By: Austin Scholp

University of Wisconsin - Madison

Excised Booth Program Guide

Contents

[Front Panel Guide 3](#_Toc79743141)

[Main Menu 3](#_Toc79743142)

[Calibrate Sensors 3](#_Toc79743143)

[Collect Data 4](#_Toc79743144)

[Record Audio 5](#_Toc79743145)

[Analyze 6](#_Toc79743146)

[Convert LVM 6](#_Toc79743147)

[Split Files 7](#_Toc79743148)

[Open Manual 7](#_Toc79743149)

[Program Guide 7](#_Toc79743150)

[Report Error/Submit Request 7](#_Toc79743151)

[Block Diagram Guide 8](#_Toc79743152)

[Main Menu 8](#_Toc79743153)

[Calibrate Sensors 9](#_Toc79743154)

[Save Calibration File SubVI 9](#_Toc79743155)

[Collect Data 10](#_Toc79743156)

[Time Since Calibration SubVI 10](#_Toc79743157)

[Data Collection VI 10](#_Toc79743158)

[Check for Calibration Values SubVI 11](#_Toc79743159)

[Parellel Loops Section 11](#_Toc79743160)

[Event Handling Loop 11](#_Toc79743161)

[Data Collection Loop 13](#_Toc79743162)

[Data Save Loop 14](#_Toc79743163)

[Error Detection Loop 16](#_Toc79743164)

[Record Audio 16](#_Toc79743165)

[Analyze Data 17](#_Toc79743166)

[Load and Split Signals SubVI 17](#_Toc79743167)

[Pressure Flow Analysis SubVI 18](#_Toc79743168)

[Initialization 18](#_Toc79743169)

[Trigger Markers 18](#_Toc79743170)

[While Loop 19](#_Toc79743171)

[Closing 19](#_Toc79743172)

[Save Booth Data SubVI 20](#_Toc79743173)

[Convert LVM 20](#_Toc79743174)

[Split Files 21](#_Toc79743175)

[Open Manual 21](#_Toc79743176)

[Program Guide 21](#_Toc79743177)

[Report Error/Submit Request 22](#_Toc79743178)

[Timeout 22](#_Toc79743179)

# Front Panel Guide

## Graphical user interface Description automatically generatedMain Menu

The Booth Program, marked by the neat little UW Crest in the taskbar, is used to collect flow, pressure, EGG, and acoustic data on excised larynges within the booth. Unless changes need to be made to the program, you can and should open it through the shortcut. All the program files should be saved in *G:\1-Booth Program Updated – 2019*.

Graphical user interface, text, application, chat or text message

Description automatically generatedOnce the main controller is opened, you can choose one of the 11 options listed. If you hover over the button, a short description of its function is available.

If the user spends more than two minutes without clicking any buttons, the timeout prompt will trigger. If there is no input after another minute, the program will close.

The following subsections will go over what happens when each button is pressed. The second half of this guide will describe the block diagrams (the code portion) for all these events.

### Calibrate Sensors

Graphical user interface

Description automatically generatedWhen the **Calibrate Sensors** button is pressed, it will open this SubVI. All the user needs to do is make sure there is no flow though the flowmeter and that supra- and sub-glottal pressure sensors (these should be labeled) are open to the air (i.e., reading atmospheric pressure). Then, simply hit the calibrate button. This will zero all the sensors.

There is no need to collect data points at different pressure and flow levels as was done in previous versions of this program.

**Note:** The slope value for the supra-glottal pressure transducer is currently the same as the sub-glottal pressure transducer. This is because they use the same model of transducer. If the supra-glottal transducer is replaced with a more sensitive transducer (which will likely be needed by the straws team at some point), the slope value will need to be updated within this SubVI. Please see the All-Purpose Calibration guide for more details on how to do this.

### Collect Data

A picture containing text, screenshot, indoor

Description automatically generatedOnce the collection SubVI is opened, it will start running automatically. You should see data on all the front panel charts. If you turn up the flow, you should see a response in the flow chart (top chart on the left side) as well as the numerical readout on the top bar. You can do the same thing for pressure. If both show up as unchanging zeros, you might have overwritten the calibration files and will have to re-run calibration or move previous files into the Calibration Files folder. If only one does not response, there might be a problem with the connection to the DAQ board. On the right ¾ of the front panel, you will see, an acoustic intensity graph, an FFT display where you should see frequency peaks if you play a tone into the microphone, and EGG intensity graph, and an SPL chart which should respond to changes in sound intensity.

