Ultrasonics Spectrometer Code Manual

Matthew Rothfuss[*]

Department of Animal Science and Food Industry, Kansas State University

May 12, 2015

[*] mrengr@ksu.edu, mrengr@phys.ksu.edu

Contents

List of Figures

List of Tables

Chapter 1

LabView Basics

1.1 Introduction

LabVIEW (short for **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) is a development environment for visual programming, developed by National Instruments (www.ni.com). The code files (or program files) are identified by the **.vi** extension called **Virtual Instruments** or **VIs** for short. This graphical language is most commonly used for data acquisition, instrument control, signal processing (analysis), industrial automation, and more.

The next section will cover some basics of LabVIEW design and operation. For additional resources, the current (2013) LabVIEW Getting Started Manual is located here.

1.1.1 Additional Resources

[1]

Chapter 2

Theory of Operation

Define the background concepts of how/what this program is accomplishing. Make refs to papers but don't do the math here (don't have time for that). Just outline the basics of what we want to do, what goes into the system, what the system does (ref manuals and such for theory & papers), and what the system outputs.

Chapter 3

Code Structure

Theory of code operation goes here. ie case structure, state machine,

3.1 Main VI

Define the outline of the Main VI (the main program) and hit on each part of it. Don't spend time explaining the subvi's here since i'm doing that in the **Custom VI's** section. Make sure to to be thorough on all the code that is not included in the subvi section.

The main program **ASUDS_v13.vi** is contained within a Project file called (file name here). When the program is ran it will first load the auto generated system settings files (via the LC931C_Read.vi [**??**] or DPR300_Read.vi) [3.2]) from the last time the program ran. A set of default system settings files are included for first time use. Next the Oscilloscope and the JSR are both initialized via LC931C_Int.vi and DPR300_Int.vi [3.1, 3.2] with the loaded settings as-well-as set the front panel controls to the loaded settings. At the same time any leftover front panel controls are set to their default values and all block diagram cases are set to there initial positions.

Once the settings clusters have been set properly and

3.2 Custom VI's

List of custom VI's and a short description of what they do. In the next section we will take a deeper look into each of these subvi's.

These files load in most of the front panel controls, some are missing since more code has been added but the vi that saves the system settings has not been updated to include the new additions.

3.2.1 Oscilloscope VIs

Oscilloscope		
VI	File Name	Description
	LC9310C_Read.vi	Load Oscilloscope Setting from System Generated File
	LC9310C_Int.vi	Initialize Oscilloscope Settings
	LC9310C_settings.vi	Apply Settings to Oscilloscope
	LC9310C_single-wave-output.vi	Acquire Single Wave from Oscilloscope and Average
	LC9310C_norm-pad-hilbert.vi	Oscilloscope Tab Settings
	LC9310C-Config-Write-Close.vi	Write Oscilloscope settings to System File and close Oscilloscope resources

Table 3.1: Oscilloscope Custom VIs

3.2.1.1 LC9310C_Read.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the **error (in)** input then the code in the green box does not execute and the **LeCroy 9310C Settings** cluster outputs a set of default values.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectively. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence when integer counter =0. =1, =2, =3, then data is outputted).

Back to Osc. Table 3.1

3.2.1.2 LC9310C_Int.vi

The LC931C_Int.vi initializes the LeCroy 9310C Oscilloscope settings: *trigger setup, vertical setup, and time setup (horizontal)*. This VI requires the LeCroy lc930x1x Library. Figure (3.2) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute, instead the case not shown passes through the **LeCroy 9310C Settings** unchanged.

The **LCDSO Initialize** VI takes the GPIB location of the oscilloscope and creates a duped VISA session for the instrument. Next the **LCDSO Revision** reads the instruments Driver, Firmware, and Options Installed. All are put into a cluster called **Oscilloscope Info**, but first we write to the VISA session **IND?* an identification command. Next read the response with a count of 100 (the length of our expected response), this will give an identification string that is placed into the **Oscilloscope Info** cluster.

From the **LeCroy 9310C Settings** in cluster we set the **LC930x TimeBase**, **LC930x Vertical Setup**, and **LC930x Trigger** VIs. The cluster is unbundled and each setting is routed to it's respective input. Note the **LeCroy 9310C Settings** cluster is just passed through in this VI and not altered, because it is only pulling out data to setup other VIs.

Back to Osc. Table 3.1

3.2.1.3 LC9310C_Settings.vi

The LC931C_Settings.vi applies changes to the *trigger setup, vertical setup, and time setup (horizontal)* oscilloscope VIs. This VI requires the LeCroy lc930x1x Library. Figure (3.3) is the block diagram. On the "error" case the **LeCroy 9310C Settings** cluster is passed through and so is the error cluster.

