Politechnika Gdańska
Wydział Elektroniki, Telekomunikacji i Informatyki

Zastosowanie Procesorów Sygnałowych

Ćwiczenie:
Stroik do gitary elektrycznej z wykorzystaniem ADSP-21161N.

Krzysztof Wiśniewski
Robert Piotrowski
Bolesław Wojtowicz

Gdańsk, 2006

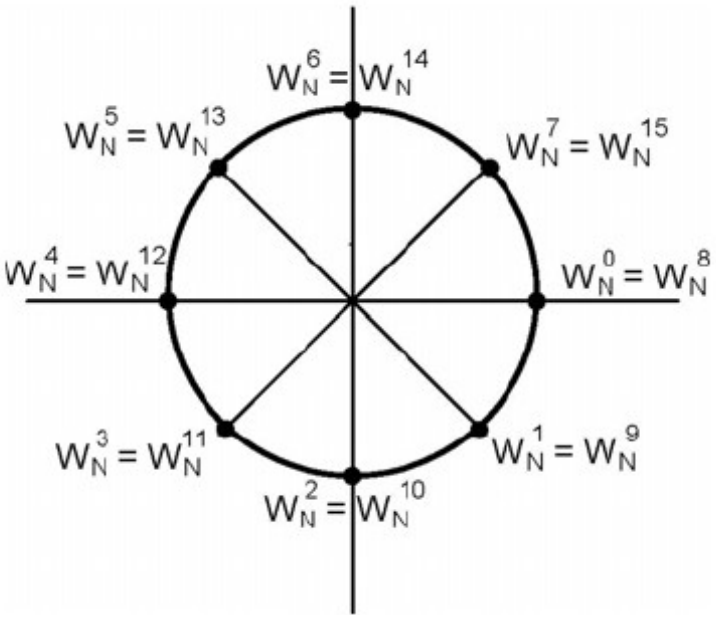
1. Wstęp.

Laboratorium te ma nauczyć Ciebie obsługi narzędzia „Analog Devices VisualDSP++”, programowania w Asemblerze/C++, oraz pokazanie Tobie przykładowych zastosowań układów DSP.

1.1 Dyskretna transformacja Fouriera.

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) W_N^{-kn} \text{ , k=0,1,2,…,N-1}$$

Obliczenie DFT z definicji wymaga mnożeń zespolonych i *N(N-1)* dodawań:

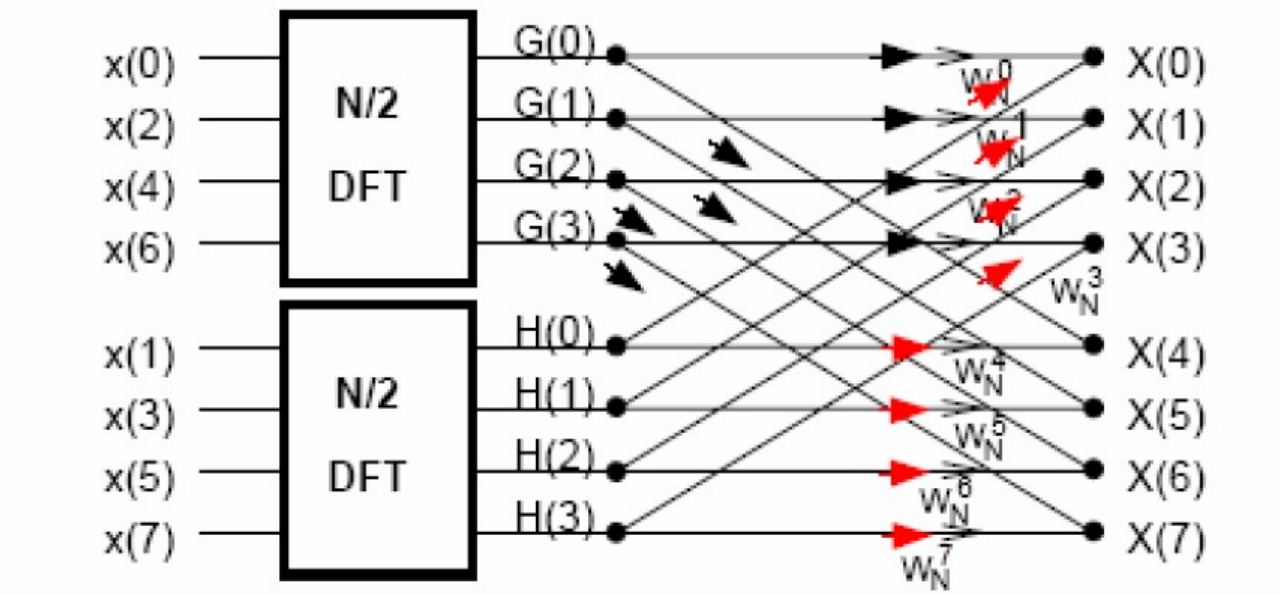


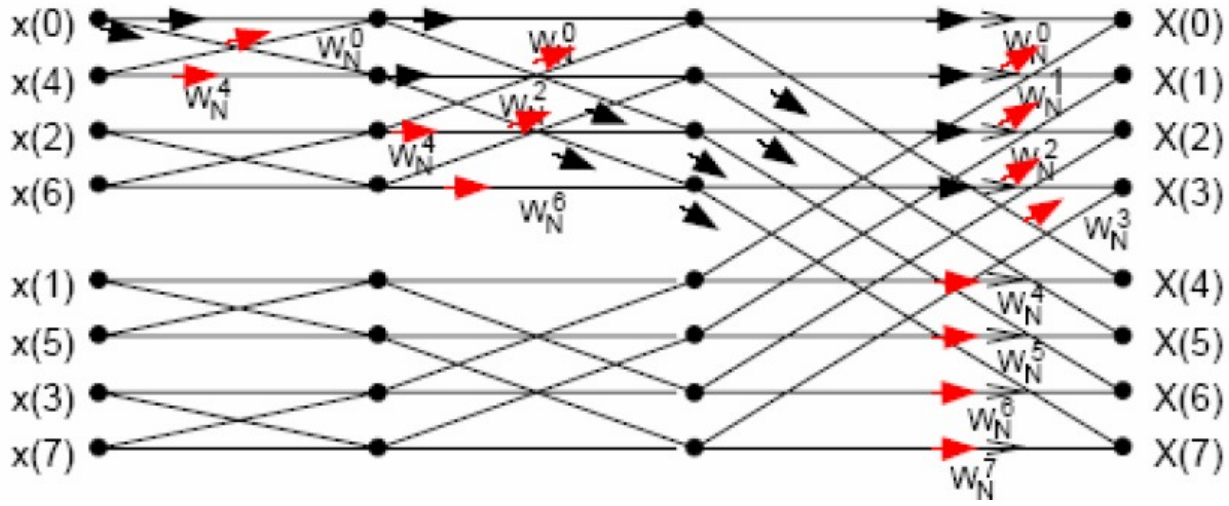
1.2 Szybka transformacja Fouriera (ang. Fast Fourier Transform - FFT)

