Spis treści

1 Ustaw układ licznikowy T0 w trybie licznika 16 bitowego: 2

2 Ustaw układ licznikowy T1 w trybie czasomierza 16 bitowego: 5

3 Proste wykorzystanie czasomierza: 6

4 Proste wykorzystanie czasomierza: 7

5 Złożone programy wykorzystujące oba układy licznikowe: 9

1 Ustaw układ licznikowy T0 w trybie licznika 16 bitowego:

1.1 Przygotuj makroinstrukcje do konfiguracji układu licznikowego T0:

Prezentacja implementacji rozwiązania:

```
1000
    TMOD = 0x01;  //0101 tryb 16-Bit Timer/Counter
1002
    //Calculate initial value - count 5, maximum value beafore
1004 //overflow is 2 power 16 is equal 65536 (mode 1 - 16 bitt).
    counter_timer_0_initial_value = (65536-5); //liczy od tej wartosci 5 impulsow
      i powrot
1006
    //Set Timer 0 high byte and low byte. //resjest na ktorym operuje timer
1008 TL0 = counter_timer_0_initial_value;    //Set low byte (8 bits)
    //from 16 bits initial value (bits 0-7)
1010 TH0 = counter_timer_0_initial_value>>8; //Set high byte (8 bits)
//  from 16 bits initial value (bits 8-15)
1012
    //Run timer/counter number 0
1014 TR0 = 1; //Set by user to turn on Timer/Counter 0. //zalaczenie timera
```

Wnioski:
Powyższy listnig przedstawia sposób konfiguracji układu licznikowego.

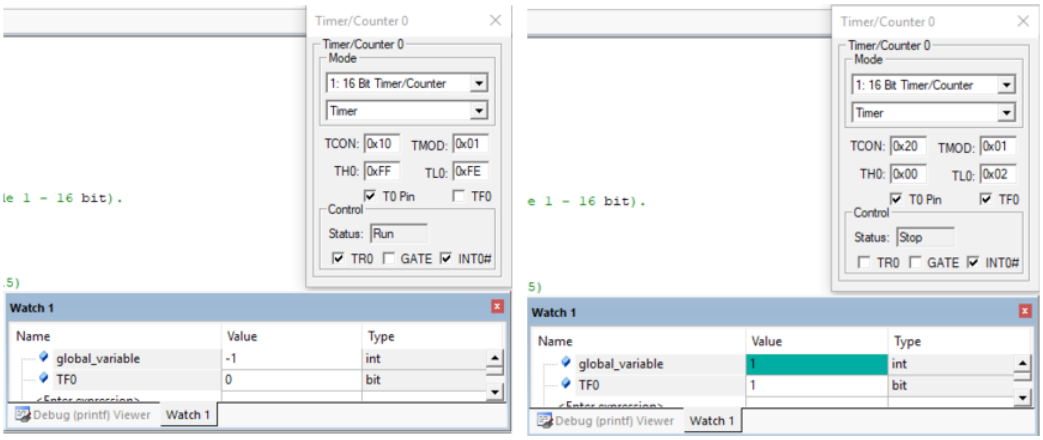
1.2 Napisz fragment kodu w pętli głównej sprawdzający stan flagi:

