

Spis treści

1	Analiza działania przetwornika cyfrowo-analogowego:	2
2	„Ręczne” ustawienie DACx:	2
3	Wygeneruj sygnał piłokształtny:	2
4	Wygeneruj sygnał trójkątny:	6
5	Napisz interfejs użytkownika do ustawiania parametrów sygnału trójkątnego:	13
6	Rozszerz interfejs:	13

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 /*
1001  a.  uzywajac dowolnego czasomierza wygeneruj sygnał piłokształtny o maksymalnej
1002      amplitudzie
1003  i dowolnym okresie, wyniki zarejestruj oscyloskopem,
1004  */
1005
1006 #include "aduc831.h" //Definitions of ADuC831 registers name
1007 #include "stdint.h" //Standard integers
1008 #include "stdfloat.h" //Standard floatt
1009 #include "IO.h" //Input/output definitions
1010 #include <math.h>
1011
1012 #define PRESCALER 12 //dzielnik
1013 #define CLOCK 11058000 //czestotliwosc
1014 #define TIME 1000 // czas w milisekundach
1015 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
1016 // (wartosc licznika) //FFDF?//FFEB
1017
1018 #define DAC_UREF 5.0
1019 #define DAC_RES 4095.0
1020 #define DAC_V2R(voltage) voltage/DAC_UREF+DAC_RES
1021
1022 uint16_t v=0;
1023 uint16_t i=0;
1024 uint16_t counter_timer_0_initial_value;
1025 uint16_t r;
1026
1027 void counter_timer_0 (void) interrupt 1
1028 {
1029     TF0=0; //flaga przepelnienia licznika t0 kasujemy
1030     //Set Timer 0 high byte and low byte.
1031     TL0=N&0xFF; //lows bytes of N
1032     TH0=N>>8;
1033     r = DAC_V2R(6) ;
1034     i++;
1035     if (i==5) //okres probkowania ms
1036     {
1037         v+=16; //(napiecie maksymalne / 16) = ilosc stopni kwantyzacji na 1/2T =
1038         1000
1039         i=0;
1040     }
1041
1042     if (v>r) //1638 napiecie maksymalne
1043     {
1044         v=0;
1045     }
1046 }
```

```

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

1050 void main(void)
{
1052 //control registerr dac 12bit/8bit voltage 0 to Vref
//0.Set to 1 = Power-On DAC0.
1054 //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.
1056 //3.DAC0 Clear Bit. Set to 1 = DAC1 Output Normal.
//4.DAC1 Clear Bit. Set to 1 = DAC1 Output Normal.
1058 //5.Set to 1 = DAC0 Range 0 V DD.
//6.Set to 1 = DAC1 Range 0 V DD .
1060 //7.Set to 0 = 12-Bit Mode.
DACCON=0x7D; //0111 1101
1062 //dataa registers
DAC0H=0x00;
1064 DAC0L=0x00;

1066 EA=1; //uruchomienie przerwan
ET0=1; //uruchomienie przerwan licznika 0
1068 IE0=1; //externall INTO flag(autoleared on vector to ISR)
//16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1070 TMOD=0x01; //0101 16-Bit Timer/Counter
//Timer okreslaczestotliwosc
1072 TL0=N&0xFF;
TH0=N>>8;
1074 TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)

1076 while(1);
}

```

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

```

1000 #include "aduc831.h" //Definitions of ADu831 registers name
#include "stdint.h" //Standard integers
1002 #include "stdint.h" //Standard floatt
#include "IO.h" //Input/output definitions
1004 #include <math.h>
#define PRESCALER 12 //dzielnik
1006
#define CLOCK 11058000 //czestotliwosc
1008
#define TIME 2000 // czas w milisekundach
1010 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
//(wartosc licznika)
1012
#define DAC_UREF 5.0
1014 #define DAC_RES 4095.0
#define DAC_V2R(voltage) (voltage/DAC_UREF)DAC_RES
1016
uint16_t v=0;

```

```

1018 uint16_t i=0;

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 INTO flag(autoleared 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;

