



**Hideo Okawara's
Mixed Signal Lecture Series**

**DSP-Based Testing – Fundamentals 51
CIC Decimation and FIR Filter**

*ADVANTEST Corporation
August 2013*

Preface to the Series

ADC and DAC are the most typical mixed signal devices. In mixed signal testing, an analog stimulus signal is generated by an arbitrary waveform generator (AWG) which employs a D/A converter inside, and an analog signal is measured by a digitizer or a sampler which employs an A/D converter inside. The stimulus signal is created using a mathematical method, and the measured signal is processed with mathematical method, extracting various parameters. It is based on digital signal processing (DSP) so that our test methodologies are often called DSP-based testing.

Test/application engineers in the mixed signal field should have a thorough knowledge about DSP-based testing. FFT (Fast Fourier Transform) is the most powerful tool here. This corner will deliver a series of fundamental knowledge of DSP-based testing, especially FFT and its related topics. It will help test/application engineers comprehend what the DSP-based testing is and its assorted techniques.

Editor's Note

For other articles on this series, please visit the Advantest web site at <http://www1.verigy.com/ate/news/newsletter/index.htm>

Preface

A delta-sigma type ADC core requires an appropriate digital low pass filter to convert the 1-bit data stream into the multi-bit word data. The sampling rate of data is decimated during this conversion process. The most typical filter is a CIC decimation filter plus an FIR post-filter with the SINC compensation. In some cases, such delta-sigma ADC output can be directly monitored as a 1-bit stream which should be evaluated with a specific post processing. This article describes how such filters can be programmed in our test methods.

Decimation Filter

Figure 1 illustrates a simple schematic of a delta-sigma ADC, whose 1-bit data stream is converted into a specific multi-bit word by the decimation filter. Decimation here is actually a very simple operation, which is just an averaging. By accumulating the bit stream for a certain period of time and dividing the sum by the number of periods, the data resolution is increased from 1-bit to multi-bit in exchange for the sampling rate. The sampling rate is slowed down in this stage. That is why this is called a decimation filter.

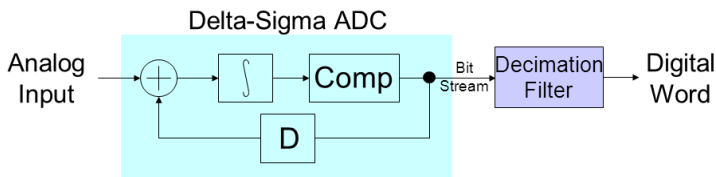


Figure 1: Delta-sigma ADC

Averaging is equivalent to a low pass filter (LPF). Once the signal is band-limited by the LPF, you can reduce the sampling rate without aliasing. This is the decimation filter depicted in Figure 2. The sampling rate “ f_s ” is reduced to “ f_s/R .” The R is the decimation factor.

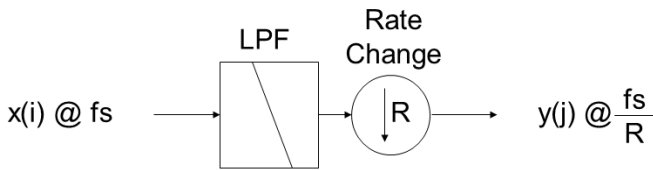


Figure 2: Decimation Filter

Digital LPF for this purpose is usually realized as the CIC (cascaded integrator comb) filter.¹ Figure 3 illustrates schematics of a comb filter, an integrator and a CIC.

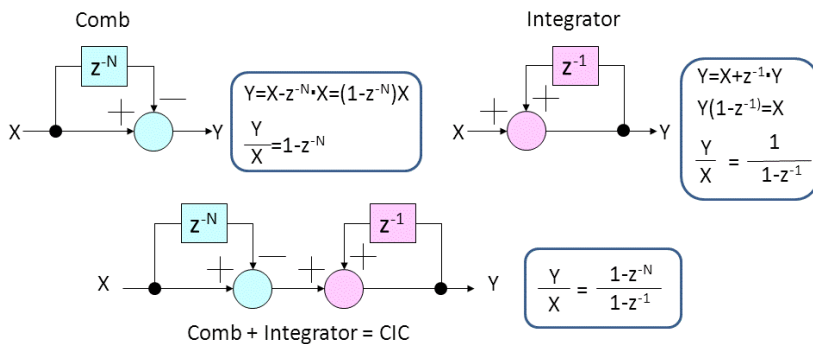


Figure 3: Comb + Integrator

¹ Mixed Signal Lecture Series DSP-Based Testing – Fundamentals 16 “Moving Average”