Ponieważ DFT charakteryzuje się złożonością obliczeniowa rzędu $\mathbf{N^2}$, dzieląc sygnał na dwie części i transformując je osobno, musimy wykonać dwa razy po $\mathbf{(N/2)^2}$ operacji plus niewielką liczbę operacji potrzebnych na „skonsolidowanie” widm częściowych. Przykładowo dla *N*=1024, zamiast wykonywać mnożeń $\mathbf{N^2=1048676}$ wykonuje się $\mathbf{2*(N/2)^2=524288}$ mnożeń, czyli **dwa razy** mniej.
Oczywiście podziału zbioru próbek można dokonywać dalej –aż do uzyskania zbiorów dwuelementowych.
W dalszej części niniejszej instrukcji zostanie zaprezentowany algorytm radix-2 DIT (ang. *Decimation In Time*), ze względu na prostotę realizacji i powszechność stosowanego podejścia.
W algorytmie tym zakłada się, że transformowany sygnał posiada $\mathbf{N=2^p}$ próbek, a następnie:
1) dokonuje się zmiany kolejności próbek (realizowane jest to sprzętowo przez adresowanie z inwersją bitów), dzieląc je rekurencyjnie na próbki o indeksach parzystych i nieparzystych, aż do uzyskania zbiorów dwuelementowych;

2) wykonuje się serię **N/2** dwupunktowych przekształceń DFT;
3) następnie składa się widma dwukrążkowe (będące wynikiem przekształceń DFT z pkt.2) w widma czterokrążkowe. Procedurę powtarza się aż do momentu „złożenia" pełnego -N - punktowego widma.
Dla danego *N* algorytm FFT w wersji radix-2 DIT *(decymacja w czasie) wymaga wykonania $\mathbf{N\log_2 N}$ mnożeń i tyle samo dodawań zespolonych

1• *-szybsza wersja algorytmu redukuje liczbę mnożeń do połowy tej wielkości.

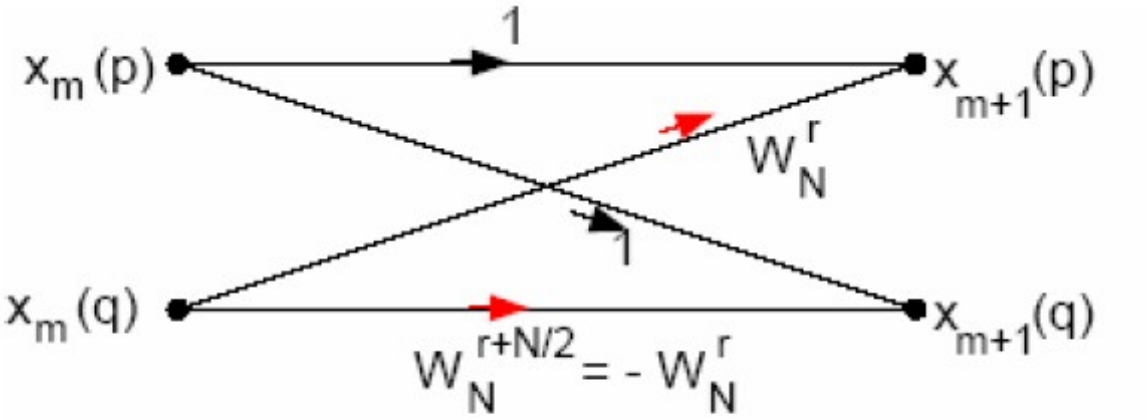


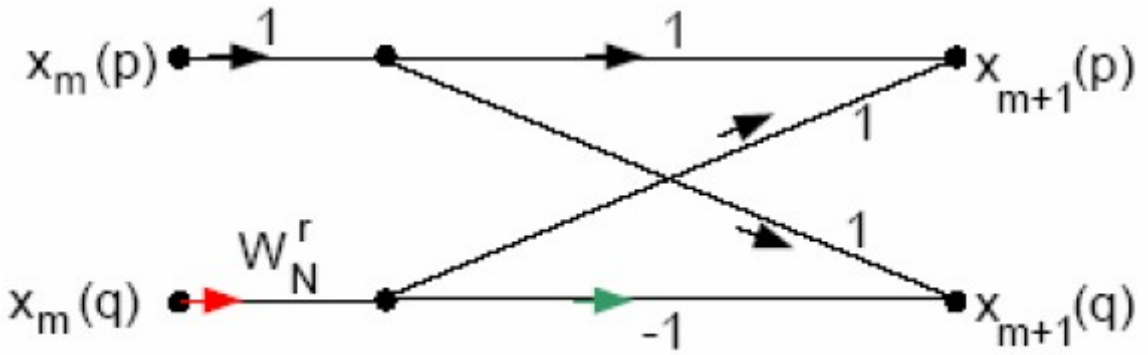


Próbki sygnału x(n) są uporządkowane według zasady inwersji bitów indeksu.

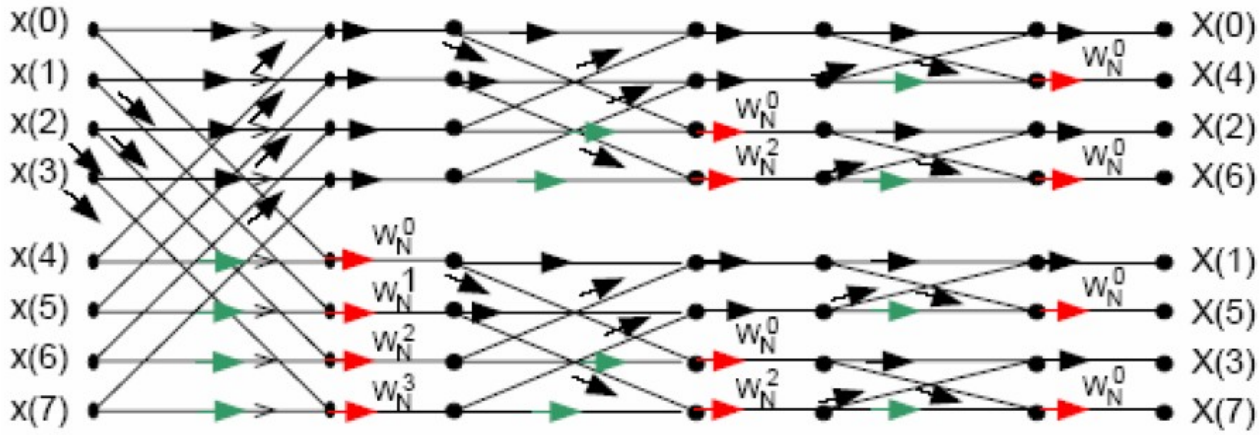
N	N^2	$N\log_2 N$	$N^2 / N\log_2 N$
8	64	24	2.67
64	4096	384	10.7
1024	1048576	10240	102.4

Liczbę mnożeń można jeszcze zmniejszyć zastępując operacje motylkowe:





W podobny sposób można opisać algorytm z podziałem częstotliwości (tutaj decymacja ma miejsce „po stronie” częstotliwości):



Przedstawiony powyżej algorytm radix-2 DIT nie jest jedynym szybkim algorytmem typu DIT pozwalającym na wyznaczenie DFT. W literaturze przedstawiono wiele wersji algorytmów FFT, w tym efektywniejsze od *radix-2* algorytmy *radix-4* (4 „części”), gdzie wykonuje się **3N/4** mnożenia zespolone, zamiast **N/2** mnożeń (*radix-2*),ale etapów jest **dwa razy mniej**.

2. Cel ćwiczenia.

Ćwiczenie te prezentuje przykładowe zastosowanie układu DSP. Jest nim stroik do gitary elektrycznej.

Po ukończeniu tego ćwiczenia nauczysz się:

- obsługi programu Analog Devices VisualDSP++;
- programowania płytki ADSP-21161N;
- wykorzystania FFT;

3. Opis programu.

Program główny znajduje się w pliku **„Main.c”**.

Działanie programu.