You will notice that the ‘Start Trial’ button is initially disabled. This I because you must select a place to save you data prior to collecting said data. Do this by clicking on the folder icon to the right of “Select Base Save Path”.

After a path is selected, you are ready to collect data. Presumably, you have your larynx mounted, tested it to see if it will phonate at all, and have the high-speed camera set up if you would like to record videos. All you must do to record flow, pressure, and acoustics is press ‘Start Trial’. The button should turn red and now say ‘Stop Trial’. Additionally, the Recording light should turn red, and the Saving Data light should turn green.

During a trial, you can trigger the camera from the LabVIEW collection program. To have the ‘Camera’ button trigger collection on the PFV program, you must have the program set to record. You should see ‘Trigger In’ instead of ‘Record’ on the live view in the PFV program.

Once you have finished recording your trial, clock the ‘Stop Trial’ button. At this point, there may be data that has not been written to the file. If this happens, the Saving Data light will remain green, and the ‘Start Trial’ button will be disabled. You must wait until the save is complete before closing the program or starting a new trial. This should only take a few moments.

### Record Audio

Clicking the Record Audio button brings up an interface similar to the main collection program. The difference is that only acoustic signals are read from this program. F­0 and SPL are still calculated during this collection; however, the SPL is not currently calibrated for this purpose so will only have relative values. You will need to run the Gain Estimator VI to get a new gain estimation and change that constant in the block diagram.

There are large text tabs on the right. These are vocal tasks that are performed as part of CAPE-V data collection and analysis. You can change what text is displayed by clicking on the different tabs on at the top.

Graphical user interface, application

Description automatically generated

### Analyze

A screenshot of a computer

Description automatically generated with low confidenceWhen you click the Analyze Data button, you will be prompted to select a file to analyze. The file will then be loaded and displayed on the two graphs. It may take a while to load the data depending on the size of the file.

Once the data is loaded, the user can select sections of data to analyze. This is done through the use of cursors. On the upper graph, you can move the cursors left and right to control the bounds of the bottom graph. The cursors on the bottom graph can be moved to select the section of data that you would like averaged. When you are ready, you can click the ‘Mark Selection’ button and the averages will be stored. When you have marked all the sections, click ‘OK’ and you will be prompted for a save location.

### Convert LVM

To analyze the acoustic data that you have previously collected, you will need to convert the LVM (LabVIEW Measurement File) into a WAV file so that it can be run in our other analysis programs.

Graphical user interface, chart

Description automatically generatedTo do this, click the ‘Convert LVM’ button on the main controller. This will open up the file explorer and allow you to select the folder where your data is saved. Select the folder, and the program will run through all the LVM files in that folder and create WAV files that are the same name as the LVMs. These are then saved in the same folder.

When files are being converted, acoustic data will be displayed on screen along with a progress bar for how many of the found files have been converted.

### Split Files

You should only need to run this if you make a data file that is too large to be opened by the analysis program. Hitting this button calls a Python script called *LVMSplitter.py* that should be in the *Python* folder. Running this will prompt the user to select a folder. It will then search that folder for extra-large LVM files and split them into usable chunks.

### Open Manual

This opens the 2015 Excised Booth manual. This document describes all the parts of the excised booth. It does not contain a user guide.

### Program Guide

This button opens whatever version of this program guide is in the project folder. Hopefully, it will be the same one as what you are reading now.

