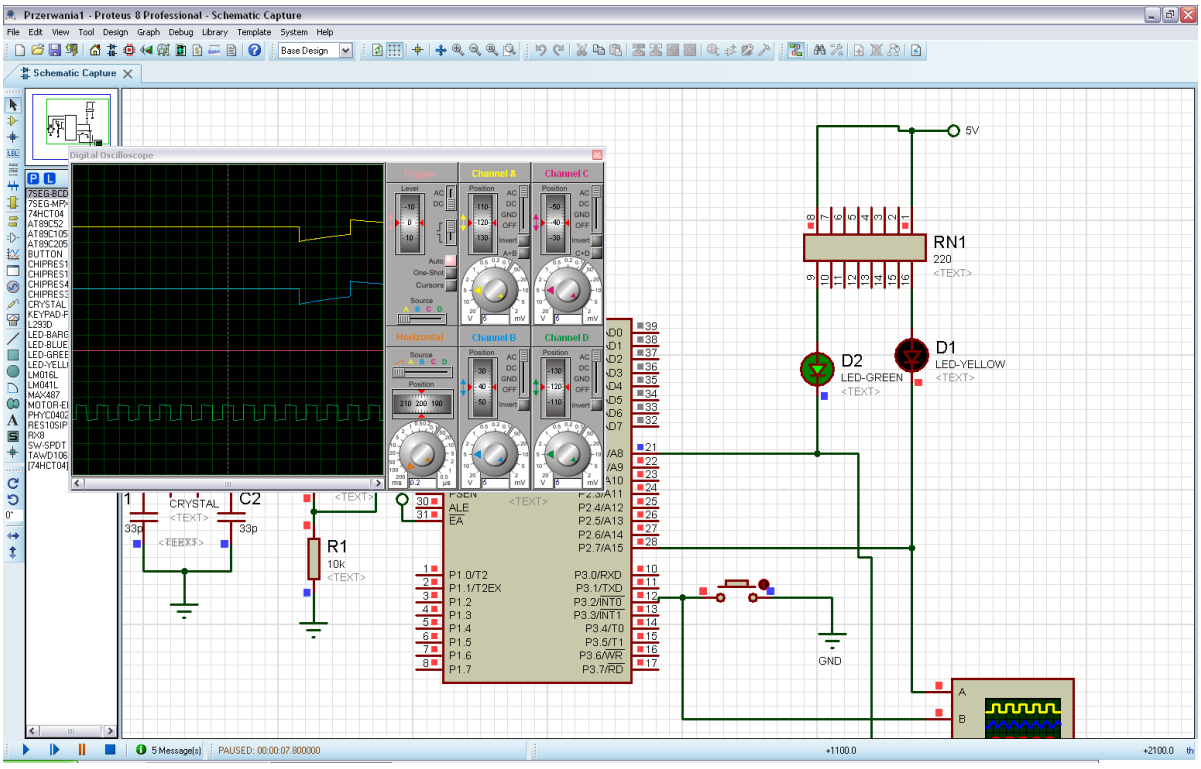
Pomiar czasu w Proteusie między zmianami stanu diody. Na zrzucie widoczny pomiar czasu między dwoma stanami. Różnica między nimi to 780ms-644ms = 136ms (w przybliżeniu spodziewana liczba).



Działanie programu (wciśnięcie przycisku). Dioda żółta się świeci, widać na wyświetlaczu oscyloskopu moment zmiany stanu.



Wciśnięcie i puszczenie przycisku. Widać na wyświetlaczu oscyloskopu moment świecenia się żółtej diody.



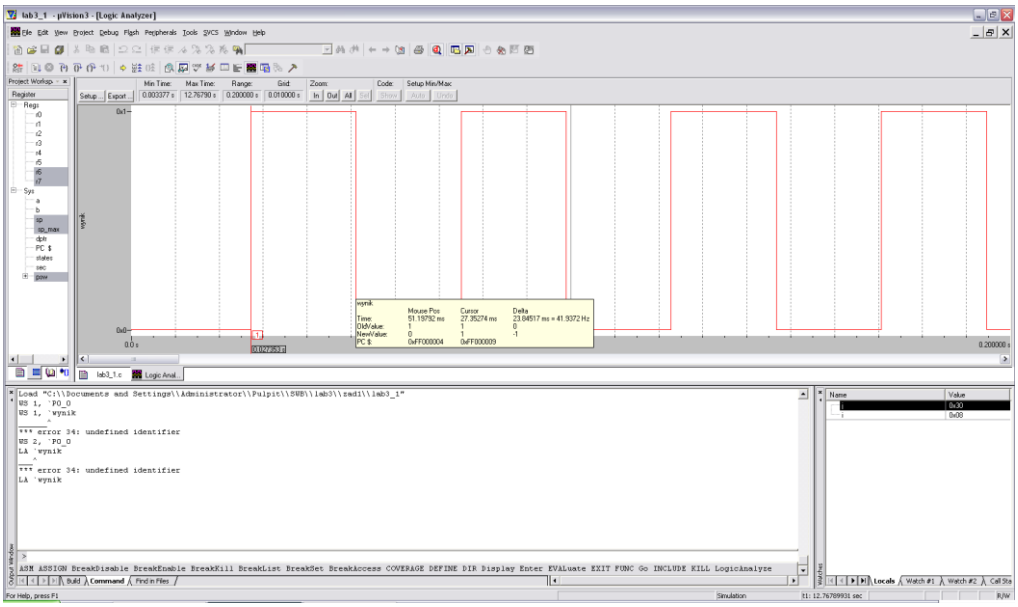
C. Opracowywanie funkcji sleep()

W ramach ćwiczenia potrzebne będzie utworzenie funkcji sleep(), która przez odpowiednie zagnieżdżenie pętli będzie wykonywać się przez około 0,5 sekundu. W tym celu stworzę minimalny program, który pozwoli mi badać i dostosowywać tą funkcję do zamierzonego efektu. Na początku w pętlach ustawię wartości 9 i 100. Funkcja testowa Delay() wygląda następująco:

```
#include <REGX52.H>
void Delay(void);
void main (void)
{
```

```
    unsigned char wynik;
    while(1)
    {
        wynik=0;
        Delay();
        wynik=1;
        Delay();
    }
}
void Delay(void)
{
    unsigned char i, j;
    for(i=0;i<9;i++)
        for(j=0;j<100;j++) {;}
}
```

Czas działania tej funkcji jest pokazany poniżej:



Czas to niecałe 24 ms.

Aktualnie funkcja wykonuje 900 operacji (9 * 100), jak widać na powyższym zrzucie ekran tyle operacji zajmuje programowi ponad sekundę. W związku z tym przy pomocy prostego wzoru przeliczę, ile mniej więcej operacji będzie potrzebnych, aby zbliżyć się do połowy sekundy.

$$ilość\ operacji\ na\ pół\ sekundy = \frac{aktualna\ ilość\ operacji}{aktualny\ czas} * \frac{1}{2} = {}^{900}/_{0{,}02385517} * \frac{1}{2} \approx 18\,871$$

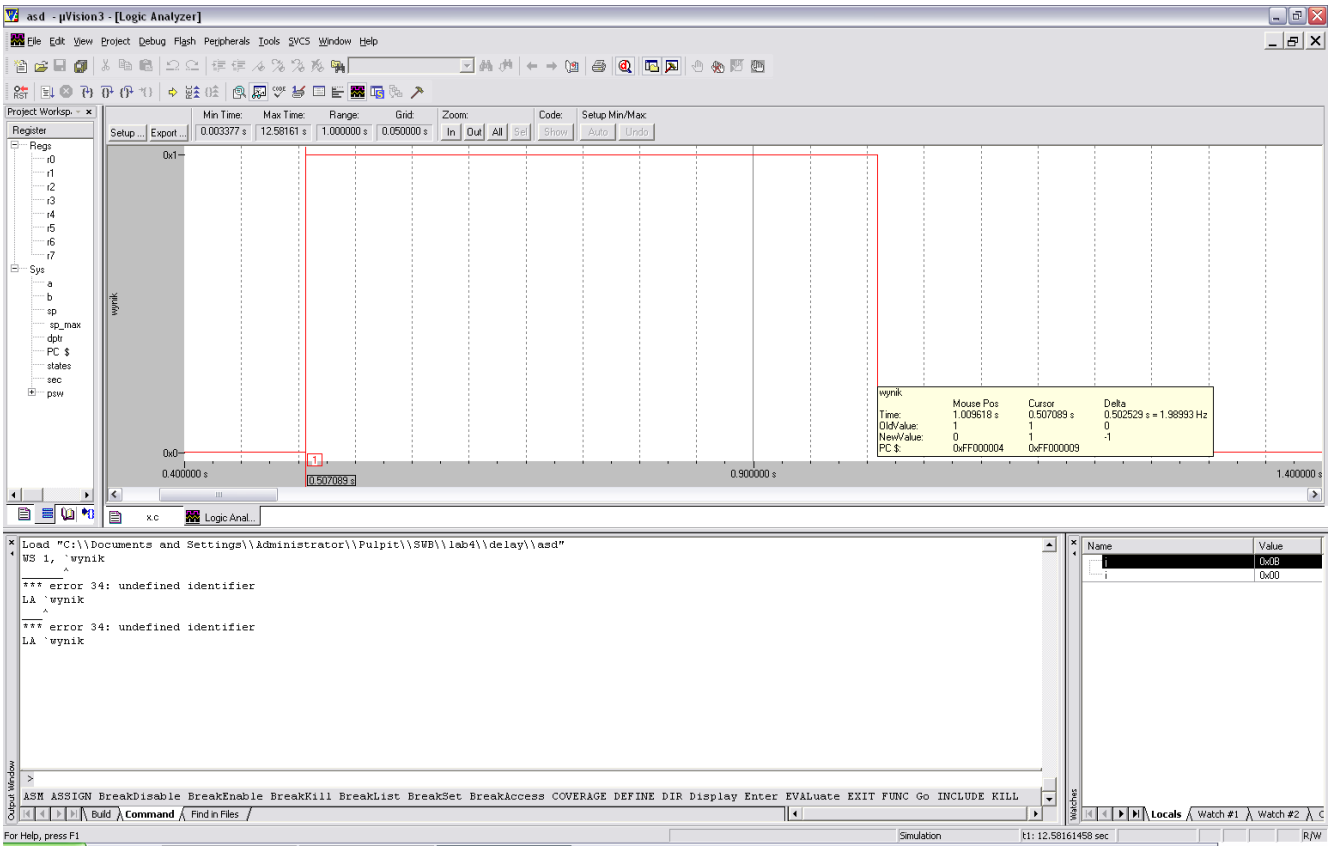
Rozpiszę ten wynik na mniejsze liczby: $18\,871 \approx 190 * 100$

