Procesory Sygnałowe i Logika Programowalna
Laboratorium
Ćw. 2

Zajęcia wprowadzające II

1. Wstęp

Cwiczenie jest drugim etapem praktycznego zapoznawania się studentów z własnościami i metodami programowania zmiennoprzecinkowych procesorów sygnałowych Texas Instruments TMS320C6713 (C6713). Zadaniem studentów będzie implementacja przykładowych programów napisanych w języku C, na TI DSP Starter Kit'cie (DSK) przy wykorzystaniu oprogramowania narzędziowego Code Composer Studio (CCS) wersji 3.1.
Punktem wyjścia będzie projekt „loop_stereo.pjt" przeanalizowany i rozszerzony w ramach Ćw. Nr 1. Wydruk rozbudowanego pliku „loop_stereo.c zamieszczono poniżej.

```
*************************************************************************

//Loop_stereo.c  Stereo input/output to/from both channels

#include "C:\CCStudio_v3.1\C6000\dsk6713\include\dsk6713_aic23.h"
                                        //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;                                      //set
sampling rate

#define         LEFT  0
#define         RIGHT         1
union {Uint32 combo; short channel[2];} AIC23_data;

interrupt void c_int11()                                        //interrupt service routine
{
        AIC23_data.combo = input_sample();                              //input 32-bit
sample

        if ( DSK6713_DIP_get(0) == 0 ) {
          output_left_sample(AIC23_data.channel[LEFT]);         //I/O left channel
          DSK6713_LED_on(0);
          DSK6713_LED_off(3);
        }
        else  {
          output_right_sample(AIC23_data.channel[RIGHT]);       //I/O rigrt channel
          DSK6713_LED_on(3);
          DSK6713_LED_off(0);
        }

//      output_sample(AIC23_data.combo);          //I/O left & right channels

        return;
}
```

```
void main()                                                    //main function
{
        comm_intr();                        //init DSK, codec, McBSP
        DSK6713_DIP_init();
        DSK6713_LED_init();
        while(1);                   //infinite loop
}
```

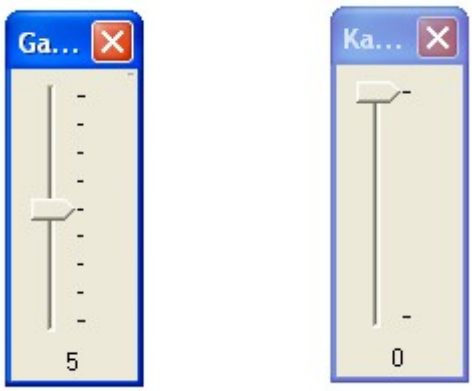
****

Program pozwala na sterowanie za pomocą DIP Switch'a nr „0” odsłuchem sygnału stereofonicznego wczytywanego na wejście DSK.

- Gdy DIP Switch nr „0” nie jest wciśnięty (DSK6713_DIP_get(0) zwraca „1”), świeci dioda nr „3” i słychać w prawym kanale sygnał z prawego wejścia.
- Gdy DIP Switch nr „0” wciśnięty (DSK6713_DIP_get(0) zwraca „0”) , świeci dioda nr „0” i słychać w prawym kanale sygnał z prawego wejścia.

2. Funkcje GEL

CCS posiada mechanizm sterowania wykonywaniem programu za pomocą obiektów GEL (ang. General Extention Language). Aby stworzyć Slider - regulator



Należy utworzyć plik z rozszerzeniem „*.gel”. Przykładowy plik „loop_stereo.gel” zamieszczono poniżej

```
**********************************************************************
menuitem "Wzmocnienie"

slider Gain1(1,9,1,2,gain_parameter) /*zmiany od 1 do 9 co 1 przy klikaniu w slider skoki co 2 */
{
      nGain = gain_parameter;        /*zmiana poziomu syglału*/
                  nGain -= 5;
}

slider KanalLewy(0,1,1,1,channel_parameter) /*zmiany od 0 do 1 co 1 przy klikaniu w slider skoki
co 1 */
{
      bLeft = channel_parameter;        /*zmiana kanału*/
}
**********************************************************************
```

i dodać go do projektu klikając File->Load GEL. W przypadku jakichkolwiek zmian w pliku konieczne jest przeładowanie pliku (klikamy prawym klawiszem myszy na nazwę pliku w projekcie i wybieramy RELOAD).
Zaproponowane regulatory posłużą do zmiany poziomu sygnału wysyłanego na wyjście oraz zmianie odsłuchiwanego kanału.

