APPLICATION NOTE AN19

DC BLOCKING FILTERS

DC offset often exists in the microphone output. This can be removed with a DC blocking filter to provide clean audio for downstream signal processing. This document describes a relatively simple implementation that can be used to block DC offset in many applications. It also includes an example fixed point implementation.

DESIGN

The common design of a DC blocking filter is with one pole and one zero. Its transfer function, H(z), is defined as:

$$H(z) = \frac{1 - z^{-1}}{1 - a * z^{-1}}$$

The coefficient "a" determines the cut off frequency depending on the system sample rate.

Larger "a" coefficients produce a slower DC blocking response but with less attenuation at lower frequencies. Conversely, smaller "a" coefficients achieve a faster DC blocking response but with more low-frequency attenuation. Choosing the best coefficient value is always a trade-off in applications.

Some applications require both during the startup stage. That is, the filter must have a quick DC blocking response while having minimal attenuation at the lower frequencies during normal operation. In this scenario, the goal can be achieved by choosing a smaller DC blocking filter coefficient at startup and switching to a larger coefficient one once the filter is blocking all or most of the DC offset. An audible glitch could occur during the switch, so muting the audio output may be necessary during the transition of coefficients.

Typically, designers have flexibility in choosing coefficients. Implementation may also limit this flexibility. For example, in a floating point implementation, coefficients of $(1-2^{-11}) = 0.99951171875$ or $(1-2^{-12}) = 0.999755859375$ may work for a typical DC Blocking filter. However, filter parameters chosen for optimal precision in a fixed-point implementation typically utilize filter coefficents less than 0.999 (eg. a ≤ (1-2-9) = 0.998046875).

For faster startup time, choose a smaller coefficient tailored to the application.

For a coefficient of 0.99951171875, Table 1 shows the -3 dB cut off frequency (Low Frequency Roll Off or LFRO) for various sample rates.

Table 1: Filter characteristics with $a = 1-2^{-11} = 0.99951171875$

Sample Rate (Hz)	-3 dB Cutoff Frequency (Hz)	20 Hz Attenuation (db)
8000	.625	-0.0042
16000	1.250	-0.0169
24000	1.875	-0.0380
32000	2.500	-0.0673
48000	3.750	-0.1501

Similarly, Table 2 shows the cut off frequency for various sample rates for a coefficient of 0.998046875.

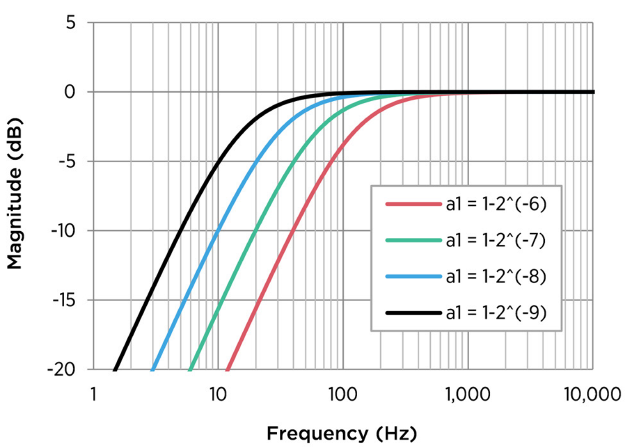
You can note the higher LFRO frequency and corresponding attenuation at 20 Hz.

Table 2: Filter characteristics with $a = 1-2^{-9} = 0.998046875$

Sample Rate (Hz)	-3 dB Cutoff Frequency (Hz)	20 Hz Attenuation (db)
8000	2.500	-0.0673
16000	5.000	-0.2633
24000	7.500	-0.5714
32000	10.000	-0.9691
48000	15.000	-1.9382

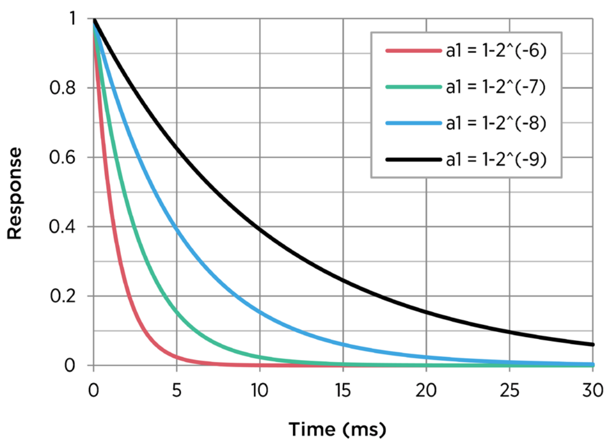
The filter's frequency response for a sampling rate (Fs) of 48 kHz for various filter coefficients is shown in Figure 1.

Figure 1: Filter Frequency Response, Fs=48kHz



Also important to filter performance is DC blocking time. As described earlier, smaller coefficients will provide faster blocking times, but have more low-frequency attenuation. Figure 2 below shows blocking time for various coefficients.

Figure 2: Filter DC Blocking Time, Fs=48kHz



IMPLEMENTATION

Reference MATLAB and fixed-point C code are shown here for demonstration purposes

MATLAB code for the filter example is:

```
function y = DCBlock ( x, c )% c - coefficient
     b = [1 - 1];           % filter coefficient b
     a = [1 - c];           % filter coefficient a
     y = filter (b,a,x);
end
```

A fixed-point C reference code example is:

```
/*
 * File      : DC_Blocking_filter.c
 *
 * Copyright (C) 2017 Knowles Electronics,
 * Itasca, IL USA, All Rights Reserved
 *
 */

#include <stdio.h>

#define A1 32511 // (1-2^(-7))     Q32:1.31

#define TO_16BIT_SHIFT 15
#define MAX_Uint32_PCMBIT_SIZE 4294967296
#define MAX_UNSIGN_PCMBIT_SIZE 65536
#define MAX_SIGN_POS_PCMBIT_SIZE 32768
#define MAX_SIGN_NEG_PCMBIT_SIZE -32768

static Int16 x_prev=0;
static Int32 y_prev=0;

void dc_filter(Uint16 *pcmIn)
{
   Int16 sampleIn, delta_x, sampleOut;
   Int32 a1_y_prev;

   sampleIn = (Int16)*pcmIn;
   delta_x = sampleIn-x_prev;
   a1_y_prev = A1*y_prev/MAX_SIGN_POS_ PCMBIT_SIZE;
   sampleOut = delta_x+(Int16)a1_y_prev;

   x_prev = sampleIn;
   y_prev = (Int32)sampleOut;

   *pcmIn = (Uint16)sampleOut;
}
```


Knowles Corporation
1151 Maplewood Drive
Itasca, Illinois 60143

Phone: 1 (630) 250-5100
Fax: 1 (630) 250-0575
sales@knowles.com

Model/Reference Number:
APPLICATION NOTE AN19