Spis treści

1 Analiza działania przetwornika cyfrowo-analogowego:

2 „Ręczne” ustawienie DACx:

3 Wygeneruj sygnał piłokształtny:

3.1 używając dowolnego czasomierza wygeneruj sygnał piłokształtny o maksymalnej amplitudzie i dowolnym okresie, wyniki zarejestruj oscyloskopem:

```
1000 /*
     a.  uzywajac dowolnego czasomierza wygeneruj sygnal pilokształtny o maksymalnej
         amplitudzie
1002 i dowolnym okresie, wyniki zarejestruj oscyloskopem,
     */
1004
     #include "aduc831.h"  //Definitions of ADuC831 registers name
1006 #include "stdint.h"   //Standard integers
     #include "stdfloat.h" //Standard floatt
1008 #include "IO.h"       //Input/output definitions
     #include <math.h>
1010
     #define PRESCALER 12 //dzielnik
1012 #define CLOCK 11058000 //czestotliwosc
     #define TIME 1000 // czas w milisekundach
1014 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
         (wartosc licznika) //FFDF?//FFEB

1016 #define DAC_UREF 5.0
     #define DAC_RES 4095.0
1018 #define DAC_V2R(voltage) voltage/DAC_UREF+DAC_RES

1020 uint16_t v=0;
     uint16_t i=0;
1022 uint16_t counter_timer_0_initial_value;
     uint16_t r;
1024
     void counter_timer_0 (void) interrupt 1
1026 {
         TF0=0; //flaga przepelnienia licznika t0 kasujemy
1028     //Set Timer 0 high byte and low byte.
         TL0=N&0xFF; //lows bytes of N
1030     TH0=N>>8;
         r = DAC_V2R(6);
1032     i++;
         if(i==5) //okres probkowania ms
1034     {
           v+=16; //(napiecie maksymalne / 16) = ilosc stopni kwantyzacji na 1/2T =
         1000
1036       i=0;
         }
1038
         if(v>r) //1638 napiecie maksymalne
1040     {
           v=0;
1042     }
```

```
    // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
    DAC0H=v>>8;
    DAC0L=v;//bo rejestry 8bit//kasowanie
    //lub DAC0L=voltage-(0x0F & DAC0H);
}

void main(void)
{
    //control registerr dac 12bit/8bit voltage 0 to Vref
    //0.Set to   1   = Power-On DAC0.
    //1.Set to   0   = Power-Off DAC1.,
    //2.DAC0/1 Update Synchronization Bit. When set to   1   the DAC outputs
   update as soon as DACxL SFRs are written.
    //3.DAC0 Clear Bit. Set to   1   = DAC1 Output Normal.
    //4.DAC1 Clear Bit. Set to   1   = DAC1 Output Normal.
    //5.Set to   1   = DAC0 Range 0 V  DD.
    //6.Set to   1   = DAC1 Range 0 V  DD .
    //7.Set to   0  = 12-Bit Mode.
    DACCON=0x7D; //0111 1101
    //dataa registers
    DAC0H=0x00;
    DAC0L=0x00;

    EA=1; //uruchomienie przerwan
    ET0=1; //uruchomienie przerwan licznika 0
    IE0=1; //externall INT0 flag(autocleared on vector to ISR)
    //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
    TMOD=0x01; //0101  16-Bit Timer/Counter
    //Timer okreslaczestotliwosc
    TL0=N&0xFF;
    TH0=N>>8;
    TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)

  while(1);
}
```