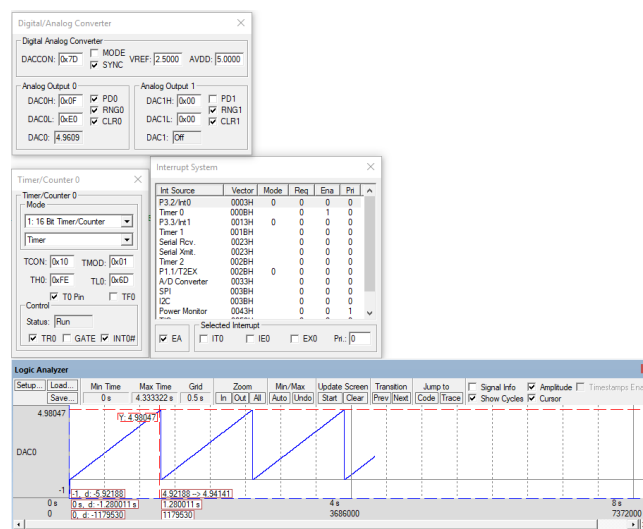
```

```

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

```

3.3 Prezentacja wyników testów:



Rysunek 1: A

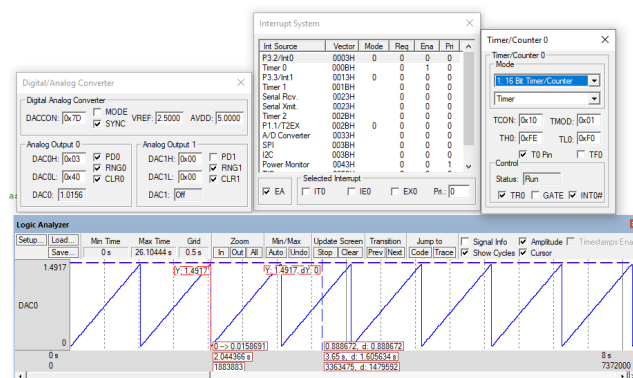
```

#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 pilokształtny o maksymalnej możliwej amplitudzie równej 4.98 V z napięciem zasilania układu równym 5 V (ustawionym w przytoczonym makrze `DAC_UREF5.0`). Na ilustracji 1, jest ukazany między innymi `Logic Analyzer`, na którym widana jest zmienna `0x7D`). Następnie zerujemy rejestry `DAC0H` i `DAC0L`. Uruchamiamy przerwanie globalnie i od timer'a. Ustawiamy `0x01` w pracy 16-bitowej (65536 wartości przepełnienia). Na rysunku 2 ukazane jest makro wykorzystane do obliczenia `TL0 = N * 0xFF`; przepisujemy do rejestru `0x00` obliczonej wartości `N`, natomiast `N >> 8`; ignorujemy do rejestru `0x01` starsze bity rejestru `0x00`. Następnie uruchamiamy timer i po jego przerwie (`r`) pozwalamy zakończyć wzrost bocznej części i wracamy do początku. Stachwilowy sygnał zapisujemy w `v >> 8`; zapisujemy 4 starsze bity rejestru `0x01`. `DAC0L = (v - 256 * (DAC0H >> 8))`; kiedy `DAC0H = 0x0F` to `>> 8` (czyli dzielenie przez 256) daje wynik 0. Cała formuła kasuje nam rejestr dziesiętny i zapisujemy

W kolejnych opisach będę pomijał elementy wspólne programów. W niniejszym podpunkcie, jak to widać na ilustracji 4 w części „Logic Analyzer” 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 /*
1001  a.  używając dowolnego czasomierza wygeneruj sygnał pilokształtny o maksymalnej
1002      amplitudzie
1003  i  dowolnym okresie, wyniki zarejestruj oscyloskopem,
1004  */
1005
1006 #include "aduc831.h" //Definitions of ADuC831 registers name
1007 #include "stdint.h"  //Standard integers
1008 #include "stdfloat.h" //Standard float
1009 #include "IO.h"       //Input/output definitions
1010 #include <math.h>
1011
1012 #define PRESCALER 12 //dzielnik
1013 #define CLOCK 11058000 //czestotliwosc
1014 #define TIME 2000 // czas w milisekundach
1015 #define N (0xFFFF - (CLOCK / (PRESCALER * TIME))) // obliczenie impuls w // zmienna N
1016      (wartosc licznika)
1017
1018 #define DAC_UREF 5.0
1019 #define DAC_RES 4095.0
1020 #define DAC_V2R(voltage) (voltage / DAC_UREF) * DAC_RES
1021
1022 uint16_t v = 0;
1023
1024 uint16_t r;
1025 float32_t vv = 4.9;
1026 char i = 0;
```