Gotowy kod:

```
#include <REGX52.H>
void Delay(void);
void main (void)
{
    unsigned char wynik;
    while(1)
    {
        wynik=0;
        Delay();
```

```
        wynik=1;
        Delay();
    }
}
void Delay(void)
{
    unsigned char i, j;
    for(i=0;i<190;i++)
        for(j=0;j<100;j++) {;}
}
```

Dzięki temu udało się uzyskać czas 0,507089 sekundy.



Opracowanie funkcji Delay() wykorzystam do funkcji sleep(), stosując te same wartości w pętlach, aby osiągnąć również dla niej czas 0,5s.

D. Zadanie na ocenę dobrą

Opis mojego rozwiązania

Do tego zadania wykorzystam obliczoną funkcję sleep(). Program ma za zadanie dla każdego przycisku wywołać ilość błysków równych numerowi kolumny przycisku, a następnie jeszcze 1 raz błyśnie.

Do tego zadania ustawiam tryb timera 1 (włączony: TR1=1) na 8-bitowy z zadaną wartością początkową oraz automatyczne ładowanie (TMOD=0x22, timer0 jest wyłączony: TCON=0x10). Wartości początkowe timera1 to TH1=0x00 oraz TL1=0x00. Włączam przerwania i ich obsługę (EA=1, EX0=1, IT0=1).

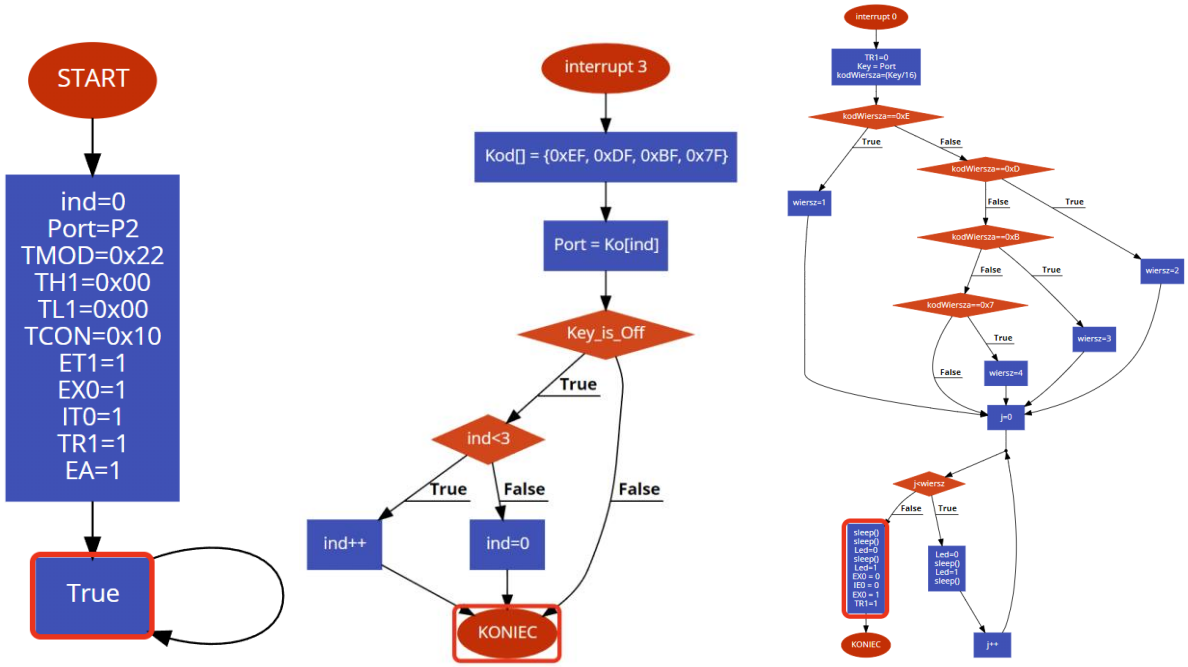
Wędrujące zero w klawiaturze jest rozwiązane przy pomocy wspomnianego timera 1, którego licznik poprzez przepełnienie wywołuje interrupt 3 zmieniający położenie zera na łącza klawiatury.

Pobieranie przycisku odbywa się w poniższej kolejności:

1. Wywołanie przerwania (interrupt 0)

2. Wyłączenie timera 1.
3. Pobranie kodu klawisza i obliczenie jego numeru wiersza.
4. Wywołanie sekwencji błysków (wiersze, przerwa, błyśnięcie).
5. Wyłączenie przerwania, ponowne włączenie timera 1.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>     // plik nagl. rej. procesora

#define Port P2
#define No_Key 0x0C
#define Key_is_Off P3_2==1
#define ON  0
#define OFF 1

volatile unsigned char data ind; // indeks wiersza klawiatury, musi byc globalny,
uzywany w proc IRQ
sbit Led = P0^0;

void Init(void)                     // Funkcja inicjujaca elementy wewnetrzne uC
{
    TMOD = 0x22;            // Konfiguracja Timera 1
    TH1  = 0x00;            // konfiguracja Timera 1
    TL1  = 0x00;            // konfiguracja Timera 1
    TCON = 0x10;            // konfiguracja Timera 1
    ET1  = 1;               // dolaczenie przerwania od Timer1 do systemu przerwan

    EX0 = 1;                // wlaczenie przerwania od INT0
    IT0 = 1;                // przerwanie aktywne zboczem opadajacym

//    SCON  = 0x40;           // serial control - 8-Bit var. Baud Rate
//    TCLK  = 1;              // dolacz taktowanie do nadajnika portu szeregowego
//    T2CON = 0x10;           // Timer2 jako Baud Rate Generator 2400b/sek
//    RCAP2H = TH2 = 0xFF;    // wartosc poczatkowa licznika T2 - czesc H
```

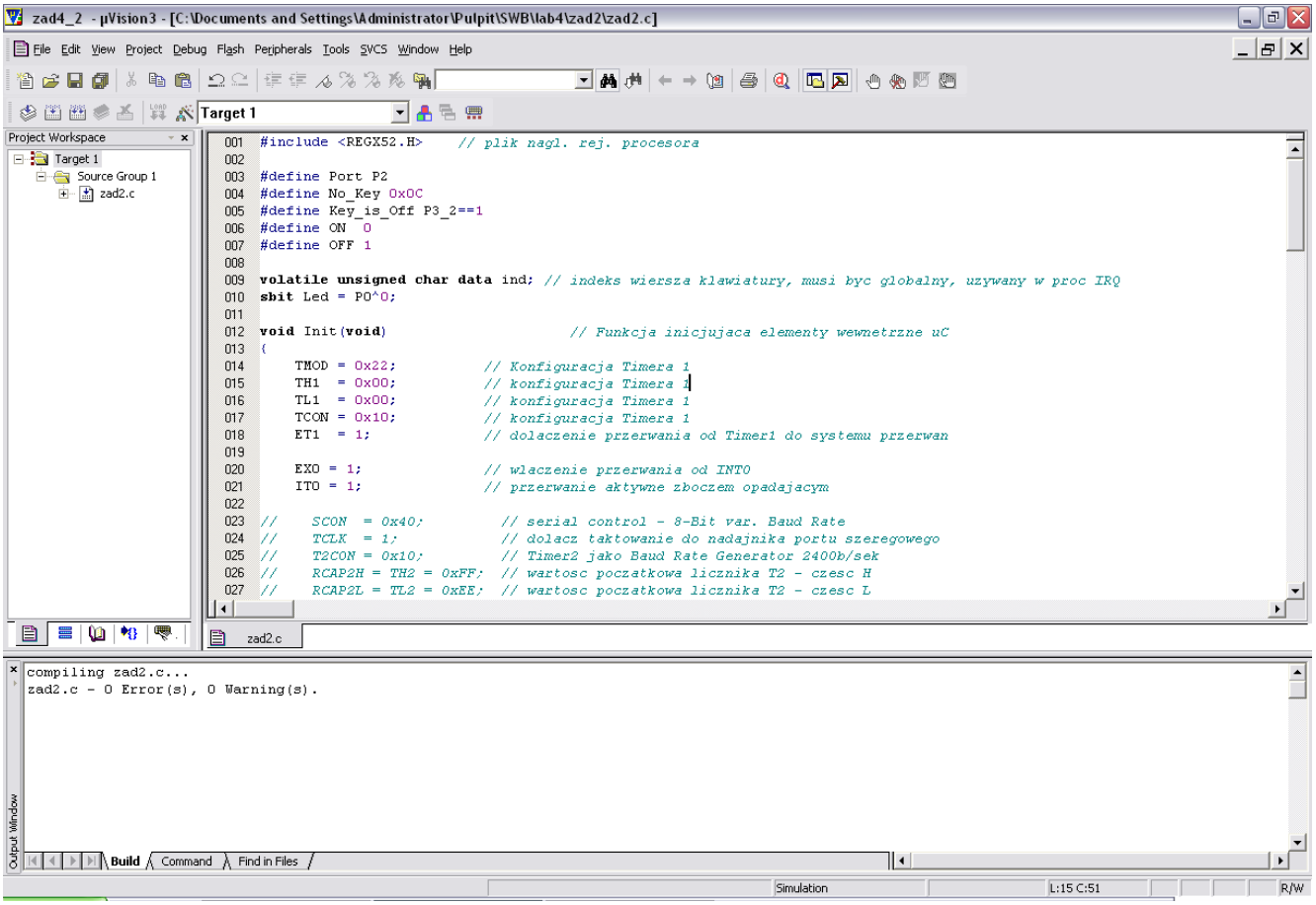
```
//    RCAP2L = TL2 = 0xEE;  // wartosc poczatkowa licznika T2 - czesc L
//    TR2 = 1;              // uruchom Timer 2 - taktowanie nadajnika RS
    TR1 = 1;              // wlaczenie timera 1
    EA = 1;               // wlaczenie systemu przerwan
}

