Spectral HRM Toolkit Manual

University of Wisconsin - Madison | 1300 University Ave. 2766 MSC

By: Austin Scholp

Last Updated: August 19, 2021

2021

Table of Contents

[Front Panel 2](#_Toc80352434)

[Main Menu 2](#_Toc80352435)

[Open HRM Exam File 2](#_Toc80352436)

[Pick Event & Select Sensors/Add Annotations 2](#_Toc80352437)

[Run SPARC 3](#_Toc80352438)

[SPARC Settings 4](#_Toc80352439)

[Batch SPARC 4](#_Toc80352440)

[Run NMD 4](#_Toc80352441)

[Create Figures 4](#_Toc80352442)

[Compile Data Reports 5](#_Toc80352443)

[Help 6](#_Toc80352444)

[Block Diagram 6](#_Toc80352445)

[Main Menu – Initialization 6](#_Toc80352446)

[Main Menu – Event Handling 6](#_Toc80352447)

[Open 6](#_Toc80352448)

[Add Annotations 6](#_Toc80352449)

[Event & Sensor Select 6](#_Toc80352450)

[Batch 6](#_Toc80352451)

[Settings 6](#_Toc80352452)

[SPARC 7](#_Toc80352453)

[NMD 7](#_Toc80352454)

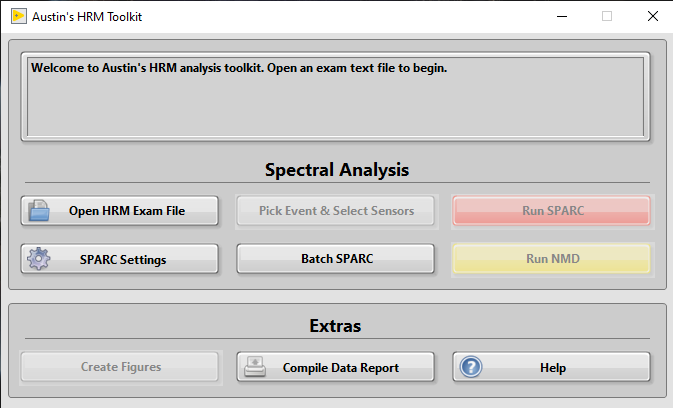
[Figures 7](#_Toc80352455)

[Extract 7](#_Toc80352456)

[Help 7](#_Toc80352457)

# Front Panel

## Main Menu

The main menu, shown on the right, is the first window shown to the user.

The following sections describe what happens when each of these buttons is pressed.

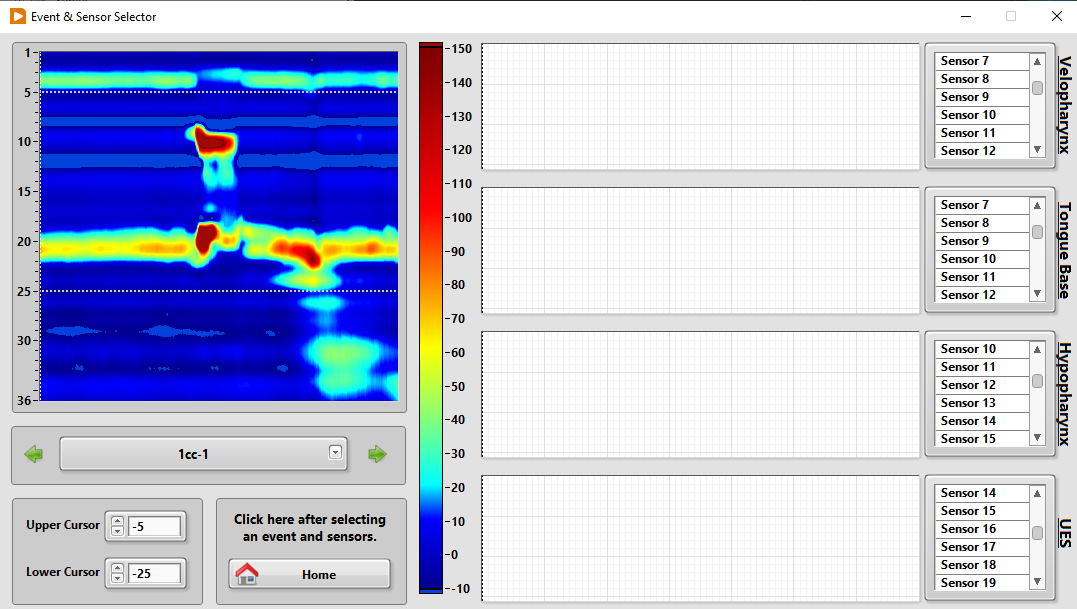
In the second half of this document, you can find detailed descriptions of the code (Block Diagram).