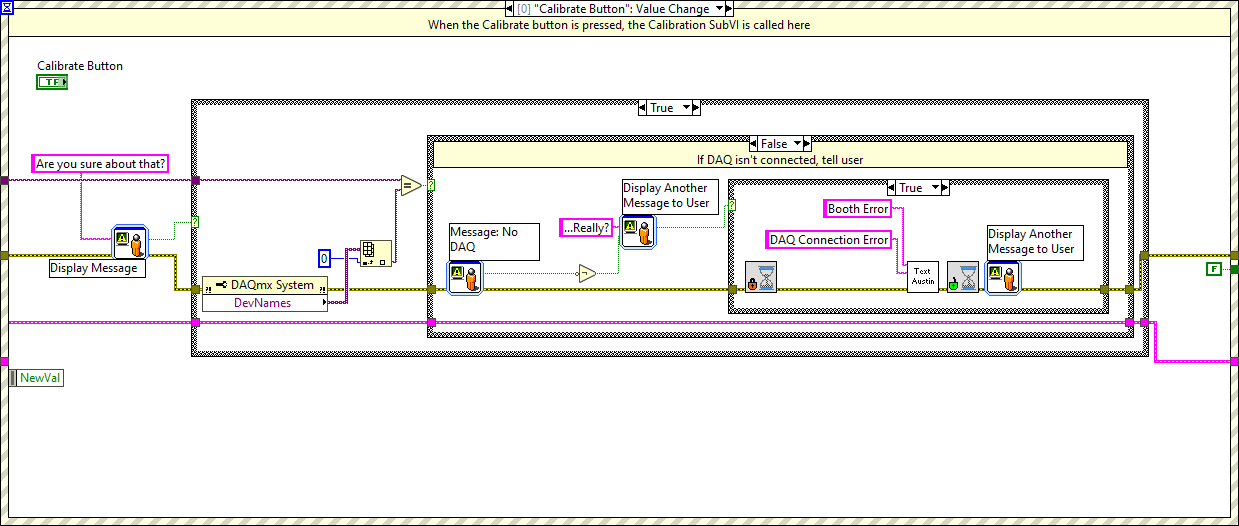
### Report Error/Submit Request

Graphical user interface

Description automatically generated with medium confidenceClicking this button will open a Google Form in your default internet browser. From there, you can report errors that you run into, request edits be made on a program, or request a completely new program. All submission on this form will notify Austin and he will respond whenever he can.

# Block Diagram Guide

## Main Menu

The main menu uses an event structure. Each button on the front panel has a case in the block diagram. There are also some extra cases. Below are the different events that this is setup to handle:

* **Calibrate Button** will open the calibration SubVI. All the other stuff in this case is to check if the user actually wants to open calibration (this is to avoid accidentally erasing the calibration values) and check if the DAQ board is connected. Once the SubVI closes, it will output the Flow Array, and the new slope and intercept values. They are also written to text files. (See calibration section)
* **Collect Button** will open the data collection SubVI in order to start reading from the sensors. It also reads in the previously collected slopes and intercepts. The extra stuff here checks if the DAQ board is connected and when last time calibration occurred. The user cannot continue if it has been more than a week.
* **Record Audio Button** will open a data collection SubVI similar to the main collection program that only records acoustic signals. There is also a check like the calibration and collection cases to see if the DAQ is connected.
* **Analyze Button** opens a couple SubVIs to analyze and save calculated values.
* **Convert Button** allows you to choose a folder full of LVM files and convert their recorded acoustic data into WAV files.
* **Split Button** runs a python script written by Kieran Paddock that splits LVM files if they are too long to be run by the Analyze SubVI.
* **Help Button** opens up the Booth Manual (not this document that you are reading but one that Graham Johnson wrote in 2015).
* **Email Austin Button** sends me an email to let me know that there is a problem with the booth. I will respond as soon as I can.
* **Report Error/Request Change Button** opens up a Google Form where someone can report errors or request changes to one of my LabVIEW programs.
* **Exit Button** does what you think it does.
* **TIMEOUT** if nothing happens for 20 minutes, this case will be triggered. The user will be prompted to check if they are still there.

### Calibrate Sensors

Shape

Description automatically generated with medium confidenceThis VI reads in signals through the DAQ assistant, like most of my data collection programs. The signals are then all run through a low-pass filter with a cutoff frequency of 5 Hz.

Diagram

Description automatically generated with medium confidenceThe signal is then split into subglottal pressure, supraglottal pressure, and flow. Each signal is then averaged. This is done to minimize the amount of variance in the signal. Each time the **Calibrate** button is pressed, the mean voltages, along with the hard-wired slopes, are used to calculate the intercepts.