The frequency responses of the comb, integrator and CIC are described in Figure 4, where the number of comb delay stages N is 4. The gain of this CIC becomes 4 ($=N$) at DC so that the output should be scaled to $1/N$ at the end. The first null point resides at f_s/N so that the number of delay N establishes the passband width. The total response of the LPF actually looks something like a SINC ($\sin X/X$) curve. It is not flat, so a wide band signal waveform would be distorted to some extent.

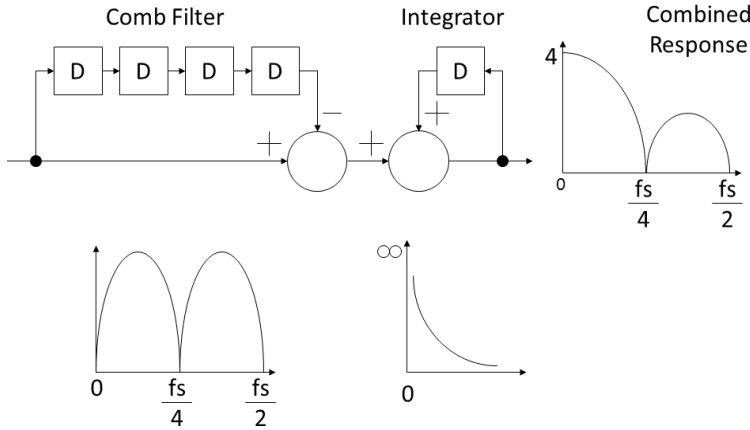


Figure 4: Frequency Responses of CIC

In order to improve the low pass characteristics, multiple stages of the single CIC can be cascaded as Figure 5 illustrates. Each of the CIC components can be relocated with each other.

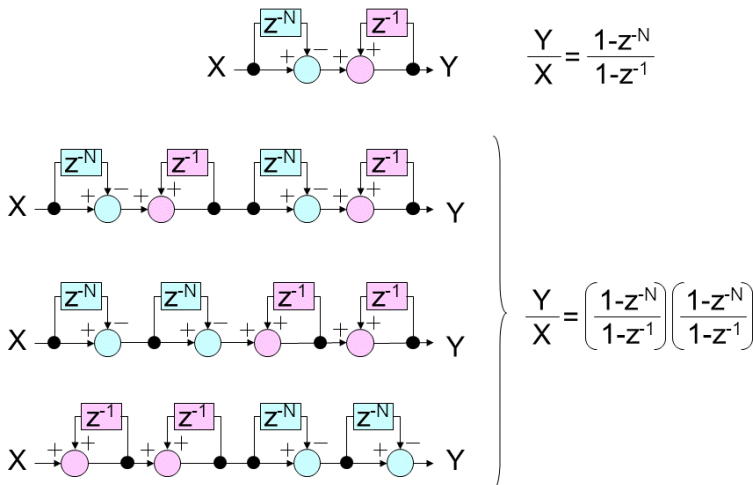


Figure 5: Multi-stage CIC Filter

In Figure 6 the comb and the integrator sections are simplified as "C(N)" and "I". As the signal is band-limited by the LPF, you can change the sampling rate slower. R represents the rate change or decimation factor.