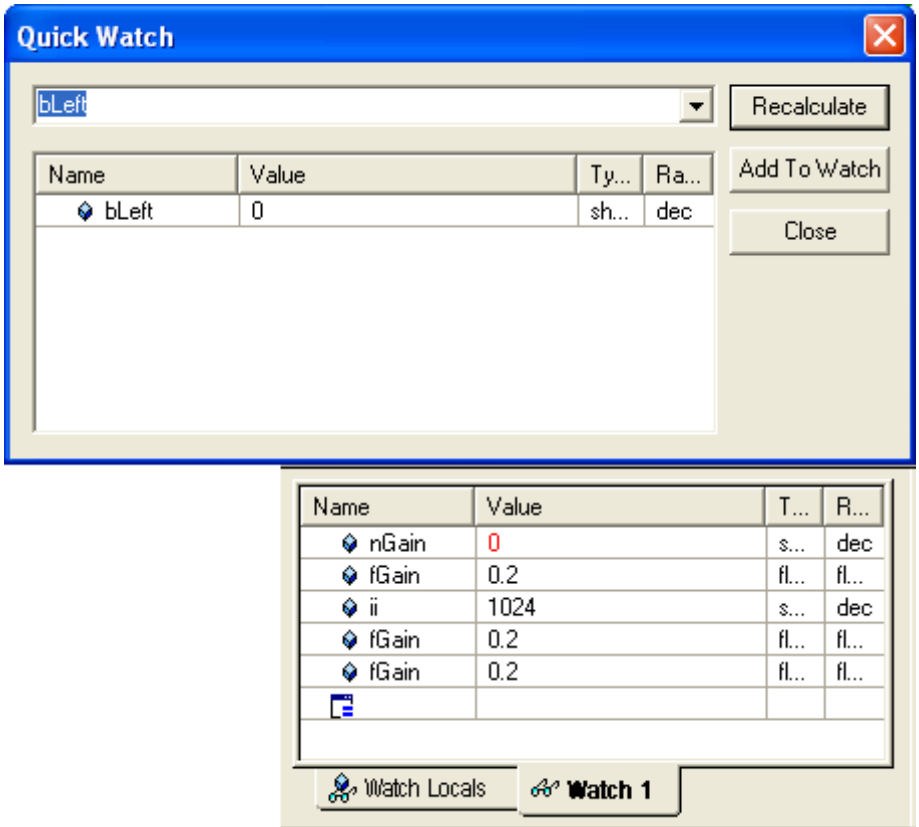
Do programu należy wprowadzić następujące zmienne.

```
short bLeft = FALSE;
short nGain = -4;
float fGain = 0.2;
float fAux;
```

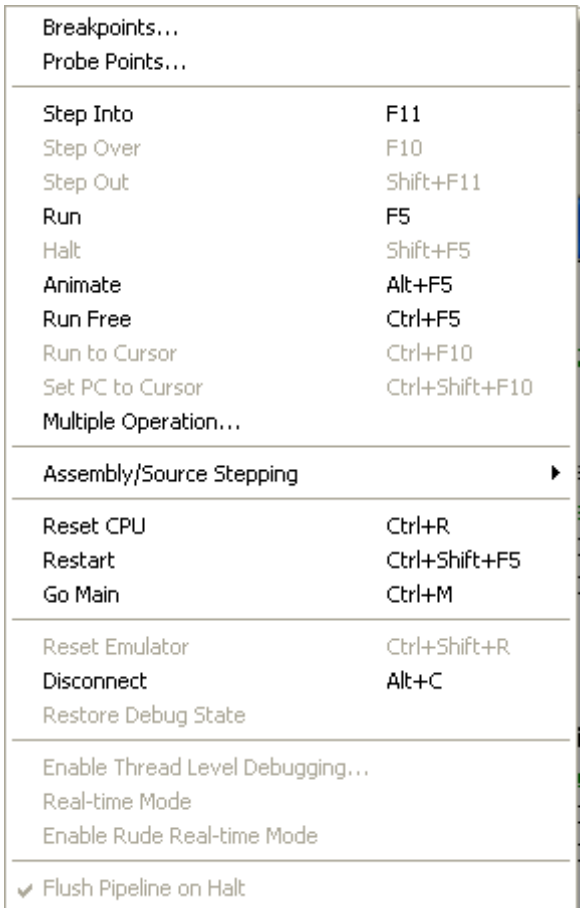
oraz w odpowiednich miejscach poniżej podane instrukcje (szczegóły poda prowadzący):

```
fGain = ( nGain >= 0 ) ? ( ( float )( nGain + 1 ) ) :( 1.0 / ( ( float )( -nGain + 1 ) ) );

fAux = ( float )( AIC23_data.channel[LEFT] ) * fGain;
output_left_sample( (short) fAux );        //I/O left channel

fAux = ( float )( AIC23_data.channel[RIGHT] ) * fGain;
output_right_sample( ( short ) fAux );     //I/O rigrt channel
```

