

|  |  |  |
| --- | --- | --- |
| Student group:  1 | Professor’s comments: | Record editor:  Muhammad Adnan |
| Date:  11.06.18 | Group members:  Ievgenii Nudga  Muhammad Adnan |
| Professor:  Prof.Dr. Sauvagerd |

Contents

1. Pre-lab part
2. Attachment A……………………………………………………………………………………… page 2
3. Attachment B……………………………………………………………………………………… page 3
4. Attachment D……………………………………………………………………………………… page 3
5. Attachment E……………………………………………………………………………………… page 4
6. MATLAB code…………………………………………………………………………………….. page 4
7. Lab part
8. Figure 6: Ellip LP and Chebyshev1 LP..………………………………………………… page 10
9. Figure 7: Ellip HP.……………………………………………………………………………….. page 10
10. Figure 8: Ellip Band Stop…………………………………………………………………….. page 10
11. C code………………………………………………………………………………………………… page 11

**Pre-lab part**

**Attachment A**

The specifications are:

* passband edge frequency: 1950 Hz,
* stopband edge frequency: 3000 Hz,
* maximum passband ripple: 0.01,
* minimum stopband attenuation: 40 dB,
* sampling frequency: 8 kHz.

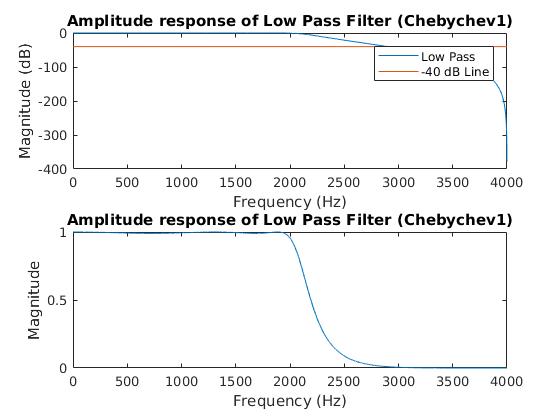


Figure 1: Amplitude responses of LP filter (Cheb1)

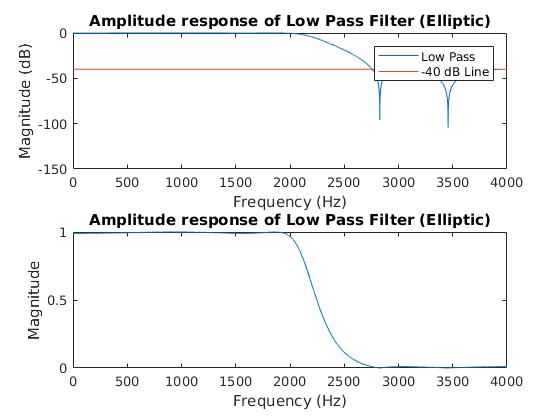


Figure 2: Amplitude responses of LP filter (Ellip)

**Attachment B**

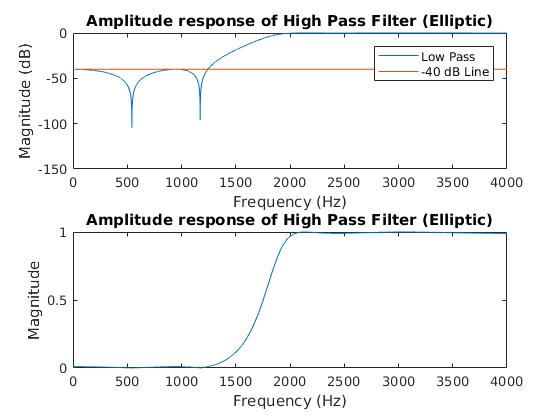


Figure 3: Amplitude responses of HP filter (Cheb1)

