Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:

Systemy wbudowane

Sprawozdanie z ćwiczenia laboratoryjnego nr 6:

Sterowanie mocą, transmisja danych

Prowadzący:
mgr inż. Artur Miktus

Wykonał: Radosław Relidzyński
Nr albumu: 76836
Grupa: WCY20IY4S1
Data laboratoriów: 24.06.2022 r.
Deklarowana ocena: 5

Spis treści

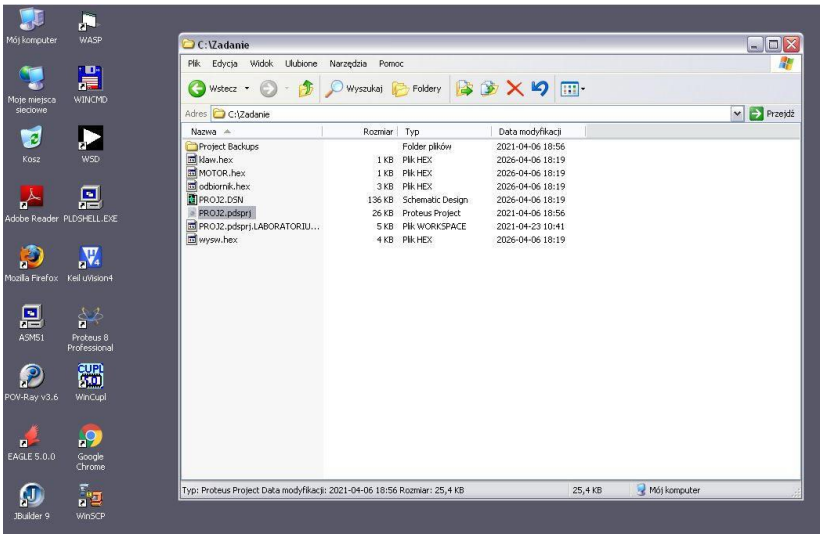
A. Treść zadania

Opanowanie programowania systemu wielomikrokomputerowego.

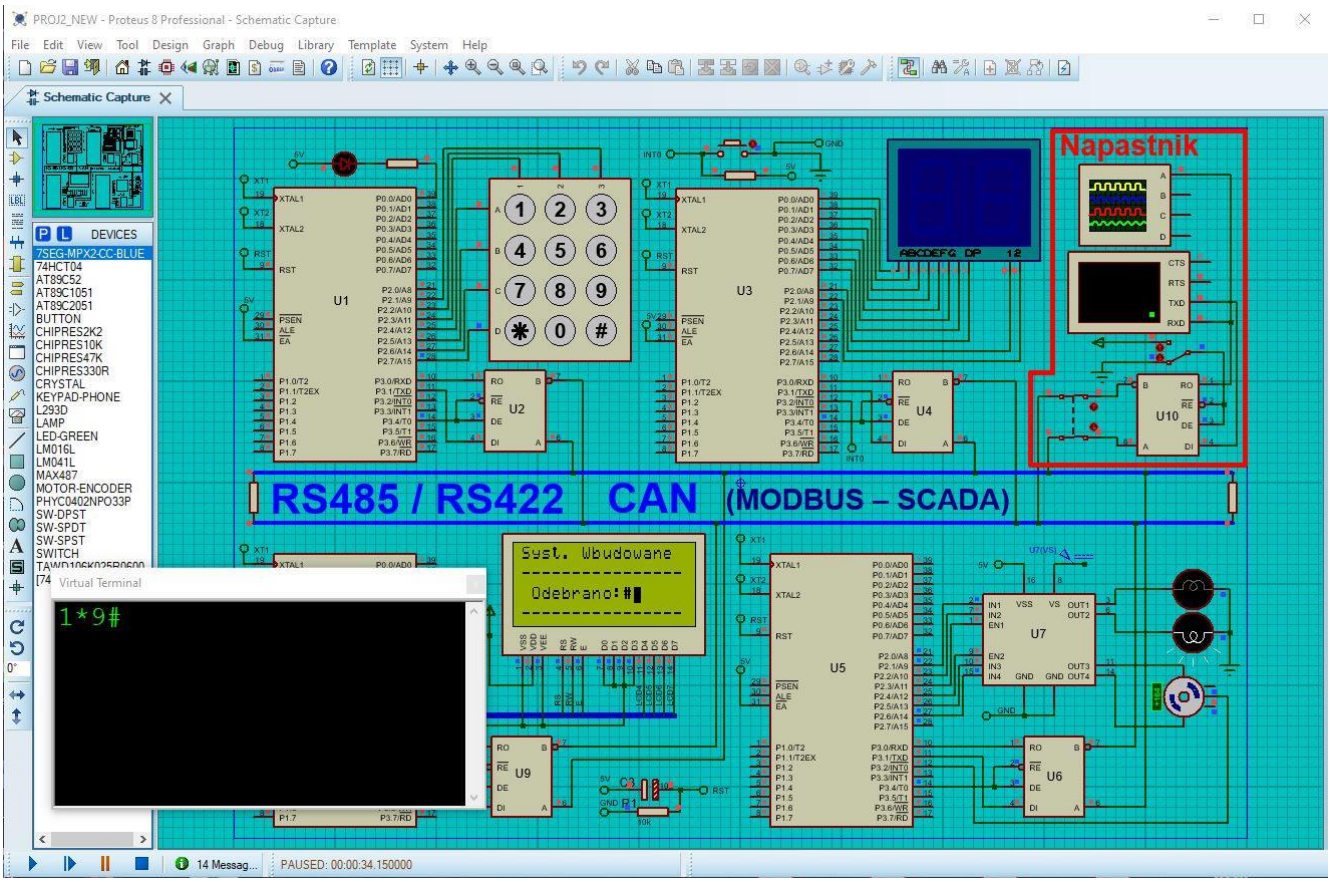
Założenia:

1. Jako wskaźniki poprawności działania programu należy wykorzystać wszystkie elementy zewnętrzne względem danego mikrokontrolera
   (w zadaniach bez komunikacji między mikrokontrolerami nie dotyczy nadajników/odbiorników typu **Max487**
   o symbolach odpowiednio U2, U4, U6 i U9).
2. Udowadnianie działania programu przez wystawianie wartości liczbowych na porty mikrokontrolera jest niedozwolone.
3. Wszystkie elementy wejściowe i wyjściowe danego mikrokontrolera muszą zostać obsłużone
   - w zadaniach bez komunikacji między mikrokontrolerami nie dotyczy nadajników/odbiorników typu **Max487** o symbolach odpowiednio U2, U4, U6 i U9,
4. Program musi kompilować się bez błędów i bez ostrzeżeń (nie dotyczy ważności licencji, nie dotyczy warningów przy linkowaniu kodu z wykładów dla wyświetlacza LCD, podłączonego do U8).
5. Program nie może generować zwarć w układzie. Wystąpienie choć jednego zwarcia oznacza ocenę niedostateczną za projekt i niezaliczenie laboratorium, **bez względu na wcześniejsze oceny**.
6. Schemat układu przekazany poniżej do zadania **nie może być modyfikowany**. Modyfikacja schematu oznacza ocenę niedostateczną. Ograniczenie **nie dotyczy == dołączenia w dowolne miejsce oscyloskopu dla zaprezentowania przebiegów == oraz wyłączenia z symulacji nieużywanych układów** np. gdy Student pisze program tylko na U1, może wyłączyć z symulacji U3, U5, U8. Trzeba wtedy opisać i uzasadnić w sprawozdaniu podjęte działania. Proszę jednak brać pod uwagę, że przy próbie komunikacji za pomocą Max487 wyłączenie mikrokontrolera macierzystego z symulacji zablokuje możliwości transmisji przez linię różnicową dla pozostałych układów. Dla zachowania możliwości komunikacji zalecam napisanie na układy nieużywane programu "zaślepki" z właściwym ustawieniem pinu P3_4.

Przykład projektu układu: Folder C:\Zadanie\PROJ2.pdsprj w Windows XP w maszynie wirtualnej do przedmiotu



Aktualny schemat do projektu dla semestru letniego 2022:



Oceny:

(3) jeden działający program, kompilacja bez błędów i ostrzeżeń. Jeśli wykonywane zadanie wykorzystuje klawiaturę przy U1, reakcja na każdy klawisz **musi być unikalna** - np. nie przyjmę zadania typu "na naciśnięcie dowolnego klawisza dioda ma mignąć jeden raz"

(3+) dwa działające programy, każdy na innym mikrokontrolerze, kompilacja bez błędów i ostrzeżeń, w zadaniach na dst+, db i db+można oczywiście wykorzystywać transmisję szeregową dla urozmaicenia zadania, ale bez podniesienia oceny za fakt komunikacji szeregowej;

(4) trzy działające programy, każdy na innym mikrokontrolerze, kompilacja bez błędów i ostrzeżeń.
(4+) cztery działające programy, każdy na innym mikrokontrolerze, kompilacja bez błędów i ostrzeżeń.
(5) cztery działające programy, każdy na innym mikrokontrolerze, kompilacja bez błędów i ostrzeżeń, ponadto co najmniej dwa układy mikrokontrolerów ze sobą "rozmawiają" przez magistralę różnicową za pomocą Max487.

Uwaga - podniesienie oceny za komunikację między mikrokontrolerami dotyczy **tylko sytuacji czterech poprawnie działających programów** - jeżeli studentka/ student napisze np. dwa programy na dwa mikrokontrolery, które się będą komunikować, to ocena po spełnieniu wszystkich warunków wyniesie 3+.

Każdy Student wymyśla własne, indywidualne i unikalne zadanie. Zadanie to formułuje w sprawozdaniu jawnie w punkcie "SFORMUŁOWANIE PROBLEMU na układ Ux", gdzie Ux to symbol układu mikrokontrolera - odpowiednio U1, U3, U5, U8.

W sprawozdaniu w osobnym punkcie, zatytułowanym "OPIS POMYSŁU NA REALIZACJĘ problemu na układ Ux", gdzie Ux to symbol układu mikrokontrolera - odpowiednio U1, U3, U5, U8 koniecznie musi wystąpić opis pomysłu na realizację zadania - bez niego nawet poprawnie działający program na daną ocenę nie zostanie zaliczony.

Każdy Student samodzielnie rozwiązuje postawione przez siebie zadania i sprawozdanie przysyła na podany adres mailowy.

Nie trzeba wcześniej uzgadniać ze mną swoich tematów.

Student ma prawo do wykorzystania kodów programów, prezentowanych na wykładach i laboratoriach, ale żadne zadanie nie może się do ich wykorzystania ograniczać.

Uwaga:
W przypadku znacznego podobieństwa przysłanego zadania do innych otrzymanych opisów problemów i uzyskanych rozwiązań wykładowca ma prawo zażądać ponownej realizacji zadania, o czym poinformuje studenta przez e-mail.
W przypadku rażącej nieuczciwości nauczyciel podejmuje decyzje o ocenie niedostatecznej dla wszystkich osób oszukujących bez możliwości "dosłania nowych rozwiązań na żądanie".

B. Zadanie na układ U1

Sformułowanie problemu

W ramach zadania należy zaprogramować mikrokontroler tak, aby był on w stanie niezależnie przechwycić każdy klawisz z klawiatury, a następnie pozyskać kod ASCII znaku z danego klawisza. Obliczoną wartość należy wyświetlić przy pomocy podpiętej diody w postaci binarnej w taki sposób, że:

- Wszystkie sygnały (zera i jedynki) są wyświetlane w równych odstępach czasu.
- Każdy sygnał reprezentowany jest jako zapalenie się diody.
- Istnieje przerwa między każdym sygnałem, każde zapalenie diody oznacza jedno zero lub jedną jedynkę.
- Zero binarne reprezentowane jest czterokrotnie krótszym czasem świecenia diody.

Odczytany symbol należy wysłać na magistralę.

Opis mojego rozwiązania

Na początku podaję w nieskończonej pętli wędrujące zero po wierszach. W momencie wciśnięcia dowolnego klawisza wywołuje się funkcja „print_char()”. Funkcja ta określa na podstawie wartości na porcie P2 który klawisz został wciśnięty i pobiera jego wartość ASCII z tablicy „Koder[]”. Następnie kolejno wyświetla każdy bit zaczynając od najstarszego. Zero binarne Wyświetlane jest przez czas określony funkcją „smallDelay()”, a jedynka binarna funkcją „Delay()”. Każde wyświetlenie kończy się dodatkowym wywołaniem przeciwnej funkcji (na przykład jeśli wyświetlaliśmy jedynkę z funkcją Delay(), po zgaszeniu diodwy wywołujemy funkcję smallDelay()) w celu zapewnienia równego czasu działania programu dla wyświetlenia zarówno zera, jak i jedynki.

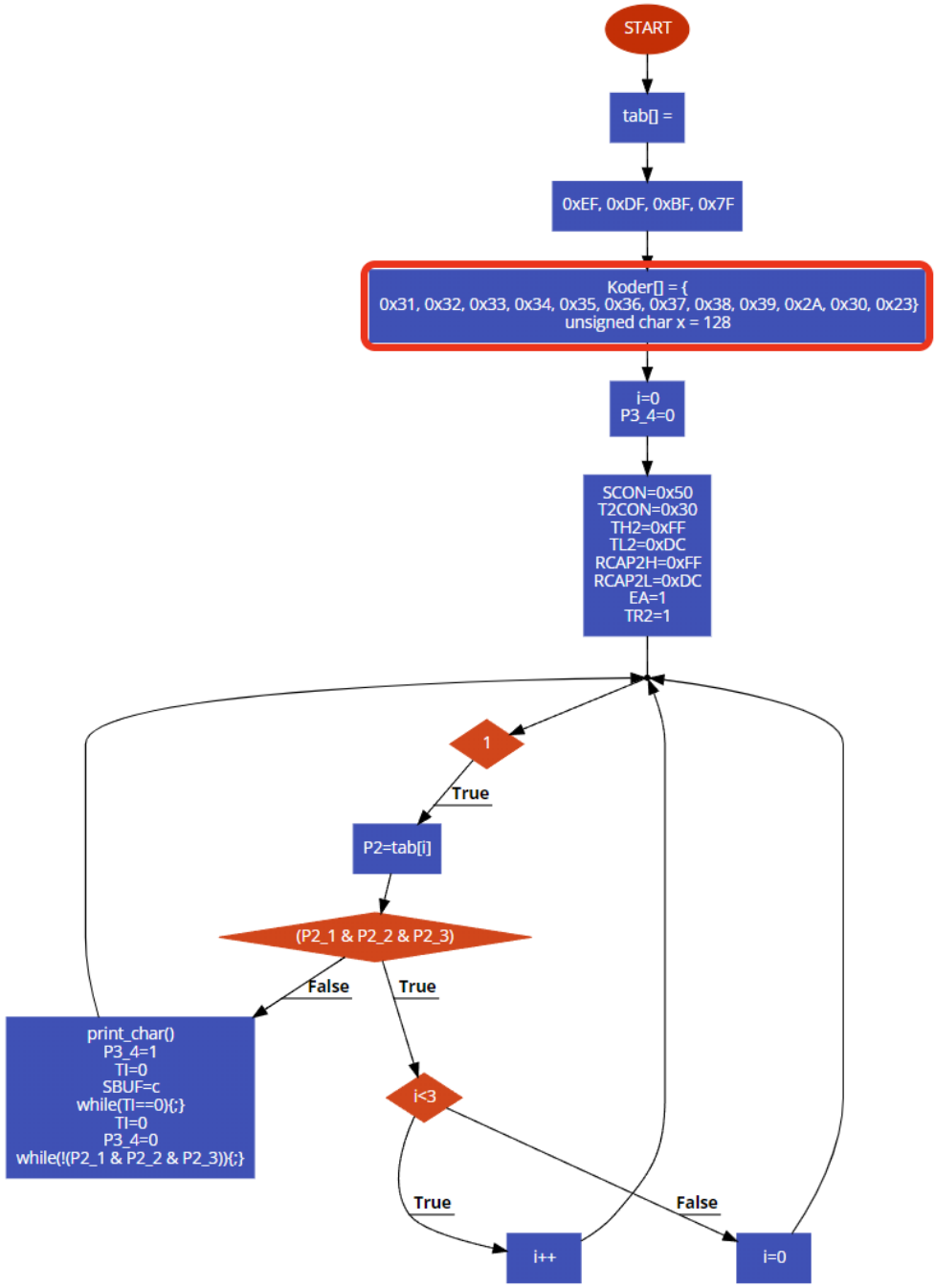
Określanie klawisza odbywa się przez określenie jego numeru wiersza (row) oraz kolumnu (col), a następnie na podstawie tych zmiennych określa się indeks, z którego należy pobrać wartość z tablicy Koder[] (wzór: row*3+col).

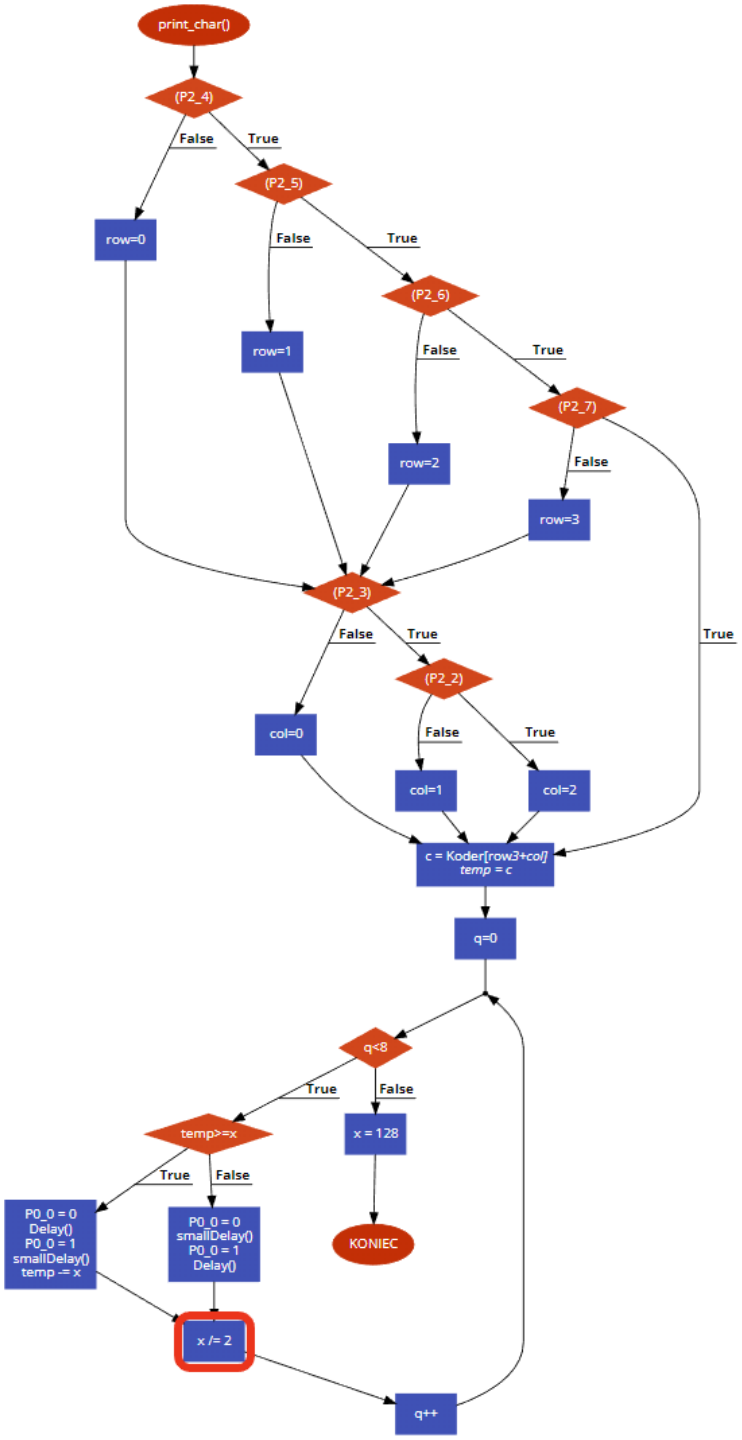
W celu wysłania odczytanego symbolu włączam port szeregowy (TR2=1) oraz konfiguruję go tak, żeby był w stanie wysłać bajt na magistralę. Używam do tego poniższych ustawień:

- Tryb działania (SCON=0x50, czyli 8 bit var Baudrate, T2CON=0x30)
- Prędkość działania portu (TH2, TL2) oraz wartość do przeładowywania (RCAP2H, RCAP2L) – dla obu par rejestrów wartość high wynosi 0xFF, a wartość low wynosi 0xDC
- Włączenie przerwań (EA=1) oraz portu szeregowego (TR2=1)

Przez większość czasu program ustawiony jest na odbiór (P3_4=0), jednak w momencie wysłania wiadomości zmienia się na tryb wysyłania (P3_4=1) i wysyła na bufor (SBUF) odpowiedni bajt.