Na samym początku kompilator sprawdza co jest zdefiniowane i robi obliczenia dla poszczególnych definicji.

```
/*wybor przez uzytkownika trybu strojenia */
#define Stroik              //odznaczyc dla zgrubnego strojenia
//#define Stroik_Gitara     //odznaczyc dla dokladniejszego strojenia
```

#define Stroik – odznaczenie spowoduje, że zakres strojenia będzie od 0Hz do 2xczęstotliwość wybranej struny(powyżej tej częstotliwości będzie się świecić skrajna prawa dioda).
#define Stroik - odznaczenie spowoduje, że zakres strojenia będzie od -20 próbek do +20 próbek od częstotliwości wybranej struny. Powyżej +20 próbek jak i poniżej -20 próbek będą się świecić skrajne diody, kolejno prawa albo lewa.

```
#define dziel 16        //dzielnik czestotliwoci probkowania - ustawiamy w ten sposob software'owa czestotliwosc probkowania
#define z dziel*2048/48000
#define e1 329.63*z     //nr probki dla struny obiczany przez kompilator
#define h2 246.96*z
#define g3 196*z
#define d4 146.83*z
#define a5 110*z
#define e6 82.41*z
```

#define dziel 16 – zmieniając wartość **dziel** ustawiamy programowo częstotliwość próbkowania sygnału wejściowego.

Program zostaje załadowany do płytki ADSP-21161N.
Po wgraniu programu do płytki wyłączane są diody. Po tej operacji następuje sprawdzenie jaki przycisk został wciśnięty (w przypadku nie wciśnięcia żadnego z przycisków częstotliwością podstawową jest 440Hz). Każdemu przyciskowi (SW) została przypisana odpowiednia struna (częstotliwość drgania strun).

SW1 (Flag3) - E1 - 329,63 Hz -> 225 próbka (dla dziel=16)

SW2 (Flag2) - H2 - 246,96 Hz -> 168 próbka (dla dziel=16)

SW3 (Flag1) - G3 - 196 Hz -> 134 próbka (dla dziel=16)

SW4 (Flag0) - D4 - 146,83 Hz -> 100 próbka (dla dziel=16)

SW3 i SW4 - A5 - 110 Hz -> 75 próbka (dla dziel=16)

SW2 i SW3 - E6 - 82,41 Hz -> 56 próbka (dla dziel=16).

Numer próbki został wyliczony w następujący sposób:

$$\frac{f_U}{dziel} = f_{pr}$$

$$\frac{f_{real}}{l_{pr}} = wsp$$

$$\frac{f_{str}}{wsp} = nr_{pr}$$

gdzie: $\mathbf{f_U}$ – częst. próbkowania układu, $\mathbf{f_{pr}}$ – częst. programu(zaprogramowana częst), **dziel** – dzielnik częst., $\mathbf{l_{pr}}$ – ilość pobranych próbek, **wsp** – współczynnik ilości Hz na próbke, $\mathbf{f_{str}}$ – częst. drgania struny, $\mathbf{nr_{pr}}$ – numer próbki

Następnie pobierane są próbki z częstotliwością $\mathbf{f_{pr}}$. Po pobraniu 2048 próbek zostaje sprawdzone czy został nasycony przetwornik AC.

```
//sprawdzenie warunku nasycienia sie przetwornik
for (i=1;i<N;i++)
{   wart_max=real_input[i];
    wart_max1=real_input[i-1];
    if ((wart_max == wart_max1)&&(wart_max>=0.1||wart_max<=-0.1))
    {
        blad=1;
        Leds = 0x3F;
        *(int*) IOFLAG = Leds;
        break;   //kiedy wykryje nasycenie przetwornika zaswieca sie wszystkie diody
    }
```

W przypadku wykrycia dwóch sąsiadujących próbek o tych samych wartościach następuje zaświecenie się wszystkich diod (sygnalizacja błędu) i program ponownie zaczyna pobierać próbki z wejścia. Aby zobaczyć działanie podprogramu należy przełączyć przełącznik trzy-pozycyjny w prawo. Wtedy zaświecą się wszystkie diody. Po przełączeniu przełącznika w pozycję środkową program powinien powrócić do prawidłowego działania tzn. nie powinny się świecić wszystkie diody.

Jeżeli układ nie wykryje błędu to wykonane jest FFT – jest to FFT Radix-2 2048 punktowa (funkcja zaimplementowana w „Analog Devices VisualDSP++”). Po wykonaniu powyższej funkcji otrzymujemy tablice z wartościami Re i Im. Z tych macierzy wyliczany jest moduł.

Potem przeszukiwana jest tablicę z modułami w celu znalezienia wartości maksymalnej i jej odpowiadającej nr próbki.

```
rfft2048 (real_input, real_output, imag_output);
wart_max=0;
probka=0;

//obliczenie modułu z otrzymanych wartości
for (i=0;i<N;i++)
{
    real_2=(real_output[i])*(real_output[i]);
    imag_2=(imag_output[i])*(imag_output[i]);
    real_input[i]=sqrt(real_2+imag_2);
}

//wyszukanie wartości maksymalnej i nr próbki jej odpowiadającej
for (i=0;i<N;i++)
{
    if (real_input[i]>wart_max)
    {
        wart_max=real_input[i];
        probka=i;
    }
}
```

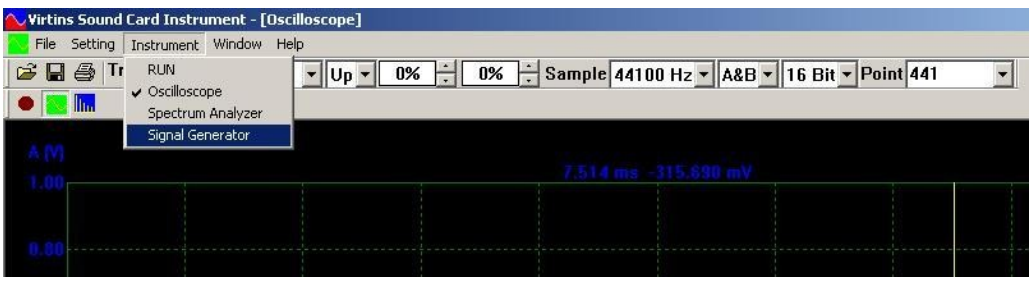
Następnie zostaje sprawdzana różnica numeru próbki wzorcowej (określonej dla każdej ze strun) a numeru próbki znalezionej w tablicy. W zależności od różnicy (różnicy częstotliwości) zostają włączone odpowiednie diody.

