

**3.1.1 Lowpass and highpass filter designs**

All FIR filters used in this lab are designed with Matlab using functions *firpmord()* and *firpm()*.

The filter coefficients retrieved from function *firpm()* are written on files as short integers via function write\_coeffs(). The codes for the Matlab files used for this lab are presented in *Listing 1* at the end of the report.

*Attachment A:*

FIR filter specifications:

Fs = 8 kHz

Fpass = 500 Hz

Fstop = 1000 Hz

Dpass = 0.01

Minimum stopband attenuation: 40 dB

Dstop = 10^(-40/20) = 0.01

The amplitude response of the generated LP FIR filter is shown on Fig1.

The minimum stopband attenuation is at -40dB. The passband ripple is between

-0.01(-0.1dB) and 0.01(0.1dB) shown on Fig2 and Fig3.



Fig1. LP FIR filter amplitude response

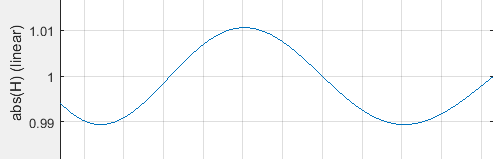
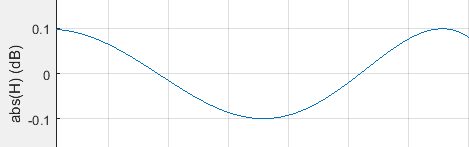


Fig.2 Passband ripple in dB Fig.3 Passband ripple linear

The short int filter coefficients are stored in file FIR\_LP.h:

#define N\_FIR\_coeffs 33

short h[N\_FIR\_coeffs]={

-119, 172, 258, 330, 297, 105, -237, -643,

-954, -981, -565, 355, 1705, 3267, 4731, 5773,

6150, 5773, 4731, 3267, 1705, 355, -565, -981,

-954, -643, -237, 105, 297, 330, 258, 172,

-119,};

*Attachment B:*

A HP FIR filter with the above specifications is designed and both the LP and HP amplitude responses are shown on Fig4.



Fig.4 LP and HP filter amplitude response

The short int HP filter coefficients are stored in file FIR\_HP.h:

#define N\_FIR\_coeffs 33

short h[N\_FIR\_coeffs]={

119, -172, -258, -330, -297, -105, 237, 643,

954, 981, 565, -355, -1705, -3267, -4731, -5773,

26618, -5773, -4731, -3267, -1705, -355, 565, 981,

954, 643, 237, -105, -297, -330, -258, -172,

119,};

*Attachment C:*

The impulse response of both filters is plotted and shown on Fig5.



Fig.5 LP and HP filter impulse response

Looking at the both impulse responses we can conclude that the LP and HP FIR filters that have the same specifications differ only in:

1. The magnitude of the central coefficient and
2. The sign in all the rest coefficients

Hence in order to convert from LP to HP and vice versa, the signs of all the filter coefficients but the middle one are inverted, and that middle coefficient is scaled with an appropriate factor.

**3.1.2 Bandpass filter design**

BP FIR filter specifications:

Fs = 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

The amplitude response of the BP filter is shown linear in Fig6 and in dB in Fig7

*Attachment D and E:*



Fig.6 BP filter amplitude response - linear



Fig.7 BP filter amplitude response - dB

A signal consisting of 5 cosine waves with amplitude 0.2 and different frequencies is generated:

f1 = 500;

f2 = 1250;

f3 = 2000;

f4 = 2500;

f5 = 3000;

t = 0:1/Fs:1;

x\_n = 0.2\*(cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t)+cos(2\*pi\*f3\*t)+cos(2\*pi\*f4\*t)+

+cos(2\*pi\*f5\*t))

*Attachment F and G:*

The signal is filtered with the generated BP filter using function filter() and both x\_n and the outputted signal y\_n are shown in Fig8.

y\_n = filter(h\_BP, 1, x\_n)



Fig.8 Input x\_n and output y\_n

*Attachment H and K:*

The FFT of both x\_n and y\_n are plotted and shown in Fig9.



Fig.9 FFT of x\_n and y\_n

As we can see the BP filter has passed only the frequencies 1250 Hz and 2500 Hz as they are in the filter’s pass band range. The rest frequencies presented in x\_n at 500, 2000 and 3000 Hz are filtered out as they are out of that range.

**3.2 Analog Transfer Characteristic of the DSK Board CODEC AIC23**

*Attachment L:*

For this task the get\_started.c program is loaded on the DSK Board and started with a sampling frequency of 8kHz. The amplitude response of the board is measured with the help of a spectrum analyzer and its output is shown in Fig10.

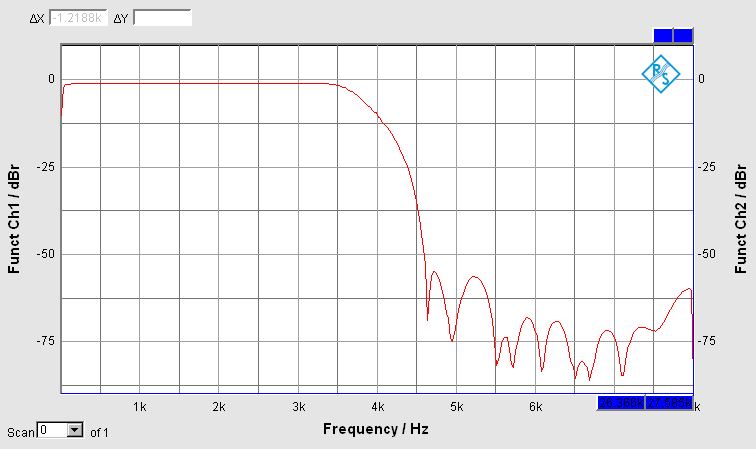


Fig.10 System amplitude response

It can be observed that after the Nyquist frequency which is 4 kHz in this case, the output gets distorted.

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

**3.3.1 Lowpass and Highpass filter**

*Attachment M:*

The procedure of implementing the LP FIR filter is as follows:

“Inside a FOR-loop, the sum of the products of the coefficients and

delayers b\_FIR\_LP[i] \* delays[i] (both short int) must be saved in a

32-bit integer variable FIR\_accu32. After the loop, the output signal

of the FIR lowpass filter is saved in a short int 16-bit variable

FIR\_out\_LP. FIR\_out\_LP is sent to the DAC for the left channel.”

The result of the multiplication of the delays (16 bits) and the filter coefficients (16 bits) is stored first in a 32 bits variable. This variable is then shifted to the right with 15 bits and casted to short(16 bits). This procedure is very efficient and advantageous with respect to noise and accuracy because any inaccuracy in the least significant 15 bits is discarded and does not affect the output.

*Attachment N:*

The code for the LP FIR filter - only the declarations and the ISR

*FIR\_LP.c*

#include "c6713dskinit.h"

#include "dsk6713.h"

#include <math.h>

#include "FIR\_LP.h"

#define LEFT 1

#define RIGHT 0

#ifdef SIMULATOR

MCBSP\_Handle DSK6713\_AIC23\_DATAHANDLE;

#else

extern MCBSP\_Handle DSK6713\_AIC23\_DATAHANDLE;

#endif

static Uint32 CODECEventId;

Uint32 fs=DSK6713\_AIC23\_FREQ\_8KHZ;

short buff\_delay[N\_FIR\_coeffs] = {0}; //initially set all delays to 0

int i, FIR\_accu32;

short FIR\_out\_LP;

union {

Uint32 both;

short channel[2];

} AIC23\_data;

//----------ISR-------------

interrupt void intser\_McBSP1()

{

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

for(i = N\_FIR\_coeffs -1; i>0; i--) //update delay buffer

buff\_delay[i] = buff\_delay[i-1];

buff\_delay[0] = AIC23\_data.channel[LEFT]; //new sample at position 0

FIR\_accu32 = 0; // reset to 0

for (i=0; i<N\_FIR\_coeffs; i++) //calculate the output

FIR\_accu32 += h[i]\*buff\_delay[i]; //store it in 32 bit variable

FIR\_out\_BP = (short)(FIR\_accu32 >> 15); //shift to the right and convert to short

AIC23\_data.channel[LEFT] = FIR\_out\_LP; //assign the filter output to left channel

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

return;

}

On Fig11 is shown the filter output from the spectrum analyser from 0 to Fs/2, in this case 4kHz.

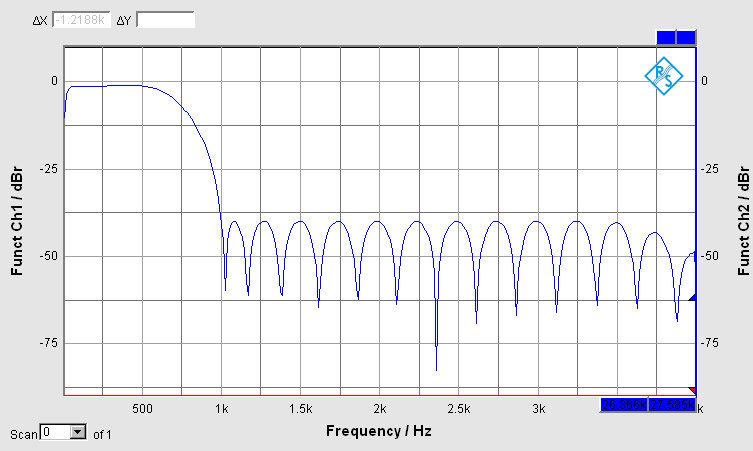


Fig.11 LP FIR filter amplitude response from spectrum analyzer

For implementing the HP filter the header file FIR\_HP.h is included and the 16 bit output FIR\_out\_HP is sent to the left channel. On Fig.12 is shown the amplitude response of the HP filter.

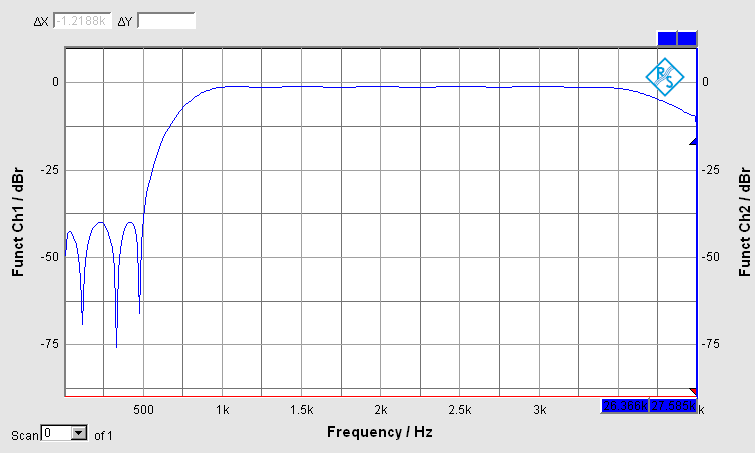


Fig.12 HP FIR filter amplitude response from spectrum analyzer

*Attachment O:*

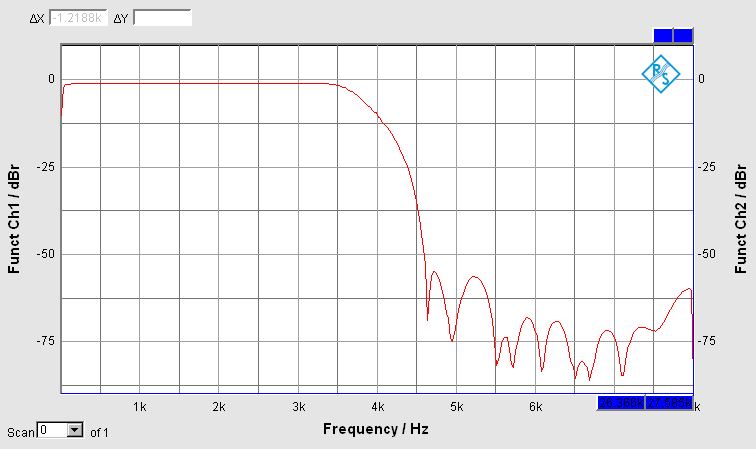
 

Fig.4’ Matlab simulation up to Fs Fig.10 System response – Spectrum analyzer up to Fs

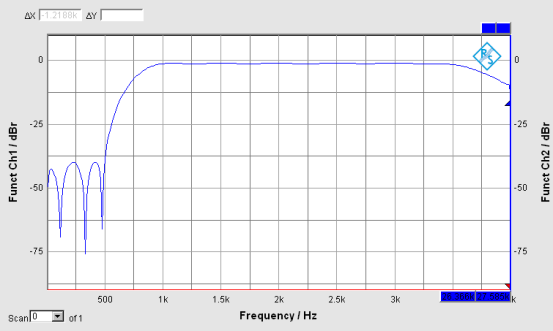
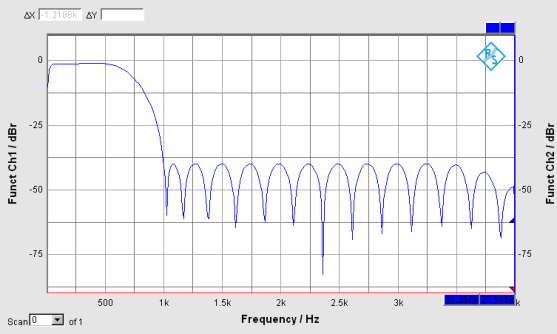


Fig.11 LPF - Spectrum analyzer up to Fs/2 Fig.12 HPF - Spectrum analyzer up to Fs/2

In the Matlab simulation on Fig.4’ we can see that the output level stays constant through the whole range in the stopband and passband of the filter amplitude response. In our lab experiment the situation is different. Due to the amplitude response of the real life system, shown on Fig.10, the signal gets attenuated as it gets close to Fs/2 and it will be completely distorted after that point.

If we increase the frequency band even further up to Fs, this distortion can clearly be seen as on Fig.13

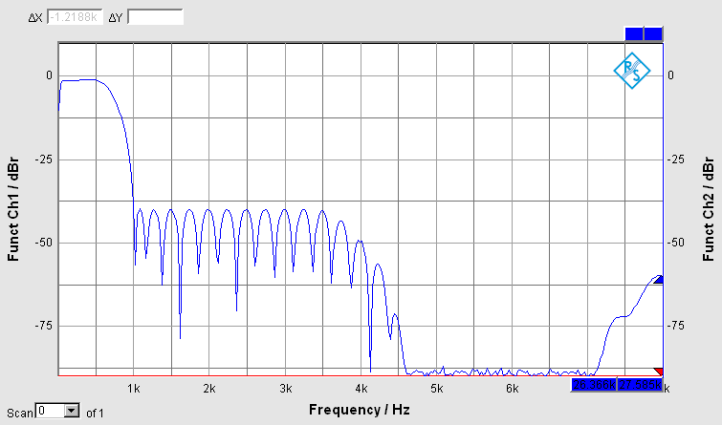


Fig.13 LPF - Spectrum analyzer up to Fs

**3.3.2 Bandpass Filter**

*Attachment P:*

The header file FIR\_BP.h containing the BP filter coefficients was added to the program and the output is shown on the spectrum analyzer on Fig.14

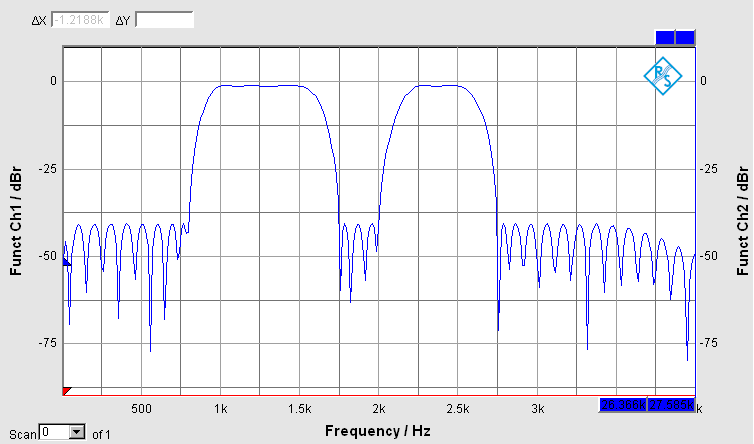


Fig.14 BP FIR filter amplitude response from spectrum analyzer

THEFORCE.wav file is fed into the DSK Board input and the filtered output is listened on the speakers. The filter’s bandwidth (1000-1500Hz and 2250-2500Hz) is inside the human’s voice bandwidth (300-3400Hz) and the sampling rate is 8 kHz that is why the output sounds like a bad quality telephone connection.

The filtered output is shown on Fig.15

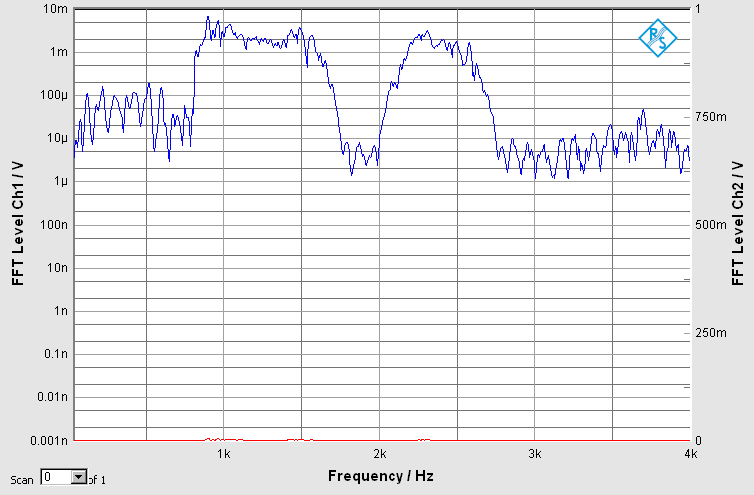


Fig.15 FFT of filtered output of THEFORCE.wav file

Conclusion:

In this lab session we implemented, analyzed and tested 3 types FIR filters: LP, HP and BP.

We explored how a HP FIR filter can be implemented from the LP filter coefficients with the same specifications and vice versa. A method, robust to noise and rounding errors was used in the ISR of the program to calculate the filter output. With the help of the spectrum analyzer, the amplitude response of the filters and the effect of the system was visualized, investigated and compared against the Matlab simulations. All results were documented and approved.

Listing 1:

Matlab files:

*sp\_fir\_lab\_ass1.m*

clear all;

close all;

% all frequency values are in Hz.

Fs = 8000; % sampling frequency

Fpass = 500; % passband frequency

Fstop = 1000; % stopband frequency

Dpass = 0.01; % passband ripple

Dstop\_dB = 40; % stopband attenuation in dB

Dstop = 10^(-Dstop\_dB/20); %stopband attenuation linear

% design a LP FIR filter with the above specifications

[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, Dstop]);

%design a HP FIR filter with the same specifications

%note that the Fpass and Fstop are flipped

[N\_hp, Fo\_hp, Ao\_hp, W\_hp] = firpmord([Fpass, Fstop]/(Fs/2), [0 1], [Dstop, Dpass]);

%make the filter order and hence the number of coefficients equal

while N < N\_hp

N = N+1;

end

while N\_hp < N

N\_hp = N\_hp+1;

end

% calculate the coefficients/impulse response.

h\_FIR\_LP = firpm(N, Fo, Ao, W);

h\_FIR\_HP = firpm(N\_hp, Fo\_hp, Ao\_hp, W\_hp);

%frequency responce

[H,f] = freqz(h\_FIR\_LP, 1, 2048);

[H1,f1] = freqz(h\_FIR\_HP, 1, 2048);

%plot amplitude response in dB

figure('Name','Equiripple Linear-Phase FIR Filter - Amplitude reasponse');

%plot(f/pi, 20\*log10(abs(H))); grid on; title('LP FIR Filter Amplitude response');

%xlabel(Normalized Frequency (\times\pi rad/sample)');ylabel('abs(H) (dB)');

plot(f/pi, 20\*log10(abs(H)), f1/pi, 20\*log10(abs(H1))); grid on;

title('LP and HP FIR Filter Amplitude response'); ylabel('abs(H) (dB)');

xlabel(Normalized Frequency (\times\pi rad/sample)'); legend('LPF', 'HPF');

%plot impulse response for both filters on the same graph

figure('Name','Equiripple Linear-Phase FIR Filter - Impulse response');

stem(0:N, h\_FIR\_LP);hold on

stem(0:N\_hp, h\_FIR\_HP);title('Impulse responses of both HP and LP filters'); legend('LPF','HPF');

%write the coefficients as short ints on files

write\_coeffs(N, h\_FIR\_LP, 'l');

write\_coeffs(N\_hp, h\_FIR\_HP, 'h');

*sp\_fir\_lab\_ass2.m*

clear all;

close all;

% all frequency values are in Hz.

Fs = 8000;

Fstop1 = 800;

Fpass1 = 1000;

Fpass2 = 1500;

Fstop2 = 1750;

Fstop3 = 2000;

Fpass3 = 2250;

Fpass4 = 2500;

Fstop4 = 2750;

Dstop\_dB = 40; %Stopband attenuation in dB

Dstop = 10^(-Dstop\_dB/20); % Stopband attenuation linear

Dpass = 0.01; % Passband Ripple

% design a BP FIR filter with the above specifications.

[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2 Fstop3 Fpass3 Fpass4 Fstop4]/(Fs/2), [0 1 0 1 0], [Dstop Dpass Dstop Dpass Dstop]);

% calculate the coefficients/impulse response.

h = firpm(N, Fo, Ao, W);

%amplitude response

[H,f] = freqz(h,1,2048);

figure;

plot(f/pi, 20\*log10(abs(H)));grid on; title('BP FIR Filter Amplitude response'); xlabel(Normalized Frequency (\times\pi rad/sample)');ylabel('abs(H) (dB)');

figure;

plot(f/pi, abs(H));grid on; title('BP FIR Filter Amplitude response');

xlabel(Normalized Frequency (\times\pi rad/sample)');ylabel('abs(H) (linear)');

%write the coefficients as short ints on files

write\_coeffs(N, h, 'bp');

%generate signal

f1 = 500;

f2 = 1250;

f3 = 2000;

f4 = 2500;

f5 = 3000;

t = 0:1/Fs:1;

x\_n = 0.2\*(cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t)+cos(2\*pi\*f3\*t)+cos(2\*pi\*f4\*t)+cos(2\*pi\*f5\*t));

%filter the signal with the constructed BP FIR filter

y\_n = filter(h, 1, x\_n);

%plot original and filtered signals

figure('Name','Generated signal');

subplot(2,1,1);

plot(t, x\_n);axis([0 .02 -.5 1]);xlabel('Time (s)');ylabel('x\_n');

subplot(2,1,2);

plot(t, y\_n);axis([0 .02 -.5 1]);xlabel('Time (s)');ylabel('y\_n');

%plot FFT of x\_n and y\_n

figure;

subplot(2,1,1);

plot(0:Fs, abs(fft(x\_n)));title('FFT of generated signal');

xlabel('Frequency (Hz)');ylabel('FFT(x\_n)');

subplot(2,1,2);

plot(0:Fs, abs(fft(y\_n)));title('FFT of filtered signal');

xlabel('Frequency (Hz)');ylabel('FFT(y\_n)');

*write\_coeffs.m*

%function writes the coefficients on a header file

%input arguments are: Order of filter; h - vector containing the

%coefficients (usually returned from "firpm" function); Type of filter

function write\_coeffs(Filter\_order, h, Filter\_type)

N\_coeffs = Filter\_order + 1;

% create header file FIR\_??.h depending on the type specified

switch Filter\_type

case 'l'

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

case 'h'

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

case 'bp'

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

case 'bs'

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

otherwise

return;

end

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

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

j = 0;

for i= 1:N\_coeffs

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);