Schemat blokowy rozwiązania





Listing programu

```
#include <REGX52.H>

unsigned char code tab[] = {0xEF, 0xDF, 0xBF, 0x7F};

unsigned char code Koder[] = {
    0x31, // 1
    0x32, // 2
    0x33, // 3
    0x34, // 4
    0x35, // 5
    0x36, // 6
    0x37, // 7
    0x38, // 8
```

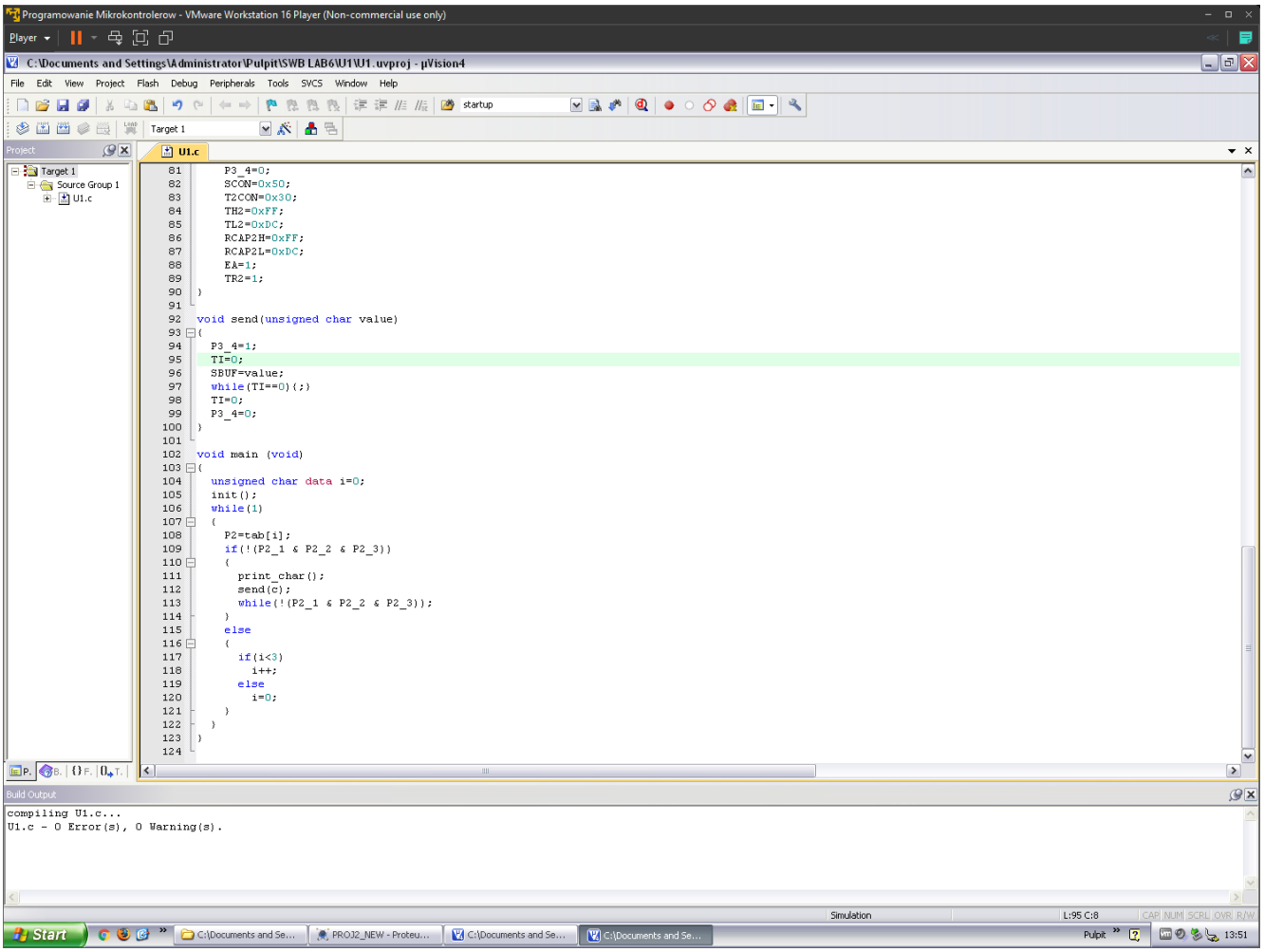
```
    0x39, // 9
    0x2A, // *
    0x30, // 0
    0x23, // #
};

unsigned char c;
unsigned char temp;
unsigned char row;
unsigned char col;
unsigned char x = 128;
unsigned char q;

void Delay(void)
{
    unsigned char i, j;
    for(i=0;i<100;i++)
        for(j=0;j<50;j++) {;}
}

void smallDelay(void)
{
    unsigned char i, j;
    for(i=0;i<100;i++)
        for(j=0;j<10;j++) {;}
}

void print_char(void)
{
    // Pobierz wiersz
        if(!(P2_4)) row=0;
    else if(!(P2_5)) row=1;
    else if(!(P2_6)) row=2;
    else if(!(P2_7)) row=3;

      // Pobierz kolumne
        if(!(P2_3)) col=0;
    else if(!(P2_2)) col=1;
    else if(!(P2_1)) col=2;

    c = Koder[row*3+col];
        temp = c;

    for(q=0; q<8; q++)
    {
        if(temp>=x) // 1 bin
        {
            P0_0 = 0;
            Delay();
            P0_0 = 1;
            smallDelay();
            temp -= x;
        }
```

```
        else // 0 bin
        {
            P0_0 = 0;
            smallDelay();
            P0_0 = 1;
            Delay();
        }
        x /= 2;
    }
        x = 128;
}

void init(void)
{
    P3_4=0;
    SCON=0x50;
    T2CON=0x30;
    TH2=0xFF;
    TL2=0xDC;
    RCAP2H=0xFF;
    RCAP2L=0xDC;
    EA=1;
    TR2=1;
}

void send(unsigned char value)
{
    P3_4=1;
    TI=0;
    SBUF=value;
    while(TI==0){;}
    TI=0;
    P3_4=0;
}

void main (void)
{
    unsigned char data i=0;
    init();
    while(1)
    {
        P2=tab[i];
        if(!(P2_1 & P2_2 & P2_3))
        {
            print_char();
      send(c);
            while(!(P2_1 & P2_2 & P2_3));
        }
        else
        {
            if(i<3)
                i++;
            else
```

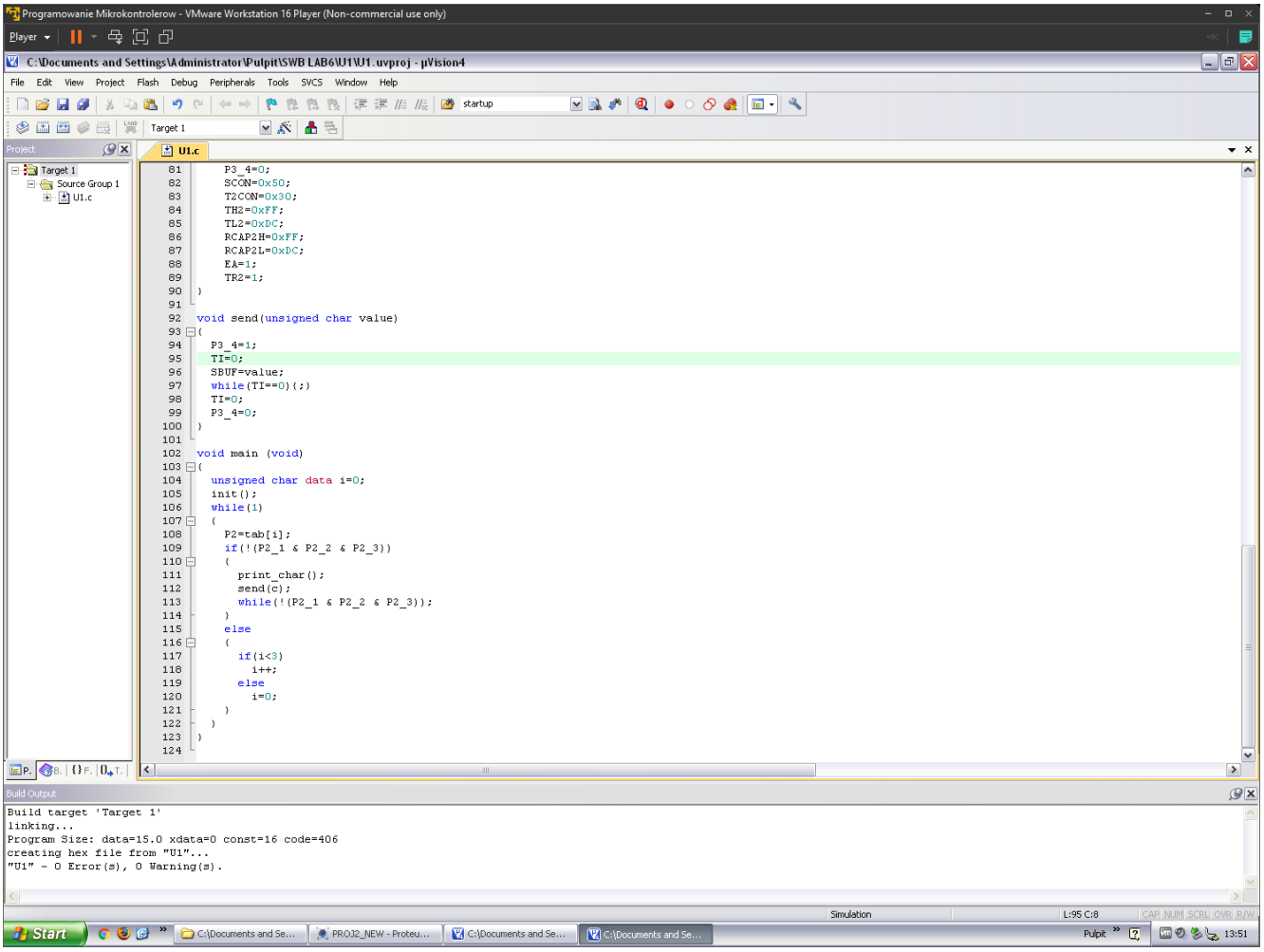
```
                i=0;
            }
        }
}
```

Sprawdzenie poprawności

Kompilowanie



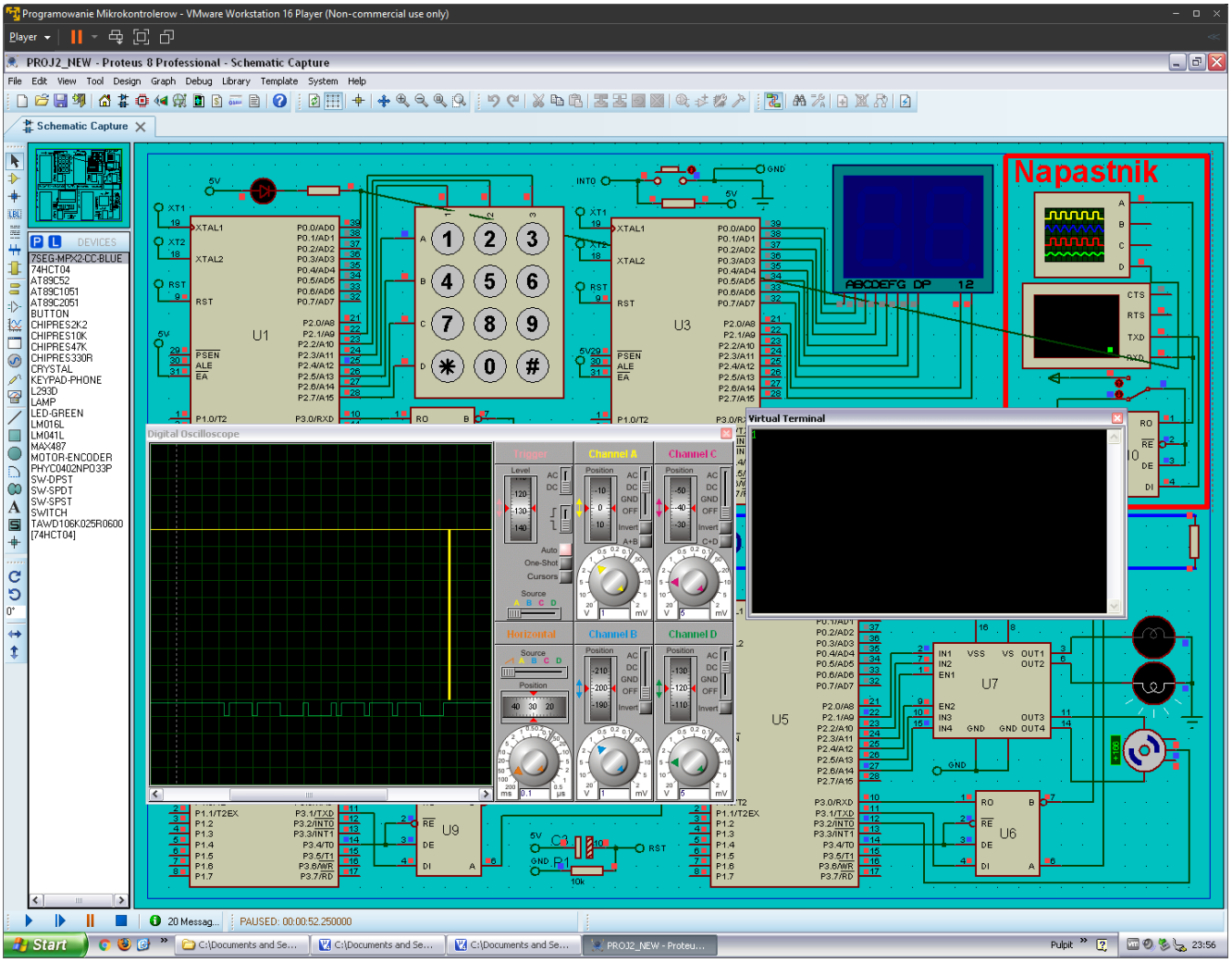
Linkowanie

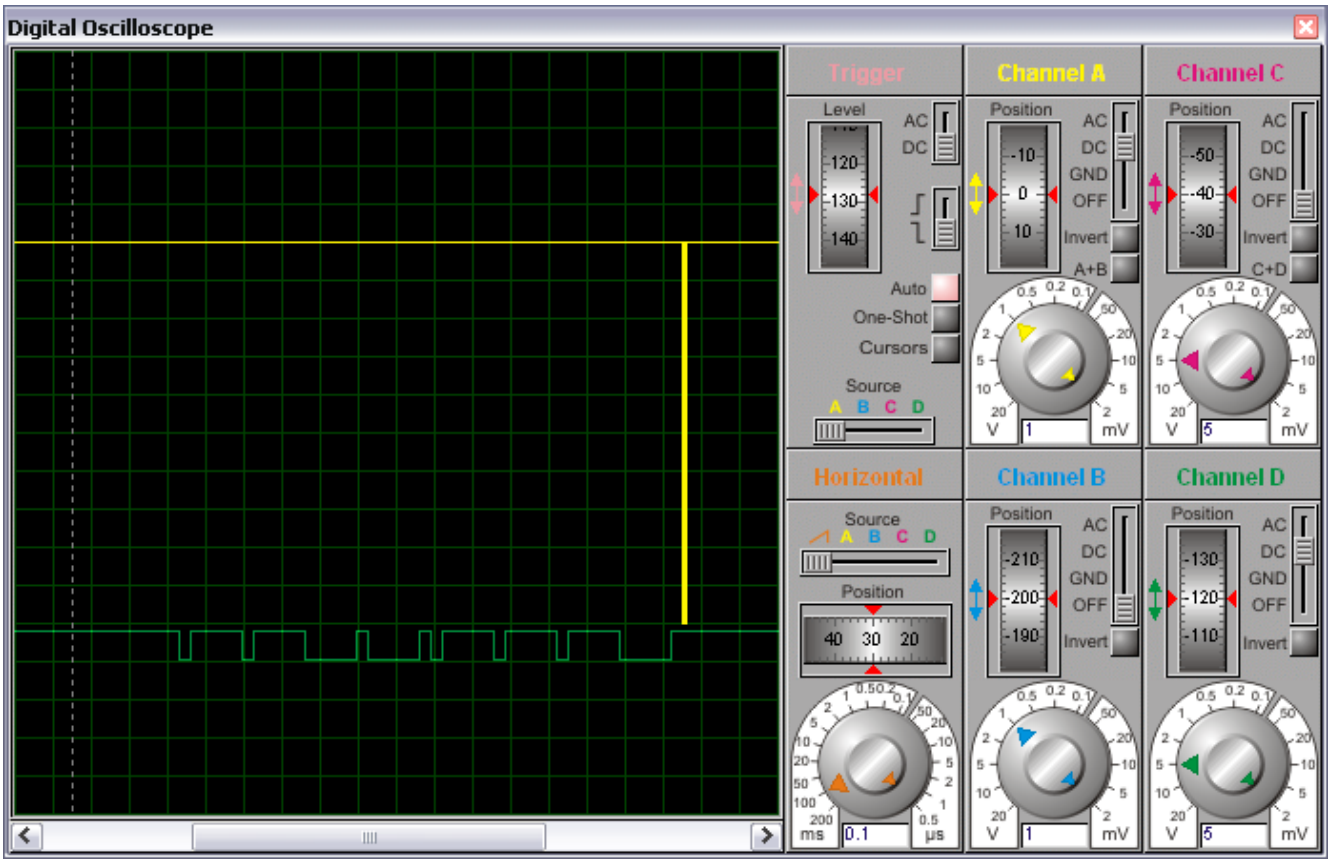


Prezentacja realizacji zadania przez program

Dla zobrazowania wyników podłączam oscyloskop do wyjścia diody.

Wciśnięcie 1:

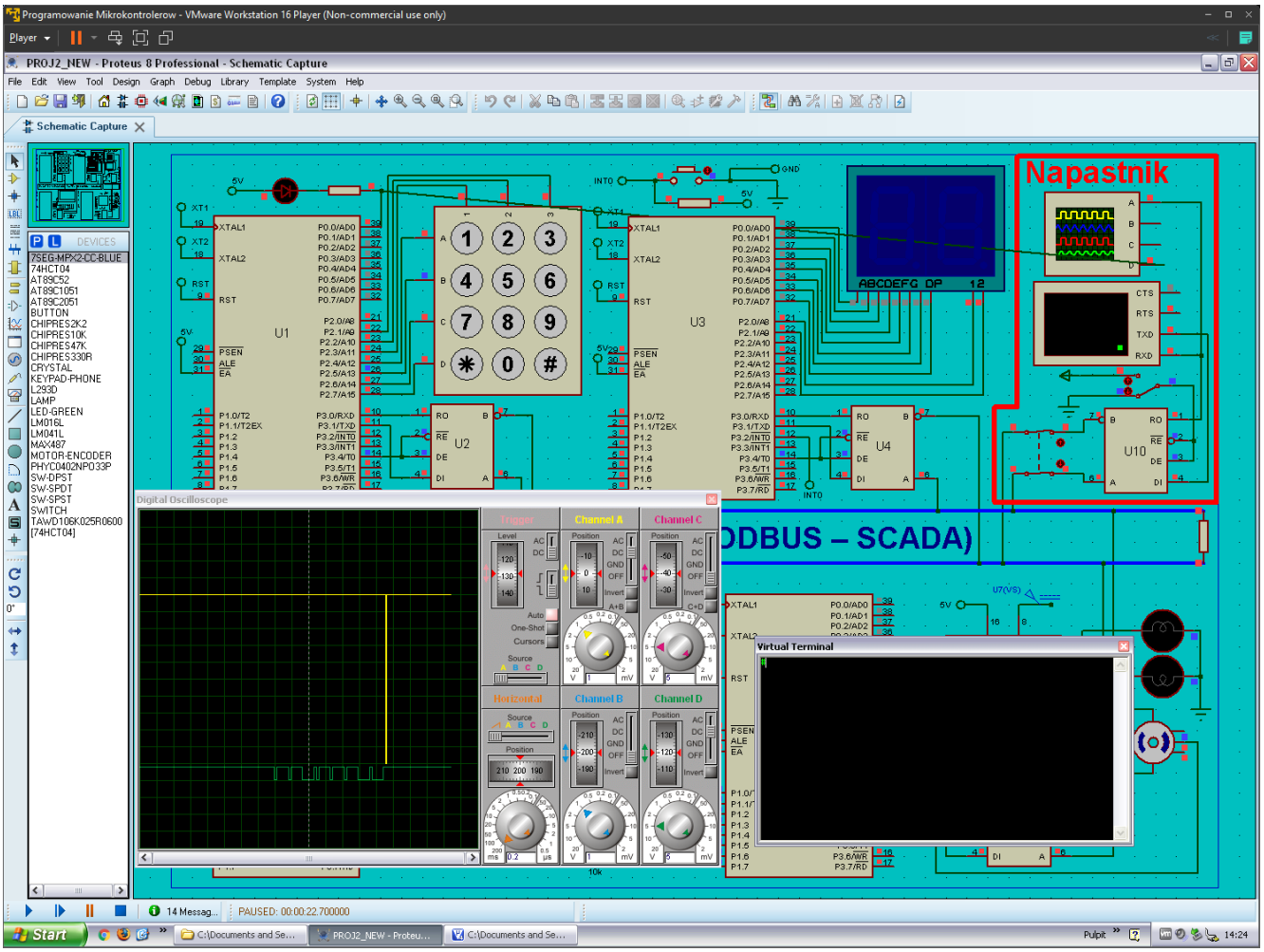




Krótkie zera oznaczają 0 binarne, długie zera oznaczają 1 binarne. Widać tutaj kod $00110001_2 = 32 + 16 + 1 = 49$. Jest to wartość ASCII znaku „1”. Żółty chwilowy spadek pokazuje przesłanie bitu na magistralę.

Widok terminalu:

Wciśnięcie #: kod 00100011_2 = 32 + 2 + 1 = 35, wartość ASCII znaku „#”



A. Zadanie na układ U3

Sformułowanie problemu

Zadaniem układu jest odczytanie bajtu z magistrali i wyświetlenie jej wartości ASCII na wyświetlaczu.

Mikrokontroler ma regularnie dekrementować każdą cyfrę, a kiedy któraś osiągnie wartość 0 następną wartością jest ta pierwotna (na przykład dla liczby 12 kolejne wartości to 01, 10, 02, 11, 00, 12, …).

Wciśnięcie przycisku powoduje przywrócenie pierwotnej wartości na wyświetlaczu.

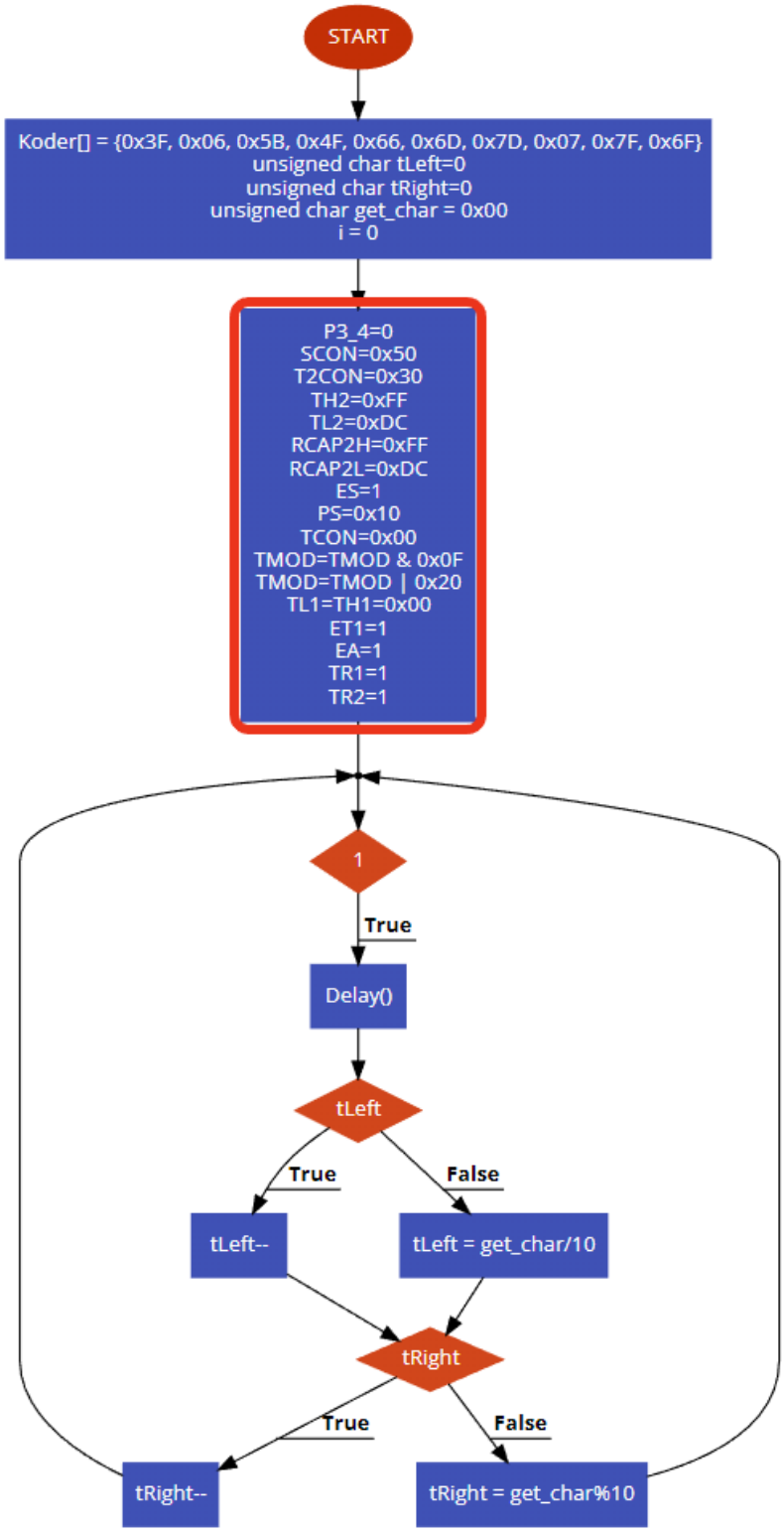
Opis mojego rozwiązania

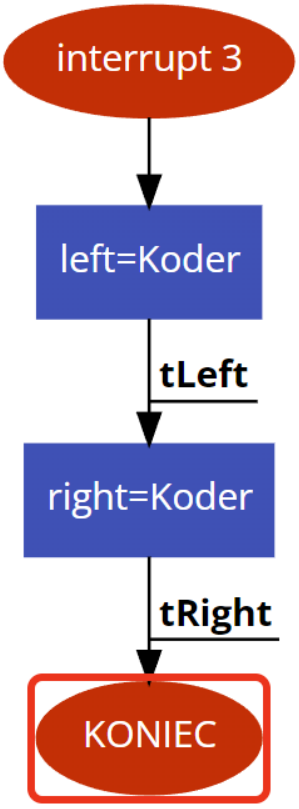
Inicjalizuję układ jako odbiornik sygnału z magistrali. Wykorzystuję do tego funkcję init() zastosowaną w układzie U1 oraz rozszerzoną o możliwość odbioru sygnału (ES=1). Nadaję wyższy priorytet poprzez ustawienie PS na 0x10. Oprócz tego, włączę timer 1 (tryb 8bid auto reload, TCON=0x00), który poprzez przerwania interrupt 3 będzie podawał na wyświetlacz odpowiednie wartości.

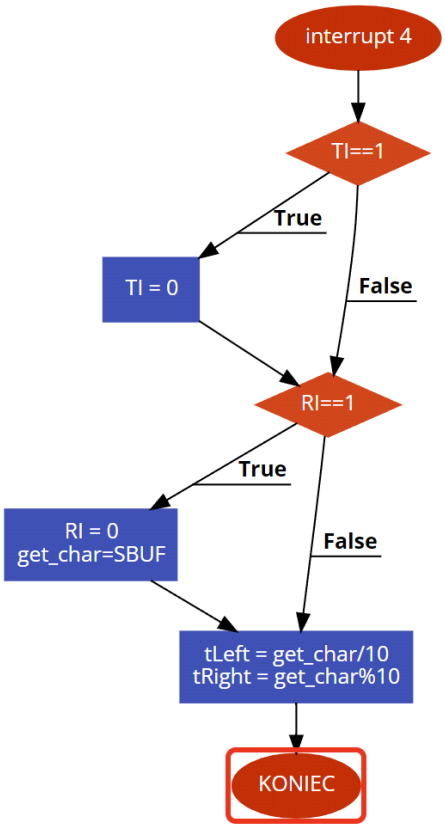
Stworze również funkcje receive() do obsługi przerwań interrupt 4 dla odbierania sygnału z magistrali. Pobiera ona wartość na podstawie SBUF, a następnie deklaruje nowe wartości do wyświetlenia (określa wartość indeksów tLeft oraz tRight, na podstawie których pobierane są odpowiednie wartości z tablicy Koder do podania na wejścia wyświetlacza). Wartość tLeft to liczba dziesiątek (get_char/10), a wartość tRight to liczba jedności (get_char%10).

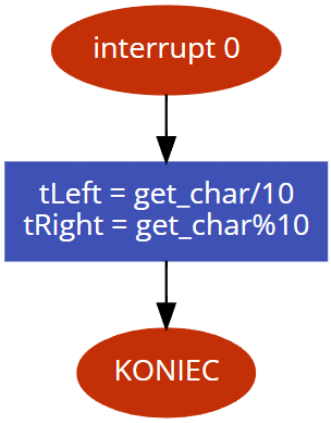
Program w nieskończonej pętli cyklicznie co dany odcinek czasu zmienia wartości indeksów tLeft oraz tRight, dekrementując je do zera, a następnie resetując do danej cyfry na podstawie odczytanego bajtu.

Schemat blokowy rozwiązania









Listing programu

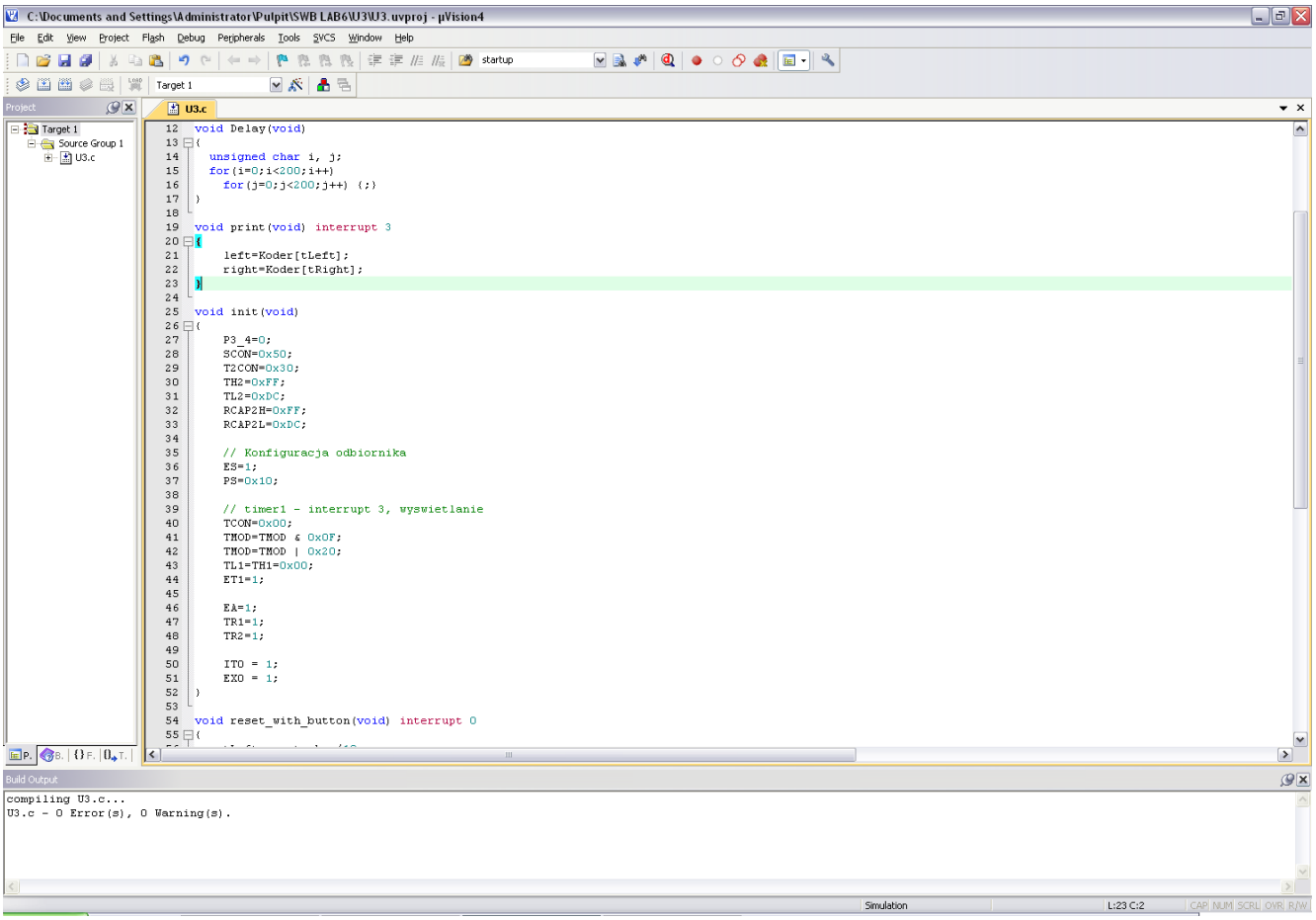
```
#include <REGX52.H>

unsigned char xdata left _at_ 0xFE00;
unsigned char xdata right _at_ 0xFD00;

unsigned char code Koder[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
                              0x6D, 0x7D, 0x07, 0x7F, 0x6F};
unsigned char tLeft=0;
unsigned char tRight=0;
unsigned char get_char = 0x00;

void Delay(void)
{
    unsigned char i, j;
    for(i=0;i<200;i++)
        for(j=0;j<200;j++) {;}
}

void print(void) interrupt 3
{
    left=Koder[tLeft];
    right=Koder[tRight];
}

void init(void)
{
    P3_4=0;
    SCON=0x50;
    T2CON=0x30;
```

```
    TH2=0xFF;
    TL2=0xDC;
    RCAP2H=0xFF;
    RCAP2L=0xDC;

        // Konfiguracja odbiornika
        ES=1;
    PS=0x10;

        // timer1 - interrupt 3, wyswietlanie
      TCON=0x00;
    TMOD=TMOD & 0x0F;
    TMOD=TMOD | 0x20;
    TL1=TH1=0x00;
    ET1=1;

    EA=1;
        TR1=1;
    TR2=1;

        IT0 = 1;
        EX0 = 1;
}

void reset_with_button(void) interrupt 0
{
        tLeft = get_char/10;
        tRight = get_char%10;
}

void receive(void) interrupt 4
{
    if(TI==1)
        TI = 0;
    if(RI==1){
        RI = 0;
        get_char=SBUF;
    }
        tLeft = get_char/10;
        tRight = get_char%10;
}

void main (void){
    unsigned char i = 0;
    init();
    while(1){
            Delay();
                if(tLeft)
            tLeft--;
        else
            tLeft = get_char/10;

        if(tRight)
```

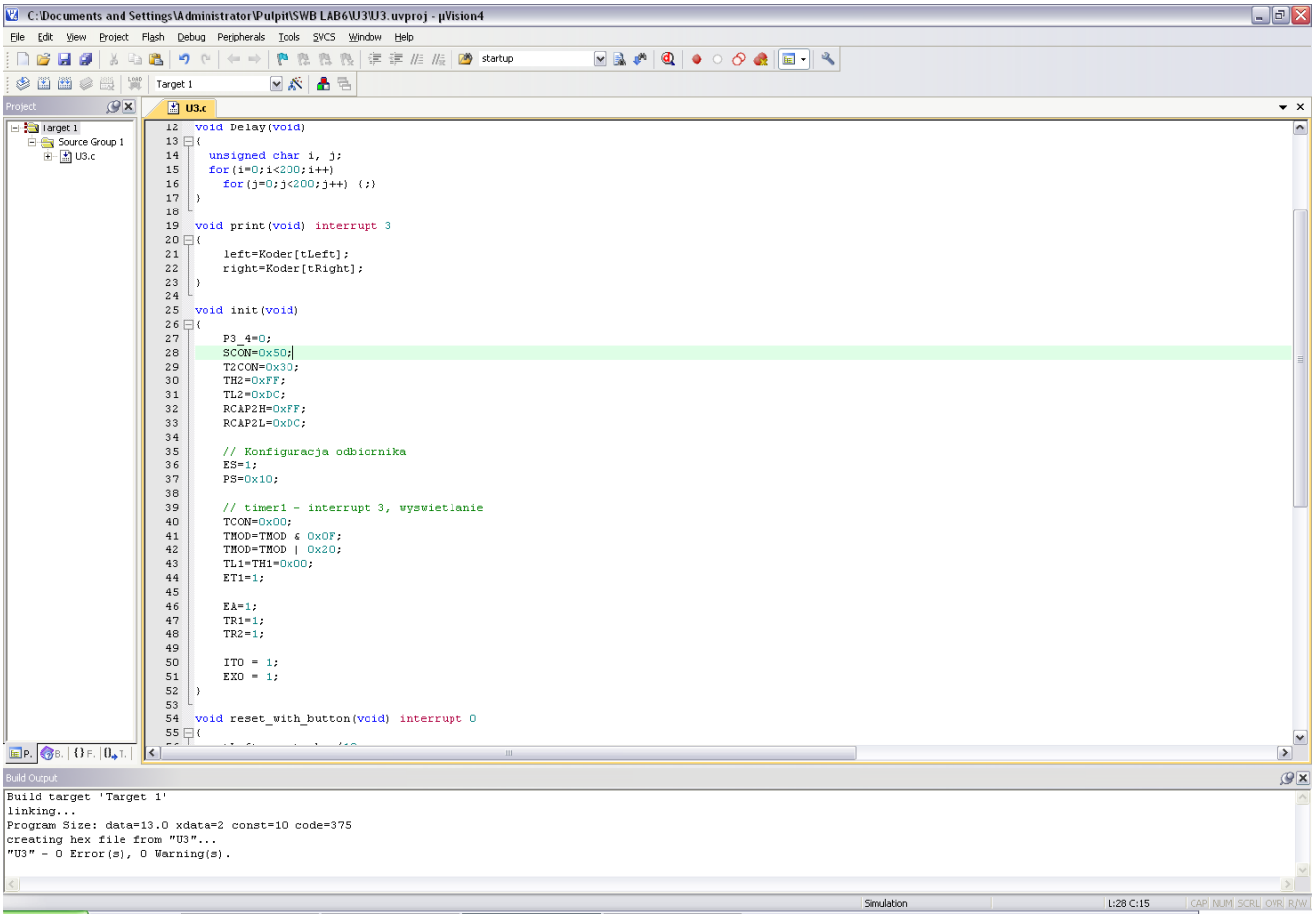
```
            tRight--;
        else
            tRight = get_char%10;
        }
}
```

Sprawdzenie poprawności

Kompilowanie

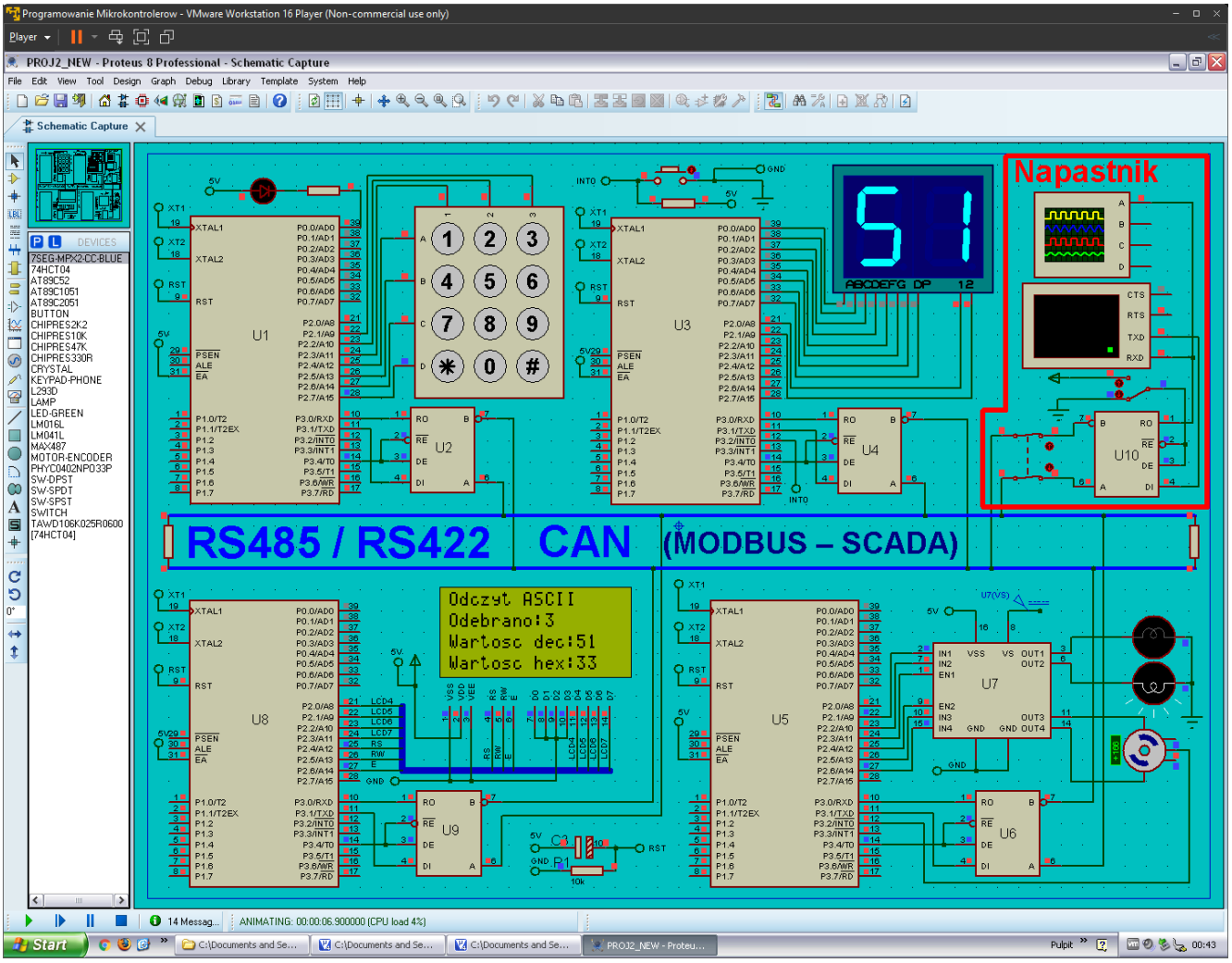


Linkowanie

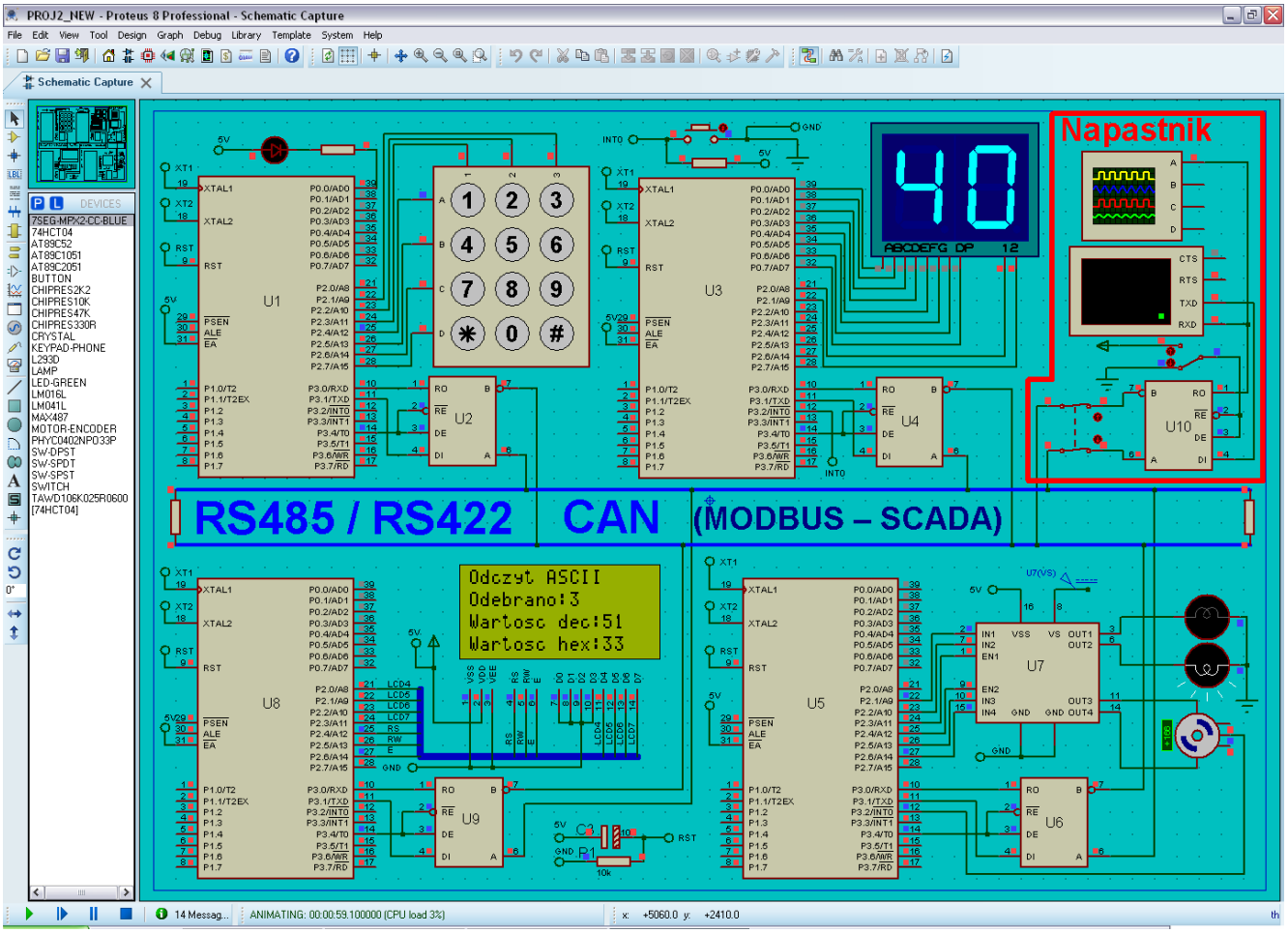


Prezentacja realizacji zadania przez program

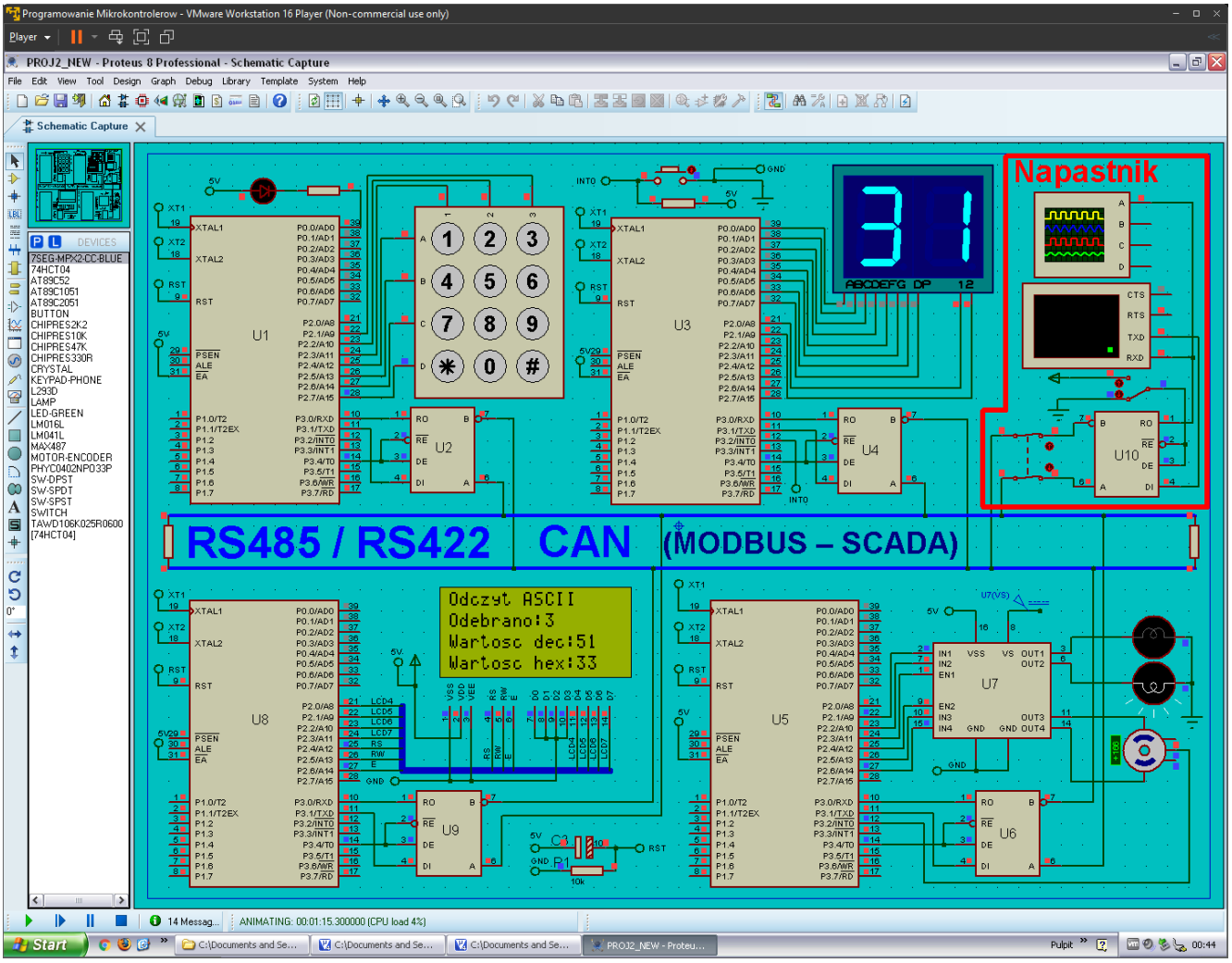
Naciśnięcie 3, odebranie przez wyświetlacz:



Pierwsza zmiana: 5-1 = 4, 1-1 = 0, do wyświetlenia: 40



Druga zmiana: 4-1 = 3, 0 zamienione na 1, do wyświetlenia: 31



A. Zadanie na układ U5

Sformułowanie problemu

Zadaniem układu jest pobranie znaku z magistrali i wykonania na jego podstawie odpowiedniego działania z poniższych:

- Dla cyfry parzystej nadaje obroty silnika w prawo, żarówka dolna się świeci, a górna zmienia stan z każdym obrotem silnika
- Dla cyfry parzystej nadaje obroty silnika w lewo, żarówka górna się świeci, a dolna zmienia stan z każdym obrotem silnika
- Dla gwiazdki (*) przełącza działanie żarówek (każde nieparzyste wciśnięcie ma spowodować, że żarówki w ogóle się nie zapalą, a każde parzyste przywraca działanie zgodnie z poprzednimi punktami)
- Dla krzyżyka (#) zmienia zachowanie układu między parzystościami (z zachowania dla nieparzystej na zachowanie dla parzystej i na odwrót)

Początkowo w układzie silnik oraz diody są wyłączone

Opis mojego rozwiązania

Inicjalizuję układ jako odbiornik sygnału z magistrali. Wykorzystuję do tego funkcję init() zastosowaną w układzie U1 oraz rozszerzoną o możliwość odbioru sygnału (ES=1). Nadaję wyższy priorytet poprzez ustawienie PS na 0x10. Oprócz tego, włączę timer 1 (tryb 8bid auto reload, TCON=0x00), który poprzez przerwania interrupt 3 będzie zmieniał sygnał na wejście en2 (będzie to fala prostokątna o niewielkim współczynniku wypełnienia).