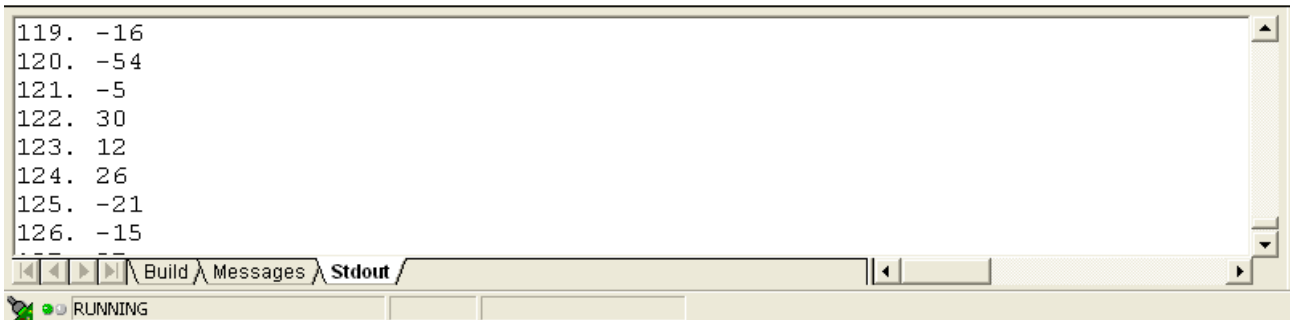
3. Źledzenie zmian wartości zmiennych
Wszystkie zmienne programu można oglądać w trakcie jego wykonywania. W tym celu należy kliknąć na View->Quick Watch i wybrać zmiena, którą chcemy obserwować.



4. Debugowanie programu
CCS pozwala na łatwe debagowanie programów. Większość opcji można znaleźć w zakładce „Debug”



5. Zapis danych do pliku
Dane zgromadzone w tablicach i pojedynczych zmiennych można zapisywać do plików np. za pomocą funkcji „fprintf” jak również wysyłać na ekran do standardowego strumienia wyjściowego jakim jest okienko „stdout”



do programu należy dodać plik nagłówkowy funkcji we/wy

#include <stdio.h>

oraz trzy zmienne:
short nBuf[BUFSIZE];
short ii = 0, jj;
FILE *fptr;

oraz odpowiedni fragment kodu (szczegóły poda prowadzący).

```
if ( ii < BUFSIZE )  {
          nBuf[ ii ] = ( short ) fAux;
        }

ii++;

if (ii > BUFSIZE)  {
  ii = BUFSIZE;
        bZapisano = TRUE;
        DSK6713_LED_off(1);
        DSK6713_LED_on(2);
        if ( DSK6713_DIP_get(3) == 0 ) {
          ii = 0;
          DSK6713_LED_on(1);
          DSK6713_LED_off(2);
        }
}
```

