|  |  |  |
| --- | --- | --- |
| **DIGITAL SIGNAL PROCESSING**  **LAB 3**  FIR Filter implementation in MATLAB and in C | | |
| Date of lab class:  04-June-2015 | Handed-in date:  11-June-2015 | Team members:  Amin Bakhtizin  Sukrat Khanna |
| Professor:  Prof. Dr.  Ulrich Sauvagerd |
| Professor’s remarks: | | |

Contents

[Objectives 3](#_Toc421805640)

[3.1 MATLAB Simulations of Lowpass and Bandpass Filters 4](#_Toc421805641)

[3.1.1 Lowpass and Highpass filter designs 4](#_Toc421805642)

[3.1.2 Bandpass filter design 5](#_Toc421805643)

[Attachment D and E 6](#_Toc421805644)

[Attachment F and G 6](#_Toc421805645)

[Attachment H and K 7](#_Toc421805646)

[3.2 Analog Transfer Characteristics of the DSK Board CODEC AIC23 7](#_Toc421805647)

[Attachment M 8](#_Toc421805648)

[Attachment N 8](#_Toc421805649)

[Attachment O 9](#_Toc421805650)

[3.3.2 Bandpass Filter 10](#_Toc421805651)

[Attachment P 10](#_Toc421805652)

[Appendix 1: C implementation 11](#_Toc421805653)

[Source code and Header for LP implementation (interrupt handler only) 11](#_Toc421805654)

[Source code and Header for HP implementation (interrupt handler only) 11](#_Toc421805655)

[Source code and Header for BP implementation (interrupt handler only) 12](#_Toc421805656)

[Appendix 2.1: MATLAB implementation of LP and HP 13](#_Toc421805657)

[Appendix 2.2: MATLAB implementation of BP 14](#_Toc421805658)

[Appendix 3: MATLAB implementation of Attachments F and G 15](#_Toc421805659)

[Appendix 4: MATLAB implementation of Attachments H and K 15](#_Toc421805660)

# Objectives

The task was to design and implement in real-time FIR digital filters on the TMS320C6713 DSP. For design an evaluation purposes as well as for simulation MATLAB was used.

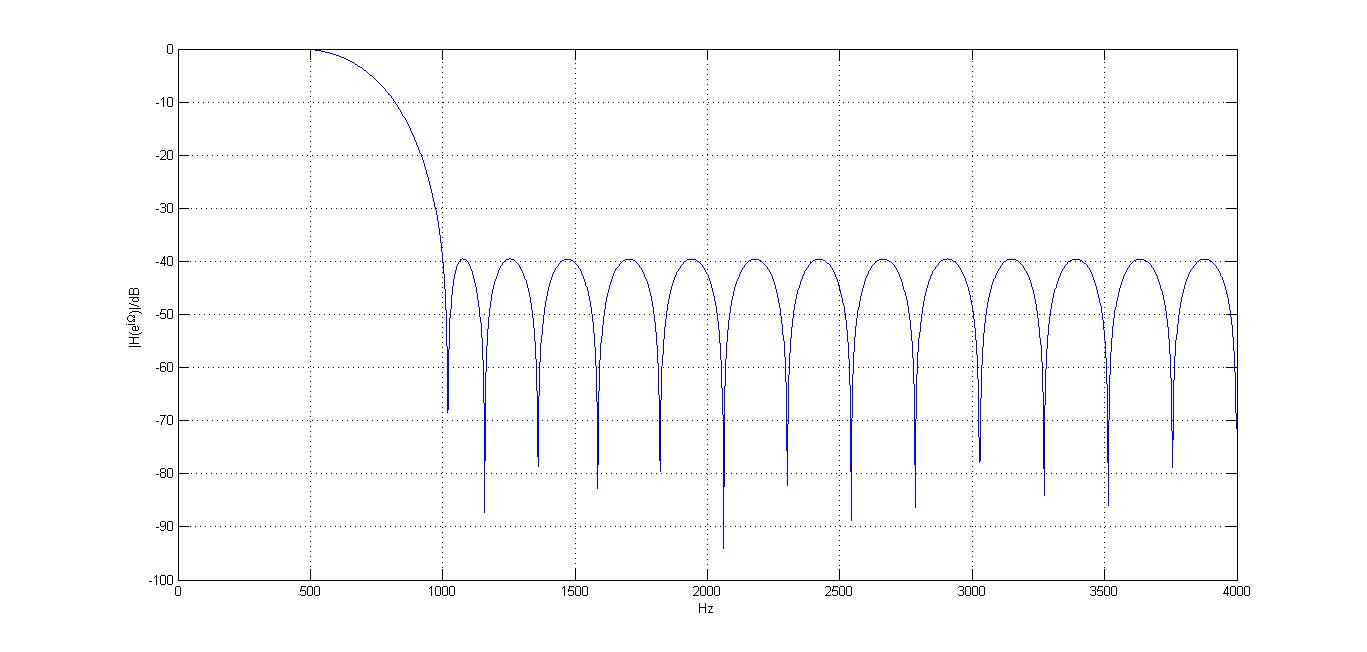
This report is structured in such a manner that it is easy to read with lab task description material. Therefore, same numbering of sections and figure names (as they were requested in task description) were used.

