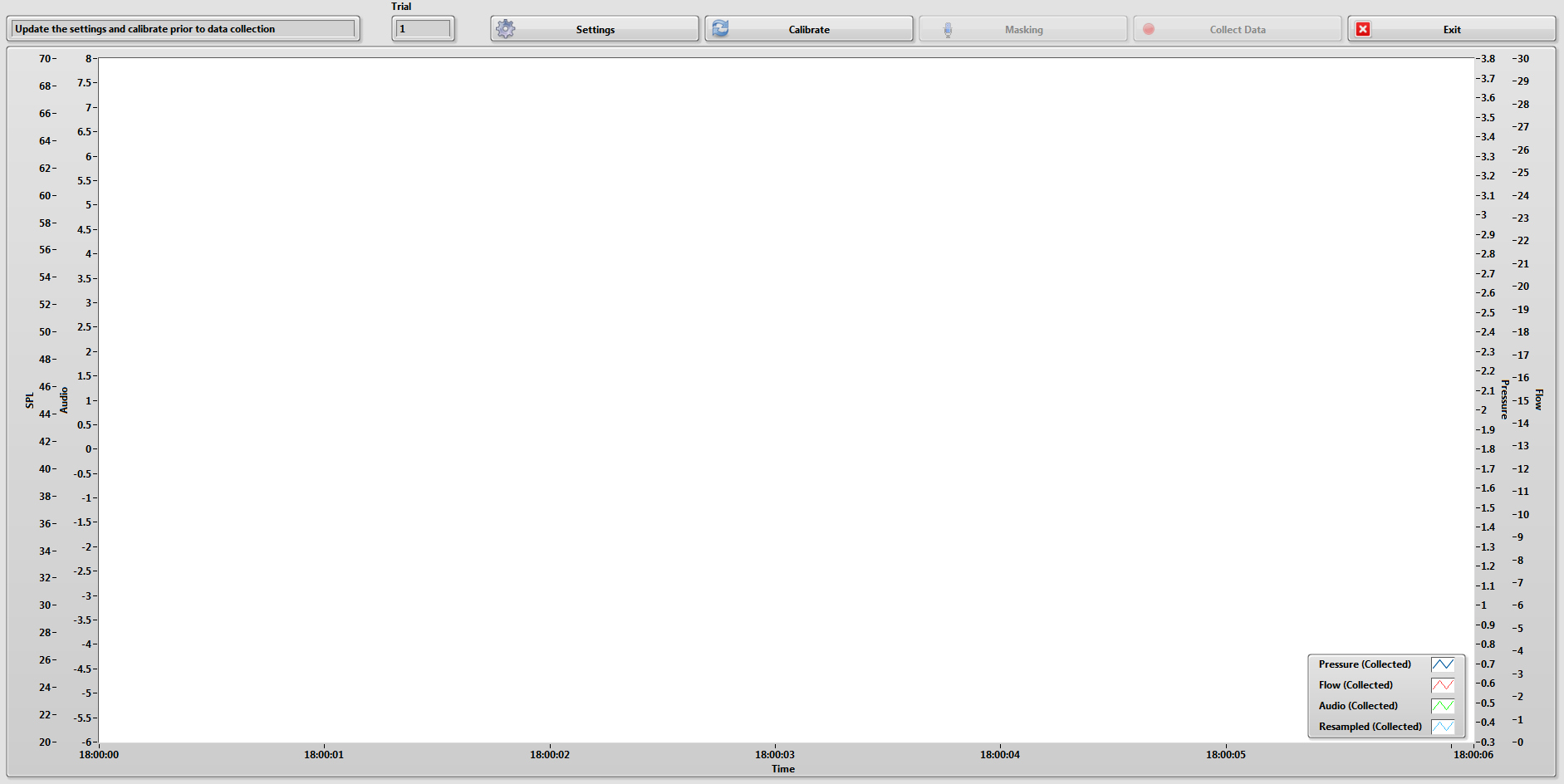
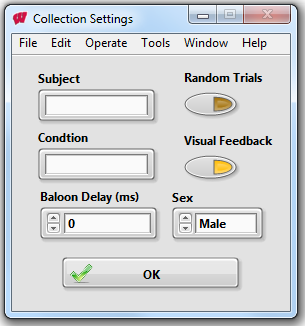
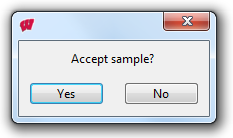
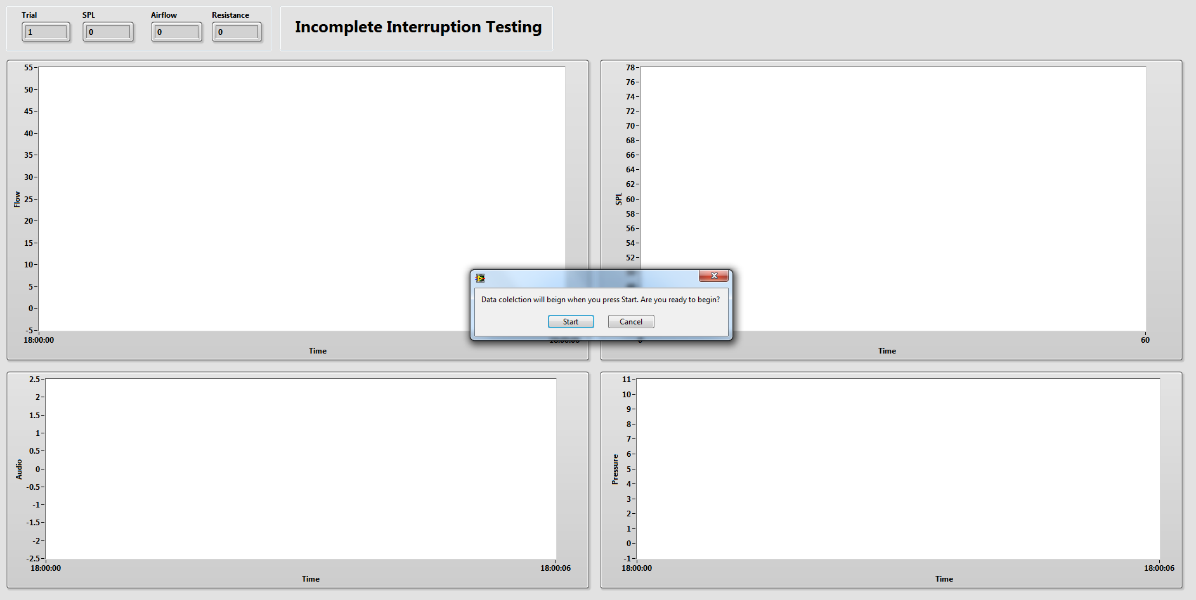
# Incomplete Interruption Collection

