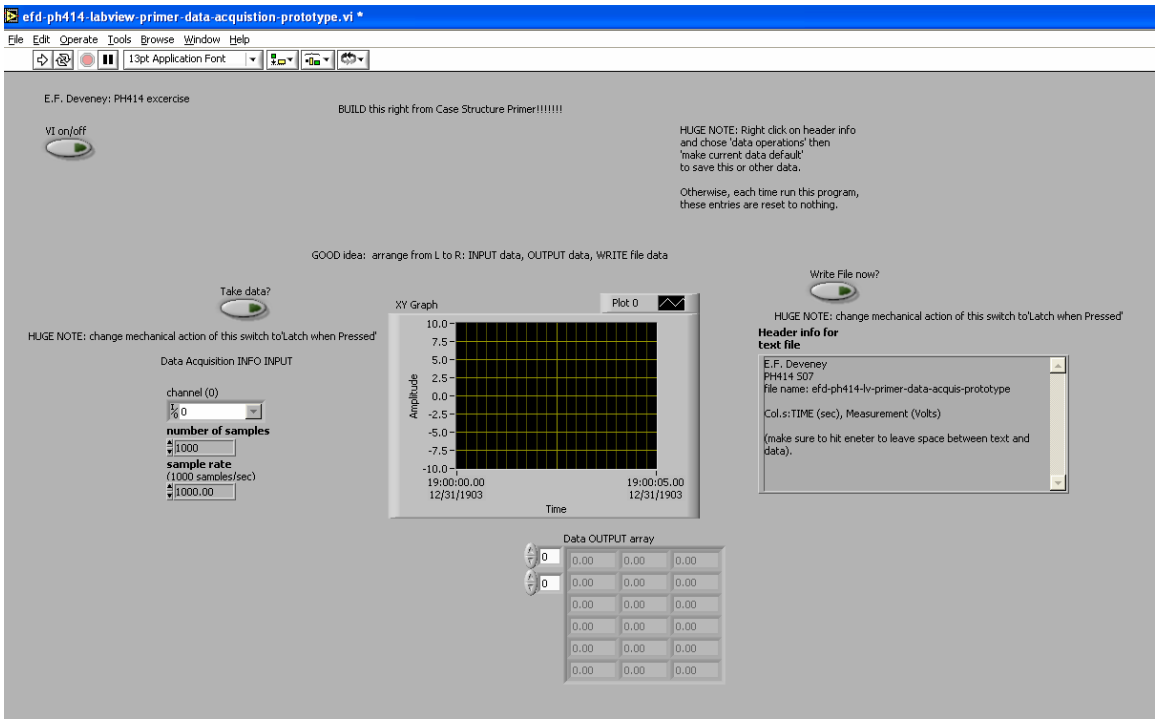
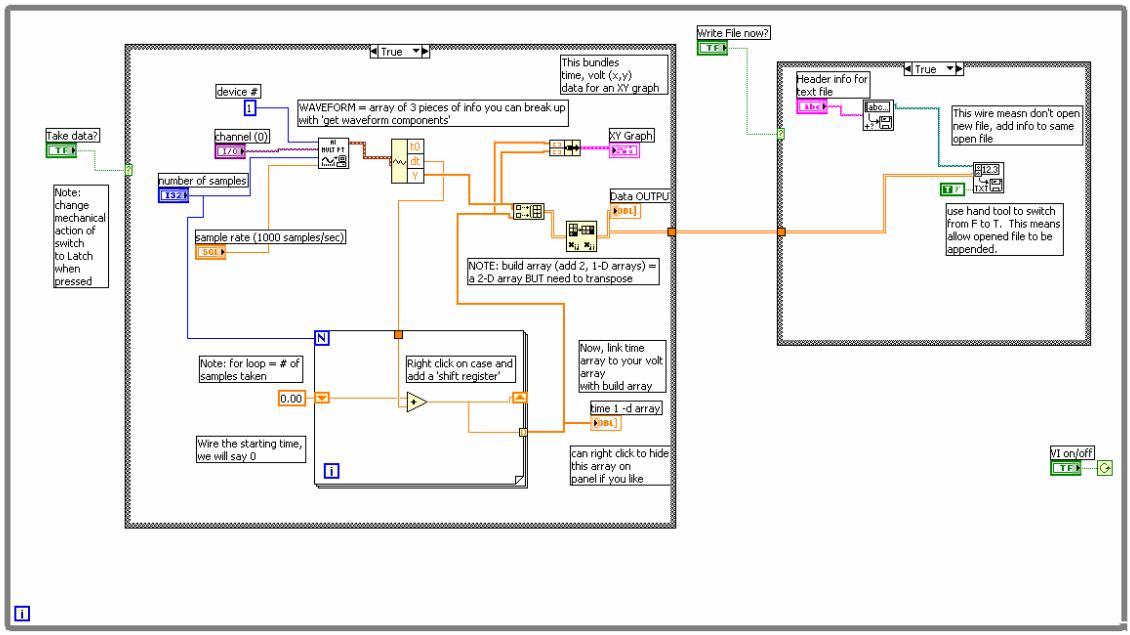
E.F. Deveney:
PH414 Experimental.
RLC Filters Experiment Data Acquisition LabVIEW Code

We'll start with our general purpose data acquisition program that we used for the simple pendulum experiments (note: we also built our Fourier lab code from this same prototype).





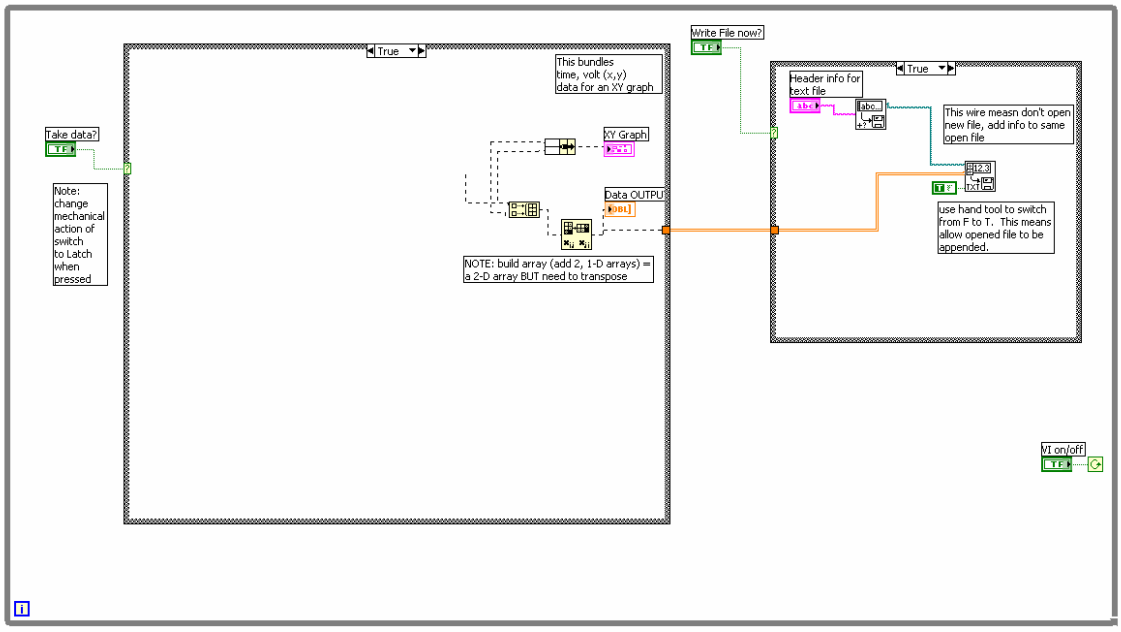
The idea now is to measure two sinusoidal signals $(V_{in}$ & $V_{out})$, pull out info from each such as amplitudes, frequencies and phases and then perform computations such as $\frac{V_{out}}{V_{in}}$ (note, up to now we have only been measuring one signal at a time). The way to measure two signals a time in LabVIEW is to use a different analog input (AI). Instead of

(acquire multiple points, one waveform – note one squiggly line) we will use

(acquire multiple points from waveform**S** – note several squiggly lines).

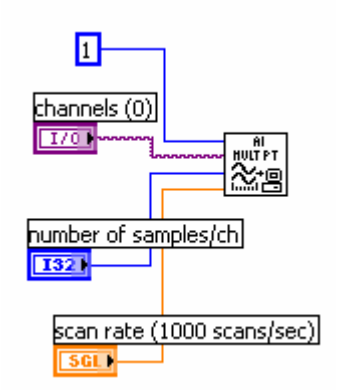
Important to realize how LabVIEW does take multiple signal measurements: for a given sample rate, say 100 Hz, instead of taking 100 samples per second from channel 1, it multitasks between the two channels so it takes one sample from channel 1, then 1 from channel 2 and back and forth between the two channels. The upshot is that the actual sample rate for each channel is ½ of the measurement rate. You must keep this in mind as you decide on the sample rate and ultimately how you interpret your data.

OK, we will still need to make a plot and then save and write data to a text file to be used in a spreadsheet program like EXCEL. We will not need the for-loop. So, the diagram after removing some things, looks as follows:



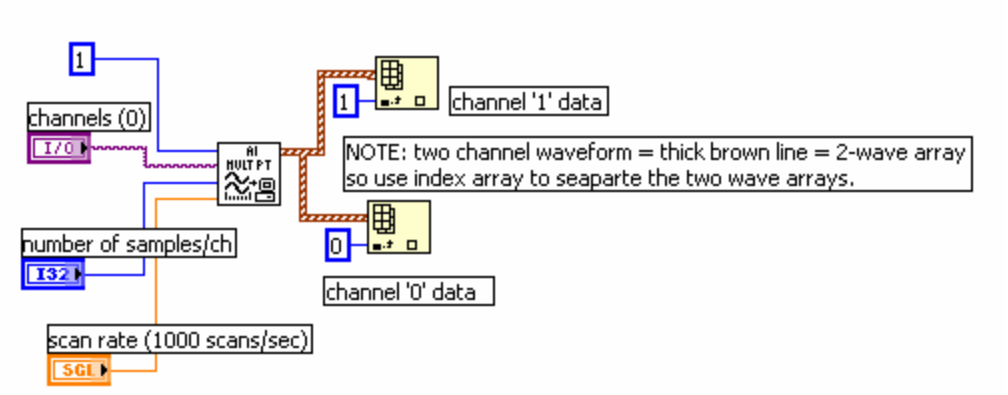
Remember that to erase controls and indicators, you must delete them from the front panel.

Now add the acquire waveforms AI from the ‘data acquisition\analog input’ function. With the wiring tool, create a constant for the device number, and then controls for the channels, number of samples per channel and sample rate. Note that channel is now channel**S** and number of samples is per channel.



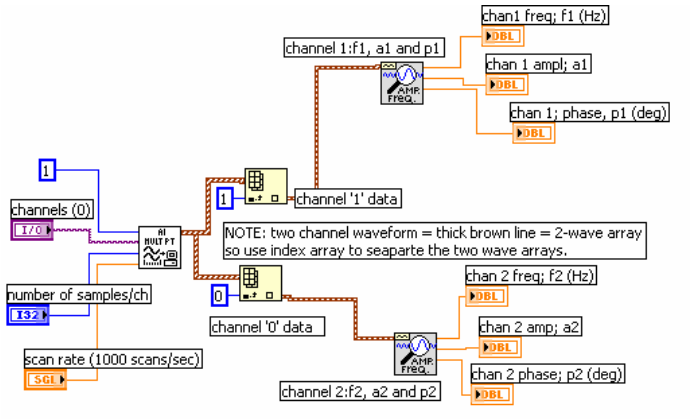
On the front panel, channels control looks like this  to take measurements from channels 0 and 1.

The next step is to separate the data from each channel. This is accomplished as follows:



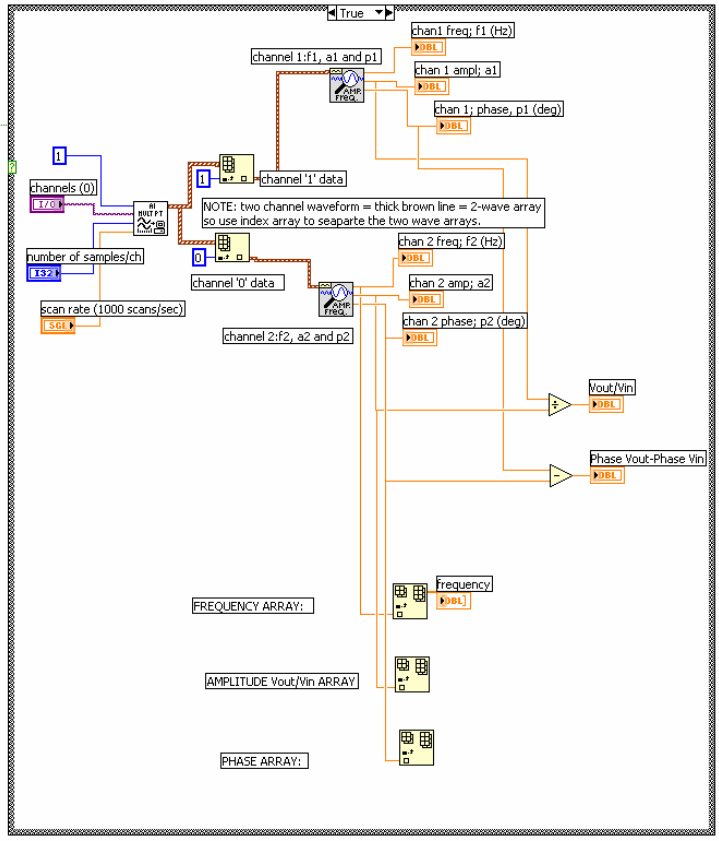
The outputs from the index arrays are just the individual waveforms from each channel.

We can now use LabVIEW’s waveform analyzer (measurements) tools to get waveform info, particulary the amplitude, frequency and phase (I’ve wired indicators to these outputs):



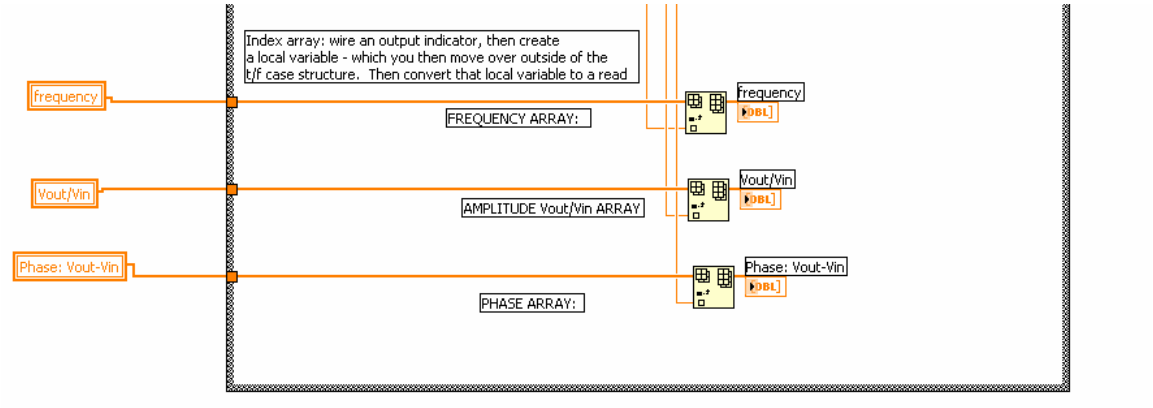
Now we can look at the actual data we are interested in, $\frac{V_{out}}{V_{in}}$ and the relative phase (p1-p2) (the two frequencies should be the same… ie the driving or reference frequency).

Next: Add three 'index arrays' (one each for the frequency, voltage ratio and phase difference data) and wire to each the appropriate data into the 'new element/sub array' input. The idea is that we will be making arrays - lists of the frequency, voltage ratio and phase difference – one point at a time. Each time we hit the get data button, LabView will collect this data for the a given frequency we control. The index array will add the data for the current frequency to the next frequency and so on.

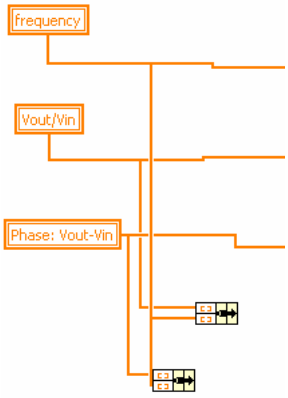


The way we get LabView to add the data for each new frequency is to make what amounts to a shift register for the case (true/false) we have. We used shifts registers to keep adding data from a 'for loop' earlier.

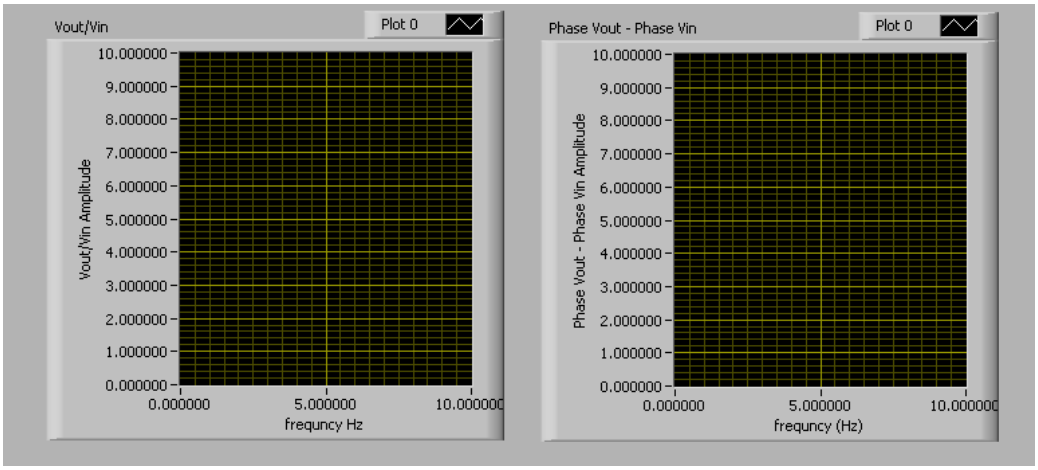
Here is how. Right click the output of an array and create an indicator. Then right click, click 'create' the 'local variable'. This is essentially a copy of the array. Now move this copy outside of the loop, right click and select 'change to read', and then wire this array back into the index array inside of the loop. Do this for each of the index arrays.



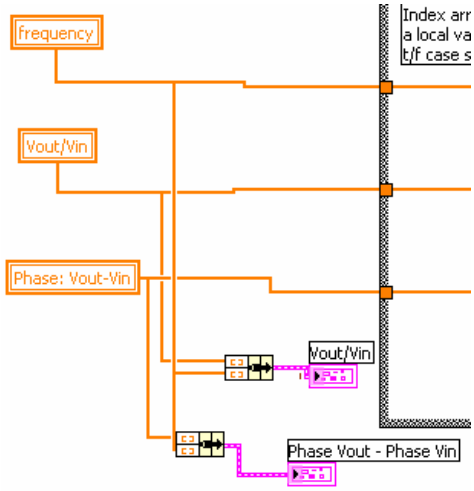
Now we will bundle together $\frac{V_{out}}{V_{in}}$ vs frequency for one plot and (Phase Vout – Phase Vin) for a second plot.



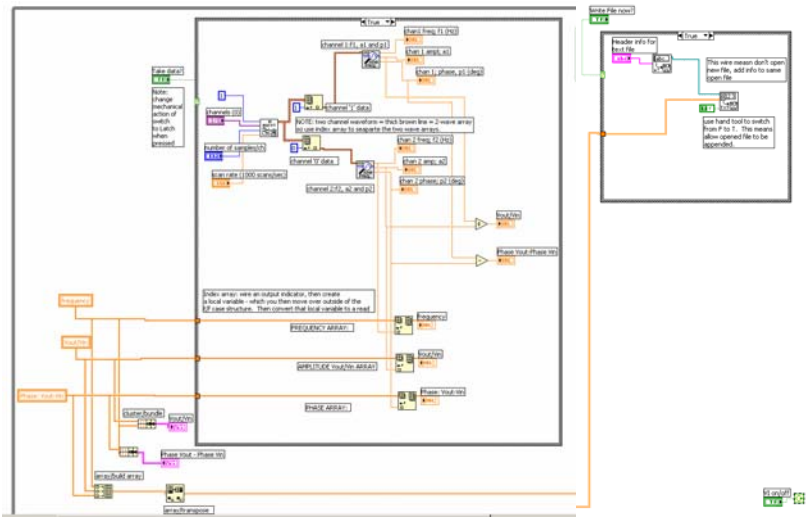
Go to the front panel and create two xy plots. Use the text tool to change the labels of the axis and then right click on each axis, select format – decimal, and then number of significant figures.



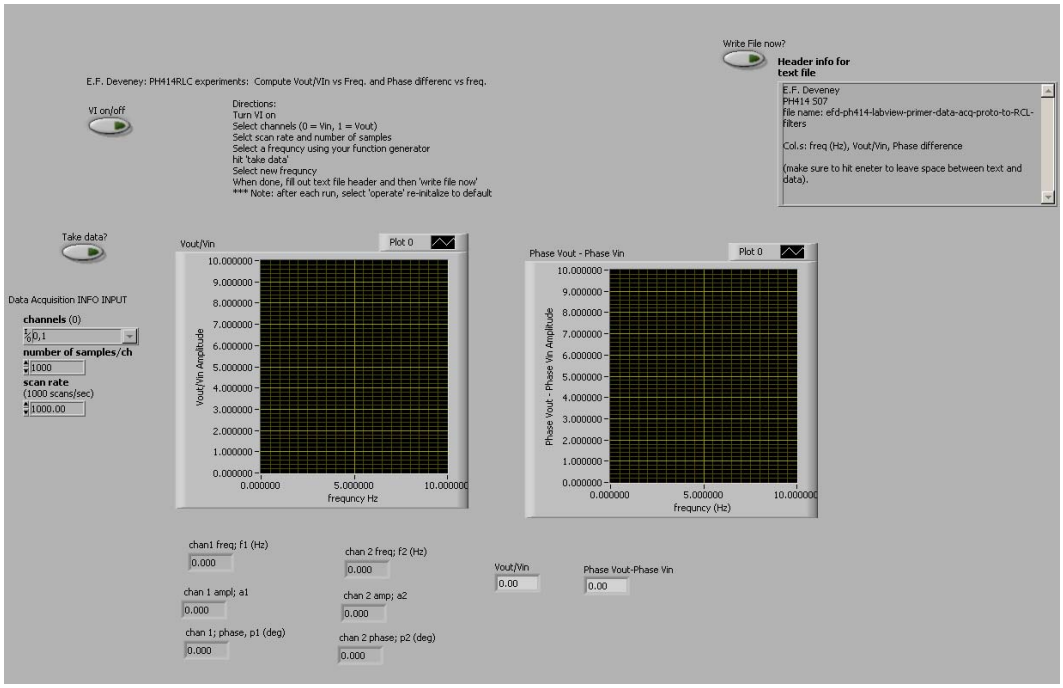
Wire these to our bundled arrays:



Next we have to take this data and make a ASCII text file = most generic form of file that can be read by just about anything. We will build a new array with all pieces of information (frequency, Vout/Vin, and phase difference) then transpose it (so EXCEL likes it – puts the arrays of data in columns instead of rows) and send this to the 'write file' part of our program.



The last things you have to do are:
1. Clean up unwanted things: You can do this by looking or if you double click the broken go arrow, 
And then click on each error you can see what needs to be done.
2. You will also have to clean up the 'False' case of the case structure. Select False and then just erase everything inside. Since the case is not wired to the write directly anymore, you do not need to define default values.
3. Then just arrange the front panel so that it is neat and makes good sense.

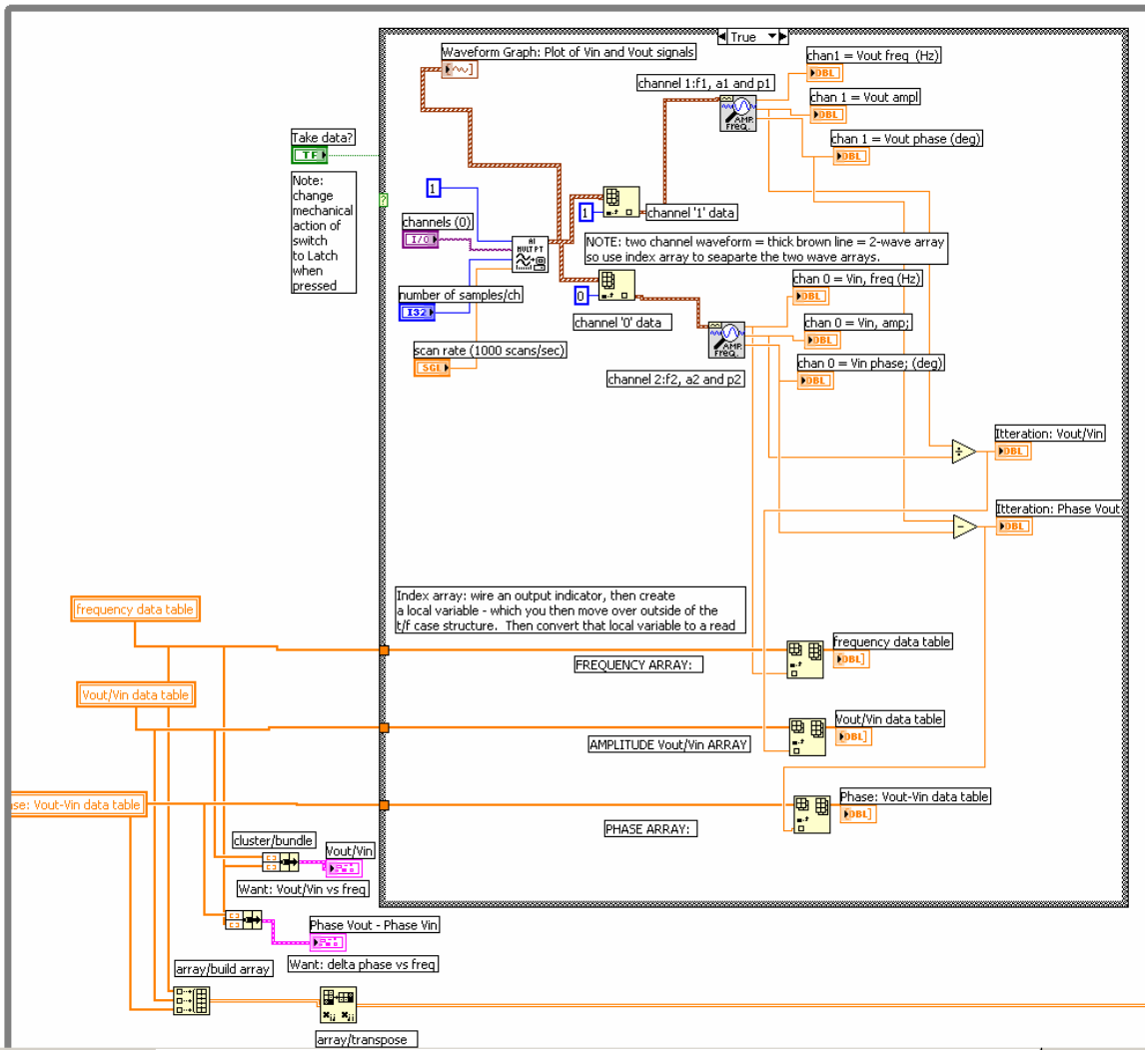


NOTE: good idea to put in 'Directions"
CORRECTIONS = see next page

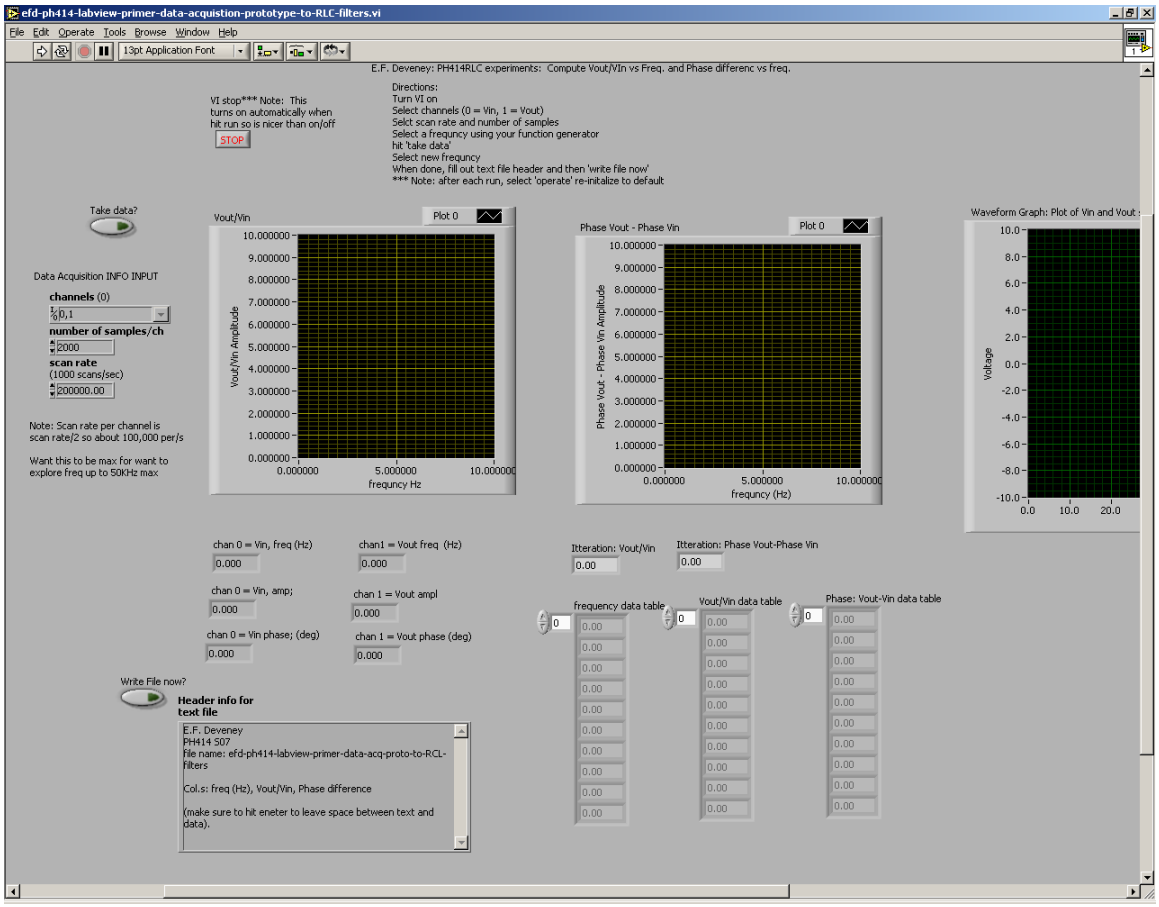
CORRECTIONS:

Oops… here are a couple of wiring corrections plus some ideas on how to spruce up the front panel – to better identify and keep track of Vout and Vin – plus added a plot of the actual scan of Vout and Vin together so you can see the signals and compare amplitudes and phases.

Note: added Waveform graph right after the AI MULT Pt vi
Note: rewired data for 'itteration Vout/Vin' to idex for Vout/Vin data table (same for phase). This was wrong before.
Note: Just made data channel names more clear, Vout and Vin



Here is how the cleaned up front panel looks:



Note the Waveform Graph has been added to look at the time domain scans for Vin and Vout