# EEE 304 – Lab 1

## Basic Speech Processing using LabVIEW

# 1. Introduction

This lab introduces some fundamental concepts in National Instruments LabVIEW, through a simple programming example. LabVIEW stands for **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. It is a graphical programming language and has the ability to interface with a wide range of external devices. LabVIEW programs are called Virtual Instruments (VIs). Each VI has three main parts: the block diagram, the front panel and the icon/connector. Each VI in turn can contain sub VIs and other structures. Controls and indicators on the front panel allow the user to input data into or extract data from a running virtual environment. The objective of this lab exercise is to explore some of the basics in speech processing using in-built functions in LabVIEW. This exercise also enables the student to

- Gain familiarity with the LabVIEW environment and programming fundamentals.
- Learn to create and debug VI files.
- Build intuitive Front Panel to view the results.

# 2. Basic Concepts

### *2.1 LabVIEW terms*

Each VI contains three main parts

- *Front Panel* – Determines how the user interacts with the UI.
- *Block Diagram* – Code that controls the program.
- *Icon/Connector* – Means of connecting a VI to other VIs.

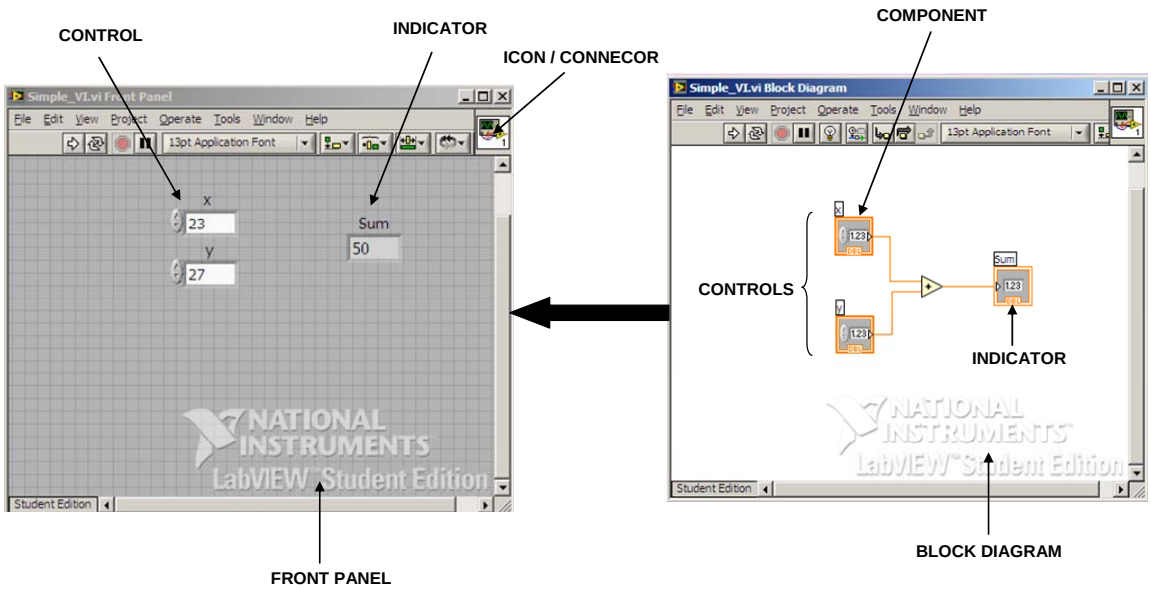



**Figure 1. Illustration of components of a VI**

The ***Front Panel*** is used to interact with the user when the program is running. Users can control the program, change inputs and see data updated in real-time. ***Controls*** are used as inputs and ***Indicators*** are used as outputs. Block diagram is the accompanying program for the front panel. A block diagram is created by wiring the LabVIEW components together. When a VI runs, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed to other functions or indicators.

An example of a VI is shown in Figure 1. Let us get started with a very simple LabVIEW exercise to gain familiarity.

**Example :**

- Open LabVIEW 8.5
- Open a Blank VI
- You should be able to see two windows open. a) Front Panel and b) Block Diagram. In case you do not see the block diagram, you can press **CTRL+E**.
- In the ***Front Panel***, when you right click, you should be able to see the ***Controls*** palette.
- In the Controls palette, click on ***Modern >> Numeric >> Numeric Control*** and place it on the Front Panel.