```

1028 char poz = 0;
1030 void counter_timer_0 (void) interrupt 1
1032 {
1034     r = DAC_V2R(vv);
1036     TF0=0; //flaga przepelnienia licznika t0 kasujemy
1038     //Set Timer 0 high byte and low byte.
1040     TL0=N&0xFF; //lows bytes of N
1042     TH0=N>>8;
1044
1046     i++;
1048     if (poz==0)
1050     {
1052         if (i==5) //okres probkowania ms
1054         {
1056             v+=13; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1058             i=0;
1060         }
1062     }
1064
1066     if (poz==1)
1068     {
1070         if (i==5)
1072         {
1074             v-=13; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1076             i=0;
1078         }
1080     }
1082
1084     if (v>r) //1228 napiecie maksymalne
1086     {
1088         poz=1;
1090     }
1092     if (v<=0)
1094     {
1096         poz=0;
1098     }
1100
1102     // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
1104     DAC0H=v/256;
1106     DAC0L=(v-256*(DAC0H>>8)); //bo rejestry 8bit //kasowanie
1108     //lub DAC0L=voltage-(0x0F & DAC0H);
1110 }
1112 void main(void)
1114 {
1116     //control registerr dac 12bit/8bit voltage 0 to Vref
1118     //0.Set to 1 = Power-On DAC0.
1120     //1.Set to 0 = Power-Off DAC1. ,
1122     //2.DAC0/1 Update Synchronization Bit. When set to 1 the DAC outputs
1124     update as soon as DACxL SFRs are written.
1126     //3.DAC0 Clear Bit. Set to 1 = DAC1 Output Normal.
1128     //4.DAC1 Clear Bit. Set to 1 = DAC1 Output Normal.
1130     //5.Set to 1 = DAC0 Range 0 V DD.
1132     //6.Set to 1 = DAC1 Range 0 V DD .
1134     //7.Set to 0 = 12-Bit Mode.
1136     DACCON=0x7D; //0111 1101
1138     //dataa registers
1140     DAC0H=0x00;
1142     DAC0L=0x00;
1144
1146     EA=1; //uruchomienie przerwan

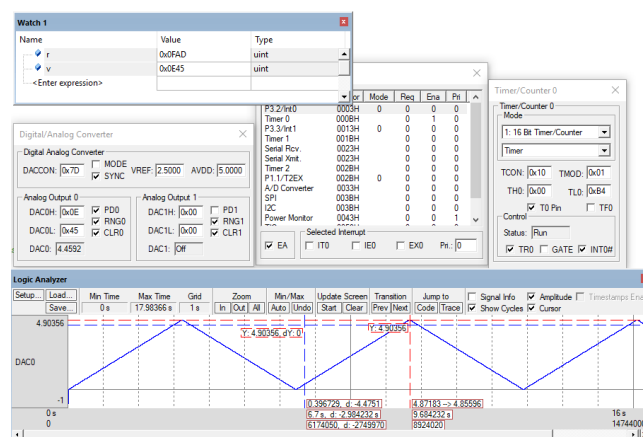
```

```

1086 ET0=1; //uruchomienie przerwan licznika 0
1087 IE0=1; //externall INTO flag(autoleared on vector to ISR)
1088 //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1089 TMOD=0x1; //0101 16-Bit Timer/Counter
1090 //Timer okreslaczestotliwosc
1091 TL0=N&0xFF;
1092 TH0=N>>8;
1093 TR0=1; //Set by user to turn on Timer/Counter 0.(default 0 tmod)
1094
1095 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 /*
1001 a.  uzywajac dowolnego czasomierza wygeneruj sygnał pilokształtny o maksymalnej
1002     amplitudzie
1003 i dowolnym okresie, wyniki zarejestruj oscyloskopem ,
1004 */
1005
1006 #include "aduc831.h" //Definitions of ADuC831 registers name
1007 #include "stdint.h"  //Standard integers
1008 #include "stdfloat.h" //Standard floatt
1009 #include "IO.h"       //Input/output definitions
1010 #include <math.h>
1011
1012 #define PRESCALER 12 //dzielnik
1013 #define CLOCK 11058000 //czestotliwosc
1014 #define TIME 2500 // czas w milisekundach
1015 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
1016 // (wartosc licznika)
1017 //63535

```