4. Przebieg laboratorium.

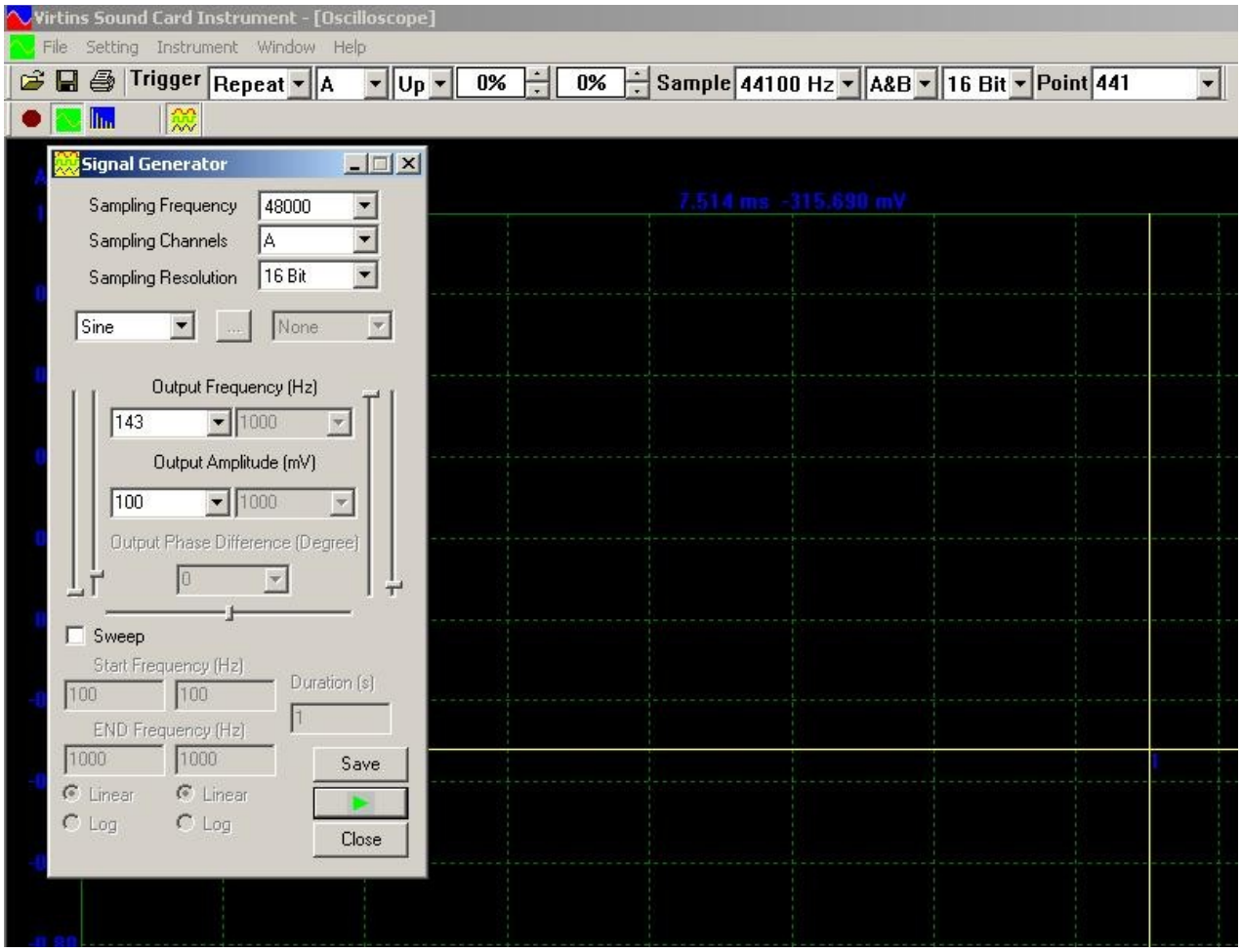
Należy podłączyć płytkę ADSP-21161N do komputera, uruchomić program „Analog Devices VisualDSP++” i załadować projekt „**laborka.dpj”**.

Po zapoznaniu się z plikiem **„Main.c”** należy połączyć kablem „mini_Jack” komputer (zielone gniazdo) i wejście „mic_in” znajdujące się na płytce ADSP-21161N.

Odpalamy program Virtins Sound Card Instrument (Scins) klikamy na Instrument i wybieramy Signal Generator:



ustawiamy jak na rysunku poniżej:



i klikamy na zielona strzałkę w celu uruchomienia generatora. Generowany sygnał to sinus o amplitudzie 100mV i częstotliwości 143Hz.

Odznaczamy linię **#define nie_z_pliku** aby wyglądała jak poniżej.

```
#define nie_z_pliku     //dla operacji wczytania próbek z plikow - zakomentowana linie
```

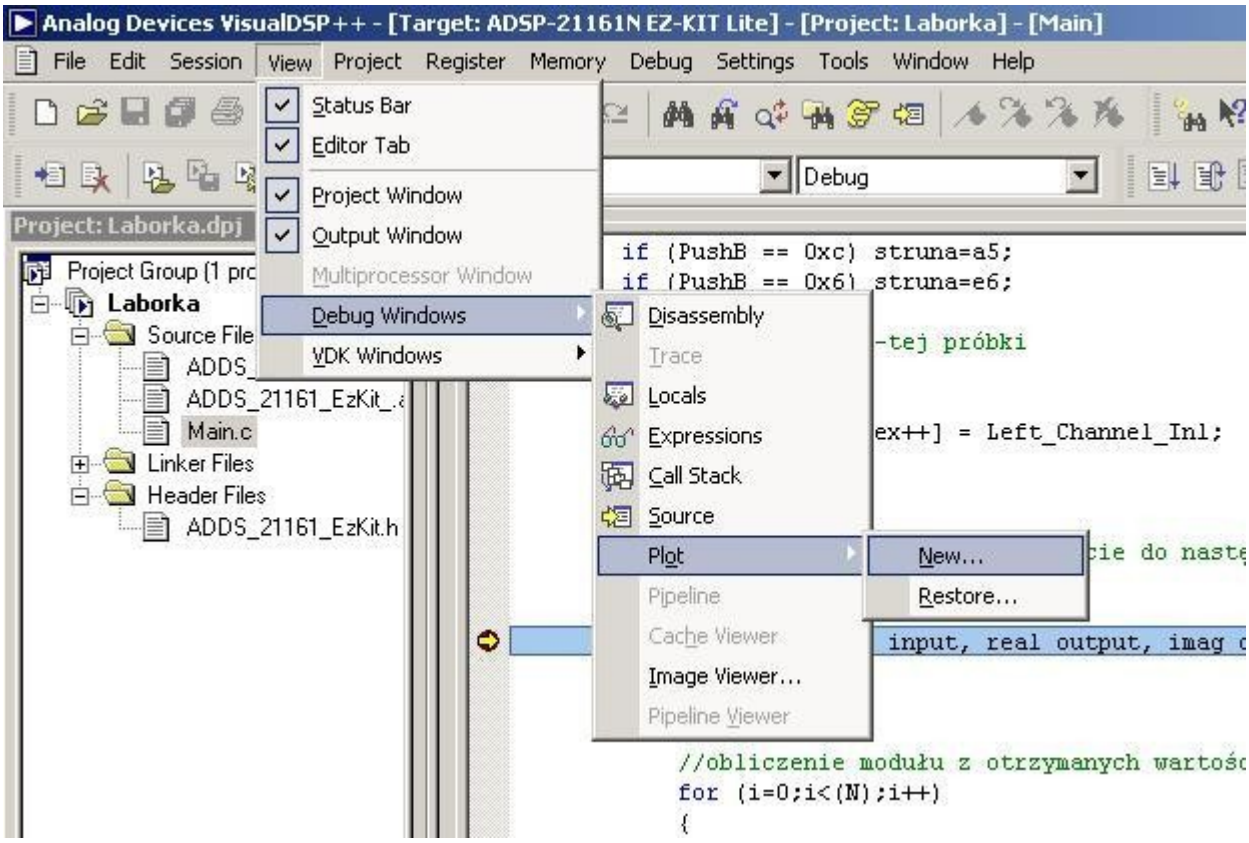
Przed kompilacją, w Main.c ustawiamy breakpointy, klikamy na:

- **Setup_ADSP21161N();** i naciskamy F9
- **rfft2048 (real_input, real_output, imag_output);** naciskamy F9
- **if (probka==struna)** naciskamy F9