## Purpose and USE

This program is designed to collect data through a DAQ board using the incomplete interruption device. If you do not know what that is, you should figure that out before you attempt to *fix* my program.

Anyway, here is the step by step process to using this program:

1. Open the program through the Master Aero program. You can also open it through LabVIEW if you would like, it is eaiser to use the installed program or the executable.
2. At first, the “Masking” and “Collect Data” buttons are disabled. This is because you must update your settings prior to creating an audio mask and collecting data.
3. Once you open settings, you can set:
   1. the subject number
   2. the condition to be tested
   3. how long the balloon will be inflated during the interruption
   4. whether or not you will be doing randominze resistences
   5. whether or not you want to have visual feedback
   6. and the sex of the subject
4. After you set your…uh…settings, you can now create an audio mask and collect data. While you do not have to make a mask, most trials will be using audio feedback. If no mask is made, there will be not feedback.
5. To create a mask, press the “Masking” and a window will pop up. Once you are ready to create the mask, press “Start” and have the subject say “Ah” into the microphone for about 3 seconds. You should hear a tone played back to you that should match the subject’s pitch.
6. If the playback pitch did not match, just hit “No” and the program will start listening for a new pitch. You can repeat this as many times as needed until you find a comfortable pitch for the subject (The first time usually is fine though).
7. Now you are ready to collect data…so click “Collect Data”
8. Assuming you left visual feedback on in the Settings window, the Data Collect window should look something like the image below. Once you click “Start”, the program will begin collecting flow, pressure, and acoustic data.
9. The program will record for a set number of seconds. At the time I am writing this, it is set to 6 seconds (this is a constant that we’ll get to later). This might have changed based on the needs of the project.
10. The program also automatically trigger the balloon valve to fill for a single interruption. This also might have changed to suit the needs of the project.
11. After it is done recording, the program will prompt you asking if the trial was acceptable, assuming nothing went wrong, you can click “yes” and more prompts will follow regarding saving the data. This part is pretty self-explanatory.
12. Once saved, you will be prompted to run another trial. If you select not to run again, the trial counter resets to 1. It is easier to run multiple trials under 1 condition.
13. Repeat ad nauseam.