Prezentacja implementacji rozwiązania:

```
1000 #include "aduc831.h"  //Definitions of ADuC831 registers name
    #include "stdint.h"   //Standard integers
1002 #include "stdfloat.h" //Standard float
    #include "IO.h"       //Input/output definitions
1004
    //Global variable definitions
1006 int global_variable = 1;  //wartosc zero
    uint16_t counter_timer_0_initial_value; //deklaracja typu
1008
    void main(void) //funkcja glowna
1010 {
      TMOD = 0x01;  //0101 tryb 16-Bit Timer/Counter
1012
      //Calculate initial value - count 5, maximum value beafore
1014  //overflow is 2 power 16 is equal 65536 (mode 1 - 16 bitt).
      counter_timer_0_initial_value = (65536-5); //liczy od tej wartosci 5 impulsow
        i powrot
1016
      //Set Timer 0 high byte and low byte. //resjest na ktorym operuje timer
1018  TL0 = counter_timer_0_initial_value;    //Set low byte (8 bits)
      //from 16 bits initial value (bits 0-7)
1020  TH0 = counter_timer_0_initial_value>>8; //Set high byte (8 bits)
    //  from 16 bits initial value (bits 8-15)
```

```
1022
   //Run timer/counter number 0
1024  TR0 = 1; //Set by user to turn on Timer/Counter 0. //zalaczenie timera

1026  while(1) //petla nieskaczona
      {
1028    if (TF0==1) //przepelnienie if TF0 == 1// flaga ustawiona
        {
1030      TR0 = 0; //stop licznika
          global_variable =~global_variable+1;  //zmiana stanu na przeciwny
1032
          TL0 = counter_timer_0_initial_value;  //ponowne zapelnienie rejestru
1034      TH0 = counter_timer_0_initial_value>>8;

1036      TF0 = 0; //kasowanie flagi
          TR0 = 1; //uruchomienie licznika
1038    }
      };
1040  }
```

Prezentacja wyników testów:



Rysunek 1: Left: Stan 1. Right: Stan 2.

Wnioski:
Zgodnie z komentarzami na listingu po wczesniejszym ustawieniu układu licznika w pętli głównej następuję cykliczna zmiana stanu flagi $global_variablenaprzeciwny.Jesttoprzedstawionenarysunku1.C$

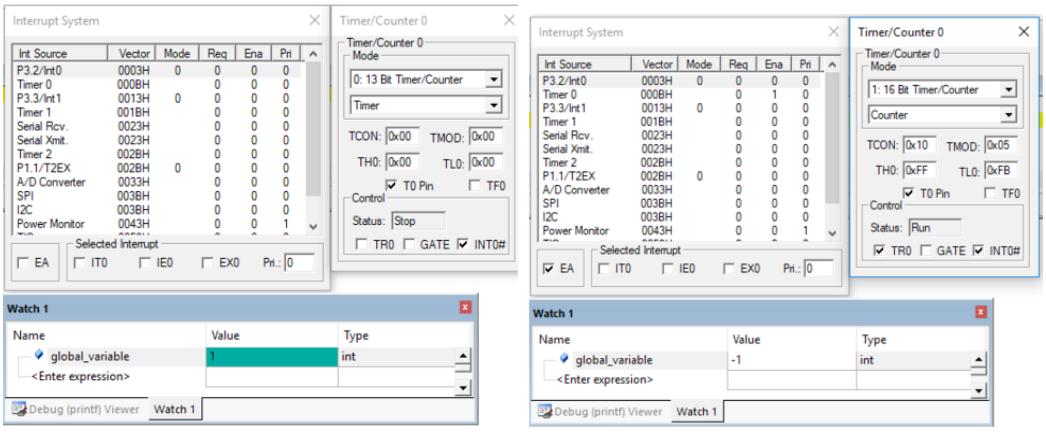
1.3 Napisz funkcję obsługującą przerwanie od przepełnienia układu licznikowego T0:

Prezentacja implementacji rozwiązania:

```
1000 #include "aduc831.h"   //Definitions of ADuC831 registers name
     #include "stdint.h"    //Standard integers
1002 #include "stdfloat.h" //Standard float
     #include "IO.h"        //Input/output definitions
1004
     //Global variable definitions
1006 int16_t global_variable = (0x01<<0);
```

```
uint16_t counter_timer_0_initial_value;

/**
* Handle counter/timer number 0 interrupt
*/
void counter_timer_0 (void) interrupt 1 //przerwanie nr1 counter'a 0
{
  //operacje w przerwaniu
    TR0 = 0; //zatrzymuje licznik
    global_variable = ~global_variable+1;
    TL0 = counter_timer_0_initial_value;
    TH0 = counter_timer_0_initial_value>>8;;
    TR0 = 1;
}
void main(void)
{
  TMOD = 0x05;//0101  16-Bit Timer/Counter

  //Calculate initial value - count 5, maximum value beafore overflow is 2 power
    16 is equal 65536 (mode 1 - 16 bitt).
  counter_timer_0_initial_value = 65536-5;

  //Set Timer 0 high byte and low byte.
  TL0 = counter_timer_0_initial_value;
  TH0 = counter_timer_0_initial_value>>8;

  //Run timer/counter number 0
  TR0 = 1; //Set by user to turn on Timer/Counter 0.
  ET0 = 1; //uruchomienie przerwan licznika 0
  EA = 1; //uruchomienie przerwan

  while(1)
  {
  };
}
```

Prezentacja wyników testów:



Rysunek 2: Left: Stan 1. Right: Stan 2.