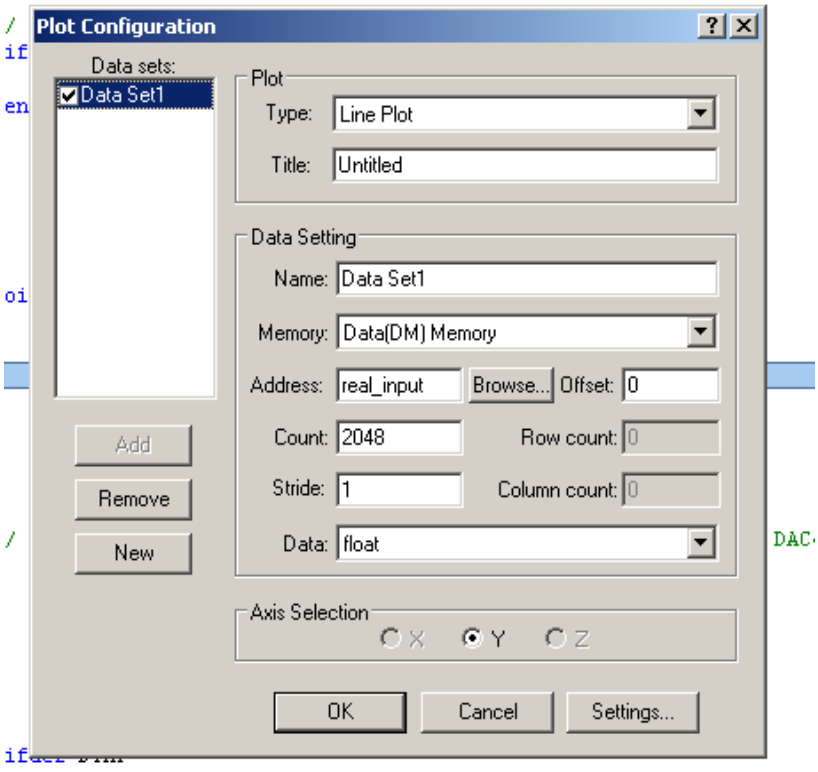
wciskamy Ctrl+F7 (Built File) po tym F7 (Built Project).

- podświetli nam sie pierwszy breakpoint **Setup_ADSP21161N()**,
- naciskamy przycisk LED_2(Flag0) i naciskamy F5(RUN),
- podświetli sie linia **rfft2048(....);**

Kursorem myszki najeżdżamy na View/Debbug Windows/Plot/New i klikamy.



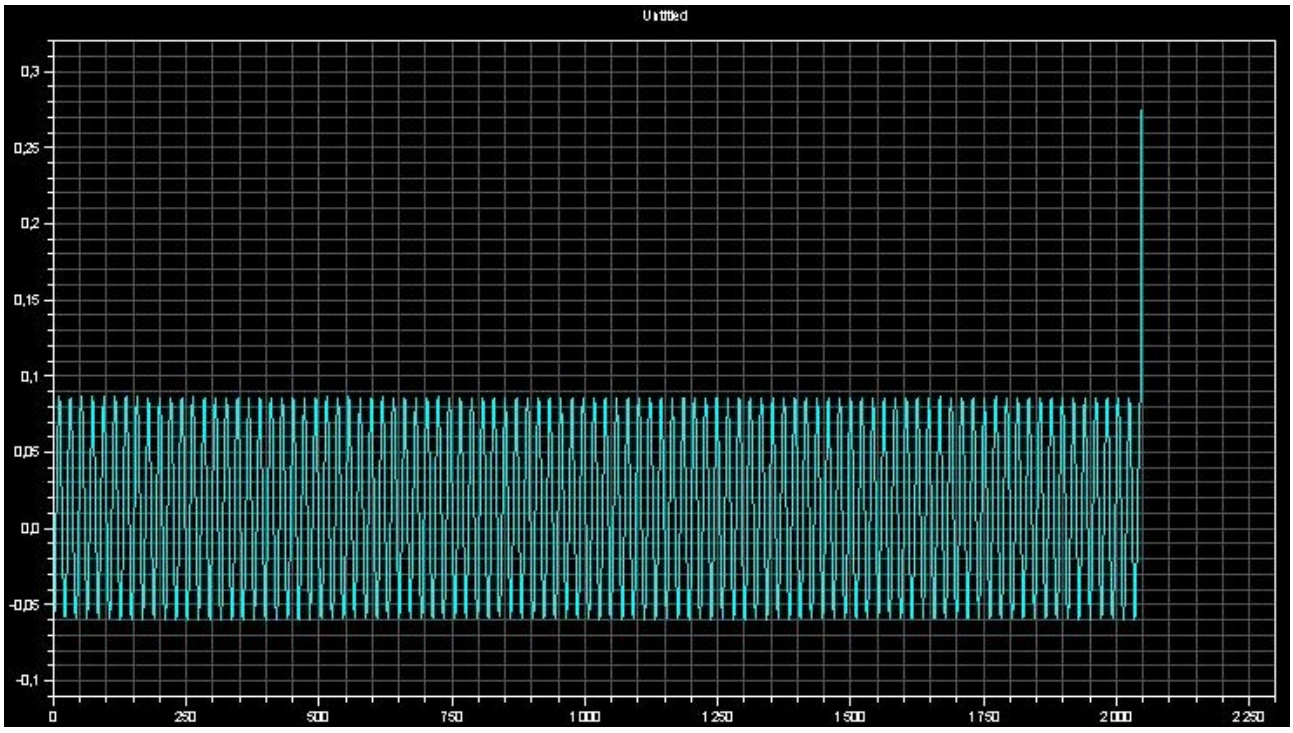
Otworzy nam się Plot Configuration:



kliknąć na Browse... i wybrać real_input w Count wpisać 2048 a Data ustawić float –

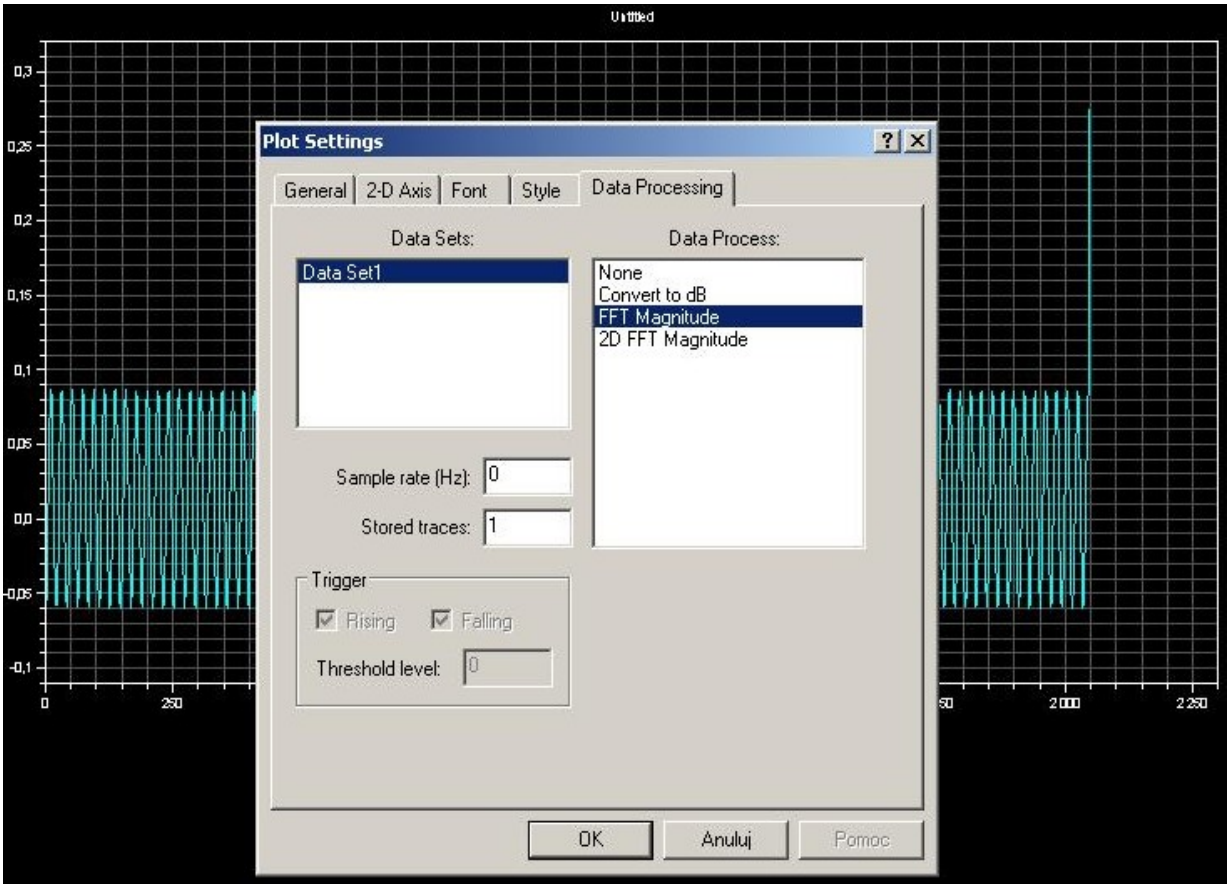
zatwierdzić OK.

