Running the Echo Cancellation program

Note: The code cannot be run directly from the CD. Please copy the relevant code onto the hard-disk of the computer and execute it thereafter.

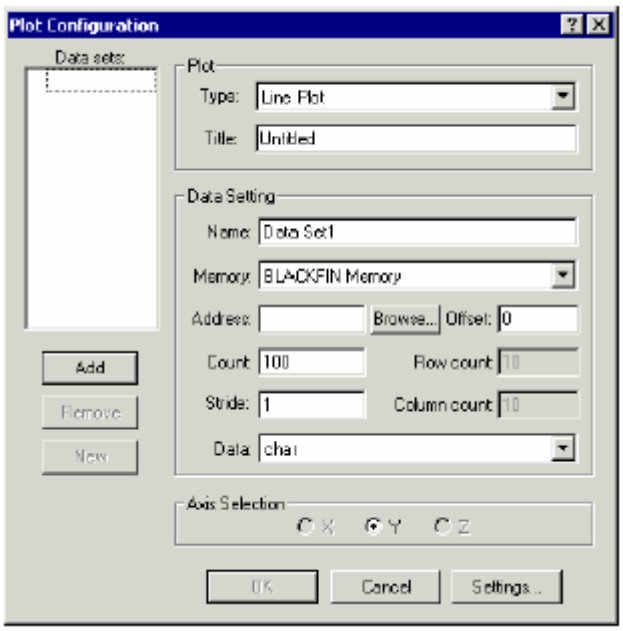
The program can be run in different configurations to achieve different goals. These configurations are as follows:
1. Run the program with an internally generated (i.e. simulated) echo.
2. Run the program with a real echo (i.e. generated externally).

Furthermore, there are several options that can be changed in either of the configurations, such as the sampling frequency, length, and nature of the echo (the last entity being applicable only to the simulated echo).

Building the project
First of all, in order to compile and run the code, the following steps need to be performed.

1. Open DSP++ and open C_Talkthrough_TDM.dpj . Right click it and select "build project".
2. After the project has been built and compiled, press F5 to run it.
3. It may be halted as per the user's wishes by pressing shift+F5 (while it is running)

Viewing the plots
At any point while running the system, in order to have a look at the impulse response of the adaptive LMS filter, the following needs to be done…
First of all, halt the system, by pressing shift+F5. Then do the following.
1. From the **View** menu, choose **Debug Windows** and **Plot**. Then choose **New** to open the **Plot Configuration** dialog box (shown below).

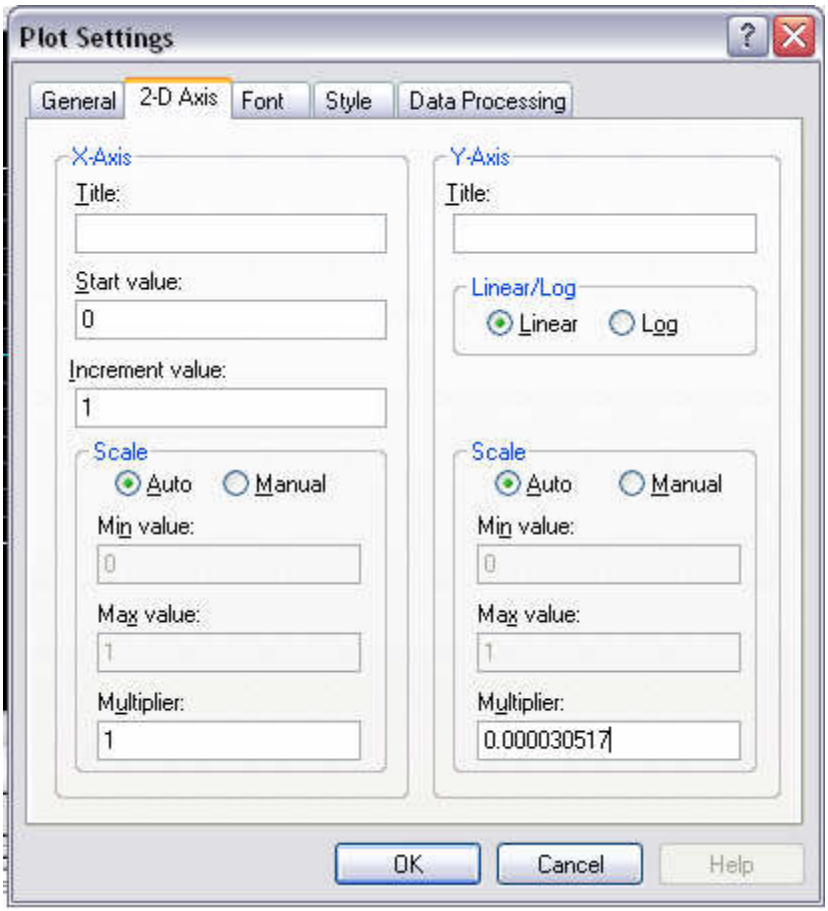


Here you will add the data sets that you want to view in a plot window.