```

1016 #define DAC_UREF 5.0
1018 #define DAC_RES 4095.0
1020 #define DAC_V2R(voltage) (voltage/DAC_UREF)*DAC_RES
1022
1024 uint16_t v=0;
1026
1028 uint16_t r;
1030 float32_t vv=1.5;
1032 char i = 0;
1034 char poz = 0;
1036
1038 void counter_timer_0 (void) interrupt 1
1040 {
1042     r = DAC_V2R(vv);
1044     TF0=0; //flaga przepelnienia licznika t0 kasujemy
1046     //Set Timer 0 high byte and low byte.
1048     TL0=N&0xFF; //lows bytes of N
1050     TH0=N>>8;
1052
1054     i++;
1056     if (poz==0)
1058     {
1060         if (i==5) //okres probkowania ms
1062         {
1064             v+=25; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1066             i=0;
1068         }
1070     }
1072
1074     if (poz==1)
1076     {
1078         if (i==5)
1080         {
1082             v-=25; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1084             i=0;
1086         }
1088     }
1090
1092     if (v>r) //1228 napiecie maksymalne
1094     {
1096         poz=1;
1098     }
1100
1102     if (v<=0)
1104     {
1106         poz=0;
1108     }
1110
1112     // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
1114     DAC0H=v/256;
1116     DAC0L=(v-256*(DAC0H>>8)); //bo rejestry 8bit //kasowanie
1118     //lub DAC0L=voltage-(0x0F & DAC0H);
1120 }
1122
1124 void main(void)
1126 {
1128     //control registerr dac 12bit/8bit voltage 0 to Vref
1130     //0.Set to 1 = Power-On DAC0.
1132     //1.Set to 0 = Power-Off DAC1.,

```

```

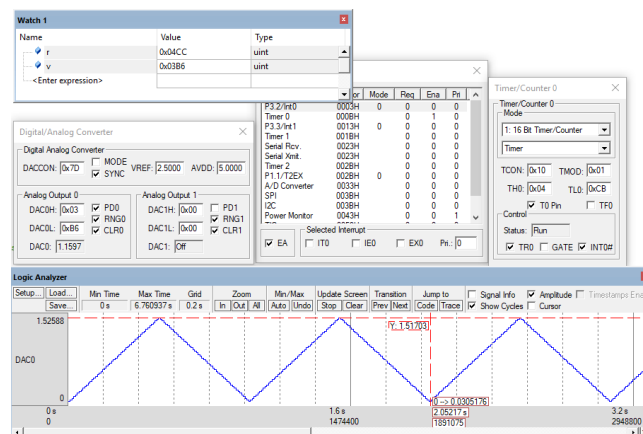
1076 //2.DAC0/1 Update Synchronization Bit. When set to 1 the DAC outputs
update as soon as DACxL SFRs are written.
1078 //3.DAC0 Clear Bit. Set to 1 = DAC1 Output Normal.
//4.DAC1 Clear Bit. Set to 1 = DAC1 Output Normal.
1080 //5.Set to 1 = DAC0 Range 0 V DD.
//6.Set to 1 = DAC1 Range 0 V DD .
1082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