//unsigned char Szukaj(unsigned char Kod_przycisku) // zamiana scankodu na dowolny
kod ASCII
//{
//    unsigned char data Value, licznik;                              // zmienne
pomocnicze
//    unsigned char code Koder[] = {0xE7, 0xEB, 0xED, 0xD7, 0xDB, 0xDD,
//                                  0xB7, 0xBB, 0xBD, 0x7B, 0x77, 0x7D};    //
tablica scankodow
//    unsigned char code New_Char[] = {'1','2','3','4','5','6',
//                                     '7','8','9','0','*','#'}; // tablica kod?w
ASCII
//
//    licznik=0x00;                                                   // zacznij od
poczatku
//    while((Koder[licznik]!=Kod_przycisku)&(licznik<No_Key)){licznik++;} // znajdz
scankod
//    Value = No_Key;
//    if(licznik!=No_Key){Value = New_Char[licznik];}                 // przypisz
kod ASCII
//    return Value;                                                   // zwroc kod
ASCII lub No_Key
//}

void ISR_Timer1 (void) interrupt 3  // Timera 1 - generacja krazacego zera na porcie
P2
{
unsigned char code Kod[] = {0xEF, 0xDF, 0xBF, 0x7F};     // skanuj kolejne wiersze
klawiatury

Port = Kod[ind];                                    // wystawienie zera na port
if(Key_is_Off) {if(ind<3) {ind++;} else {ind=0;}} // Wazne, bramka wprowadza
opoznienie 10ns.
// Zmiana indeksu tablicy generowana jest tylko
// wtedy, gdy zaden przycisk nie jest wcisniety
}

void sleep(void)              // odczekaj chwile
{
    unsigned char data x,y;
    for(x=0;x<190;x++)
        for(y=0;y<100;y++){;}
}

//void sleep_half(void)             // odczekaj chwile
//{
//    unsigned char data x,y;
//    for(x=0;x<190;x++)
```

```
//        for(y=0;y<50;y++){;}
//}

void ISR_INT0 (void) interrupt 0      // INT0 - wylapywanie wcisnietych przycisku
{
unsigned char data Key, j, kodWiersza, wiersz;  // zmienne pomocnicze
TR1=0;               // wylacz timer 1
Key = Port;               // pobranie kodu klawisza
//Value = Szukaj(Key);     // okresl kod ASCII przycisku
//if(Value!=No_Key)        // wykonaj jesli kod ASCII jest prawidlowy
//{
//Led=0;
//sleep_half();
//Led=1;
//sleep();

kodWiersza=(Key/16);
if(kodWiersza==0xE) wiersz=1;
else if(kodWiersza==0xD) wiersz=2;
else if(kodWiersza==0xB) wiersz=3;
else if(kodWiersza==0x7) wiersz=4;

for(j=0; j<wiersz; j++){
Led=0;
sleep();
Led=1;
sleep();
}
sleep();
sleep();
Led=0;
sleep();
Led=1;

EX0 = 0;             // wylacz przerwanie INT0 (czytanie przyciskow)
//TI = 0;             // przygotuj wysylanie znakow
//SBUF = Value;      // Obsluga: transmisja znaku portem szeregowym
//while(TI==0){;}    // odczekaj dopoki znak jest transmitowany
IE0 = 0;             // na wszelki wypadek wygas flage przerwania od INT0
EX0 = 1;             // wlacz obsluge INT0
//}
TR1=1;                  // wlacz timer 1
}
void Default(void)
{
    ind  = 0;              // indeks wiersza klawiatury
    Port = P2;              // inicjowanie portu klawiatury
}