2. In the **Plot** group box, specify the following values.

- In the **Type** box, select **Line Plot** from the drop-down menu.
- In the **address** box type lms_filter_coeff.
- In the **count** box enter the lms_TAP_NUM value that was used in the algorithm.
- **Stride** should be 1.
- Select 'short' in the data section.
- Click 'add' and then 'ok'.

The plot displayed will be shown in short and it needs to be converted to a floating value. On the plot window, right click and goto modify settings. In the 2-D axis tab, enter 1/32768 = 0.00003051757 as shown in the following figure.



Now, the coefficients should be displayed on a scale of -1 to 1 (on the y axis).

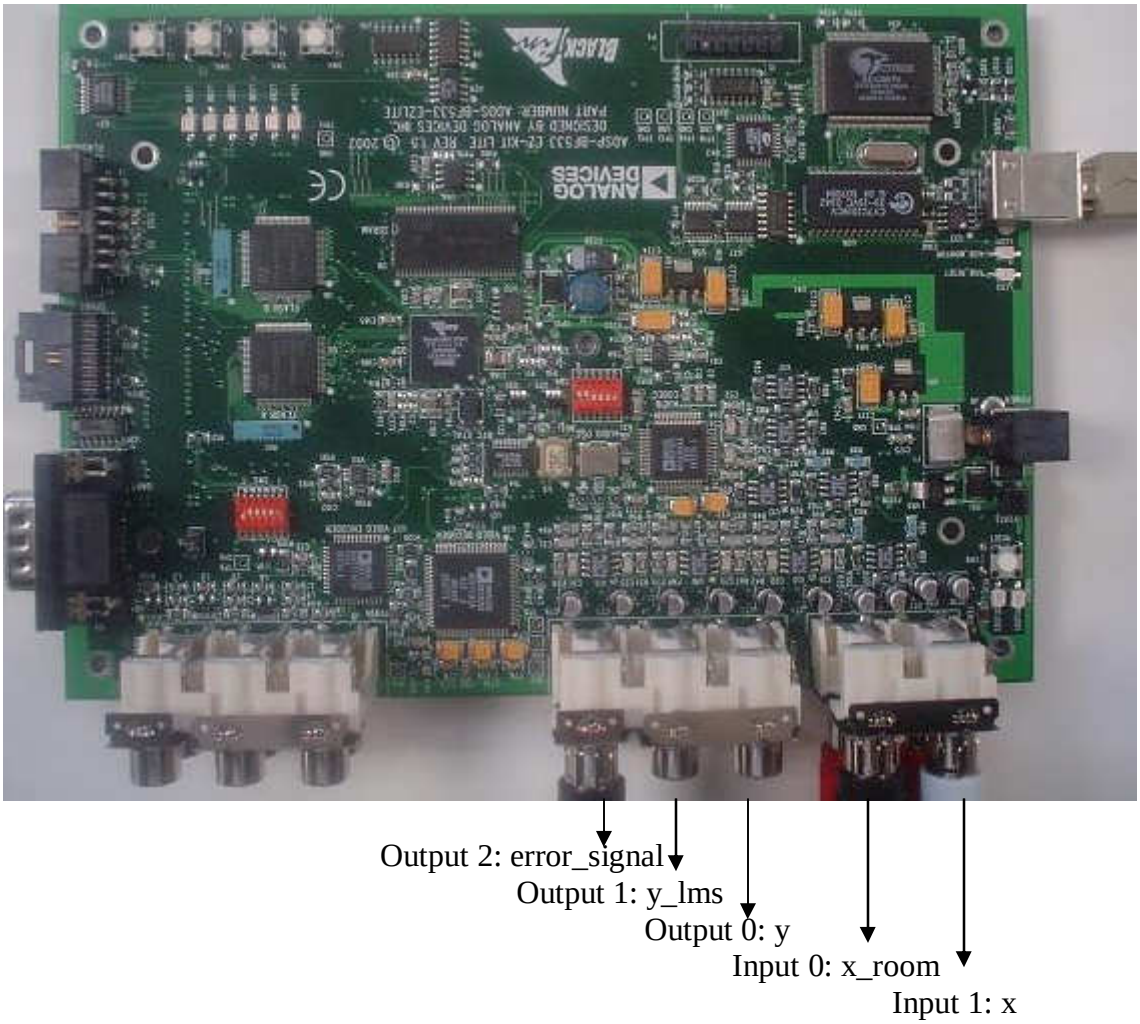
The plot can only be read when the system is halted, *not* when it is running. The plot will auto-refresh every time the memory values change.