### Open HRM Exam File

Here, the user is prompted to select an examination file for analysis. Note that only text files will be accepted. Once selected, the program checks if the file has events annotated. The annotation status is added to the main menu message box. If the selected file has annotations, the **Pick Event & Select Sensors** button will be enabled.

If there are no annotations, the **Add Annotations** button will appear. Clicking this will prompt the user to open the XML file that corresponds to the selected exam file. The annotations will then be added from the XML file to the exam text file.

### Pick Event & Select Sensors/Add Annotations

In this menu, the user selects an event they want to analyze using the dropdown menu. They can move the data window one second forward or backward using the green arrows.

Users then assign sensors to each pharyngeal region by either ctrl-clicking or shift-clicking the sensor numbers in the boxes on the right.

Changing the upper and lower cursor values will move the dotted white lines and disable selection of sensors outside those bounds.

Once satisfied with their select, the user can press the home button to return to the main menu where the **Run Sparc** and **Run NMD** buttons should now be enabled.

### Run SPARC

Diagram

Description automatically generated with medium confidenceThe user is first prompted to select the swallow region using two vertical cursors.

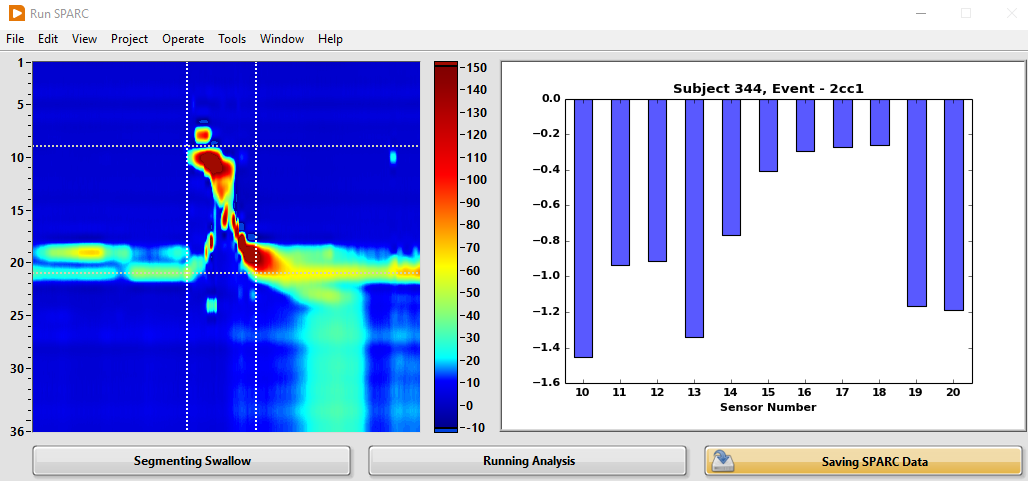
The left bound should be placed a little before where pressure starts to increase on the uppermost sensor trace.

The right bound should be placed at the peak in UES pressure that indicates the end of the swallow. Everything that follows should happen automatically.

Chart

Description automatically generatedThe user will only need to interact with this window (left) if the program cannot not find a definite peak in the last sensor. The user is then prompted to select it manually.

If the program can find the peak by itself, the user will only see this window for a brief moment before it moves on to the next one.

Here, the program displays where the select swallow region is and SPARC values for each sensor. The buttons on the bottom will light up to indicate where the program is during the process of segmenting and analyzing the signals.

When complete, the user will be notified if the analysis results were saved to a new file or added to an existing one.

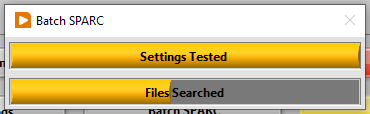
### SPARC Settings

Graphical user interface, application

Description automatically generatedThis button will open a menu to allow users to change the SPARC analysis settings (Cut-off frequency, amplitude threshold, and pad level).

These settings are saved in a text file (aplty named ‘SPARC Settings’). The program will use the settings found in that file so the user doe not need to update the settings every time they user the program.

### Batch SPARC

This will run SPARC analysis on a folder of exam segments. The segments are created for each event that is analyzed during the regular analysis. Thus, regular analysis needs to be completed for a subject before this can be used.