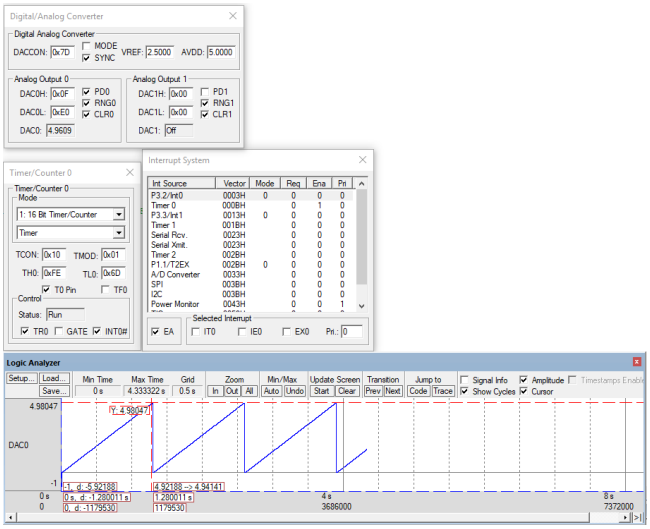
3.2 wygeneruj sygnał piłokształtny o parametrach podanych przez prowadzącego - amplituda oraz okres:

```
#include "aduc831.h" //Definitions of ADuC831 registers name
#include "stdint.h" //Standard integers
#include "stdfloat.h" //Standard floatt
#include "IO.h"  //Input/output definitions
#include <math.h>
#define PRESCALER 12 //dzielnik

#define CLOCK 11058000 //czestotliwosc

#define TIME 2000 // czas w milisekundach
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
//(wartosc licznika)

#define DAC_UREF 5.0
#define DAC_RES 4095.0
#define DAC_V2R(voltage) (voltage/DAC_UREF)DAC_RES

uint16_t v=0;
```

```
1018 uint16_t i=0;
1019
1020 uint16_t r;
     float32_t vv=1.5;
1022
     void counter_timer_0 (void) interrupt 1
1024 {
         r = DAC_V2R(vv);
1026     TF0=0; //flaga przepelnienia licznika t0 kasujemy
         //Set Timer 0 high byte and low byte.
1028     TL0=N&0xFF; //lows bytes of N
         TH0=N>>8;
1030
         i++;
1032     if(i==5) //okres probkowania ms
         {
1034         v+=13; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
             i=0;
1036     }
         if(v>r) //1228 napiecie maksymalne
1038     {

1040      v=0;

1042     }

1044     // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
         DAC0H=v/256;
1046     DAC0L=(v-256(DAC0H>>8));//bo rejestry 8bit//kasowanie
         //lub DAC0L=voltage-(0x0F & DAC0H);
1048 }

1050 void main(void)
     {
1052
         //control registerr dac 12bit/8bit voltage 0 to Vref
1054     //0.Set to    1    = Power-On DAC0.
         //1.Set to    0    = Power-Off DAC1.,
1056     //2.DAC0/1 Update Synchronization Bit. When set to    1    the DAC outputs
         //update as soon as DACxL SFRs are written.
1058     //3.DAC0 Clear Bit. Set to    1    = DAC1 Output Normal.
         //4.DAC1 Clear Bit. Set to    1    = DAC1 Output Normal.
1060     //5.Set to    1    = DAC0 Range 0 V  DD.
         //6.Set to    1    = DAC1 Range 0 V  DD .
1062     //7.Set to    0   = 12-Bit Mode.

1064     DACCON=0x7D; //0111 1101

1066     //dataa registers
         DAC0H=0x00;
1068     DAC0L=0x00;

1070     EA=1; //uruchomienie przerwan
         ET0=1; //uruchomienie przerwan licznika 0
1072     IE0=1; //externall INT0 flag(autocleared on vector to ISR)
         //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1074     TMOD=0x1; //0101 16-Bit Timer/Counter

1076     //Timer okreslaczestotliwosc
         TL0=N&0xFF;
```

```
1078        TH0=N>>8;
            TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)
1080
            while(1);
1082
    }
```

3.3 Prezentacja wyników testów:

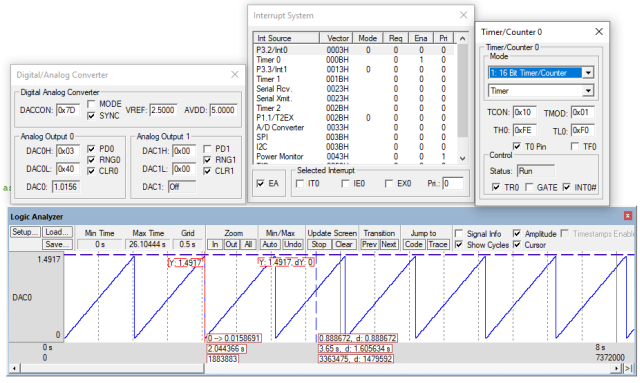


Rysunek 1: A

```
#define PRESCALER 12 //dzielnik
#define CLOCK 11058000 //czestotliwosc
#define TIME 1000 // czas w milisekundach
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME)))
```

```
#define DAC_UREF 5.0
#define DAC_RES 4095.0
#define DAC_V2R(voltage) voltage/DAC_UREF+DAC_RES
```