Otrzymaliśmy przebieg sygnału wejściowego:



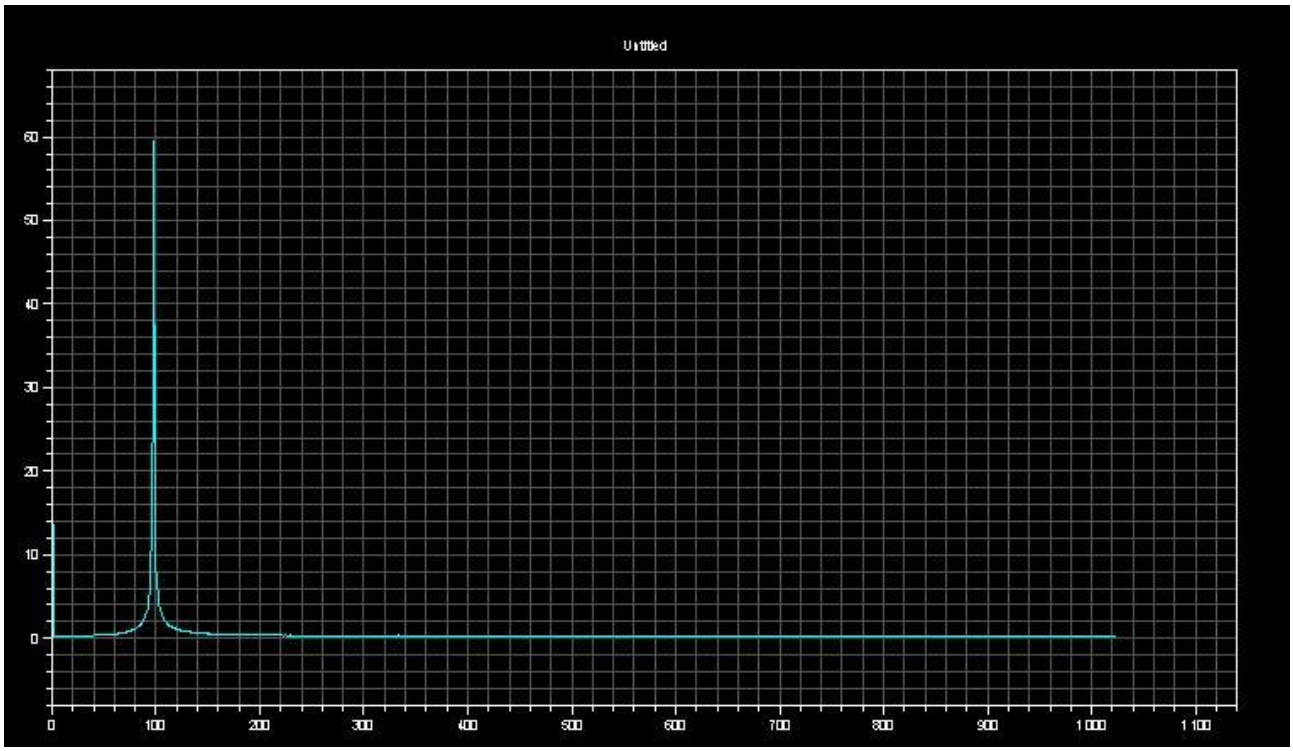
klikajac na otrzymanym wykresie prawym przyciskiem myszy wybieramy Modify Settings...

Klikamy na zakladke Data Processing i wybieramy FFT Magnitude (przyklad1_FFT)



Zatwierdzamy OK.

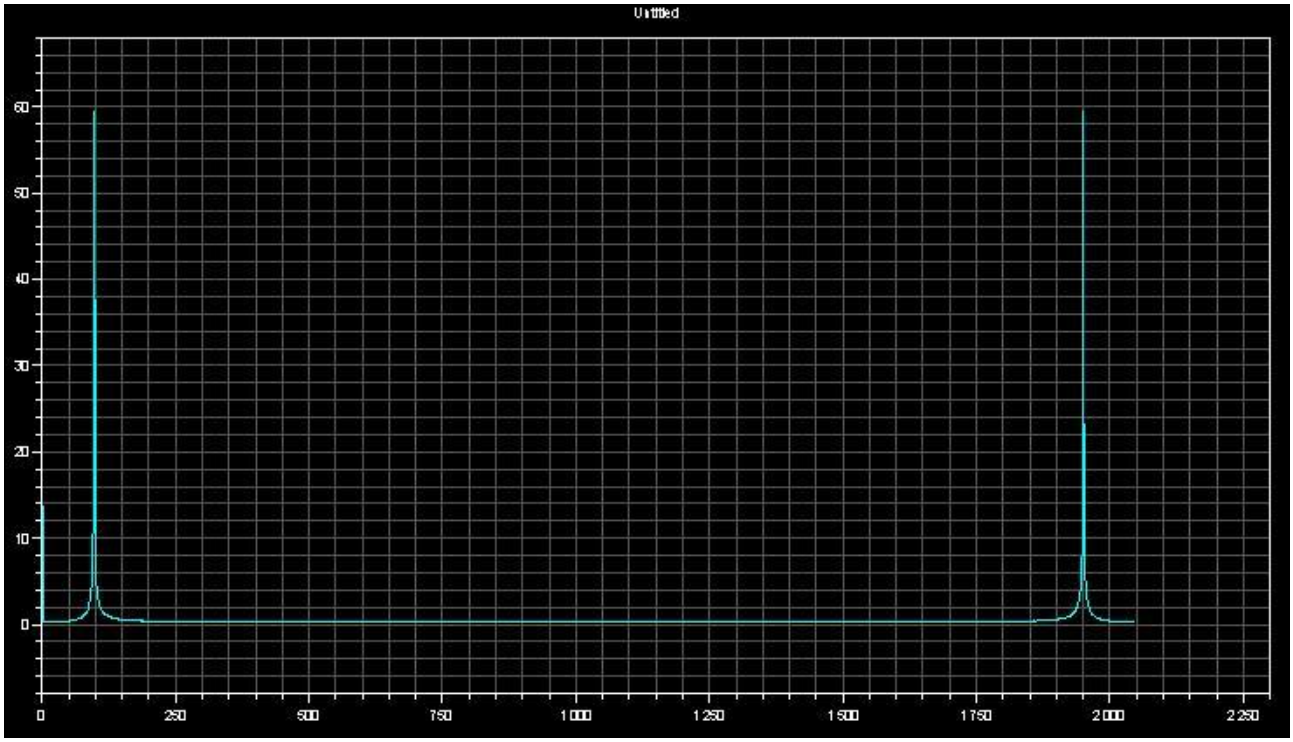
W ten sposób otrzymaliśmy wykres FFT sygnału wejściowego:



Następnie naciskamy na F5,podświetli nam sie ostatni breakpoint. Po tej akcji zostaje wyliczone FFT z funkcji.