# 3.1 MATLAB Simulations of Lowpass and Bandpass Filters

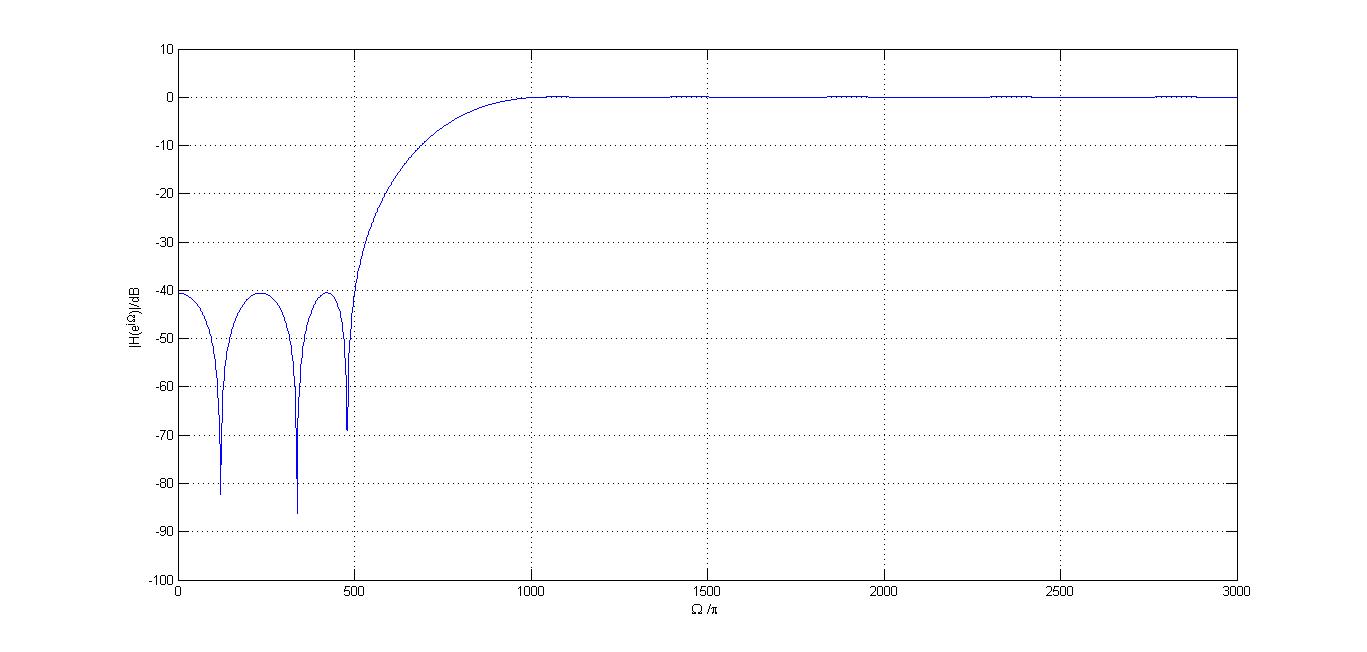
## 3.1.1 Lowpass and Highpass filter designs

It is was specified FIR lowpass filter with following specification was designed in MATLAB:

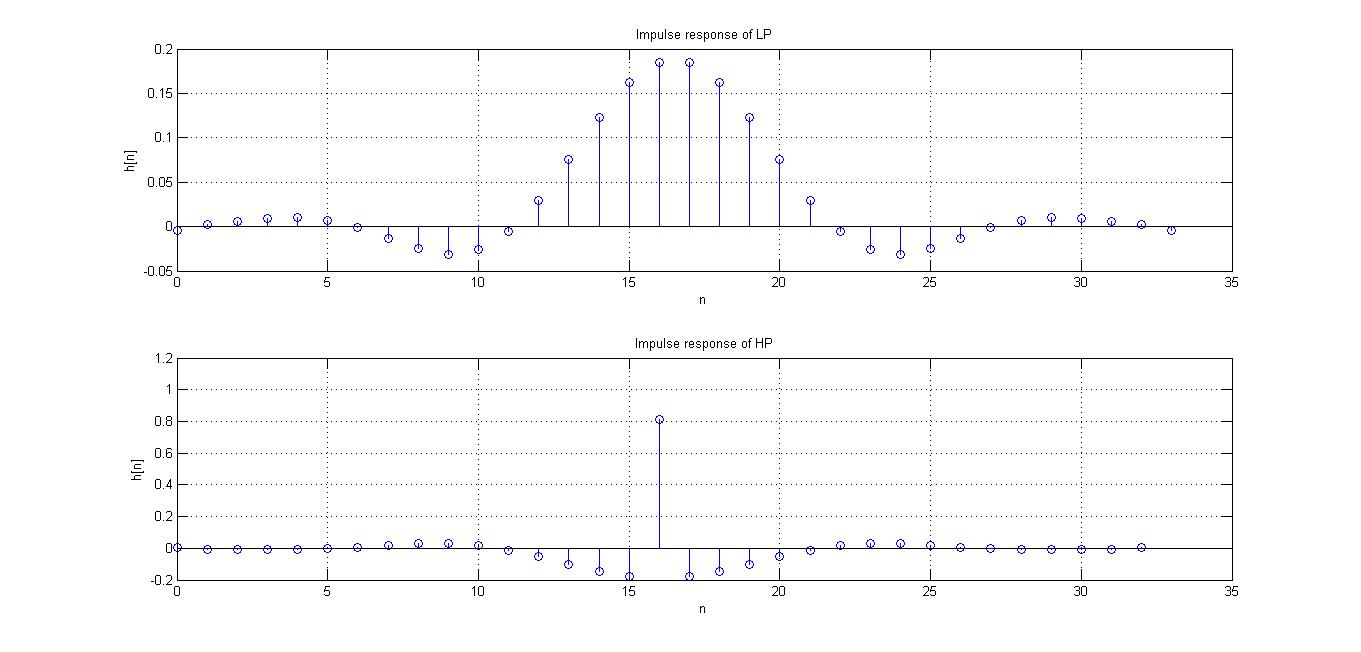
* passband edge frequency: 500 Hz,
* stopband edge frequency: 1000 Hz,
* maximum passband ripple: 0.01, (this is NOT in dB)
* minimum stopband attenuation: 40 dB,
* sampling frequency: 8 kHz.

***Attachment A***

From the Attachment A above it can be seen that the minimum stopband damping is maintained.

 ***Attachment B***

Attachment B depicts the amplitude response of highpass FIR filter, which meets the requirements regarding the minimum stopband damping.



***Attachment C***

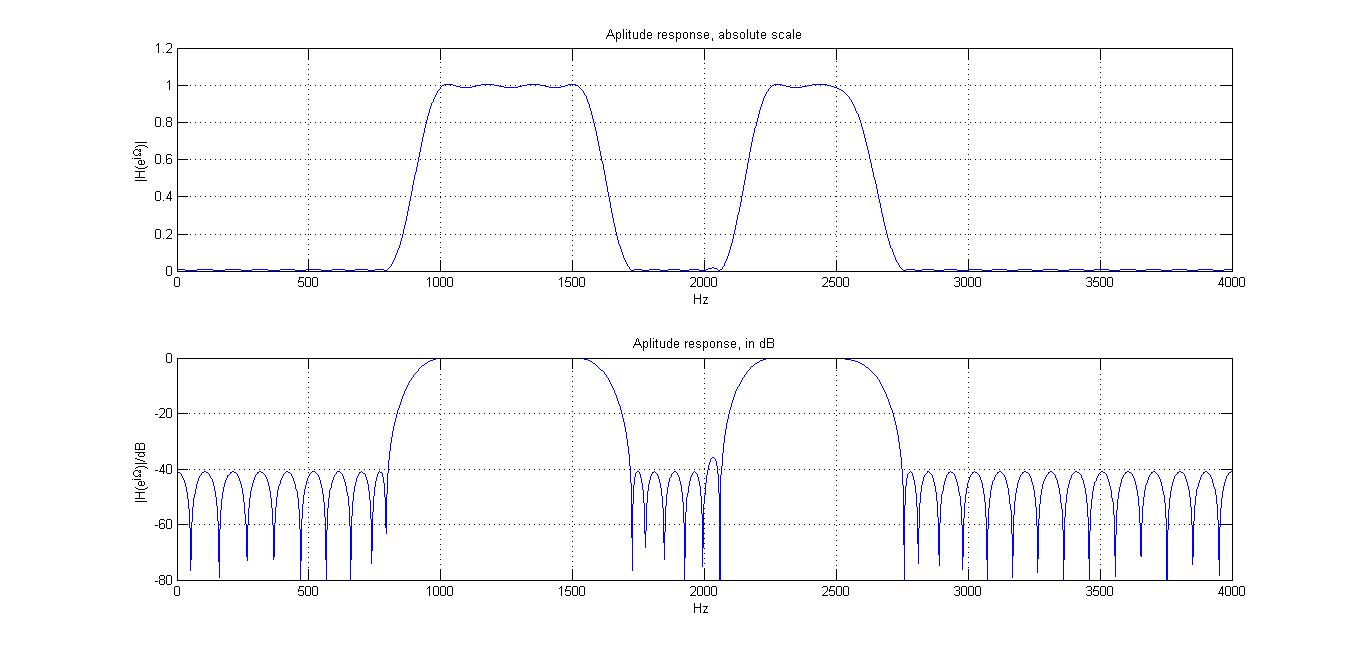
Attachment C describes the impulse responses of lowpass and highpass filters.

## 3.1.2 Bandpass filter design

The bandpass FIR filter with the following specifications was designed:

* The sampling rate is 8 kHz.
* There are two passbands.
* Passband 1 is between 1000 and 1500 Hz. In this frequency range, |H|=1 with delta\_p=0.01.
* Passband 2 is between 2250 and 2500 Hz. In this frequency range, |H|=1 with delta\_p=0.01.
* There are three stopbands.
* Stopband 1 is between 0 and 800 Hz.
* Stopband 2 is between 1750 and 2000 Hz.
* Stopband 3 is between 2750 and 4000 Hz.
* The minimum stopband attenuation should be 40 dB.
* The stopband ripple should be 0.01.

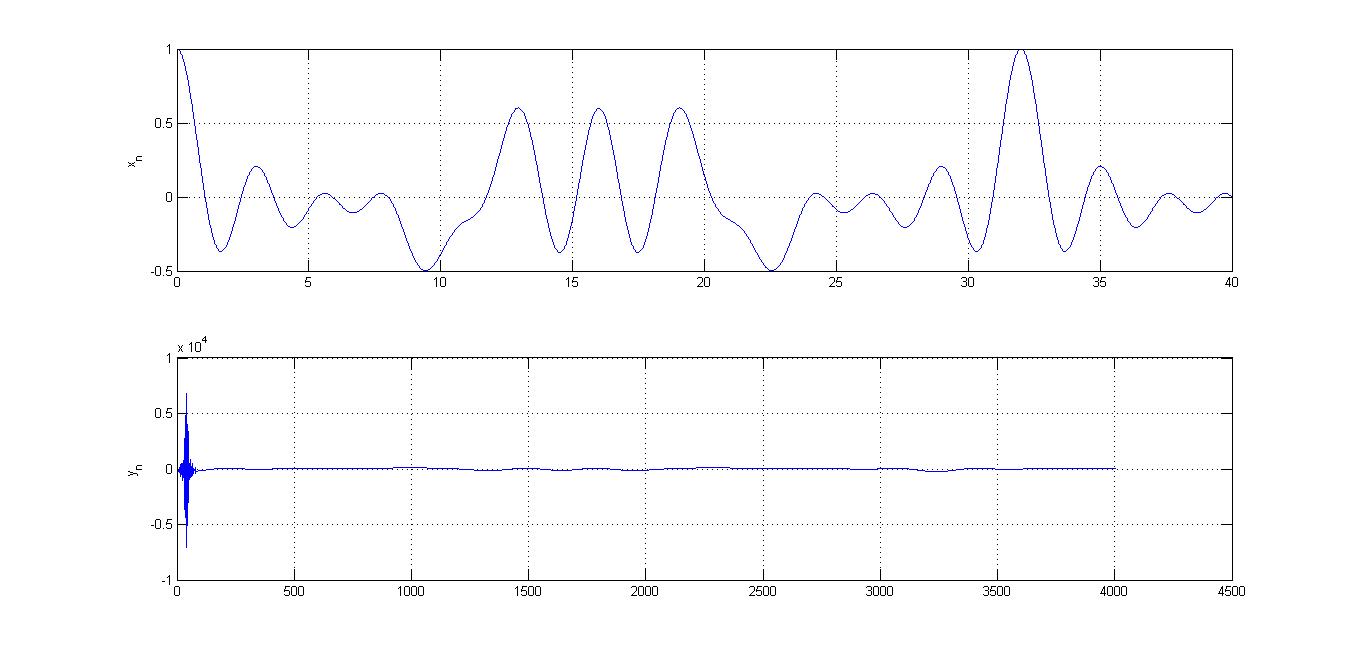
### Attachment D and E



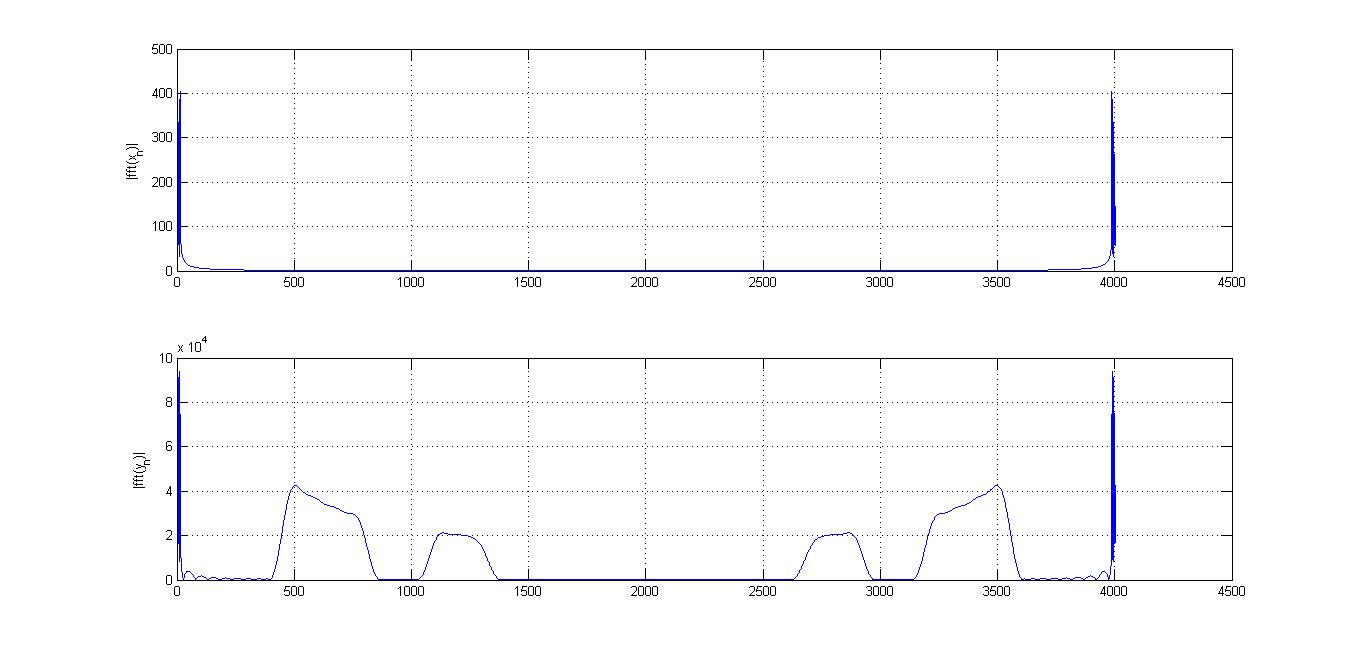
As it can be seen that stopband attenuation requirement was met.

### Attachment F and G

As it can be seen from figure below bandpass filter properly filters the input signal consisting of 5 cosine signals.



### Attachment H and K

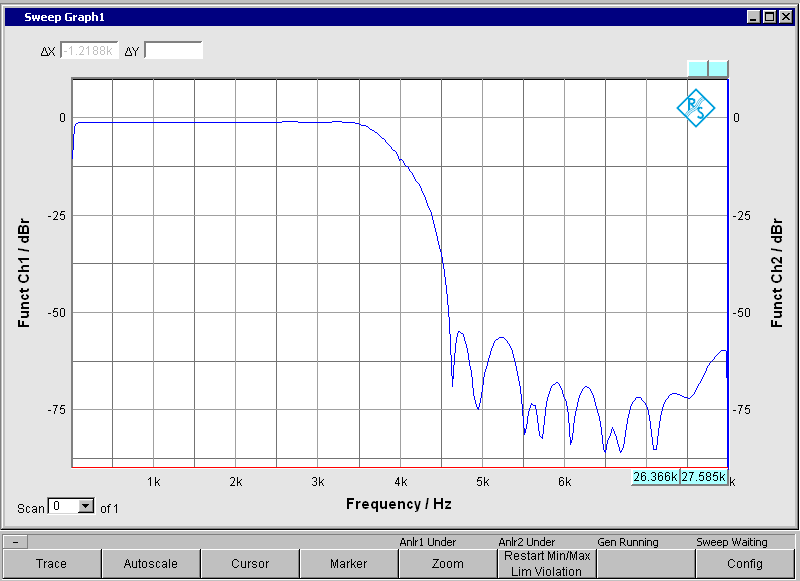


# 3.2 Analog Transfer Characteristics of the DSK Board CODEC AIC23

Simple program provided in the Lab starter folder was used and Fs=8kHz was used.

The frequency response of the board was measured with the help of a spectrum analyzer and set-file sweep-up-to-8kHz.set at the UPV analyzer was used. The amplitude was set such that no overflow could occur.

Following graph shows the measured amplitude response of the DSK-board:



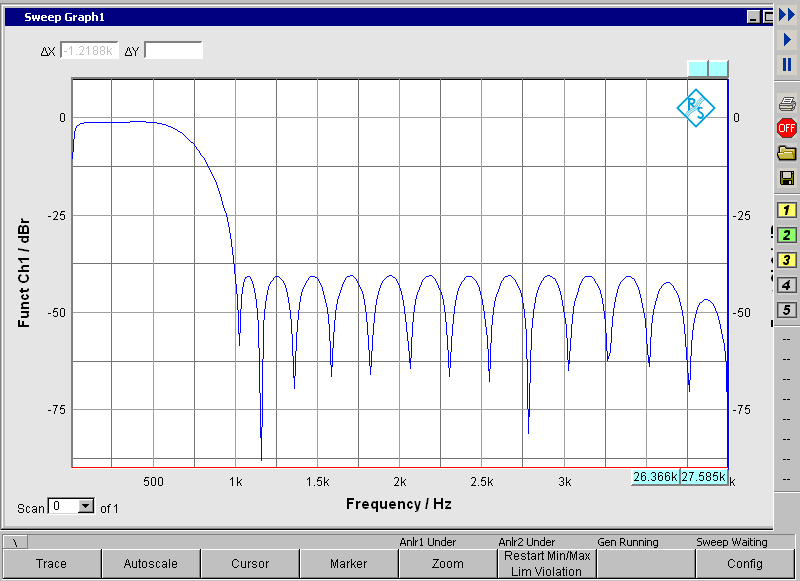
***Attachment L***

3.3 Real-time Fixed-Point Implementation of the FIR Filters

## Attachment M

The procedure is advantageous because by shifting the output to the right by 15 most of the noise is removed (since MSB and following bits with most weight are taken).

1. FIR filter amplitude response verification using the spectrum analyzer



## Attachment N

Having nothing changed in main function therefore it was decided include only interrupt function which implements the logic:

interrupt void intser\_McBSP1() {

AIC23\_data.both = MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //input data

for (i = N\_FIR\_LP\_coeffs-1; i >= 1; i--) {

delays[i] = delays[i - 1];

}

delays[0] = AIC23\_data.channel[LEFT];

for (i = 0; i < N\_FIR\_LP\_coeffs; i++) {

FIR\_accu32 += b\_FIR\_LP[i] \* delays[i];

}

FIR\_out\_LP = (short) (FIR\_accu32 >> 15);

FIR\_accu32 = 0;

AIC23\_data.channel[LEFT] = FIR\_out\_LP;

MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE, AIC23\_data.both); //output 32 bit data

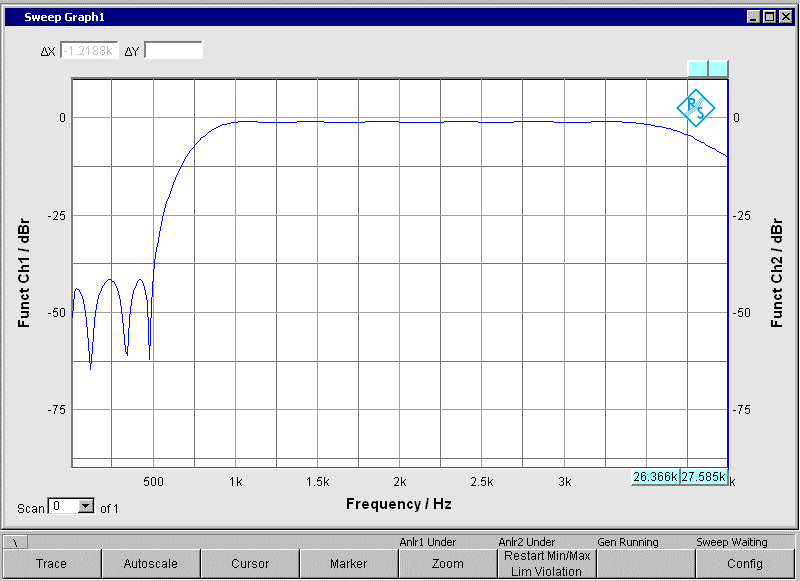
return;

}

## Attachment O

A clever way to obtain the highpass filter output signal is to calculate the coefficients of highpass filer by simply subtracting the impulse response of lowpass filter from impulse.

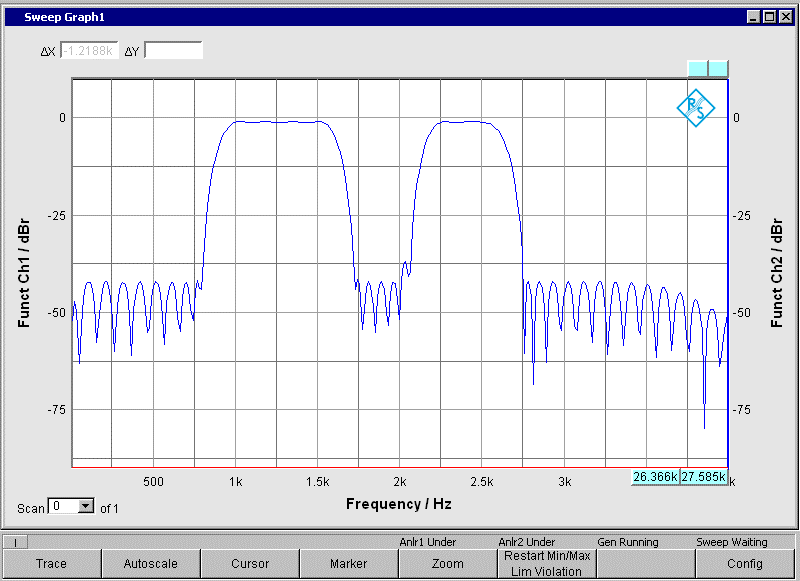
Following figure shows the amplitude response of implemented highpass filter:



## 3.3.2 Bandpass Filter

### Attachment P

This is the amplitude response of bandpass filter obtained by using a spectrum analyzer and set-file sweep-up-to-4kHz.set at the UPV:



# Appendix 1: C implementation

## Source code and Header for LP implementation (interrupt handler only)

interrupt void intser\_McBSP1() {

AIC23\_data.both = MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //input data

for (i = N\_FIR\_LP\_coeffs-1; i >= 1; i--) {

delays[i] = delays[i - 1];

}

delays[0] = AIC23\_data.channel[LEFT];

for (i = 0; i < N\_FIR\_LP\_coeffs; i++) {

FIR\_accu32 += b\_FIR\_LP[i] \* delays[i];

}

FIR\_out\_LP = (short) (FIR\_accu32 >> 15);

FIR\_accu32 = 0;

AIC23\_data.channel[LEFT] = FIR\_out\_LP;

MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE, AIC23\_data.both); //output 32 bit

return;

}

Header File content:

#define N\_FIR\_LP\_coeffs 34

short b\_FIR\_LP[N\_FIR\_LP\_coeffs]={

-147, 93, 197, 305, 339, 231, -44, -438,

-825, -1021, -844, -176, 981, 2476, 4035, 5330,

6064, 6064, 5330, 4035, 2476, 981, -176, -844,

-1021, -825, -438, -44, 231, 339, 305, 197,

93, -147};

## Source code and Header for HP implementation (interrupt handler only)

interrupt void intser\_McBSP1() {

AIC23\_data.both = MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //input data

for (i = N\_FIR\_HP\_coeffs-1; i >= 1; i--) {

delays[i] = delays[i - 1];

}

delays[0] = AIC23\_data.channel[LEFT];

for (i = 0; i < N\_FIR\_HP\_coeffs; i++) {

FIR\_accu32 += b\_FIR\_HP[i] \* delays[i];

}

FIR\_out\_HP = (short) (FIR\_accu32 >> 15);

FIR\_accu32 = 0;

AIC23\_data.channel[RIGHT] = FIR\_out\_HP;

MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE, AIC23\_data.both); //output 32 bit data, LEFT

return;

}

Header File content:

#define N\_FIR\_HP\_coeffs 35

short b\_FIR\_HP[N\_FIR\_HP\_coeffs]={

160, -26, -128, -258, -346, -317, -122, 229,

645, 966, 999, 584, -341, -1698, -3270, -4741,

-5789, 26599, -5789, -4741, -3270, -1698, -341, 584,

999, 966, 645, 229, -122, -317, -346, -258,

-128, -26, 160};

## Source code and Header for BP implementation (interrupt handler only)

interrupt void intser\_McBSP1() {

AIC23\_data.both = MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //input data

for (i = N\_FIR\_BP\_coeffs-1; i >= 1; i--) {

delays[i] = delays[i - 1];

}

delays[0] = AIC23\_data.channel[LEFT];

for (i = 0; i < N\_FIR\_BP\_coeffs; i++) {

FIR\_accu32 += b\_FIR\_BP[i] \* delays[i];

}

FIR\_out\_BP = (short) (FIR\_accu32 >> 15);

FIR\_accu32 = 0;

AIC23\_data.channel[RIGHT] = FIR\_out\_BP;

MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE, AIC23\_data.both); //output 32 bit

return;

}

Header File content:

#define N\_FIR\_BP\_coeffs 81

short b\_FIR\_BP[N\_FIR\_BP\_coeffs]={

194, -152, -329, -19, 139, 4, 2, 48,

130, 125, -163, -205, -151, -342, 240, 857,

-109, -577, 468, 289, -725, -571, -210, 340,

1039, 413, -29, -55, -1288, -417, 1314, -1751,

-2143, 4351, 4115, -2359, -2070, -2105, -5384, 1979,

9920, 1979, -5384, -2105, -2070, -2359, 4115, 4351,

-2143, -1751, 1314, -417, -1288, -55, -29, 413,

1039, 340, -210, -571, -725, 289, 468, -577,

-109, 857, 240, -342, -151, -205, -163, 125,

130, 48, 2, 4, 139, -19, -329, -152,

194};

# Appendix 2.1: MATLAB implementation of LP and HP

%Desired passband cutoff frequency

OmegaP = 500;

%Desired stopband cutoff frequency

OmegaS = 1000;

%Desired passband ripple (linear)

DeltaP = 0.01;

%Desired stopband tolerance (dB)

DeltaSS = 40;

DeltaS = 10^(-DeltaSS/20);

% estimation of filter order

dev = [DeltaP DeltaS]; % deviation

f = [OmegaP OmegaS]; % frequency bands

FS = 8000; % frequency scaling (sampling frequency)

lp = [1 0]; % amplitudes

hp = [0 1]; % amplitudes

[N1,f1,m1,w1] = firpmord(f,lp,dev,FS);

[N2,f2,m2,w2] = firpmord(f,hp,dev,FS);

N1 = N1 + 2; % compensate for estimation error

% filter design by Remez algorithm

h\_lp = firpm(N1,f1,m1,w1); % impulse response

h\_hp = firpm(N2,f2,m2,w2); % impulse response

% frequency response

M = 2048; % number of frequency samples

[H1,W1] = freqz(h\_lp,1,M,FS);

[H2,W2] = freqz(h\_hp,1,M,FS);

%%Write LP filter's coefficients

% filnam = fopen('FIR\_LP.h', 'w'); % generate include-file

% fprintf(filnam,'#define N\_FIR\_LP\_coeffs %d\n', N+1);

% fprintf(filnam,'short b\_FIR\_LP[N\_FIR\_LP\_coeffs]={\n');