The user will be prompted to run analysis using the current settings or to run through different permutations of the settings. The permutations option should only be used if the user wants to see which combination of setting produced the best ROC curve.

### Run NMD

This is very similar to the SPARC analysis. The user is prompted to select the bounds of the swallow, then NMD analysis is run, and the results are displayed to the user.

### Create Figures

Opens a menu allowing the user to select different figures to create. The different figures are described in the next subsections, starting on the next page.

#### Graphical user interface, application Description automatically generatedCreate Spectral Plot

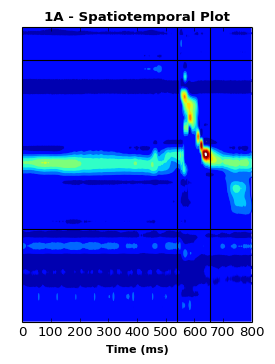
The user is prompted to select a segment then what they want the lower bound of the bar plot to be. They are then presented with a figure displaying the pressure traces, speed profiles, the full Fourier transform spectrum, the trimmed spectrum, and the SPARC values for each sensor. Each part of the figure shown is saved in its own PDF.

#### Create Bar Plot

Chart, box and whisker chart

Description automatically generatedThe user is prompted to select the ‘Subject Means’ text file that is created after the user clicks the **Compile Data Reports** button on the main menu. Using the data in the ‘Subject Means’ file, averages (bar lengths) and standard deviations (error bars) are calculated. These are them compiled into a bar chart.

#### Save Spatiotemporal Plot

This will save a PDF of the spatiotemporal plot for whatever event the user selected in the Pick Event & Sensors stage.

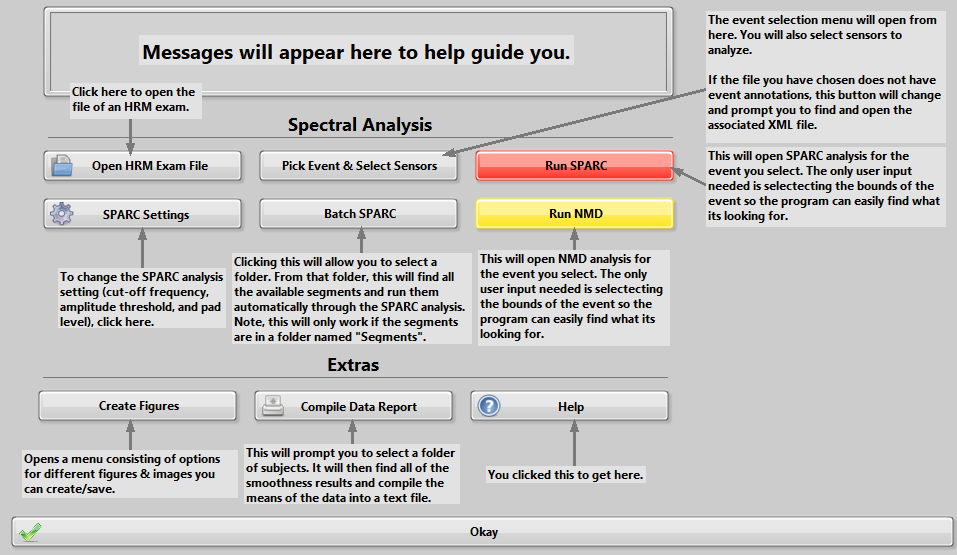
If no event was loaded and/or no sensors were assigned to the pharyngeal regions, the user will receive a warning.

Before the plot is saved, the user is prompted to mark the start and end of the swallow. This is done through the same window that appears when running analysis

### Compile Data Reports

This prompts the user to select a folder. It then compiles the SPARC data from all the smoothness analysis files in that folder and saves it into one file.

### Help

This just opens up a window that describes what each button does.

# Block Diagram

## Main Menu – Initialization

A picture containing application

Description automatically generatedFirst, the program establishes references for all of the front panel buttons. These are then bundled into the user interface (UI) cluster. A data cluster is also created prior to the while loop.

Diagram

Description automatically generatedThe references for the **Extract** and **Add Annotations** buttons are built into an array and wired to the Enable Buttons VI, enabling them on the front panel. The other buttons are disabled initially.

### Enable Buttons SubVI

Diagram

Description automatically generatedThe for loop iterates through the array of references wired in from the calling VI. All the controls referenced will be enabled or disabled based on the status of the ‘Enable?’ Boolean. It also controls whether they will be visible or not.

## Main Menu – Event Handling

I hope you are familiar with event structures by now. The following sections describe the code that is called for each user event that this structure handles.