The *Controls Palette* can be used to place controls and indicators on the front panel. Select ***Window>>Show Controls Palette*** or right-click the front panel workspace to display the palette. Similarly, the *Functions Palette* can be used to build the block diagram. It contains the LabVIEW's native components listed under different categories. Select ***Window>>Show Functions Palette*** or right-click on the block diagram workspace to display the palette. *Tools palette* contains the set of tools that can be used to edit and build the block diagram. If automatic tool selection is enabled and we move the cursor over the objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the palette. The VI can be executed by clicking the **Run** button in the status toolbar.

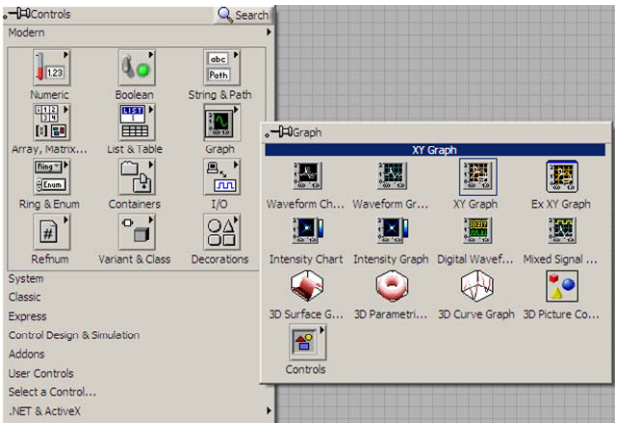



**Figure 2(a) Controls Palette**

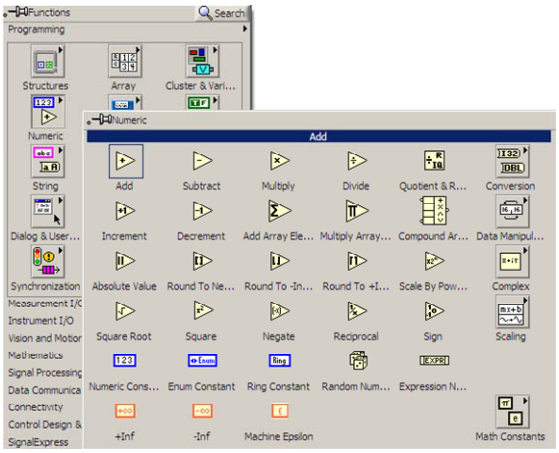



**Figure 2(b) Functions Palette**


**Figure 2(c) Tools Palette**

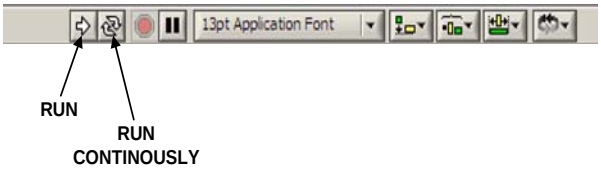



**Figure 2(d) Status Toolbar**

### *2.2 Creating a VI*

When an object is created on the front panel, a terminal will be created on the block diagram. These terminals give access to the front panel objects from the block diagram code. Each terminal contains useful information about the front panel object. For example, the color and symbols provide the data type. Figure 3 illustrates the some of the different data types supported.

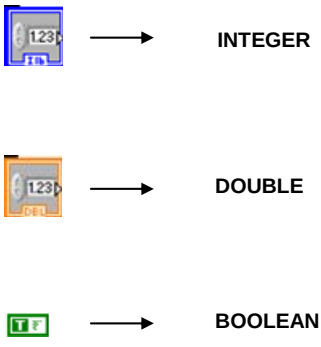



**Figure 3. Illustration of examples of terminals**

Hence, it is obvious that orange terminals should wire to orange terminals, green to green and so on. Type-casting can be done similar to the 'C' language. Controls have an arrow on the right side and a thick border. Indicators have an arrow on the left and a thin border. The blocks are connected by wiring the output terminal of one block to the input terminal of the other block. For a brief description of the functions, place the desired function in the block diagram workspace and select ***Help>>Show Context Help***. When you move the mouse over the function, the description appears in the Context Help window.