void main(void)  // program glowny
{
    Default();   // inicjowanie zmiennych
    Init();      // inicjowanie urzadzen wewnetrznych
```

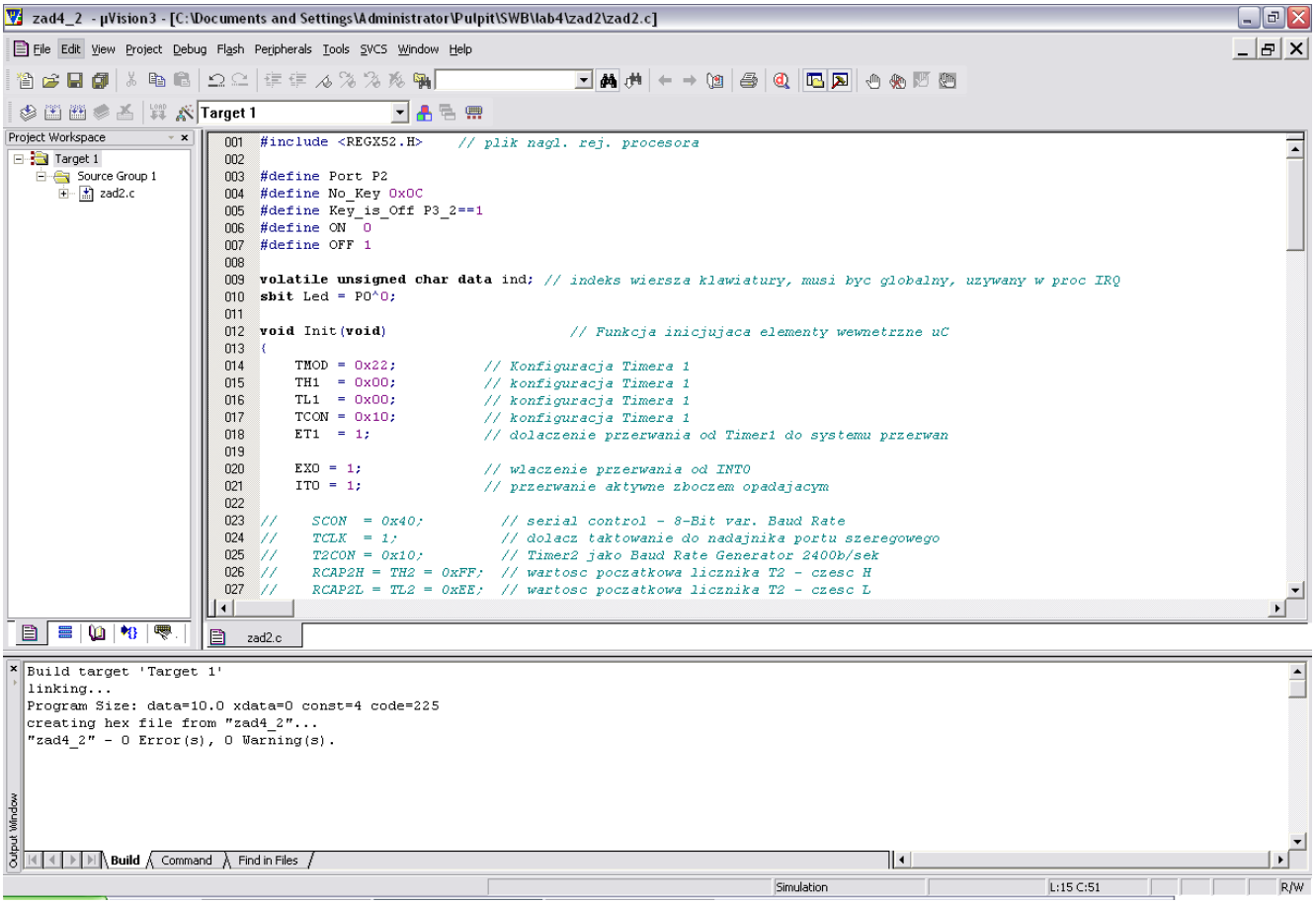
```
    while(1) {;} // pusta petla programu, a program dziala :)
}
```

Sprawdzenie poprawności

Kompilowanie

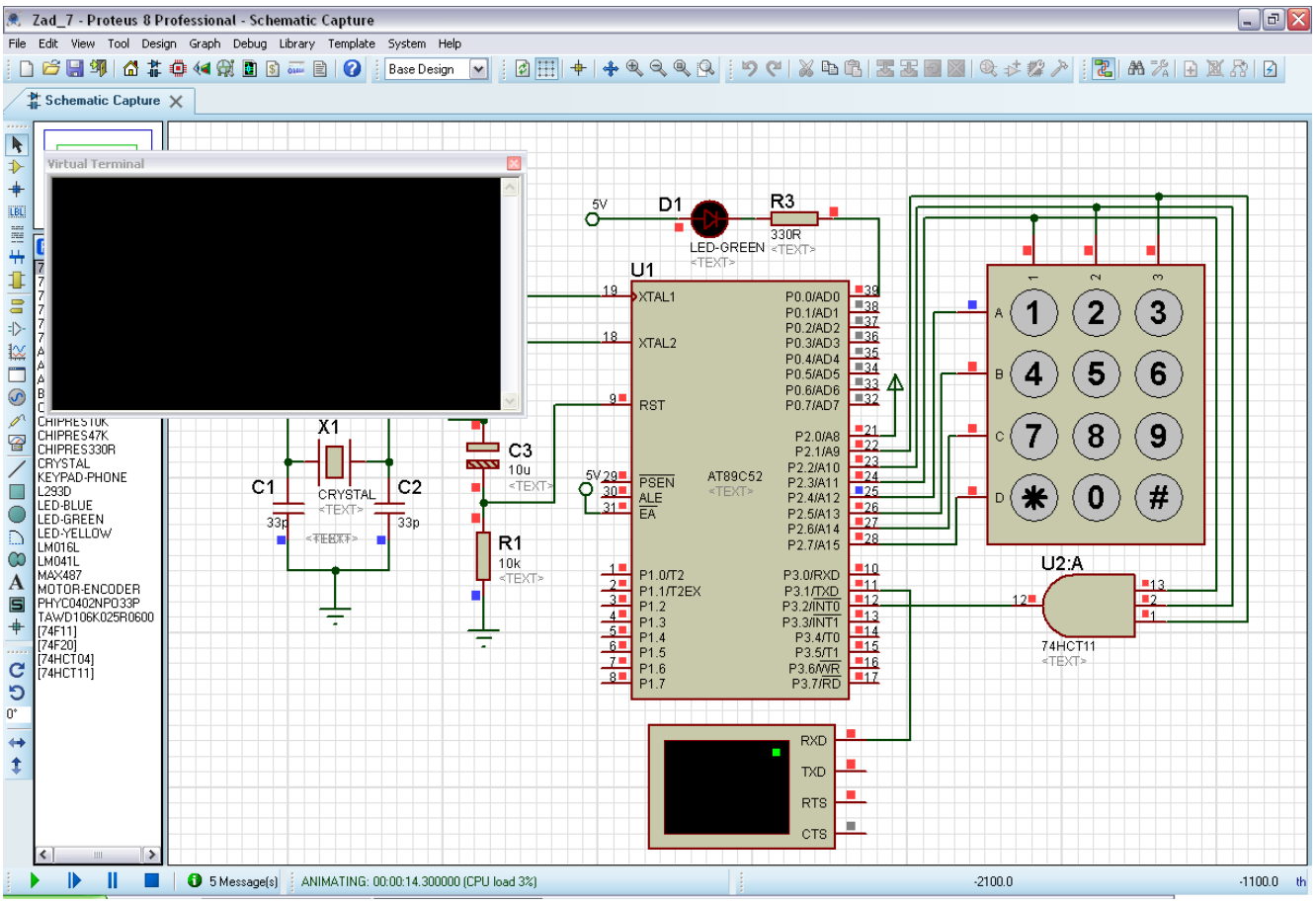


Linkowanie

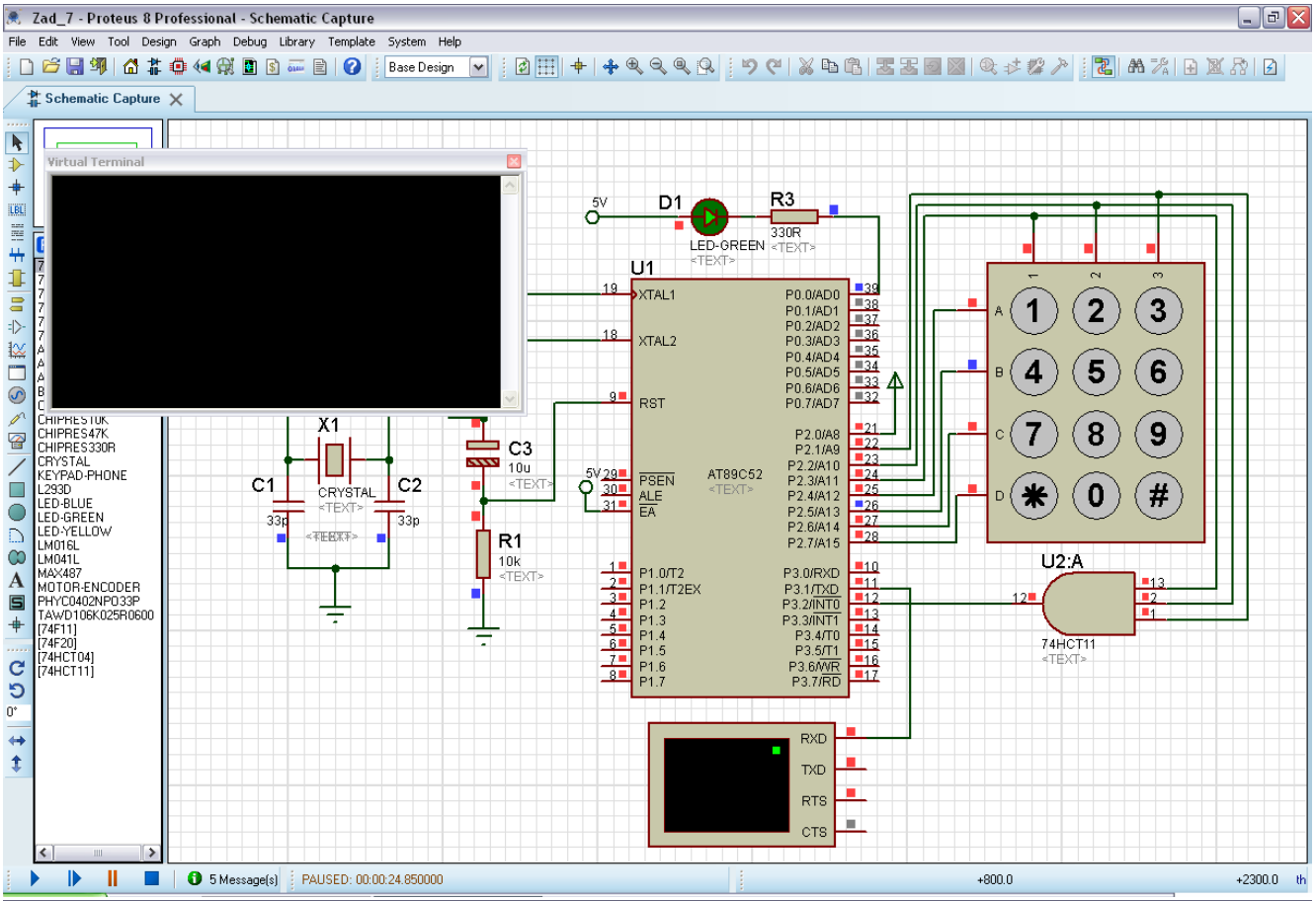


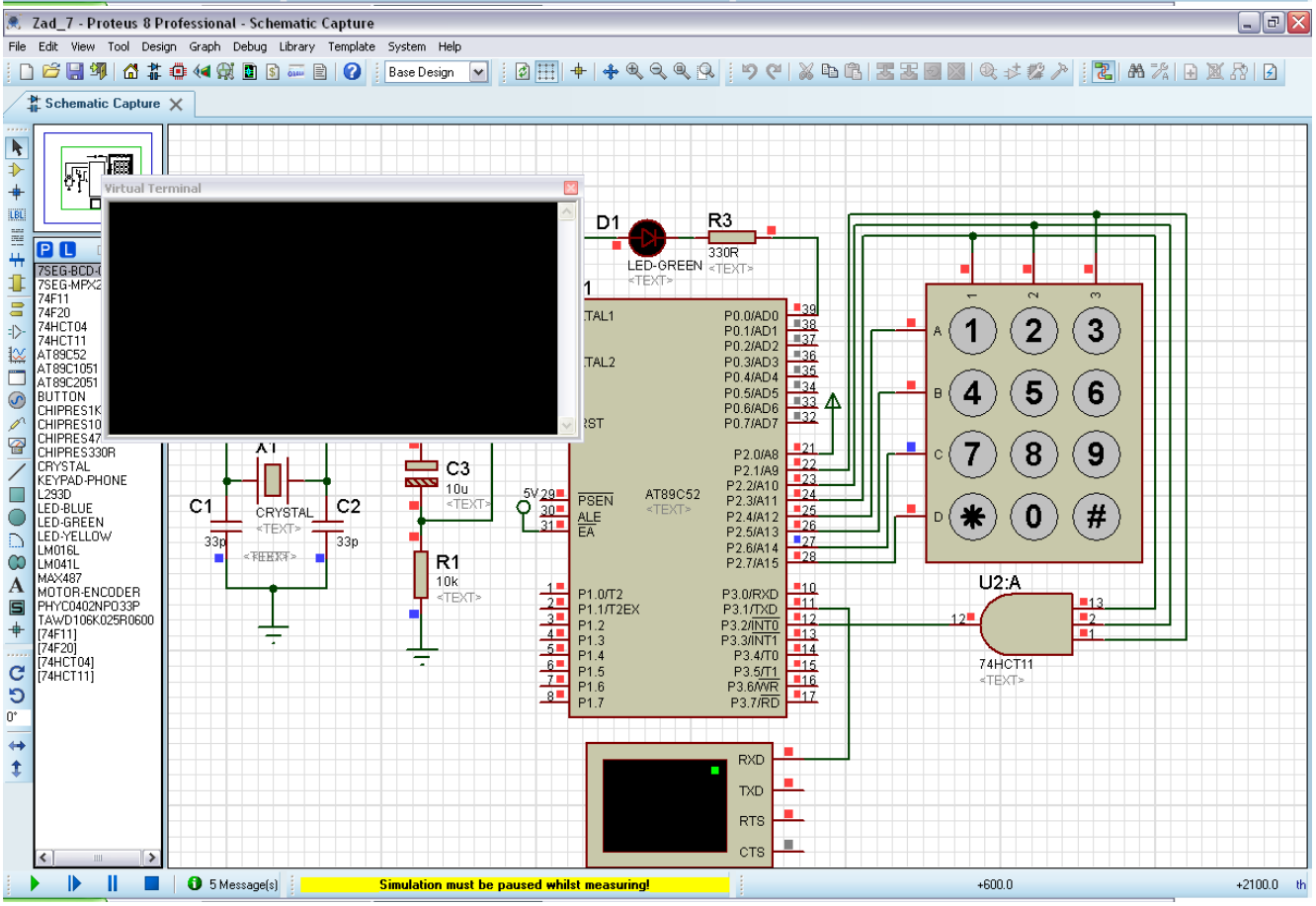
Prezentacja realizacji zadania przez program

Stan początkowy, wędrujące zero



Wciśnięcie przycisku 4 – 2 błyski, przerwa, 1 błysk (terminal dalej pusty, nawet po zakończeniu działania)





E. Zadanie na ocenę bardzo dobrą

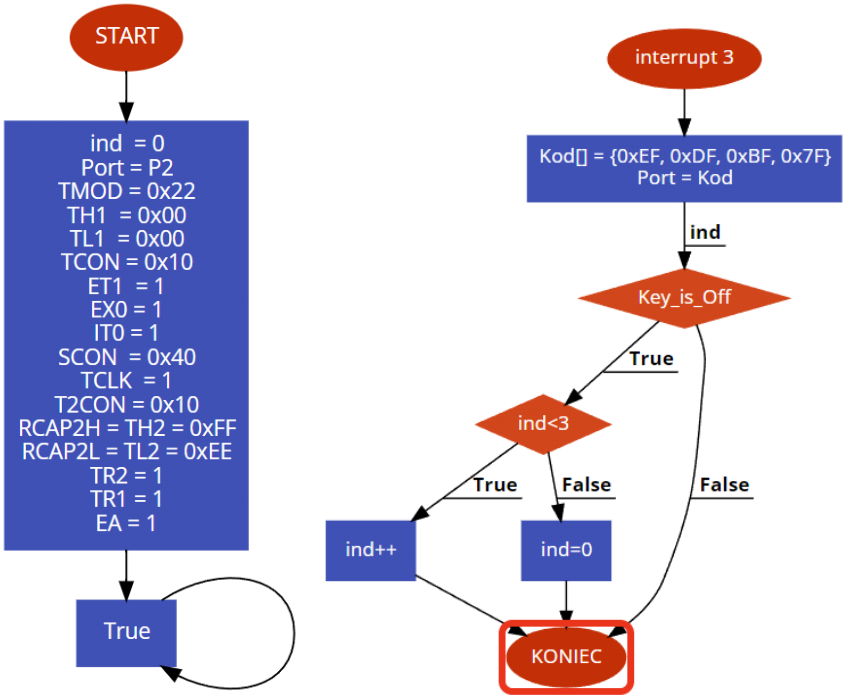
Opis mojego rozwiązania

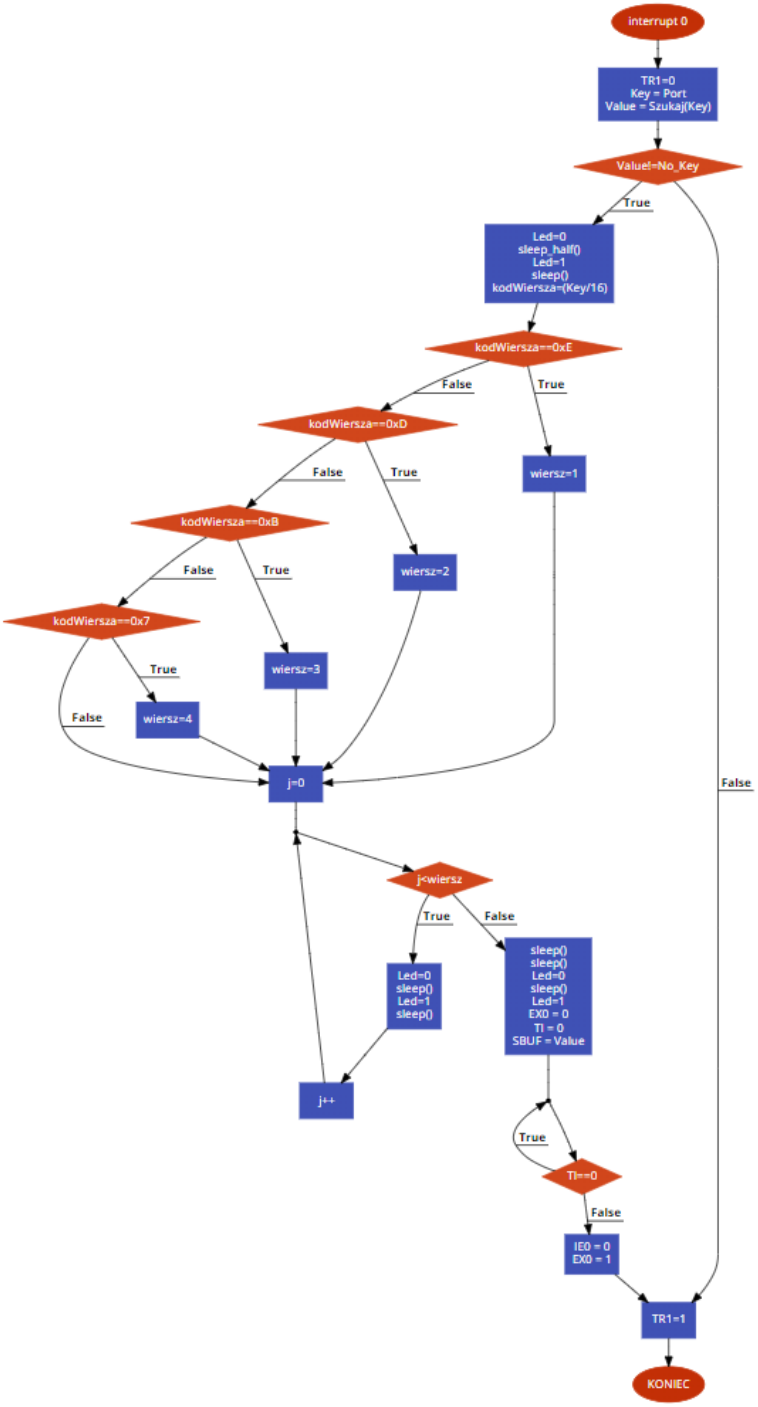
W zadaniu tym wykorzystuję poprzedni program i rozszerzam go o następujące elementy:

1. Zmienne do obsługi elementów związanych z wyświetlaniem na terminalu
    a. SCON = 0x40; // serial control, tryb pracy portu szeregowego jako transmisja 8-bitowa.
    b. TCLK = 1; // dołączenie taktowania do nadajnika portu szeregowego.
    c. T2CON = 0x10; // Baud Rate Generator, timer2 będzie źródłem impulsów zegarowych
    d. RCAP2H = TH2 = 0xFF; // Wartość początkowa timera 2
    e. RCAP2L = TL2 = 0xEE; // Wartość początkowa timera 2