Since we are assuming gauge pressure is zero (reading atmospheric pressure), the voltage just needs to be multiplied by the negative slope.

If the **Calibrate** button is pressed multiple times, the intercepts will be averaged, and the mean is saved to the calibration text file through the Save Calibration File VI.

#### Save Calibration File SubVI

Diagram

Description automatically generatedFirst, starting at the application directory (path where the program is located), this while loop strips the path until it reaches the root drive (e.g., D:/). Then, in the root drive, a folder named ‘Booth Calibration Files’ is created if it does not exist already. A folder named ‘Old Calibrations’ is also created (if it did not already).

The aerodynamics master uses this same setup to save new calibration files.

1. File/Directory Info locates the old calibration file and outputs the timestamp of the last time it was modified (same as date created).
2. Get Time/Date String converts the timestamp to a string.
3. A new name for the old calibration file using the timestamp is created using the concatenation string function.
4. The Move VI moved the old calibration file to a new path. The new path now ends with <*\Old Calibrations\Name\_date.txt*>
5. The new calibration values are then written to a ‘new’ text file that uses the same name as the original.
   1. I say ‘new’ because the file with the same name and path no longer exists.

### Collect Data

A picture containing diagram

Description automatically generatedBefore the collection VI is called, there is a check to see how long it has been since the booth sensors were calibrated. The user is asked when the last time the sensors were calibrated. Depending on the information from the Time Since Calibration SubVI and the user’s input, they may move on to the collection VI. Or they may be forced to run calibration .

#### Time Since Calibration SubVI

Diagram

Description automatically generatedThis VI uses the File/Directory Info function to obtain a timestamp from the most recently created calibration file. It then subtracts that time from the current time. Using some basic math, this time difference is split up into several time units.

#### Graphical user interface, diagram Description automatically generatedData Collection VI

First, the ‘Error!’ and ‘Stopped?’ Booleans are set to be False so that the loops do not immediately stop running.

Diagram

Description automatically generatedA picture containing chart

Description automatically generatedThe start of the three queues is shown to the left. The error queue is on top, the message queue in the middle, and the data queue on the bottom. These are all necessary to maintain communication between the four while loops. Here, we also see that the initial message (waiting) is enqueued.

Next, there are three copies of the SubVI called “Check for Calibration Values.” This SubVI opens the text files created during calibration and outputs the slope and intercept values for the subglottal, supraglottal, and flow sensors. The next subsection contains more details for this SubVI.

##### Check for Calibration Values SubVI

Diagram

Description automatically generatedFirst, the root drive folder is located. Then a path is built to the Booth Calibration Files folder and the specific file based on what is wired into the file name control. The scan from string function converts the text file string into two doubles.

##### Parellel Loops Section

The four loops of the collection VI are described here.

###### Graphical user interface Description automatically generatedEvent Handling Loop

Here is the Event Handling loop. The following sections describe each case handled.

START/STOP

A picture containing graphical user interface

Description automatically generatedIf the **START/STOP** Button is pressed, this event will be triggered. If the button is changing from False to True (starting trial) the Recording message will be sent. Additionally, the current timestamp is wired out to the shift register to be used by the other event cases.

Graphical user interface

Description automatically generated with medium confidenceIf the button is changes from True to False (stopping trial), the Saving message will be sent. Also, the timestamps for each time the camera trigger button was pressed during recording are written to a text file.

Diagram

Description automatically generatedEvery time the **STAR/STOP** button is pressed, the number of elements in the data queue is checked using the Get Queue Status function. This is used to update the maximum value on the progress bar. It will also be used in the data saving loop.

Camera Trigger

When the trigger button is pressed, the Recording message is added to queue to make sure that data is being recorded at the same time as the camera. It also sets the timeout time to 3000 msec by way of shift register. The trigger button needs to be reset to false automatically, which is what the trigger event case will do.

Trigger

Three seconds after the trigger button is pressed, this event is called. This turns the camera button off again and sets the timeout value to -1.

Exit

If the exit button is pressed, this case sends the Exit message…duh. It also wires a true Boolean out of the event structure to stop the event handling loop.

Error!

If an error occurs, this case also sends Exit message and stops the event handling loop.

Application Instance Close

If the LabVIEW application is closed, this case sends the Exit message and stops the loop.

Panel Close?

If the window with the data collection VI is closed, this case calls the Exit message and stops the event handling loop.

###### Data Collection Loop

Diagram, schematic

Description automatically generatedI know that this can look very intimidating, but you just need to follow the wires. I have tried to clean it up as much as possible, but there still a lot of overlapping wires.

Here is the Data Collection loop where the voltages are read in from the DAQ and displayed on the Front Panel. The DAQ Assistant Express VI opens the channels to read on the DAQ board and outputs them all in one signal wire. This is then split up and filtered. The pressure and flow signals go through similar low pass filters while the acoustic signal goes through a high pass filter. This is also where Boolean data is sent to the camera to trigger it. I do not know why it needs to be in an array, it just does.

The pressure and flow both need to be translated into their actual values using the slope and intercept found from calibration. This is done through the formula SubVI which just takes the voltage input *(x)* multiplies by the slope *(m)* and add the intercept *(b)* the outputs the actual pressure or flow *(y)*. These new calculated signals are then merged with all the others.

The acoustics go through three different SubVIs, Tone Measurements, Spectral Measurements, and the SPL Calculator. The first outputs the frequency peak of the FFT, also known as F0. The second just creates the FFT graph for the front panel. The SPL calculator outputs SPL.



All the signals (Flow, Pressure, Acoustics, SPL, EGG, F­0, and F­0-EGG) are then merged together before being aligned. If recording (start/stop button is set to true), the merged and aligned data will be added to the data queue. If not, nothing happens with the data. The last thing in this loop is the error info is added to the error queue.

SPL Calculator subvi

Diagram

Description automatically generatedThis is where SPL is calculated from the filtered acoustic signal. The equations used in this calculation are below.

­­

###### Data Save Loop

Graphical user interface

Description automatically generated with medium confidenceThis loop contains a state machine that is controlled by the enumerated messages sent from the event handling loop. There are four cases that can be called that dictate what happens in other parts of the loop. There is also a Logic SubVI that controls the disabled status of the **Exit** and **START/STOP** buttons, as well as the First/After case structure system.

Waiting Case

This is initially called before any loop starts. It will continuously send new ‘Waiting’ messages while the **START/STOP** button is false. It also de-queues elements from the data queue so that there is no data in the queue when it comes to start recording/saving data. When the **START/STOP** is true, this case sends the ‘Recording’ message.

Recording Case

Thissends a true value out to enable data saving to file. While the **START/STOP** button is true, it will keep sending ‘Recording’ messages until it is turned off. At this point, the saving case will be called. This case updates also the number of data elements in the queue and stores this value in a shift register.

Saving Case

While there are still data elements in the queue, this case will continue to send itself ‘Saving’ messages until the queue is empty. This ensures that all the trial data is written to file. Specifically, this saves the data that gets buffered in the queue during collection since the program cannot save the data as fast as it is collected. After there are no more elements left, the ‘Waiting’ message will be sent again.

Exit Case

Adds the ‘Exit’ message to the queue.

Logic SubVI

Diagram

Description automatically generatedThis is the logic controlling when the **Exit** and **Start/Stop** buttons are disabled. It also controls whether an error sound is played.

* If Saving Data is set to true OR the **START/STOP** button is on, then the **Exit** button will be disabled and the First or After Boolean will be set to false.
* If the **START/STOP** button is on AND the size of the data file being saved is greater than 80 MB, then the warn user Boolean will be set to true.
* The **START/STOP** button will be disabled if the Valid Path Boolean is set to false OR if the **START/STOP** not on, but Saving Data is set to true.
* The **Camera** trigger button will only be enabled if **START/STOP** is on.

In other words, the user cannot press the **START/STOP** button if data is being saved after recording has stopped or an invalid path is being used for the save file. The user also cannot exit the program while recording or saving data.

Error Sound SubVI

If the user keeps recording for too long, the file they create will be too big for the analysis VI to handle. To prevent that, this SubVI will play a Windows error sound when the save file grows beyond 80 MB. The **START/STOP** button will also start flashing.

###### Error Detection Loop

Graphical user interface, application, Word

Description automatically generatedThe Error Detection Loop pulls the error cluster data from the Data Save and Data Collection loops. If there is an error read from either loop, the Error! light is set to True and the loop is stopped. With the change, the other three loops will receive signals telling them to stop.

An error message will then pop up through the Simple Error Handler SubVI.

### Record Audio

Diagram

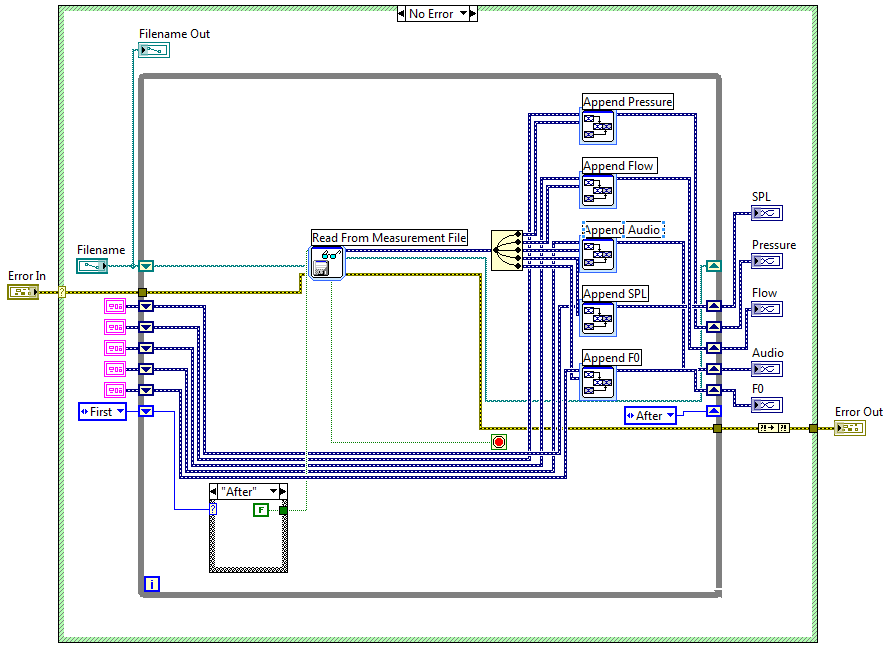
Description automatically generatedThis VI is set up the same way as the Data Collection VI. However, it is much simpler since only acoustic data is being recorded. Of the four while loops, only the data collection loop is different. I hope you can figure this out from here.

### Analyze Data

Diagram

Description automatically generatedThis process is comprised of three SubVIs placed within the ‘Analyze Button’ event case. The first loads and splits the signals read from a select LVM file. The second opens up a user interface where you select the sections you would like to average. The third saves the data the user selected into a TXT file.

#### Load and Split Signals SubVI

Since the data is saved in segments, it must be read again is segments. Each segment read in from the file and then appended to the previous chunk. The whole signals are output. In retrospect, I could have appended everything before splitting the signals.

#### Pressure Flow Analysis SubVI

After all the signals are loaded, they are displayed through this SubVI. A large portion of this block diagram is dedicated to managing the cursors on the two front panel graphs.

##### Initialization

1. The cursors that mark where the camera was triggered are created through the Trigger Markers SubVI.
2. Diagram, schematic

   Description automatically generatedThe initial positions and colors for the moveable cursors are assigned using the cursor list bundle that is wired out of Trigger Markers SubVI.
3. Graphical user interface, diagram, text

   Description automatically generated The for loop below sets the line type and width for all the cursors in the bottom (expanded data tract) graph.
4. Diagram

   Description automatically generatedThe dynamic data for the signals is converted to waveforms so it can be read later in the while loop.
5. Several arrays are also initialized here. These will store values to be saved later.

###### Trigger Markers

1. Qr code

   Description automatically generatedThe text file containing the trigger times is located and opened.
2. Diagram

   Description automatically generatedThe string containing the trigger times is converted to a 1D array of doubles.
3. The default values for the left and right (moveable) cursors are bundled into the pink cursor cluster.
4. All pink cursor is created for each time found in the array of doubles.