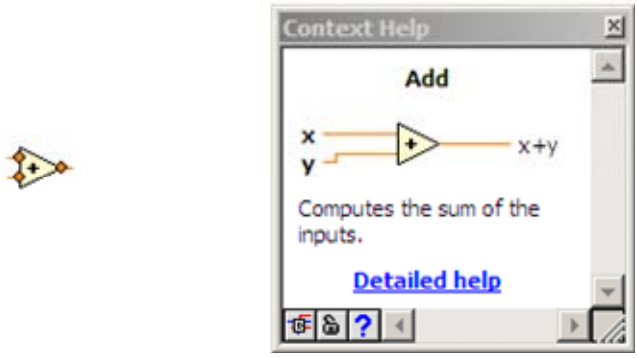


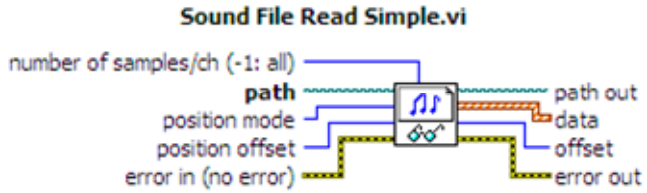

**Figure 4. Using the Context Help**

When the created VI is not executable due to errors or missing components, a broken arrow is displayed in the Run button in the status bar. Clicking on the broken arrow, will list the errors. The bad object is located by clicking on the error message.

# 3. LabVIEW functions used in this exercise

### *3.1 Sound File Read Simple*

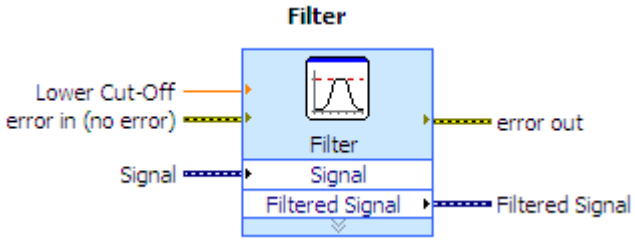
This function reads data from a .wav file into an array of waveforms. This block opens, reads and closes the .wav file automatically.




It is very important to note that, the output of this is not a single array of scalars but an array of waveforms. Please note that in addition to the standard data types, LabVIEW supports several custom defined data types and waveform is one such data type. To use this data for array processing functions like finding length of the data, you need to convert the 1D-array of waveforms into a double array. This function can be found under the category ***Programming>>Graphics & Sound>>Sound>>Files>>Input***.

### *3.2 Filter (Express VI)*

This function processes signals through filters and windows. In this exercise, we are interested in building a low pass filter. A low-pass filter is a filter that passes low-frequency signals but attenuates the signals with frequencies higher than the cutoff frequency. The actual amount of attenuation for each frequency varies from filter to filter.



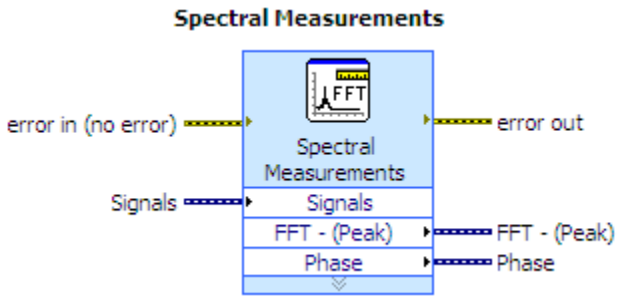

For the given sampling frequency *fs* (8000 Hz), the cut-off frequency of the low pass filter needs to be in the range $0 < f_c \leq fs/2$. The cut-off frequency of the filter is allowed to change dynamically in this exercise through a numeric control. Double-clicking on the function opens a *Configure Dialog* window and you need to provide the filter specifications using that. The parameters that need to be provided in the *Configure Dialog* are as follows

- Filter Type : Lowpass
- IIR/FIR : Finite Impulse Response (FIR) filter
- Taps : 11

As explained, the numeric control that is used to control the cut-off frequency can run from 10Hz to a maximum of 4000 Hz. This function can be found in the functions palette under the section ***Express>>Signal Analysis***.

### *3.3 Spectral Measurements (Express VI)*

This function performs FFT-based spectral measurements, such as the averaged magnitude spectrum, power spectrum, and phase spectrum of a signal. We double-click on this block and choose the Magnitude (peak) measurement in the dialog for this exercise. This measures the spectrum and displays the results in terms of peak amplitude. You typically use this measurement with more advanced measurements that require magnitude and phase information. The magnitude of the spectrum is measured in peak values. For example, a sine tone of amplitude A yields a magnitude spectral value of A at the sine tone frequency. This function can be found in the functions palette under the section ***Express>>Signal Analysis.***

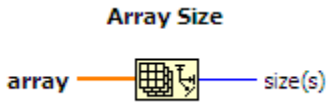



For this block, choose the following options in the dialog

- Spectral Measurement : Magnitude (Peak)
- Result : dB
- Window : Hamming

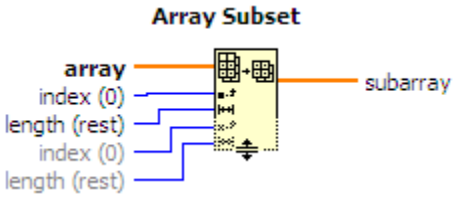
### *3.4 Array Size*

This function returns the number of elements in each dimension of the array. This can be found in the functions palette under the section ***Programming>>Array***.