% j = 0;

% for i= 1:N+1

% fprintf(filnam,' %6.0d,', round(h\_lp(i)\*32768) );

% j = j + 1;

% if j >7

% fprintf(filnam, '\n');

% j = 0;

% end

% end

% fprintf(filnam,'};\n');

% fclose(filnam);

%%

%%Write HP filter's coefficients

% filnam = fopen('FIR\_HP.h', 'w'); % generate include-file

% fprintf(filnam,'#define N\_FIR\_HP\_coeffs %d\n', N+1);

% fprintf(filnam,'short b\_FIR\_HP[N\_FIR\_HP\_coeffs]={\n');

% j = 0;

% for i= 1:N+1

% fprintf(filnam,' %6.0d,', round(h\_hp(i)\*32768) );

% j = j + 1;

% if j >7

% fprintf(filnam, '\n');

% j = 0;

% end

% end

% fprintf(filnam,'};\n');

% fclose(filnam);

%impulse responses of LP

subplot(6,1,1), stem(0:N1,h\_lp), grid

xlabel('n'), ylabel('h[n]')

%axis([0 N1-1 -.2 .5]);

title('Impulse response of LP');

subplot(6,1,2), stem(0:N2,h\_hp), grid

xlabel('n'), ylabel('h[n]')

%axis([0 N2 -.2 .5]);

title('Impulse response of HP');

% frequency response of LP

subplot(6,1,3)

plot(W1,abs(H1)),axis([0 4000 0 1]); grid

xlabel('Hz'), ylabel('|H(e^{j\Omega})')

title(' Low Pass');

subplot(6,1,4) % attenuation

plot(W1,20\*log10(abs(H1))), axis([0 4000 -100 0]), grid

xlabel('Hz'), ylabel('|H(e^{j\Omega})|')

title(' Low Pass');

% frequency response of HP

subplot(6,1,5)

plot(W2,abs(H2)),axis([0 4000 0 1]); grid

xlabel('Hz'), ylabel('|H(e^{j\Omega})|')

title(' High Pass');

subplot(6,1,6) % attenuation

plot(W2,20\*log10(abs(H2))), axis([0 4000 -100 0]), grid

xlabel('Hz'), ylabel('|H(e^{j\Omega})|/dB')

title(' High Pass');

# Appendix 2.2: MATLAB implementation of BP

%Desired passband ripple (linear)

DeltaP = 0.01;

%Desired stopband tolerance (dB)

DeltaSS = 40;

DeltaS = 10^(-DeltaSS/20);

dev = [0.01 0.01 0.01 0.01 0.01]; % deviation

f = [800 1000 1500 1750 2000 2250 2500 2750]; % frequency bands

FS = 8000; % frequency scaling (sampling frequency)

m = [0 1 0 1 0]; % amplitudes

[N,f0,m0,w] = firpmord(f,m,dev,FS);

N = N + 2; % compensate for estimation error

h = firpm(N,f0,m0,w); % impulse response

% frequency response

M = 2048; % number of frequency samples

[H,f] = freqz(h,1,M,FS);

%%Write BP filter's coefficients

% filnam = fopen('FIR\_BP.h', 'w'); % generate include-file

% fprintf(filnam,'#define N\_FIR\_BP\_coeffs %d\n', N+1);

% fprintf(filnam,'short b\_FIR\_BP[N\_FIR\_BP\_coeffs]={\n');

% j = 0;

% for i= 1:N+1

% fprintf(filnam,' %6.0d,', round(h(i)\*32768) );

% j = j + 1;

% if j >7

% fprintf(filnam, '\n');

% j = 0;

% end

% end

% fprintf(filnam,'};\n');

% fclose(filnam);

% impulse responses

subplot(3,1,1), stem(0:N,h), grid

xlabel('n'), ylabel('h[n]')

axis([0 N -.2 .5]);

% frequency response

subplot(3,1,2), plot(f,abs(H))

axis([0 4000 0 1.2]);

grid

xlabel('Hz '), ylabel('|H(e^{j\Omega})|')

title('Aplitude response, absolute scale')

subplot(3,1,3) % attenuation

plot(f,20\*log10(abs(H))), axis([0 4000 -80 0]), grid

xlabel('Hz '), ylabel('|H(e^{j\Omega})|/dB ')

title('Aplitude response, in dB')

# Appendix 3: MATLAB implementation of Attachments F and G

figure('Name','Attachment F and G',...

'NumberTitle','off','Units','normal')

subplot(2,1,1), plot(t,x\_n)

grid

ylabel('x\_n')

subplot(2,1,2) % attenuation

plot(y\_n), grid

ylabel('y\_n')

# Appendix 4: MATLAB implementation of Attachments H and K

figure('Name','Attachment H and K',...

'NumberTitle','off','Units','normal')

subplot(2,1,1), plot(abs(fft(x\_n)))

grid

ylabel('|fft(x\_n)|')

subplot(2,1,2) % attenuation

plot(abs(fft(y\_n))), grid

ylabel('|fft(y\_n)|')