Exercise 8-7: **Processing Data for Each Channel in an N-Channel Data Array**

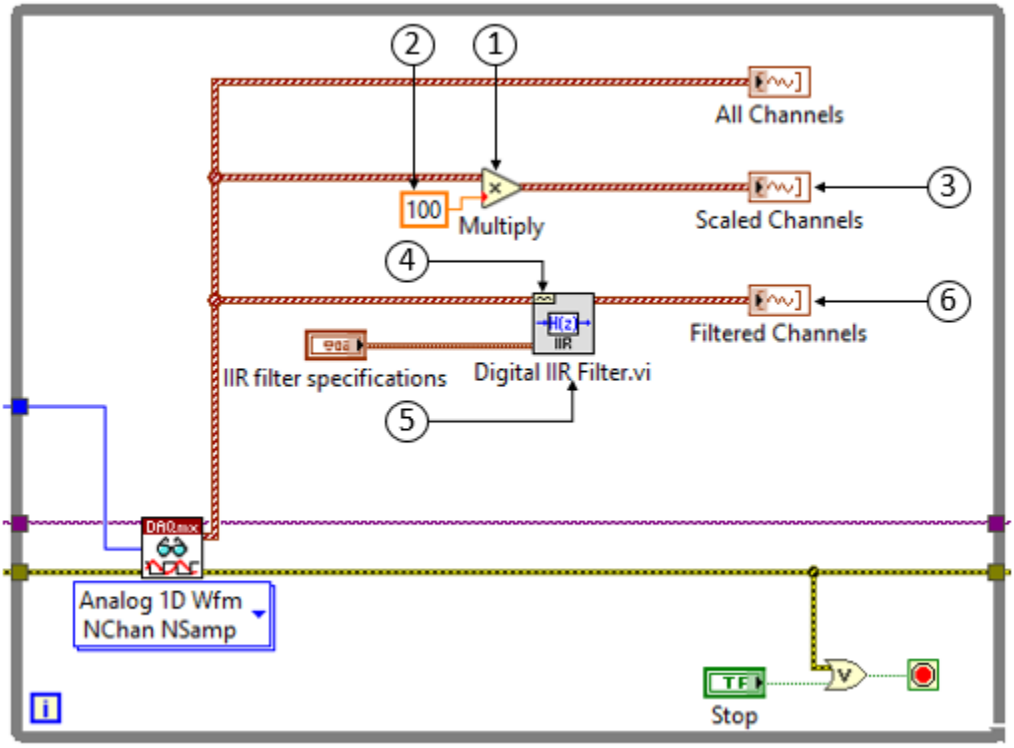
Goal

- Process data for each channel by using a For Loop to iterate through each channel in an *N*-channel data array.

Process Channels Using VIs Compatible with *N*-Channel Array Data Types

1. Open `C:\Exercises\LabVIEW Core 1\Process N-Channel Array\Process N-Channel Array.lvproj`.
2. From the **Project Explorer** window, open Process *N*-Channel Array VI.
3. Examine the VI.
   - Explore the front panel and the block diagram.
   - Notice that the **data** output of the DAQmx Read VI is a 1D waveform array data type.
   - Run the VI.
   - Stop the VI when finished.

4. Modify the block diagram, as shown in the figure below, to process all channels by scaling their values by 100 and filtering the signals using the following items.

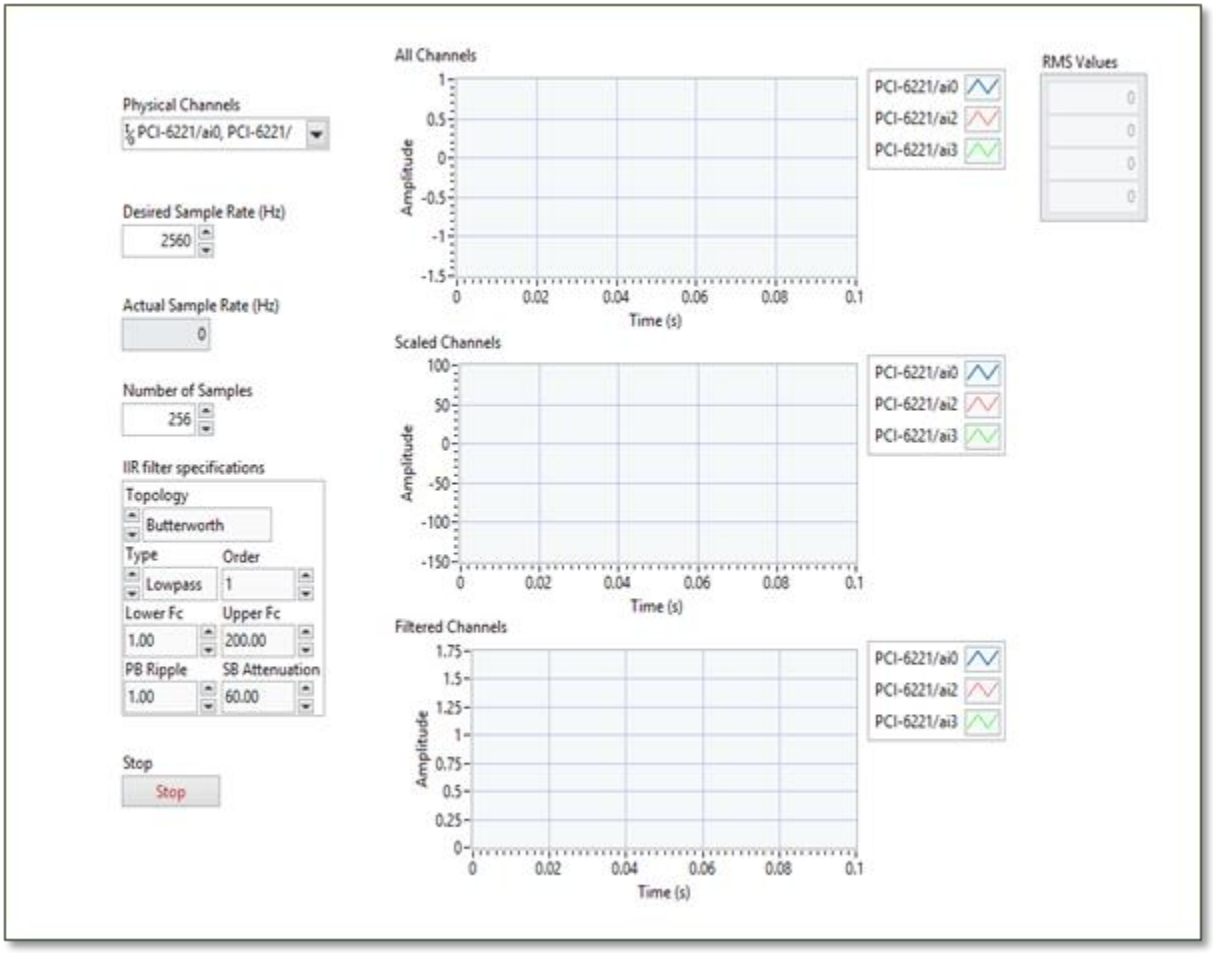


---

1. **Multiply** function – The input of this function can adapt to accept the 1D waveform array data type.
2. Right-click in the **block diagram** and search for the DBL Numeric Constant, then wire it to the second input of the Multiply function. Set the value of the constant to `100`. This function will multiply all Y values in every waveform element in the array by 100.
3. Right-click the **output of the Multiply function** and select **Create Indicator**. Rename the indicator `Scaled Channels`.
4. **Digital IIR Filter VI** – Right-click in the **block diagram**, click **Search** from the Functions palette and type `Digital IIR Filter`. Drag the first appeared result into the while loop. Wire the 1D waveform array from the DAQmx Read VI to the input of the Digital IIR Filter VI. Notice that the Digital IIR Filter VI will process each waveform contained in the array.
5. Right-click the **IIR filter specifications** input and select **Create Control**.
6. Right-click the **signal out** output of the Digital IIR Filter VI and select **Create Indicator**. Rename the indicator as `Filtered Channels`.

Note: Many Math VIs and functions can adapt to accept the 1D waveform array data type. A real-world application can use Math VIs and functions to scale acquired data from units of voltage to the appropriate engineering unit, such as temperature (deg Celsius), acceleration (g), etc.

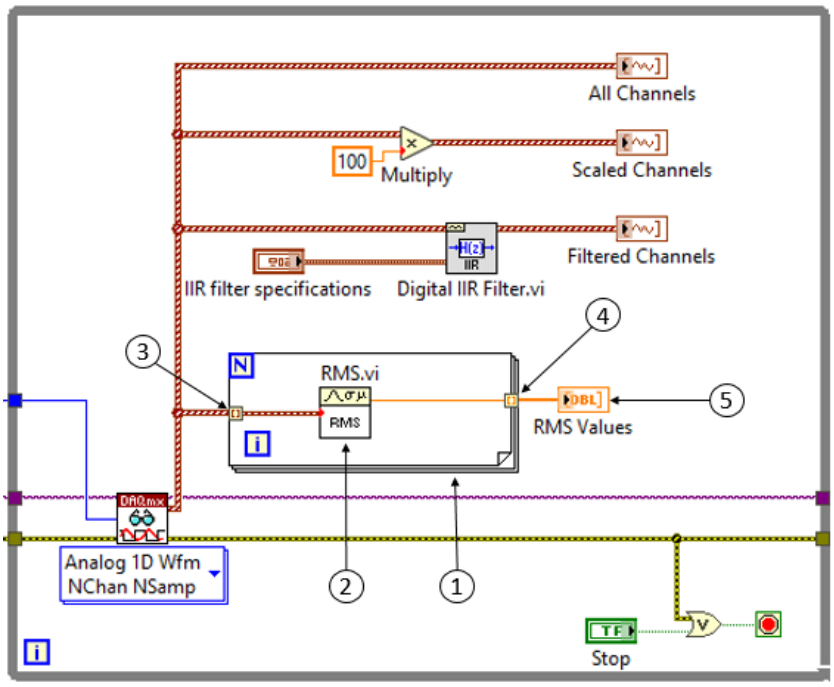
5. Arrange and modify your front panel similar to the following figure. Make sure that the **IIR filter specifications** cluster control is set as shown in the figure.



6. Examine the behavior of the VI.
   - On the front panel set the **Physical Channels** control to **PCI-6221/ai0, PCI-6221/ai2:3.**
     - **(BNC-2120)** Make sure that you have Sine/Triangle BNC Connector connected to the Analog Input 2 and the TTL Square Wave BNC Connector connected to the Analog input 3, also make sure that the Sine/Triangle Waveform Switch is set to Sine.
     - **(BNC-2120 and Simulated Hardware)** Set the value of Lower Fc control to `1`.
   - Run the VI.
   - Notice that the **Scaled Channels** and **Filtered Channels** waveform graph indicators show that the Multiply function and Digital IIR Filter VI processed all channels in the 1D waveform array.
   - Stop the VI when finished.

Process Channels Using a VI that is not Fully Compatible with *N*-Channel Array Data Types

1. Modify the block diagram, as shown in the following figure, to run all channels through a VI that is not fully compatible with the 1D waveform data type.

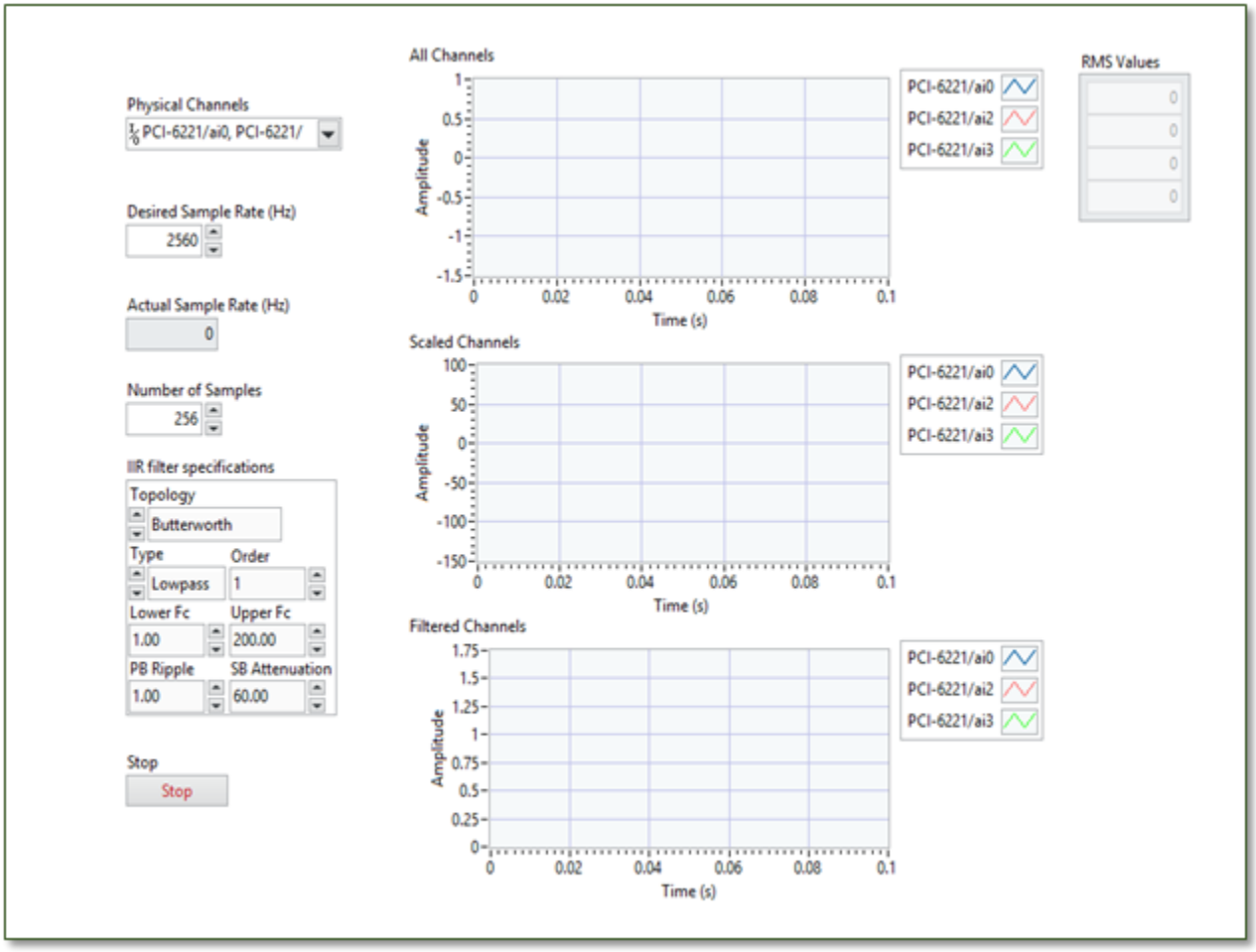


---

1. **For Loop** – Add a For Loop to the block diagram.
2. **RMS VI** – Right-click in the **block diagram**, click **Search** from the Functions palette, type `RMS.vi`. Drag the first appeared result into the while loop. The input of this VI isn't fully compatible with 1D Waveform data type, because if you wire the waveform directly to the input of this VI, the output will show the RMS value for only one channel, instead of four used in this exercise.
3. Wire the **data** output of DAQmx Read VI to the left border of the For Loop and into the RMS VI input. Notice the indexing input tunnel. On the first iteration of the For Loop, this indexing input tunnel will return the first element in the array. On the second iteration of the For Loop, this input tunnel will output the second element in the array, and so on.
4. Wire the **rms value** output of the RMS VI to the right border of the For Loop. This creates an indexing output tunnel. When the For Loop finishes executing, the output of this indexing output tunnel will be a 1D array with its elements containing what the indexing output tunnel received in each For Loop iteration.
5. Right-click the **indexing output tunnel** and select **Create Indicator**.

Note: This For Loop code works for any *N*-channel array, whether it contains 2 channels or 16 channels.

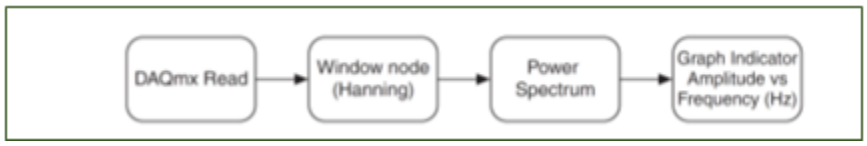
2. Arrange your front panel similar to the following figure.



3. Examine the behavior of the VI.
   - Run the VI.
   - Notice that the **RMS values** 1D DBL array indicator shows the RMS value of each channel, which means that the For Loop processed all channels in the 1D waveform array.
   - Stop the VI when finished.
4. Save the VI.

Your Turn

The following pseudocode shows how to take one channel of data and process it to display the data in the frequency domain (frequency vs. amplitude graph).



1. Create a copy of the VI in this exercise.
2. Modify the VI to perform the above algorithm on all channels.

Answer

The answer is located in the `C:\Solutions\LabVIEW Core 1\Exercise 8-7` directory.

End of Exercise 8-7