Wnioski:
Zastosowano tutaj przerwanie (void counter-timer-0 (void) interrupt 1). Działa ono niezależnie

od pętli while(1) dlatego też jest ona niużywana w tym przypadku. Kiedy licznik odmierzy daną wartość procedura obsługi przerwania zostanie wywołana. Licznik się wyłączy zmieni się wartość flagi. Licznik zostanie zresetowany i ponownie załączony. Przedstawiono wizualizację na rysunku 2.

2 Ustaw układ licznikowy T1 w trybie czasomierza 16 bitowego:

Prezentacja implementacji rozwiązania:

```
#include "aduc831.h"   //Definitions of ADuC831 registers name
#include "stdint.h"    //Standard integers
#include "stdfloat.h" //Standard float
#include "IO.h"        //Input/output definitions

#define IMPULS 20 //definiujemy impuls 0x14
#define N (0xFFFF-(IMPULS)) //zmienna N (wartosc licznika) //FFDF?//FFEB

//Global variable definitions
int16_t global_variable = 1;
uint16_t counter_timer_0_initial_value;

void main(void)
{
  TMOD = 0x10; //11010000 tryb 16-Bit Timer/Counter

  //Set Timer 0 high byte and low byte.
  TL1 = N&0xFF; //
  TH1 = N>>8;  //przesuniecie bitowe w prawo

  //Run timer/counter number 0
  TR1 = 1; //Set by user to turn on Timer/Counter 0.

  while(1)
  {
    if (TF1==1)
    {
      TR1=0;
      global_variable=~global_variable+1;
      TL1=N&0xFF; //lows bytes of N
      TH1=N>>8;
      TF1=0;
      TR1=1;
    }
  };
 }
```

Prezentacja wyników testów:

Wnioski:
Układ został ustawiony jako czasomierz 16-bitowy (TMCD = 0x10;). Następnie ustawiono wartość rejestru przedstawiającego wartość zliczaną. W pętli głownej kiedy flaga TF1 == 1 następuję zmiana wartości global-variable i cały proces zostaje powturzony. Rysunek 3 left przed załączeniem. Rysunek Right ukazuję zależność pomiędzy global-variable a TF1.

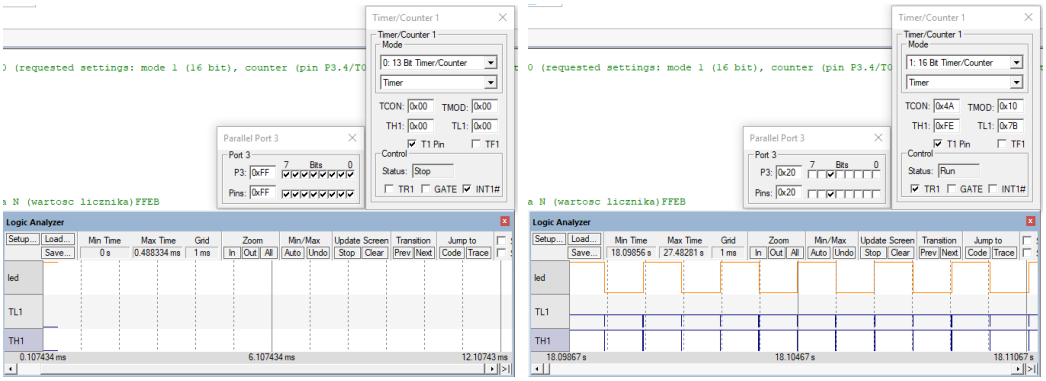
Rysunek 3: Left: Po zaniku sygnału P3.3. Right: Przed zanikiem sygnału P3.3

3 Proste wykorzystanie czasomierza:

Prezentacja implementacji rozwiązania:

```
#include "aduc831.h"   //Definitions of ADuC831 registers name
#include "stdint.h"    //Standard integers
#include "stdfloat.h" //Standard float
#include "IO.h"        //Input/output definitions

#define PRESCALER 12 //dzielnik
#define CLOCK 11058000 //czestotliwosc zegara
#define TIME 1000 // czas w milisekundach = 1s
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
    (wartosc licznika)FFEB
sbit led=P3^5; //przypisanie diody do portu
//921//64614//1111 1100 0110 0110
//Global variable definitions
int16_t global_variable = 1;
uint16_t counter_timer_0_initial_value;

void counter_timer_0 (void) interrupt 3 //zmiana stanu diody na przeciwny
{
       TR1=0;
       led=~led;
       TL1=N&0xFF; //lows bytes of N
       TH1=N>>8;
       TR1=1;
}

void main(void)
{
  P3=0x00;
  led=0;
  TMOD = 0x10;   //0xX5 in hexadecimal //11010000 tryb 16-Bit Timer/Counter
  TL1 = N&0xFF;      //Set low byte (8 bits) from 16 bits initial value (bits
    0-7)  //operacja bitowa AND
  TH1 = N>>8;  //Set high byte (8 bits) from 16 bits initial value (bits 8-15)
  ET1=1;
  EA=1;

  TR1 = 1; //Set by user to turn on Timer/Counter 0.

  while(1)
  {
  };
}
```

Prezentacja wyników testów:



Rysunek 4: Left: Stan 1. Right: Stan 2.

Wnioski:

Program cyklicznie zmienia stan diody na przeciwny w funkcji obsugującej przerwanie z częstotliwościa równą 1s. Rysunek 4 right przedstawia zapalanie i gaśnięcie diody.

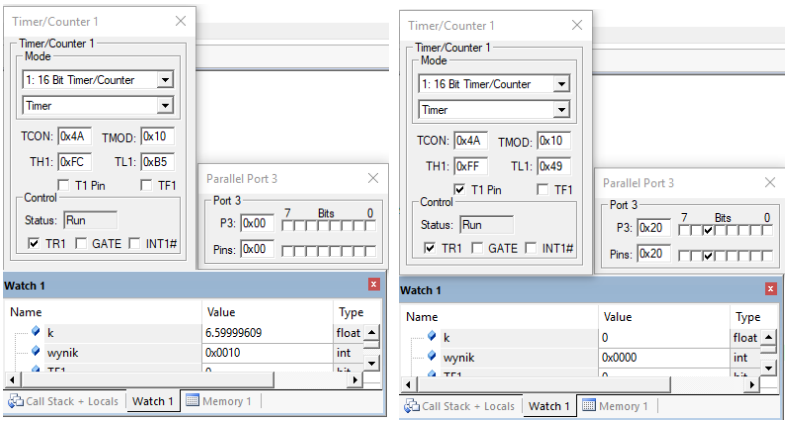
4 Proste wykorzystanie czasomierza:

Prezentacja implementacji rozwiązania:

```
#include "aduc831.h"   //Definitions of ADuC831 registers name
#include "stdint.h"    //Standard integers
#include "stdfloat.h" //Standard float
#include "IO.h"        //Input/output definitions
#include <math.h>

#define PRESCALER 12 //dzielnik
#define CLOCK 11058000 //czestotliwosc zegara
#define TIME 1000 // czas w milisekundach
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
    (wartosc licznika) //FFDF?//FFEB
sbit led1=P3^5; //przypisanie diody do portu
sbit led2=P3^6;
float k;
int wynik; //zmienna globalna wynik deklaracja
//921//64614//1111 1100 0110 0110
//Global variable definitions
int16_t global_variable = 1;
uint16_t counter_timer_0_initial_value;

void counter_timer_0 (void) interrupt 3
{
        TR1 = 0;
        led1 = ~led1;
        TL1 = N&0xFF;
        TH1 = N>>8;
        TR1 = 1;
```

```
1026 }
1028 void main(void)
     {
1030
       wynik = 0;
1032   P3 = 0x00; //czyszczenie portu (wpisanie 0)
       led1 = 0; //wartosc poczatkowa
1034   led2 = 0;

1036   TMOD = 0x10;  //0xX5 in hexadecimal //11010000 tryb 16-Bit Timer/Counter

1038   //Set Timer 0 high byte and low byte.
       TL1 = N&0xFF;      //Set low byte (8 bits) from 16 bits initial value (bits
        0-7)  //operacja bitowa AND
1040   TH1 = N>>8;  //Set high byte (8 bits) from 16 bits initial value (bits 8-15)
       ET1 = 1;
1042   EA = 1;

1044   TR1 = 1; //Set by user to turn on Timer/Counter 0.

1046   while(1)
       {
1048       if(TF1==1)
           {
1050       TR1 = 0;
           k=k+0.1; //dodanie skladnika do sumy(zmiana cykliczna)
1052       wynik=k*sqrt(k);
           if(k>10)k = 1;
1054       {
             led2=~led2;
1056       }
           TL1 = N&0xFF; //lows bytes of N
1058       TH1 = N>>8;
           TR1 = 1;
           }
1060    };

1062
     }
```

Prezentacja wyników testów:



Rysunek 5: Left: Stan 1. Right: Stan 2.

Wnioski:

Umieszczenie obliczeń w pętli głównej w odróżnieniu od umieszczenia ich w przerwaniu powoduje, że mogą one zostać przerwane przez pojawienie się przerwania.

5 Złożone programy wykorzystujące oba układy licznikowe:

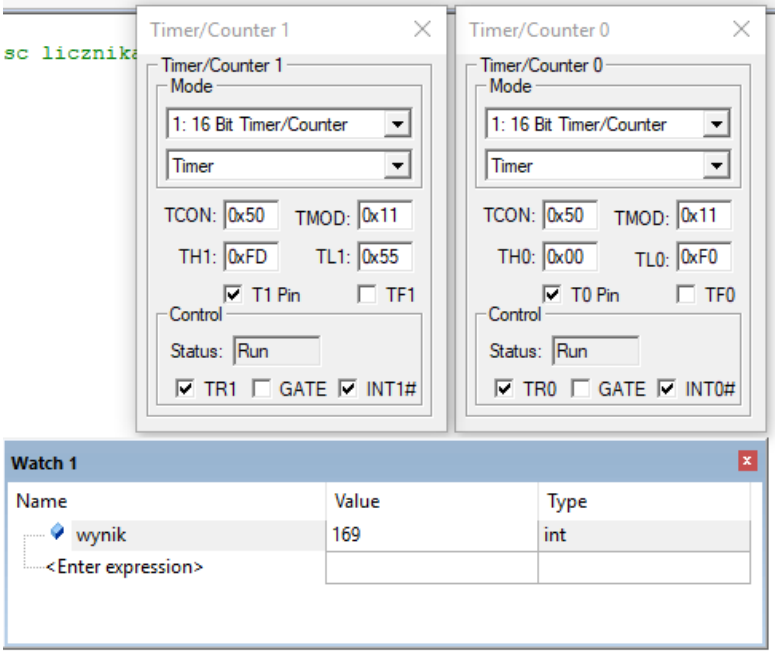
5.1 Wyznaczyć liczbę impulsów zewnętrznych w zadanym okresie czasu:

Prezentacja implementacji rozwiązania:

```
#include "aduc831.h"   //Definitions of ADuC831 registers name
#include "stdint.h"    //Standard integers
#include "stdfloat.h" //Standard float
#include "IO.h"        //Input/output definitions
#include <math.h>

#define PRESCALER 12 //dzielnik
#define CLOCK 11058000 //czestotliwosc
#define TIME 1000 // czas w milisekundach
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
    (wartosc licznika) //FFEB

int16_t wynik;
//921//64614//1111 1100 0110 0110
//Global variable definitions
int16_t global_variable = 1;
uint16_t counter_timer_0_initial_value;

void counter_timer_0 (void) interrupt 3
{
     TR1 = 0;
     TR0 = 0;
     wynik = (TH0>>8) + TL0;  //czas pomiedzy impulsami zewnetrznymi
     TL0 = 0x00; // z tego rejestru pobrane dane
     TH0 = 0x00; // z tego rejestru pobrane dane
     TL1 = N&0xFF;
     TH1 = N>>8;
     TR0 = 1;
     TR1 = 1;
}
void main(void)
{

  TMOD = 0x11;
   wynik = 0;

   TL0 = 0;
   TH0 = 0;

   TL1 = N&0xFF;
   TH1 = N>>8;
   ET0 = 0;
   ET1 = 1;
   EA = 1;

   TR0 = 1;
   TR1 = 1;
```

```
1046
    while(1)
1048 {
    };
1050
}
```

Prezentacja wyników testów:



Rysunek 6: Liczba impulsów równa 169.

Wnioski:
Na rysunku 6 został przedstwiony wynik obliczeń. Zmienna out typu int przedstawia liczbę impulsów równą 169.

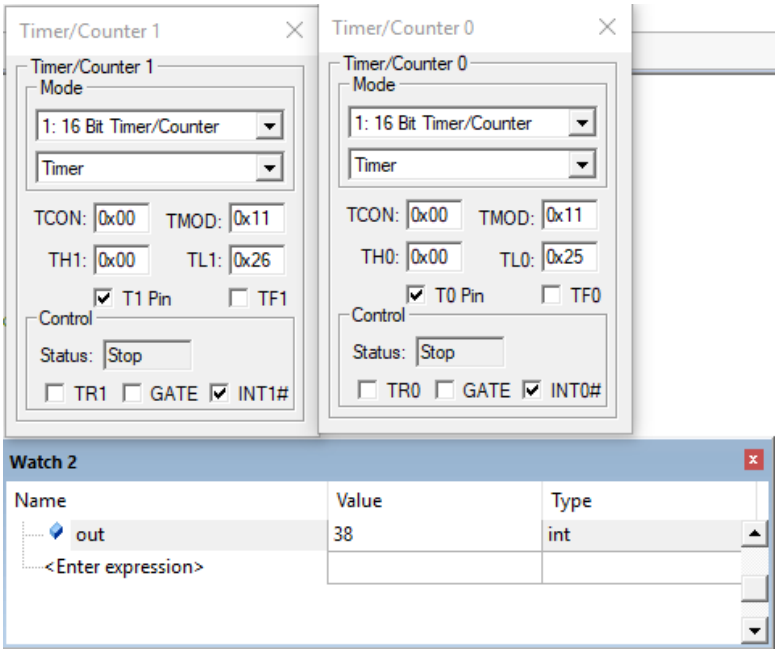
5.2 Wyznaczyć czas między kolejnymi impulsami zewnętrznymi:

Prezentacja implementacji rozwiązania:

```
1000 #include "aduc831.h"   //Definitions of ADuC831 registers name
     #include "stdint.h"    //Standard integers
1002 #include "stdfloat.h" //Standard float
     #include "IO.h"        //Input/output definitions
1004 #include <math.h>

1006 #define PRESCALER 12 //dzielnik
     #define CLOCK 11058000 //czestotliwosc
1008 #define TIME 1000 // czas w milisekundach
     #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
         (wartosc licznika) //FFDF?//FFEB
1010
     int out;
1012
     void counter_timer_0 (void) interrupt 1
```

```
{
    TR0 = 0;
    TR1 = 0;
    out = (TH1<<8) + TL1; //czas pomiedzy impulsami zewnetrznymi
    TL1 = 0x00;;
    TH1 = 0x00;;
    TL0 = 0xFF; // z
    TH0 = 0xFF;
    TR0 = 1;
    TR1 = 1;
}

void main(void)
{
  out = 0x00;;
  TMOD = 0x11;
  TL0 = 0xFF;
  TH0 = 0xFF;

  TL1 = 0x00;//operacja bitowa AND
  TH1 = 0x00;
  ET0 = 1;
  ET1 = 0;
  EA = 1;

  TR0 = 1;
  TR1 = 1;

  while(1);
}
```

Prezentacja wyników testów:



Rysunek 7: A

Wnioski:
Czas pomiędzy impulsami jest równy 38 co zostało przedstawione na rysunku 7.