### *3.5 Array Subset*

This function returns a portion of the array starting at *index* and containing *length* elements. This can be found in the functions palette under the section ***Programming>>Array***.

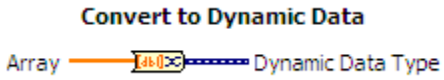



### *3.6 Play Waveform (Express VI)*

This function plays the sound output device using finite sampling. This express VI automatically configures an output task and clears the task after the output completes. This can be found in the functions palette under the section ***Express>>Output***.

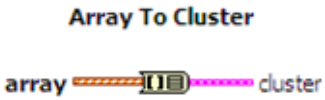
### *3.7 Convert to Dynamic data (Express VI)*

The data input that Express VIs requires is of the *dynamic data type*. This function converts numeric, Boolean, waveform and array data types to the dynamic data type for use with the Express VIs. This can be found in the functions palette under the section ***Express>>Signal Manipulation***.



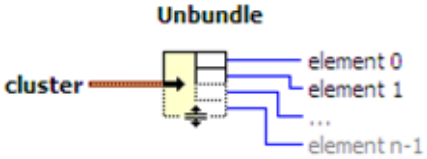

### *3.8 Array to Cluster*

A ***cluster***, analogous to a *struct* in C programming, combines one or more data types into one data type. The cluster may contain different data types such as boolean, string and integer. The wire on the block diagram that carries data from a cluster can be thought of as a bundle of smaller wires like a telephone cable. This reduces clutter on the bock diagram. You can unbundle a cluster to access its individual data components. This function converts a 1D array to a cluster of elements of the same type as the array elements. Right-click on the function and select the option *Cluster Size* from the shortcut-menu. Type the value '1' for the number of elements in the cluster. This can be found under ***Programming>>Array.***



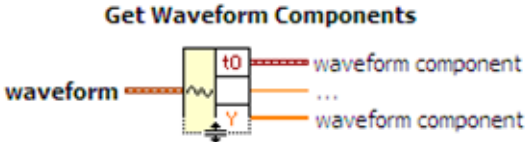

### *3.9 Unbundle*

This function splits a cluster into each of its individual elements (**Programming>>Cluster**)




### *3.10 Get Waveform Components*

This function returns the components of the waveform (**Programming>>Waveform**).




## 4. Exercise

The exercise involves building a VI to load and display real-time speech data frame-by-frame. In addition, the program will filter the input speech using a low pass filter, whose cut-off frequency can be dynamically varied. The signal length (in samples) and the frame count (depends on the

frame size) will also displayed in the Front Panel. An option to playback the filtered speech will also be provided. The guidelines to build the front panel and the block diagram are provided in this section. The layout of the block diagram (code) to perform the tasks specified is provided in Figure 5.

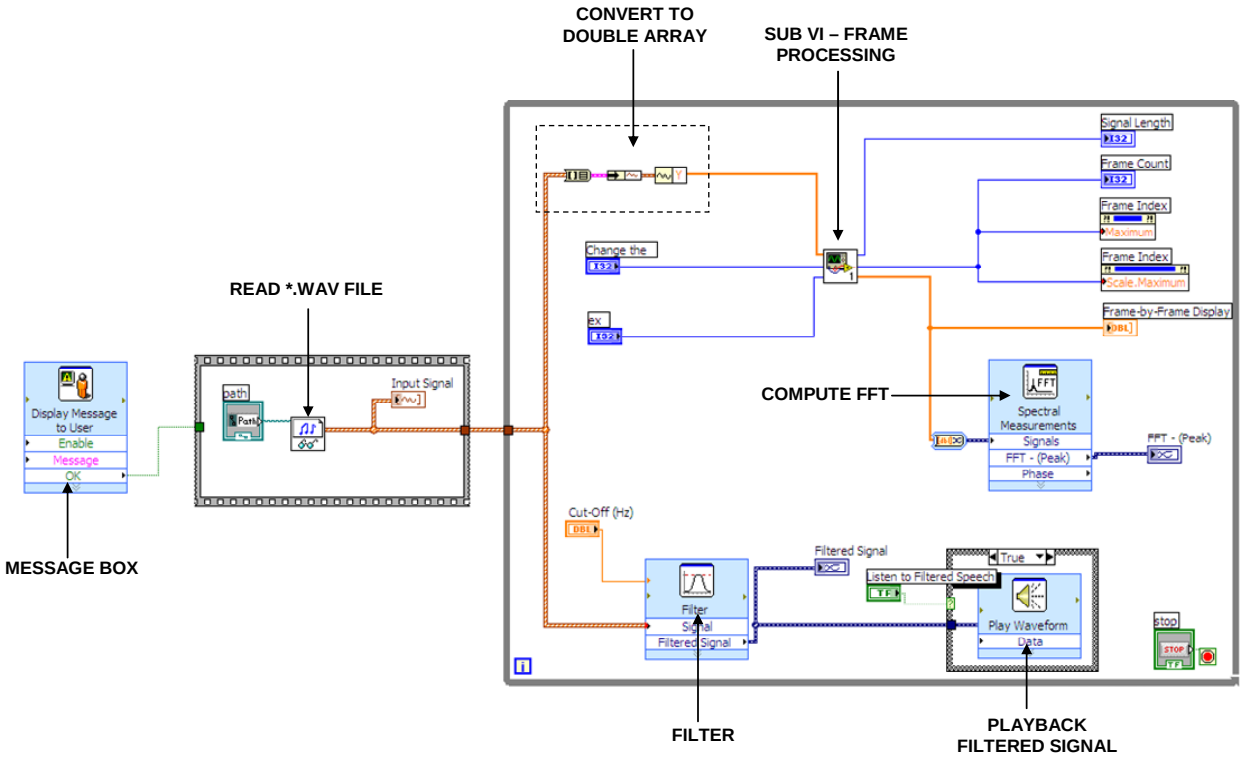



**Figure 5. Layout of Block Diagram for this exercise**

### *4.1 Build a VI*

- Run National Instruments LabVIEW in your system
- Select **File>>New VI** to open a new Front Panel
- Save the VI in your current working directory as *Exercise.vi*
- Select **Window>>Show Block Diagram** or press **ctrl+E** to switch to the Block Diagram workspace
- You will now create a message box to be displayed when the VI is executed. Right-click on the workspace and choose the **Dialog & User Interface** section under the category **Programming**. Select the *express* VI under this section, **Display Msg**, which will allow you to do this.
- Click the selected function again to place it in your block diagram.
- In the dialog that pops up, type the message *"This is my first Speech Processing Exercise"* and click OK to accept the change and save the VI.
- Switch to the Front Panel (**ctrl+E**) and click the **Run** button in the status bar.

What do you observe on executing this VI?

### *4.2 Acquire Sound*

- Place the ***Sound File Read Simple*** function in your block diagram. This function is explained in section 3.1.
- Right-click on the input terminal ***Path*** and choose **Create>>Control**. This creates a control on the Front Panel using which you can specify the path of the speech file (*.wav*)
- Connect the output terminal ***data*** of the function to a waveform graph. You can create a waveform graph in the Front Panel by right-clicking and choosing **Modern>>Graph>>Waveform Graph** from the Controls Palette. This will display the input speech acquired from the user.
- Change the label of the graph to *Input Speech*, by double clicking on the label (either in the Front Panel or the Block Diagram). The data that is passed by this function is *1D array of waveforms*.
- To convert the data obtained into an array of scalars to perform basic array processing, convert the data to a cluster using the function ***Array to Cluster*** (configure as shown in section 3.8)**.**
- Pass the cluster obtained in the previous step to ***Unbundle*** and split the cluster into its individual components.
- Finally convert it to a double array by connecting the cluster component to the function ***Get Waveform Components***. The output of this function can be directly used with general array processing functions.

### *4.3 Filter the speech data*

- As explained earlier, in this exercise we intend to dynamically change the cut-off frequency of the low pass filter. To achieve that, you can perform all the processing on the input speech inside a ***While Loop***.
- Create a ***While Loop*** (***Express>>Exec Control***) in your block diagram. You will observe that a Boolean control Stop is automatically created (corresponding to a **Stop** button in the Front Panel). This indicates that the execution will stop, when the **Stop** button is pressed by the user. The size of the while loop can be adjusted using the mouse.
- Place the ***Filter*** function in your block diagram and configure as explained in section 3.2.
- Connect the ***Data*** terminal of the ***Sound File Read Simple*** function to the input terminal ***Signal*** of the ***Filter*** block.
- To change the cut-off frequency dynamically, you need to create a numeric control in you program. Numeric controls can be created either from the block diagram or the Front Panel (Creating in one of them automatically reflects in the other).
- Switch to the Front Panel (**Ctrl+E**) to create the control. Right-click to open the Controls Palette and choose ***Modern>>Numeric>>Vertical Point Slide***.