Setting the volume levels

In order to get the best results, the volume levels should be set as per the following:

- Windows media player volume should be full (where the audio files are played)
- Main volume control from windows should be full.
- Wave volume from windows should be full.
- AudioBuddy preamp gain should be half full.
- Altec Lansing desktop speakers volume should be half full.
- When recording the output of the board to the PC, the line-in recording level of windows should be half full.

Input-Output port configuration:
The following figure shows the input and output port configuration that was used for testing and evaluation purposes.



NOTE: It is important to note here that only the left (white) pins of the input and output audio ports are used. The right (red) pins are un-responsive. Effectively, we are working with mono outputs.

Work with Simulated Echo:

In order to run the program with the simulated echo, the following changes need to be made:

Wire Connections:

Connect input 1 to the original source (eg: PC, ipod, mic etc.) – *This connection should be made with a stereo plug to 2 RCA plugs cable.*
Output 0 will be the output from the simulated echo generator – *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*
Output 1 will be the output from the adaptive lms filter - *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*
Output 2 will be the error signal - *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*

Mandatory Code modifications:

Main procedure in main.c file. Make the following modifications:
Make sure the echo_generator(); procedure call is un-commented.

```
// Comment echo_generator(); if working with real world echo
echo_generator();    //Call the inbuilt echo_generator
adaptive_lms();      //Call adaptive lms

y = echo_output << 16; //Need to shift input sample -anomaly
y_lms = out_fir << 16; //Need to shift input sample -anomaly
output_error = error_sig << 16; //Need to shift input sample
```

Options:

1. Sampling Frequency: EX_INTERRUPT_HANDLER(Sport0_RX_ISR) procedure in the ISR.c file.

For a sampling frequency of 48 khz change freq_count to 0.
For a sampling frequency of 24 khz change freq_count to 1.
For a sampling frequency of 16 khz change freq_count to 2.
For a sampling frequency of 8 khz change freq_count to 5.

The system has been tested with sampling freq. of 8 and 16 khz. Going beyond that might work or might cause undesirable behaviour.

```
EX_INTERRUPT_HANDLER(Sport0_RX_ISR)
{
    // confirm interrupt handling
    *pDMA1_IRQ_STATUS = 0x0001;

    if(freq_count == 5)
    {
        *pSIC_IMASK = 0x00000000; // Disable interrupts
        ssync();

        freq_count = 0;
        // copy input data from dma input buffer into variables
```

2. Echo TAP NUM: variable echo_TAP_NUM in the file talkthrough.h.

This will change the length of the simulated echo filter.

```
#define FLOW_1  0x1000

//User Defined TAP lengths
#define echo_TAP_NUM 3500   //Sets the length of the simulated echo
#define lms_TAP_NUM 4000    //Sets the length of the adaptive lms f

//-----------------------------------------------------------------
```

3. Number of simulated echoes: main procedure in main.c file.

In order to change the location of the echoes, variables echo_reader_first, echo_reader_second and echo_reader_third need to be changed which cause multiple echoes. In this example figure, echo_reader_first = 1 means that the echo will have a delay of 3500 samples (assuming echo_TAP_NUM of 3500) and echo_reader_second = 1000 means that the second echo will have a delay of 3500-1000=2500 samples and so on.

Note: The values of the three echo readers should not exceed **echo_TAP_NUM – 1.**

```
    //Variable Initialization
    echo_writer = 0;              //Writer to the echo delay buffer
    echo_reader_first = 1;        //First echo reader - read from e
    echo_reader_second = 1000;    //Second echo reader - read from
    echo_reader_third = 3000;     //Third echo reader - read from e
    freq_count = 0;
    ADC_flag = 0;
    step_size = 0x0a00;           //Initialize the step size. It ca
                                  //since the NLMS algorithm will u
```

4. LMS TAP NUM: variable lms_TAP_NUM in the file talkthrough.h.

This will change the length of the adaptive lms filter. Please do not exceed the value of 4000 TAPs.

```
#define FLOW_1  0x1000

//User Defined TAP lengths
#define echo_TAP_NUM 3500   //Sets the length of the simulated echo
#define lms_TAP_NUM 4000    //Sets the length of the adaptive lms f

//-----------------------------------------------------------------
```

5. LMS TAP NUM: variable P2 in the file NLMS_step_updater_sum.asm .

Please change this variable if the lms TAP length is changed from 4000 to any lower value.

For a TAP length of 4000 make it (4000/250)/2 = 8
For a TAP length of 1000 make it (1000/250/2 = 2
For a TAP length of 500 make it (500/250)/2 = 1

Please note that the TAP length of an LMS filter in essence needs to be an even value greater than 500 and be a multiple of 250.

```
L0=R3;                  // Initialize delay line buffer ler
L1=R3;                  // Initialize delay line buffer ler
P2=8;                   // Outer Loop index - need to chang
                        // For a TAP length of 4000 make it
                        // For a TAP length of 1000 make it
                        // For a TAP length of 500 make it

P3=250;                 // Inner Loop index - DO NOT CHANGE
nop;nop;nop;
```

Work with Real-World Environment:

In order to run the program with the real-world environment, following changes need to be made:

Wire Connections:

Connect input 1 to the original source (eg: PC or ipod etc., but not a mic) – *This connection should be made with a stereo plug to 2 RCA plugs cable. At the PC or iPOD end, use a stereo plug to 2 stereo jacks cable, where one jack connects to the speakers and the other to the cable connected at input 1.*

Connect input 0 to the microphone (used audio-technica© microphone) – *This connection should be made with a stereo plug to 2 RCA plugs cable. The stereo plug should then be connected to the output of the mic. Preamplifier.*

Output 0 will be the output as recorded by the microphone – *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*

Output 1 will be the output from the adaptive lms filter - *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*

Output 2 will be the error signal - *This connection should be made with a stereo jack to 2 RCA plug cable. This can be further connected to headphones or speakers.*

Mandatory Code modifications:

Main procedure in main.c file. Make the following modifications:

Make sure that the echo_generator(); procedure call is commented.

```
// Comment echo_generator(); if working with real world echo
echo_generator();    //Call the inbuilt echo_generator
adaptive_lms();      //Call adaptive lms

y = echo_output << 16; //Need to shift input sample -anomaly
y_lms = out_fir << 16; //Need to shift input sample -anomaly
output_error = error_sig << 16; //Need to shift input sample
```

Options:

1. Sampling Frequency: EX_INTERRUPT_HANDLER(Sport0_RX_ISR) procedure in the ISR.c file.

For a sampling frequency of 48 khz change freq_count to 0.

For a sampling frequency of 24 khz change freq_count to 1.

For a sampling frequency of 16 khz change freq_count to 2.
For a sampling frequency of 8 khz change freq_count to 5.

The system has been tested with sampling freq. of 8 and 16 khz. Going beyond that might work or might cause undesirable behaviour.

```
EX_INTERRUPT_HANDLER(Sport0_RX_ISR)
{
    // confirm interrupt handling
    *pDMA1_IRQ_STATUS = 0x0001;

    if(freq_count == 5)
    {
        *pSIC_IMASK = 0x00000000; // Disable interrupts
        ssync();

        freq_count = 0;
        // copy input data from dma input buffer into variables
```

2. LMS TAP NUM: variable lms_TAP_NUM in the file talkthrough.h.

This will change the length of the adaptive lms filter. Please do not exceed the value of 4000 TAPs.

```
#define FLOW_1  0x1000

//User Defined TAP lengths
#define echo_TAP_NUM 3500   //Sets the length of the simulated echc
#define lms_TAP_NUM 4000    //Sets the length of the adaptive lms f

//-----------------------------------------------------------------
```

3. LMS TAP NUM: variable P2 in the file NLMS_step_updater_sum.asm .

Please change this variable if the lms TAP length is changed from 4000 to any lower value.

For a TAP length of 4000 make it (4000/250)/2 = 8
For a TAP length of 1000 make it (1000/250/2 = 2
For a TAP length of 500 make it (500/250)/2 = 1

Please note that the TAP length of an LMS filter in essence needs be an even value greater than 500 and be a multiple of 250.

```
            L0=R3;              // Initialize delay line buffer ler
            L1=R3;              // Initialize delay line buffer ler
            P2=8;               // Outer Loop index - need to chang
                                // For a TAP length of 4000 make it
                                // For a TAP length of 1000 make it
                                // For a TAP length of 500 make it

            P3=250;             // Inner Loop index - DO NOT CHANGE
            nop;nop;nop;
```