    f. TR2 = 1; // Włączenie timera 2
2. Dodanie funkcji „Szukaj(Kod_przycisku)”
    a. Zawiera tablice scankodów oraz tablice kodów ASCII wszystkich znaków klawiatury.
    b. Odpowiada za znalezienie bieżącego znaku i zwrócenie na jej podstawie odpowiedniej wartości (do wyświetlenia na terminalu).
3. Rozszerzenie działania dla interrupt 0:
    a. Pobieranie kodu klawisza na podstawie funkcji „Szukaj()”.
    b. Sprawdzenie, czy został zwrócony kod (jeśli nie, kończy przerwanie).
    c. Jedno początkowe błyśnięcie na początku (o długości 250ms, funkcja „half_sleep()”).
    d. Transmisja znaku (przygotowanie TI=0, wysłanie znaku SBUF=Value, czekanie na koniec wysyłania znaku).

Dzięki implementacji tych 3 zagadnień rozszerzony program oprócz wykonania zadania na ocenę dobrą dodatkowo przed błyskami wiersza jest jedno dodatkowe błyśnięcie oraz po zakończeniu cyklu obsługi wciśnięcia na terminali wyświetli się wciśnięty znak.

Schemat blokowy rozwiązania


START
ind = 0
Port = P2
TMOD = 0x22
TH1 = 0x00
TL1 = 0x00
TCON = 0x10
ET1 = 1
EX0 = 1
IT0 = 1
SCON = 0x40
TCLK = 1
T2CON = 0x10
RCAP2H = TH2 = 0xFF
RCAP2L = TL2 = 0xEE
TR2 = 1
TR1 = 1
EA = 1
True
interrupt 3
Kod[] = {0xEF, 0xDF, 0xBF, 0x7F}
Port = Kod
ind
Key_is_Off
True
False
ind<3
True
False
ind++
ind=0
KONIEC



Listing programu

```
#include <REGX52.H>     // plik nagl. rej. procesora

#define Port P2
#define No_Key 0x0C
#define Key_is_Off P3_2==1
#define ON  0
#define OFF 1

volatile unsigned char data ind; // indeks wiersza klawiatury, musi byc globalny,
uzywany w proc IRQ
sbit Led = P0^0;

void Init(void)                     // Funkcja inicjujaca elementy wewnetrzne uC
{
```

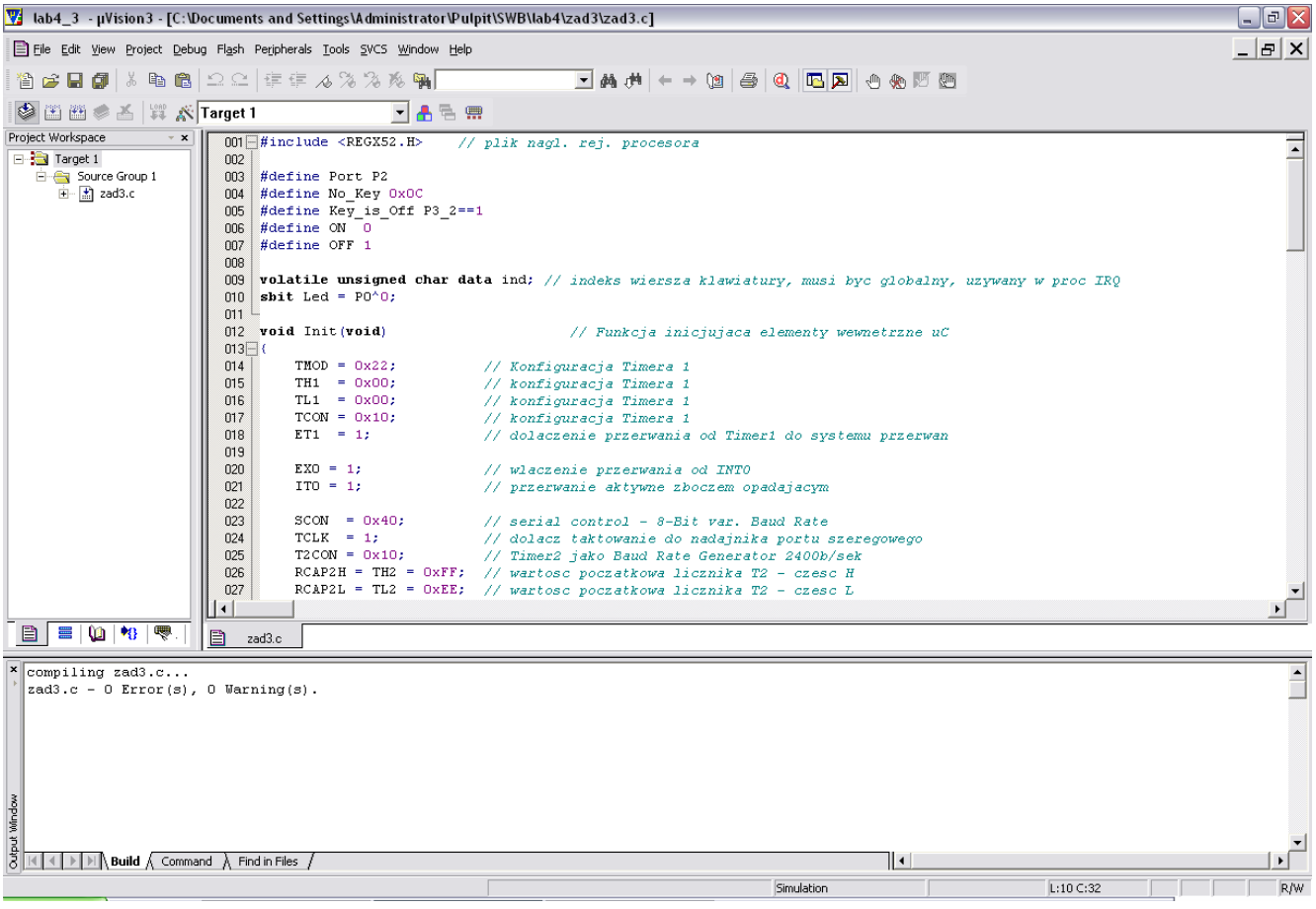
```
    TMOD = 0x22;          // Konfiguracja Timera 1
    TH1  = 0x00;          // konfiguracja Timera 1
    TL1  = 0x00;          // konfiguracja Timera 1
    TCON = 0x10;          // konfiguracja Timera 1
    ET1  = 1;             // dolaczenie przerwania od Timer1 do systemu przerwan

    EX0 = 1;              // wlaczenie przerwania od INT0
    IT0 = 1;              // przerwanie aktywne zboczem opadajacym

    SCON  = 0x40;         // serial control - 8-Bit var. Baud Rate
    TCLK  = 1;            // dolacz taktowanie do nadajnika portu szeregowego
    T2CON = 0x10;         // Timer2 jako Baud Rate Generator 2400b/sek
    RCAP2H = TH2 = 0xFF;  // wartosc poczatkowa licznika T2 - czesc H
    RCAP2L = TL2 = 0xEE;  // wartosc poczatkowa licznika T2 - czesc L
    TR2 = 1;              // uruchom Timer 2 - taktowanie nadajnika RS
    TR1 = 1;              // wlaczenie timera 1
    EA = 1;               // wlaczenie systemu przerwan
}

unsigned char Szukaj(unsigned char Kod_przycisku) // zamiana scankodu na dowolny kod
ASCII
{
    unsigned char data Value, licznik;                              // zmienne
pomocnicze
    unsigned char code Koder[] = {0xE7, 0xEB, 0xED, 0xD7, 0xDB, 0xDD,
                                  0xB7, 0xBB, 0xBD, 0x7B, 0x77, 0x7D};    // tablica
scankodow
    unsigned char code New_Char[] = {'1','2','3','4','5','6',
                                     '7','8','9','0','*','#'}; // tablica kod?w ASCII

    licznik=0x00;                                                   // zacznij od
poczatku
    while((Koder[licznik]!=Kod_przycisku)&(licznik<No_Key)){licznik++;} // znajdz
scankod
    Value = No_Key;
    if(licznik!=No_Key){Value = New_Char[licznik];}                 // przypisz kod
ASCII
    return Value;                                                   // zwroc kod
ASCII lub No_Key
}

void ISR_Timer1 (void) interrupt 3  // Timera 1 - generacja krazacego zera na porcie
P2
{
unsigned char code Kod[] = {0xEF, 0xDF, 0xBF, 0x7F};     // skanuj kolejne wiersze
klawiatury

Port = Kod[ind];                                    // wystawienie zera na port
if(Key_is_Off) {if(ind<3) {ind++;} else {ind=0;}} // Wazne, bramka wprowadza
opoznienie 10ns.
// Zmiana indeksu tablicy generowana jest tylko
// wtedy, gdy zaden przycisk nie jest wcisniety
}
```

```
void sleep(void)              // odczekaj chwile
{
    unsigned char data x,y;
    for(x=0;x<190;x++)
        for(y=0;y<100;y++){;}
}

void sleep_half(void)              // odczekaj chwile
{
    unsigned char data x,y;
    for(x=0;x<190;x++)
        for(y=0;y<50;y++){;}
}

void ISR_INT0 (void) interrupt 0      // INT0 - wylapywanie wcisnietych przycisku
{
unsigned char data Value, Key, j, kodWiersza, wiersz;  // zmienne pomocnicze
TR1=0;                // wylacz timer 1
Key = Port;                // pobranie kodu klawisza
Value = Szukaj(Key);     // okresl kod ASCII przycisku
if(Value!=No_Key)        // wykonaj jesli kod ASCII jest prawidlowy
{
Led=0;
sleep_half();
Led=1;
sleep();

kodWiersza=(Key/16);
if(kodWiersza==0xE) wiersz=1;
else if(kodWiersza==0xD) wiersz=2;
else if(kodWiersza==0xB) wiersz=3;
else if(kodWiersza==0x7) wiersz=4;

for(j=0; j<wiersz; j++){
Led=0;
sleep();
Led=1;
sleep();
}
sleep();
sleep();
Led=0;
sleep();
Led=1;

EX0 = 0;             // wylacz przerwanie INT0 (czytanie przyciskow)
TI = 0;             // przygotuj wysylanie znakow
SBUF = Value;       // Obsluga: transmisja znaku portem szeregowym
while(TI==0){;}     // odczekaj dopoki znak jest transmitowany
IE0 = 0;             // na wszelki wypadek wygas flage przerwania od INT0
EX0 = 1;             // wlacz obsluge INT0
}
```

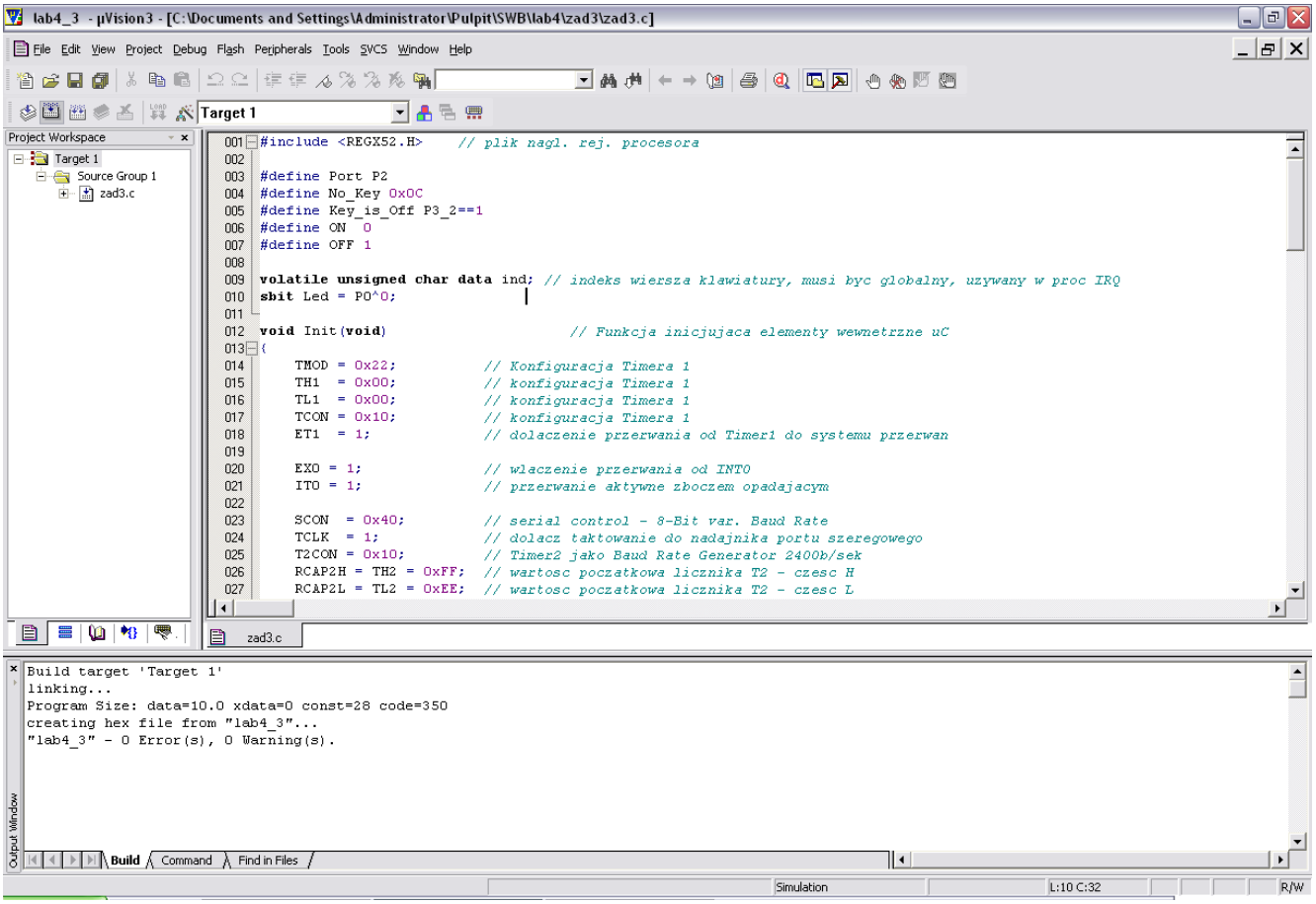
```
TR1=1;                  // wlacz timer 1
}
void Default(void)
{
    ind  = 0;              // indeks wiersza klawiatury
    Port = P2;              // inicjowanie portu klawiatury
}