- Right-click on the control and select **Properties** to configure it. In the ***Appearance*** tab, change the Label to 'Cut-Off (Hz)'. As discussed in section 3.2, change the Minimum and Maximum (under the ***Data Range*** tab) to 10 and 4000 respectively. Also, modify the scale range that appears on the control (***Scale*** tab) to have the values 10 and 4000 for Minimum and Maximum respectively.
- Now switch to the block diagram workspace. To plot the filtered speech, right-click on the output terminal ***Filtered Signal*** of the ***Filter*** function and select ***Create>>Graph Indicator*** from the shortcut-menu.

### *4.4 Playback Filtered Speech*

- The express VI ***Play Waveform*** is used to play back the filtered speech data.
- Since, the VI is continuously running, you need to control the playback using a ***Play*** button (i.e. Play only when desired).
- This can be achieved using a ***Case Structure*** in LabVIEW (***Express>>Exec Control***). Place a Case structure with a Boolean control as the case selector. The control can be created by right-clicking on the input terminal ***Case Selector*** and choosing ***Create>>Control*** from the dropdown menu.
- Double-clicking on it takes you to a newly created button in the Front Panel. You can change this to a button as in Figure 5 from the list of Boolean controls (right-clicking and choosing ***Replace>>Boolean***).
- Switch to the block diagram workspace. Under the case ***True*** of the case structure, place the function ***Play Waveform***. The output of the ***Filter*** function is connected to the input terminal ***Data*** of ***Play Waveform***.

### *4.5 Frame-by-Frame processing of input speech*

- In this section, you will build a sub VI to display the speech data frame by frame in addition to displaying the Frame Count and the Signal Length (samples). You need to construct the part of the VI as shown in Figure 6.
- In the front panel, create numeric controls for ***Frame Index*** (Knob control) and ***Frame Size*** (Numeric control). Right-click on the control *Frame Size*, choose Properties and under the tab Data Entry, provide the values 64 and 256 for the Minimum and Maximum respectively. For both of the, set the *Response to value outside limits* to be Coerce. Similarly, for the control Frame Index, the minimum is fixed at 1 and the maximum will be adjusted dynamically. In addition, under the Scale tab set the minimum value to 1.
- Connect the speech data from the function ***Get Waveform Components*** to the function ***Array Size***.
- The array size gives the length of the speech signal and dividing it by the frame size computes the frame count. Place a ***Divide*** function (***Programming>>Numeric***) and

connect the array size to the first input terminal. Connect the control frame size to the other terminal.