### Open

A picture containing square

Description automatically generatedWhen the **Open HRM Exam File** button is pressed, this case will be called. Here, the Get Data from File VI is used to do most of the information processing. This case also updates parts of the front panel based on what can be found in the file that the user chooses.

If the chosen file contains annotations, the user will be prompted to go onto the next step of analysis. Otherwise, they will be asked add annotations to the exam file.

#### Get Data from File SubVI

The user is prompted to select an exam file using the File Dialog express VI. Assuming that file selection was not cancelled, the program moves on to open the file and extract the data formatted as a spreadsheet string (see next subsection).

Diagram, text

Description automatically generatedThe spreadsheet string in then converted into a 2D array of strings, and a 2D array of doubles. The array of doubles is broken up into three subsets. One is just the sensor data (36 sensors, all time points) The second is just the first row (just the sensor numbers). The third is just the first column (timestamps). These are all later bundled into the data cluster.

Using the path of the selected file, the subject number is extracted (i.e., the name of the containing folder) and the Check Subject Group VI checks a reference text to find what group the selected subject belongs to (e.g., control group).

Outside the case structure, the Find Annotation Times VI is uses the 2D array of strings to produce four arrays. The Event Times array is just all the timepoints where an annotation is found. The Start and End Times indicate 4 seconds before and after the Event Times. The Event Names contains all the annotation text.

##### Get Spreadsheet String SubVI

Diagram

Description automatically generatedThis SubVI uses the Open/Create/Replace File, Read from Text File, and Close File functions to open the user-selected exam file and obtain its data in spreadsheet string format.

##### Check Subject Group SubVI

This SubVI uses the Get Spreadsheet String SubVI to open the text file called ‘Subject Group References’. This file should contain a list of subject numbers and the group to which they belong. The subject number for the user-selected exam file is cross-referenced with this list to find the appropriate group name. It does this by

1. Converting the spreadsheet string to a 2D array of strings.
2. Indexing the 2D array to obtain the first (index zero) column.
3. The Search Array function is then used to locate the subject number in the indexed array of strings.
4. Diagram

   Description automatically generatedIf the subject number is found, the index at which it was located is used to index the 2D array to find the group name. If the subject is not found, the user will be notified via the one-button dialog.

##### Find Annotation Times SubVI

This SubVI just calls two other SubVIs (Notes Table and Adjust Time) to obtain the event, start, and end times along with the event labels.

###### Notes Table SubVI

Diagram, schematic

Description automatically generatedTaking the first column of the full text file, this SubVI searches for what row (if any) contains the string ‘Annotations’. Every row after that should contain the event times in the first column and the labels in the second.

Find Swallow Start-End SubVI

If the end of a swallow is annotated, then the Find Swallow Start-End VI will exclude the that label and time. If the pound sign (#) is included in an annotation, only what comes before it will be included in the labels.

###### Adjust Time SubVI

This SubVI iterates through the string array containing all the event times that are associated with annotations. Each string is first converted to a double and added to the event times array. Then four seconds or four-thousand milliseconds are added and subtracted from the event time. Whether seconds or milliseconds are used depends on how the annotation times are formatted.

Diagram

Description automatically generatedIf the event times are formatted as seconds, they will include decimals. Thus, the quotient and remainder would not be zero. If formatted as milliseconds, there will be no decimals and the remainder would be zero.

### Add Annotations

Diagram

Description automatically generated with low confidenceThis case calls the Add Annotations to TXT VI. Here, the [Get Spreadsheet String VI](#_Get_Spreadsheet_String) is used to get the text exam data and the XML Parsing VI is used to obtain the annotations.

A 2D array of strings is output from the XML Parsing VI and the string ‘Annotations:’ is added to the start of the array. The 2D array containing all the annotations, times, and the ‘Annotations’ label is added to the bottom of the 2D array containing the exam data. Everything is converted back into a single spreadsheet string and written to a new file.

#### XML Parsin SubVI

This SubVI is set up to parse XML and RML files. First, an XML parser session is started. This allows the program to use invoke nodes to read XML and RML files. Even I do not really understand how this works. Most of this SubVI is pulled from things I found online.

##### XML

Diagram, schematic

Description automatically generatedTwo invoke nodes are used to search for elements with the tags ‘text’ and ‘time. Each invoke node outputs an array of references. For loops then iterate through these arrays and use the Get Node Text Context VI (comes with LabVIEW) to extract the relevant text. The labels (text) and times are indexed into arrays which are then combined into a single 2D array of string called XML Notes and Times.