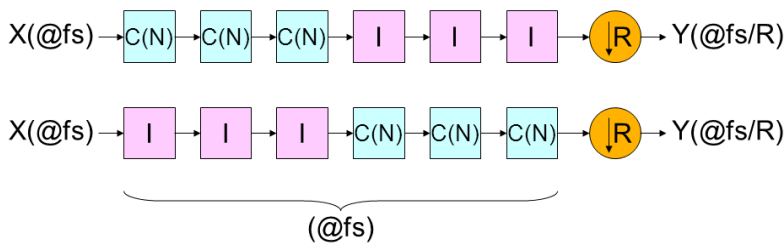


Figure 6: Decimation by 1/R

The integrator has a primitive low pass characteristics, so decimation section can be moved forward between the integrators and the combs as Figure 7 illustrates. The integrator stages run at the rate of f_s . Then the sampling rate is changed by the factor of R . Consequently the comb stages can run at the rate of f_s/R and the delay factor N should be substituted to N/R for keeping the same low pass characteristics. So N should be a multiple of R for making N/R an integer. Since the number of delay elements reduces from N to N/R , circuit complexity or calculation routine's complexity is simplified. Figure 7 depicts the original frequency response of the CIC ($N=8$) as black and the modified frequency response as coloured when $R=2$.

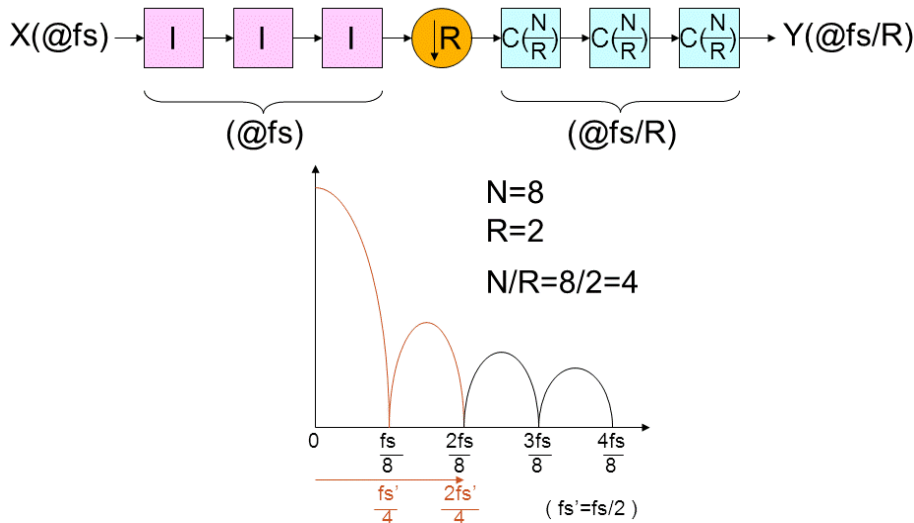


Figure 7: Frequency Response

The frequency response of CIC filter is actually not so sharp as expected. Therefore, in order to improve the rejection, an FIR digital filter is usually attached right after the CIC section as Figure 8 illustrates. Furthermore, the FIR section is utilized to compensate for the passband flatness as Figure 9 illustrates. This combination of CIC+FIR is the most typical decimation of LPF.