**Attachment D**

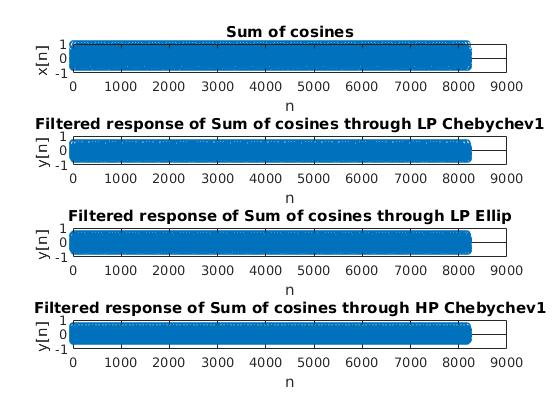


Figure 4: Input Signal and Output Signal through respective Filters

**Attachment E**

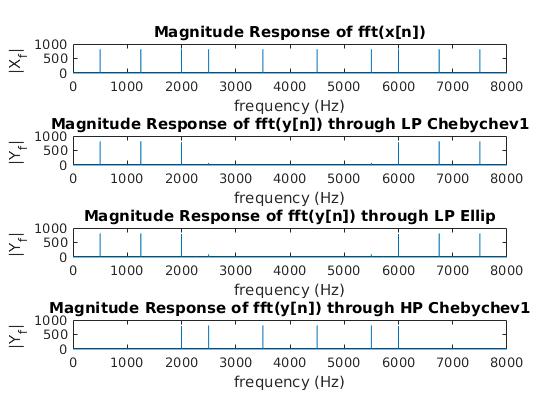


Figure 5: FFT of Input Signal and Output Signal after filtering

**MATLAB code**

%------------------ DSP Lab 3 HP and LP IIR Filters ------------------

clear all;

%------------------ Filter Specifications ------------------

Fs = 8000; %sampling frequency in Hz (8 KHz)

Fpass = 1950 ; %passband edge frequency in Hz

Fstop = 3000 ; %stopband edge frequency in Hz

Rp = 0.01; %ripple in passband (dB)

Rp = -20\*log10(1-Rp); % passband deviation in dB

Rs = 40 ; %ripple in stopband (dB)

Wpass = Fpass / (Fs/2); %normalized edge frequency of passband in rad /s

Wstop = Fstop / (Fs/2); %normalized edge frequency of stopband in rad /s

%----------------------------------------------------------------------------------

%------------------ Chebychev1 IIR Low Pass ------------------

[N\_LP\_cheb1, Wn\_LP\_cheb1] = cheb1ord(Wpass, Wstop, Rp, Rs); %Order estimation using Chebychev1

[b\_LP\_cheb1, a\_LP\_cheb1] = cheby1(N\_LP\_cheb1, Rp, Wn\_LP\_cheb1); %coefficients using Chebychev1

[sos\_LP\_cheb1,gn\_LP\_cheb1] = tf2sos(b\_LP\_cheb1, a\_LP\_cheb1);

sos\_LP\_cheb1(:,1:3) = (nthroot(gn\_LP\_cheb1, size(sos\_LP\_cheb1, 1) ) ) \* sos\_LP\_cheb1(:,1:3); %sos after gain divided equally

%---------------------------------------------------------------------------------

%------------------------- Elliptic IIR Low Pass -------------------------

[N\_LP\_ellip, Wn\_LP\_ellip] = ellipord(Wpass, Wstop, Rp, Rs); %Order estimation using Elliptic

[b\_LP\_ellip, a\_LP\_ellip] = ellip (N\_LP\_ellip, Rp, Rs, Wn\_LP\_ellip); %coefficients using Elliptic

[sos\_LP\_ellip,gn\_LP\_ellip] = tf2sos(b\_LP\_ellip, a\_LP\_ellip);

sos\_LP\_ellip(:,1:3) = (nthroot(gn\_LP\_ellip, size(sos\_LP\_ellip, 1) ) ) \* sos\_LP\_ellip(:,1:3); %sos after gain divided equally

%----------------------------------------------------------------------------------

%------------------------- Elliptic IIR High Pass by making every odd index negative -------------------------

N\_HP\_ellip = N\_LP\_ellip;

b\_HP\_ellip = b\_LP\_ellip;

a\_HP\_ellip = a\_LP\_ellip;

b\_HP\_ellip(2:2:length(b\_LP\_ellip)) = -b\_HP\_ellip(2:2:length(b\_LP\_ellip));

a\_HP\_ellip(2:2:length(a\_LP\_ellip)) = -a\_HP\_ellip(2:2:length(a\_LP\_ellip));

[sos\_HP\_ellip,gn\_HP\_ellip] = tf2sos(b\_HP\_ellip,a\_HP\_ellip);

sos\_HP\_ellip(:,1:3) = (nthroot(gn\_HP\_ellip, size(sos\_HP\_ellip, 1) ) ) \* sos\_HP\_ellip(:,1:3); %sos after gain divided equally

%----------------------------------------------------------------------------------

%-------------------------- x\_n made up of 5 cosine and filtering it through Band Pass -----------------------------------

M = 8192; % # of frequency sample

f = [500 1250 2000 2500 3500]; % frequencies of cos

w\_norm = 2\*pi\*f / Fs;%normalized angular freq

n = 0:M-1;

x\_n = 0;

for i = 1:length(w\_norm)

x\_n = x\_n + cos(n\*w\_norm(i));

end

x\_n = 0.2 \* x\_n;

y\_n\_HP\_ellip = filter(b\_HP\_ellip,a\_HP\_ellip, x\_n);

y\_n\_LP\_ellip = filter(b\_LP\_ellip,a\_LP\_ellip, x\_n);

y\_n\_LP\_cheb1 = filter(b\_LP\_cheb1,a\_LP\_cheb1, x\_n);

%------------------------------------------------------------------------------------

%-------------------------- Frequency Response -----------------------------------

[H\_LP\_cheb1, F\_LP\_cheb1] = freqz(b\_LP\_cheb1, a\_LP\_cheb1, M, Fs); %low pass IIR frequency response of cheb1

[H\_LP\_ellip, F\_LP\_ellip] = freqz(b\_LP\_ellip, a\_LP\_ellip, M, Fs); %low pass IIR frequency response of ellip

[H\_HP\_ellip, F\_HP\_ellip] = freqz(b\_HP\_ellip, a\_HP\_ellip, M, Fs); %high pass IIR frequency response of ellip

%----------------------------------------------------------------------------------

%-------------------------- Plotting -----------------------------------

%-----Low Pass amplitude response dB (Attachment A)-------

figure(1);

subplot(2,1,1);

plot(F\_LP\_cheb1, mag2db(abs(H\_LP\_cheb1)), F\_LP\_cheb1, -40\*ones(1, length(F\_LP\_cheb1))); %dB scale response of cheb1

title('Amplitude response of Low Pass Filter (Chebychev1)')

xlabel('Frequency (Hz)'), ylabel('Magnitude (dB)')

legend('Low Pass', '-40 dB Line');

subplot(2,1,2);

plot(F\_LP\_cheb1, (abs(H\_LP\_cheb1))); %absolute scale response of cheb1

title('Amplitude response of Low Pass Filter (Chebychev1)')

xlabel('Frequency (Hz)'), ylabel('Magnitude')

figure(2);

subplot(2,1,1);

plot(F\_LP\_ellip, mag2db(abs(H\_LP\_ellip)), F\_LP\_ellip, -40\*ones(1, length(F\_LP\_ellip))); %dB scale response of Low Pass Elliptic

title('Amplitude response of Low Pass Filter (Elliptic)')

xlabel('Frequency (Hz)'), ylabel('Magnitude (dB)')

legend('Low Pass', '-40 dB Line');

subplot(2,1,2);

plot(F\_LP\_ellip, (abs(H\_LP\_ellip))); %absolute scale response of Low Pass Elliptic

title('Amplitude response of Low Pass Filter (Elliptic)')

xlabel('Frequency (Hz)'), ylabel('Magnitude')

%---------------------------------------------------------

%-----Low Pass amplitude response dB (Attachment B)-------

figure(3);

subplot(2,1,1);

plot(F\_HP\_ellip, mag2db(abs(H\_HP\_ellip)), F\_HP\_ellip, -40\*ones(1, length(F\_HP\_ellip))); %dB scale response of High pass Elliptic

title('Amplitude response of High Pass Filter (Elliptic)')

xlabel('Frequency (Hz)'), ylabel('Magnitude (dB)')

legend('Low Pass', '-40 dB Line');

subplot(2,1,2);

plot(F\_HP\_ellip, (abs(H\_HP\_ellip))); %absolute scale response of High pass Elliptic

title('Amplitude response of High Pass Filter (Elliptic)')

xlabel('Frequency (Hz)'), ylabel('Magnitude')

%-----------------------------------------------------------------------------------

%--------------------------Writing coeffcients of Low Pass Chebychev1 Filter-----------------------

N\_IIR\_LP = N\_LP\_cheb1;

num\_IIR = b\_LP\_cheb1;

den\_IIR = a\_LP\_cheb1;

% create header file IIR\_LP\_ellip\_cheby1.h

% assumption: IIR filter coefficients are stored in num\_IIR, den\_IIR and that

% the filter has a degree of N\_IIR\_LP

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* This stores only the coeffcients of Low Pass Chebychev1 filter ONLY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filnam = fopen('IIR\_LP\_ellip\_cheby1.h', 'w');

% generate include-file

fprintf(filnam,'#define N\_IIR\_LP\_CHEBY1 %d\n', N\_IIR\_LP);

fprintf(filnam,'#define NO\_OF\_ROWS\_CHEBY1 %d\n', size(sos\_LP\_cheb1, 1));

%fprintf(filnam,'short num\_IIR\_LP\_cheby1[N\_IIR\_LP]={\n');

%j = 0;

%for i= 1:N\_IIR\_LP + 1

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

% j = j + 1;

% if j >7

% fprintf(filnam, '\n');

% j = 0;

% end

%end

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

% use a similar approach for denominator

%fprintf(filnam,'short den\_IIR\_LP\_cheby1[N\_IIR\_LP]={\n');

%j = 0;

%for i= 1:N\_IIR\_LP + 1

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

% j = j + 1;

% if j >7

% fprintf(filnam, '\n');

% j = 0;

% end

%end

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

% use a similar approach for SOS

fprintf(filnam,'short sos\_iir\_lp\_cheby1[NO\_OF\_ROWS\*6]={\n');

for i= 1:size(sos\_LP\_cheb1, 1)

for j = 1:6

fprintf(filnam,' %6.0d,', round(sos\_LP\_cheb1(i,j)\*32768) );

end

fprintf(filnam, '\n');

end

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

fclose(filnam);

%----------------------------------------------------------------------------------------------------------

%--------------------------Writing coeffcients of Low Pass Elliptic Filter-----------------------

N\_IIR\_LP = N\_LP\_ellip;

% create header file IIR\_ellip\_LP.h

% assumption: IIR filter coefficients are stored in num\_IIR, den\_IIR and that

% the filter has a degree of N\_IIR\_LP

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* This stores only the coeffcients of Low Pass Elliptic filter ONLY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filnam = fopen('IIR\_ellip\_LP.h', 'w');

% generate include-file

fprintf(filnam,'#define N\_IIR\_LP\_ELLIP %d\n', N\_IIR\_LP);

fprintf(filnam,'#define NO\_OF\_ROWS\_LP\_ELLIP %d\n', size(sos\_LP\_ellip, 1));

% use a similar approach for SOS

fprintf(filnam,'short sos\_iir\_lp\_ellip[NO\_OF\_ROWS\_ELLIP\*6]={\n');

for i= 1:size(sos\_LP\_ellip, 1)

for j= 1:6

fprintf(filnam,' %6.0d,', round(sos\_LP\_ellip(i,j)\*32768) );

end

fprintf(filnam, '\n');

end

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

fclose(filnam);

%-----------------------------------------------------------------------------------------------------------

%--------------------------Writing coeffcients of High Pass Elliptic Filter-----------------------

N\_IIR\_HP = N\_LP\_ellip;

% create header file IIR\_LP\_ellip\_cheby1.h

% assumption: IIR filter coefficients are stored in num\_IIR, den\_IIR and that

% the filter has a degree of N\_IIR\_LP

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* This stores only the coeffcients of High Pass Elliptic filter ONLY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

filnam = fopen('IIR\_ellip\_HP.h', 'w');

% generate include-file

fprintf(filnam,'#define N\_IIR\_HP\_ELLIP %d\n', N\_IIR\_HP);

fprintf(filnam,'#define NO\_OF\_ROWS\_HP\_ELLIP %d\n', size(sos\_HP\_ellip, 1));

%fprintf(filnam,'short num\_IIR\_HP\_ellip[N\_IIR\_HP]={\n');

% use a similar approach for SOS

fprintf(filnam,'short sos\_iir\_hp\_ellip[NO\_OF\_ROWS\_ELLIP\*6]={\n');

j = 1;

for i= 1:size(sos\_HP\_ellip, 1)

for j= 1:6

fprintf(filnam,' %6.0d,', round(sos\_HP\_ellip(i,j)\*32768) );

end

fprintf(filnam, '\n');

end

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

fclose(filnam);

**Lab part**

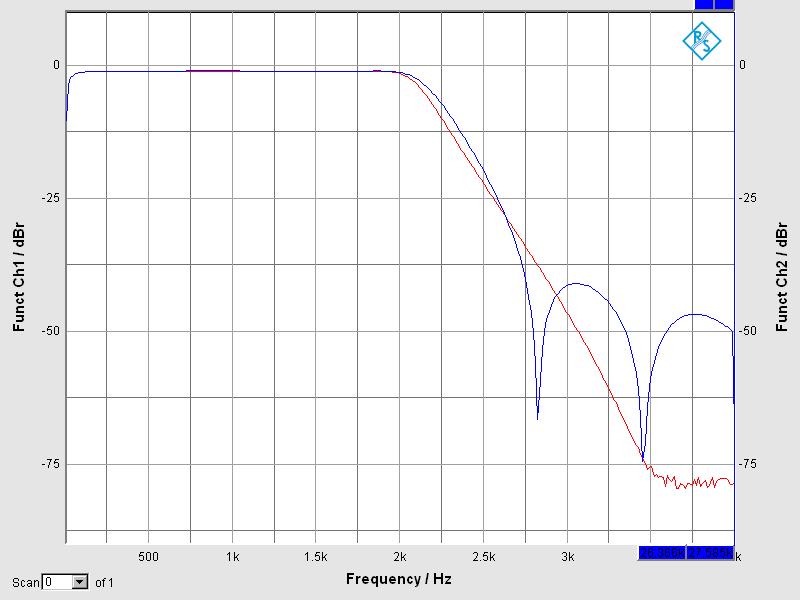


Figure 6: Ellip and cheb1 (both LP)

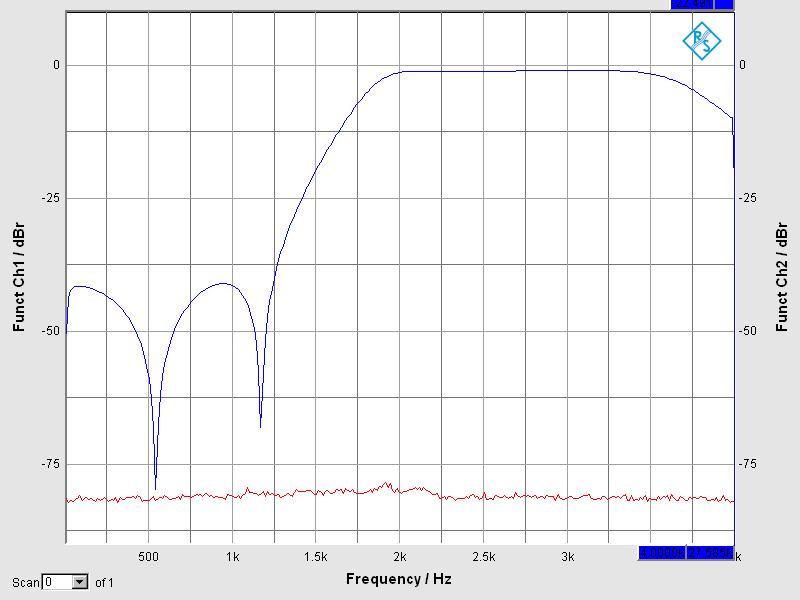


Figure 7: Ellip HP (blue line)

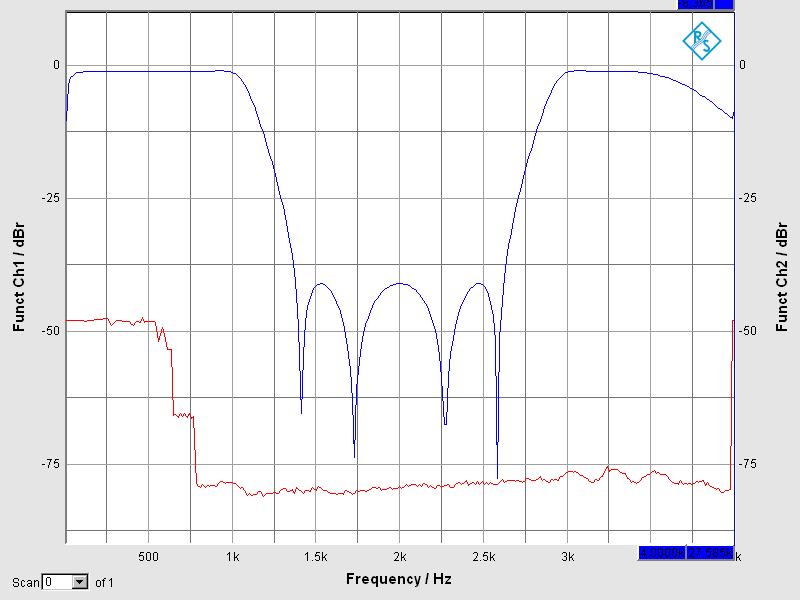


Figure 8: Ellip band Stop (blue line)

For this task, we perform the Filter Transformation to get Band Stop filter using Low Pass filter coefficients. Since we know we simply put z = z2 in z domain. This is equivalent to increase the number of coefficients to twice and make every odd coefficient 0. This was done in the C-Code. Since z = z2, therefore we can say that Fs decreases to half. This compresses the LP filter response, which gives us above band pass result.

The **C code** is listed below (after it there are also the IIR\_ellip\_LP.h, IIR\_LP\_ellip\_cheby1.h and IIR\_ellip\_HP.h listings).

//-----------------------------------------------------------

// Digital Signal Processing Lab

// Testprogram read/write

//

// AIC23 version

//

// Filename : get\_started.c

// Authors: Muhammad Adnan, Ievgenii Nudga, 11.06.2018

//

// Author Svg 8-Jan-07

//

// version 1 : modified 27-Nov-08, Kup

// version 2 : 31-Jan-10, JR

//#define SIMULATOR

// for usage of input MIC\_IN and output HEADPHONE with DC coupling:

//#define MIC\_IN

#include "c6713dskinit.h" //codec-DSK support file

#include "dsk6713.h"

#include <math.h> //math library

#include "IIR\_ellip\_LP.h"

#include "IIR\_LP\_ellip\_cheby1.h"

#include "IIR\_ellip\_HP.h"

#define LEFT 1

#define RIGHT 0

#define BUFLEN 1000

// external beim DSK-Board, hier zu deklarieren fьr Simulator:

#ifdef SIMULATOR

MCBSP\_Handle DSK6713\_AIC23\_DATAHANDLE;

#else

extern MCBSP\_Handle DSK6713\_AIC23\_DATAHANDLE;

#endif

static Uint32 CODECEventId;

Uint32 fs=DSK6713\_AIC23\_FREQ\_8KHZ; //for sampling frequency

//Uint32 fs; //for sampling frequency

short int x\_n\_el, x\_n\_cheb1;

short int i, y\_n\_el, y\_n\_cheb1;

//int T1\_el[NO\_OF\_ROWS\_LP\_ELLIP], T2\_el[NO\_OF\_ROWS\_LP\_ELLIP], accu32\_el;

int T1\_el[NO\_OF\_ROWS\_LP\_ELLIP], T2\_el[NO\_OF\_ROWS\_LP\_ELLIP], T3\_el[NO\_OF\_ROWS\_LP\_ELLIP], T4\_el[NO\_OF\_ROWS\_LP\_ELLIP] , accu32\_el;

int T1\_cheb1[NO\_OF\_ROWS\_CHEBY1], T2\_cheb1[NO\_OF\_ROWS\_CHEBY1], T3\_cheb1[NO\_OF\_ROWS\_CHEBY1], T4\_cheb1[NO\_OF\_ROWS\_CHEBY1],accu32\_ch1;

int switch\_filters = 1;

// int switch\_filters = 2;

union {

Uint32 both;

short channel[2];

} AIC23\_data;

void ellip\_LP()

{

for(i = 0; i < NO\_OF\_ROWS\_LP\_ELLIP; i++)

{

accu32\_el = sos\_iir\_lp\_ellip[6\*i+0] \* x\_n\_el + T1\_el[i];

y\_n\_el = (short) (accu32\_el >> 15);

T1\_el[i] = sos\_iir\_lp\_ellip[6\*i+1] \* x\_n\_el - sos\_iir\_lp\_ellip[6\*i+4]\*y\_n\_el + T2\_el[i];

T2\_el[i] = sos\_iir\_lp\_ellip[6\*i+2] \* x\_n\_el - sos\_iir\_lp\_ellip[6\*i+5]\*y\_n\_el;

x\_n\_el = y\_n\_el;

}

AIC23\_data.channel[LEFT] = y\_n\_el;

}

void ellip\_HP()

{

for(i = 0; i < NO\_OF\_ROWS\_HP\_ELLIP; i++)

{

accu32\_el = sos\_iir\_hp\_ellip[6\*i+0] \* x\_n\_el + T1\_el[i];

y\_n\_el = (short) (accu32\_el >> 15);

T1\_el[i] = sos\_iir\_hp\_ellip[6\*i+1] \* x\_n\_el - sos\_iir\_hp\_ellip[6\*i+4]\*y\_n\_el + T2\_el[i];

T2\_el[i] = sos\_iir\_hp\_ellip[6\*i+2] \* x\_n\_el - sos\_iir\_hp\_ellip[6\*i+5]\*y\_n\_el;

x\_n\_el = y\_n\_el;

}

AIC23\_data.channel[LEFT] = y\_n\_el;

}

void cheb1\_LP()

{

for(i = 0; i < NO\_OF\_ROWS\_CHEBY1; i++)

{

accu32\_ch1 = sos\_iir\_lp\_cheby1[6\*i+0] \* x\_n\_cheb1 + T1\_cheb1[i];

y\_n\_cheb1 = (short) (accu32\_ch1 >> 15);

T1\_cheb1[i] = sos\_iir\_lp\_cheby1[6\*i+1] \* x\_n\_cheb1 - sos\_iir\_lp\_cheby1[6\*i+4]\*y\_n\_cheb1 + T2\_cheb1[i];

T2\_cheb1[i] = sos\_iir\_lp\_cheby1[6\*i+2] \* x\_n\_cheb1 - sos\_iir\_lp\_cheby1[6\*i+5]\*y\_n\_cheb1;

x\_n\_cheb1 = y\_n\_cheb1;

}

AIC23\_data.channel[RIGHT] = y\_n\_cheb1;

}

void ellip\_LP\_2T()

{

for(i = 0; i < NO\_OF\_ROWS\_LP\_ELLIP; i++)

{

accu32\_el = sos\_iir\_lp\_ellip[6\*i+0] \* x\_n\_el + T1\_el[i];

y\_n\_el = (short) (accu32\_el >> 15);

T1\_el[i] = T3\_el[i];

T3\_el[i] = sos\_iir\_lp\_ellip[6\*i+1] \* x\_n\_el - sos\_iir\_lp\_ellip[6\*i+4]\*y\_n\_el + T2\_el[i];

T2\_el[i] = T4\_el[i];