## THE BLOCK DIAGRAM

If you need to change something on this program, please review this prior to making edits to the LabVIEW files.

This program is set up in a queued message handler (producer/consumer) structure. If you do not know what that means, please refer back to the section of this binder that covers program structures (advanced tab).

The program contains 2 parallel loops: the Event Handling Loop and the Message Handling Loop.

### Here is the EVENT handling loop

This loop contains the event structure. If this phrase confuses you, go review what an event structure is. This structure has 8 cases. Cases 0 through 3 and number 6 correspond to events where a button on the front panel is pressed. The rest handle application/panel closing. These are just different exit situations that you should not change.

### Here is the Message Handling Loop

This loop has a case structure with 13 cases for different messages sent in. You’ll find descriptions for each case on the following page.

Note: These cases are not in the order as they appear in the case structure, but in an order of how they would be called in a typical run of the program.

1. **Initialize** – Sets control references for each of the items on the front panel. These are then bundled into a cluster so that can be used later. Also, the next two cases are added to the message queue.
2. **Initialize Data** – Sets the default values for all of the data to be stored between cases.
3. **Initialize Panel** – This sets the front panel objects to their default states.
4. **Update Display** –Sends a new message to the UI to display to the user. It also updates the data display and trial number with the most recent info added to the data queue.
5. **Calibrate** – This opens up the sensor calibration SubVI. Calibration values are then added to the data array. The Update Display case is called again.
6. **Settings ­**­– The settings SubVI is opened and the user selected info is added to the data queue. Also, the Collect and Masking buttons are enabled. The Update Display case is called again.
7. **Masking** –Opens the audio mask setup SubVI and saves the frequency found to the data queue. The Update Display case is called again.
8. **Collect** – First, the exit button is disabled then the collection SubVI is opened. After the data is collected, it is taken from the collection SubVI and input into the title string creator and the save data SubVI. User input messages are also presented to control what to do after collection.
9. **Exit** – The message queue is released and the loop is stopped. All other loops are also stopped.
10. **Error** – If an error is thrown from the error detection SubVIs in the Event or Message handling loops, this case will be called. Right now, this just stops the loop.
11. **Confirm Quit** – If I wanted to double check if the user wanted to exit, I would do that here. Right now, there is no check, the loop is just stopped.

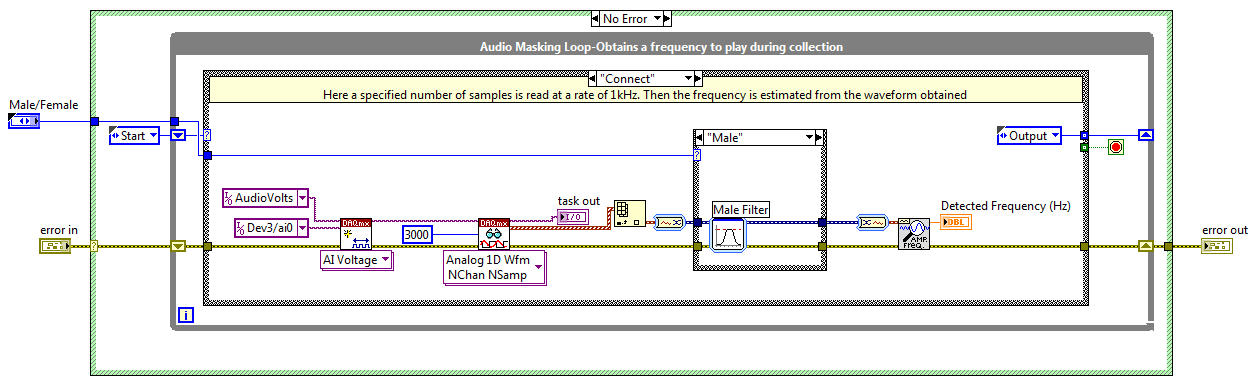
#### The Settings SubVI

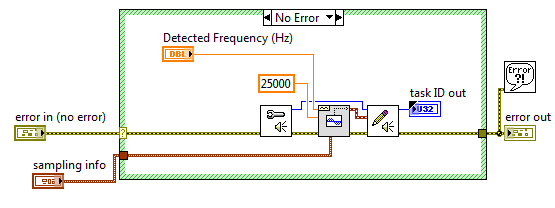
The settings VI is set up in a while loop that waits for the user to press the OK button. When that happens, the values the user inputs through the front panel are output from the SubVI to the controlling collection VI.