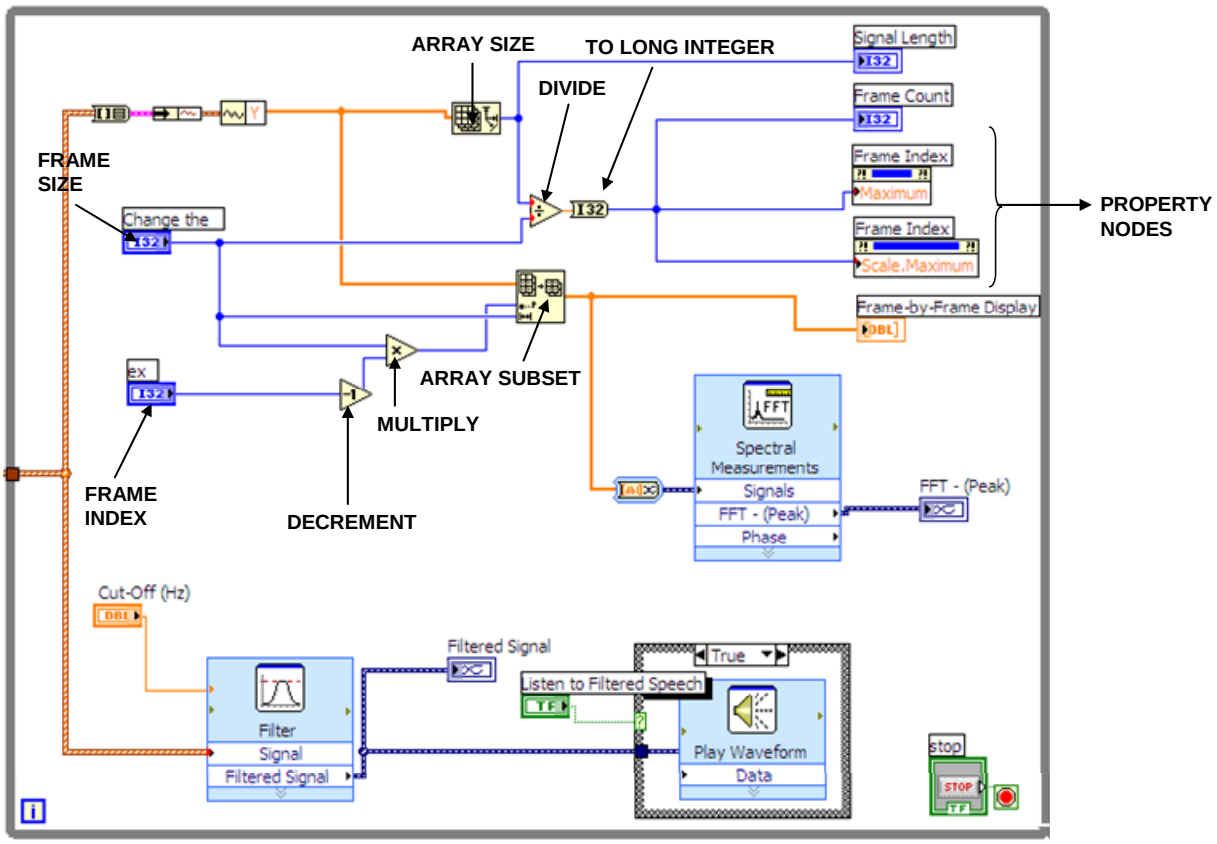



**Figure 6. Block Diagram in Figure 5 with sub VI also expanded**

- To dynamically modify the Maximum range of the numeric control, Frame Index (based on the Frame Count), make use of the property nodes in LabVIEW. To create a property node, right-click on the control and choose ***Create>>Property Node*** and select the desired property from the list.
- To fix the maximum range, right–click on the control Frame Index and place the following two property nodes
    - *Create>>Property Node >>Data Range>>Maximum*
    - *Create>>Property Node >>Scale>>Range>>Maximum*

  Right-click on the property nodes and select ***Change to Write***. Connect the Frame count computed to both these property nodes.
- Obtain the specific frame of data for the Frame index specified using the ***Array Subset*** function. Provide the following inputs to the function
    - array : connect the input speech array.
    - index : (Frame Index-1)*Frame Size
    - length: Frame size

- Finally, pass the array subset computed to the ***Spectral Measurements*** function. This computes the FFT magnitude (dB). To do this, first convert the array data to dynamic data using the function specified in section 3.7. Right-click on the output terminal ***FFT – (Peak)*** and create a graph indicator to display the result.
- The final step of the exercise is to create a subVI for the processing steps created in this section. Figure 7 demonstrates the creation of a subVI. Select the following section of code shown in the figure with the dotted lines around, using your mouse and select the function ***Create SubVI*** from the ***Edit*** menu.

- This replaces the selected unit of code with a subVI as shown in Figure 5. Double-click on the newly created SubVI, change the control and indicator labels to make the connections understandable. Finally, save the subVI as ***Frame_Processing.vi***