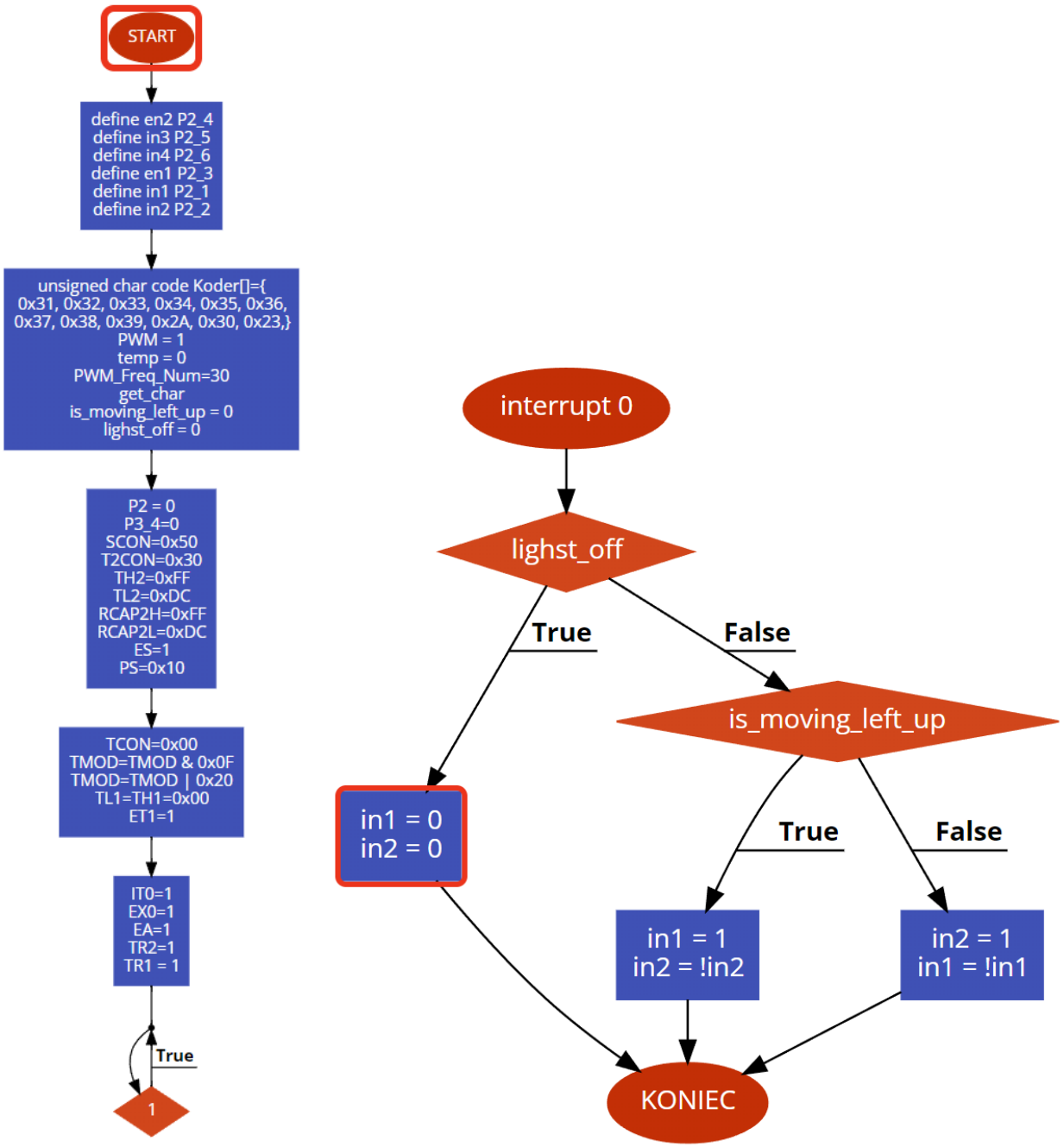
Stworze również funkcje receive() do obsługi przerwań interrupt 4 dla odbierania sygnału z magistrali. Pobiera ona wartość na podstawie SBUF, a następnie z pomocą flag is_moving_left_up oraz lighst_off wykonuje odpowiednią operację:

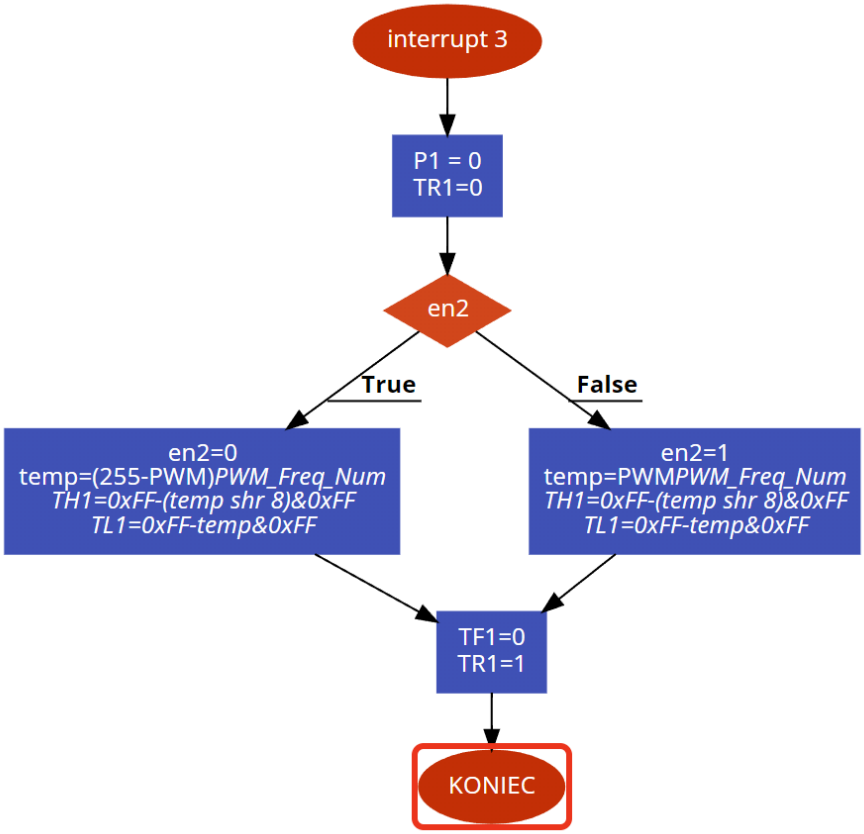
- Dla cyfry parzystej nadaje obroty silnika w prawo (in1=0, in2=1), żarówka dolna się świeci (in4=1), a górna zmienia stan z każdym obrotem silnika (początkowo in3=0). Flaga is_moving_left_up ustawia się na wartość 1.
- Dla cyfry parzystej nadaje obroty silnika w lewo (in1=2, in2=0), żarówka górna się świeci (in3=1), a dolna zmienia stan z każdym obrotem silnika (początkowo in4=0). Flaga is_moving_left_up ustawia się na wartość 0.
- Dla gwiazdki (*) przełącza działanie żarówek. Odwraca stan flagi lighst_off.
- Dla krzyżyka (#) zmienia zachowanie układu między parzystościami. Jeśli żarówki się świecą (lighst_off=0) zmienia ich stany. Neguje również wejścia dla silnika (in3=!in3, in4=!in4) oraz zmienia stan flagi is_moving_left_up.

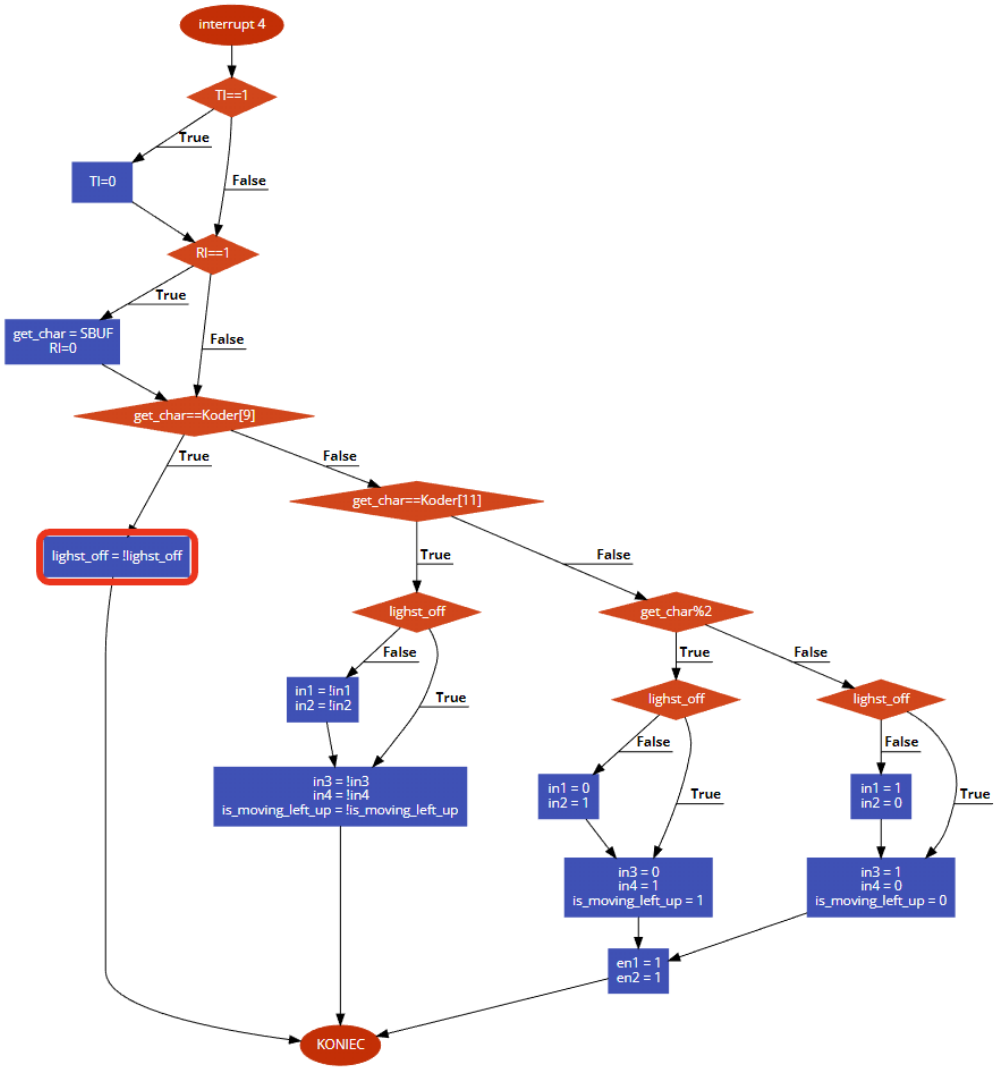
Przełączanie żarówek w momencie, kiedy mają zmieniać stan co obrót silnika, jest obsługiwane przez interrupt 0 i odbywa się w następujący sposób:

- Jeśli żarówki mają być wyłączone (lighst_off=1) to wyłącz żarówki.
- Jeśli silnik porusza się w lewo (is_moving_left_up=1) to zapal górną (in1=1) i zmień stan dolnej (in2=!in2)
- Jeśli silnik porusza się w prawo (is_moving_left_up=0) to zapal dolną (in2=1) i zmień stan górnej (in1=!in1)

Schemat blokowy rozwiązania




interrupt 3
P1 = 0
TR1=0
en2
True
False
en2=0
temp=(255-PWM)PWM_Freq_Num
TH1=0xFF-(temp shr 8)&0xFF
TL1=0xFF-temp&0xFF
en2=1
temp=PWMPWM_Freq_Num
TH1=0xFF-(temp shr 8)&0xFF
TL1=0xFF-temp&0xFF
TF1=0
TR1=1
KONIEC


interrupt 4
TI==1
True
False
TI=0
RI==1
True
False
get_char = SBUF
RI=0
get_char==Koder[9]
True
False
get_char==Koder[11]
lighst_off = !lighst_off
True
False
lighst_off
get_char%2
False
True
False
in1 = !in1
in2 = !in2
True
lighst_off
lighst_off
False
True
False
True
in1 = 0
in2 = 1
in1 = 1
in2 = 0
in3 = !in3
in4 = !in4
is_moving_left_up = !is_moving_left_up
in3 = 0
in4 = 1
is_moving_left_up = 1
in3 = 1
in4 = 0
is_moving_left_up = 0
en1 = 1
en2 = 1
KONIEC

Listing programu

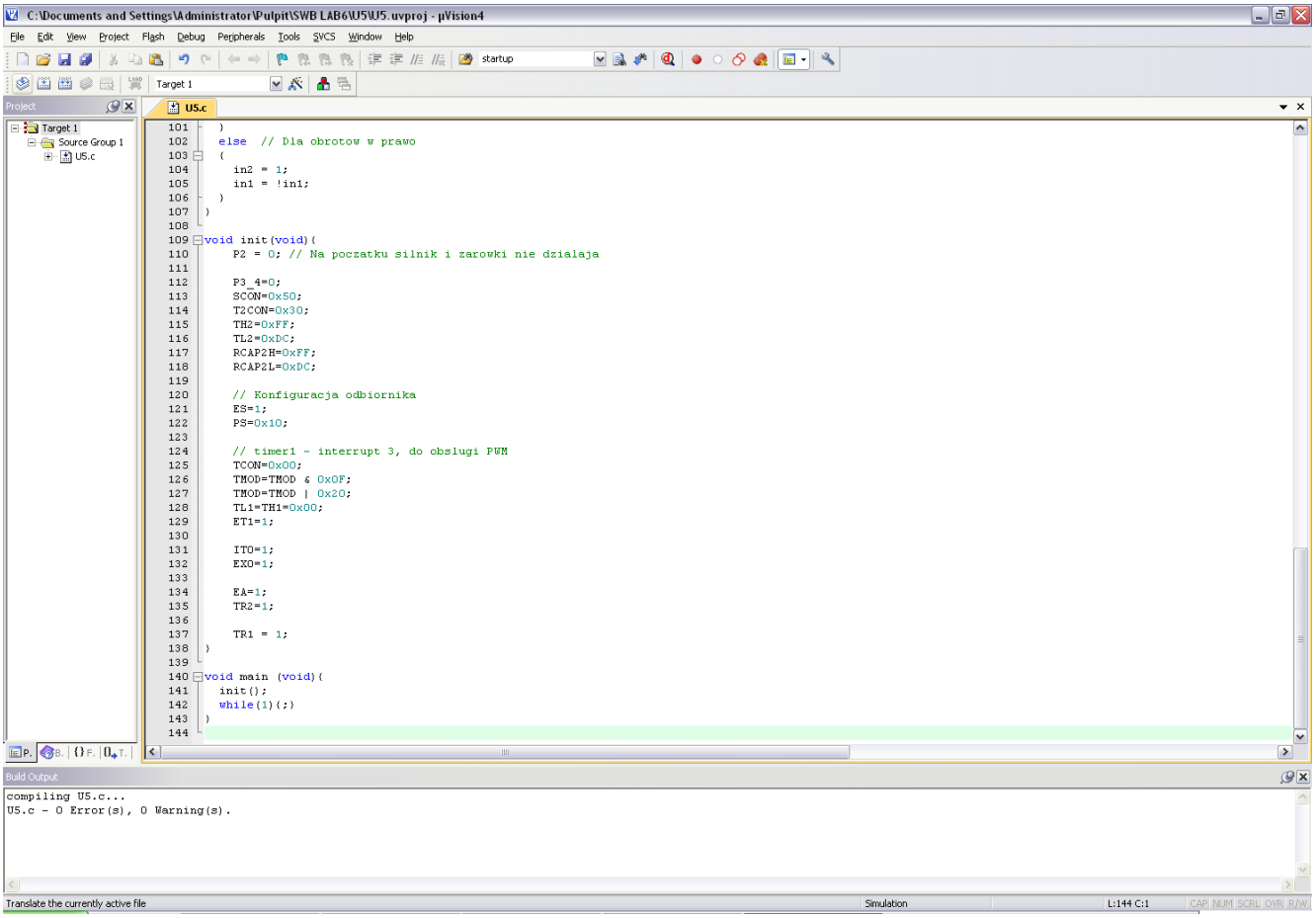
```
#include <REGX52.H>

#define en2 P2_4
#define in3 P2_5
#define in4 P2_6
#define en1 P2_3
#define in1 P2_1
#define in2 P2_2

unsigned char code Koder[] = {
    0x31, // 1
  0x32, // 2
  0x33, // 3
  0x34, // 4
    0x35, // 5
  0x36, // 6
  0x37, // 7
  0x38, // 8
    0x39, // 9
  0x2A, // *
  0x30, // 0
  0x23, // #
};

unsigned char PWM = 1; //255*0.5;     // wartosc od 0 (0% duty cycle) do 255 (100%
duty cycle)
unsigned int temp = 0;    // zmienna robocza w procedurze obslugi przerwania Timer0
#define PWM_Freq_Num 30  //  1 = najwyzsza czestotliwosc gdy PWM_Freq_Num, zakres 1 -
255

unsigned char get_char;
bit is_moving_left_up = 0;
bit lighst_off = 0;

void recive(void) interrupt 4
{
    // Odbior znaku
        if(TI==1){TI=0;}
    if(RI==1)
    {
    get_char = SBUF;
    RI=0;
    }

        if(get_char==Koder[9]) // * - wlacz/wylacz zarowki
        {
            lighst_off = !lighst_off;
        }
        else if(get_char==Koder[11]) // # - zmiana kierunku obiotow
        {
            if(!lighst_off)
                {in1 = !in1; in2 = !in2;}
```

```
            in3 = !in3; in4 = !in4;
            is_moving_left_up = !is_moving_left_up;
        }
        else if (get_char%2) // nieparzysta - ruch w lewo w gore
        {
            if(!lighst_off)
                {in1 = 0; in2 = 1;}
            in3 = 0; in4 = 1;
            is_moving_left_up = 1;

            en1 = 1; en2 = 1;
        }
        else // parzysta - ruch w prawo w dol
        {
            if(!lighst_off)
                {in1 = 1; in2 = 0;}
            in3 = 1; in4 = 0;
            is_moving_left_up = 0;

            en1 = 1; en2 = 1;
        }
}

void handle_PWM(void) interrupt 3   {
    P1 = 0;
    TR1=0;
    if(en2){
        en2=0;
        temp=(255-PWM)*PWM_Freq_Num;
        TH1=0xFF-(temp>>8)&0xFF;
        TL1=0xFF-temp&0xFF;
    }else{
        en2=1;
        temp=PWM*PWM_Freq_Num;
        TH1=0xFF-(temp>>8)&0xFF;
        TL1=0xFF-temp&0xFF;
    }
    TF1=0;
    TR1=1;
}


void handle_engine(void) interrupt 0{
    if(lighst_off) // Dla wylaczonych
        {in1 = 0; in2 = 0;}
    else
    if(is_moving_left_up)   // Dla obrotow w lewo
    {
        in1 = 1;
        in2 = !in2;
    }
    else    // Dla obrotow w prawo
    {
```

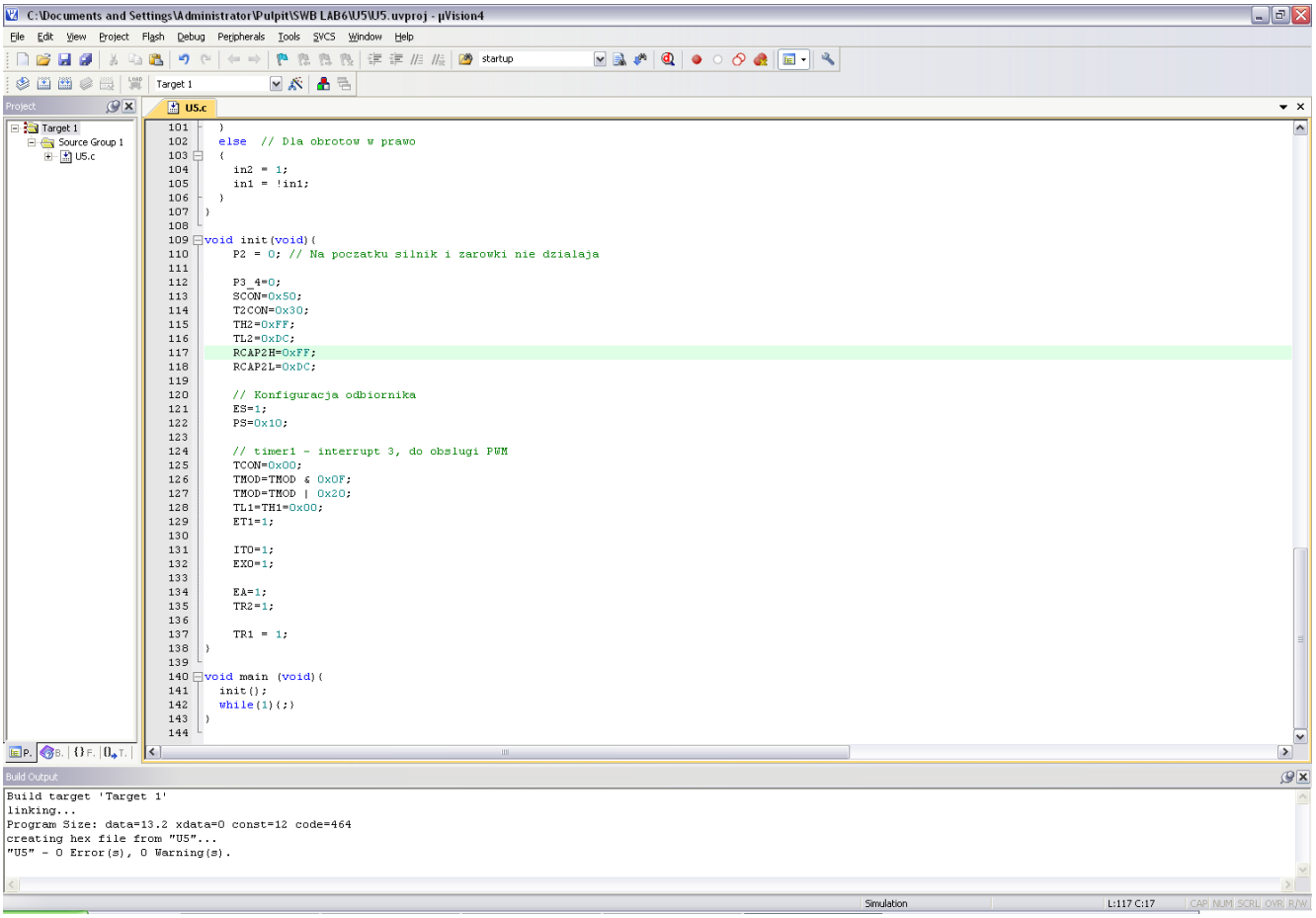
```
        in2 = 1;
        in1 = !in1;
    }
}

void init(void){
        P2 = 0; // Na poczatku silnik i zarowki nie dzialaja

    P3_4=0;
    SCON=0x50;
    T2CON=0x30;
    TH2=0xFF;
    TL2=0xDC;
    RCAP2H=0xFF;
    RCAP2L=0xDC;

        // Konfiguracja odbiornika
        ES=1;
        PS=0x10;

        // timer1 - interrupt 3, do obslugi PWM
        TCON=0x00;
        TMOD=TMOD & 0x0F;
        TMOD=TMOD | 0x20;
        TL1=TH1=0x00;
        ET1=1;

        IT0=1;
        EX0=1;

    EA=1;
    TR2=1;

        TR1 = 1;
}

void main (void){
    init();
    while(1){;}
}
```

Sprawdzenie poprawności

Kompilowanie

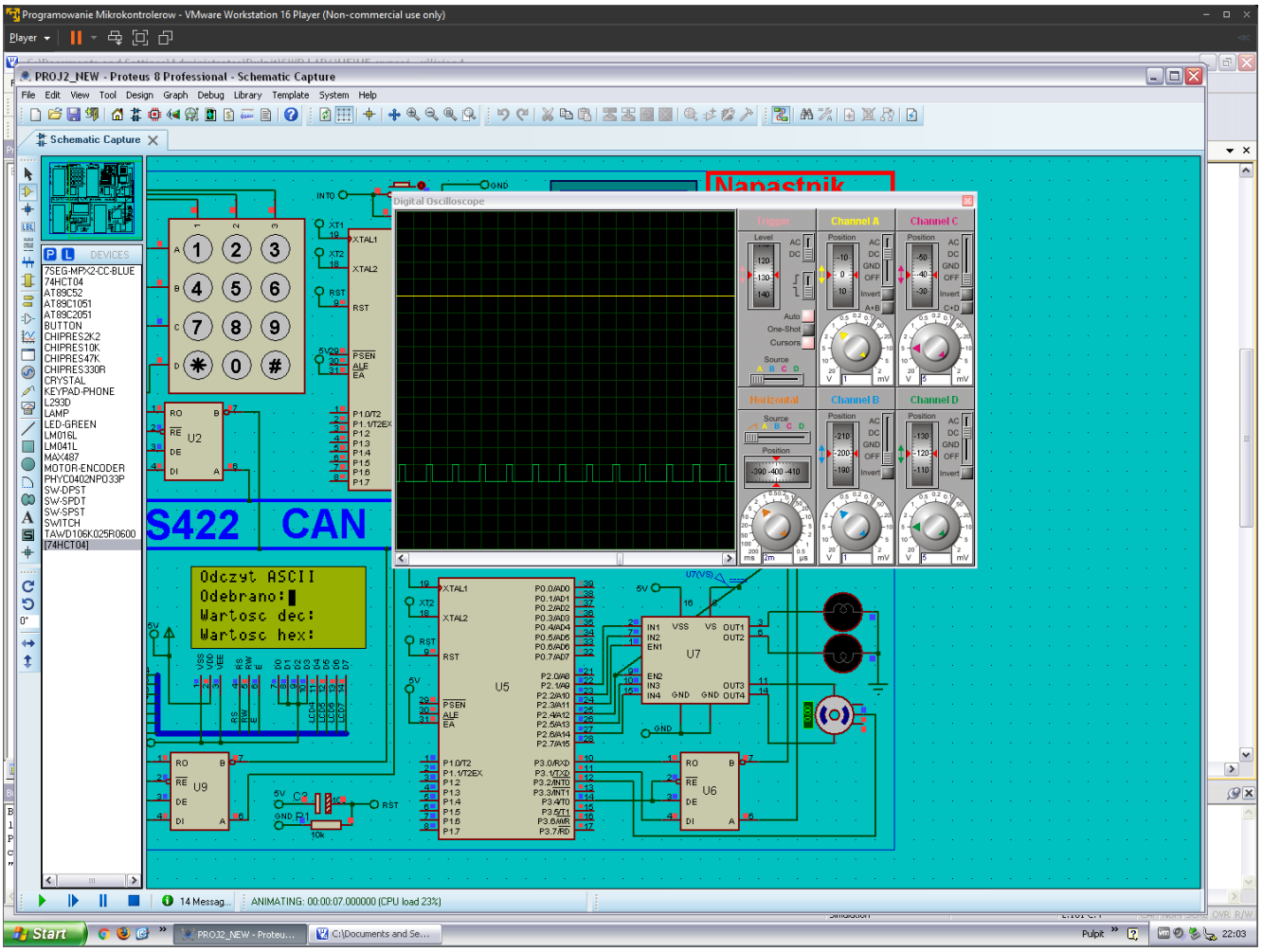

C:\Documents and Settings\Administrator\Pulpit\SWB LAB6\U5\U5.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
startup
Target 1
Project
Target 1
Source Group 1
U5.c
U5.c
101)
102 else // Dla obrotow w prawo
103 {
104 in2 = 1;
105 in1 = !in1;
106 }
107 }
108
109 void init(void){
110 P2 = 0; // Na poczatku silnik i zarowki nie dzialaja
111
112 P3_4=0;
113 SCON=0x50;
114 T2CON=0x30;
115 TH2=0xFF;
116 TL2=0xDC;
117 RCAP2H=0xFF;
118 RCAP2L=0xDC;
119
120 // Konfiguracja odbiornika
121 ES=1;
122 PS=0x10;
123
124 // timer1 - interrupt 3, do obslugi PWM
125 TCON=0x00;
126 TMOD=TMOD & 0x0F;
127 TMOD=TMOD | 0x20;
128 TL1=TH1=0x00;
129 ET1=1;
130
131 IT0=1;
132 EX0=1;
133
134 EA=1;
135 TR2=1;
136
137 TR1 = 1;
138 }
139
140 void main (void){
141 init();
142 while(1){;}
143 }
144
Build Output
compiling U5.c...
U5.c - 0 Error(s), 0 Warning(s).
Translate the currently active file
Simulation
L:144 C:1
CAP NUM SCRL OVR R/W

Linkowanie

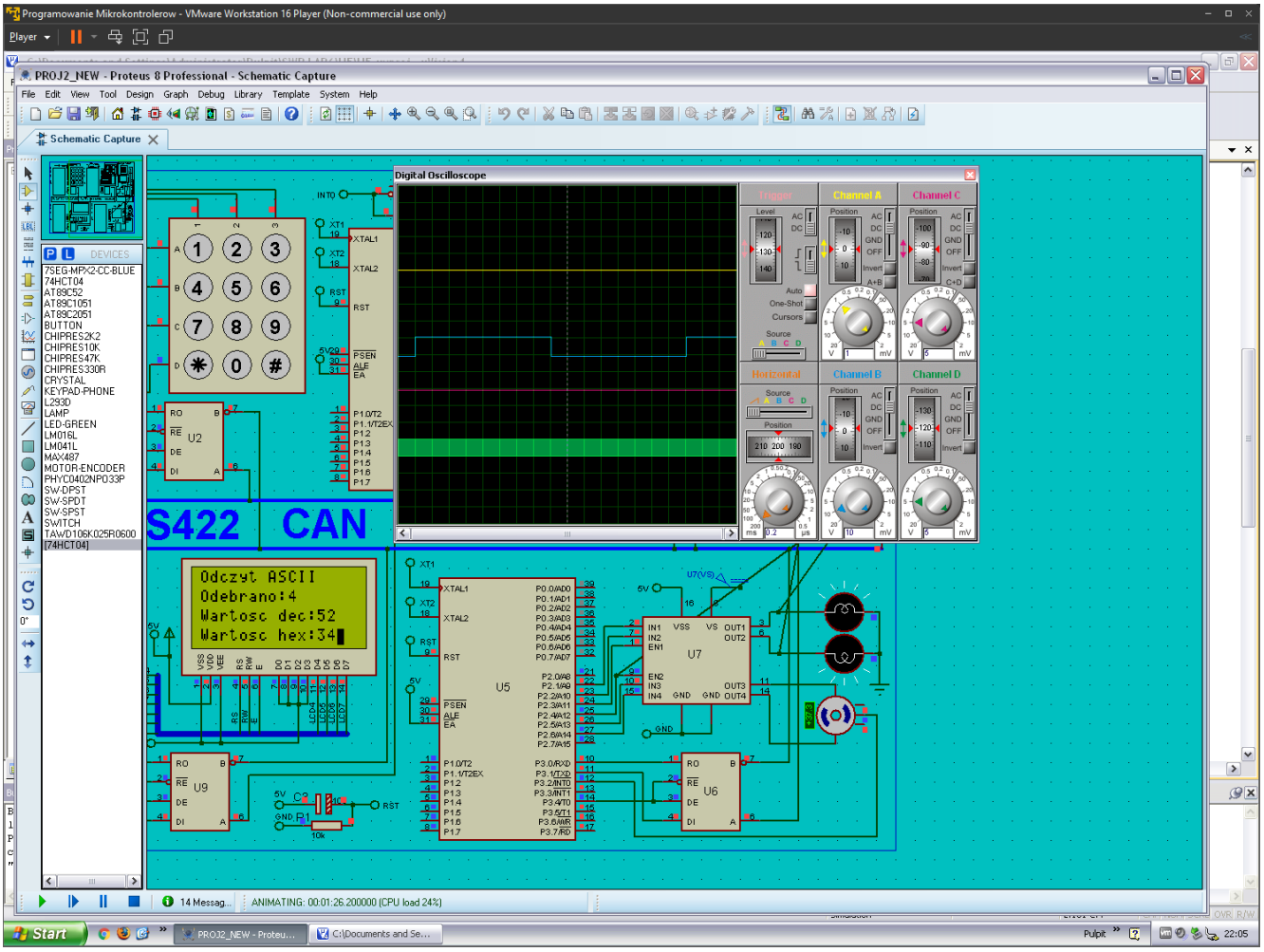

C:\Documents and Settings\Administrator\Pulpit\SWB LAB6\U5\U5.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
startup
Target 1
Project
Target 1
Source Group 1
U5.c
U5.c
101)
102 else // Dla obrotow w prawo
103 {
104 in2 = 1;
105 in1 = !in1;
106 }
107 }
108
109 void init(void){
110 P2 = 0; // Na poczatku silnik i zarowki nie dzialaja
111
112 P3_4=0;
113 SCON=0x50;
114 T2CON=0x30;
115 TH2=0xFF;
116 TL2=0xDC;
117 RCAP2H=0xFF;
118 RCAP2L=0xDC;
119
120 // Konfiguracja odbiornika
121 ES=1;
122 PS=0x10;
123
124 // timer1 - interrupt 3, do obslugi PWM
125 TCON=0x00;
126 TMOD=TMOD & 0x0F;
127 TMOD=TMOD | 0x20;
128 TL1=TH1=0x00;
129 ET1=1;
130
131 IT0=1;
132 EX0=1;
133
134 EA=1;
135 TR2=1;
136
137 TR1 = 1;
138 }
139
140 void main (void){
141 init();
142 while(1){;}
143 }
144
Build Output
Build target 'Target 1'
linking...
Program Size: data=13.2 xdata=0 const=12 code=464
creating hex file from "U5"...
"U5" - 0 Error(s), 0 Warning(s).
Simulation
L:117 C:17
CAP NUM SCRL OVR R/W

Prezentacja realizacji zadania przez program

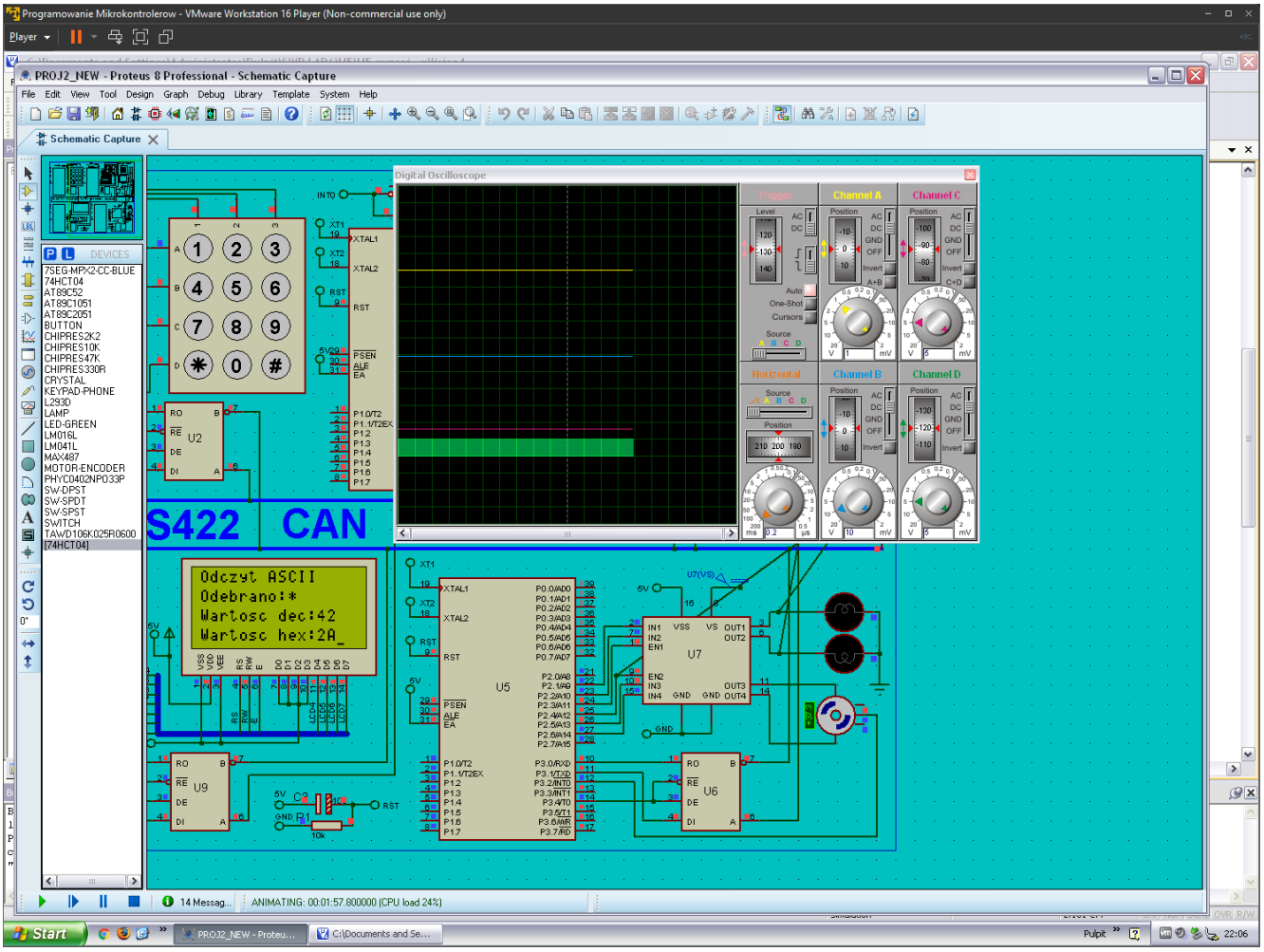
Stan początkowy, fala prostokątna, brak obrotów, brak świecenia



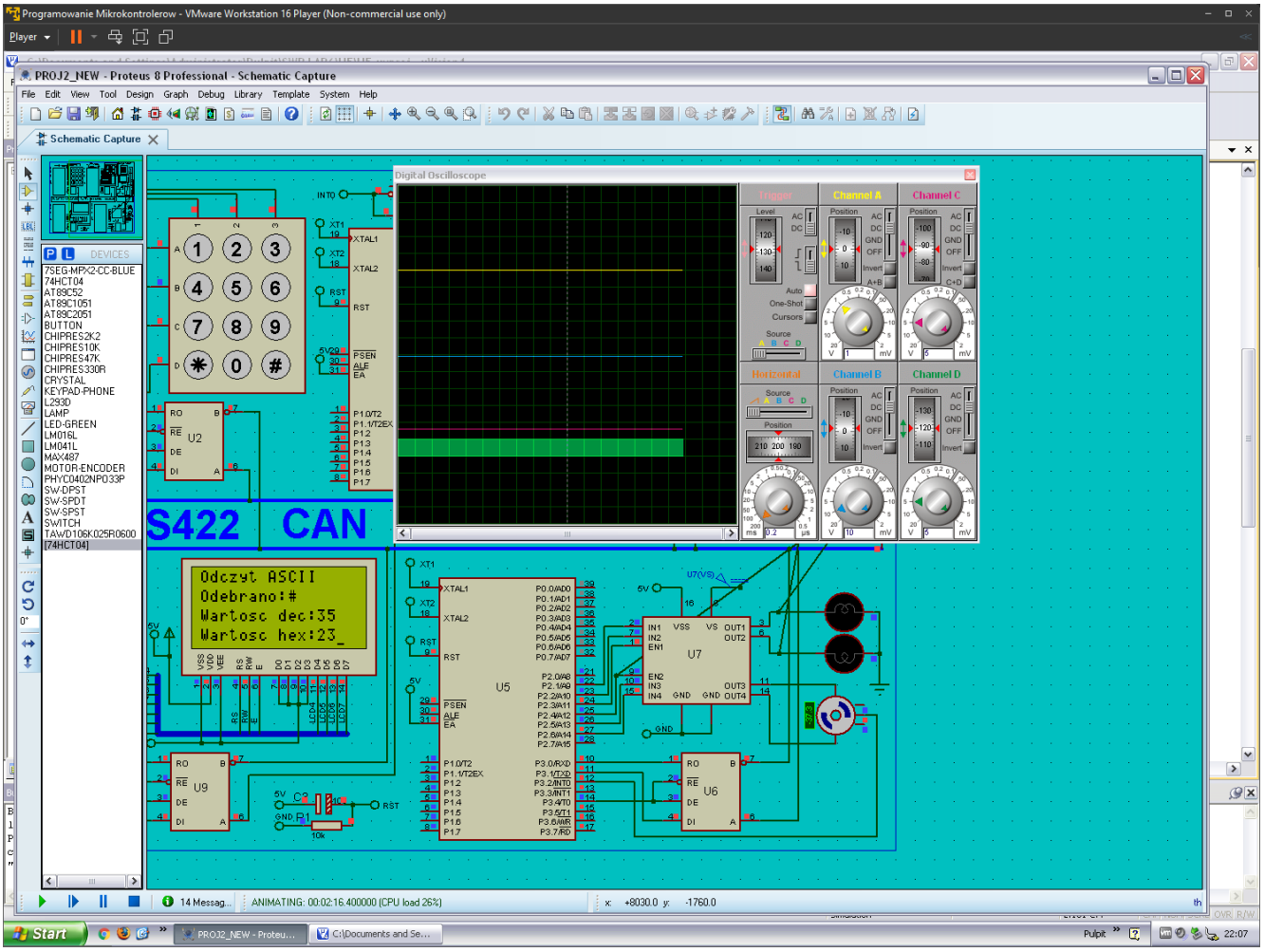
Wciśnięcie 4, obroty w prawo, dolna żarówka się się świeci, górna żarówka zmienia stan (niebieska fala reprezentuje żarówkę górną, czerwona żarówkę dolną).