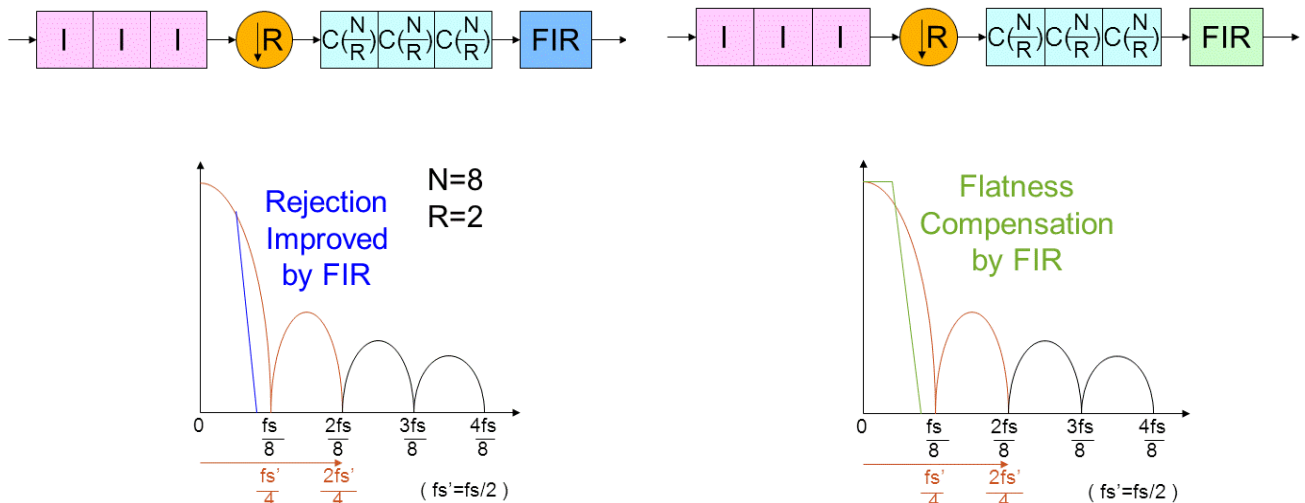


Figure 8: FIR for More Rejection

Figure 9: Loss Compensation

Experimental Result

Let us look at the actual performance of a decimation filter. This can be observed by simulation. Figure 10 shows the input data stream which is constructed with 1/0 data. This is a 2nd order delta-sigma modulated bit stream which actually contains 100 tones.² The FFT spectrum of the stream is shown in Figure 11. The shaped noise floor is clearly disclosed. The close-up view shows the 100 tones in the lowest band. For simplicity, the sampling rate of the original bit stream is supposed to be 65.536 kHz. The multi-tone is allocated from 1 Hz to 100 Hz.

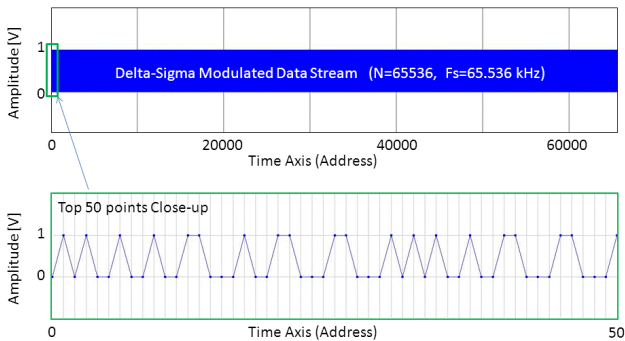


Figure 10: Test Signal Waveform

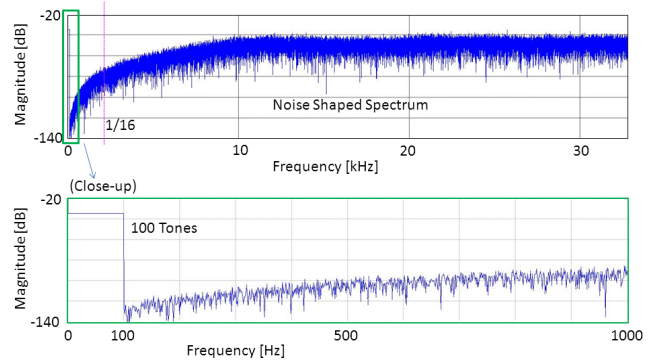


Figure 11: FFT Spectrum

The decimation filter we will demonstrate here has 3 stages of CIC section. The decimation factor is 16, so the sampling rate becomes $65.536 \text{ kHz}/16 = 4.096 \text{ kHz}$. The number of comb delays is 4 each. As discussed in Figure 9, the FIR section is designed to enhance not only the rejection characteristics but also the pass band flatness.

Figure 12 shows the primitive LPF profile of the FIR (length=64). The pass band response is inversely proportional to the SINC curve which is expected by the 3-stage CIC section. The inverse FFT (IFFT) converts the profile into the primitive FIR which is weighted by the Hanning curve and normalized.³ The completed FIR looks as shown in Figure 13.

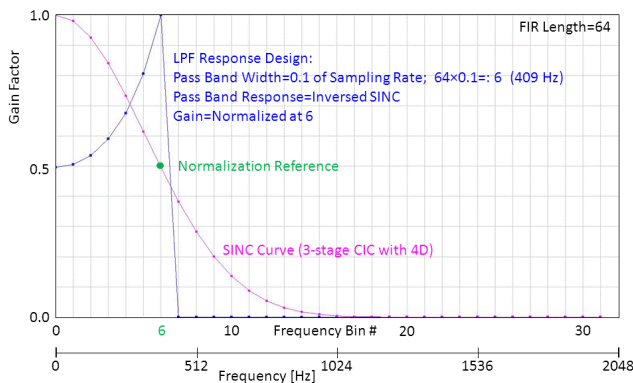


Figure 12: FIR Filter Response Design

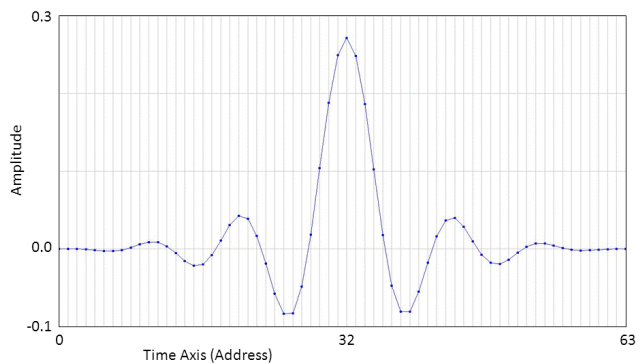


Figure 13: FIR

The frequency response of the CIC section and after the FIR are depicted in Figure 14. The frequency range is reduced to 1/16 against the original range in Figure 11. There are 3 CIC sections, however, the frequency response of the CIC is so dull that you can hardly tell the band width of the filter. The flatness of the pass band is also very poor. However, the CIC+FIR response looks excellent as a LPF and the pass band flatness is greatly improved.

² DSP-Based Testing Fundamentals 45 – Analog Signal Generator by Digital Pin Driver II

³ DSP-Based Testing Fundamentals 14 – FIR Filter

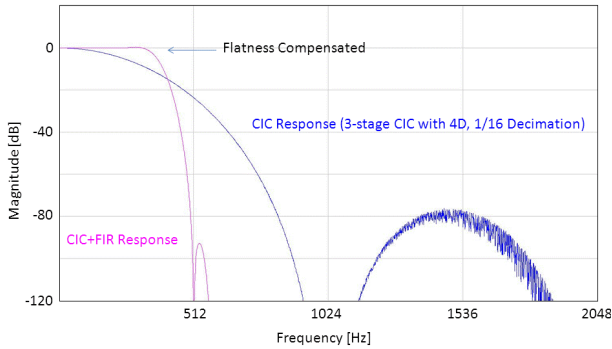


Figure 14: Frequency Response

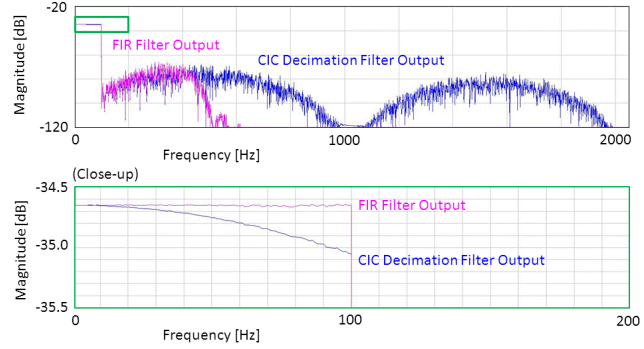


Figure 15: Output Signal Spectrum

The signal shown in Figures 10 and 11 is processed by this CIC+FIR filter, which reveals the spectrum and the waveform as Figures 15 and 16 respectively. The frequency range is reduced by 1/16. The spectrum shows how the flatness compensation appropriately works to the multi-tone from 1 Hz to 100 Hz. The waveform recovered in Figure 16 looks like a very typical shape of a well controlled multi-tone which is constructed with a hyperbolic phase offset.⁴ It looks something like a frequency swept signal. This type of multi-tone has a very low peak-to-rms ratio so if the full-scale of the device is fixed, each one of the tones can be distributed while conveying as much power as possible.

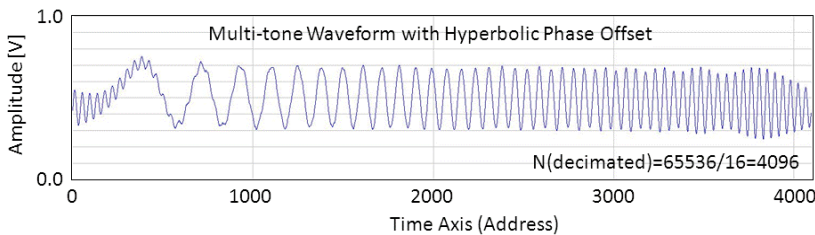


Figure 16: Output Signal Waveform

Example Source Code

This demonstration is performed by using the program listed below. The input and output signal waveforms are stored in the array "dWave0[]" and "dWave2[]". The CIC output is stored in "dWave1[]". Lines 38 to 59 in the main program works as the CIC with decimation. The decimation factor "Ndcm" is set 16 at Line 17. The comb delay "Nbufs" is set as 4 at Line 21. These parameters can be modified by changing the numbers themselves. There are 3 CIC sections, whose number is defined as "Nstages=3" at Line 20. This parameter relates to the coding of Lines 29 to 59. If you would like to change the number of stages, you should carefully modify coding from 29 to 59 as well as "Nstages".

⁴ DSP-Based Testing Fundamentals 4 – Multi-tone Data Generation

```

10: virtual void run()
11: {
12:     ARRAY_D dWave0,dWave1,dWave2;
13:
14:     // dWave0[] :                // Input Waveform Data (Rate: Fs)
15:
16:     int     Ndata=dWave0.size(); // Input Waveform Size
17:     int     NdcM=16;             // Decimation Factor    (1/16)
18:     int     Npts=Ndata/NdcM;     // Decimated Data Size (Fs/NdcM)
19:
20:     int     Nstages=3;           // # of CIC stages
21:     int     Nbufs=4;             // # of Delay in a COMB
22:     int     Nfir=64;             // FIR length
23:
24:     double  dBW=0.1;             // FIR Bandwidth (against Fs/NdcM)
25:
26:     int     i,j,k;
27:     int     m1,m2,m3;
28:
29:     double  dVa1,dVb1,dVc1,dVad1,dVbd1,dVcd1;
30:     double  dVa2,dVb2,dVc2,dVad2,dVbd2,dVcd2;
31:     double  dVa3,dVb3,dVc3,dVad3,dVbd3,dVcd3;
32:
33:     ARRAY_D dBuf1,dBuf2,dBuf3;
34:     ARRAY_D dFIR;
35:
36:     dWave1.resize(Npts);         // CIC Output Container
37:
38:     dBuf1.resize(Nbufs); dBuf1.init(0.0); // Initialization
39:     dBuf2.resize(Nbufs); dBuf2.init(0.0);
40:     dBuf3.resize(Nbufs); dBuf3.init(0.0);
41:
42:     dVa1=dVb1=dVc1=0.0; dVad1=dVbd1=dVcd1=0.0; // Initialization
43:     dVa2=dVb2=dVc2=0.0; dVad2=dVbd2=dVcd2=0.0;
44:     dVa3=dVb3=dVc3=0.0; dVad3=dVbd3=dVcd3=0.0;
45:
46:     j=0; k=0; m1=m2=m3=0;           // 3-stage CIC
47:     for (i=0;i<(2*NdcM*Ndata);i++) { // Many loops for filling buffers
48:         dVa1=dWave0[j]; dVb1=dVa1+dVad1; dVad1=dVb1; // Integrator #1
49:         dVa2=dVb1; dVb2=dVa2+dVad2; dVad2=dVb2; // Integrator #2
50:         dVa3=dVb2; dVb3=dVa3+dVad3; dVad3=dVb3; // Integrator #3
51:         if (!(i%NdcM)) {           // 3-stage Comb
52:             dVc1=dVb3-dBuf1[m1]; dBuf1[m1]=dVb3; m1++; if (m1==Nbufs) m1=0;
53:             dVc2=dVc1-dBuf2[m2]; dBuf2[m2]=dVc1; m2++; if (m2==Nbufs) m2=0;
54:             dVc3=dVc2-dBuf3[m3]; dBuf3[m3]=dVc2; m3++; if (m3==Nbufs) m3=0;
55:             dWave1[k]=dVc3/Nbufs/Nbufs/Nbufs/NdcM/NdcM/NdcM; // Scaling
56:             k++; if (k==Npts) k=0;
57:         }
58:         j++; if (j==Ndata) j=0;
59:     } // End of CIC Decimation Section
60:
61:     dsp_LPF_FIR_SINC(Nbufs,Nstages,Nfir,dBW,dFIR); // FIR Generation
62:     DSP_CONVOL(dFIR,dWave1,dWave2,ON); // Filtering
63:
64:     // dWave2[] // Filtered Output Waveform
65:
66:     return;
67: }