The random trials button currently controls the default condition string. This feature is not currently used.

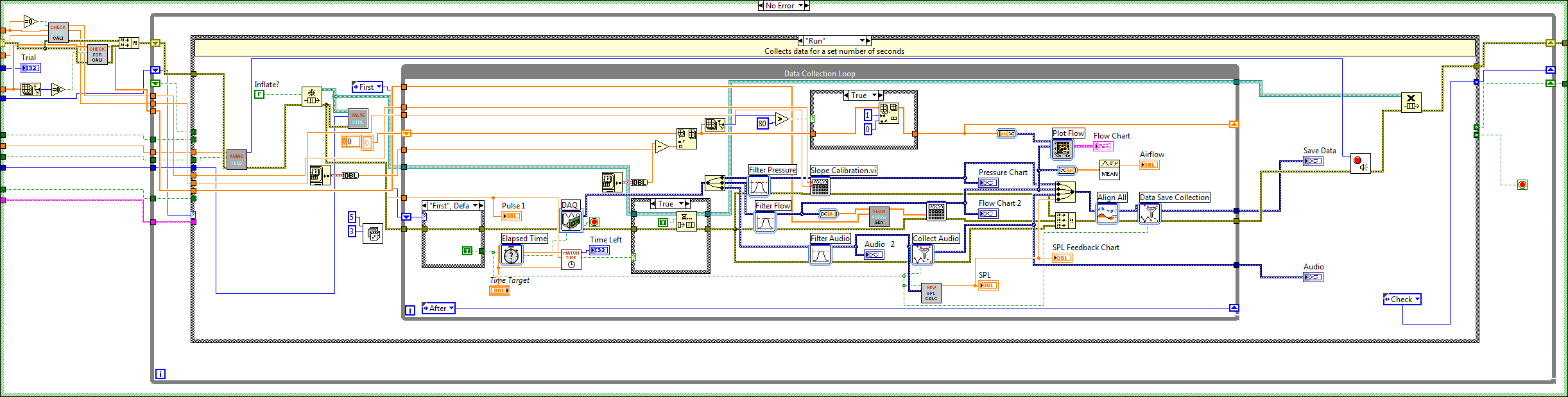
The sex selection is used during the masking SubVI to control filter settings. This is not saved anywhere else.

#### The masking Subvi

This SubVI is set up in a simple state machine. The cases are outlined below:

* **Start** – Contains a while loop that stops when the user hits Start through the One Button Dialog. The Connect case is called next through the shift register.
* **Connect** – 3000 samples at 1kHz are read in and the frequency of the signal is calculated. There are also 2 filter options (male and female) that make the calculated frequency more accurate for different subjects. The output case is called next.
* **Output** – The detected frequency is put into the Sound Out SubVI, which simply outputs a sound with the detected frequency to the speakers. The check case is called afterwards.
* **Check** – Prompts the user to accept the sample. If it is accepted, the Quit case is called, if not, it will return to Connect.
* **Quit** – Stops the loop.

#### Collection SUBVI

This SubVI is set up in a state machine structure. I made this one first, that is why is different from the complete interruption and redirector collection programs. I could remake it so it is cleaner and more similar to the others, but that takes time that I am not going to spend right now. Anyway, the cases are listed on the next page.

* **Start** – Initializes the UI objects to their default values. If visual feedback is turned off, that will hide the charts in this case. There is also a prompt for user input. The user can either start the trial or exit the program.
* **Run** – Collects data from the DAW board for a set number of seconds. There is only one interruption set for this program. Once the elapsed time matches the randomly generated number, the balloon will be triggered. Once the time target is met, the check case is called next.
* **Check** – The user is prompted to retry the collection trial, or accept and save it. If accepted, the Exit case will be called. If not, the start case is called.
* **Exit** – Exits the loop.

After the data is collected, the Collection SubVI is exited and data is output to a couple other SubVIs that create the file name and then save it to an LVM file. If you know how case structures work, the rest of this case is pretty self-explanatory. Also, the data collected is displayed on the main front panel.

Now that I am going through this program, I realize how messy it is. If you are reading this and the program does not look like the one I have described here then I have probably remade it to be more similar to the complete interruption collection program.