##### RML

Diagram, text

Description automatically generatedI do not remember if the RML Parsing VI came with LabVIEW or if I found it online. All I know is that it searches an RML file for XML strings and outputs arrays of the event labels and times. The for loop here is used to remove some of the tags found in the RML file. For example, the upper functions take the data between ‘Start=”’ and the ending quotes.

#### Create Annotation File Name SubVI

Qr code

Description automatically generatedThis SubVI just concatenates several strings to create a new filename for the annotated file. This is so the original file is not overwritten in case something goes wrong.

### Event & Sensor Select

This SubVI is organized in a state machine design pattern. The following subsection describe what happens during each state.

#### Initialize

Whether this case is called depends on the value of the Starting State control. This control will usually be set to ‘initialize’. It is set to ‘wait’ when there is still HRM data in the cluster and the event selector SubVI has been called previously.

This state calls two SubVIs. The first is used to initialize values for the heatmap used for displaying the HRM data. The second initializes the menus for selecting sensors. Everything else in this state is just setting initial values for various front panel objects through property nodes and bundling information in the data cluster.

##### Colormap to Array

Diagram

Description automatically generated

The conditional disable structure is used to set up the path for locating the colormap text file. If the VI is using the run-time engine (i.e., the user is using the executable version of the program), then the file will be found in the ‘data’ folder. Otherwise, it will be in ‘Extras’.

Diagram

Description automatically generatedThe first for loop takes the data from the colormap file (2D array of doubles) and splits it into three values (RGB). These are then wired into the RGB to color VI. Iterating through all the rows of the 2D array, a 1D array of integers is created that contains singular values for each RGB array. The second for loop scales the color values to the min and max pressures.

##### Diagram, text Description automatically generatedSet Sensor Menus

This SubVI iterates through all the references and sets all their default values.

#### Wait

This state contains an event structure. The following subsections describe each case.

##### Event Name

This case is called when the user selects an event from the drop-down menu. It simply calls the ‘Find Segment’ state by way of shift register.

##### Upper/Lower Cursor

Whenever the user changes the values of either the upper or lower cursor, this case is called. First, the values for the cursors are added to the data cluster.

A picture containing diagram

Description automatically generatedThe first for loop iterates through all the sensor numbers. It also checks if each sensor is between the upper and lower cursors. If it is not, that index is added to an array. This array is used in the second for loop to disable those sensors in the menu.

##### Previous/Next Button

This case is called when the user clicks either the forward or backward arrows. It simply calls the ‘Find Segment’ state by way of shift register.

##### Segment Select

Whenever the user clicks on the segment select control (the display for the HRM data), tis case is called. If a left click is detected, the ‘wait’ state is called. If a right click is detected, the Reset Min-Max Pressure SubVI is called. The new min and max pressures are then used to [reset the colormap](#_Colormap_to_Array) on the HRM display. Then the ‘wait’ state is called.

###### Reset Min-Max Pressure

This SubVI is simply a while loop that waits until the user clicks okay. Whatever values the user sets the min and max pressure to are wired out.

##### Listbox (1-4)

This case is called when the user selects a sensor in any of the boxes. It simply calls the ‘Find Segment’ state by way of shift register.

##### Key Up

This case is called whenever the user inputs something from the keyboard. If that something is the escape button, then the ‘exit’ state is called. If any other button is pressed, the ‘wait’ state is called.

##### Home Button

When the home button is pressed, the ‘exit’ state is called.

#### Find Segment

This state finds the data for the user-selected event and displays it on the front panel. It also updates all the other front panel controls. In retrospect, I should have renamed it ‘Update UI’ or something like that. Anyway, it uses a couple different SubVIs to update all the different plots.

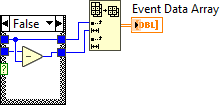
##### Find Event Data

Diagram

Description automatically generatedThe primary purpose of this SubVI is to locate the data for a specified event. This is done by indexing the start and end times arrays. This finds the actual time at which the specified event starts and stops.

The times array is then searched, looking for the times from the start and stop arrays. The indices for those times are used to update the 2D array of event data. If either the start or stop times cannot be found (i.e., the search 1D function yeilds a -1), the default time index difference used in the array subset function will be 800.

Diagram

Description automatically generatedThe nested case structures only matter if the user pressed either the forward or backward arrows. Each case adds or subtracts one second from the start and stop times. These are added to the data cluster so the times will remain altered until the program restarts.