Rysunek 2: A



Rysunek 3: A

3.4 Wnioski:

Powyższy kod programu generuje sygnał piłokształtny o maksymalnej możliwej amplitudzie równej 4.98 V z napięciem zasilania układu równym 5 V (ustawionym w przytoczonym makrze $\mathrm{DAC}_U REF5.0)Nailustracji1, jestukazanymidzyinnymi`LogicAnalizerɔ, naktrymwidanaosirzdnychma$
$0x7D).Nastpniezerujemyrejestry DAC0HiDAC0L.Uruchamiamyprzerwaniaglobalnieiodtimera.Usta$
$0x01)wpracy16bitowej(65536˘wartoprzepenienia).NaRysunku2ukazanejestmakrowykorzystanedooblic$
$TL0 = N0xFF; przepisujemodszebitydorejestrudolnegozobliczonejwczeniej16bitowejwartociN, natom$
$N >> 8; ignorujemodszebityiprzepisujestarszedorejestrugrnego.Nastpnieuruchamiamytimeripojegop$
$r))pozwalanamzakoczywzrostzboczanarastajcegoiwrcidonapiciarwnego0.Stanchwilowysygnauzapisuje$
$v >> 8; zapisuje4starszebitydorejestrugrnego.DAC0L = (v-256*(DAC0H >> 8)); kiedyDAC0H =$
$0x0Fto >> 8(czylidzieleniepreprzez256)dajewynik0.Caaformuakasujenamrejestrdzikiczemuwpisujemy$

W kolejnych opisach będę pomijał elementy wspólne programów. W niniejszym podpunkcie, jak to widać na ilustracji 4 w części „Logic Analizer" amplituda sygnału została ustalona na wartości 1,5 V, natomiast okres T wynosi 1 Hz. W stosunku do poprzedniego programu zmianie uległy napięcie podawane do makra z rysunku 3 oraz skok jednego stopnia kwantyzacji.

4 Wygeneruj sygnał trójkątny:

4.1 używając dowolnego czasomierza wygeneruj sygnał trójkątny o maksymalnej amplitudzie, dowolnym okresie oraz jednakowym nachyleniu:

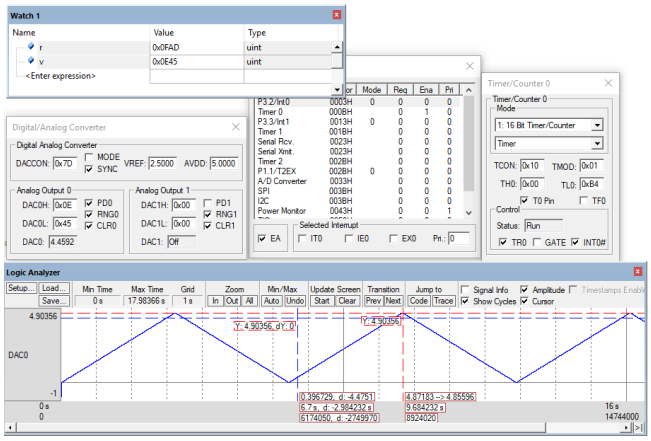
4.1.1 Prezentacja implementacji rozwiązania:

```
1000 /*
     a.  uzywajac dowolnego czasomierza wygeneruj sygnal pilokształtny o maksymalnej
         amplitudzie
1002 i dowolnym okresie, wyniki zarejestruj oscyloskopem,
     */
1004
     #include "aduc831.h"  //Definitions of ADuC831 registers name
1006 #include "stdint.h"   //Standard integers
     #include "stdfloat.h" //Standard floatt
1008 #include "IO.h"       //Input/output definitions
     #include <math.h>
1010
     #define PRESCALER 12 //dzielnik
1012 #define CLOCK 11058000 //czestotliwosc
     #define TIME 2000 // czas w milisekundach
1014 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
         (wartosc licznika)


1016
     #define DAC_UREF 5.0
1018 #define DAC_RES 4095.0
     #define DAC_V2R(voltage) (voltage/DAC_UREF)*DAC_RES
1020
     uint16_t v=0;
1022


1024 uint16_t r;
     float32_t vv=4.9;
1026 char i = 0;
```

```
char poz = 0;

void counter_timer_0 (void) interrupt 1
{
     r = DAC_V2R(vv);
     TF0=0; //flaga przepelnienia licznika t0 kasujemy
     //Set Timer 0 high byte and low byte.
     TL0=N&0xFF; //lows bytes of N
     TH0=N>>8;

     i++;
     if(poz==0)
     {
       if(i==5) //okres probkowania ms
       {
         v+=13; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
         i=0;
       }
     }

     if(poz==1)
     {
       if(i==5)
       {
         v-=13; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
         i=0;
       }
     }
     if(v>r) //1228 napiecie maksymalne
     {
       poz=1;
     }
     if(v<=0)
     {
       poz=0;
     }
     // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
     DAC0H=v/256;
     DAC0L=(v-256*(DAC0H>>8));//bo rejestry 8bit//kasowanie
     //lub DAC0L=voltage-(0x0F & DAC0H);
}

void main(void)
{
     //control registerr dac 12bit/8bit voltage 0 to Vref
     //0.Set to   1   = Power-On DAC0.
     //1.Set to   0   = Power-Off DAC1.,
     //2.DAC0/1 Update Synchronization Bit. When set to   1   the DAC outputs
    update as soon as DACxL SFRs are written.
     //3.DAC0 Clear Bit. Set to   1   = DAC1 Output Normal.
     //4.DAC1 Clear Bit. Set to   1   = DAC1 Output Normal.
     //5.Set to   1   = DAC0 Range 0 V  DD.
     //6.Set to   1   = DAC1 Range 0 V  DD .
     //7.Set to   0  = 12-Bit Mode.
     DACCON=0x7D; //0111 1101
     //dataa registers
     DAC0H=0x00;
     DAC0L=0x00;

     EA=1; //uruchomienie przerwan
```

```
1086      ET0=1; //uruchomienie przerwan licznika 0
          IE0=1; //externall INT0 flag(autocleared on vector to ISR)
1088      //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
          TMOD=0x1; //0101  16-Bit Timer/Counter
1090      //Timer okreslaczestotliwosc
          TL0=N&0xFF;
1092      TH0=N>>8;
          TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)
1094
      while(1);
1096  }
```

4.2 Prezentacja wyników testów:



Rysunek 4: A

4.3 wygeneruj sygnał trójkątny o parametrach podanych przez prowadzącego - amplituda oraz okres:

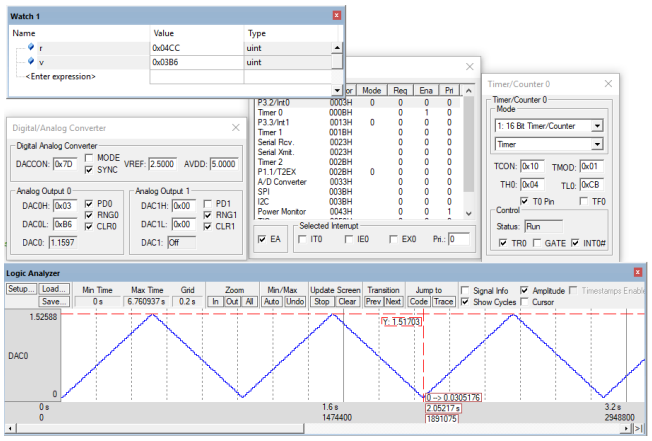
4.3.1 Prezentacja implementacji rozwiązania:

```
1000  /*
      a.  uzywajac dowolnego czasomierza wygeneruj sygnal pilokształtny o maksymalnej
          amplitudzie
1002  i dowolnym okresie, wyniki zarejestruj oscyloskopem,
      */
1004
      #include "aduc831.h"  //Definitions of ADuC831 registers name
1006  #include "stdint.h"   //Standard integers
      #include "stdfloat.h" //Standard floatt
1008  #include "IO.h"       //Input/output definitions
      #include <math.h>
1010
      #define PRESCALER 12 //dzielnik
1012  #define CLOCK 11058000 //czestotliwosc
      #define TIME 2500 // czas w milisekundach
1014  #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
          (wartosc licznika)
      //63535
```

```
1016
#define DAC_UREF 5.0
1018 #define DAC_RES 4095.0
#define DAC_V2R(voltage) (voltage/DAC_UREF)*DAC_RES
1020

1022 uint16_t v=0;

1024
uint16_t r;
1026 float32_t vv=1.5;
char i = 0;
1028 char poz = 0;

1030 void counter_timer_0 (void) interrupt 1
{
1032     r = DAC_V2R(vv);
     TF0=0; //flaga przepelnienia licznika t0 kasujemy
1034     //Set Timer 0 high byte and low byte.
     TL0=N&0xFF; //lows bytes of N
1036     TH0=N>>8;

1038     i++;
     if(poz==0)
1040     {
       if(i==5) //okres probkowania ms
1042       {
         v+=25; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1044         i=0;
       }
1046     }

1048     if(poz==1)
     {
1050       if(i==5)
       {
1052         v-=25; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
           i=0;
1054       }
     }
1056     if(v>r) //1228 napiecie maksymalne
     {
1058       poz=1;
     }
1060     if(v<=0)
     {
1062       poz=0;
     }
1064     // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
     DAC0H=v/256;
1066     DAC0L=(v-256*(DAC0H>>8));//bo rejestry 8bit//kasowanie
     //lub DAC0L=voltage-(0x0F & DAC0H);
1068 }

1070 void main(void)
{
1072     //control registerr dac 12bit/8bit voltage 0 to Vref
     //0.Set to   1   = Power-On DAC0.
1074     //1.Set to   0   = Power-Off DAC1.,
```

```
    //2.DAC0/1 Update Synchronization Bit. When set to 1 the DAC outputs
    update as soon as DACxL SFRs are written.
1076    //3.DAC0 Clear Bit. Set to 1 = DAC1 Output Normal.
    //4.DAC1 Clear Bit. Set to 1 = DAC1 Output Normal.
1078    //5.Set to 1 = DAC0 Range 0 V DD.
    //6.Set to 1 = DAC1 Range 0 V DD .
1080    //7.Set to 0 = 12-Bit Mode.
    DACCON=0x7D; //0111 1101
1082    //dataa registers
    DAC0H=0x00;
1084    DAC0L=0x00;

1086    EA=1; //uruchomienie przerwan
    ET0=1; //uruchomienie przerwan licznika 0
1088    IE0=1; //externall INT0 flag(autocleared on vector to ISR)
    //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1090    TMOD=0x1; //0101 16-Bit Timer/Counter
    //Timer okreslaczestotliwosc
1092    TL0=N&0xFF;
    TH0=N>>8;
1094    TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)

1096  while(1);
}
```

4.4 Prezentacja wyników testów:



Rysunek 5: A

4.4.1 Wnioski:

Jak widać na ilustracji 6 przyjęta amplituda wynosi 1.5 V natomiast częstotliwość 2 Hz. Częstotliwość zmieniamy poddając modyfikacji parametr TIME w makrze ustawiającym czas do przepełnienia timera.

4.5 wygeneruj sygnał trójkątny o parametrach podanych przez prowadzącego – amplituda, okres, czas trwania zbocza narastającego, czas trwania zbocza opadającego:

4.5.1 Prezentacja implementacji rozwiązania:

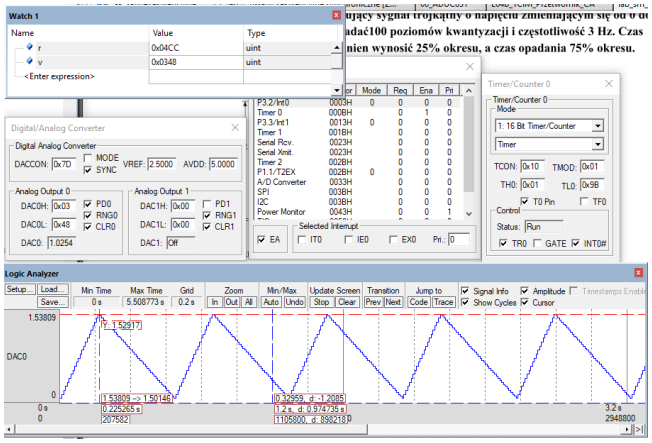
```
/*
a.  uzywajac dowolnego czasomierza wygeneruj sygnal pilokształtny o maksymalnej amplitudzie
i dowolnym okresie, wyniki zarejestruj oscyloskopem,
*/

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

#define PRESCALER 12 //dzielnik
#define CLOCK 11058000 //czestotliwosc
#define TIME 2500 // czas w milisekundach
#define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N (wartosc licznika)
//63535

#define DAC_UREF 5.0
#define DAC_RES 4095.0
#define DAC_V2R(voltage) (voltage/DAC_UREF)*DAC_RES


uint16_t v=0;


uint16_t r;
float32_t vv=1.5;
char i = 0;
char poz = 0;

void counter_timer_0 (void) interrupt 1
{
    r = DAC_V2R(vv);
    TF0=0; //flaga przepelnienia licznika t0 kasujemy
    //Set Timer 0 high byte and low byte.
    TL0=N&0xFF; //lows bytes of N
    TH0=N>>8;

    i++;
    if(poz==0)
    {
      if(i==5) //okres probkowania ms
      {
        v+=60; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
        i=0;
      }
    }

    if(poz==1)
    {
      if(i==5)
      {
```

```
1052            v-=30; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
                i=0;
1054          }
        }
1056    if(v>r) //1228 napiecie maksymalne
        {
1058      poz=1;
        }
1060    if(v<=0)
        {
1062      poz=0;
        }
1064    // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
        DAC0H=v/256;
1066    DAC0L=(v-256*(DAC0H>>8));//bo rejestry 8bit//kasowanie
        //lub DAC0L=voltage-(0x0F & DAC0H);
1068 }

1070 void main(void)
    {
1072    //control registerr dac 12bit/8bit voltage 0 to Vref
        //0.Set to   1   = Power-On DAC0.
1074    //1.Set to   0   = Power-Off DAC1.,
        //2.DAC0/1 Update Synchronization Bit. When set to   1   the DAC outputs
       update as soon as DACxL SFRs are written.
1076    //3.DAC0 Clear Bit. Set to   1   = DAC1 Output Normal.
        //4.DAC1 Clear Bit. Set to   1   = DAC1 Output Normal.
1078    //5.Set to   1   = DAC0 Range 0 V  DD.
        //6.Set to   1   = DAC1 Range 0 V  DD .
1080    //7.Set to   0  = 12-Bit Mode.
        DACCON=0x7D; //0111 1101
1082    //dataa registers
        DAC0H=0x00;
1084    DAC0L=0x00;

1086    EA=1; //uruchomienie przerwan
        ET0=1; //uruchomienie przerwan licznika 0
1088    IE0=1; //externall INT0 flag(autocleared on vector to ISR)
        //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1090    TMOD=0x1; //0101  16-Bit Timer/Counter
        //Timer okreslaczestotliwosc
1092    TL0=N&0xFF;
        TH0=N>>8;
1094    TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)

1096   while(1);
    }
```

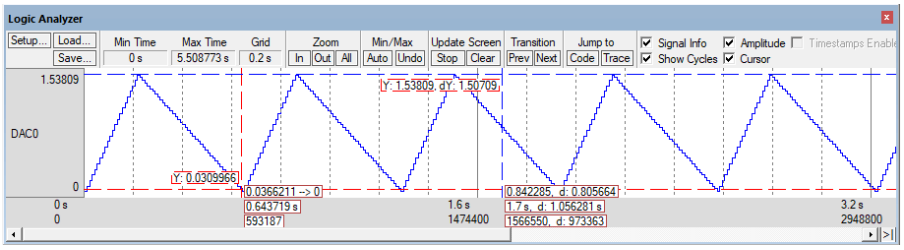
4.6 Prezentacja wyników testów:

4.6.1 Wnioski:

W tym przypadku celem programu było otrzymanie rożnych czasów trwania zbocza narastającego i opadającego. Analizując Ilustracje 7 oraz 8 dochodzimy do wniosku, że czas narastania zbocza pierwszego wynosi t=0.22 s a opadania drugiego t =0.42 s. Efekt taki uzyskana zmieniając wysokość stopnia kwantyzacji co powoduje dłuższą lub szybszą w czasie zmianę amplitudy sygnału.



Rysunek 6: A



Rysunek 7: A

5 Napisz interfejs użytkownika do ustawiania parametrów sygnału trójkątnego:

6 Rozszerz interfejs: