Lab 13: LabVIEW DSP Module Examples

This lab includes three examples showing how the LabVIEW DSP Module can be used to run DSP graphical codes directly on a DSP target board without performing any C programming. These examples correspond to the waveform generation, digital filtering, and adaptive filtering labs covered in the previous chapters.

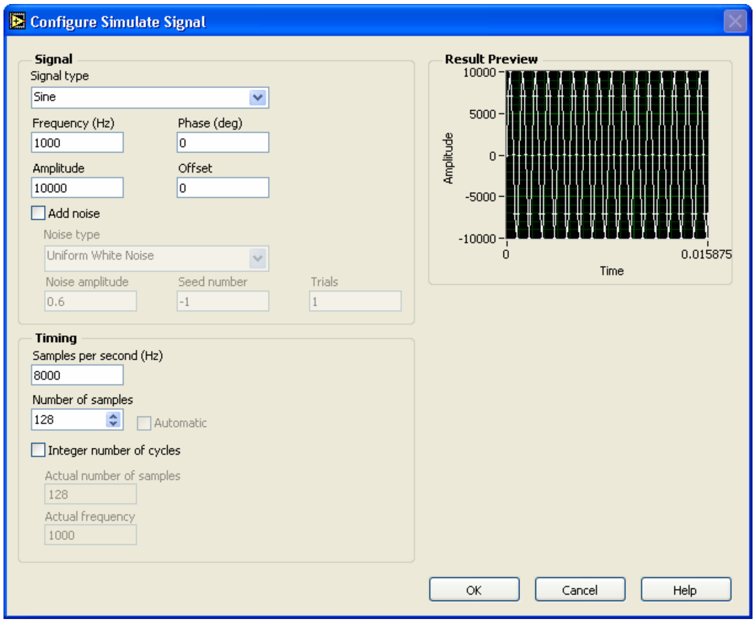
As stated earlier, to begin designing DSP systems by using the LabVIEW DSP Module, double-click on the LabVIEW Embedded Edition icon on the Windows desktop. After the appearance of the main dialog, select a proper target from the **Execution Target** pull-down menu. Throughout this lab, the C6713 DSK is used with the consideration that all the codes for DSK 6713 can also be run on the C6711 DSK and the SPEEDY-33 board without the need to make any modifications.

L13.1 Waveform Generation and Frequency Analysis

The first example consists of waveform generation and frequency analysis. Create a blank VI and locate a `Simulate Signal` VI (**Functions → Embedded Signal Generation → Simulate Signal**) in the BD to generate a waveform. Then, double-click on the VI to configure its parameters. In the configuration dialog, change the parameters as indicated in Figure 13-1. To observe the frequency characteristics of the generated signal, place a `Spectral Measurements` Express VI (**Functions → Signal Processing → Frequency Domain → Spectral Measurements**) in the BD and configure it as illustrated in Figure 13-2. Next, locate an `Analog Output` function (**Functions →**

Elemental I/O → Analog Output) and wire it to the output of the Simulated Signal VI. This function writes data from the DSP board in the form of an analog signal through the codec on the board. Double-click on the Analog Output function to change its configuration as shown in Figure 13-3.

Now, toggle to the FP and place two Waveform Graphs (**Controls → Graph → Waveform Graph**) and a Horizontal Pointer Slide (**Controls → Numeric → Horizontal Pointer Slide**). Each of these waveform graphs is wired to the simulated signal output and the spectral measurement output. The slide control is wired to the Frequency node of the Simulate Signal VI in order to change the frequency of the signal. As the final step, locate a While Loop and enclose all the BD objects to run the graphical code continuously. The completed BD of the VI is illustrated in Figure 13-4. Notice that the DSP board label is displayed at the bottom left corner of the BD and FP.



Configure Simulate Signal

Signal

Signal type: Sine

Frequency (Hz): 1000

Phase (deg): 0

Amplitude: 10000

Offset: 0

□ Add noise

Noise type: Uniform White Noise

Noise amplitude: 0.6

Seed number: -1

Trials: 1

Timing

Samples per second (Hz): 8000

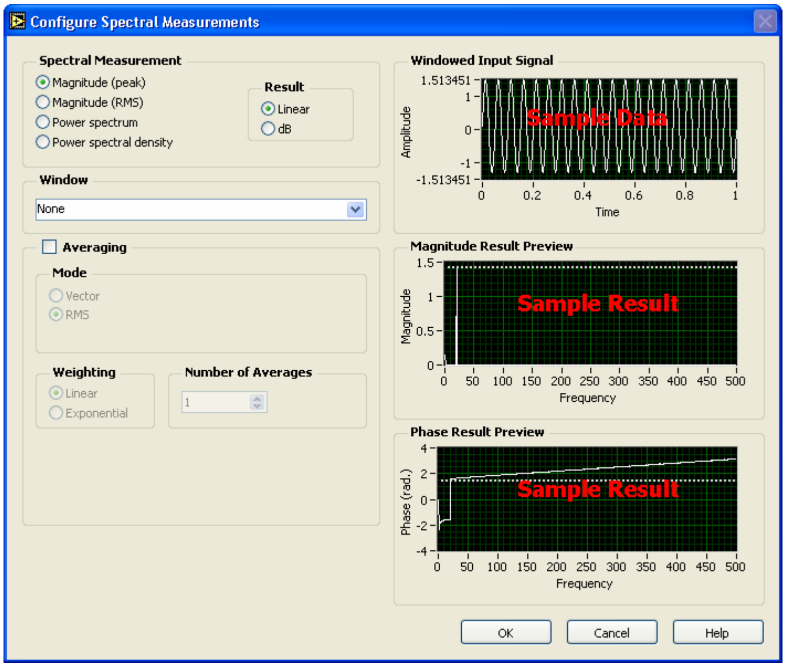
Number of samples: 128 □ Automatic

□ Integer number of cycles

Actual number of samples: 128

Actual frequency: 1000

Figure 13-1. Configuration dialog of Simulate Signal Express VI.



Configure Spectral Measurements

Spectral Measurement

● Magnitude (peak)
○ Magnitude (RMS)
○ Power spectrum
○ Power spectral density

Result

● Linear
○ dB

Window: None

□ Averaging

Mode

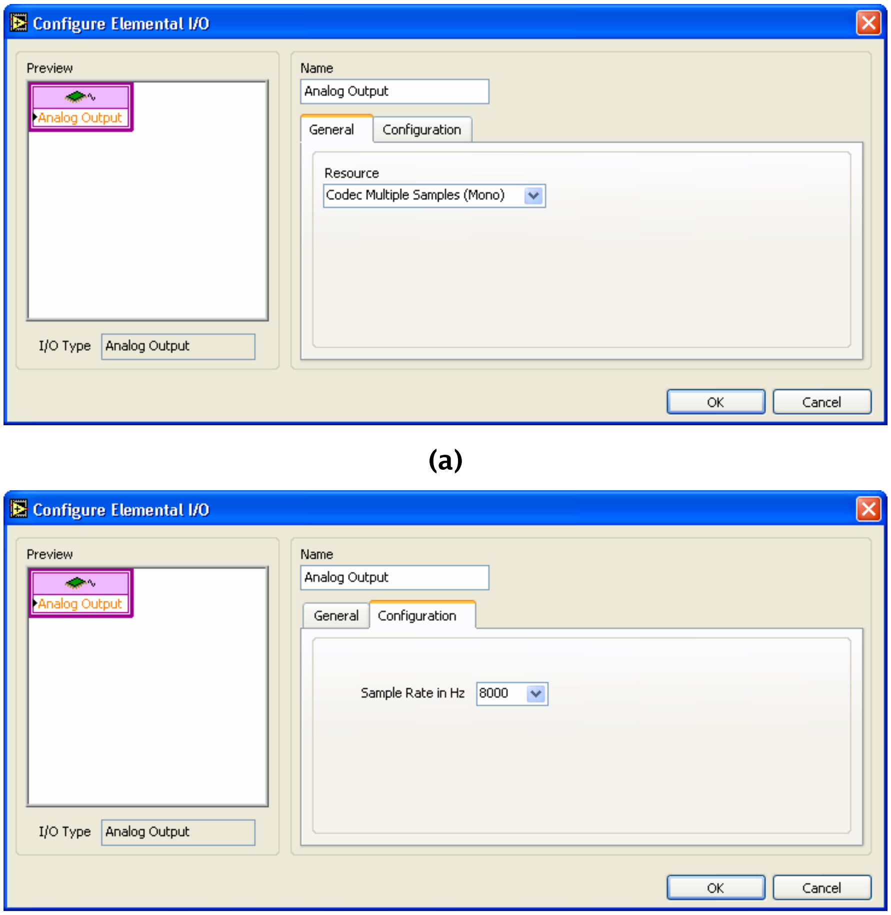
○ Vector
● RMS

Weighting

● Linear
○ Exponential

Number of Averages: 1

Figure 13-2. Configuration dialog of Spectral Measurements Express VI.



Configure Elemental I/O

Preview

Name: Analog Output

General | Configuration

Resource: Codec Multiple Samples (Mono)

I/O Type: Analog Output

OK | Cancel

(a)

Configure Elemental I/O

Preview

Name: Analog Output

General | Configuration

Sample Rate in Hz: 8000

I/O Type: Analog Output

OK | Cancel

(b)

Figure 13-3. Configuration dialog of Elemental I/O function: (a) output type (b) sampling frequency.

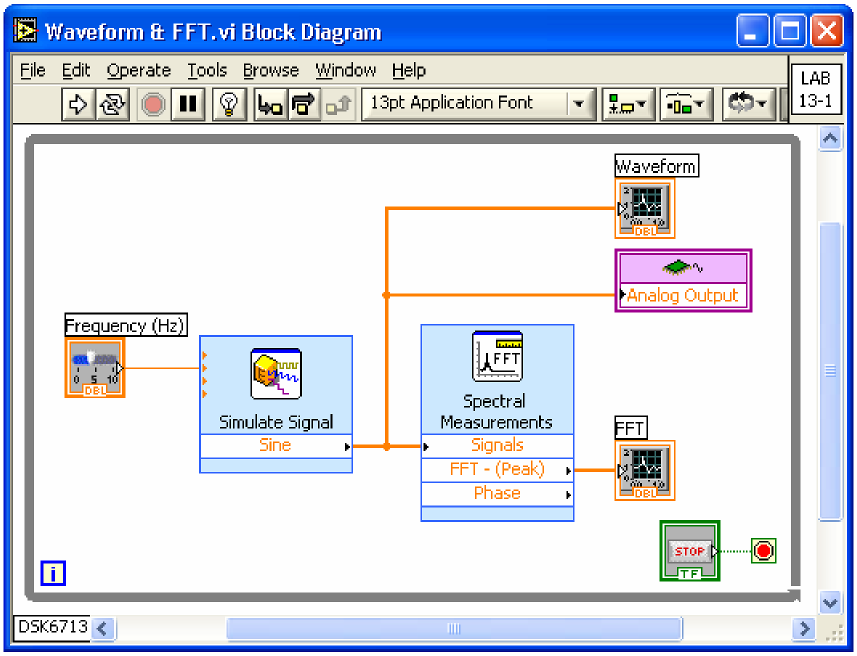


Figure 13-4. BD of Waveform & FFT VI.

By clicking the **Run** button, the VI gets compiled, downloaded, and executed on the DSP target. Also, a LabVIEW DSP Module Status Monitor window is brought up displaying the status of the code. One should be able to see the FP shown in Figure 13-5.

Connect a speaker to the Line Out port of the DSK board in order to hear the generated tone signal. However, one hears ticks coming out of the speaker instead of the tone signal. This is due to the slow communication speed between the DSK and the host PC in updating the FP. To resolve this, check the **Disconnect Front Panel** box on the LabVIEW DSP Module Status Monitor window, see Figure 13-6. This allows the graph update on the FP to be disabled and the tone signal to be properly heard through the speaker.

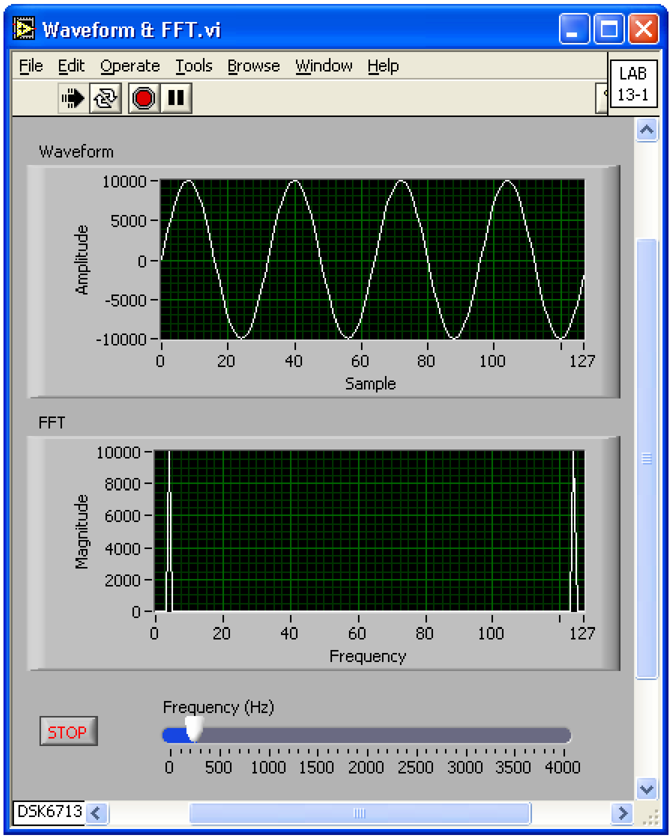
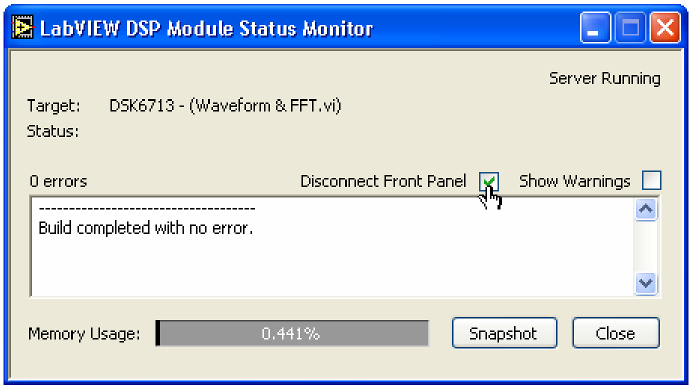


Figure 13-5. FP of Waveform & FFT VI.



Server Running

Target: DSK6713 - (Waveform & FFT.vi)

Status:

0 errors Disconnect Front Panel ☑ Show Warnings □

```
-------------------------------------
Build completed with no error.
```

Memory Usage: 0.441%

Figure 13-6. Disconnect Front Panel.

The frequency control on the FP can be adjusted to alter the tone frequency while the VI is running. Note that the FP controls and indicators/graphs can get updated by clicking the **SnapShot** button.

L13.2 Digital Filtering

This section provides two approaches for implementing a digital filtering system using the LabVIEW DSP Module.

L13.2.1 Digital Filtering Using Filter Express VI

One approach involves modifying the `Waveform & FFT` VI presented in the previous section by adding the `Filter` Express VI (**Functions** → **Signal Processing** → **Filters** → **Filter**), see Figure 13-7.

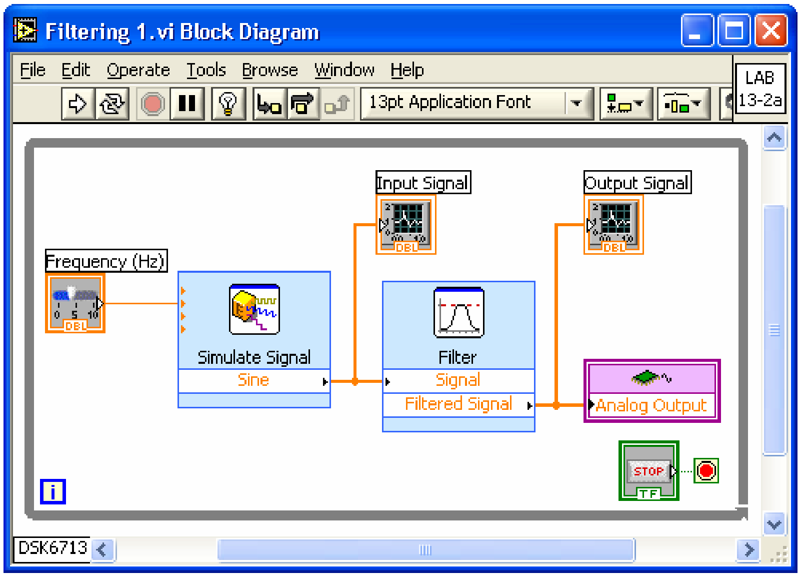
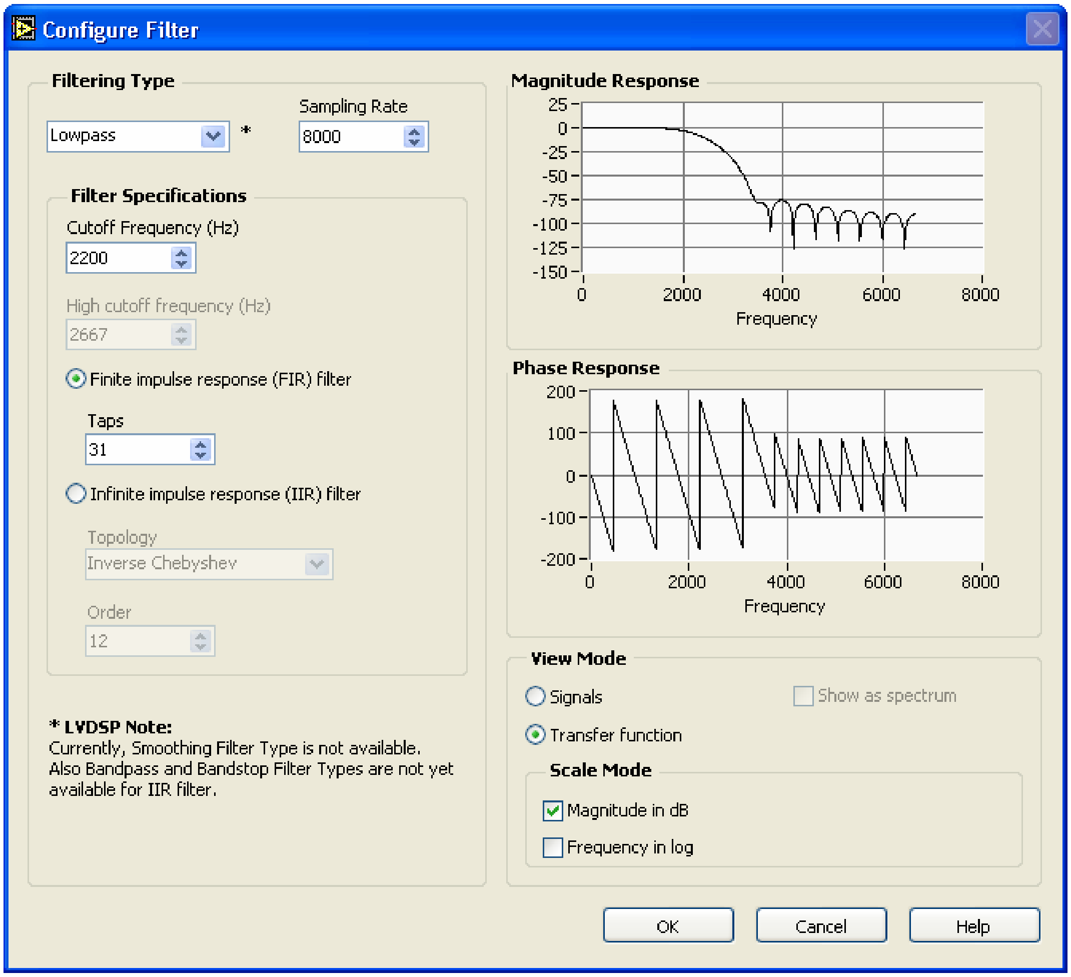


Figure 13-7. Digital Filtering using Filter Express VI.

As an example, let us design a lowpass filter with the cut-off frequency of 2200 Hz. In the configuration window of the Express VI, the specification of the filter can be adjusted in an interactive graphical way, see Figure 13-8.



Configure Filter

Filtering Type

Lowpass *

Sampling Rate: 8000

Filter Specifications

Cutoff Frequency (Hz): 2200

High cutoff frequency (Hz): 2667

● Finite impulse response (FIR) filter

Taps: 31

○ Infinite impulse response (IIR) filter

Topology: Inverse Chebyshev

Order: 12

*** LVDSP Note:**
Currently, Smoothing Filter Type is not available.
Also Bandpass and Bandstop Filter Types are not yet available for IIR filter.

View Mode

○ Signals

□ Show as spectrum

● Transfer function

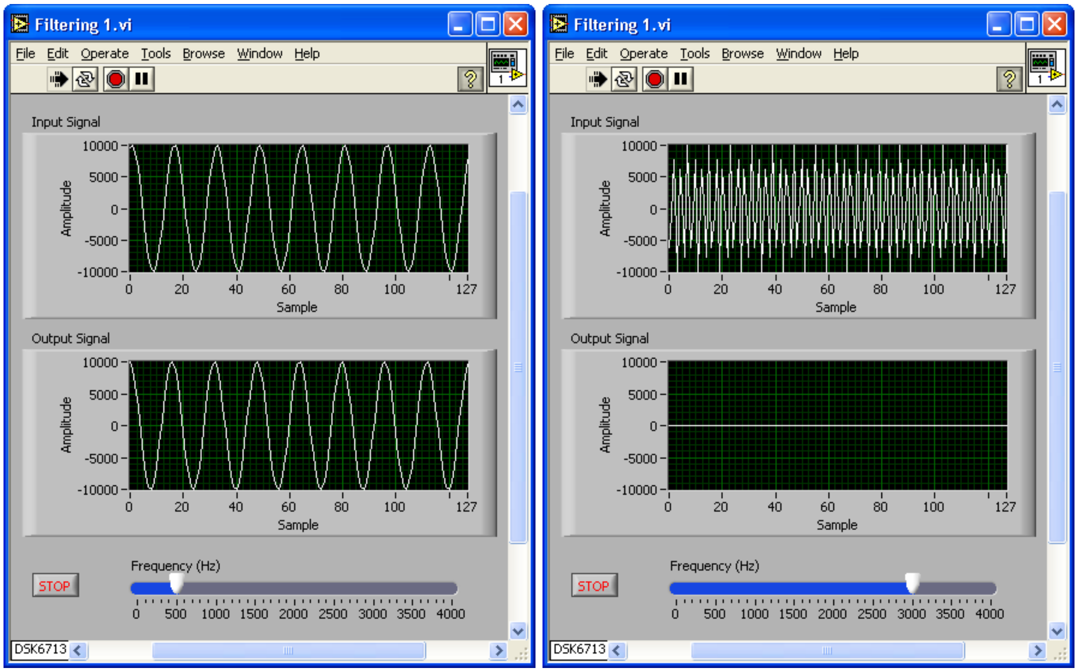
Scale Mode

☑ Magnitude in dB

□ Frequency in log

Figure 13-8. Configuration dialog of Filter Express VI.

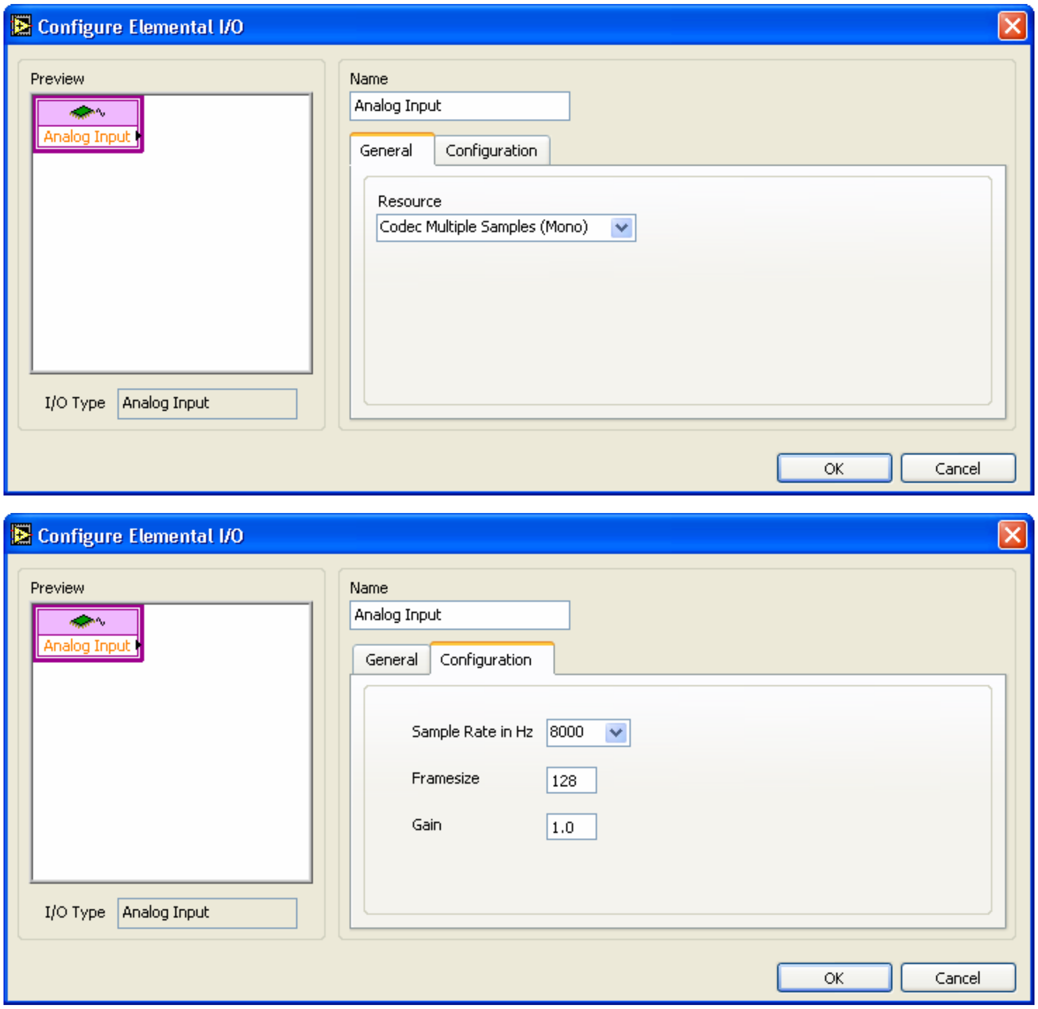
Two instances of the generated input and output signals are shown in Figure 13-9. Figure 13-9 (a) illustrates the passband input signal, the 500 Hz signal, and its filtered version, while Figure 13-9 (b) illustrates the stopband input signal, the 3000 Hz signal, and its filtered version.



(a) (b)

Figure 13-9. BD of Waveform & FFT VI: input and output signal in (a) passband (b) stopband.

Next, instead of using a simulated signal, the above filtering system is modified to process an incoming signal originating from the input port of the DSK board. This requires replacing the waveform simulation portion of the BD with an `Analog Input` function (**Functions → Elemental I/O → Analog Input**). Configure the Elemental I/O function as shown in Figure 13-10. The BD incorporating this modification is shown in Figure 13-11. By connecting a microphone or a signal source to the MIC In port of the DSK board, the filtering can be done by the DSP on the board.



Name: Analog Input

Resource: Codec Multiple Samples (Mono)

I/O Type: Analog Input

Name: Analog Input

Sample Rate in Hz: 8000

Framesize: 128

Gain: 1.0

I/O Type: Analog Input

Figure 13-10. Configuration dialog of Elemental I/O function.

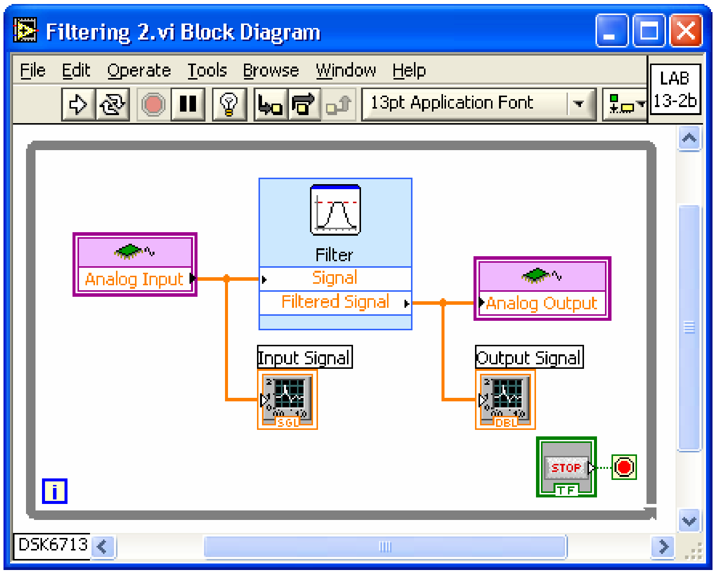
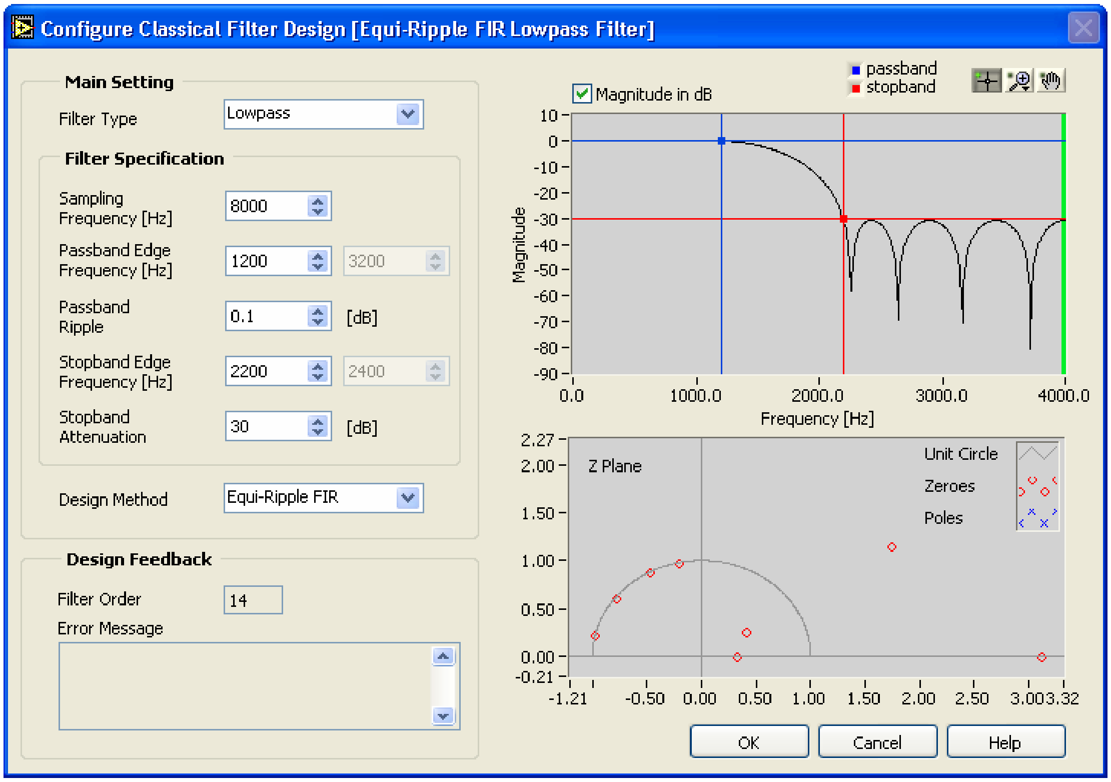


Figure 13-11. Filtering system with input and output signals.

L13.2.2 Digital Filtering using DFD Filter Express VI

The second approach of implementing digital filters involves using the `DFD Filter` Express VI of the LabVIEW DSP Module, which utilizes the Digital Filter Design toolkit. Let us begin by designing the lowpass filter covered in Lab 4. Use the regular LabVIEW, not the LabVIEW Embedded Edition, in order to design the filter with the DFD toolkit. In a blank VI, locate the `DFD Classical Filter Design` Express VI (**Functions** → **All Functions** → **Digital Filter Design** → **Filter Design** → **DFD Classical Filter Design**) and configure it as shown in Figure 13-12. Place the `DFD Save to File` VI (**Functions** → **All Functions** → **Digital Filter Design** → **Utilities** → **DFD Save to File**) and wire the `filter in` node of the VI to the `filter out` node of the `DFD Classical Filter Design` Express VI, refer to Figure 13-13.



Configure Classical Filter Design [Equi-Ripple FIR Lowpass Filter]

Main Setting

Filter Type: Lowpass

Filter Specification

Sampling Frequency [Hz]: 8000

Passband Edge Frequency [Hz]: 1200 | 3200

Passband Ripple: 0.1 [dB]

Stopband Edge Frequency [Hz]: 2200 | 2400

Stopband Attenuation: 30 [dB]

Design Method: Equi-Ripple FIR

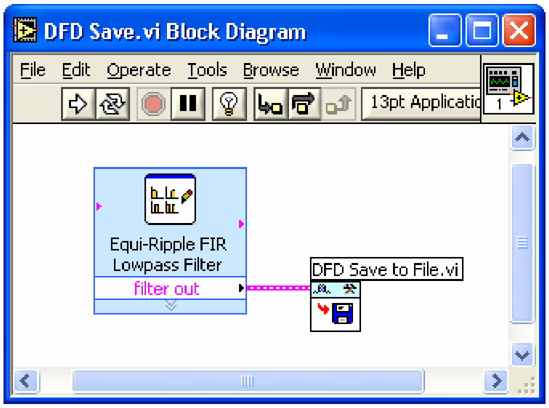
Design Feedback

Filter Order: 14

Error Message

☑ Magnitude in dB

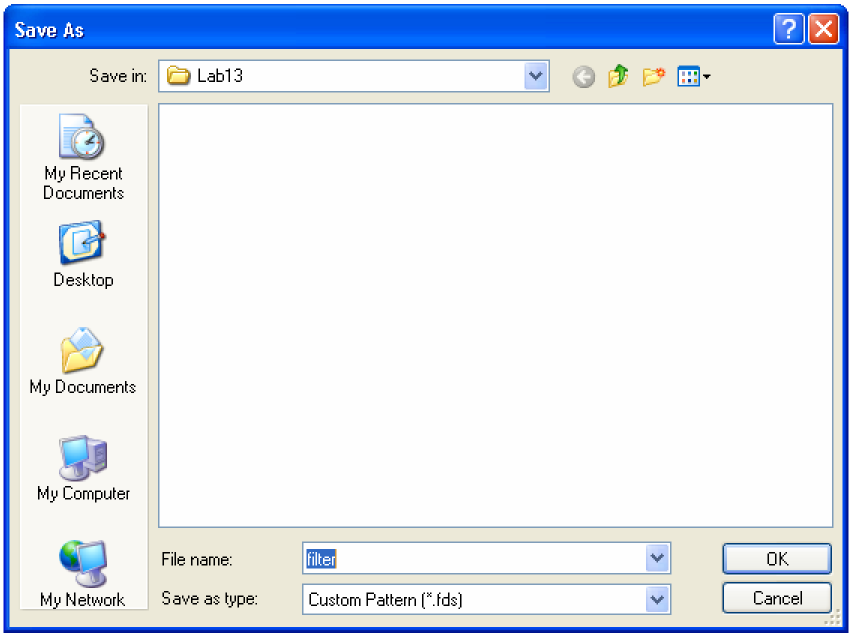
Figure 13-12. Configuration dialog of DFD Classical Filter Design Express VI.



DFD Save.vi Block Diagram

Figure 13-13. Filter Design with DFD toolkit.

Upon running the VI, a window is brought up which asks the file path for saving the designed filter. Type `filter.fds` as the filename.



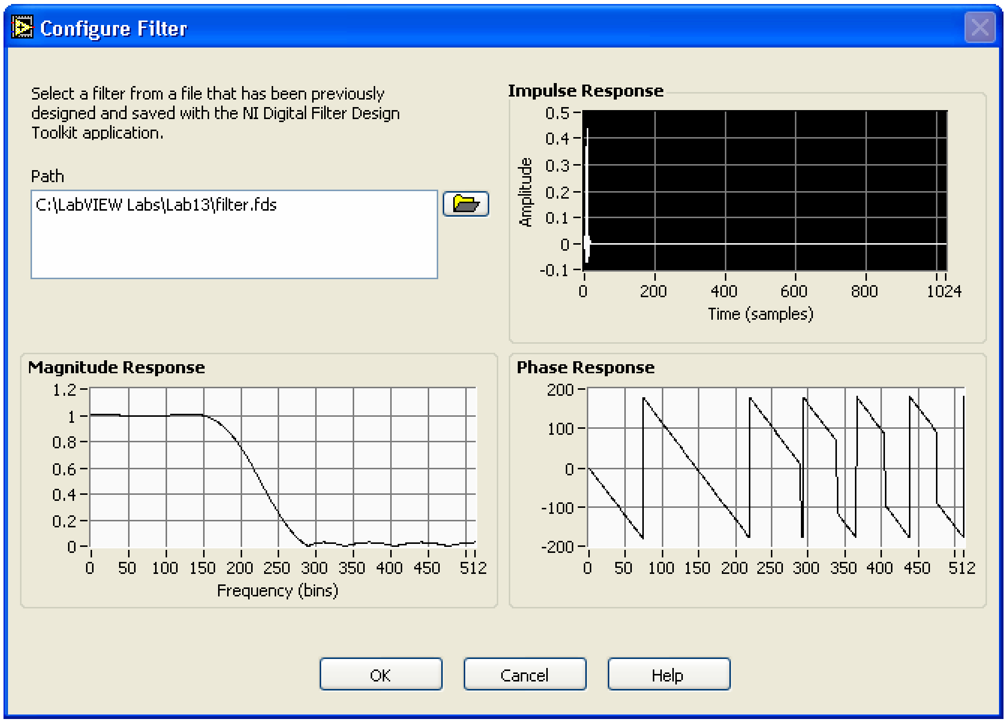
Save in: Lab13

File name: filter

Save as type: Custom Pattern (*.fds)

Figure 13-14. Save a designed filter.

Now, close all the windows associated with LabVIEW and run the Embedded Edition to build the filter by using the `DFD Filter` Express VI of the LabVIEW DSP Module. Replace the `Filter` Express VI in Figure 13-7 with the `DFD Filter` Express VI and double-click on it to configure the file path of the filter. If the filter is successfully linked, its response should appear as shown in Figure 13-15. The complete BD of the filtering system based on the `DFD Filter` Express VI is shown in Figure 13-16.

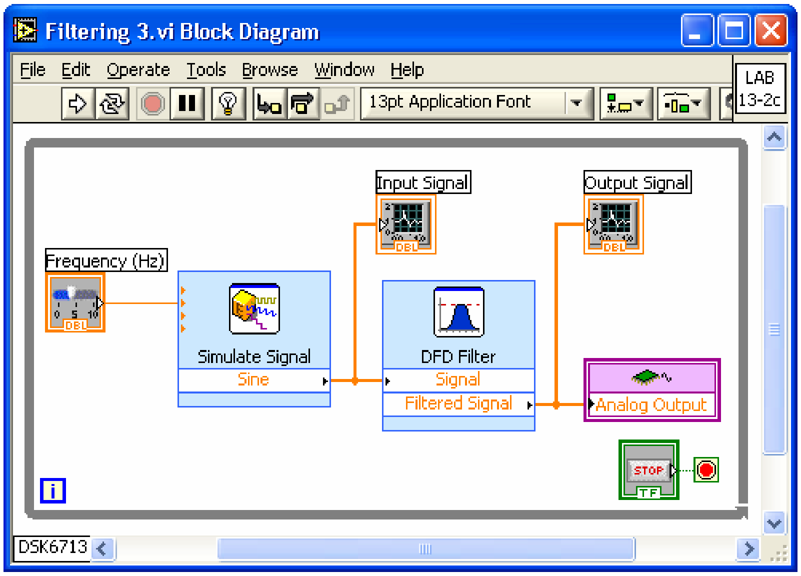


Configure Filter

Select a filter from a file that has been previously designed and saved with the NI Digital Filter Design Toolkit application.

Path: C:\LabVIEW Labs\Lab13\filter.fds

Figure 13-15. Configuration dialog of DFD Filter Express VI.



Filtering 3.vi Block Diagram

Figure 13-16. Filtering system using DFD Filter Express VI.

L13.3 Adaptive Noise Cancellation

The adaptive noise cancellation system covered in Lab 6 is repeated here using the LabVIEW DSP Module. The BD of the adaptive noise cancellation system is illustrated in Figure 13-17. Let us briefly mention the VIs and functions of this BD.

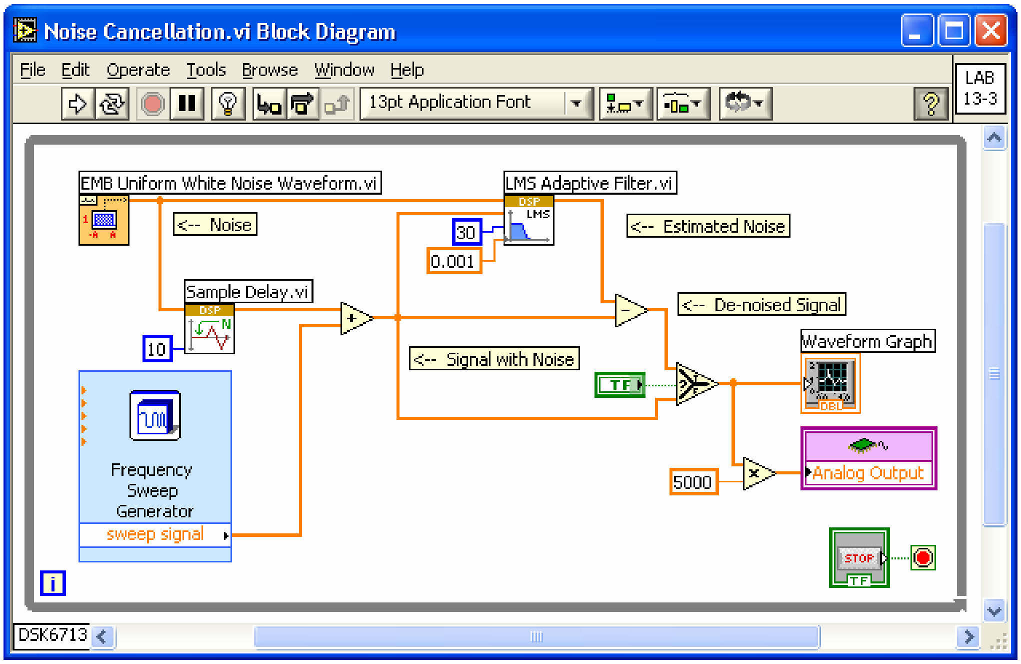
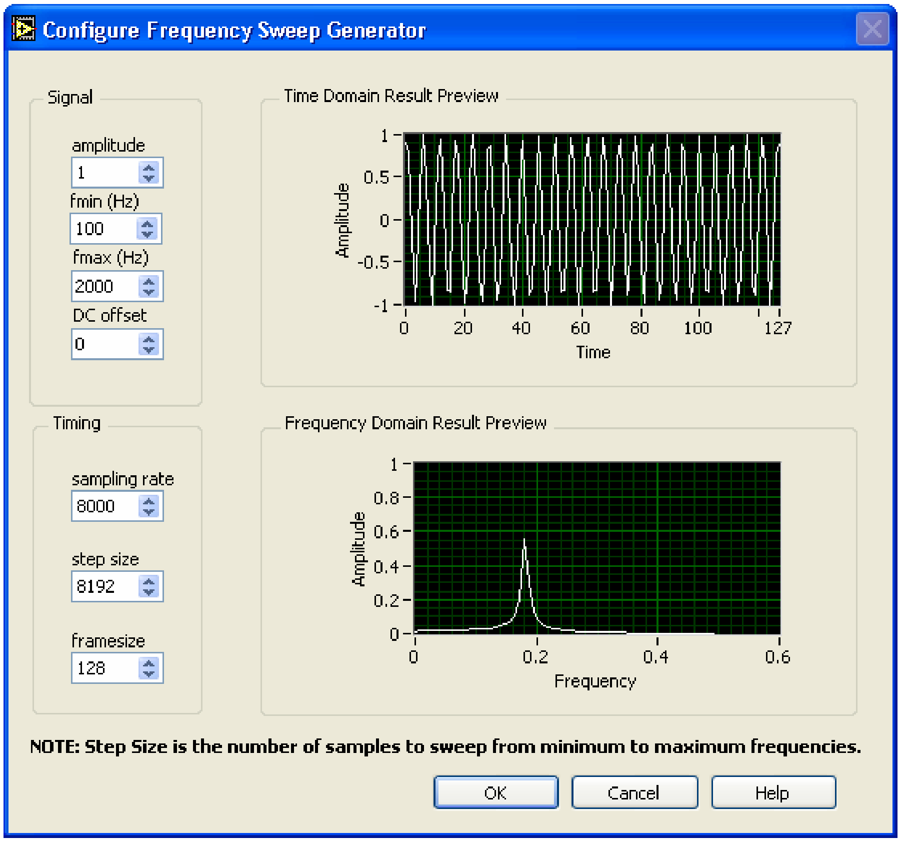


Figure 13-17. Filtering system using DFD Filter Express VI.

The `EMB Uniform White Noise Waveform` VI (**Functions → Embedded Signal Generation → EMB Uniform White Noise Waveform**) is used to generate a white noise signal which is then added to an input signal. Before adding the noise signal, a delay, say 10, is introduced by using the `Sample Delay` VI (**Functions → Signal Processing → Time Domain → Sample Delay**). This is done to simulate an ideal channel which causes a time delay with no gain or frequency change. The input signal consists of a chirp signal whose frequency sweeps between two frequencies. This is implemented by using the `Frequency Sweep Generator` Express VI (**Functions → Embedded Signal Generation → Frequency Sweep Generator**). The configuration of this VI is shown in Figure 13-18. Note that the default values are used for all the fields. This signal and the delayed noise signal are summed together to construct the signal to be processed.



Configure Frequency Sweep Generator

Signal

amplitude: 1

fmin (Hz): 100

fmax (Hz): 2000

DC offset: 0

Timing

sampling rate: 8000

step size: 8192

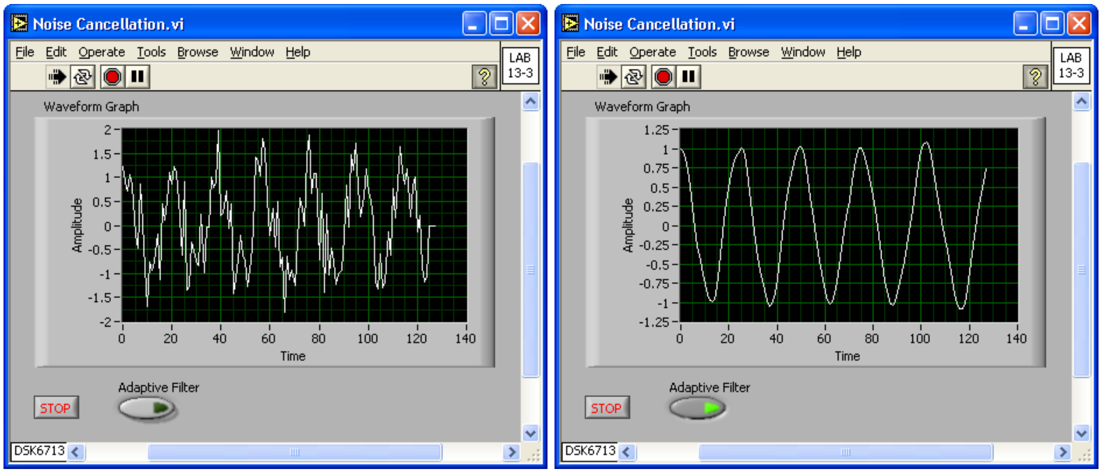
framesize: 128

NOTE: Step Size is the number of samples to sweep from minimum to maximum frequencies.

Figure 13-18. Filtering system using DFD Filter Express VI.

Next, add the `LMS Adaptive Filter` VI (**Functions → Signal Processing → Filters → LMS Adaptive Filter**) to the BD. Wire the noise signal to the `LMS input0` node of the VI to act as the reference signal and wire the summed signal to the `LMS input1` node. Also, wire the numeric constants for the filter order and convergence factor to the `LMS Adaptive Filter` VI. Notice that the output of the `LMS Adaptive Filter` VI corresponds to the estimated noise. Thus, the output of the LMS filter needs to be subtracted from the summed signal to obtain the de-noised signal.

Place the function `Select` to compare the noise corrupted input signal and the de-noised output signal. Either of these signals can be wired to the `Waveform Graph` and `Analog Output` function for sending out data through the codec output port. Figure 13-19 shows the noise corrupted input signal and the de-noised output signal.



(a) **(b)**

Figure 13-19. Adaptive Noise Cancellation signals.