Diagram

Description automatically generatedThis SubVI also updates the upper and lower cursors on the HRM plot. They are negated because the plot is inverted on the y axis.

##### Diagram Description automatically generatedBuild Plots

This SubVI takes the 2D array of doubles that contains the data for the user-selected event. It also iterates through the values in whichever listbox (where the sensor numbers are selected) is wired in.

Qr code

Description automatically generatedFor each selected sensor, the array corresponding to the is formatted into dynamic data. The sensor name (e.g., ‘Sensor 3’) is added to the dynamic data for that sensor. All the data is merged into a single wire and output as ‘Sensor Plots’.

All the sensor names are iterated through as well. If a number can be found in the string (the scan from string does not throw an error), the string will be added to a 1D array of strings. These are wired out as ‘Labels’.

##### Mark Sensor Plots

Diagram

Description automatically generatedThis SubVI adds the vertical dotted lines (cursors) that mark the start and end of a segment on the HRM plot. The start and end times need to be added to the segment start time to that they actually appear.

##### Smooth Data

Diagram

Description automatically generatedThis SubVI resamples the event data so that it contains more data points between the existing points. This is also called interpolation. This makes it so the HRM plot looks visibly smoother.

### Batch

When the **Batch SPARC** button is clicked, a SubVI of the same name is opened. This SubVI is just a bunch of nested for loops. The main loop iterates through all the different settings. If the user chooses to only run SPARC on the current settings, this loop will only run once.

If the user chooses to run through permutations, the Permutations of Settings VI is used to create an array of strings containing all the variations of SPARC settings that will be used during analysis.

Qr code

Description automatically generated

#### Permutations of Settings SubVI

Diagram

Description automatically generatedThe user input expressVI is used to prompt the user to enter ranges for the threshold and cutoff settings. The upper for loop creates an array of doubles containing values between the minimum and maximum cutoff in increments of 0.5. The lower for loop creates an array for the thresholds in increments of 0.01. The final nested for loops create the array of strings with all the different permutations. Currently, the pad level is set to only be 4 (third setting).

#### Main For Loop

This loop runs for each permutation of settings. It first uses the Recursive File List VI to find all the text files in the user-selected folder. It will exclude any files found in the Permutations and Batch Analysis folders since no segments will be found there. From there, an array of file paths is wired into the second for loop.

##### Extract Data SubVI

After the second and third loops finish running it calls the Extract Data SubVI. This finds all the exam files of a selected folder. It then extracts all the SPARC or NMD data in each file. These data are averaged and saved to text files.

##### For Loop to Move Files

This loop (below) takes the analysis files to their own folder. If that folder does not already exist, it will be created.

#### Diagram Description automatically generated with medium confidenceSecond For Loop

This loop iterates through all the text files found in the user-selected folder. Using the Match Regular Expression function, it checks if any of the text files are found in a folder named “Segments”. If they are, the program will move into the next for loop.

##### Extract Previous Sensor Locations SubVI

A picture containing graphical user interface

Description automatically generated

This SubVI takes the folder path from the first for loop. The first for loop found here iterates through the files in this folder path until it finds the file that has ‘Smoothness’ in its name (there should only be one). The text of that file is then extracted into a string array. The second for loop searches the first column of the string array to find the index where the sensor labels start. The final for loop searches through the columns of that row to find the previously used sensor numbers.

#### Third For Loop

This loop iterated through the files in the ‘Segments’ folder that was found in the second for loop. Everything that takes place in this loop is similar to what happens during a single analysis. It then adds all data to the end of a batch analysis file.

### Settings

This state opens the SPARC settings VI. Here, the [Get Spreadsheet String VI](#_Get_Spreadsheet_String) is used to find and display the current settings. The while loop waits until the user clicks the OK button. When that happens, the settings shown on the front panel are saved to the SPARC Settings text file. This replaces the one that was read at the beginning of the VI.

Diagram

Description automatically generated

### SPARC

Clicking the **Run** **SPARC** button simply calls the Run SPARC VI.

### NMD

Clicking the **Run** **NMD** button simply calls the NMD Analysis VI.

### Figures

Clicking the **Create Figures** button opens a SubVI of the same name. This SubVI has its own user interface that is made up of four buttons. Each button click is handled by an event structure. The **Home** button stops the while loop so the user can return to the main menu. The other three buttons open different SubVIs to create their respective figures.

### Extract

Clicking the **Compile Data Report** button.

### Help

The only thing that this case calls is the Help Window VI. All that VI contains is a while loop with an Okay button. It just waits until the user is done looking at the help window.