Wciśnięcie *, żarówki się nie świecą, silnik dalej pracuje



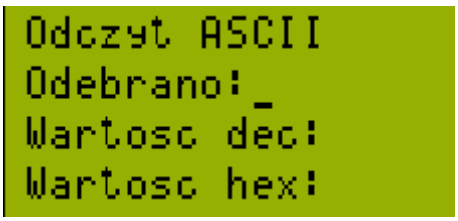
Wciśnięcie #, silnik zmienia kierunek obrotów, żarówki nadal się nie świecą.



A. Zadanie na układ U8

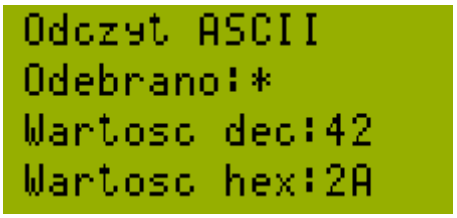
Sformułowanie problemu

Stworzyć program, który wyświetla na wyświetlaczu krystalicznym znaki w poniższej konfiguracji:



Program ma za zadanie przechwycić bajt wysłany przez układ U1 na magistralę, a następnie uzupełnić powyższy tekst wyświetlany odebranym znakiem, jego wartością decymlną ASCII oraz jego wartością hexadecymalną ASCII.

Przykład. Po odebraniu znaku „*” na wyświetlaczu widać taki zapis:



Zastrzeżenie jest takie, że nie może być żadnej przerwy między znakami „:” a wyświetlanymi symbolami.

Opis mojego rozwiązania

Aby umożliwić odbiór sygnału z magistrali, wykorzystam funkcję init() z programu dla układu U1, rozrzerzoną o możliwość odbioru sygnału (ES=1) oraz nadaniu mu wyższego priorytetu (PS=0x10).

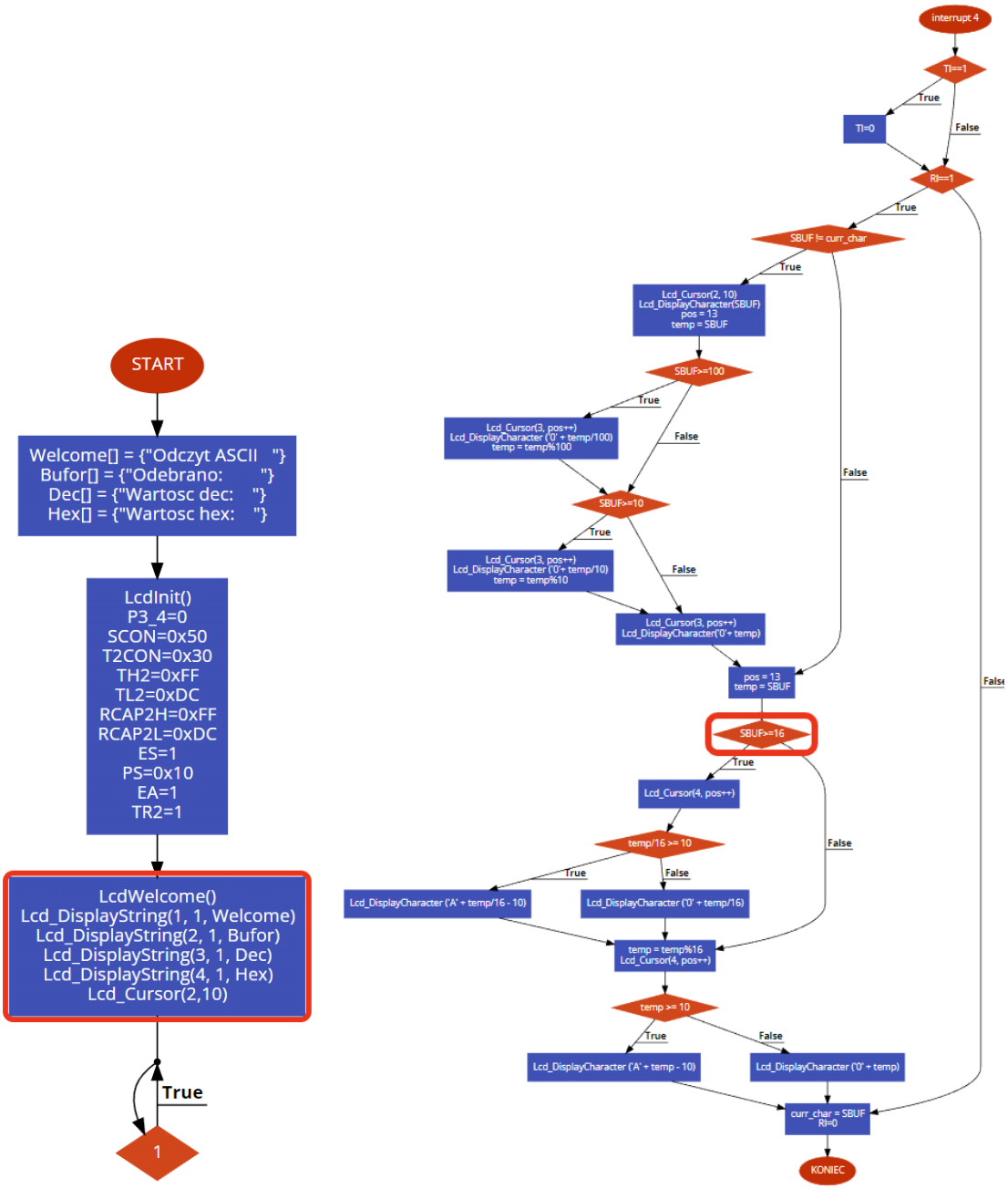
Do rozwiązania wykorzystam plik „LCD.c" oraz „wait1.a51". Pozwolą one na wypisywanie pojedynczych znaków, jak i całych łańcuchów znaków na wyświetlaczu. Do obsługi wyświetlacza będę wykorzystywał następujące funkcje z pliku „LCD.c":

- LcdInit() – inicjalizacja wyświetlacza (4-wierszowy, z 4-bitową magistralą danych)
- Lcd_Cursor() – do ustawiania położenia migającego kursora (przyda się do wpisywania pojedynczych znakow), położenie określone jest numerem wiersza oraz kolumny
- Lcd_DisplayCharacter() – do wypisywania pojedynczego znaku
- Lcd_DisplayString() – do wypisywania łańcuchu znaków

Na początku zadeklarowałem łańcuchy znaków, aby wypisać je na wyświetlaczu. Następnie stworzyłem funkcję ISR_Serial, która na podstawie interrupt 4 pobiera z SBUF nowy znak. Jeśli jest to nowy znak, wtedy wywołuje funkcję display_new(). Działanie tej funkcji jest następujące:

1. Wypisuje znak w drugiej linii.
2. Wypisuje cyfra po cyfrze wartość decymalną w ASCII tego znaku w trzeciej linii.
3. Wypisuje znak po znaku wartość hexadecymalną w ASCII tego znaku w czwartej linii.

Schemat blokowy rozwiązania



Listing programu

Program główny: U5.c

```
#include <REGX52.H>

extern void LcdInit();
extern void LcdWelcome();
extern void Lcd_Cursor(char row, char column);
extern void Lcd_DisplayCharacter(char a_char);
extern void Lcd_DisplayString(char row, char column, char *string);
extern void Lcd_WriteControl(unsigned char LcdCommand);

unsigned char data Var1, Var2, Var3;
volatile unsigned char data Title[] = {"Odczyt ASCII   "};

unsigned char curr_char;
unsigned char pos;
```

```
unsigned char temp;

void init(void)
{
    P3_4=0;
    SCON=0x50;
    T2CON=0x30;
    TH2=0xFF;
    TL2=0xDC;
    RCAP2H=0xFF;
    RCAP2L=0xDC;

        // Konfiguracja odbiornika
        ES=1;
    PS=0x10;

    EA=1;
    TR2=1;
}

void display_new(unsigned char c)
{
    Lcd_Cursor(2, 10);
    Lcd_DisplayCharacter(c);

    // Dec
    pos = 13;
    temp = c;

    if(c>=100)
    {
        Lcd_Cursor(3, pos++);
        Lcd_DisplayCharacter ('0' + temp/100);
        temp = temp%100;
    }
    if(c>=10)
    {
        Lcd_Cursor(3, pos++);
        Lcd_DisplayCharacter ('0'+ temp/10);
        temp = temp%10;
    }
    if(c>=0)
    {
        Lcd_Cursor(3, pos++);
        Lcd_DisplayCharacter ('0'+ temp);
    }

    // Hex
    pos = 13;
    temp = c;

    if(c>=16)
    {
```

```
        Lcd_Cursor(4, pos++);
        if(temp/16 >= 10)
            Lcd_DisplayCharacter ('A' + temp/16 - 10);
        else
            Lcd_DisplayCharacter ('0' + temp/16);
        temp = temp%16;
    }
    if(c>=0)
    {
        Lcd_Cursor(4, pos++);
        if(temp >= 10)
            Lcd_DisplayCharacter ('A' + temp - 10);
        else
            Lcd_DisplayCharacter ('0' + temp);
    }
}

void main(void)
{
    LcdInit();
    init();
    LcdWelcome();
    Lcd_DisplayString(1, 1, Title);
    Lcd_Cursor(2,10);
    while(1){;}

}

void ISR_Serial(void) interrupt 4
{
    if(TI==1){TI=0;}
    if(RI==1)
    {
    if(SBUF != curr_char)
    {
        display_new(SBUF);
        curr_char = SBUF;
    }
    RI=0;
    }
}
```

Program LCD.c

```
// Autor: dr inz. Krzysztof Murawski

#define Test 0  // 1 - do testowania
                // 0 - normalna praca
// Definicje zasobow sprzetowych
sfr  P2    = 0xA0;
sbit P2_6  = P2^6;
sbit P2_5  = P2^5;
```

```
sbit P2_4  = P2^4;
sbit P2_3  = P2^3;
sbit P2_2  = P2^2;
sbit P2_1  = P2^1;
sbit P2_0  = P2^0;

// Definicje ogolne
#define False               0
#define True                1

// Definicje podlaczenia wyswietlacza LCD
#define LCD_RS    P2_4         /* p1.4 LCD Register Select line          */
#define LCD_RW    P2_5         /* p1.5 LCD Read / Write line             */
#define LCD_E     P2_6         /* p1.6 LCD Enable line                   */
#define LCD_DB4   P2_0         /* high nibble of port 1 is used for data */
#define LCD_DB5   P2_1         /* high nibble of port 1 is used for data */
#define LCD_DB6   P2_2         /* high nibble of port 1 is used for data */
#define LCD_DB7   P2_3         /* high nibble of port 1 is used for data */

// Definicje komend wyswietlacza LCD
#define LCD_CONFIG            0x28
#define LCD_CLEAR               0x01
#define LCD_HOME                0x02
#define LCD_ENTRY_MODE        0x06
#define LCD_DISPLAY_OFF       0x08
#define LCD_CURSOR_ON         0x0A
#define LCD_DISPLAY_ON        0x0C
#define LCD_CURSOR_BLINK      0x0D
#define LCD_CURSOR_LINE       0x0E
#define LCD_CURSOR_COM        0x0F
#define LCD_CURSOR_LEFT       0x10
#define LCD_CURSOR_RIGHT      0x14
#define LCD_SHIFT_LEFT        0x18
#define LCD_SHIFT_RIGHT       0x1C
#define LCD_SET_CGRAM_ADDR  0x40
#define LCD_SET_DDRAM_ADDR  0x80

// Definicje ekranow
static unsigned char code Screan[] =  "                "
                                      "Odebrano:       "
                                      "Wartosc dec:    "
                                      "Wartosc hex:    ";

// Importowane procedury i funkcje
extern w1ms();
extern w5mS();
extern w50mS();

//Wyslanie komendy do wyswietlacza LCD. Magistrala danych 4-o bitowa
void Lcd_WriteControl (unsigned char LcdCommand)
 {
   unsigned char Lcd_Comm = 0;
   static bit LCDReady;
```

```
   LCD_RS = False;
   LCD_RW = False;

   Lcd_Comm = LcdCommand >> 4;
   P2 &= 0xF0;
   P2 |= Lcd_Comm;

   LCD_E = True;
   LCD_E = False;

   Lcd_Comm = LcdCommand & 0x0F;
   P2 &=0xF0;       // P2 = P2 & 0xF0
   P2 |= Lcd_Comm;

   LCD_E = True;
   LCD_E = False;

   P2 |= 0x0F;      // P2 = P2 | 0x0F

   LCD_RW = True;
   LCD_RS = False;

   if (Test == 0)
    {
      LCDReady = 1;
      while (LCDReady == 1)
       {
         LCD_E = True;
         LCDReady = LCD_DB7;
         LCD_E  = False;
         LCD_E = True;
         LCD_E  = False;
       }
    }
 }

// Wyslanie danych do wyswietlacza LCD. Magistrala 4-o bitowa
static void Lcd_WriteData (unsigned char LcdData)
 {
   unsigned char Lcd_Data = 0;
   static bit LCDReady;

   LCD_RS = True;
   LCD_RW = False;

   Lcd_Data = LcdData >> 4;
   P2 &= 0xF0;
   P2 |= Lcd_Data;

   LCD_E = True;
   LCD_E = False;
```

```
   Lcd_Data = LcdData & 0x0F;
   P2 &=0xF0;
   P2 |= Lcd_Data;

   LCD_E = True;
   LCD_E = False;

   P2 |= 0x0F;

   LCD_RW = True;
   LCD_RS = False;

   if (Test == 0)
    {
      LCDReady = 1;
      while (LCDReady == 1)
       {
         LCD_E = True;
         LCDReady = LCD_DB7;
         LCD_E  = False;
         LCD_E  = True;
         LCD_E  = False;
       }
    }
 }

// Wyswietlenie znaku w miejscu polozenie kursora
void Lcd_DisplayCharacter (char a_char)
 {
   Lcd_WriteData(a_char);
 }

/* Ustawienie kursora na wskazany wiersz i kolumne.
*     1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
*    ---------------------------------------------------------------
* 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
* 2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
* 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
* 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
*/
void Lcd_Cursor (char row, char column)
 {
   if (row == 1) Lcd_WriteControl (0x80 + column - 1);
   if (row == 2) Lcd_WriteControl (0xc0 + column - 1);
   if (row == 3) Lcd_WriteControl (0x90 + column - 1);
   if (row == 4) Lcd_WriteControl (0xd0 + column - 1);
 }

// Wyswietlenie ciagu znakow w konkretym wierszu (bez zawijania wiersza)
void Lcd_DisplayRow (char row, char *string)
 {
   char i;
   Lcd_Cursor (row, 1);
```

```
   for (i=0; i<16; i++) Lcd_DisplayCharacter (*string++);
 }


// Wyswietlenie calego ekranu - 64 znaki. Znaki zawarte w tablicy
/* przyklad:
** char screen[] =  "PBW-850   2003r."
**                  "---------------"
**                  "  K. Murawski   "
**                  "J. Chudzikiewicz";
**      LCD_DisplayScreen(screen);
*/
void Lcd_DisplayScreen (char *ptr)
 {
   Lcd_DisplayRow(1,ptr +  0);
   Lcd_DisplayRow(2,ptr + 16);
   Lcd_DisplayRow(3,ptr + 32);
   Lcd_DisplayRow(4,ptr + 48);
 }

// Wyswietlenie ekranu powitalnego
void LcdWelcome(void)
 {
   Lcd_DisplayScreen(Screan);
 }

// Wyswietlenie ciagu znakow od danej kolumny i wiersza
void Lcd_DisplayString (char row, char column, char *string)
 {
   Lcd_Cursor (row, column);
   while (*string) Lcd_DisplayCharacter (*string++);
 }

// Inicjalizacja wyswietlacza; 4 wiersze, 4 bitowa magistrala danych
void LcdInit(void)
 {
   w50mS();
   P2 = 0x83;
   LCD_E   = True;
     w1ms();
   LCD_E   = False;
   w5mS();
   LCD_E   = True;
     w1ms();
   LCD_E   = False;
   w1ms();
   LCD_E   = True;
     w1ms();
   LCD_E   = False;
   w1ms();
   LCD_DB4 = False;
   LCD_E   = True;
     w1ms();
```

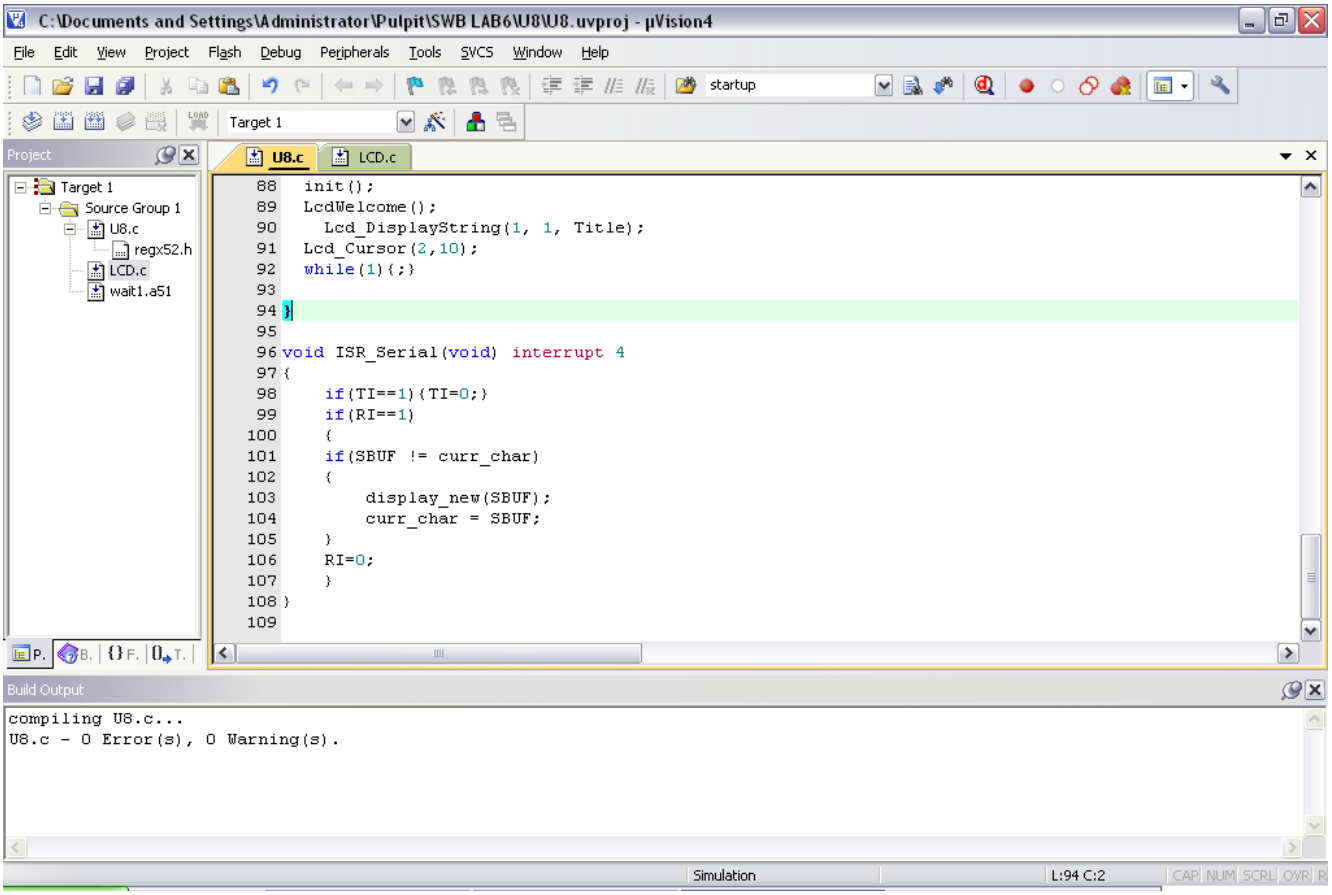
```
   LCD_E   = False;
   w1ms();
   Lcd_WriteControl(LCD_CONFIG);
   Lcd_WriteControl(LCD_CLEAR);
   Lcd_WriteControl(LCD_DISPLAY_OFF);
   Lcd_WriteControl(LCD_DISPLAY_ON);
   Lcd_WriteControl(LCD_ENTRY_MODE);
   Lcd_WriteControl(LCD_CURSOR_COM);
   Lcd_WriteControl(LCD_CLEAR);
 }
```

Program wait1.a51

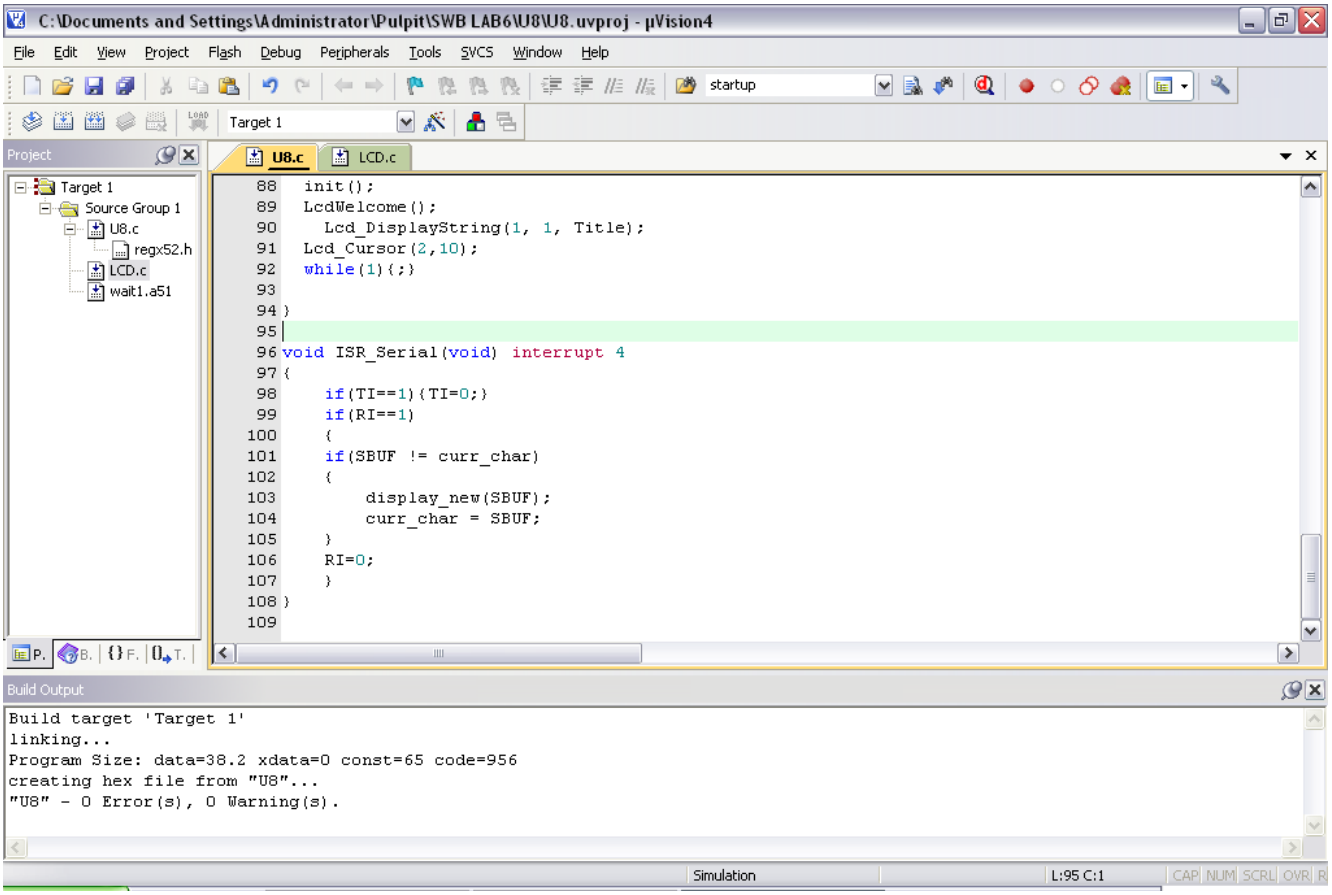
```
;Autor: dr inz. Krzysztof Murawski

NAME    wait

EXTRN   DATA (Var1, Var2, Var3)

PUBLIC  w1ms, w5mS, w50mS

Wait_ROUTINES  SEGMENT  CODE

    RSEG  Wait_ROUTINES

w1ms:                ;
      MOV    Var2,#4    ;
      MOV    Var1,#224  ;
TT2:    DJNZ Var1,TT2   ;
        DJNZ Var2,TT2   ;
        RET             ;

w5mS:                ;
      MOV    Var2,#20   ;
      MOV    Var1,#112  ;
TT3:    DJNZ Var1,TT3   ;
        DJNZ Var2,TT3   ;
        RET               ;

w50mS:               ;
      MOV    Var2,# 195 ;
      MOV    Var1,# 137 ;
TT5:    DJNZ Var1,TT5   ;
        DJNZ Var2,TT5   ;
        RET
END
```

Sprawdzenie poprawności

Kompilowanie

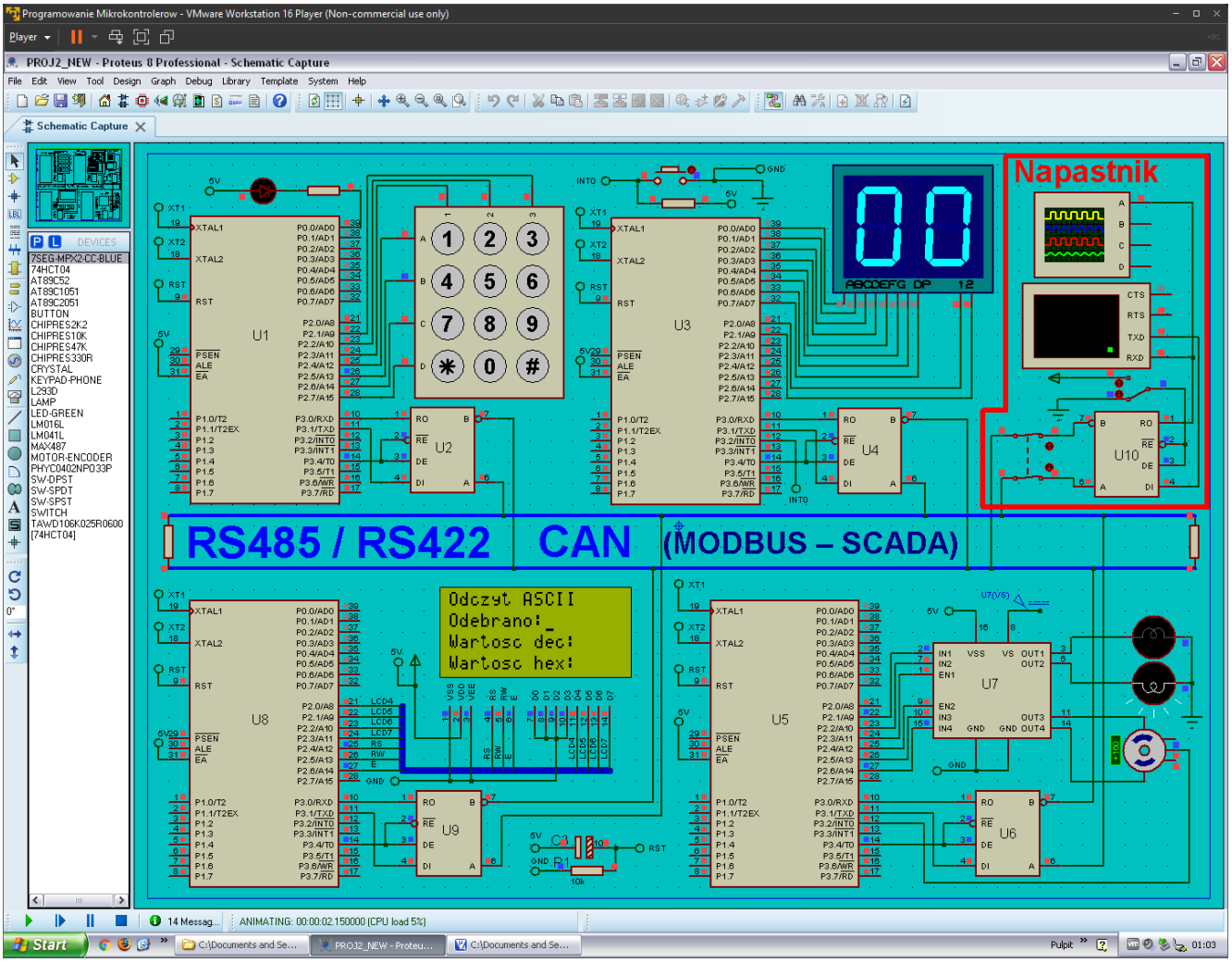


Linkowanie

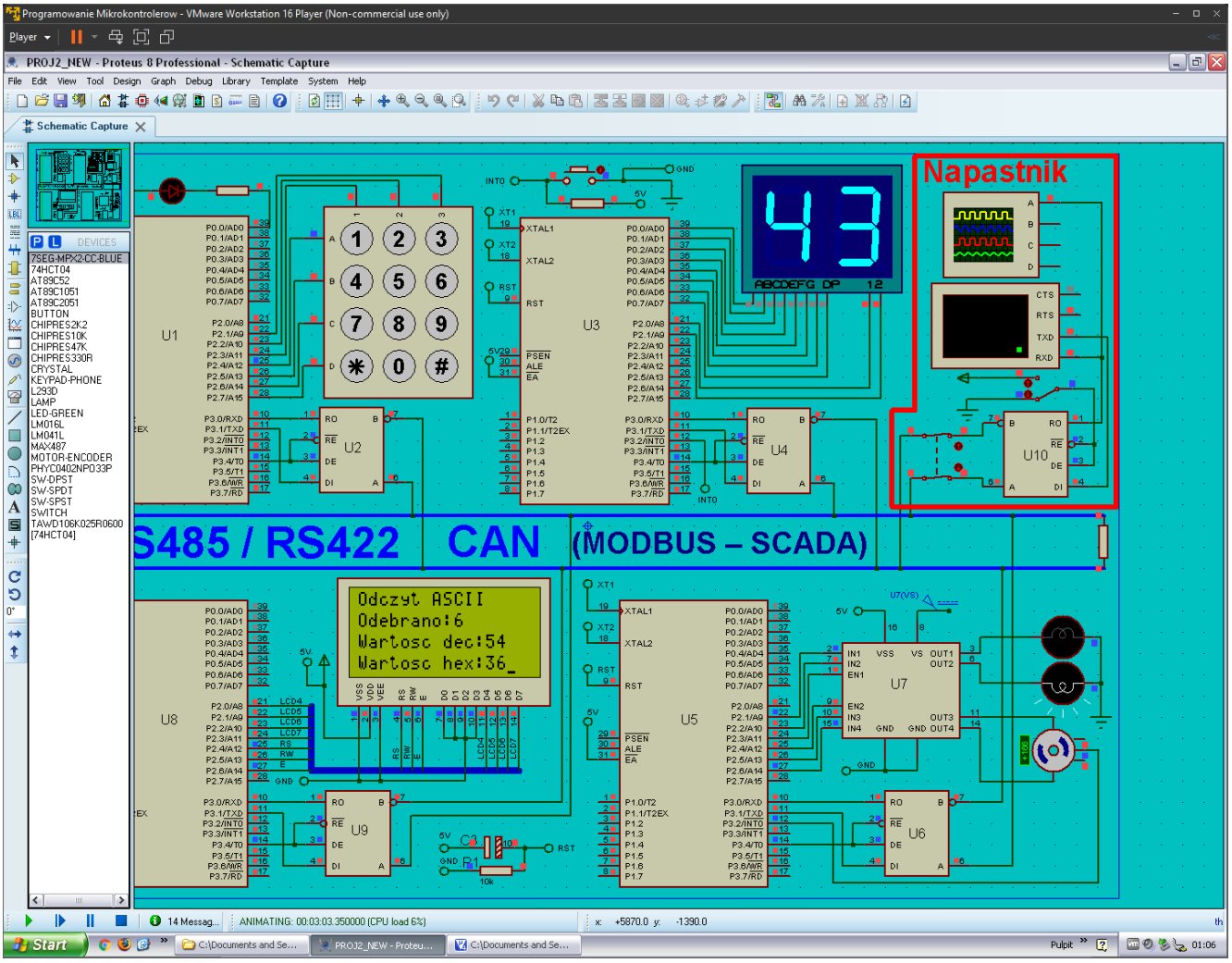


Prezentacja realizacji zadania przez program

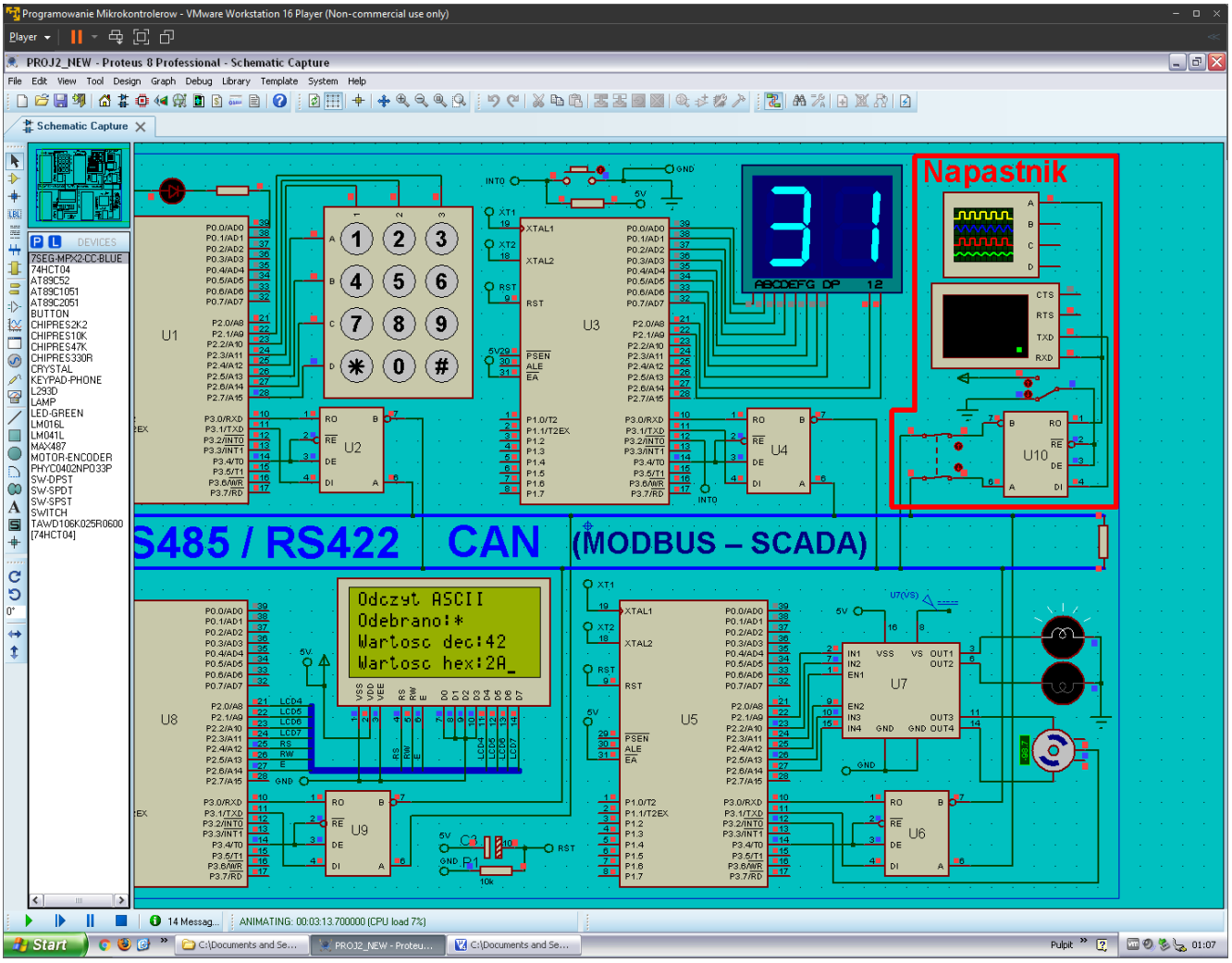
Stan początkowy:



Wciśnięcie i odebranie 6:



Wciśnięcie i odebranie *:


Programowanie Mikrokontrolerow - VMware Workstation 16 Player (Non-commercial use only)
PROJ2_NEW - Proteus 8 Professional - Schematic Capture
Napastnik
S485 / RS422
CAN
(MODBUS – SCADA)
Odczyt ASCII
Odebrano:*
Wartosc dec:42
Wartosc hex:2A_
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
14 Messag...
ANIMATING: 00:03:13.700000 (CPU load 7%)
Start
01:07