Back to Osc. Table 3.1

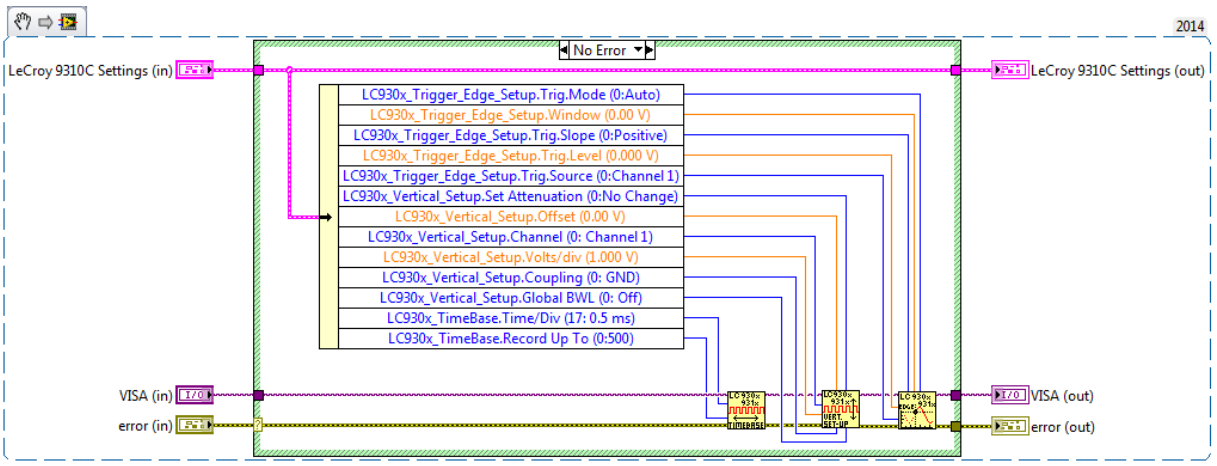

2014
No Error
LeCroy 9310C Settings (in)
LeCroy 9310C Settings (out)
LC930x_Trigger_Edge_Setup.Trig.Mode (0:Auto)
LC930x_Trigger_Edge_Setup.Window (0.00 V)
LC930x_Trigger_Edge_Setup.Trig.Slope (0:Positive)
LC930x_Trigger_Edge_Setup.Trig.Level (0.000 V)
LC930x_Trigger_Edge_Setup.Trig.Source (0:Channel 1)
LC930x_Vertical_Setup.Set Attenuation (0:No Change)
LC930x_Vertical_Setup.Offset (0.00 V)
LC930x_Vertical_Setup.Channel (0: Channel 1)
LC930x_Vertical_Setup.Volts/div (1.000 V)
LC930x_Vertical_Setup.Coupling (0: GND)
LC930x_Vertical_Setup.Global BWL (0: Off)
LC930x_TimeBase.Time/Div (17: 0.5 ms)
LC930x_TimeBase.Record Up To (0:500)
VISA (in)
error (in)
VISA (out)
error (out)
TIMEBASE
VERT. SET-UP

Figure 3.3: LC9310C_Settings.vi

3.2.1.4 LC9310C_Single-Wave-Output.vi

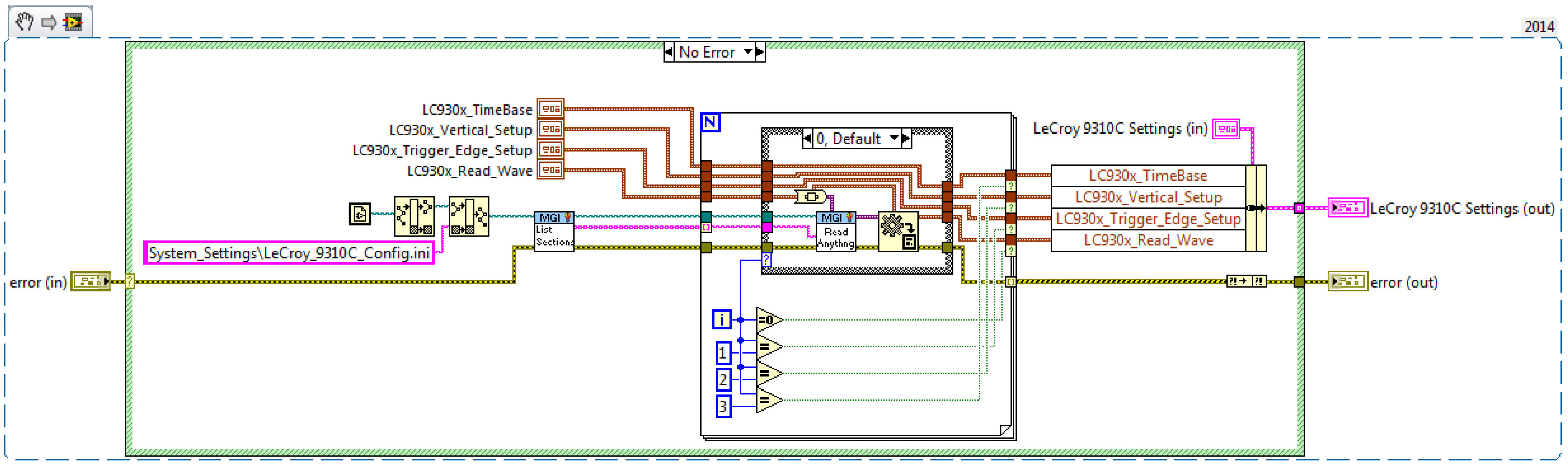


Figure 3.1: LC9310C_Read.vi

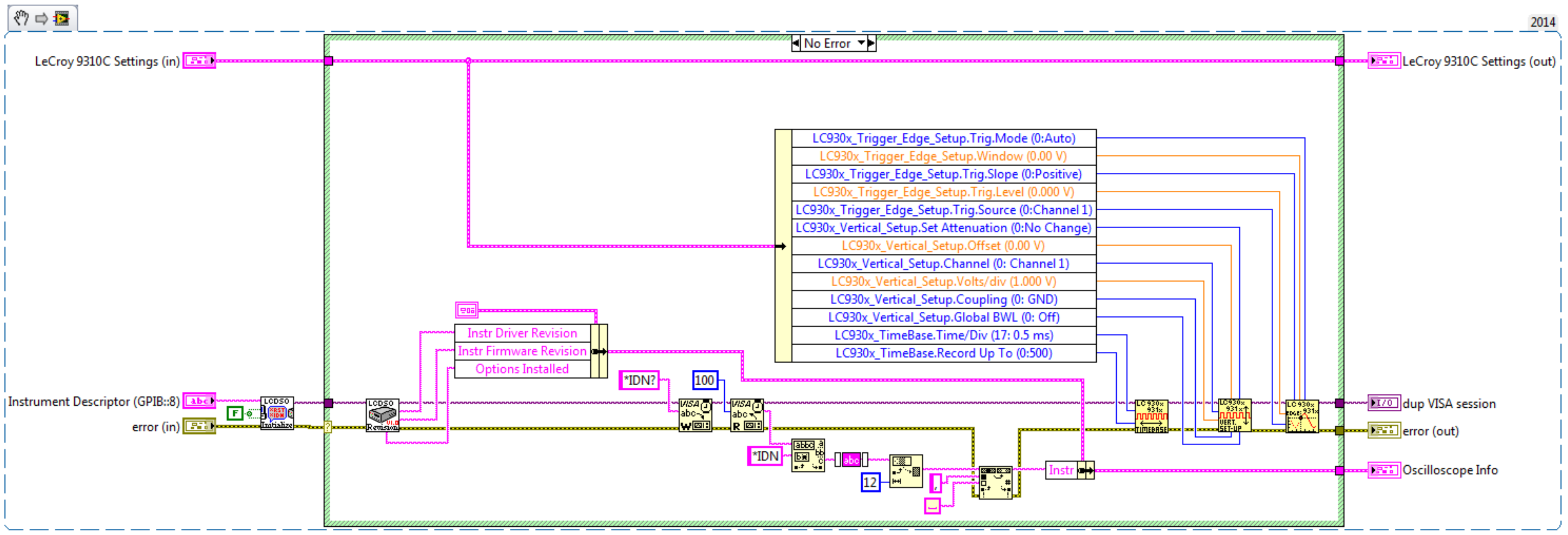
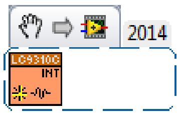
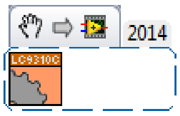