void main(void)  // program glowny
{
    Default();   // inicjowanie zmiennych
    Init();      // inicjowanie urzadzen wewnetrznych
    while(1) {;} // pusta petla programu, a program dziala :)
}
```

Sprawdzenie poprawności

Kompilowanie

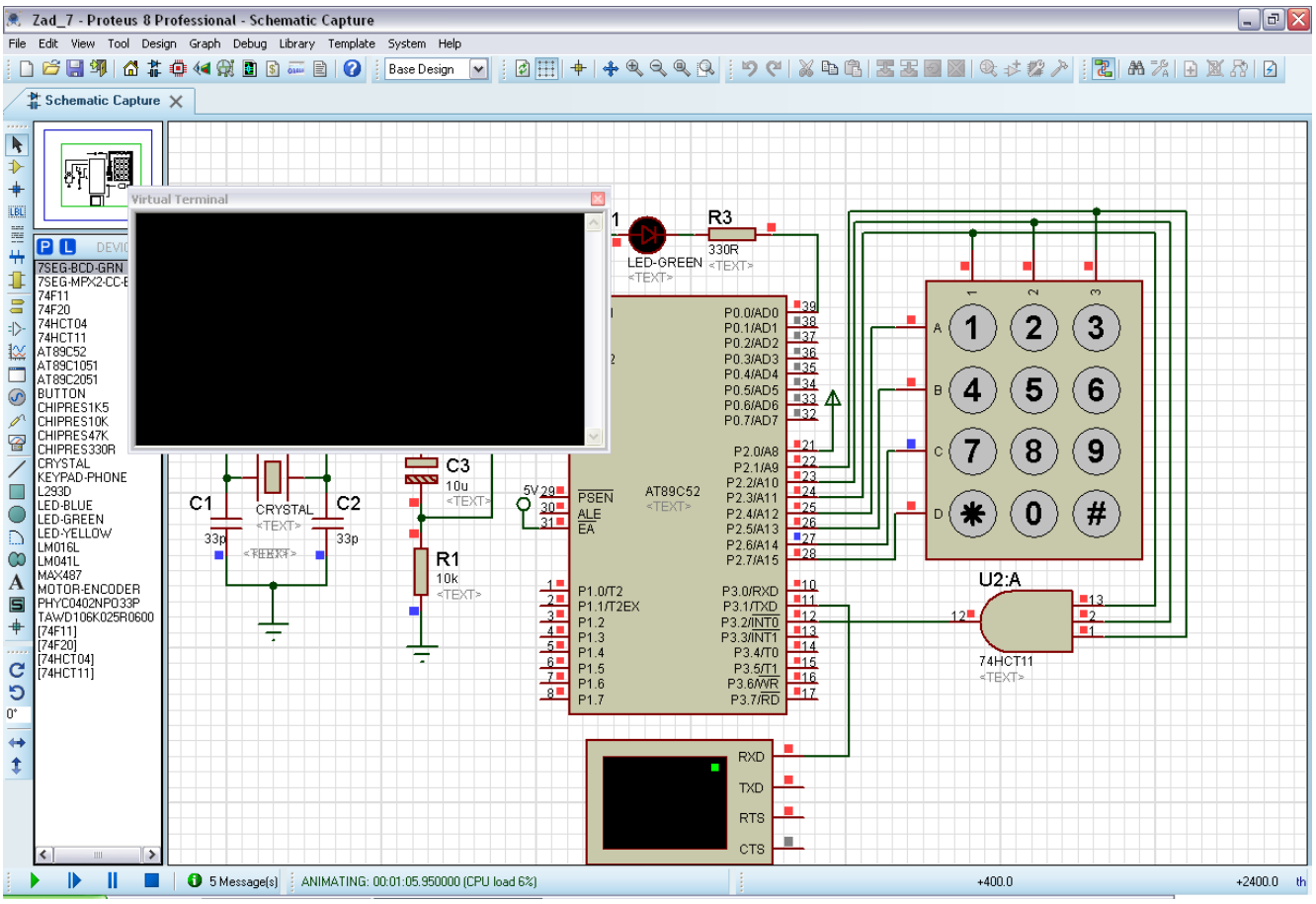


Linkowanie

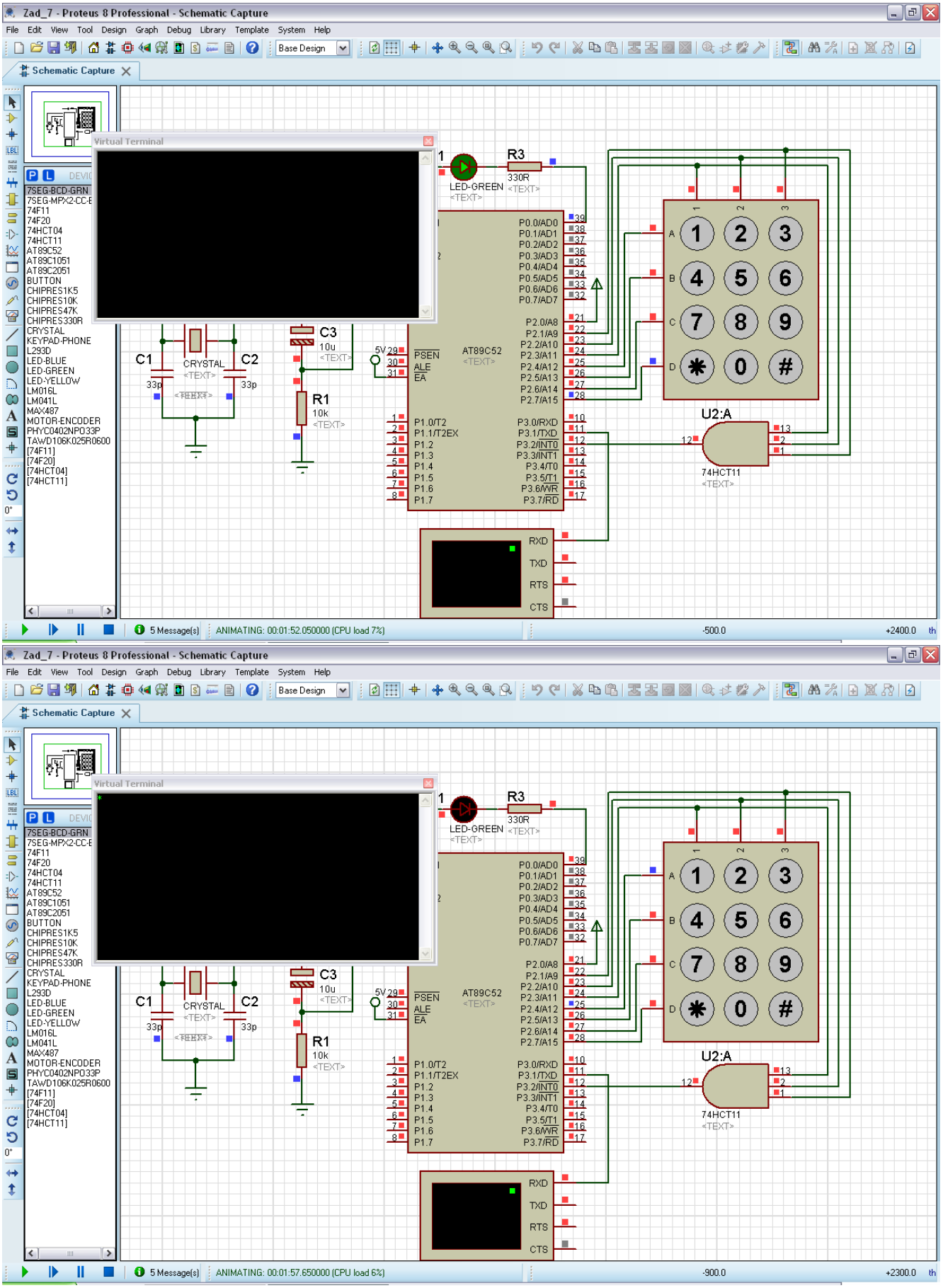


Prezentacja realizacji zadania przez program

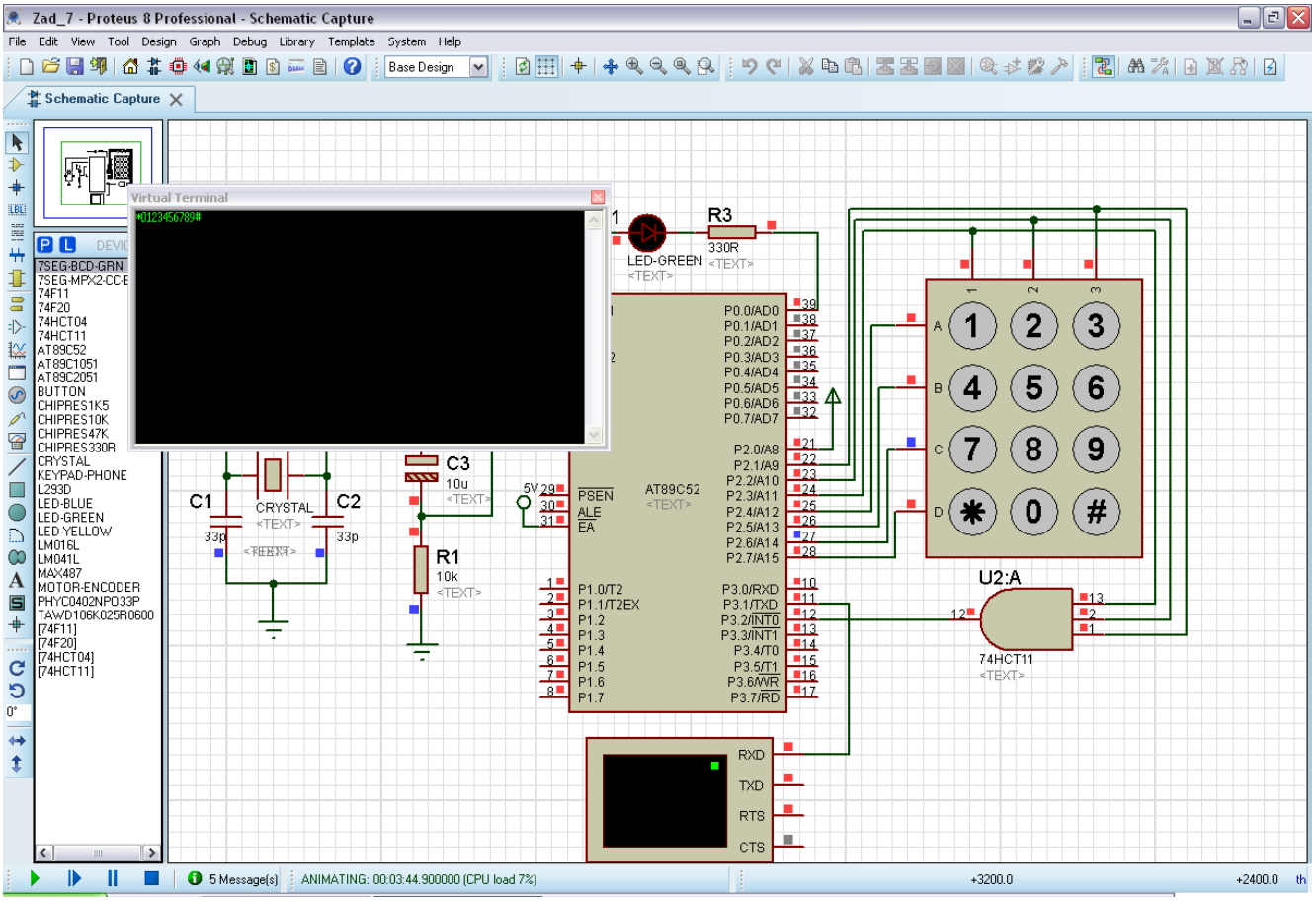
Stan początkowy, wędrujące zero



Wciśnięcie znaku * (błysk początkowy, 4 błyśnięcia, przerwa, jedno błyśnięcie, potem wypisanie znaku)



Wypisanie wszystkich cyfr i znaków


Zad_7 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
Virtual Terminal
*0123456789#
7SEG-BCD-GRN
7SEG-MPX2-CC-B
74F11
74F20
74HCT04
74HCT11
AT89C52
AT89C1051
AT89C2051
BUTTON
CHIPRES1K5
CHIPRES10K
CHIPRES47K
CHIPRES330R
CRYSTAL
KEYPAD-PHONE
L293D
LED-BLUE
LED-GREEN
LED-YELLOW
LM016L
LM041L
MAX487
MOTOR-ENCODER
PHYC0402NP033P
TAWD106K025R0600
[74F11]
[74F20]
[74HCT04]
[74HCT11]
LED-GREEN
R3
330R
C1
CRYSTAL
33p
C2
33p
C3
10u
R1
10k
5V
PSEN
ALE
EA
AT89C52
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
RXD
TXD
RTS
CTS
U2:A
74HCT11
5 Message(s)
ANIMATING: 00:03:44.900000 (CPU load 7%)
+3200.0
+2400.0