1086 EA=1; //uruchomienie przerwan
ET0=1; //uruchomienie przerwan licznika 0
1088 IE0=1; //externall INTO 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.(default 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:

```
1000 /*
1001 a.  używając dowolnego czasomierza wygeneruj sygnał pilokształtny o maksymalnej
1002     amplitudzie
1003 i dowolnym okresie, wyniki zarejestruj oscyloskopem,
1004 */
1005 #include "aduc831.h" //Definitions of ADuC831 registers name
1006 #include "stdint.h"  //Standard integers
1007 #include "stdint.h"  //Standard floatt
1008 #include "IO.h"      //Input/output definitions
1009 #include <math.h>
1010
1011 #define PRESCALER 12 //dzielnik
1012 #define CLOCK 11058000 //czestotliwosc
1013 #define TIME 2500 // czas w milisekundach
1014 #define N (0xFFFF-(CLOCK/(PRESCALER*TIME))) // obliczenie impuls w //zmienna N
1015     (wartosc licznika)
1016     //63535
1017
1018 #define DAC_UREF 5.0
1019 #define DAC_RES 4095.0
1020 #define DAC_V2R(voltage) (voltage/DAC_UREF)*DAC_RES
1021
1022 uint16_t v=0;
1023
1024
1025 uint16_t r;
1026 float32_t vv=1.5;
1027 char i = 0;
1028 char poz = 0;
1029
1030 void counter_timer_0 (void) interrupt 1
1031 {
1032     r = DAC_V2R(vv);
1033     TF0=0; //flaga przepelnienia licznika t0 kasujemy
1034     //Set Timer 0 high byte and low byte.
1035     TL0=N&0xFF; //lows bytes of N
1036     TH0=N>>8;
1037
1038     i++;
1039     if (poz==0)
1040     {
1041         if (i==5) //okres probkowania ms
1042         {
1043             v+=60; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1044             i=0;
1045         }
1046     }
1047
1048     if (poz==1)
1049     {
1050         if (i==5)
1051         {
```

```

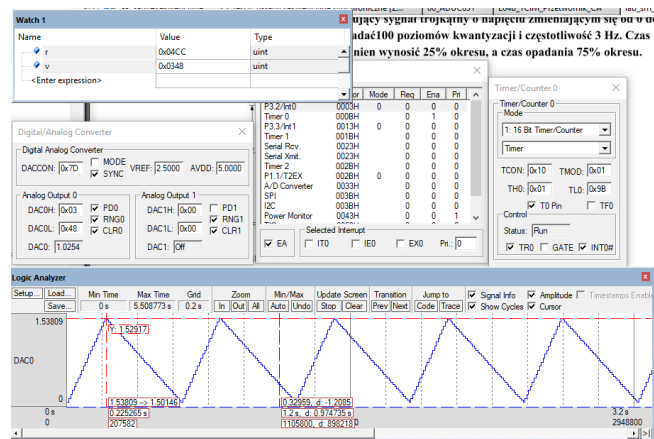
1052     v-=30; //(r / v) = ilosc stopni kwantyzacji na 1/2T = 100
1053     i=0;
1054 }
1055 }
1056 if (v>r) //1228 napiecie maksymalne
1057 {
1058     poz=1;
1059 }
1060 if (v<=0)
1061 {
1062     poz=0;
1063 }
1064 // dzielenie przez 256 to przesuniecie bitowe o 8 w prawo
1065 DAC0H=v/256;
1066 DAC0L=(v-256*(DAC0H>>8)); //bo rejestry 8bit //kasowanie
1067 //lub DAC0L=voltage-(0x0F & DAC0H);
1068 }
1069
1070 void main(void)
1071 {
1072     //control registerr dac 12bit/8bit voltage 0 to Vref
1073     //0.Set to 1 = Power-On DAC0.
1074     //1.Set to 0 = Power-Off DAC1.,
1075     //2.DAC0/1 Update Synchronization Bit. When set to 1 the DAC outputs
1076     //update as soon as DACxL SFRs are written.
1077     //3.DAC0 Clear Bit. Set to 1 = DAC1 Output Normal.
1078     //4.DAC1 Clear Bit. Set to 1 = DAC1 Output Normal.
1079     //5.Set to 1 = DAC0 Range 0 V DD.
1080     //6.Set to 1 = DAC1 Range 0 V DD .
1081     //7.Set to 0 = 12-Bit Mode.
1082     DACCON=0x7D; //0111 1101
1083     //dataa registers
1084     DAC0H=0x00;
1085     DAC0L=0x00;
1086
1087     EA=1; //uruchomienie przerwan
1088     ET0=1; //uruchomienie przerwan licznika 0
1089     IE0=1; //externall INTO flag(autoleared on vector to ISR)
1090     //16-Bit Timer/Counter. TH0 and TL0 are cascaded; there is no prescaler.
1091     TMOD=0x1; //0101 16-Bit Timer/Counter
1092     //Timer okreslaczestotliwosc
1093     TL0=N&0xFF;
1094     TH0=N>>8;
1095     TR0=1; //Set by user to turn on Timer/Counter 0.(defoult 0 tmod)
1096
1097     while(1);
1098 }

```

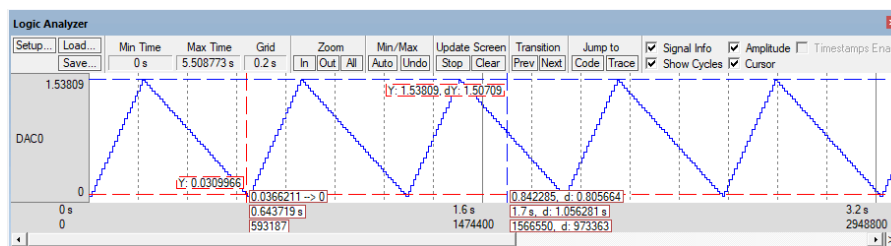
4.6 Prezentacja wyników testów:

4.6.1 Wnioski:

W tym przypadku celem programu było otrzymanie różnych 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: