AN14627

Implementation of IIR filters on i.MX RT685 Hifi4 DSP with NatureDSP APIs

Rev. 1.0 — 22 April 2025

Application note

Document information

Information	Content
Keywords	AN14627, RT685, RT595, RT798, NatureDSP, IIR, IIR filters, infinite impulse response, filter, biquad, spectrum
Abstract	This document describes the procedure of implementing the IIR filters on RT685 Hifi4 DSP with NatureDSP APIs, selecting the NatureDSP API, and adjusting filter coefficients to achieve the best signal quality.

1 Introduction

This document gives an overview of the digital filters and explains the procedure to implement the infinite impulse response (IIR) filters on i.MX RT685 HiFi4 digital signal processor (DSP) core with the NatureDSP application programming interface (API).

The purpose of this document is to explain the following:

- Selection of NatureDSP API
- Adjust the filter coefficients to achieve the best signal quality

This document does not focus on DSP theory and mathematics. For more details, see INTRODUCTION TO DIGITAL FILTERS.

The NatureDSP library also has the versions for i.MX RT595 Fusion F1 DSP, i.MX RT798 HiFi1 DSP, and i.MX RT798 HiFi4 DSP. The user can also use this document when developing the i.MX RT595 and i.MX RT798.

2 Digital filter fundamentals

A digital filter is a system that performs the mathematical operations on a sampled, discrete-time signal to reduce or enhance certain aspects of that signal. The following are the digital filters:

- Low-pass (LP) filters or LPF select the low frequencies up to the cut-off frequency fc and attenuate frequencies higher than fc. Also, a resonance can amplify the frequencies around the fc.
- High-pass (HP) filters or HPF select the frequencies higher than fc and attenuate frequencies below fc, possibly with a resonance around the fc.
- Band-pass (BP) filters select the frequencies between a lower cut-off frequency fc(Low) and a higher cut-off frequency fc(High). The frequencies below the fc(Low) and the frequencies higher than the fc(High) are attenuated.
- Band-reject (BR) filters attenuate frequencies between a lower cut-off frequency fc(Low) and a higher cut-off frequency fc(High). The frequencies below the fc(Low) and the frequencies higher than the fc(High) are passed.
- All-pass (AP) filters pass all the frequencies, but modify the phase of the input signal.

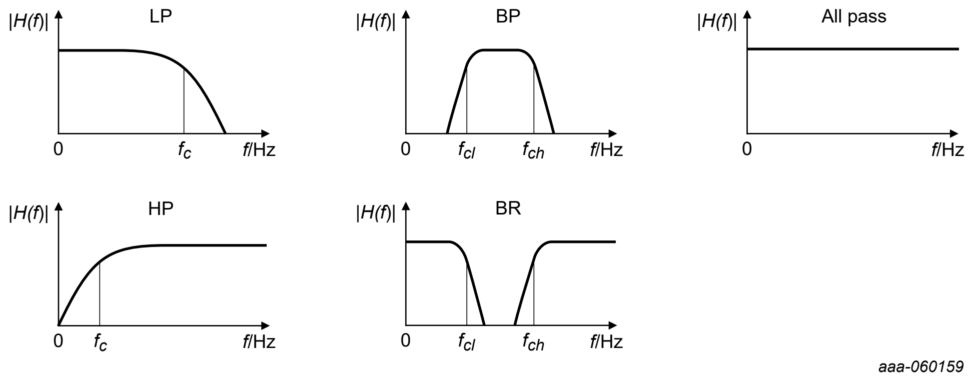


Figure 1. Filter types of LP, HP, BP, BR, and AP

The most commonly used digital filters for realizing the LP, HP, and so on, are the IIR and finite impulse response (FIR) filters. The IIR filter is also called the canonical filter. Besides the canonical filter, the following are other classifications of digital filters:

- State variable filter
- Convolution filter
- APF-based filter

- FIR filter, and so on

This document covers IIR filters only.

3 IIR filters

The IIR filters are the infinite impulse response filters. Unlike FIR filters, these filters have the feedback, which is a recursive part of a filter.

3.1 Direct form

When the transfer function or the equivalent difference equation defines the basic structure of the IIR filter directly, it is called the direct form structure. Figure 2 shows the direct form 1 (DF1) structure.

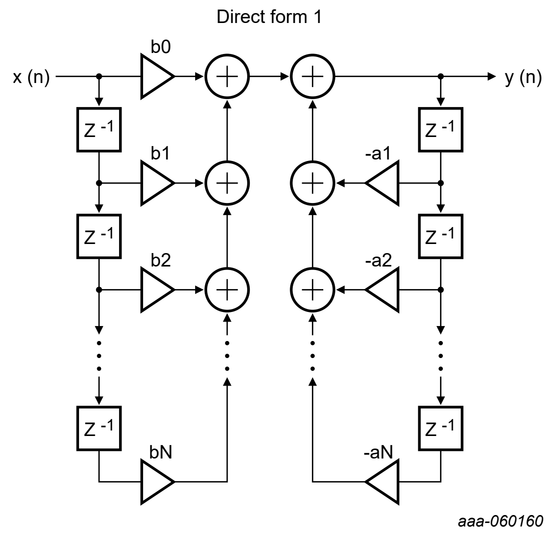


Figure 2. A N order IIR filter in DF1

To get the direct form 2 (DF2) structure, swap the first block and the second block of the structure in Figure 2. It has the same transfer function and generates the same result as DF1.

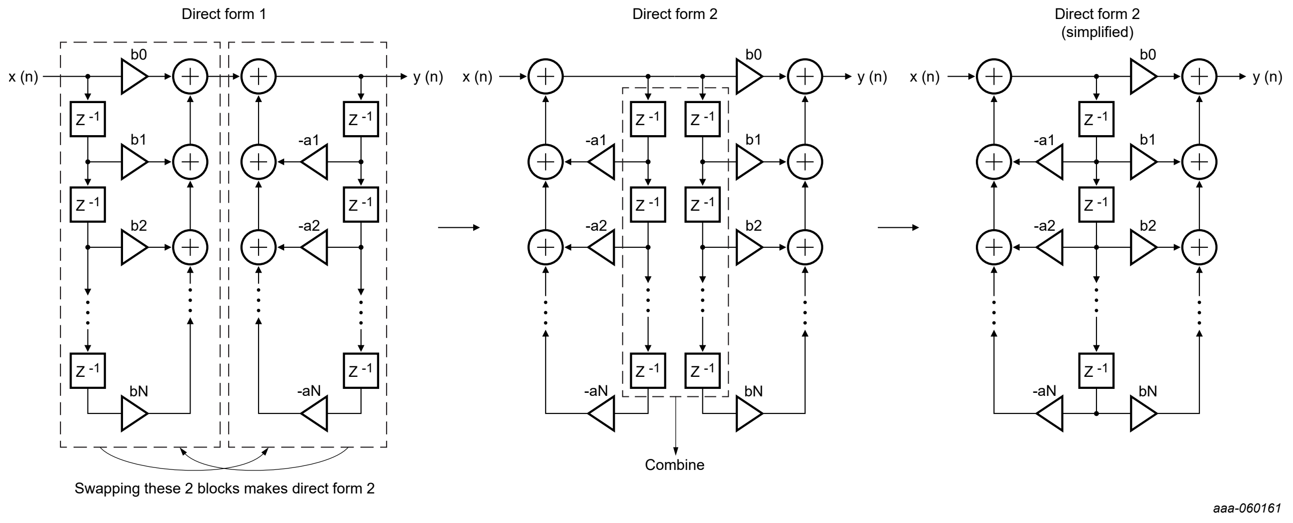


Figure 3. DF1 changes to DF2

To get the transposed form, reverse the signal flow, which does not change the transfer function. The filter transposition can also be called flow graph reversal, and transposing a filter does not alter its transfer function. The transpose of the filter is straightforward: reverse the direction of all signal paths, change signal branch-points to summers and change summers to branch-points.

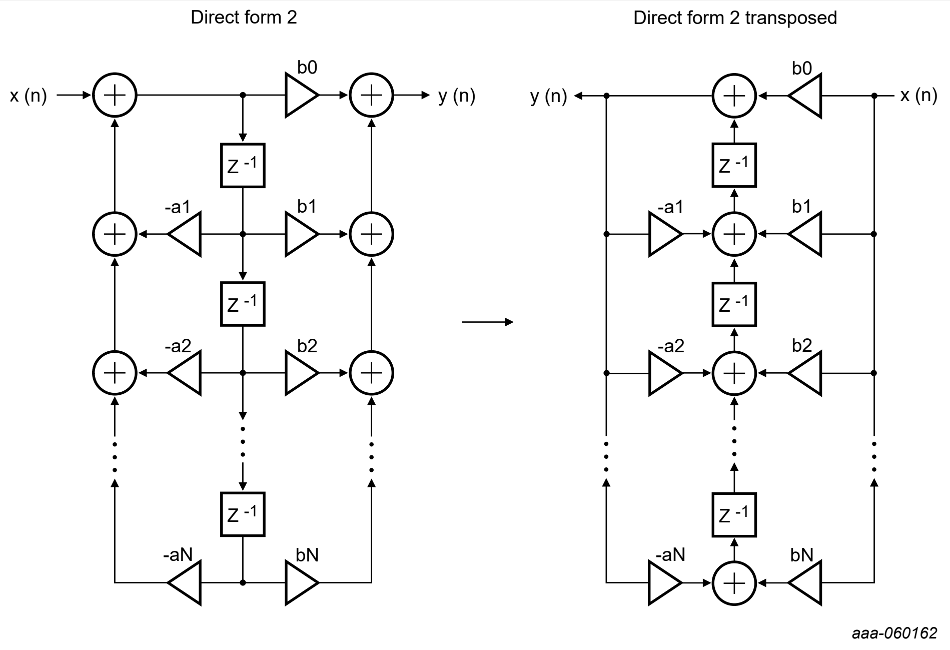


Figure 4. DF2 changes to direct form 2 transposed (DF2t)

When the IIR filter only uses the one sample delayed x(n-1), y(n-1), or the intermediate xh(n-1), it is called a 1st order IIR filter. When the IIR filter uses one more delay sample x(n-2), y(n-2), or the intermediate xh(n-2), it is called a 2nd order IIR filter.

3.2 Cascaded form

As the filter order goes higher, the direct-form IIR filter often becomes more sensitive to errors, which the coefficient quantization and the computational precision limits introduce. The high-order transfer function can be divided into lower second order sections (SOS) or the filter stages (biquad), which makes the filter less sensitive to errors.

Direct form $\longrightarrow$ Cascaded second order sections

$$H(z) = \frac{\sum_{k=0}^{M-1} b_k * z^{-k}}{1 + \sum_{k=1}^{N-1} a_k * z^{-k}} \longrightarrow H(z) = \prod_{k=1}^{N-1} \frac{b_{0k} + b_{1k}*z^{-1} + b_{2k}*z^{-1} + \cdots + b_{Nk}*z^{-1}}{1 + a_{1k}*z^{-1} + a_{2k}*z^{-1} + \cdots + a_{Nk}*z^{-1}} = H_1(z) * H_2(z) * \cdots * H_N(z)$$

$$Y(z) = H(z) * X(z) \longrightarrow Y(z) = H_1(z) * H_2(z) * \cdots * H_N(z) * X(z)$$

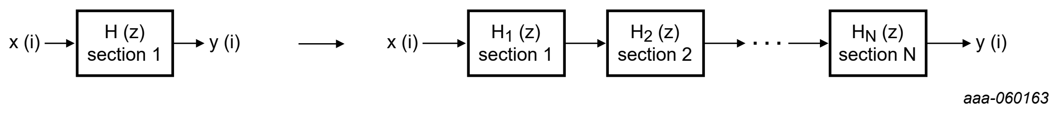
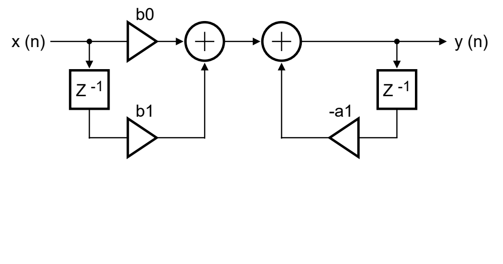
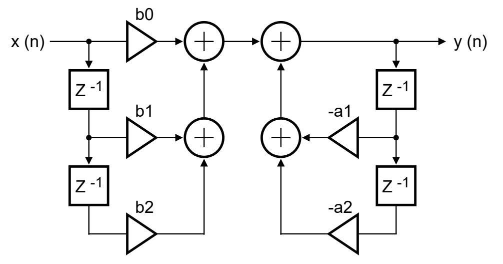
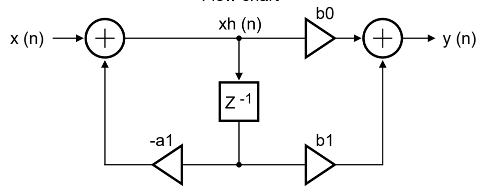
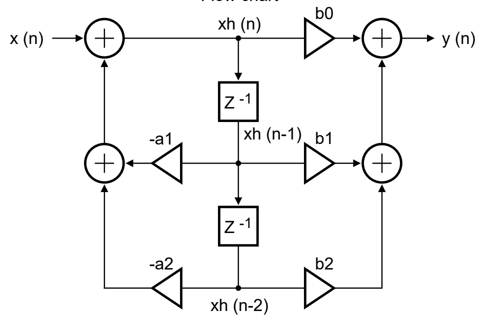
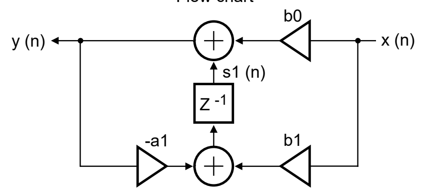
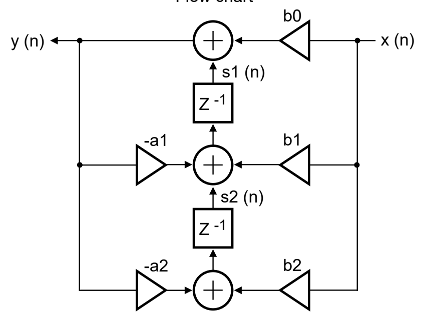


Figure 5. Direct form converts to equal cascaded SOS

Figure 6 shows all the basic information of a 1st order or a 2nd order IIR filter, which includes the following:

- Transfer function

- Flowchart
- Difference equation and the related NatureDSP APIs

1st order IIR a partial biquad a partial SOS (Second order section)	**2nd order IIR** a biquad an SOS (Second order section)
Transfer function $H(z) = \frac{b0 + b1{*}z^{-1}}{1 + a1{*}z^{-1}}$	Transfer function $H(z) = \frac{b0 + b1{*}z^{-1} + b2{*}z^{-1}}{1 + a1{*}z^{-1} + a2{*}z^{-1}}$
Direct form 1 ------- NatureDSP APIs: void bqri irXXXX_df1(...):	
Flow chart 	Flow chart  x (n) b0 y (n) Z -1 b1 -a1 Z -1 Z -1 b2 -a2 Z -1
Difference equation y (n) = b0*x (n) + b1*x (n-1) - a1*y (n-1)	Difference equation y (n) = b0*x (n) + b1*x (n-1) + b2*x (n-2) - a1*y (n-1) - a2*y (n-2)
Direct form 2 ------- NatureDSP APIs: void bqri irXXXX_df2(...):	
Flow chart 	Flow chart  x (n) xh (n) b0 y (n) Z -1 xh (n-1) -a1 b1 Z -1 -a2 b2 xh (n-2)
Difference equation xh (n) = x (n) - a1*xh (n-1) y (n) = b0*xh (n) + b1*xh (n-1)	Difference equation xh (n) = x (n) - a1*xh (n-1) - a2*xh (n-2) y (n) = b0*xh (n) + b1*xh (n-1) + b2*xh (n-2)
Direct form 2 transposed ------- NatureDSP APIs: void bqri irXXXX_df2t(...):	
Flow chart 	Flow chart  y (n) b0 x (n) s1 (n) Z -1 -a1 b1 s2 (n) Z -1 -a2 b2
Difference equation y (n) = s1 (n) + b0*x (n) s1 (n+1) = b1*x (n) - a1*y (n)	Difference equation y (n) = s1 (n) + b0*x (n) s1 (n+1) = s2 (n) + b1*x (n) - a1*y (n) s2 (n+1) = b2*x (n) - a2*y (n)

aaa-060162

Figure 6. 1st and 2nd order IIR filter in different forms

4 Biquad

The term “biquad” is short for “bi-quadratic”, and is a common name for a two-pole, two-zero, second order digital filter. Biquad is the same meaning as 2nd order IIR, which uses five coefficients (b0, b1, b2, a1, a2). The structure of a biquad can be DF1, DF2, or DF2t. Figure 7 shows the three forms of a biquad.

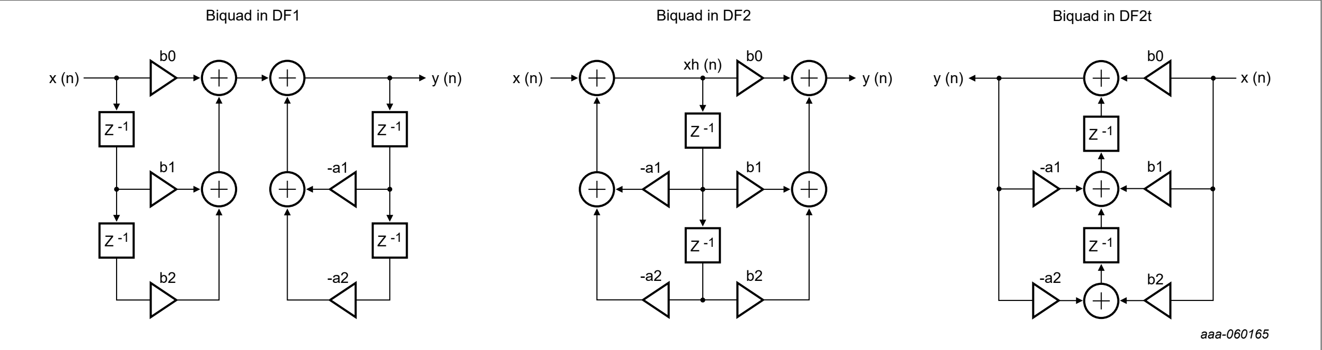


Figure 7. Biquad in DF1, DF2, and DF2t

The reason to emphasize on biquad is that the NatureDSP library IIR filter APIs are based on biquad. Whether the user needs a 1st order or 20th order IIR filter, convert the filter into a series of cascaded SOS. Then, call the biquad APIs of the NatureDSP library.

Table 1 and Table 2 shows the formulas for calculating the following biquad coefficients:

- For 1st order b0, b1, and a1
- For 2nd order b0, b1, b2, a1, a2

Table 1. 1st order IIR: LP, HP, and AP

Filters	b_0	b_1	a_1
LP	$\frac{K}{K+1}$	$\frac{K}{K+1}$	$\frac{K-1}{K+1}$
HP	$\frac{1}{K+1}$	$-\frac{1}{K+1}$	$\frac{K-1}{K+1}$
AP	$\frac{K-1}{K+1}$	1	$\frac{K-1}{K+1}$

Table 2. 2ndorder IIR: LP, HP, BP, BR, and AP

Filters	b_0	b_1	b_2	a_1	a_2
LP	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2K^2Q}{K^2Q+K+Q}$	$\frac{K^2Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
HP	$\frac{Q}{K^2Q+K+Q}$	$-\frac{2Q}{K^2Q+K+Q}$	$\frac{Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
BP	$\frac{K}{K^2Q+K+Q}$	0	$-\frac{K}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$
BR	$\frac{Q(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{Q(1+K^2)}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

Table 2. 2nd order IIR: LP, HP, BP, BR, and AP*...continued*

Filters	b_0	b_1	b_2	a_1	a_2
AP	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	1	$\frac{2Q(K^2-1)}{K^2Q+K+Q}$	$\frac{K^2Q-K+Q}{K^2Q+K+Q}$

Note: *In the Table 1 and Table 2, K = tan(Pi*fc/fs). Where fs is the sampling frequency and fc is the cut-off or center frequency and Q is the factor value.*

There are other methods for calculating IIR coefficients such as Butterworth, Chebyshev I, Chebyshev II, and elliptic. To let tune the expected filter sophistically, these methods involve more parameters, which are order numbers, pass-band amplitude, stop band amplitude, and so on. Using the MATLAB filter design tool is an easy way to generate the coefficients of variates of filter types.

Note: *When using the fixed-point fractional type IIR APIs, the following must be tuned and converted to the integer format:*

- *The generated SOS coefficients*
- *The generated SOS scale values*

5 NatureDSP library

The Cadence NatureDSP library is a collection of highly optimized DSP functions for the HiFi4 processor, which supports both fixed-point and single precision floating data types.

5.1 Overview

The library consists of the following functions:

- Mathematics functions
- Complex mathematics functions
- Vector operation functions
- IIR filter functions
- FIR filter functions
- Matrix functions
- FFT/DCT functions
- Image-processing functions, and so on

This document does not cover every NatureDSP APIs, but only aim to analyze the major issues to consider when selecting and calling an IIR API.

5.2 IIR filters API types

Table 3 shows the four variants of the IIR filters in the library.

Table 3. NatureDSP IIR filter types

Type	Description
16X16	16-bits data, 16-bits coefficients, 16-bits intermediate stage outputs (DF1, DF1 stereo, DF2 form)
32X16	32-bits data, 16-bits coefficients, 32-bits intermediate stage outputs (DF1, DF1 stereo, DF2 form)
32X32	32-bits data, 32-bits coefficients, 32-bits intermediate stage outputs (DF1, DF1 stereo, DF2 form)

Table 3. NatureDSP IIR filter types*...continued*

Type	Description
f	Floating point (DF1, DF1 stereo, DF2, and DF2t). Requires VFPU/ SFPU core option

- 16*16 means that the audio sample is in 16-bits fixed integer (signed short int), filter coefficients are in 16-bits fixed fractional format (singed short int). The same interpretation applies for 32*16 and 32*32.
- f means that the audio sample and the filter coefficients are in float format.

All the four variants have the DF1 and DF2 type filter APIs but the floating point variant also supports the DF2t.

The following two implementations for all the IIR filters are:

- With delay
- Delayless

This document focuses on the following variants:

- 32*32 type with delay
- Float type with delay

6 Fixed point and floating point multiplication

The digital signal processing can be either in the fixed point or the floating point. The i.MX RT685 HiFi4 processor supports both the fixed point and the floating point.

For example: To apply gain to an audio signal frame, the audio signal is represented in an int32 type array, and the gain value is set to 0.4.

6.1 Fixed point

Convert the gain value 0.4 to an int32 type. The conversion is done assuming that the int32 range (-2^{31}) ~ (2^{31-1}) corresponds to the float range -1.0 ~ 1.0. To get the converted result 858993459, perform 0.4 * 2^{31}.

Suppose that the input signal value X is 1000. The calculation 1000 * 858993459 >> 31 = 399 realizes the equivalent calculation to 1000 * 0.4 = 400. In this example, the gain value 0.4 is converted to a fractional 0.31 format. 0 means 0 digits for an integer part. 31 means 31 bits for the fractional part and there is 1 bit for the sign.

If the gain value is 1.4, then the conversion is done assuming that the int32 range (-2^{31}) ~ (2^{31-1}) matches the float range -2.0 ~ 2.0. The *2^{30} converts a gain value 1.4 to a fractional 1.30 format. Table 4 details how a floating value be converted to a fractional type for fixed-point multiplication.

In the NatureDSP library, the 32-bits fractional values are called F32 values and the 16-bits fractional values are called F16 values.

6.2 Floating point

The signal sample values must be converted from int32 type to float. The conversion is done assuming that the int32 range (-2^{31}) ~ (2^{31-1}) corresponds to the float range -1.0 ~ 1.0. To get the converted float type of the signal value, perform "signal int32 value/2^{31}". Then the gain processing is simply multiplying two float type values to get the float result.

Table 4. Illustration of fixed-point fractional coefficient values

Float coefficien value	Range	Format of F32	Factor to use for converting to F32	F32 value after the conversion: float value * factor	F32 value in Hex after the conversion	Sign bits	int part (in BIN)	Fractional part (in HEX)
0.1	± 1.0	0.31	2^{31}	214748365	CCCCCCC	0	X	CCCCCCC
0.9	± 1.0	0.31	2^{31}	1932735283	73333333	0	X	73333332
1.9	± 2.0	1.30	2^{30}	2040109466	79999999	0	1	39999997
3.9	± 4.0	2.29	2^{29}	2093796557	7CCCCCCC	0	11	1CCCCCC8
7.9	± 8.0	3.28	2^{28}	2120640102	7E666666	0	111	E66665E
- 0.1	± 1.0	0.31	2^{31}	- 214748365	F3333334	1	X	73333334
- 0.9	± 1.0	0.31	2^{31}	- 1932735283	8CCCCCCE	1	X	CCCCCCE
- 1.9	± 2.0	1.30	2^{30}	- 2040109466	86666669	1	0	6666669
- 3.9	± 4.0	2.29	2^{29}	- 2093796557	83333338	1	00	3333338
- 7.9	± 8.0	3.28	2^{28}	- 2120640102	819999A2	1	000	19999A2

7 RT685 demonstration project of NatureDSP IIR

This document is accompanied with the following:

- The MCUXpresso CM33 project: Rt685_NatureDspFilter_McuPrg
- The Xtensa Xplorer HiFi4 project: Rt685_NatureDspFilter_DspPrg
- Some MATLAB scripts: Used for generating and adjusting the filter coefficients

The MCUXpresso CM33 and Xtensa Xplorer HiFi4 projects create an executable program that filters the input audio signal from the universal serial bus (USB) audio host. It also streams the result signals back to the USB audio host.

The hardware for running this demo is the MIMXRT685-EVK.

Note: *The IDE for building the CM33 project is MCUXpresso IDE v11.10.0.*

The IDE for building the Hifi4 project is Xtensa Xplorer Version 10.1.11.3000 and the tool chain is nxp_rt600_RI23_11_newlib.

For more details on IDE and toolchain installation, see “Getting Started with Xplorer for EVK-MIMXRT685.pdf” in RT685 SDK.

When building the projects, due to header file inclusion, place the CM33 and DSP projects in the same folder outside the workspace folder.

7.1 Signal flowchart

Figure 8 shows the signal flowchart implemented in i.MX RT685 HiFi4 processor.

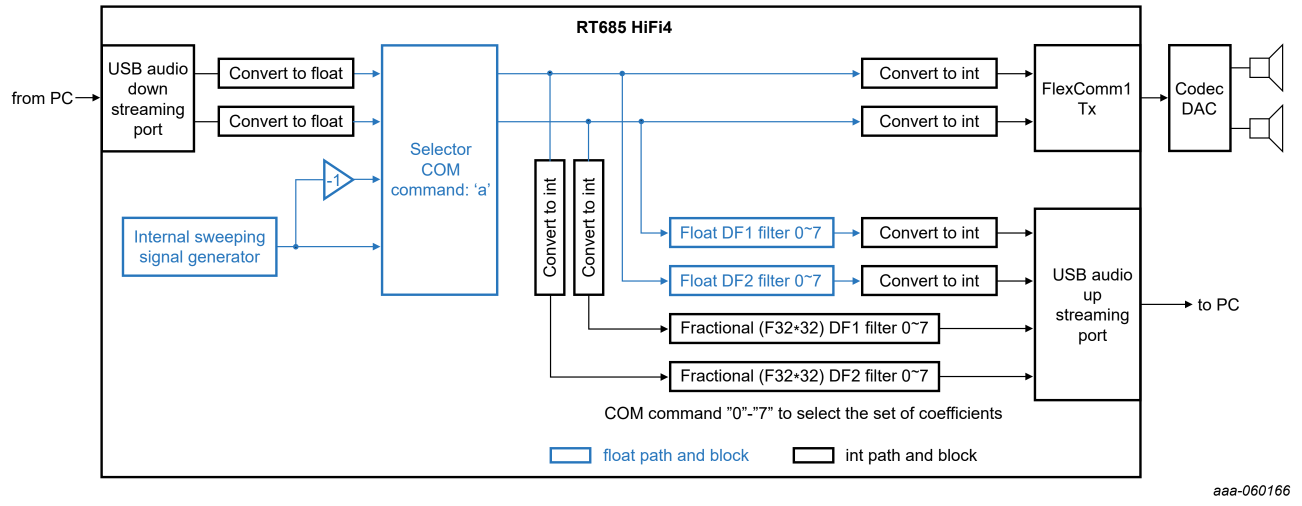


Figure 8. RT685 HiFi4 DSP internal signal flow

In the signal flowchart, the audio source is USB host down-streaming audio (two channels) or the internal sweeping tone generator (two channels). The selected one pair of audio source is duplicated to two pairs. The Pair1 (channels 1 and 2 in float) is directly I2S streamed out to the onboard codec, and the floating filters DF1/DF2 filters it. The fractional filters DF1/DF2 filters the Pair 2 (channels 3 and 4 in integer). All the four channels of the filtered audio are finally up-streamed to the USB audio host. The "float to int" and "int to float" converting blocks are inserted at all the necessary places in the flowchart.

7.2 Eight groups of filters

To demonstrate the varieties of filter coefficients, the HiFi4 DSP projects implemented the following:

- Eight float DF1 filters
- Eight float DF2 filters
- Eight fractional (F32*32) DF1 filters
- Eight fractional (F32*32) DF2 filters

When MIMXRT685-EVK is connected to the personal computer (PC), open a COM window to type in the simple commands and watch the printed information from MIMXRT685-EVK. Type "a" or "A" to switch the audio source between USB down-streaming audio or internally generated sweeping tone. Type "0~7" to select one of the eight testing filters.

Table 5. Filters implemented in the demo project

	Float D1 type	Float D2 type	F32*32 D1 type	F32*32 D2 type
Filter0	20 Hz LP, second order, Butterworth coefficients	Same as float D1 type filter0	Same as float D1 type filter0	Same as float D1 type filter0
Filter1	3000 Hz HP, second order, Elliptic coefficients	Same as float D1 type filter1	Same as float D1 type filter1	Same as float D1 type filter1
Filter2	100 Hz HP, eight order, Butterworth coefficients	Same as float D1 type filter2	Same as float D1 type filter2	Same as float D1 type filter2
Filter3	7800 Hz HP, 12th order, Elliptic coefficients	Same as float D1 type filter3	Same as float D1 type filter3	Same as float D1 type filter3

Table 5. Filters implemented in the demo project...*continued*

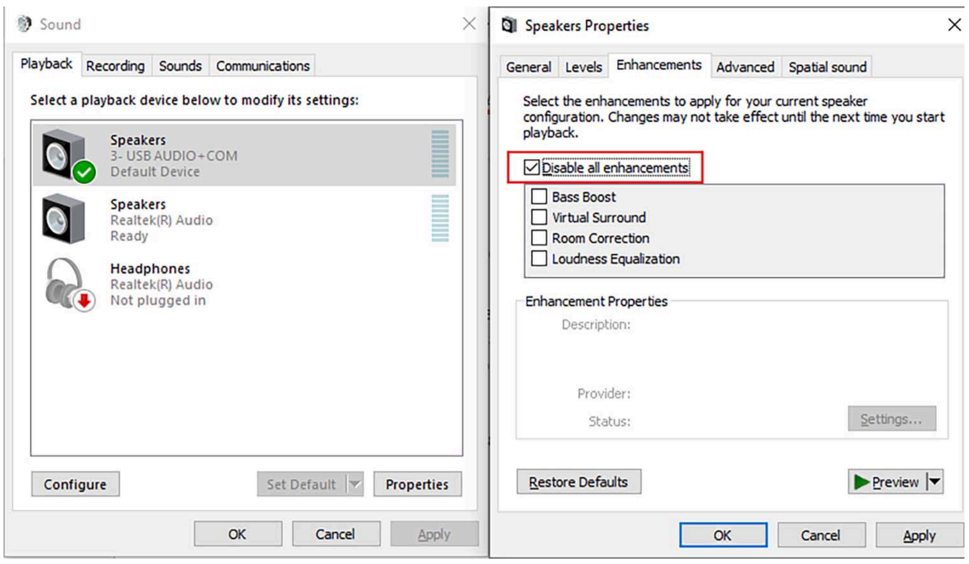
	Float D1 type	Float D2 type	F32*32 D1 type	F32*32 D2 type
Filter4	111 Hz LP, second order, standard biquad coefficients	Same as float D1 type filter4	Same as float D1 type filter4	Same as float D1 type filter4
Filter5	111 Hz HP, second order, standard biquad coefficients	Same as float D1 type filter5	Same as float D1 type filter5	Same as float D1 type filter5
Filter6	2222 Hz BP, second order, standard biquad coefficients	Same as float D1 type filter6	Same as float D1 type filter6	Same as float D1 type filter6
Filter7	2222 Hz BR, second order, standard biquad coefficients	Same as float D1 type filter7	Same as float D1 type filter7	Same as float D1 type filter7

The filters 0, 1, 2, and 3 coefficients are generated in the MATLAB filter design tool. The filters 4, 5, 6, and 7 coefficients are generated in the HiFi4 code, see Table 2.

7.3 Record the filtered audio

With the existing demo projects, the user could build and download the program to the MIMXRT685-EVK to start a quick test. Connect the MIMXRT685-EVK with a PC and ensure that the "USB AUDIO+COM" speaker device enhancements are disabled. For more details, see Figure 9.

The recording enhancements are not enabled by default because the recording interface from the MIMXRT685-EVK is 32 bits, and four channels.


Sound

Select a playback device below to modify its settings:

Speakers
3- USB AUDIO+COM
Default Device

Speakers
Realtek(R) Audio
Ready

Headphones
Realtek(R) Audio
Not plugged in

Speakers Properties

Select the enhancements to apply for your current speaker configuration. Changes may not take effect until the next time you start playback.

- [✓] Disable all enhancements
- [ ] Bass Boost
- [ ] Virtual Surround
- [ ] Room Correction
- [ ] Loudness Equalization

Enhancement Properties
Description:
Provider:
Status:

Figure 9. Disable the "USB AUDIO+COM" speaker device enhancements

Then the user can open a COM window, launch the recording APP, and start recording the up-streaming filtered audio. To select the filter, up-streaming and recording, type “a”/“A” and “0”~“7” in the COM window. For more details, see Section 7.2. To analyze and evaluate the filter and the filtered signal quality, watch the recorded signal, both in time and frequency domains.

7.4 Adjusting the filter coefficient

The MATLAB is used to generate the filter coefficients. The MATLAB generates the SOS coefficient table and a SOS scale array. To call the NatureDSP IIR filter APIs, it is not recommended to use the SOS coefficients and scales directly. To avoid undesirable overflows at the intermediate outputs, per-section scale factors can require some tuning to find a compromise between quantization noise and possible overflows. Adjusting the filter coefficients is the important process that this document illustrates.

If a simple 2nd order biquad filter is enough for an audio purpose, use the formulas in Table 1 and Table 2 to calculate the coefficients at HiFi4 DSP runtime. If the user notices an overflow, adjust the calculated coefficients.

7.4.1 For float type NatureDSP IIR APIs

The following code provides an example to initialize (with the original SOS coefficients) and call the float type IIR.

```
float CoefPtr[5] = { 1.0, -2.0, 1.0, -1.98148850, 0.9816582 };//coefficients
 are: b0,b1,b2,a1,a2
float *AudioSrcFltBuf;
 float *AudioDstFltBuf;
 int SizeInByte=bqriirf_df1_alloc(1);   //get the size that this filter will
 need
 int *p=malloc(SizeInByte);    //allocate a new space with this size for this
 filter
 bqriir32x32_df1_handle_t Filter1=bqriirf_df1_init(p, 1, CoefPtr, 0);
       //initialize this filter, telling where the coefficients are,
       //biquad section number and final gain left shift number
 bqriirf_df1 (Filter1,  AudioDstFltBuf, AudioSrcFltBuf, 128);
       //call the processing function to filter 1 frame of audio,
       //which is 128 samples
```

In the function bqriirf_df1_init, specify the following:

- Location of float coefficients
- Number of left shift bits needed for final amplifying

For float type IIR APIs, do not adjust the original SOS coefficients. For more details on the MATLAB code for adjusting the SOS values, see section 4.1.5 in the NatureDSP user guide. The signal quality is the same when comparing the filtering results using original SOS and the section 4.1.5 adjusted SOS.

7.4.2 For F32*32 type NatureDSP IIR APIs

The following code provides an example to initialize (with four biquad sections) and call the F32*32 type IIR.

```
int   CoefD1F32 [4*5]=
 {
   xxx,xxx,xxx,xxx,xxx, //section 1 coefficients: b0,b1,b2,a1,a2 --- needs to be
 adjusted and converted
   xxx,xxx,xxx,xxx,xxx, //section 2 coefficients: b0,b1,b2,a1,a2 --- needs to be
 adjusted and converted
   xxx,xxx,xxx,xxx,xxx, //section 3 coefficients: b0,b1,b2,a1,a2 --- needs to be
 adjusted and converted
   xxx,xxx,xxx,xxx,xxx, //section 4 coefficients: b0,b1,b2,a1,a2 --- needs to be
 adjusted and converted
 };
 short int SectScale[4]={xxx,xxx,xxx,xxx};
     //scale factors for each section
 short int FinalLeftShift=1;  //final gain, left shift value
```

```
int *AudioSrcIntBuf;
int *AudioDstIntBuf;
int SizeInByte=bqriir32x32_df1_alloc(4); //get the size that this filter will
need
int *p=malloc(SizeInByte);  //allocate a new space with this size for this
filter
bqriir32x32_df1_handle_t Filter2= bqriir32x32_df1_init(p, 4, CoefD1F32 ,
SectScale ,FinalLeftShift);
    //initialize this filter, telling where the coefficients are,
    //where the scale factors for each section are
    //biquad section number and final gain left shift number
bqriirf_df1 (Filter2,  AudioDstIntBuf, AudioSrcIntBuf, 128);
    //call the processing function to filter 1 frame of audio,
    //which is 128 samples
```

In the function bqriir32x32_df1_init, specify the following:

- Location of coefficients
- Location of scale factors for each section
- The final gain left shift number

The coefficients and the scale factors are in the fractional format. The MATLAB code converts them from SOS values in the NatureDSP user guide section 4.1.3. The header of this MATLAB code provides the information about the inputs and the outputs.

```
%4.1.3 bqriir32x32_df1 conversion
%-----------------------------------------------
% convert SOS+G to coefficients of IIR filter
% (bqriir32x32_df1 function)
% parameters:
% SOS,G - SOS matrix and gain vector G
% Fs - sample rate
% nfft - FFT length for analisys
% output:
% coef - vector with coefficients, Q30
% gain - biquad gains, Q30
% scale - final scale factor (amount of left shifts)
%-----------------------------------------------
function [coef,gain,scale]=cvtsos_bqriir32x32_df1(SOS,G,Fs,nfft)
```

However, if the SOS is from a certain filter configuration such as filter2 in this demo (an 8th order HPF that has a very low cut-off frequency of 100 Hz), the MATLAB code in section 4.1.3 of the NatureDSP user guide produces a scale factor of 0. This situation arises particularly with very low cut-off frequency. Another MATLAB code is created for adjusting the SOS. This MATLAB script assumes the “short int SectScale[]” to be 2^{15} that is 1.0 gaining and “FinalLeftShift” to be 0 that is 1.0 final gain. Instead, it adjusts the SOS scale array. Then the HiFi4 DSP code must convert the SOS coefficients together with the SOS scale values to fractional 1.30 values by itself. Details can be found in the source code of the HiFi4 DSP demo project.

7.4.3 Adjust the SOS

The problem of using the original SOS in fractional F32*32 type filters is overflow. As the IIR filters order number goes higher, each single section can amplify the signal at certain frequency significantly.

Figure 10 shows that the peaks reach to 20 dB~40 dB approximately. When the input signal is a full-level sweeping signal at a frequency of 700 Hz, the output from the first biquad section is 20 dB, exceeding the full-level range. It causes serious overflow issues. The user must scale the biquad coefficients to have its peak response at 0 dB.

Both the MATLAB codes from the NatureDSP user guide and from this document calculate the amplitude to frequency response of each biquad section according to the provided SOS coefficients. Find the maximum amplitude value kMax. The scale for this section is 1/kMax.

The following is the difference between the two kinds of MATLAB adjusting:

- The NatureDSP user guide MATLAB code puts the information of “1/kMax” to the scale factor of 16 bits fractional values, which must be referenced at IIR filter initializing (the “short int SectScale[]” values)
- The MATLAB code of this document writes back the information of “1/kMax” to an SOS scale array, which must be in 32-bits fractional values
- This document MATLAB code can convert or adjust the SOS successfully but the NatureDSP user guide can fail at certain filter coefficients. If 1/kMax is smaller than $1/2^{15}$:
  - The NatureDSP MATLAB code generates 0 in 16 bits
  - This document MATLAB code can still keep the useful value in 32 bits

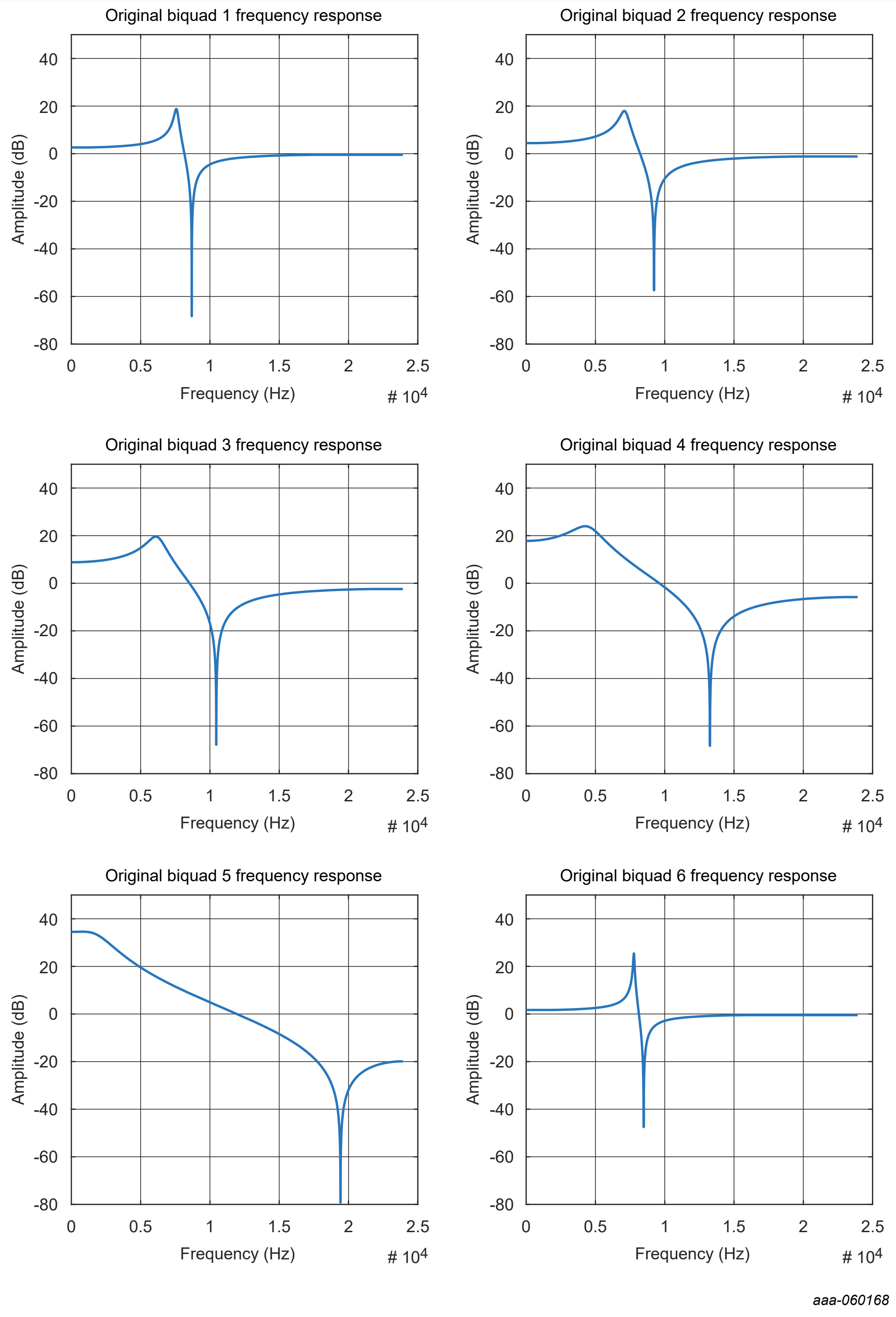


Figure 10. Amplitude to frequency response of each 1~Nth biquad of a 12 order IIR with scaling (filter3 in this demo project)

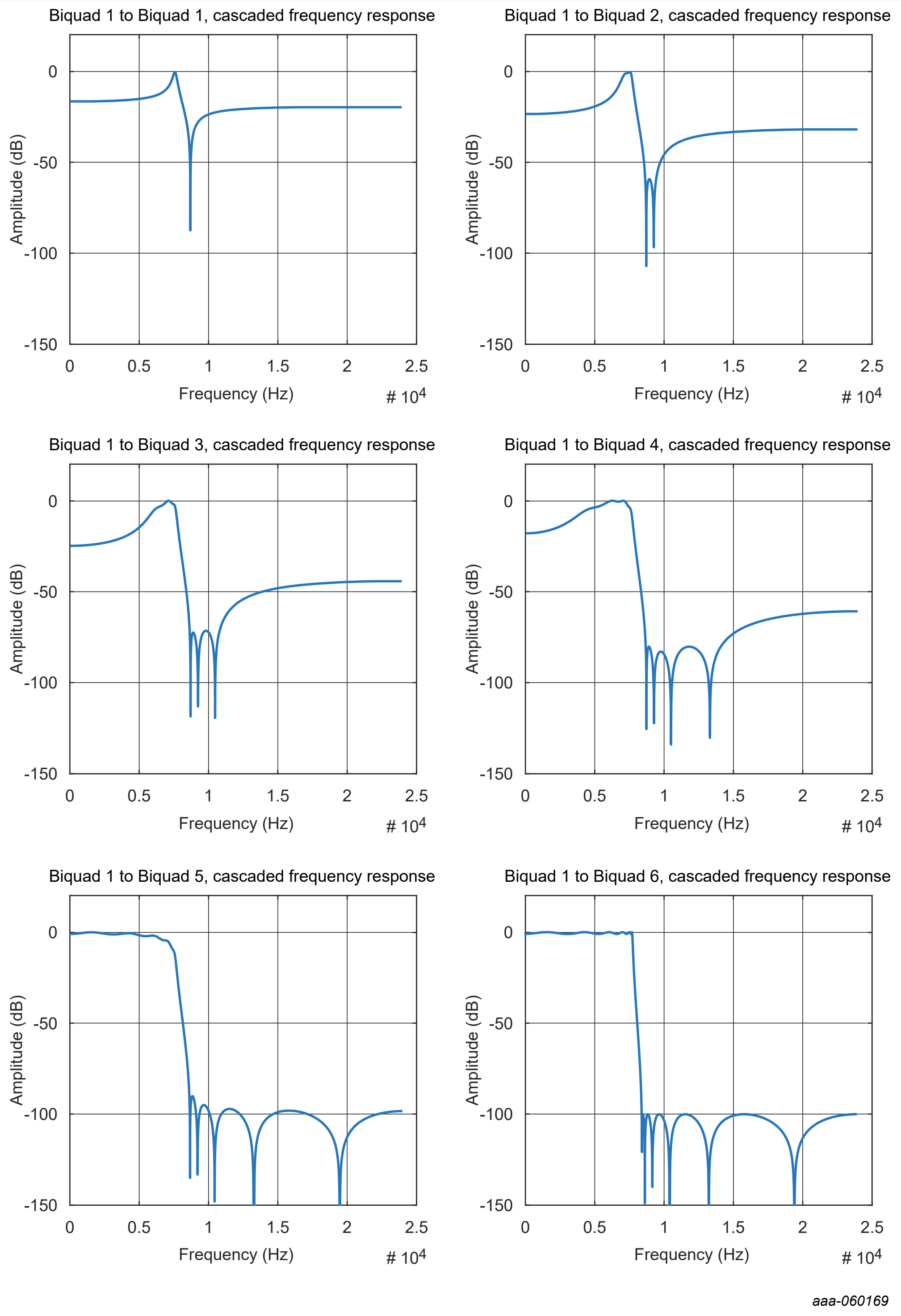


Figure 11. Amplitude to frequency response of each 1~Nth biquad of a 12 order IIR with scaling (filter3 in this demo project)

Figure 11 shows that the amplitude meets 0 dB for section1, section1 to section2, section1 to section3 … … section1 to section6. This means that there is no overflow at the output of each section output. In this demo, the “Biquad 1 to Biquad 6” section of Figure 11 is the true frequency response of the complete 12th order IIR filter3.

Note: *Even if the output from one biquad section does not overflow, it is possible that the intermediate state of this biquad section is overflowing. When the overflow happens, serious signal distortion is obvious to be seen. Try to decrease the "short int SectScale[]" value half down, and give one more left shift bit to the final gain.*

8 Filters analyzing signal quality

After building and running the existing demo, to analyze the filter performance and audio quality, start the USB audio recording. The user can select the HiFi4 internal sweeping tone generator as the signal source, and record eight audio wav files (select filter0~7 accordingly).

Figure 12 shows the recorded wav of filter3 with internal sweeping tone generator. This analysis is based on the waveform shape and spectrum curves in Adobe Audition.

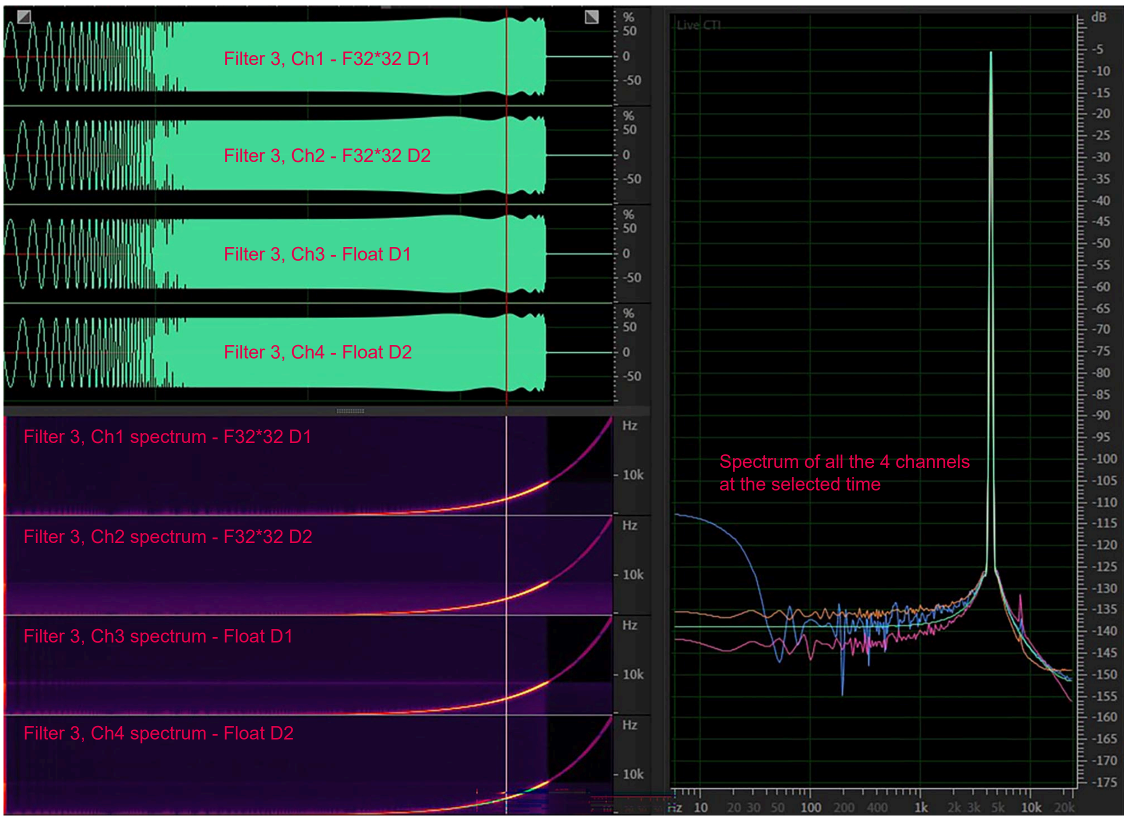

Filter 3, Ch1 - F32*32 D1
Filter 3, Ch2 - F32*32 D2
Filter 3, Ch3 - Float D1
Filter 3, Ch4 - Float D2
Filter 3, Ch1 spectrum - F32*32 D1
Filter 3, Ch2 spectrum - F32*32 D2
Filter 3, Ch3 spectrum - Float D1
Filter 3, Ch4 spectrum - Float D2
Spectrum of all the 4 channels at the selected time
%
50
0
-50
Hz
10k
dB
-5 -10 -15 -20 -25 -30 -35 -40 -45 -50 -55 -60 -65 -70 -75 -80 -85 -90 -95 -100 -105 -110 -115 -120 -125 -130 -135 -140 -145 -150 -155 -160 -165 -170 -175
Hz 10 20 30 50 100 200 400 1k 2k 3k 5k 10k 20k
aaa-060176

Figure 12. Recorded audio when filter3 is selected

8.1 Audio signal output from F32*32 type IIR filters

Among the four channels, the ch3 is the float D1 type output, and the ch4 is the float D2 type output. Observing all the eight groups of filters shows that the float D2 filter0 (20 Hz HPF), and float D2 filter5 (100 Hz HPF) have heavy noise.

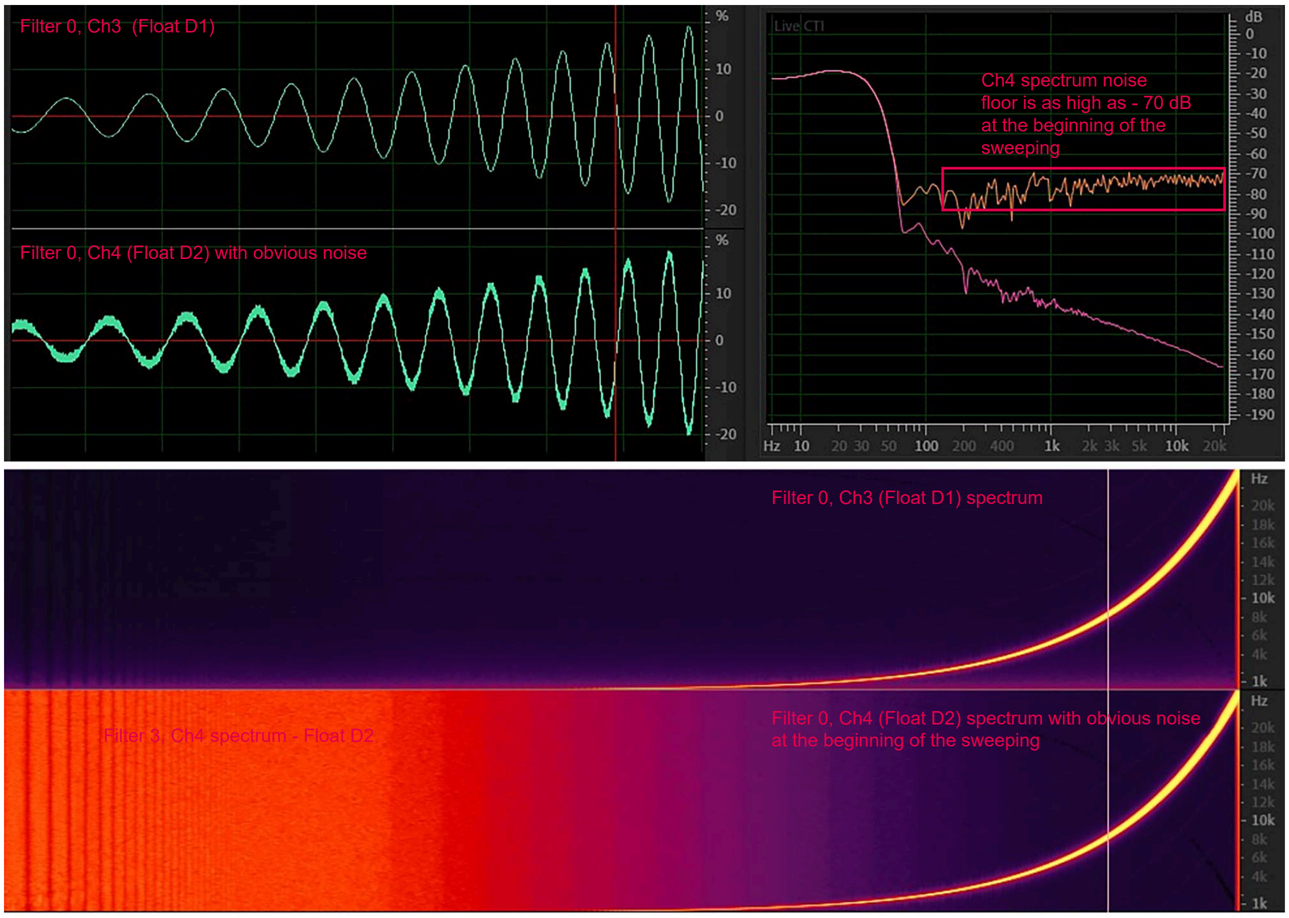


Figure 13. 20 Hz HPF - float D2 has a noise floor at -80 dB in the spectrum

All the float D2 filters except filter0 and filter5 have the same spectrum as float D1 filters and the noise floor in the spectrum is very-low.

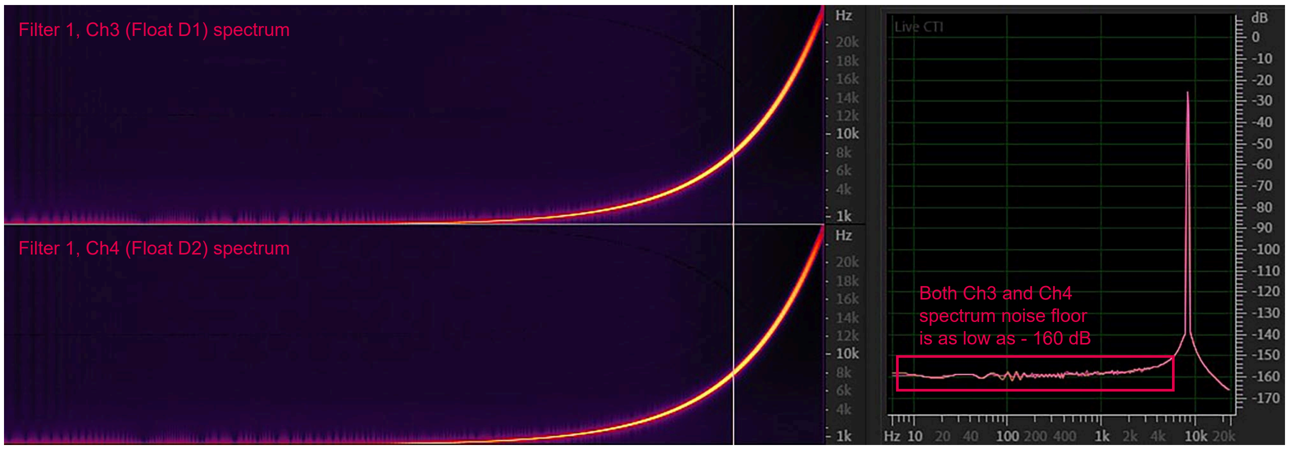


Figure 14. 3000 Hz LPF - float D1 and D2 both have noise floor at -160 dB in the spectrum

Note: *The float type D2t has the same spectrum feature as the float type D1 and there is no F32*32 type D2t form APIs.*

8.2 Audio signal output from float type IIR filters

Among the four channels, the ch1 is the F32*32 D1 type output, and the ch2 is the F32*32 D2 type output. Observe all the eight groups of filters. For more details, see Figure 15, Figure 16, Figure 17, Figure 18, and Figure 19.

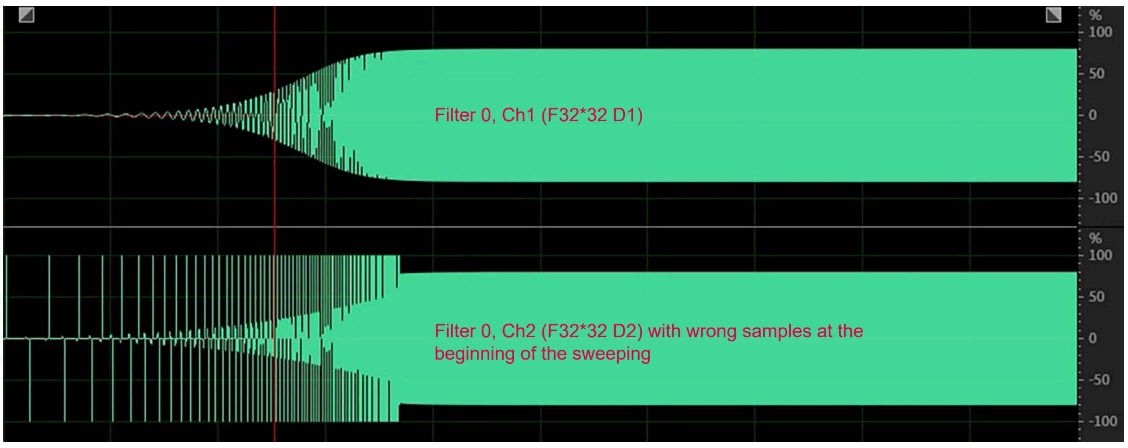


Figure 15. 20 Hz HPF - F32*32 D2 output is wrong when sweeping at very low frequency

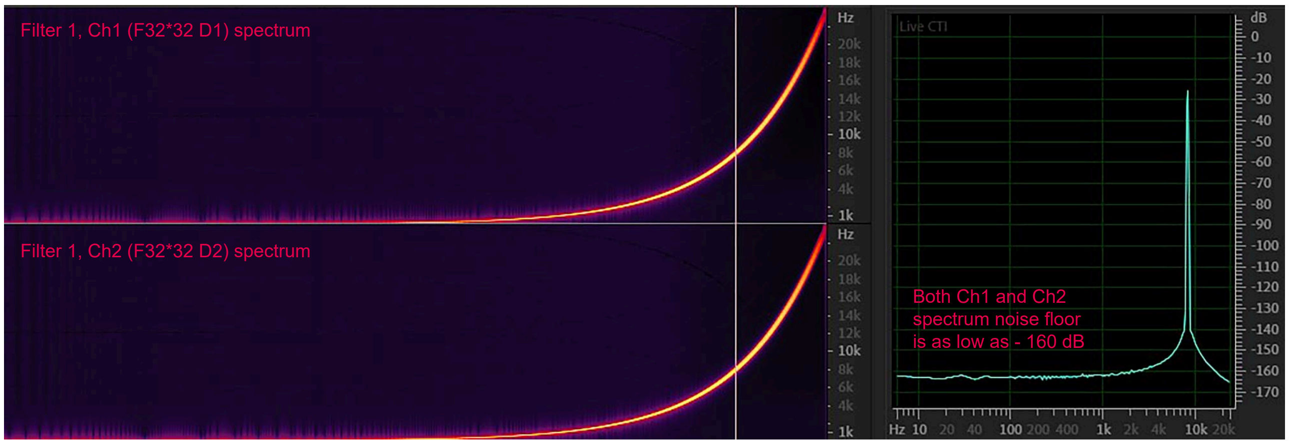


Figure 16. 3000 Hz LPF - F32*32 D1 and F32*32 D2 output are both good and noise floor in the spectrum is low

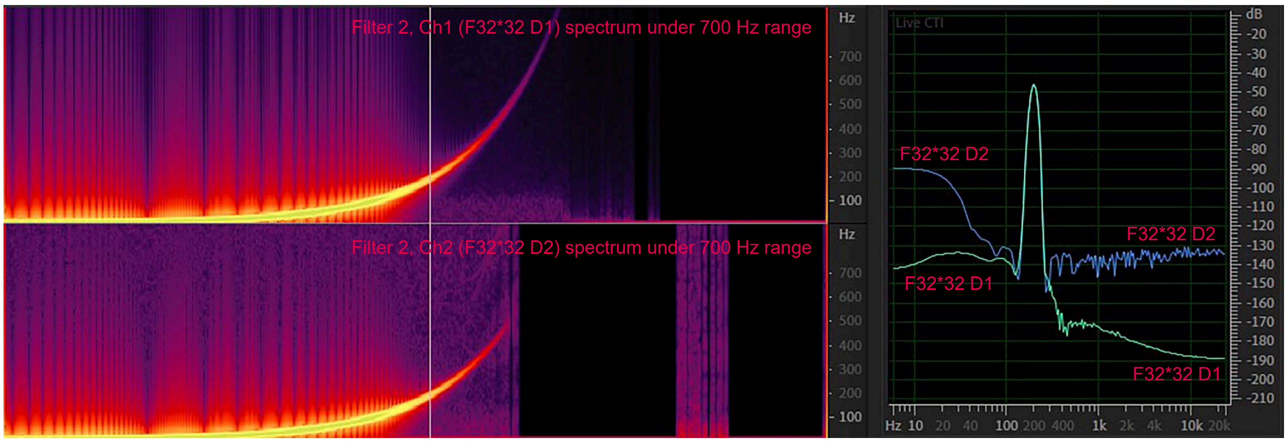


Figure 17. 100 Hz LPF - F32*32 D1 has a lower noise floor in the spectrum than F32*32 D2

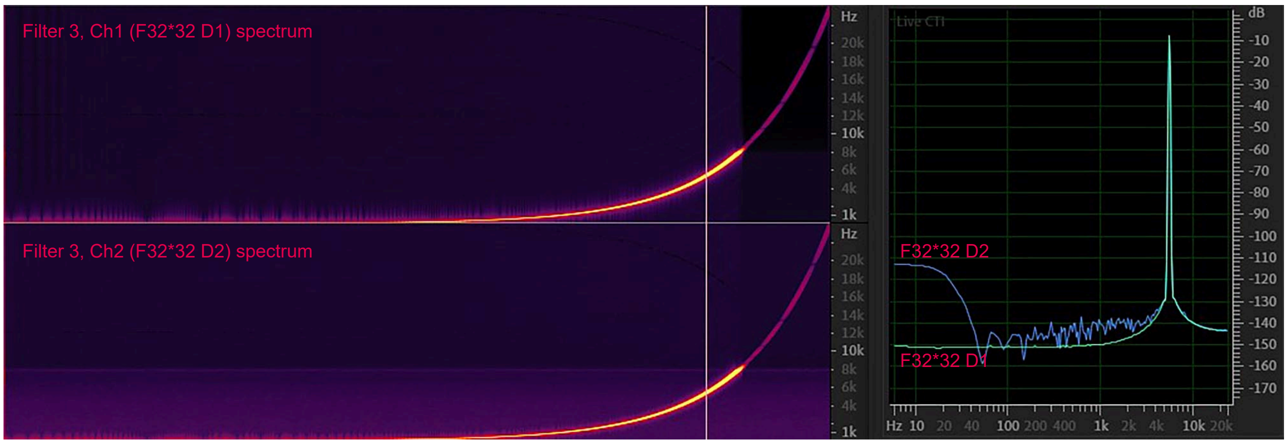


Figure 18. 7800 Hz LPF - F32*32 D2 has higher noise below 40 Hz

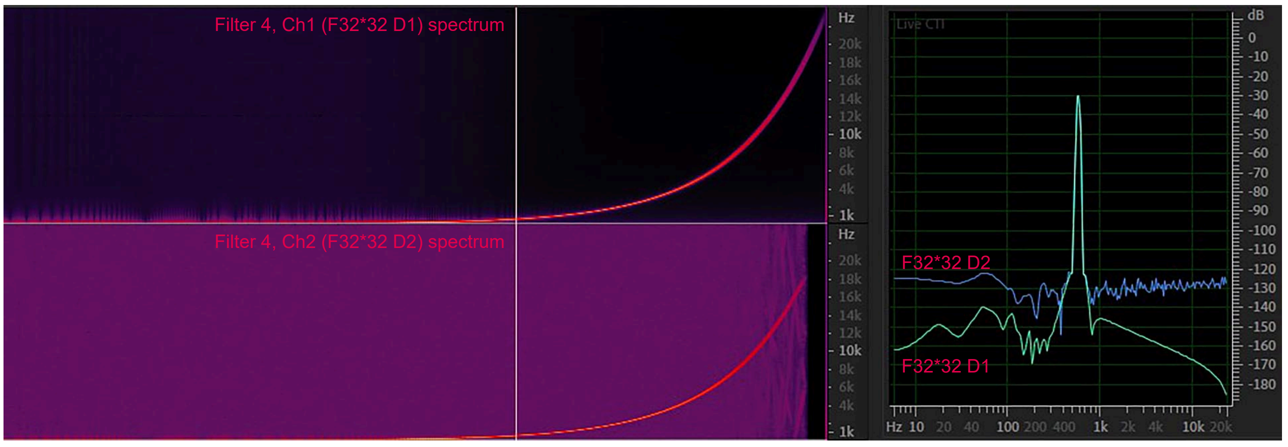


Figure 19. 111 Hz LPF - F32*32 D2 has higher noise

8.3 Comparison between float D1 type and F32*F32 D1 type filters

When the cut-off frequency of the HPF filter is too low, the float D1 type has more noise in low frequency.

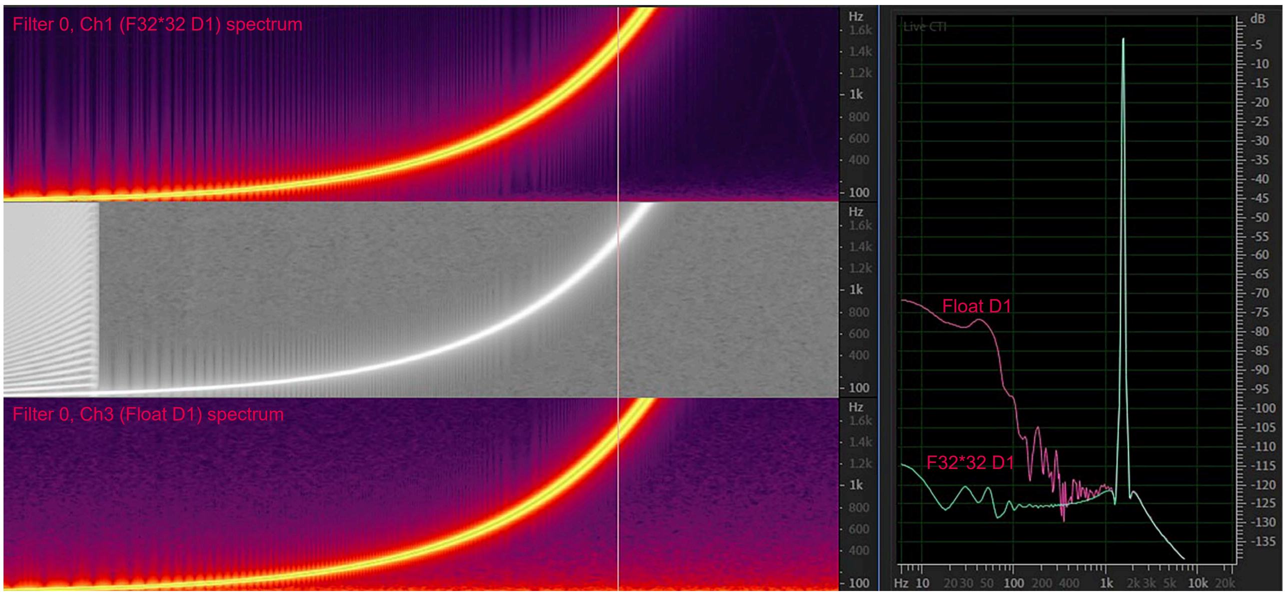


Figure 20. 20 Hz HPF - F32*32 D1 has lower noise in the spectrum

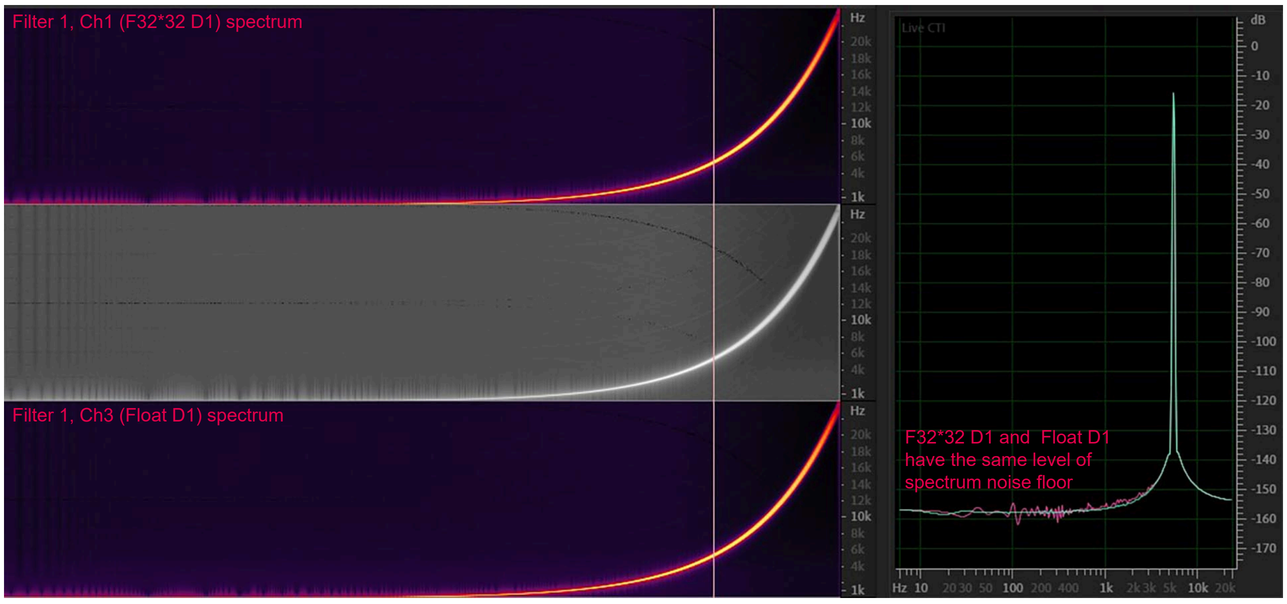


Figure 21. 3000 Hz LPF - F32*32 D1 and float D1 has the same level of noise

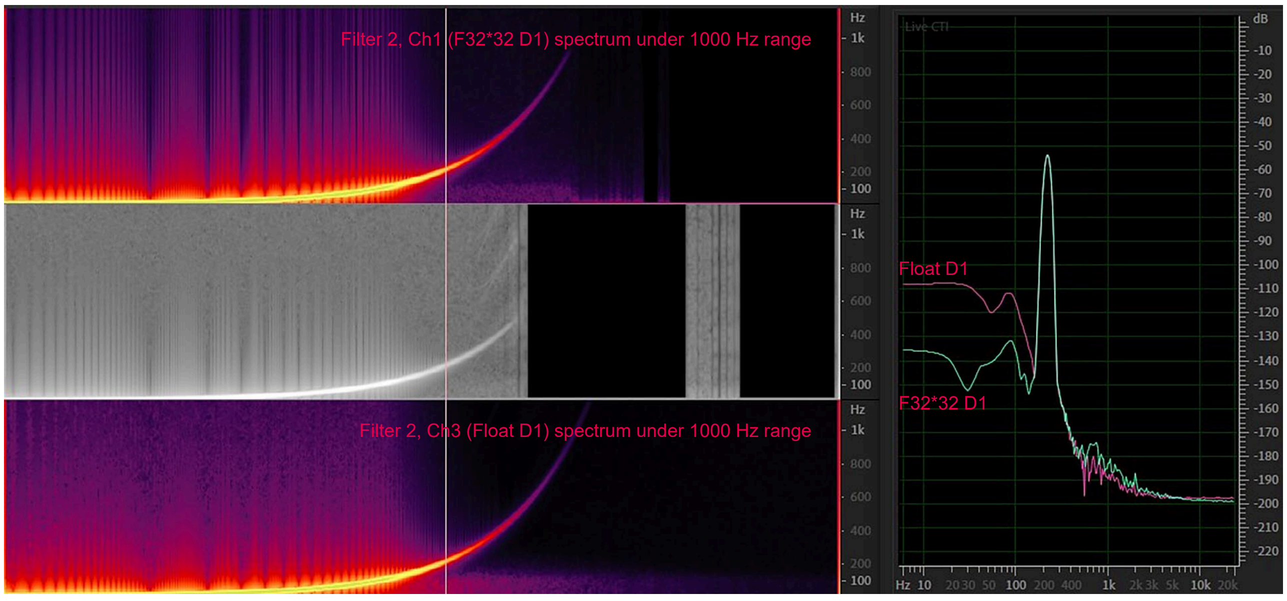


Figure 22. 100 Hz HPF - float D1 has a higher noise floor below 100 Hz

8.4 Comparison between F32*F32 type filters with different sector scale factor

When using F32*32 type IIR filter APIs, adding final gain shift bits as compensation to increase the attenuation of each intermediate biquad section results in higher noise addition to the signal.

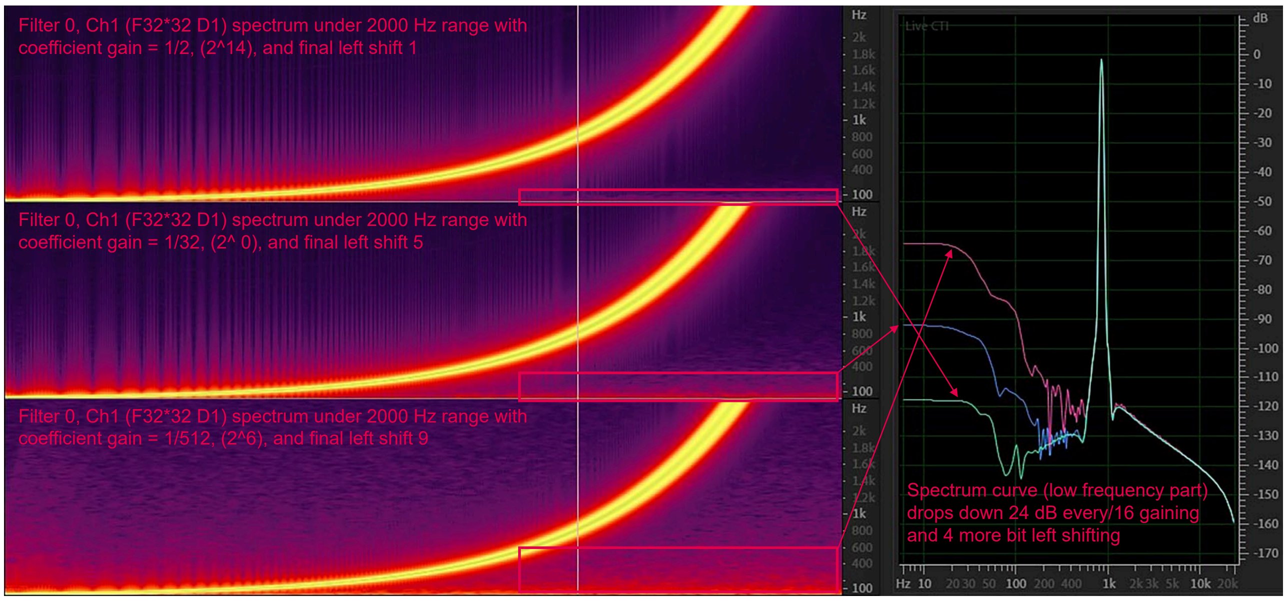


Figure 23. Comparison between F32*F32 type filters with different sector scale factor

9 Filters analyzing instruction cycles

When running the demo program, the numbers are printed once every 1 second in the COM window. These numbers are the cycle count numbers that are measured and calculated in the HiFi4 DSP code. Table 6, Table 7, and Table 8 summarize all the cycle counts that are measured with the demo project of this document.

Table 6. Cycle counts when frame size = 48 samples

	Float D1	Float D2	Float D2T	F32x32 D1	F32x32 D2
2 order	435	345	546	256	264
4 order	558	571	519	289	349
8 order	613	580	624	475	597
12 order	1052	1039	1022	660	845
Cycles per sect per sample	9.06	7.19	11.38	5.33	5.50
	5.81	5.95	5.41	3.01	3.64
	3.19	3.02	3.25	2.47	3.11
	3.65	3.61	3.55	2.29	2.93

Table 7. Cycle counts when frame size = 256 samples

	Float D1	Float D2	Float D2T	F32x32 D1	F32x32 D2
2 order	1812	1305	2442	984	992
4 order	2455	2363	2311	1017	1389
8 order	2718	2580	2832	1827	2573
12 order	4923	4703	4894	2634	3758
Cycles per sect per sample	7.08	5.10	9.54	3.84	3.88
	4.79	4.62	4.51	1.99	2.71
	2.65	2.52	2.77	1.78	2.51
	3.21	3.06	3.19	1.71	2.45

Table 8. Cycle counts when frame size = 512 samples

	Float D1	Float D2	Float D2T	F32x32 D1	F32x32 D2
2 order	3506	2489	4778	1878	1866
4 order	4789	4571	4519	1911	2666
8 order	5308	5045	5552	3489	5003
12 order	9691	9215	9662	5066	7342
Cycles per sect per sample	6.85	4.86	9.33	3.67	3.64
	4.68	4.46	4.41	1.87	2.60
	2.59	2.46	2.71	1.70	2.44
	3.15	3.00	3.15	1.65	2.39

See:

F32*32 type D2 cycle counts > F32*32 type D1 cycle counts, that is, F32*32 type D1 is quicker.

Float type D1 cycle counts > Float type D2 cycle counts, that is, Float type D2 is quicker.

Generally, F32*32 type is 30 %~150 % quicker than float type.

10 Conclusion

- After implementing the IIR filters with NatureDSP IIR filter APIs, the performance of the IIR filters must be checked. Use the sine chirp tone to sweep the IIR filter and inspect the noise in the spectrum of the output.
- The NatureDSP fractional type IIR filter APIs need SOS coefficients generated in MATLAB be tuned before the filter initializing. To tune, use the MATLAB code supplied in the NatureDSP user manual or the MATLAB code from this document.
- The NatureDSP IIR filter APIs support multiple biquad sections cascaded, so they can make a high order IIR filter. It is important to ensure that each biquad section output is not overflowing.
- After tuning the IIR filter coefficients, the gain values and the final shift values for the API to prevent any intermediate overflow. Then focus on the filter output signal quality in terms of spectrum noise floor.
- The DF2 is generating huge intermediate values, therefore it needs large-scale attenuation for each biquad sector, and accordingly adds the extra left shift bits to compensate. It brings the significant quantization noise.
- Use the advantage of HiFi4 floating point unit (FPU), and select the float D1 type of the NatureDSP IIR APIs. If extra noise at low frequency is noticed and cannot be accepted, switch to use the NatureDSP F32*32 D1 type filter. It is necessary to adjust the SOS coefficients.
- The advantage of NatureDSP F32*32 D1 type API is the speed. The F32*32 D1 is faster than the float D1.
- The F32*32 D2 and float D2 are not recommended. For the same reason, the CMSIS-DSP library does not have D2 IIR APIs.
- The Float D2t has the same spectrum feature as the float D1.

11 Acronyms

Table 9 lists the acronyms used in this document.

Table 9. Acronyms

Term	Description
AP	All-pass
API	Application programming interface
BP	Band-pass
BR	Band-reject
DF1	Direct form 1
DF2	Direct form 2
DF2t	Direct form 2 transposed
DSP	Digital signal processor
FIR	Finite impulse response
FPU	Floating point unit
HP	High-pass
HPF	High-pass filter
IIR	Infinite impulse response

Table 9. Acronyms*...continued*

Term	Description
LP	Low-pass
LPF	Low-pass filter
PC	Personal computer
SOS	Second order sections
USB	Universal serial bus

12 References

Table 10 lists the references used to supplement this document.

Table 10. Related documentation/resources

Document	Link/how to access
INTRODUCTION TO DIGITAL FILTERS	INTRODUCTION TO DIGITAL FILTERS
NatureDSP User Guide	Contact local FAE or sales representative

13 Note about the source code in the document

Sample code shown in this document has the following copyright and BSD-3-Clause license:

14 Revision history

Table 11 summarizes the revisions to this document.

Table 11. Revision history

Document ID	Release date	Description
AN14627 v.1.0	22 April 2025	Initial public release

Legal information

Contents

Date of release: 22 April 2025
Document identifier: AN14627