2014
No Error
LeCroy 9310C Settings (in)
LeCroy 9310C Settings (out)
LC930x_Trigger_Edge_Setup.Trig.Mode (0:Auto)
LC930x_Trigger_Edge_Setup.Window (0.00 V)
LC930x_Trigger_Edge_Setup.Trig.Slope (0:Positive)
LC930x_Trigger_Edge_Setup.Trig.Level (0.000 V)
LC930x_Trigger_Edge_Setup.Trig.Source (0:Channel 1)
LC930x_Vertical_Setup.Set Attenuation (0:No Change)
LC930x_Vertical_Setup.Offset (0.00 V)
LC930x_Vertical_Setup.Channel (0: Channel 1)
LC930x_Vertical_Setup.Volts/div (1.000 V)
LC930x_Vertical_Setup.Coupling (0: GND)
LC930x_Vertical_Setup.Global BWL (0: Off)
LC930x_TimeBase.Time/Div (17: 0.5 ms)
LC930x_TimeBase.Record Up To (0:500)
Instr Driver Revision
Instr Firmware Revision
Options Installed
Instrument Descriptor (GPIB::8)
error (in)
F
*IDN?
100
*IDN
12
Instr
dup VISA session
error (out)
Oscilloscope Info

Figure 3.2: LC9310C_Int.vi



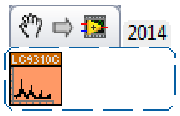


The LC931C_Single-Wave-Output.vi acquires the signal from the LeCroy 9310C Oscilloscope and will average the acquired wave according to the desired sample size. This VI requires the LeCroy lc930x1x Library. Figure (3.2) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute, instead the case not shown passes through the **LeCroy 9310C Settings**, **VISA (in)**, and **error (in)** unchanged. The **Wave Output** cluster will reset to an empty Waveform cluster during an error.

We first initialize an empty array with the size of our acquired wave from the **LCDSO Simple Read** VI. Take the Source channel setting from the **LeCroy 9310C Settings** cluster and set both **LCDSO Simple Read** VIs. The for loop is setup to first add all the arrays together (Figure 3.4b) and then once the for loop indices indicator and the number of times the loop has ran match, then the added array is divided by total number of samples and placed back into the **Wave Output** cluster. (Figure 3.4a) If no averaging is needed because the sample size is 1, meaning average over one sample (need 2 to average), then the for loop will run once and the **Wave Output** cluster will be passed out. (Figure 3.4c)

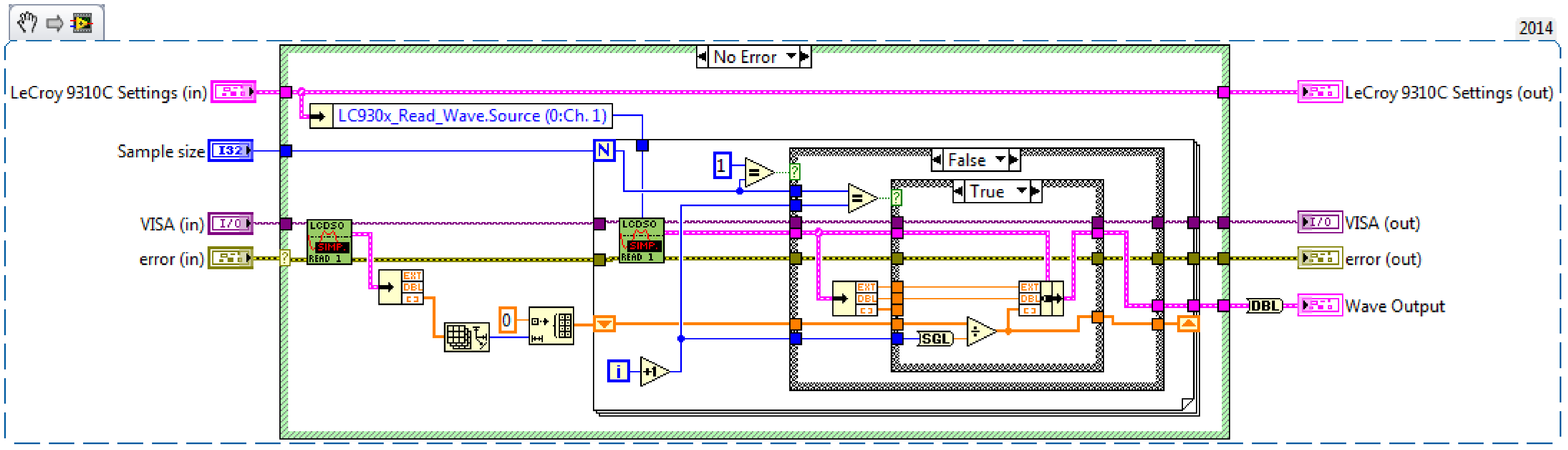
Back to Osc. Table 3.1

3.2.1.5 LC9310C_Norm-Pad-Hilbert.vi

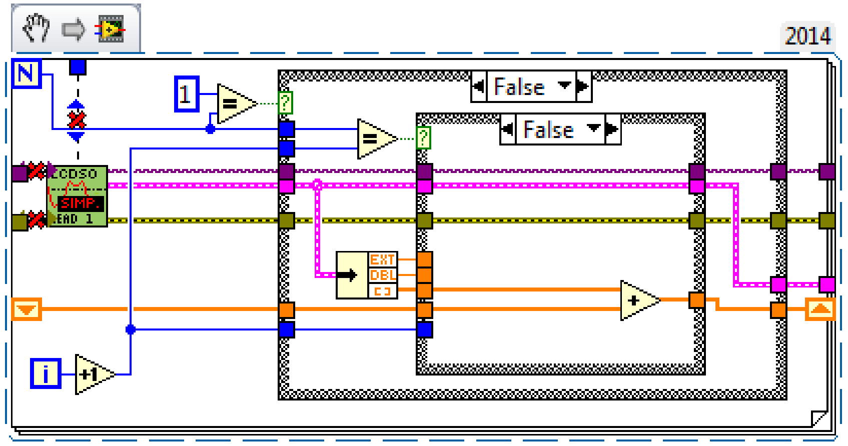


The LC931C_Norm-Pad-Hilbert.vi will Pad the Y-Array with Zeros on both sides, Normalize the Y-Array, and apply a Fast Hilbert Transform on the waveform cluster. This VI requires the **U-Sonic Hilbert Transform** VI. Figure (3.5) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute, instead the case not shown passes through the **LeCroy 9310C Settings**, **VISA (in)**, **Wave Input**, and **error (in)** unchanged.

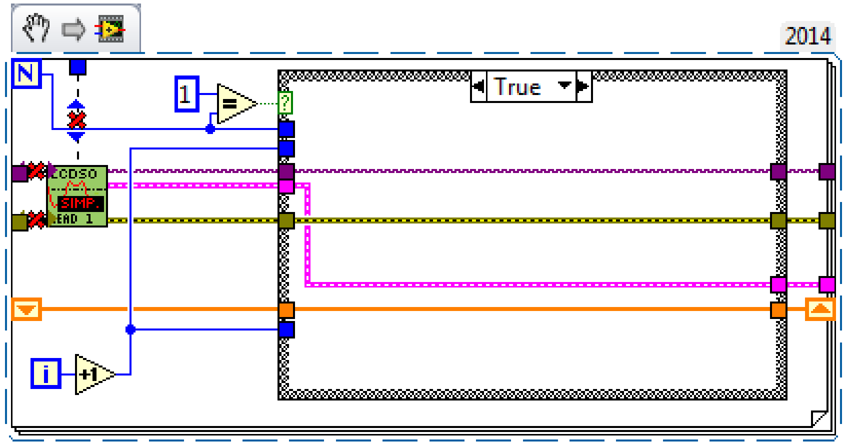
The far left case applies the option to pad both sides of the Y-Array with zeros. This is done by pulling out the array from the **Wave Input** cluster, reversing it and applying the Pad with zeros subVI. Then reverse the array again to apply the zero pad on the other end of the array. The padded array is then rebundled into the **Wave Input** cluster.



(a) Overall Code with Outer Case False & Inner Case True



(b) Both Cases False



(c) Outer Case True

Figure 3.4: LC9310C_single-wave-output.vi

The middle case was meant to normalize the Y-Array, but the way it is currently setup is incorrect. There are different ways to normalize a signal and each have a slightly different reason for doing so. If normalization is needed then remove the Mean VI and just pass through the normalized signal.

The far right case applies a Fast Hilbert Transformation to the **Wave Input** cluster. This VI is based from the Ultrasonic Non-destructive Testing Starter Kit and was modified to accompany our code. This is used to help view the oscilloscope signal but is not used it the process of data acquisition.

Back to Osc. Table 3.1

3.2.1.6 LC9310C-Config-Write-Close.vi

The LC9310C-Config-Write-Close.vi closes (releases) the oscilloscope VISA session and writes the **LeCroy 9310C Settings** cluster to the system settings file (located in the folder (**System_Settings**) which is in the root directory of the main VI) to be loaded by the LC9310C_Read.vi the next time the program is ran. This VI requires the MGI Library. Figure (3.7) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The left case structure closes the oscilloscope VISA session. The right case structure pulls out the clusters *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)* and places then in the system settings configuration file *LeCroy_9310C_Config.ini*. By default both of these case structures are set to True.

The **MGI Write Anything** VI takes the information from each cluster and places in to a section, named by the four strings built into an array, located at the file path input.

Note: It is important that the VISA session be closed and not left open. If left open, then the next time the program is ran the oscilloscope VIs may return an error. This happens because LabVIEW still thinks a device is connected to that VISA session, but it's not because VISA sessions are tied to the process ID of the LabVIEW program. Meaning each time a program is ran, it runs under a different ID so the VISA sessions don't overlap.

Back to Osc. Table 3.1

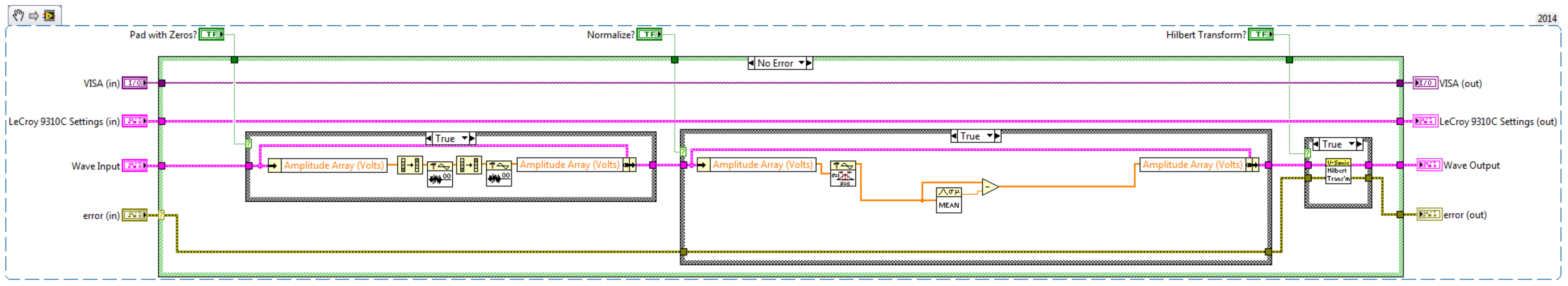


Figure 3.5: LC9310C_Norm-Pad-Hilbert.vi

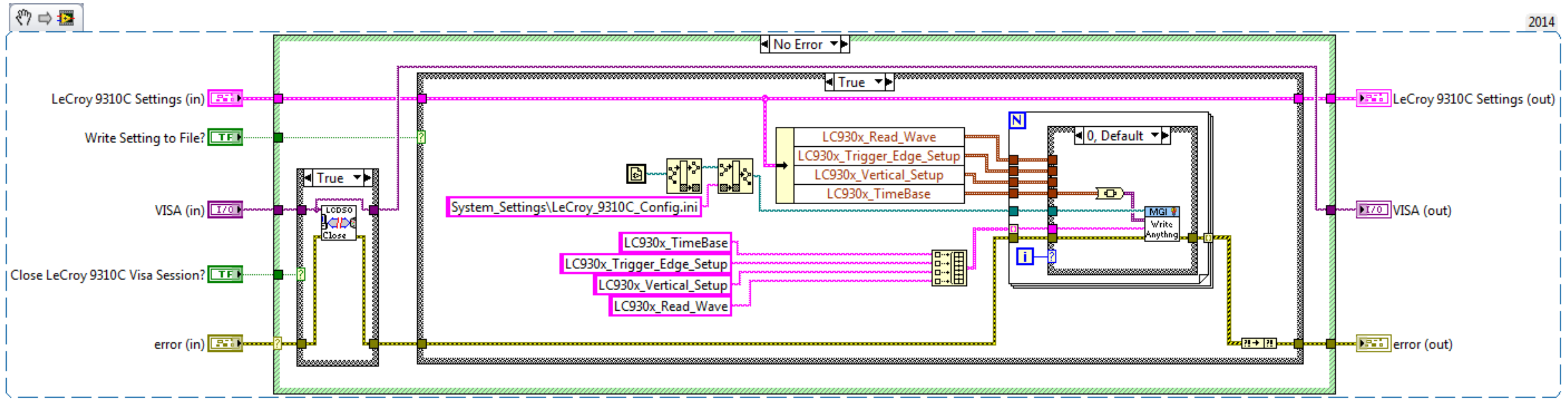


Figure 3.6: LC9310C-Config-Write-Close.vi

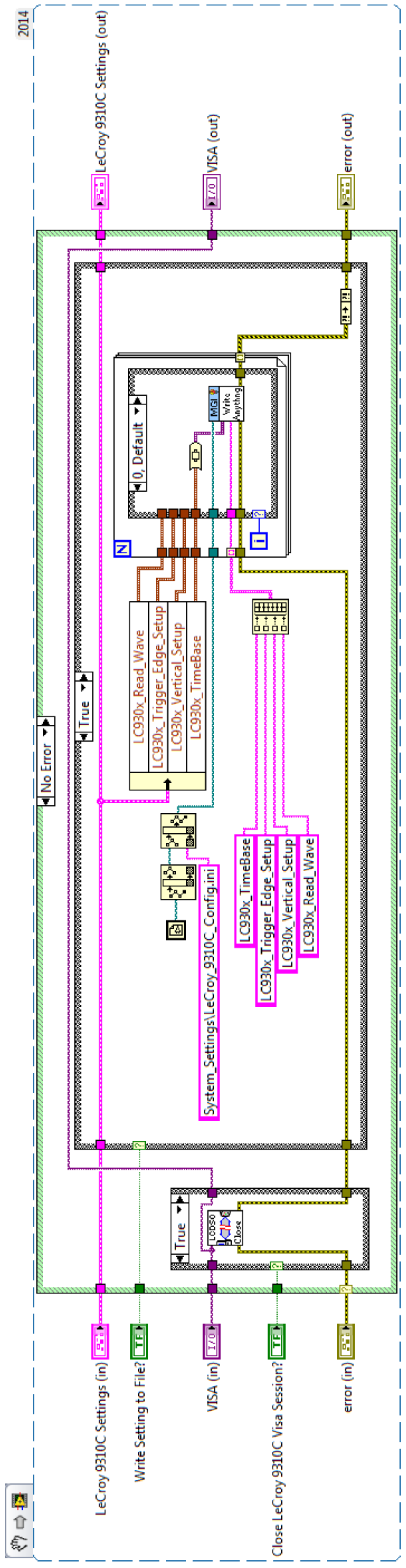


Figure 3.7: LC9310C-Config-Write-Close.vi

3.2.2 JSR Pulser/Receiver VIs

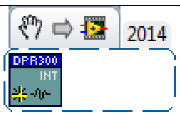
JSR Pulser/Receiver		
VI	File Name	Description
	DPR300_Read.vi	Load Pulser/Receiver Setting from System Generated File
	DPR300_Int.vi	Initialize Pulser/Receiver Settings
	DPR300_settings.vi	Apply Settings to Pulser/Receiver
	DPR300-Config-Write-Close.vi	Write Pulser/Receiver settings to System File and close Pulser/Receiver resources

Table 3.2: JSR Pulser/Receiver Custom VI's

3.2.2.1 DPR300_Read.vi

The DPR300_Read.vi reads the JSR DPR300 Pulser/Receiver settings from file and loads the values into a **DPR300 Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.8) is the block diagram. It is setup as an error case structure. When an error is detected from the **error (in)** input then the code in the green box does not execute and the **DPR300 Settings** cluster outputs a set of default values.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(DPR300_Int_Adress, DPR300_PRF_Settings, DPR300_Generated_Pulse, DPR300_Received)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectively. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence when integer counter =0. =1, =2, =3, then data is outputted).

Back to JSR Table 3.2

3.2.2.2 DPR300_Int.vi

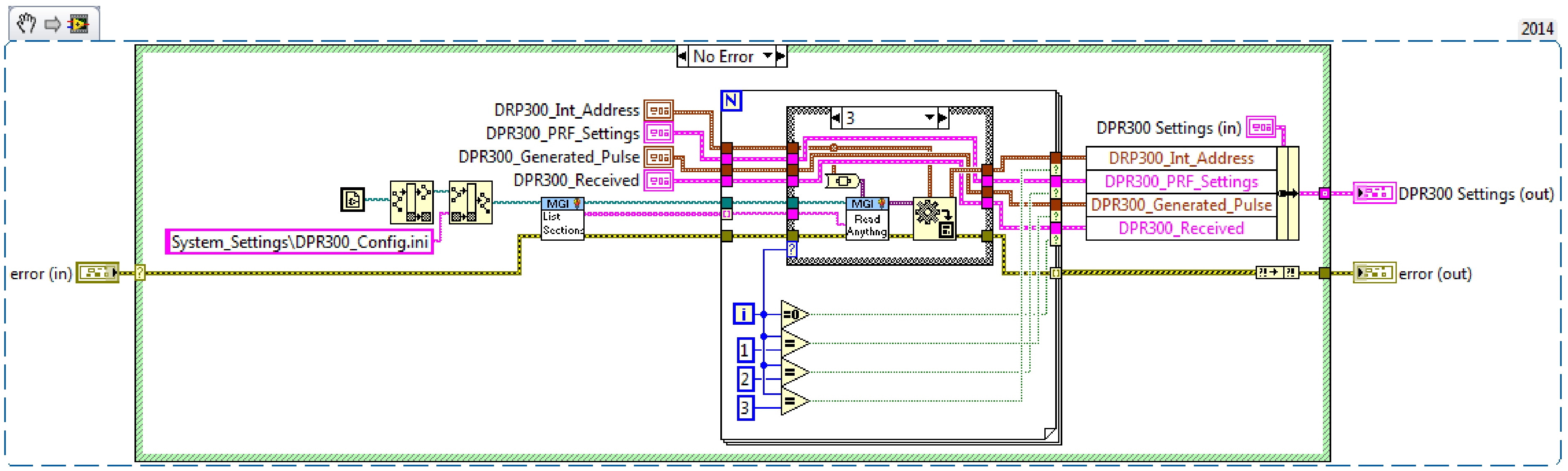


Figure 3.8: DPR300_Read.vi

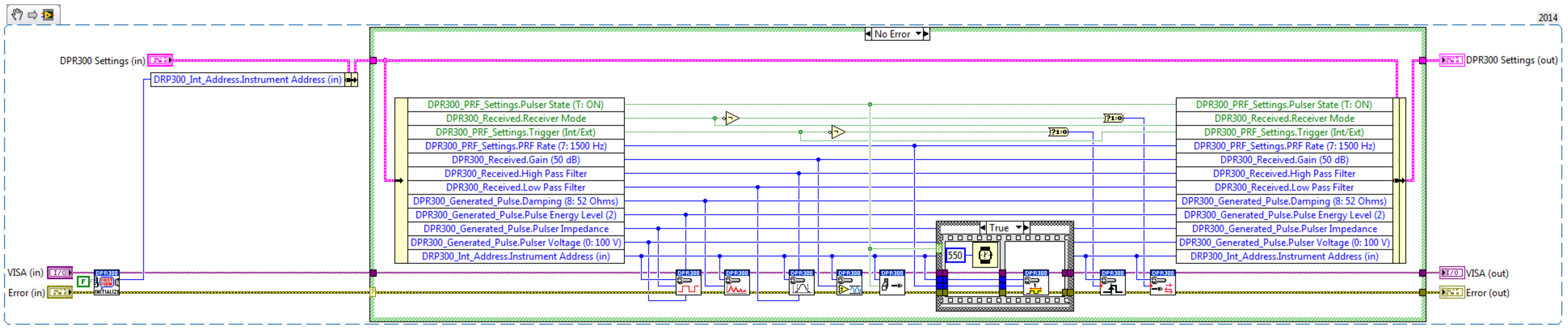

2014
No Error
DPR300 Settings (in)
DRP300_Int_Address.Instrument Address (in)
DPR300_PRF_Settings.Pulser State (T: ON)
DPR300_Received.Receiver Mode
DPR300_PRF_Settings.Trigger (Int/Ext)
DPR300_PRF_Settings.PRF Rate (7: 1500 Hz)
DPR300_Received.Gain (50 dB)
DPR300_Received.High Pass Filter
DPR300_Received.Low Pass Filter
DPR300_Generated_Pulse.Damping (8: 52 Ohms)
DPR300_Generated_Pulse.Pulse Energy Level (2)
DPR300_Generated_Pulse.Pulser Impedance
DPR300_Generated_Pulse.Pulser Voltage (0: 100 V)
DRP300_Int_Address.Instrument Address (in)
True
550
VISA (in)
Error (in)
DPR300 Settings (out)
VISA (out)
Error (out)

Figure 3.9: DPR300_Int.vi

The DPR300_Int.vi initializes the JSR DPR300 Pulser/Receiver settings: *generated pulse setup, dampening, low pass & high pass filters, gain, pulser state (on/off), pulser repition frequency, trigger (Internal/External), and receiver mode (Internal/External)*. This VI requires the JSR Ultrasonics dpr300 Library. Figure (3.9) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute, instead the case not shown passes through the **DPR300 Settings** and **error (in)** unchanged.

The **DPR300 Initialize** VI prepares JSR DPR300 Pulser/Receiver for use and identifies the instrument address which is then updated in the **DPR300 Settings** cluster. From there each value of the cluster is unbundled and sent to its respective VI. When the *Pulser State* is set to true then a wait time is needed before the rate is applied. The t/f case structure after the pulser state VI will wait when $550ms$ to apply the rate settings when true. During the false case there is no need for a wait time but we still need to update the rate settings in case later the Pulser is used.

Back to JSR Table 3.2

3.2.2.3 DPR300_Settings.vi

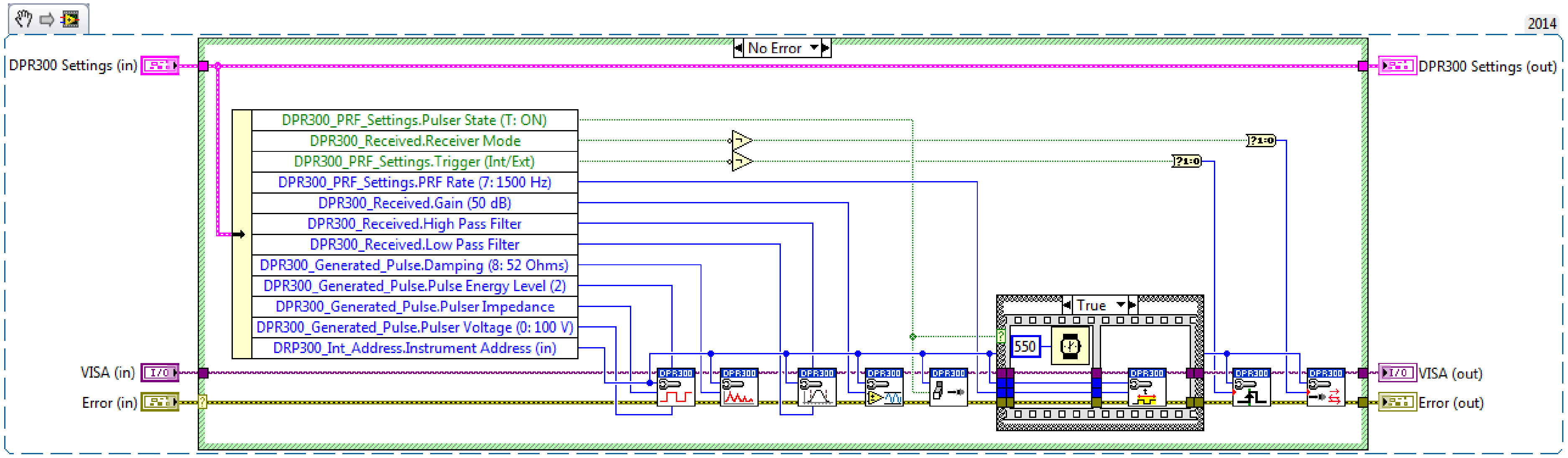
The DPR300_Settings.vi applies changes to the JSR DPR300 Pulser/Receiver settings: *generated pulse setup, dampening, low pass & high pass filters, gain, pulser state (on/off), pulser repition frequency, trigger (Internal/External), and receiver mode (Internal/External)*. This VI requires the JSR Ultrasonics dpr300 Library. Figure (3.10) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute, instead the case not shown passes through the **DPR300 Settings** and **error (in)** unchanged.

Back to JSR Table 3.2

3.2.2.4 DPR300-Config-Write-Close.vi

The DPR300-Config-Write-Close.vi closes (releases) the JSR DPR300 Pulser/Receiver VISA session and writes the **DPR300 Settings** cluster to the system settings file (located in the folder (**System_Settings**) which is in the root directory of the main VI) to be loaded by the DPR300_Read.vi the next time the program is ran. This VI requires the MGI Library. Figure (3.11) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute and **DPR300 Settings** cluster, **VISA (in)**, and *error (in)* are all passed through unchanged.

The left case structure closes the JSR DPR300 Pulser/Receiver VISA session. The right case structure pulls out the clusters *(DPR300_Received, DPR300_Generated_Pulse, DPR300_PRF_Settings,*



2014
No Error
DPR300 Settings (in)
DPR300 Settings (out)
DPR300_PRF_Settings.Pulser State (T: ON)
DPR300_Received.Receiver Mode
DPR300_PRF_Settings.Trigger (Int/Ext)
DPR300_PRF_Settings.PRF Rate (7: 1500 Hz)
DPR300_Received.Gain (50 dB)
DPR300_Received.High Pass Filter
DPR300_Received.Low Pass Filter
DPR300_Generated_Pulse.Damping (8: 52 Ohms)
DPR300_Generated_Pulse.Pulse Energy Level (2)
DPR300_Generated_Pulse.Pulser Impedance
DPR300_Generated_Pulse.Pulser Voltage (0: 100 V)
DRP300_Int_Address.Instrument Address (in)
True
550
VISA (in)
Error (in)
VISA (out)
Error (out)

Figure 3.10: DPR300_Settings.vi

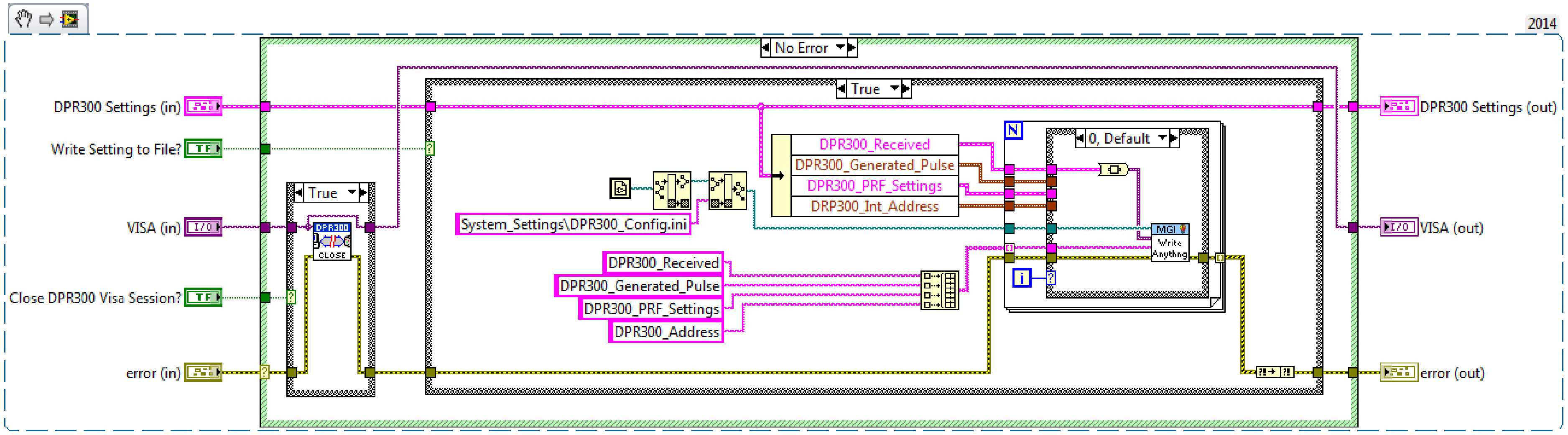


Figure 3.11: DPR300-Config-Write-Close.vi

DPR300_Address) and places then in the system settings configuration file *DPR300_Config.ini*. By default both of these case structures are set to True.

The **MGI Write Anything** VI takes the information from each cluster and places it into a section, named by the four strings built into an array, located at the file path input.

Note: It is important that the VISA session be closed and not left open. If left open, then the next time the program is ran the oscilloscope VIs may return an error. This happens because LabVIEW still thinks a device is connected to that VISA session, but it's not because VISA sessions are tied to the process ID of the LabVIEW program. Meaning each time a program is ran, it runs under a different ID so the VISA sessions don't overlap.

Back to JSR Table 3.2

3.2.3 Ultrasonic Package VIs

Ultrasonic Package		
VI	File Name	Description
	USonic-A-Scan-Config-edit.vi	Configure/Set Gates for Data Acquisition
	USonic-Gates-edit.vi	Pull Out Relevant Data from Gates for Data Acquisition
	USonic-FFT.vi	Process Gate For Quick Analysis

Table 3.3: Ultrasonic A-Scan Customized Package VI's

3.2.3.1 USonic-A-Scan-Config-edit.vi

The USonic-A-Scan-Config-edit.vi reads the **Gate Configuration** cluster and applies it to the **Data In** waveform cluster. It also finds the peak to peak time (time of fight). This VI was edited from the Ultrasonic Non-destructive Testing Starter Kit to fit our needs. The VI outputs the waveform cluster graph with the gates overlaid as-well-as it outputting the paired gate information for data analysis.

Back to USonic Table 3.3

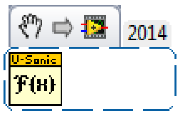
3.2.3.2 USonic-Gates-edit.vi

Pull Out Relevant Data from Gates for Data Acquisition. Don't have blockdiagram to look at right now, so will fill in this section later.

Back to USonic Table 3.3

3.2.3.3 USonic-FFT.vi

The USonic-FFT.vi takes the waveform data array and applies the single sided fast Fourier transform (FFT) which is a complex expression. From the complex expression the Magnitude and Phase are pulled



out and have option associated with them. The VI outputs the FFT, Magnitude of FFT, and Phase of FFT as a waveform graph. Also the peak magnitude, peak phase and corresponding frequency are outputted for data analysis. Figure (3.12) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute and all the outputs are set to zero while the error passes through.

The single sided fast Fourier transform VI takes the **Signal (V)** array and the **dt** double to computer the FFT and cut off one side of the FFT (thus the single sidedness). The output from the FFT is in frequency space so the **dt** now becomes **df** and the array output is now complex ($z = x + iy$). For just the FFT graph we take the complex array and **df** and rebuild the waveform for output. For the magnitude and phase we must convert the expression $z = x + iy$ to $z = r * \exp(i\theta)$ where $Magnitude = r^2$ (note the square, this is actually the power representation) and $phase = \theta$ (in rads). The true/false case structure for the FFT array gives the option to change the scaling of the array to decibel form. The true/false case structure for the phase gives the option to change the scaling to degrees as well as unwraps the phase array by eliminating discontinuities whose absolute values exceed either pi or 180.

Graphs are built as waveform clusters and originate from there respective arrays. For the math portion the processed magnitude array is built into a waveform and passed to the **Waveform Max/Min** VI to find the peak value ($y - axis$) and its corresponding frequency value ($x - axis$). The peak value is outputted as the Peak Magnitude of the Magnitude array and is feed into the **Search Array** VI that searches for the index (position in array) of the magnitude array. Then the index is used to pull out the corresponding element from the phase array, this in turn is the peak magnitude of the phase.

Back to USonic Table 3.3

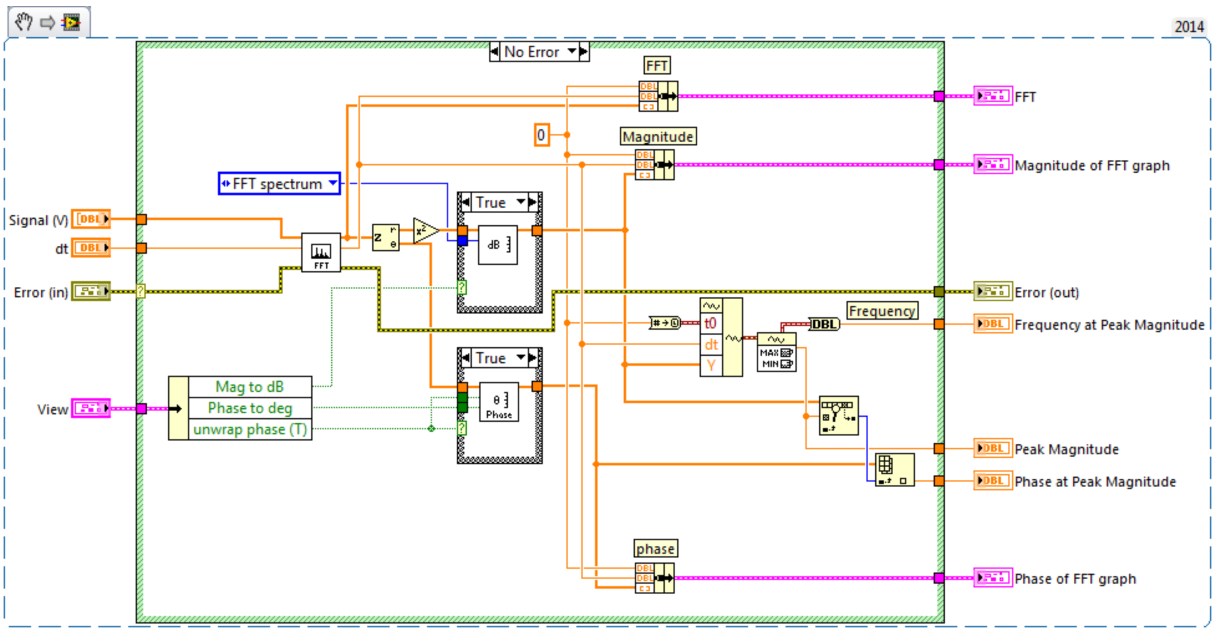


Figure 3.12: USonic-FFT.vi

3.2.4 Math VIs

Math		
VI	File Name	Description
	Waveform-to-XY-Array.vi	Convert Waveform to XY-Array
	Filter_signal.vi	Filter Wave Signal for Oscilloscope Tab (does not affect Data Acquisition)
	Average-Dynamic-Array.vi	Take the Average of N elements in a Dynamic Array
	FWHM-Poly.vi	Compute the Full Width Half Max (FWHM) of either a Waveform, XY-Graph, or Waveform cluster

Table 3.4: Custom Math VI's

3.2.4.1 Waveform-to-XY-Array.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Miscellaneous Table 3.4

3.2.4.2 Filter_signal.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Math Table 3.4

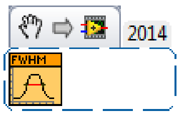
3.2.4.3 Average-Dynamic-Array.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Math Table 3.4

3.2.4.4 FWHM-Poly.vi



The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Math Table 3.4

3.2.5 Miscellaneous VIs

Miscellaneous		
VI	File Name	Description
	Save-User-Config-File.vi	Save all front panel controls to a user.ini settings file
	Load-User-Config-File.vi	Load the user.ini settings file
	Time-Data.vi	Load and Save Data Timing table
	Instrument-error-handler.vi	Pop-up error message for loss of Instrument signal

Table 3.5: Miscellaneous Custom VI's

3.2.5.1 Save-User-Config-File.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Misc. Table 3.5

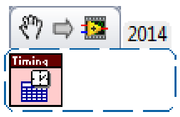
3.2.5.2 Load-User-Config-File.vi

The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Misc. Table 3.5

3.2.5.3 Time-Data.vi

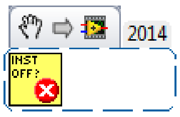


The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Misc. Table 3.5

3.2.5.4 Instrument-Error-Handler.vi



The LC931C_Read.vi reads the LeCroy 9310C oscillosope settings from file and loads the values into a **LeCroy 9310C Settings** cluster. The settings folder (**System_Settings**) is located in the root directory of the main VI. This VI requires the MGI Library. Figure (3.1) is the block diagram. It is setup as an error case structure. When an error is detected from the *error (in)* input then the code in the green box does not execute.

The for loop steps through each list section of the ".ini" file. Each list section corresponds to one of the input cluster constants *(LC930x_TimeBase, LC930x_Vertical_Setup, LC930x_Trigger_Edge_Setup, LC930x_Read_Wave)*. The **MGI Read Anything** VI needs to know the format of each section. This is accomplished by the cluster constants being converted into variants for the Read Anything variant input. For more on the MGI VI's and how they operate, refer to their help manuals respectably. Once the **MGI Read Anything** VI has pulled out the relevant data, then the **variant to data** vi is used to reformat the output back to the cluster constant. The output for each cluster only executes when its section is read (hence the when integer counter =0. =1, =2, =3, then output data).

Back to Misc. Table 3.5

Chapter 4

Operation

4.1 File Management

4.2 Data Acquisition

Bibliography

[1] Tomás E. Gómez and Álvarez-Árenas. Air-coupled ultrasonic spectroscopy for the study of membrane filters. *Journal of Membrane Science*, 213(1-2):195–207, March 2003.