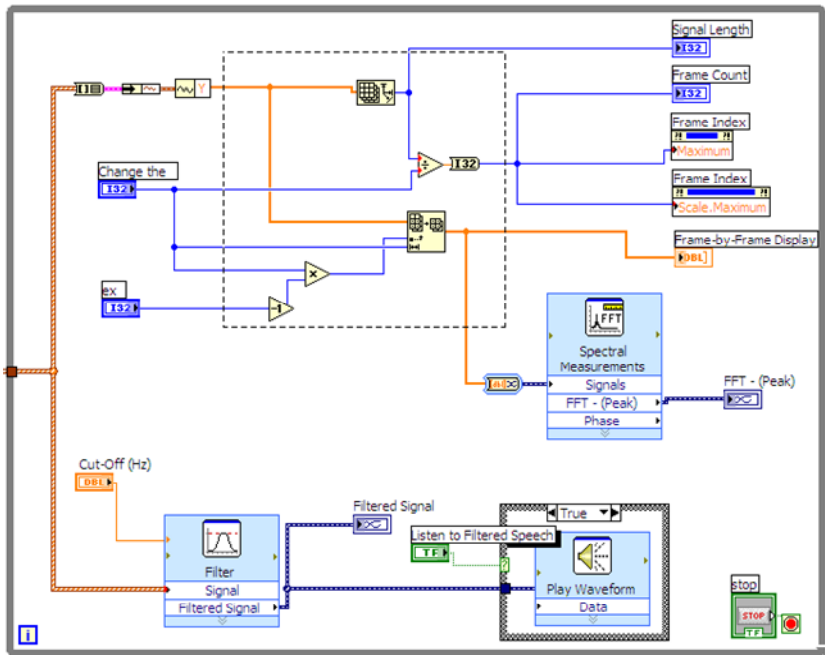



**Figure 7. Creating a SubVI**

## ASSIGNMENT

1. Submit the VIs for the exercise along with a small description of the steps you followed. The Front Panel Layout for this program is shown in Figure 8.

2. What changes do you find in the speech when it is filtered? How do the characteristics vary with change in cut-off frequency?

3. Create another VI by replacing the low pass in the exercise using a high pass filter and submit the VI.

4. Add a section of code to observe the FFT magnitude response of the filtered signal.

5. Write a paragraph explaining what have you learnt from this exercise.

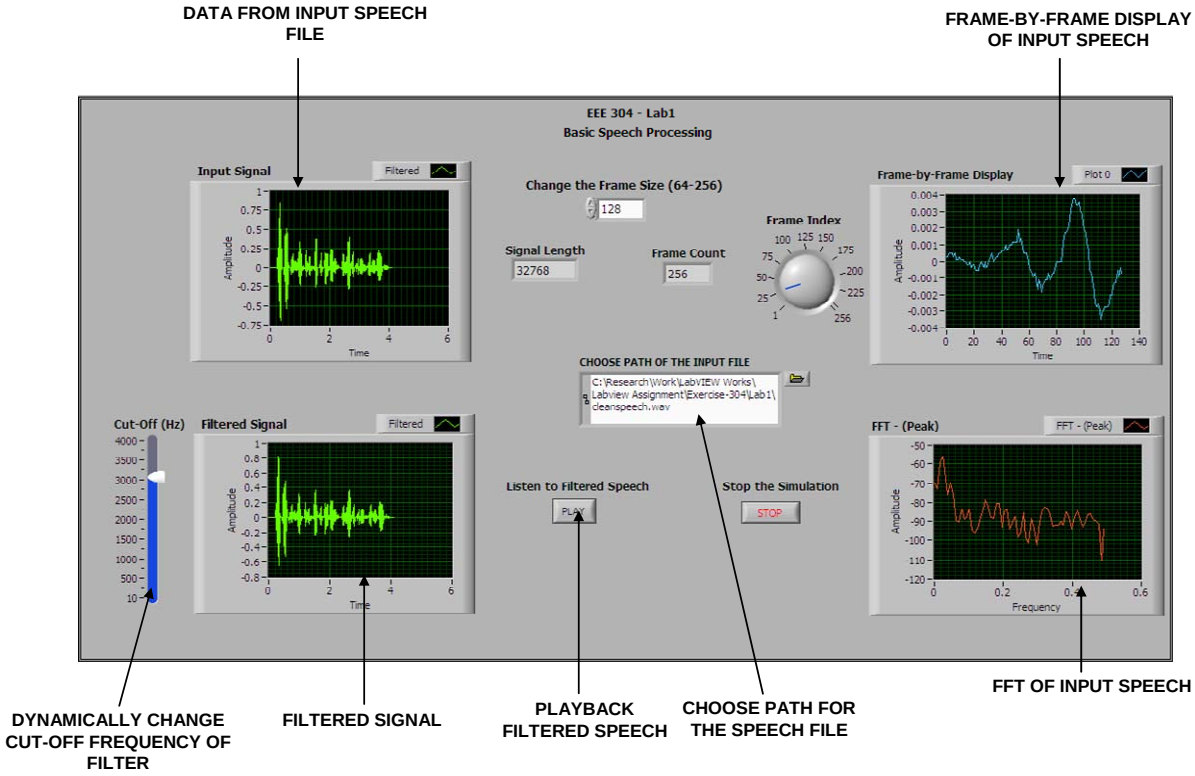



**Figure 8. Front Panel Layout**