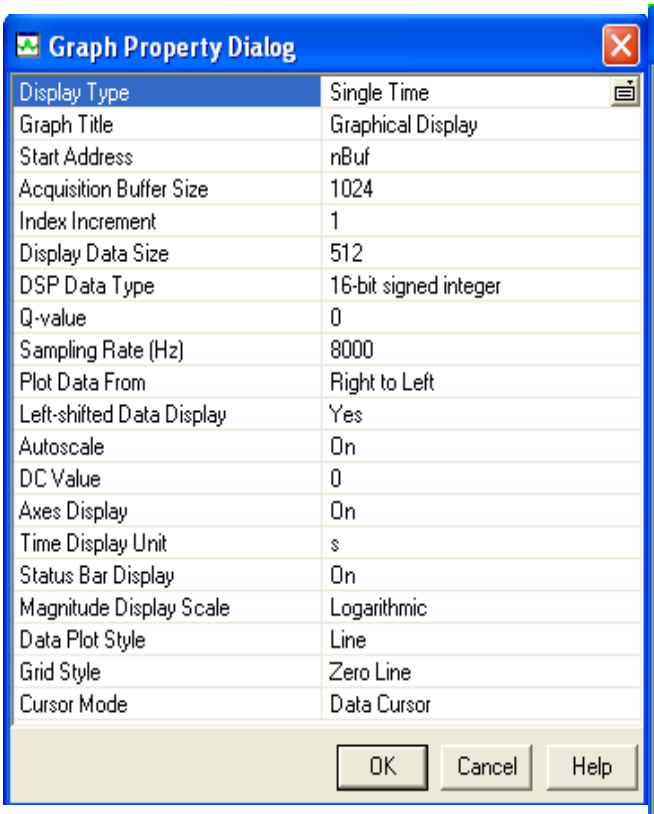
****

```
        while(1)  {                          //infinite loop
          if ( DSK6713_DIP_get(0) == 0 )  {
                fptr = fopen("Sygnal.dat", "w");
                for (jj=0; jj<BUFSIZE; jj++)  {
                  fprintf( fptr, "%d. %d\n", jj, nBuf[ jj ] );
                  if ( jj < 128 )  {
                    printf( "%d. %d\n", jj, nBuf[ jj ] );
                  }
                }
                fclose(fptr);
          }
```

****

6. Wykresy

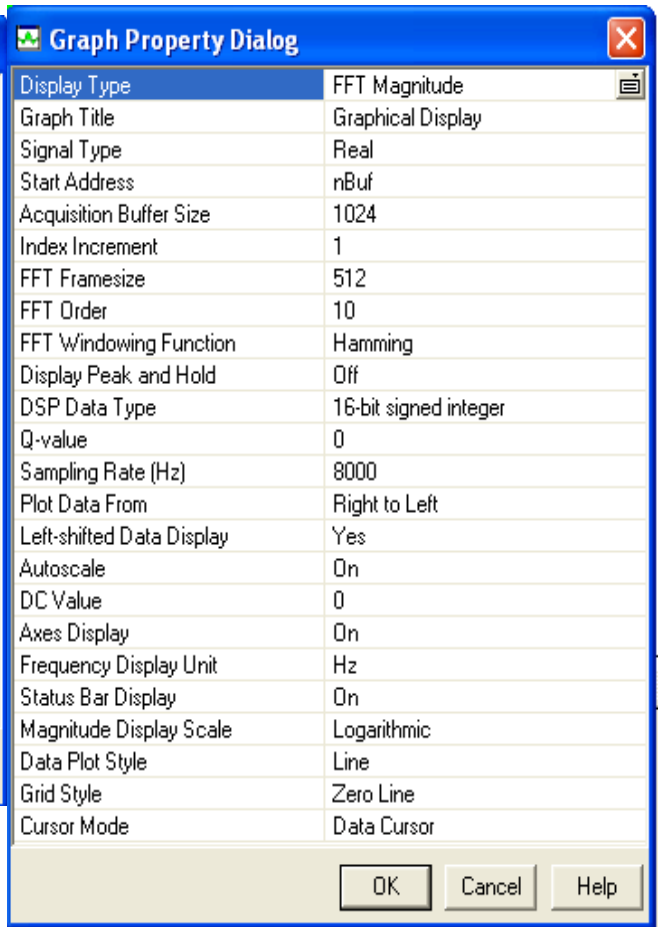
Aby oglądać wykresy czasowe oraz widma FFT danych zgromadzonych w tablicach należy wykonać komendę View->Graph->Time /frequency, a następnie wypełnić stosowne formularze.



Graph Property Dialog

Display Type	Single Time
Graph Title	Graphical Display
Start Address	nBuf
Acquisition Buffer Size	1024
Index Increment	1
Display Data Size	512
DSP Data Type	16-bit signed integer
Q-value	0
Sampling Rate (Hz)	8000
Plot Data From	Right to Left
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On
Magnitude Display Scale	Logarithmic
Data Plot Style	Line
Grid Style	Zero Line
Cursor Mode	Data Cursor