Na okno Plot klikając na otrzymanym wykresie prawym przyciskiem myszy wybieramy

Modify Settings... Klikamy na zakładkę Data Processing i wybieramy None. Zatwierdzamy.

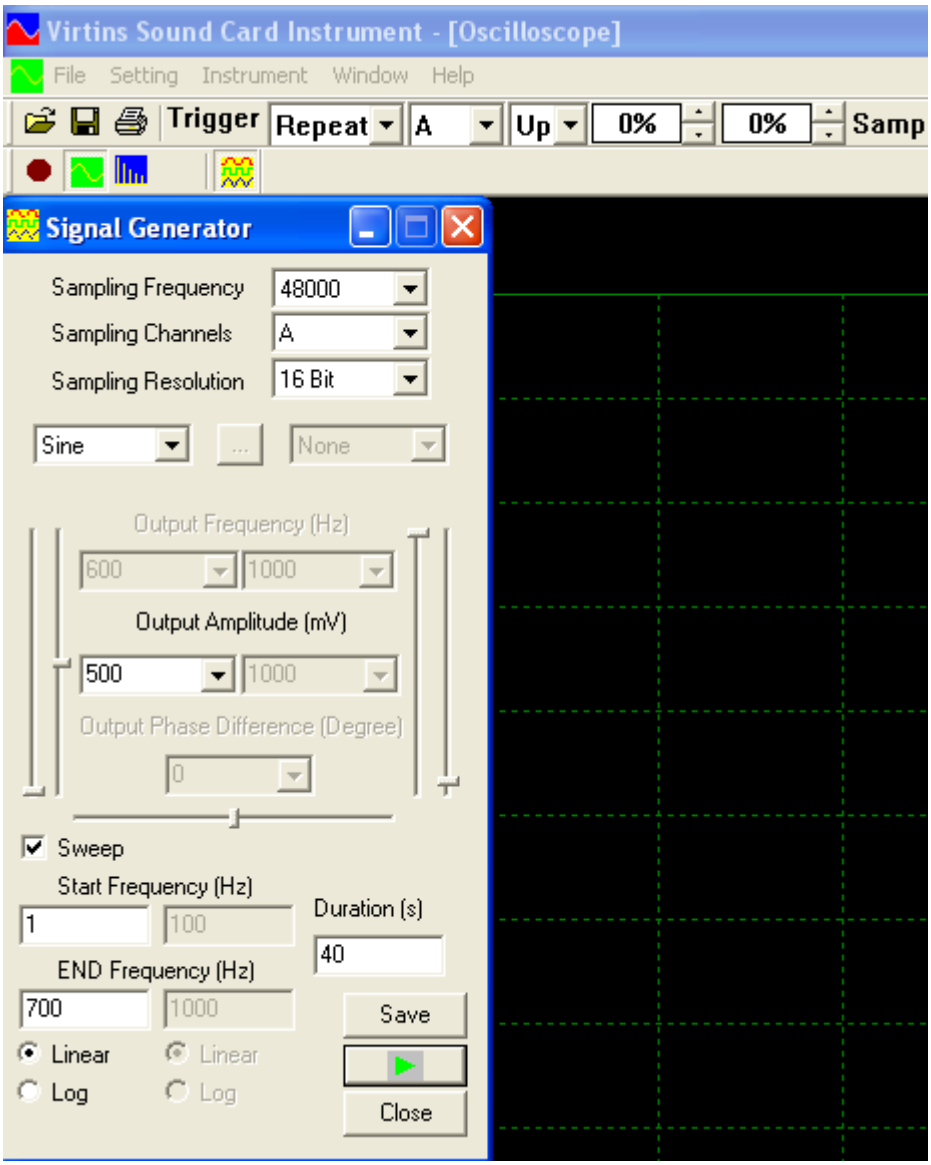


Przedstawiony rysunek obrazuje nam wynik działania funkcji FFT. Jak widać jest on taki sam jak rysunek otrzymany przez Analog Devices VisualDSP++.

Usunąć wszystkie Breakpoint'y (tak samo jak się je robiło) następnie wciskając F5

uruchamiamy program w trybie ciągłym.

Wcisnąć przycisk SW1 (Flag3), ustawić generator tak jak na rysunku poniżej:



i wcisnąć zieloną strzałkę. Zaobserwować zmiany świecenia diod.

Zmieniając częstotliwość w generatorze zaobserwować zmiany kolejności świecenia diod. Również wciskane przyciski zmieniają kolejności świecenia.

Ładownie próbek dźwięków gitary z plików.

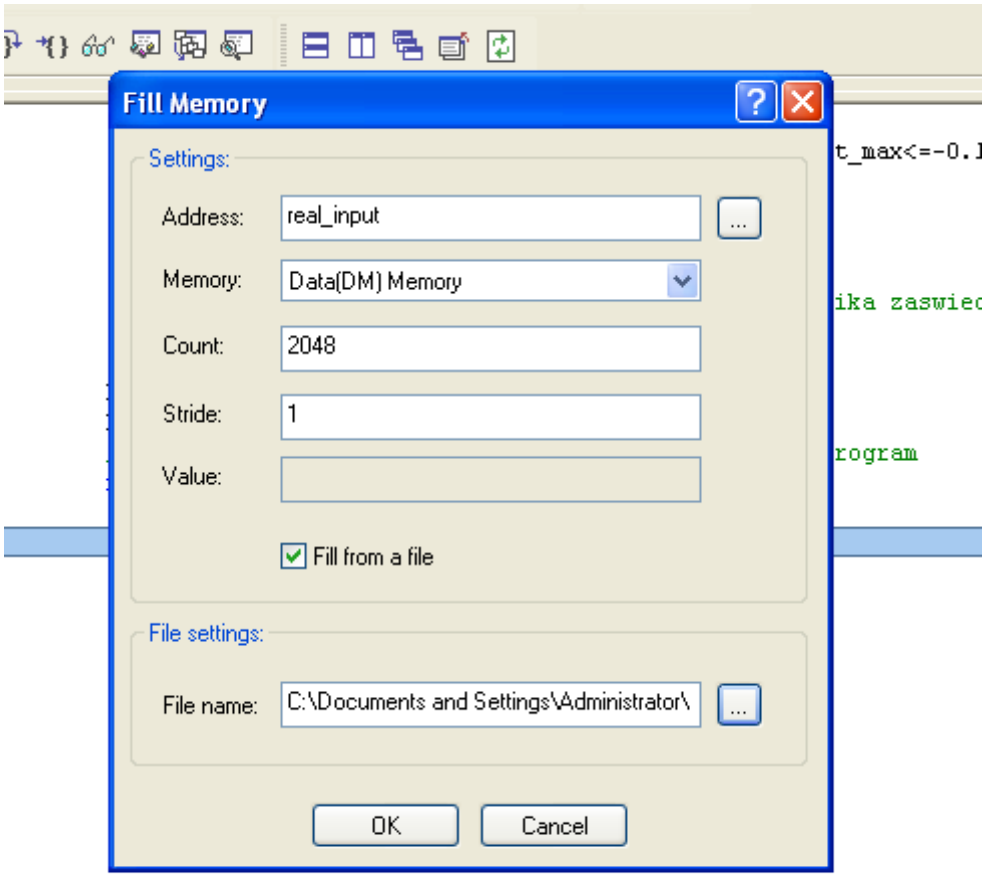
Należy:

- zrobić **Halt** (Alt+F5) i wcisnąć **Debug\Reset,**
- zakomentować linię **#define nie_z_pliku**

```
//#define nie_z_pliku       //dla operacji wczytania próbek z plikow - zakomentowana linie
```

- ustawić breakpoint'y w:
  - **rfft2048 (real_input, real_output, imag_output);**
  - **if (probka==struna)**
- Załadować program do płytki ADSP-21161N i uruchomić
- wciskać **F5** aż podświetli się linia **rfft2048 (real_input, real_output, imag_output);**

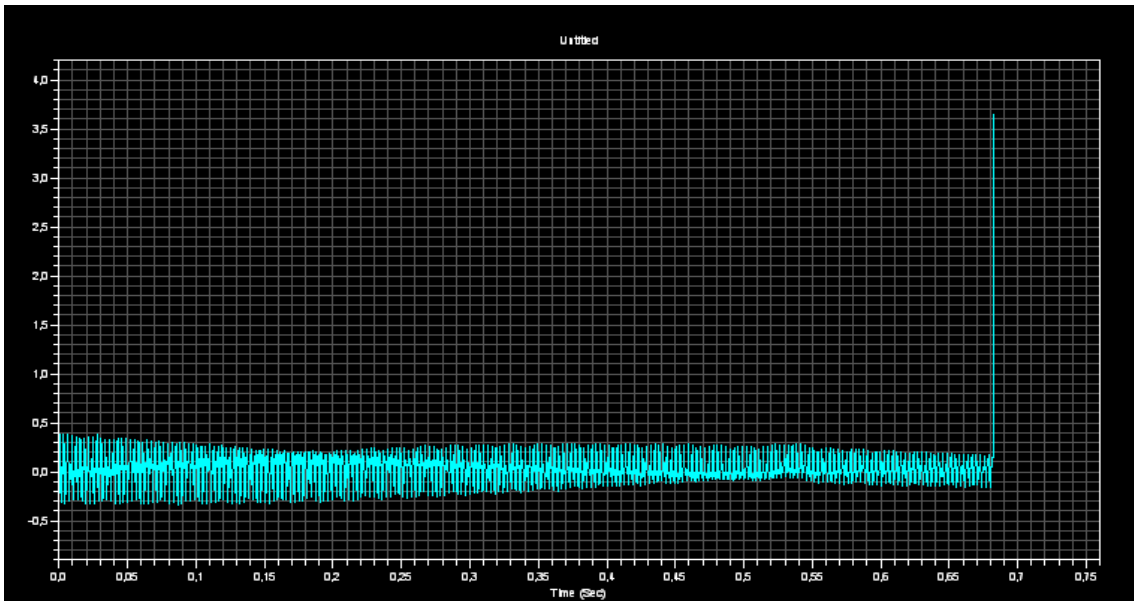
  - kliknąć na **Memory\1 Fill...** ustawić jak na rysunku poniżej:



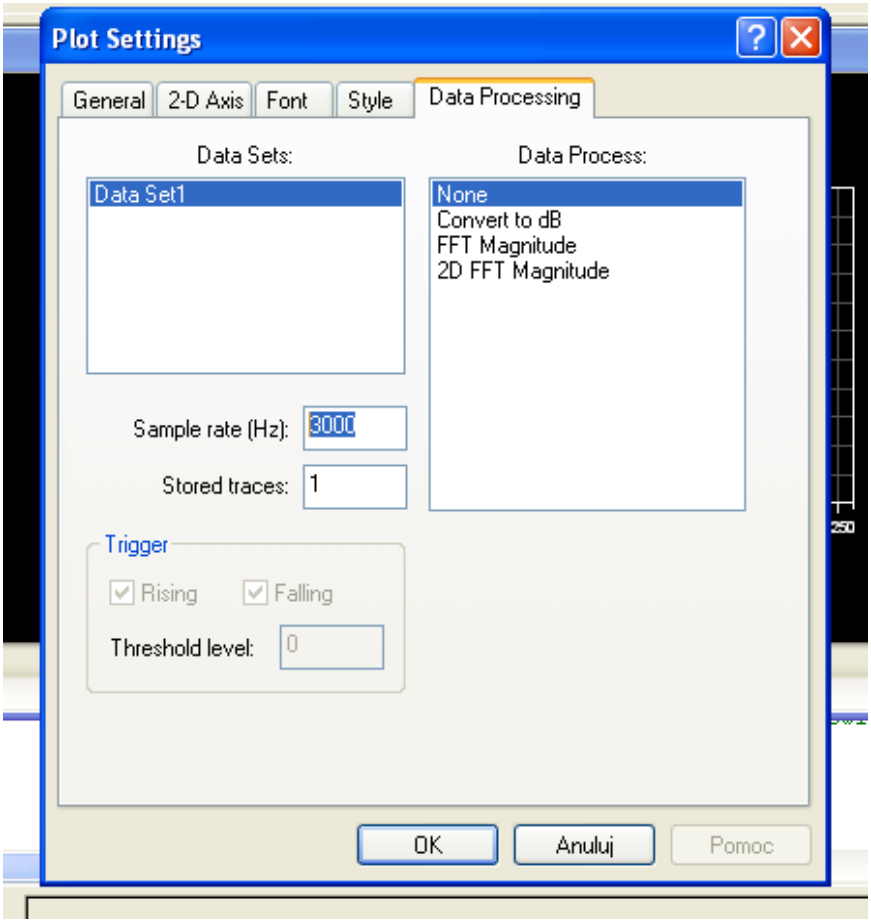
File name: podajemy plik z próbkami **(E1.dat**), który znajduje się we folderze **Gitara_Probki** znajdujący się w tym samym katalogu co program.

Otworzy się okno **Select Format** wybieramy **Floating Point 32 bit** i zatwierdzamy **OK,** następnie otwieramy **View\Debug Windows\Plot\New**. Otworzy się okno **Plot Configuration**, który ustawiamy tak samo jak zostało pokzane w ćwiczeniu wcześniejszym (powyżej).

Otrzymaliśmy:



Do odsłuchania załadowanych próbek należy na wykresie **Plot** kliknąć prawym przyciskiem myszy i wybrac **Modyfi Settings...** i kliknąć na zakładkę **Data Processing**. W **Sample rate (Hz)** wpisać **3000** i zatwierdzić.



Podłączyć słuchawki do wyjścia z komputer (zielony port).
Kliknąć prawym przyciskiem myszy na **Plot** i wybrać **Export...**. W oknie **Export Plot** wybrać **Sound Card** i kliknąć **Export**. Powinniśmy usłyszeć dźwięk w słuchawkach.

5. Wnioski.

Po zakończeniu tego ćwiczenia powinieneś oswoić się z narzędziem „Analog Devices VisualDSP++” i programowaniu w C++.

6. Literatura.

Instrukcja do laboratorium została oparta na:

- wcześniejszych wykonywanych ćwiczeniach na laboratorium Zastosowania Procesorów Sygnałowych
- danych z internetu – a szczególnie z ćwiczenia wykonywanego na Politechnice Poznańskiej : http://cygnus.et.put.poznan.pl/~piotrw/labdsp/FFT.pdf