##### While Loop

Inside the loop, the positions of the cursors are read from both graph’s property nodes. Since the upper graph’s cursors determine the lower graph’s bounds, the lower cursors (Data Trace Expanded) must be between the upper cursors.

A picture containing qr code

Description automatically generatedThis is checked with the “In Range and Coerce function”. The upper and lower limits are the right and left cursors of the upper graph (Data Trace).

Diagram

Description automatically generatedIf the bottom graph’s cursors end up outside the range, they will be moved to 0.1 seconds inside the upper graph’s cursor values. This is so the user does not accidentally lose the cursors when they zoom in to view a section of data.

A range of data to average will be needed. This is the absolute value of the difference of the positions of the two cursors on the lower graph.

Diagram

Description automatically generatedUsing the Get Waveform Subset SubVI, and the range and left cursor value of the bottom graph, a section of data is selected. The means of those sections are calculated and displayed.

Diagram

Description automatically generated with medium confidenceDiagram, schematic

Description automatically generatedWhen the ‘Mark Section’ button is clicked, the selected data sections are added to their respective arrays. The Sections Marked value is also incremented here.

As section means are added the array, the overall mean will also update. These means, along with the arrays are written out back to the controlling VI to be saved in another SubVI.

##### Closing

Once the user is done marking sections that they want to analyze, all of the data is bundled into a cluster. This cluster is then wired out to the Save Booth Data SubVI that will be covered on the next page.

#### Save Booth Data SubVI

Diagram, schematic

Description automatically generatedEach array first needs to be converted into a spreadsheet string using the Array to Spreadsheet Sting function.

Once everything is either a string or a numeric (DBL) it can be formatted into a string. The extra inputs here are titles, tabs, and new line constants.

The main mess here is the many, many wires going into the Format Into String function. Be careful if you change anything here. Make sure you follow where each wire actually goes.

After everything is formatted into a string, it can then be written to a text file.

### Convert LVM

To start, the SubVI sets the cursor to *busy* while the program is running. Then, using the List Folder function, all the LVM file names within the selected folder are output into an array of strings. The size of this array (i.e. the number of files found in the folder) determines how many times the for loop will run.

The for loop will run through each of the file names. Using the Build Path VI, it will take the file name from the array and the folder path then wire that into the while loop.

Inside the while loop, the Read From Measurement File VI will read the LVM and output all the data. It is then split (the third signal in saved data is the acoustics. This is just how it is saved in the collection program) and appended to any previous data from the loop, through the use of shift registers, and displayed on the graph.

After the data is all added together, it is taken out of the loop, converted to a 1D array of waveforms, and then put into the Sound File Write VI. The path used for this is the same as the one that is read by the Read Measurement File VI but the ‘.lvm’ at the end is replaced with a ‘.wav’.

After all the file names in the array are run, the cursor is unset from busy.

### Split Files

Text

Description automatically generated with medium confidenceWithin the event case, the Python folder is located, and a command is wired to the System Exec VI. This opens the LVM Splitter. The Python script will prompt the user to select a folder. All the files above a specific file size will up split up into usable chunks.

If you get an error here, make sure that the Python compiler is installed on the computer.

### Open Manual

A picture containing diagram

Description automatically generatedThis event case locates the Documents folder within the Project and VIs folder. It then sends a command to the System Exec VI to open the Excised Manual PDF.

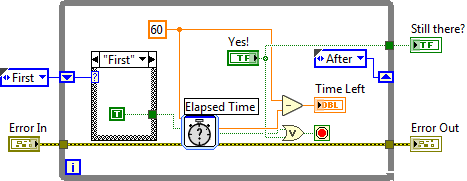
### Program Guide

This does the same thing as Open Manual, but with this document you are reading.

### Report Error/Submit Request

Uses a premade SubVI to open a URL in the default browser.

### Timeout

If 20 minutes go by without an event occurring, the user will be prompted. It is simply a while loop using the Elapsed Time Express VI. The same timeout check is used in the aerodynamics program.