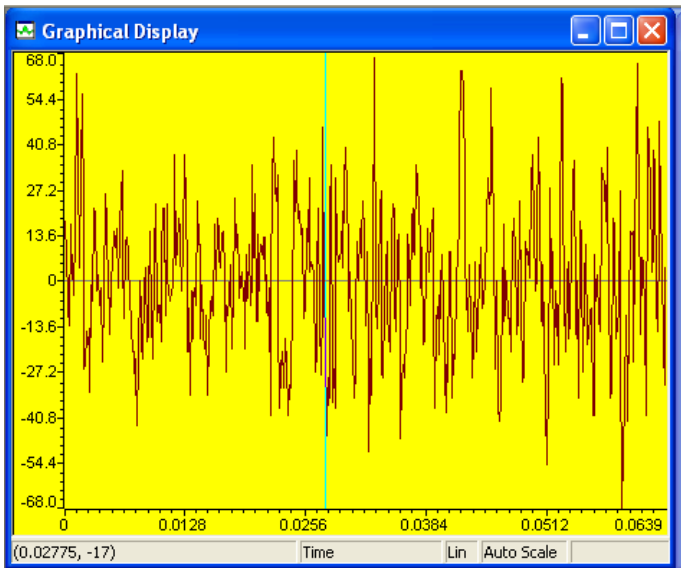
OK Cancel Help

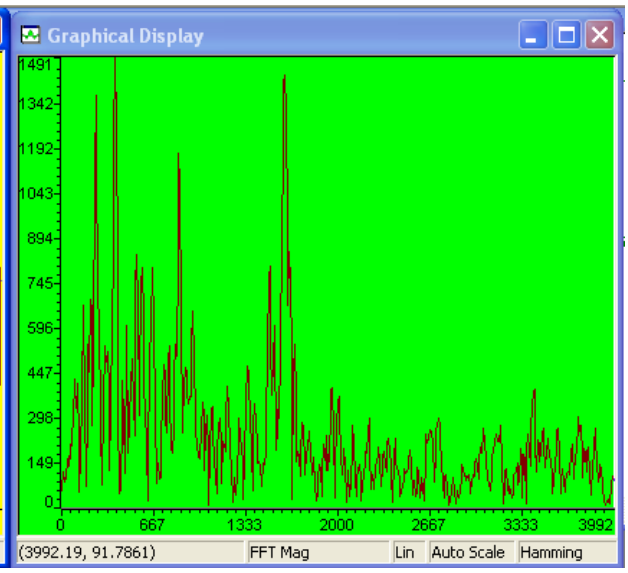


Graph Property Dialog

Display Type	FFT Magnitude
Graph Title	Graphical Display
Signal Type	Real
Start Address	nBuf
Acquisition Buffer Size	1024
Index Increment	1
FFT Framesize	512
FFT Order	10
FFT Windowing Function	Hamming
Display Peak and Hold	Off
DSP Data Type	16-bit signed integer
Q-value	0
Sampling Rate (Hz)	8000
Plot Data From	Right to Left
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Frequency Display Unit	Hz
Status Bar Display	On
Magnitude Display Scale	Logarithmic
Data Plot Style	Line
Grid Style	Zero Line
Cursor Mode	Data Cursor

OK Cancel Help





7. Wykorzystanie pamięci SDRAM

Aby można umieszczać dane w pamięci zewnętrznej SDRAM, należy skorzystać z dyrektywy preprocesora #pragma i skojarzyć zmienna, np dużą tablicę, z określonym adresem w pamięci zewnętrznej. W programie utworzono jedną duża tablicę, odpowiednie modyfikacje kodu zamieszczono poniżej.

```
#define N 65536
short nBufSDRAM[ N ];
#pragma DATA_SECTION( nBufSDRAM, ".EXT_RAM" )
```

```
/*C6713dsk.cmd  Linker command file*/

MEMORY
{
 IVECS:    org=0h,              len=0x220
 IRAM:                  org=0x00000220,      len=0x0002FDE0 /*internal memory*/
 SDRAM:      org=0x80000000, len=0x00100000 /*external memory*/
 FLASH:      org=0x90000000, len=0x00020000 /*flash memory*/
}

SECTIONS
{
 .EXT_RAM :> SDRAM
 .vectors :> IVECS    /*in vector file*/
 .text    :> IRAM     /*Created by C Compiler*/
 .bss     :> IRAM
 .cinit   :> IRAM
 .stack   :> IRAM
 .sysmem  :> IRAM
 .const   :> IRAM
 .switch  :> IRAM
 .far     :> IRAM
 .cio     :> IRAM
 .csldata :> IRAM
}
```

****
****

BIBLIOGRAFIA

R. Chassaing, Digital Signal Processing and Applications with the C6713 and C6416 DSK, Wiley & Sons, Inc., 2005,