```

List 1: Sample Code (Main Routine)

```

10: void dsp_LPF_FIR_SINC(
11:     INT      Nbufs,      // Comb Delay Size (N/R)
12:     INT      Nstages,   // # of CIC stages
13:     INT      Nfir,      // FIR Length (2^n)
14:     DOUBLE   dBW,       // LPF Bandwidth (for Decimated Sampling Rate)
15:     ARRAY_D & dFIR      // FIR Container
16: )
17: {
18:     INT      i, j, Nsp, Ncut, Nmult;
19:     DOUBLE   dX, dY, dP, dMax;
20:     ARRAY_D   dWave1, dWave2, dSINC, dSp1, dSp2;
21:     ARRAY_COMPLEX CSp, CWave;
22:
23:     Ncut=(INT)(Nfir*dBW+0.5); // Index of Cut-off Frequency
24:     Nsp=Nfir/2; // Half of FIR size
25:     Nmult=Nbufs*Nstages; // SINC Multiplier
26:
27:     dSINC.resize(Nfir); // SINC (sinX/X) Curve Container
28:     dSINC[0]=1.0;
29:     dP=(M_PI*Nbufs/2)/Nfir;
30:     for (i=1; i<Nfir; i++) {
31:         dX=dP*i;
32:         dY=fabs(sin(dX)/dX);
33:         dX=1.0;
34:         for (j=0; j<Nmult; j++) dX=dX*dY;
35:         dSINC[i]=dX;
36:     }
37:     CSp.resize(Nfir);
38:     dMax=1.0/dSINC[Ncut]; // SINC Gain at Cut-off Point
39:     for (i=0; i<=Ncut; i++) // Pass Band Design
40:         CSp[i]=COMPLEX((1.0/dSINC[i]/dMax), 0.0);
41:     for (i=Ncut+1; i<=Nsp; i++) // Rejection Band Design
42:         CSp[i]=COMPLEX(0.0, 0.0);
43:     for (i=1; i<Nsp; i++) // Complex Conjugate
44:         CSp[Nfir-i]=COMPLEX(CSp[i].real(), -CSp[i].imag());
45:
46:     DSP_IFFT(CSp, CWave); // Freq Domain --> Time Domain
47:
48:     dFIR=CWave.getReal(); // Basic FIR
49:     for (i=0; i<Nsp; i++) { // Swap left and right planes
50:         dX=dFIR[i];
51:         dFIR[i]=dFIR[Nsp+i];
52:         dFIR[Nsp+i]=dX;
53:     }
54:
55:     for (i=0; i<Nfir; i++) { // Hanning weighting
56:         dX=0.5*(1.0-cos(2.0*M_PI*i/(Nfir-1)));
57:         dFIR[i]=dFIR[i]*dX;
58:     }
59:
60:     dX=0.0; for (i=0; i<Nfir; i++) dX=dX+dFIR[i]; // Integral
61:     DSP_MUL_SCL(1.0/dX, dFIR, dFIR); // Normalization of FIR
62: } // end of "dsp_LPF_FIR_SINC"

```

List 2: Sample Code (Subroutine for LPF)