T4\_el[i] = sos\_iir\_lp\_ellip[6\*i+2] \* x\_n\_el - sos\_iir\_lp\_ellip[6\*i+5]\*y\_n\_el;

x\_n\_el = y\_n\_el;

}

AIC23\_data.channel[LEFT] = y\_n\_el;

}

interrupt void intser\_McBSP1()

{

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

// buffer monitoring input signal, reset count if BUFLEN is reached,

// then input buffer is full

x\_n\_el = AIC23\_data.channel[LEFT];

x\_n\_cheb1 = AIC23\_data.channel[RIGHT];

if(switch\_filters == 1)

{

//ellip\_LP();

//cheb1\_LP();

//ellip\_HP();

ellip\_LP\_2T();

}

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

return;

}

///////////////////////////////////////////////////////////////////

void main()

{

IRQ\_globalDisable(); //disable interrupts

#ifndef SIMULATOR

DSK6713\_init(); //call BSL to init DSK-EMIF,PLL)

#ifdef MIC\_IN

config.regs[4] = 0x14;

config.regs[5] = 0x1;

#endif

hAIC23\_handle=DSK6713\_AIC23\_openCodec(0, &config);//handle(pointer) to codec

DSK6713\_AIC23\_setFreq(hAIC23\_handle, fs); //set sample rate

#else // Nur fьr Simulator:

DSK6713\_AIC23\_DATAHANDLE= MCBSP\_open(MCBSP\_DEV1, MCBSP\_OPEN\_RESET);

#endif

MCBSP\_config(DSK6713\_AIC23\_DATAHANDLE,&AIC23CfgData);//interface 32 bits toAIC23

MCBSP\_start(DSK6713\_AIC23\_DATAHANDLE, MCBSP\_XMIT\_START | MCBSP\_RCV\_START |

MCBSP\_SRGR\_START | MCBSP\_SRGR\_FRAMESYNC, 220);//start data channel again

CODECEventId= MCBSP\_getXmtEventId(DSK6713\_AIC23\_DATAHANDLE);//McBSP1 Xmit

IRQ\_map(CODECEventId, 5); //map McBSP1 Xmit to INT5

IRQ\_reset(CODECEventId); //reset codec INT5

IRQ\_globalEnable(); //globally enable interrupts

IRQ\_nmiEnable(); //enable NMI interrupt

IRQ\_enable(CODECEventId); //enable CODEC eventXmit INT5

IRQ\_set(CODECEventId); //manually start the first interrupt

while(1); //infinite loop

}

**IIR\_ellip\_LP.h**

#define N\_IIR\_LP\_ELLIP 4

#define NO\_OF\_ROWS\_LP\_ELLIP 2

short sos\_iir\_lp\_ellip[NO\_OF\_ROWS\_LP\_ELLIP\*6]={

12286, 22373, 12286, 32767, -6000, 4173,

12286, 14874, 12286, 32767, 5391, 22296,

};

**IIR\_LP\_ellip\_cheby1.h**

#define N\_IIR\_LP\_CHEBY1 5

#define NO\_OF\_ROWS\_CHEBY1 3

short sos\_iir\_lp\_cheby1[NO\_OF\_ROWS\_CHEBY1\*6]={

11104, 11094, 0, 32767, -9981, 0,

11104, 22226, 11121, 32767, -10537, 9847,

11104, 22202, 11098, 32767, 3218, 23955,

};

**IIR\_ellip\_HP.h**

#define N\_IIR\_HP\_ELLIP 4

#define NO\_OF\_ROWS\_HP\_ELLIP 2

short sos\_iir\_hp\_ellip[NO\_OF\_ROWS\_HP\_ELLIP\*6]={

12286, -22373, 12286, 32767, 6000, 4173,

12286, -14874, 12286, 32767, -5391, 22296,

};

**Conclusions**

In this lab we have gotten familiar with IIR filter designing in MATLAB and on DSP board. One of the most crucial points of the lab is to calculate the coefficients for all required filters and then to figure in C-program how to split, i.e. filter of order 6 into 3 second-order sections. The reason is that in DSP we must never ever realize IIR filters if the degree is more than 2 using the coefficients of general transfer function.